

Mémoire de Projet de Fin d'études

FILIÈRE
Ingénieur des Sciences de Données -
Data Engineer

SUJET

MISE EN PLACE D'INFRASTRUCTURES POUR
LA COLLECTE, LE STOCKAGE ET L'ANALYSE
DES DONNÉES CHEFCLUB

Réalisé par :

LABRIJI Saad

Jury :

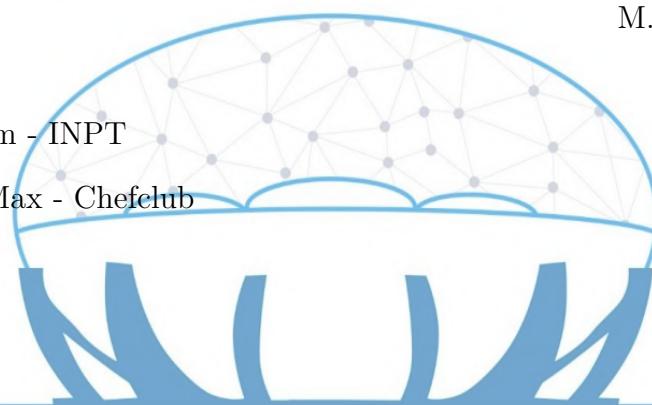
M. ET TOLBA Mohamed

M. FEKRI Mohammed

Encadré par :

Mme. EL ASRI Ikram - INPT

M. PERDRIGEAT Max - Chefclub



AGENCE NATIONALE DE RÉGLEMENTATION DES TÉLÉCOMMUNICATIONS

INSTITUT NATIONAL DES POSTES ET TÉLÉCOMMUNICATIONS

ANNÉE UNIVERSITAIRE 2022-2023

“

À mes chers parents, qui m'ont tout donné et qui trouvent toujours un moyen de donner plus. Mon cœur déborde d'une reconnaissance éternelle à votre égard.

À ma chère sœur Ibtissam et son mari Nezar, à qui je serai éternellement reconnaissant de les avoir.

A toutes mes enseignantes et à tous mes enseignants.

À mes camarades de l'INPT, grâce à qui j'ai appris des leçons impayables et vécu des souvenirs inoubliables.

À tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail,

Merci.

”

LABRIJI Saad

Remerciements

Au terme de mon Projet de Fin d'Études, je tiens à exprimer ma profonde gratitude envers toutes les personnes dont l'intervention tout au long de ce projet a grandement contribué à son accomplissement.

Je saisirai cette occasion pour adresser mes sincères remerciements à **M. Perdigeat Max**, Responsable Data et Business Analyst au sein de Chefclub, qui m'a encadré et m'a transmis de nombreuses compétences techniques et professionnelles. Grâce à lui, j'ai pu acquérir une précieuse expérience professionnelle à l'international.

Je tiens également à exprimer ma reconnaissance à ma Professeure, **Mme. EL ASRI Ikram**, pour ses enseignements extrêmement utiles, que j'ai pu mettre en pratique quotidiennement à maintes reprises tout au long de mon parcours académique à l'INPT Rabat. Son encadrement, son soutien et ses conseils ont été d'une valeur inestimable tout au long de mon séjour à l'INPT.

J'adresse également mes remerciements chaleureux à toute l'équipe de **Chefclub**. Votre accueil chaleureux, l'ambiance conviviale et votre soutien constant tout au long de notre stage ont contribué à rendre mon expérience inoubliable.

Je souhaite également exprimer notre gratitude envers l'ensemble de l'équipe pédagogique de l'Institut National de Postes et Télécommunications (INPT) ainsi que les intervenants responsables de la formation en ingénierie des Sciences de données, pour avoir assuré une formation de haute qualité.

Que tous ceux qui ont contribué de près ou de loin à l'aboutissement de ce projet trouvent l'expression de nos remerciements les plus sincères.

Résumé

Ce rapport présente mon travail réalisé chez Chefclub (organisme d'accueil) dans le cadre de mon stage de fin d'études en tant que Data Engineer et Analyst. Chefclub est une marque culinaire digitale renommée née à Paris, et qui propose du contenu divertissant et informatif sur les réseaux sociaux depuis 2016, avec un rayonnement en Europe, aux États-Unis.

L'objectif principal de mon stage était d'améliorer l'infrastructure de données de Chefclub afin d'optimiser la collecte, le stockage et l'analyse des données générées sur les réseaux sociaux pour soutenir les objectifs business de l'entreprise. J'ai travaillé avec divers outils de Data Engineering, de Cloud Computing, de Data Science et de Data Analytics.

Mes principales réalisations ont inclus la mise en place d'une architecture permettant la collecte périodique et le stockage des vastes quantités de données générées par les pages de Chefclub sur les réseaux sociaux. J'ai également développé un processus automatisé en utilisant des technologies de Data Engineering et de cloud computing qui permettait de récupérer de façon journalière les données analytiques des différentes chaînes YouTube de Chefclub.

Par ailleurs, j'ai effectué des analyses statistiques afin d'identifier des tendances et des modèles dans les données des pages Facebook de Chefclub, ce qui a permis de fournir des insights.

Enfin, j'ai eu l'opportunité de développer un modèle d'apprentissage automatique (machine learning) visant à prédire les performances d'un post sur les réseaux sociaux en utilisant nos données historiques.

Durant ce stage, nous avons utilisé la méthode SCRUM pour la gestion du projet, favorisant ainsi la flexibilité et l'efficacité de notre travail.

Mots clés : GCP, Apache Airflow, Kubernetes, Docker, API, Python, ETL, Machine Learning, insights, Cloud Computing, Data Engineering, Data Science, Data Analytics

Abstract

This report presents my work carried out at Chefclub (host organization) as part of my end-of-studies internship as a Data Engineer and Analyst. Chefclub is a renowned digital culinary brand born in Paris, which has been providing entertaining and informative content on social media since 2016, with a presence in Europe and the United States.

The main objective of my internship was to improve Chefclub's data infrastructure in order to optimize the collection, storage, and analysis of data generated on social networks to support the company's business goals. I worked with various Data Engineering, Cloud Computing, Data Science, and Data Analytics tools.

My main achievements included implementing an architecture for the periodic collection and storage of large amounts of data generated by Chefclub's pages on social media. I also developed an automated process using Data Engineering and cloud computing technologies that allowed for daily retrieval of analytical data from Chefclub's different YouTube channels.

Additionally, I performed statistical analyses to identify trends and patterns in Chefclub's Facebook page data, which provided valuable insights.

Finally, I had the opportunity to develop a machine learning model aimed at predicting the performance of a post on social media using our historical data.

During this internship, we used the SCRUM method for project management, which promoted flexibility and efficiency in our work.

Keywords : GCP, Apache Airflow, Kubernetes, Docker, API, Python, ETL, Machine Learning, insights, Cloud Computing, Data Engineering, Data Science, Data Analytics

ملخص

يقدم هذا التقرير عملي الذي قمت به في شركة Chefclub (مؤسسة الاستضافة) كجزء من مشروع التدريب الختامي في دوري كمهندس بيانات ومحلل. Chefclub هي عالمة مشهورة في المجال الطهو الرقمي ولدت في باريس، وتقدم محتوى ممتع ومفيد على وسائل التواصل الاجتماعي منذ عام 2016، وتتمتع بشهرة في أوروبا والولايات المتحدة.

الهدف الرئيسي لتدريبي كان تحسين بنية البيانات في Chefclub لتحسين جمع البيانات وتخزينها وتحليلها على وسائل التواصل الاجتماعي لدعم الأهداف الأعمال في الشركة. عملت مع مجموعة متنوعة من أدوات هندسة البيانات والحوسبة السحابية وعلوم البيانات وتحليل البيانات.

من أبرز إنجازاتي تضمنت إنشاء بنية تمكّن من جمع كميات ضخمة من البيانات المولدة بشكل دوري عن طريق صفحات Chefclub على وسائل التواصل الاجتماعي وتخزينها. كما قمت بتطوير عملية آلية باستخدام تقنيات هندسة البيانات والحوسبة السحابية تسمح بجلب البيانات التحليلية الخاصة بقنوات YouTube المختلفة لـ Chefclub يومياً.

بالإضافة إلى ذلك، أجريت تحليلات إحصائية لتحديد الاتجاهات والأنماط في بيانات صفحات Chefclub على Facebook مما ساهم في توفير رؤى قيمة.

وأخيراً، كانت لدى الفرصة لتطوير نموذج للتعلم الآلي بهدف التنبؤ بأداء منشور على وسائل التواصل الاجتماعي باستخدام البيانات التاريخية لدينا.

خلال هذا التدريب، استخدمنا منهجهية SCRUM لإدارة المشروع، مما ساهم في تعزيز المرونة والكفاءة في عملنا.

الكلمات المفتاحية: ETL، Python، API، Docker، Kubernetes، Airflow، Apache GCP، Tعلم آلي الحوسبة السحابية، علوم البيانات، تحليل البيانات.

Liste des abréviations

DS	<i>Data Science (Science des Données)</i>
ML	<i>Machine Learning (Apprentissage Automatique)</i>
SMPs	<i>Social Media Platforms (Plateformes de médias sociaux)</i>
API	<i>Application Programming Interface</i>
GCP	<i>Google Cloud Platform</i>
GCS	<i>Google Cloud Storage</i>
ETL	<i>Extract, Transform, Load</i>
BQ	<i>BigQuery</i>
DWH	<i>Data Warehouse</i>
K8s	<i>Kubernetes</i>
DAG	<i>Directed Acyclic Graph (Graphe Acyclique Orienté)</i>
NAS	<i>Network Attached Storage</i>

Liste des tableaux

2.1 Advantages and disadvantages of regression metrics	37
--	----

Table des figures

1.1	Logo Chefclub	4
1.2	Chefclub Discovery - Morocco	5
1.3	Chefclub Mission	6
1.4	Chefclub Team - Highlights from 2019	7
1.5	Chefclub Slogant	8
1.6	Chefclub Performance	9
1.7	Chefclub Data Team Agile Scrum Process	14
1.8	Project Planning Workflow	15
1.9	Project Gantt Chart	16
1.10	Google Calendar - Saad LABRIJI's View	19
1.11	Chefclub Timesheet - Data Team View	19
2.1	Data Science Word Cloud Illustration	22
2.2	Synergy of Data Science Domains	23
2.3	Data Engineering - The Lifecycle and Key Keywords	24
2.4	Machine Learning Model Creation Pipeline	25
2.5	Machine Learning Engineering - The Lifecycle and Key Keywords	26
2.6	MLOps : The Intersection of ML, DevOps, and Data Engineering	26
2.7	Illustration of a Python virtual environment (VENV) example	27
2.8	Functioning of a REST API	28
2.9	OAuth Access and Refresh Token Flow Diagram	28
2.10	Illustration of Distributed Version Control System	29
2.11	Orchestration Engine	30
2.12	Scheduling Engine	30
2.13	Example of an ETL Pipeline	31
2.14	Boxplot on a Normal Distribution	32
2.15	Detecting outliers with z-Scores	32
2.16	Comprehensive Overview of Machine Learning Techniques and Algorithms	34
2.17	Correlation Matrix example	34
2.18	Test, training and validation sets	36
2.19	Time Series Regression : Underfitting Vs. Good Fit Vs. Overfitting	37
2.20	GridSearch Cross Validation Approach	38
2.21	The Big Three Cloud Computing Providers	39
2.22	Cloud Computing Architecture	40
2.23	GCP - Logo	40
2.24	Data Science tools on Google Cloud Platform	41
2.25	BigQuery - Logo	41
2.26	Google Cloud Storage - Logo	42
2.27	Logos of Looker Studio and Google Sheets	42
2.28	Docker - Logo	43

Table des figures

2.29	Containers Vs. VMs - Logo	43
2.30	Kubernetes - Logo	44
2.31	Containerized Application Deployment with Docker and Kubernetes	44
2.32	Apache Airflow - Logo	45
2.33	ETL Workflow DAG in Airflow	45
2.34	Github - Logo	46
3.1	AWS S3 - Replication and Logo	52
3.2	Table Inventory from PostgreSQL Hosted on CloudSQL	52
3.3	Airbyte - Logo	53
3.4	Workflow and Fetch Analytics data Task	53
3.5	Data Preparation in BigQuery Using DDL with Airflow	54
3.6	Example of Upsert Queries in BigQuery planified with Airflow	54
3.7	Data Transformation Flow : Transforming and Uploading Data from GCS to CloudSQL	55
3.8	Chefclub Looker Dashboard Example	56
3.9	Principaux critères de sélection des technologies et des outils pour l'infrastructure de données de Chefclub	57
3.10	Overview of Chefclub's Data Infrastructure (Post-Migration) - June 2023	58
3.11	YouTube Analytics Data Retrieval Solution	59
3.12	Data Analysis Workflow for Facebook Posts	62
3.13	Model Training/Model Deployment Solution	64
3.14	Sequence Diagram : Model Inference via Slack	64
4.1	YouTube Analytics Data Retrieval Solution (Reminder)	69
4.2	Folder Architecture of youtube-fetch in VENV Environment	70
4.3	Configuration File - YT Analytics Reports	72
4.4	Example AVRO File Containing API Response	74
4.5	Organization of Analytics Files on GCS Bucket	74
4.6	Python Code testing in virtual environment (VENV)	75
4.7	Used Dockerfile to build the YouTube Analytics Code Image	75
4.8	Make command to build the Docker image	76
4.9	Make command to push the image to Google Artifact Registry	76
4.10	YouTube Analytics image in the Google Artifact Registry	76
4.11	Make commands for image build , container run , and image push	77
4.12	Docker Compose file for local container build	77
4.13	Testing the custom Python Code inside the container	78
4.14	YT Job Code Documentation	78
4.15	Example of Performance Channel Table in BigQuery	79
4.16	Secrets YouTube dans Google Secret Manager	80
4.17	Command to Update Secrets in Google Secret Manager with Make	80
4.18	Exemple de fichier YAML pour les secrets YouTube	80
4.19	BigQuery Temp Tables used for Upsert	84
4.20	BigQuery Native Production Table for YouTube Analytics	85
4.21	Airflow User Interface - YouTube time based DAG	85
4.22	Task Graph of the monitized DAG in Airflow	86
4.23	Airflow Error Notification on Slack	86
4.24	Execution Grid on Airflow - YouTube Job	87
4.25	Task duration Graph on Airflow - YouTube Job	87

4.26 YouTube analytics query for performing table joins in BigQuery	88
4.27 Financial Report Sample	88
4.28 Looker Studio UI - YouTube Analytics interactive Report	89
4.29 Example of YT Studio Interface	89
4.30 Data Analysis Workflow for Facebook Posts (Reminder)	90
4.31 Flow diagram showing the relationships between Posts and Videos	91
4.32 Example of posting a (Tested or Untested) post.	91
4.33 Example of a Tested-Post and Video submitted to Meta's A/B testing	92
4.34 Overview of raw data in the CSV file	93
4.35 Overview of raw data in a <i>Pandas DataFrame</i>	93
4.36 Data Preprocessing in Jupyter Notebook	94
4.37 Splitting the dataframe by page in Jupyter Notebook	96
4.38 Dataset extraction by filtering the original dataset	97
4.39 Views per day per post nature	97
4.40 Mean , sum , and count per week	98
4.41 Lineplot -Total Views per Day by Post Nature	98
4.42 Histplot - Post Count by Week by Post Nature	99
4.43 Histplot - Total Views per Week by Post Nature	99
4.44 Histplot - Average Views per Week by Post Nature	100
4.45 Histplot - Median Views per Week by Post Nature	100
4.46 Scatter plot - Post Count and Weekly Estimated Earnings	101
4.47 Optimal Post Count Strategy for Maximizing Weekly Estimated Earnings	102
4.48 Solution d'Entraînement/Déploiement du Modèle (Rappel)	104
4.49 Sequence Diagram : Model Inference via Slack	104
4.50 Example of Model Training with CLI	105
4.51 Example of a performance table for Facebook posts in BigQuery	105
4.52 Loading the BigQuery table into a <i>DataFrame</i>	106
4.53 Z-Score Outlier Detection Chart	107
4.54 Flaging and Deleting the Outliers with Z-Score	107
4.55 The process of adding historical features to the Pandas DataFrame	108
4.56 Adding historical features to the Pandas DataFrame	108
4.57 Adding temporal features to the Pandas DataFrame	108
4.58 Analysis of the Target Variable Distribution	109
4.59 Correction of the Target Variable Distribution	109
4.60 Feature Selection with the Correlation Matrix	110
4.61 Models Used for Model Training	110
4.62 Lasso Model Performance on test data	111
4.63 Lasso Model : Y_test (actual) Vs. Y_pred (predicted)	111
4.64 Optimizing Lasso Hyperparameters with GridSearchCV	113
4.65 Production Use of <code>upload_or_replace_blob()</code> Function	114
4.66 Folder Architecture of Model Training in VENV Environment	115
4.67 Model Training Evaluation in VENV Environment	115
4.68 Updated Pickle Model in Google Cloud Storage	115
4.69 Folder Architecture of Model Deployment in VENV Environment	116
4.70 Local execution of the <code>predict()</code> function using the <code>functions_framework</code>	117
4.71 Example of local model inference	117
4.72 Deployment of the Cloud Function using the Make command	118
4.73 Cloud Function deployent on GCP	118

4.74 Example of model inference using the deployed <code>predict()</code> function on GCP	118
4.75 Slack API Slash Command Configuration	119
4.76 Model inference request from Slack	119
4.77 Response from the Cloud Function	120
4.78 Slack - Message formatter for prediction bot	120
4.79 Dockerfile used to build the Model Training image	121
4.80 Model Training Docker image in Docker UI	121
4.81 Make command to push Docker image to Artifact Registry	121
4.82 Model Training image in Google Artifact Registry	122
4.83 Testing Model Training Python Code with Docker	122
4.84 Current Facebook DAG <u>before</u> adding the Model Training Task	123
4.85 Updated Facebook DAG <u>after</u> adding the Model Training Task	123
4.86 Running the Model Training Task on Airflow	124
4.87 Updated Model on Google Cloud Storage	125
A.1 Data Refinement and Labeling for Facebook Post Performance Analysis	132

Table des matières

Remerciements	III
Résumé	IV
Abstract	V
VI	ملخص
Introduction générale	1
Chapitre 1	3
1 Contexte général du projet	3
1.1 Introduction	4
1.2 Organisme d'accueil	4
1.2.1 Chefclub	4
1.2.2 Services	5
1.2.3 Valeurs de l'entreprise	6
1.2.4 Histoire de Chefclub	7
1.2.5 Chiffres clés de Chefclub	8
1.2.6 Reconnaissances et réalisations	9
1.3 Cadre général du Projet	10
1.3.1 Problématique	10
1.3.2 Objectifs du projet	11
1.3.3 Périmètre du projet	11
1.3.4 Impact du projet	12
1.4 Conduite de projet	13
1.4.1 Processus du développement	13
1.4.2 Planification du projet	15
1.5 Conclusion	20
Chapitre 2	21
2 Etat de l'art : Contexte et Technologies Associées	21
2.1 Introduction	22
2.2 Sciences des Données	22
2.2.1 Disciplines des Sciences des Données	23
2.2.2 Techniques et Concepts Clés	26
2.3 Outils Clés	39
2.3.1 Le Cloud Computing	39
2.3.2 Docker	43

2.3.3	Kubernetes	44
2.3.4	Apache Airflow	44
2.3.5	GitHub	46
2.4	Conclusion	47
Chapitre 3	48
3 Analyse et conception	48
3.1	Introduction	49
3.2	Analyse des besoins	49
3.2.1	Besoins opérationnels	49
3.2.2	Besoins de l'équipe Data	49
3.2.3	Besoins non fonctionnels	50
3.2.4	Les acteurs du système	51
3.3	Infrastructure Data de Chefclub	51
3.3.1	Stockage des données	51
3.3.2	Collecte et ingestion des données	53
3.3.3	Traitement et Préparation des données	54
3.3.4	Analyse et visualisation des données	55
3.3.5	Migration de Airbyte et Airflow vers GCP	56
3.4	Conception de la solution	59
3.4.1	Conception, Workflow et Présentation de la Solution	59
3.5	Conclusion	67
Chapitre 4	68
4 Implémentation et validation de la solution	68
4.1	Introduction	69
4.2	Développement et Test de la solution Data Engineering / DataOps	69
4.2.1	Développement et Test du Code Python qui récupère les données de l'API . .	69
4.2.2	Développement, Test et Orchestration du DAG Airflow	75
4.3	Analyse des performances des Posts sur Facebook	90
4.3.1	Contexte, Terminologie et Illustration	90
4.3.2	Développement et Analyse des données	92
4.3.3	Recommandations et Conclusion	102
4.3.4	Développement et Test du Code Python pour l'Entraînement du Modèle . .	104
4.3.5	Développement et Test de l'environnement pour la Serverless Inference . .	116
4.3.6	Développement, Test et Orchestration de la Task <i>Model Train</i> dans le DAG Facebook	120
4.4	Conclusion	125
Conclusion et perspectives	126
Bibliographie	128
Annexes	129
A Codes	130

Introduction générale

Ce rapport de stage décrit mon expérience chez Chefclub, où j'ai participé à un projet visant à améliorer l'infrastructure data de l'entreprise. Mon objectif principal était de renforcer les processus de récupération quotidienne de données, d'analyser ces données afin de soutenir les décisions basées sur les données prises par les autres équipes, et de développer un modèle d'apprentissage automatique prédictif basé sur les données historiques pour évaluer les performances futures des publications.

Chefclub est une entreprise renommée dans le domaine culinaire, connue pour son contenu divertissant et informatif sur les réseaux sociaux. Avec un rayonnement en Europe et aux États-Unis, Chefclub génère une quantité considérable de données à partir de ses pages sur les réseaux sociaux. Ce stage m'a offert l'opportunité de contribuer à l'amélioration de l'infrastructure de données existante, afin de collecter, stocker et analyser ces données de manière optimale, pour permettre aux différentes équipes de l'entreprise de prendre des décisions éclairées basées sur les données.

L'optimisation de l'infrastructure de données revêt une importance capitale dans le contexte actuel, où les entreprises cherchent à exploiter pleinement le potentiel des données pour obtenir un avantage concurrentiel. En centralisant et en organisant les données de manière efficace, Chefclub vise à créer un environnement propice à l'analyse approfondie, à la génération d'informations pertinentes et à la prise de décisions basées sur les données.

Pendant mon stage, j'ai utilisé différents outils de Data Engineering, Cloud Computing, Data Science et Data Analytics pour établir une infrastructure solide et évolutive. Parmi mes réalisations clés, j'ai mis en place des processus automatisés de collecte de données et développé un modèle d'apprentissage automatique pour prédire les performances futures.

Ce rapport mettra en évidence les réalisations spécifiques que j'ai accomplies dans le cadre du projet d'amélioration de l'infrastructure de données de Chefclub. Il décrira également les technologies et les méthodes que j'ai utilisées pour atteindre ces améliorations, ainsi que les résultats obtenus.

Introduction générale

Ce rapport, est organisé en quatre chapitres :

- Le premier chapitre aborde le contexte général du projet et explique la motivation et la problématique du projet. Il présente également l'objectif du projet ainsi que la planification et le processus de développement qui ont été adoptés.
- Le deuxième chapitre traite de l'état de l'art et présente différentes technologies utilisées dans le projet.
- Le troisième chapitre présente l'analyse et la conception de la solution pour Chefclub, en abordant les besoins en Science des Données, l'infrastructure data existante, les problématiques identifiées et les solutions proposées.
- Enfin, le quatrième chapitre se concentre sur le développement et les tests de la solution. Il couvre les différentes phases de développement du projet, en mettant l'accent sur les tests réalisés. Le rapport se termine par une conclusion et des perspectives sur l'avenir du projet.

Chapitre 1

Contexte général du projet

1.1 Introduction

Dans ce premier chapitre, je aborderai le contexte général du projet en me concentrant sur Chefclub, l'organisme d'accueil. Je montrerai divers aspects de Chefclub, tels que son identité, ses valeurs, son historique et ses performances clés. En outre, je définirai la problématique que je cherche à résoudre grâce à ce projet, ainsi que les objectifs que j'ai fixés. J'examinerai également le périmètre du projet et l'impact attendu une fois que celui-ci sera réalisé. Enfin, je fournirai un aperçu de la conduite du projet en décrivant le processus de développement, la planification établie et le diagramme de Gantt correspondant à ce projet.

1.2 Organisme d'accueil

Cette section donne une vue générale sur Chefclub en explicitant son historique, sa mission, ses chiffres clés et son organigramme, et enfin un cadrage général du projet.

1.2.1 Chefclub

Chefclub est une marque digitale de cuisine qui a été créée à Paris en 2016. Depuis ses débuts, elle a rapidement étendu son influence à travers l'Europe, les États-Unis, l'Amérique latine et l'Asie grâce à sa distribution gratuite de vidéos de recettes. Chefclub occupe une position unique à l'intersection de la cuisine et du divertissement, captivant chaque mois un public mondial de plus d'1 milliard de vues organiques sur des plateformes telles que Facebook, Instagram, Snapchat, YouTube, TikTok, Pinterest, ainsi que sur son site Internet et son application mobile. La communauté de Chefclub compte actuellement 80 millions de fidèles abonnés.

Le positionnement distinctif de Chefclub a transcendé les frontières des simples passionnés de cuisine. Dans le but d'engager un public plus large dans l'univers culinaire, Chefclub a développé un univers immersif et pédagogique spécialement conçu pour les enfants, où des personnages attachants prennent vie. Chaque mois, près de 300 millions de téléspectateurs se réunissent pour regarder les émissions culinaires de Chefclub diffusées sur les réseaux sociaux et les télévisions connectées, principalement en Europe et aux États-Unis.

En tant que spécialiste des vidéos de recettes, Chefclub produit et distribue 15 concepts innovants couvrant une variété de thématiques. Parmi ces concepts, on retrouve des émissions dédiées à la cuisine végétarienne, à l'upcycling culinaire et à la cuisine en pleine nature, pour ne citer que quelques exemples.¹ [site-Chefclub]



FIG. 1.1 : Logo Chefclub

¹ <https://www.chefclub.tv/>.

1.2.1.1 Chefclub Découverte - La cuisine marocaine

Chefclub a réalisé une série de vidéos absolument captivantes pour révéler au monde entier les trésors de la cuisine Marocaine. Ils ont plongé au cœur de notre culture culinaire et dévoilé avec passion les secrets qui font de notre cuisine une véritable merveille.² [chefclub-discovery-morocco]

La figure suivante illustre quelques moments de Chefclub Discovery - Maroc :

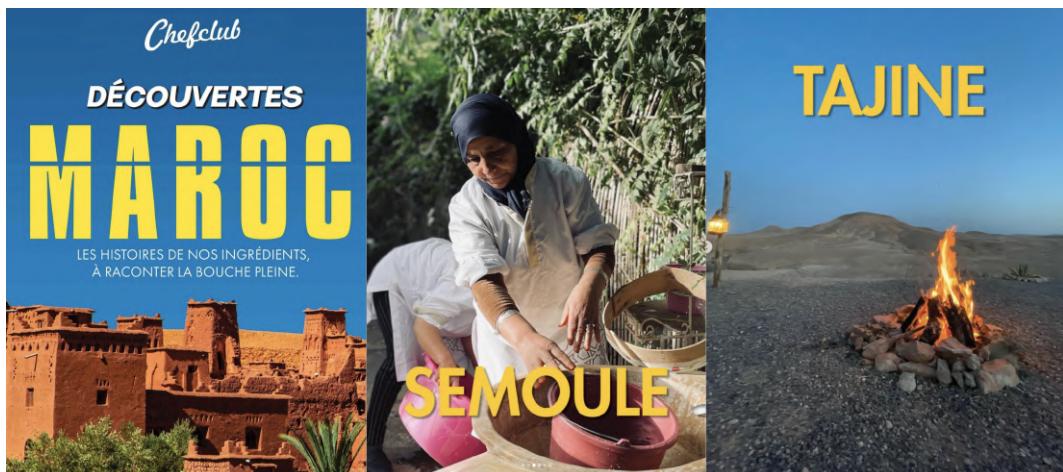


FIG. 1.2 : Chefclub Discovery - Morocco

1.2.2 Services

Chefclub offre une large gamme de services culinaires pour répondre aux besoins et aux intérêts des amateurs de cuisine. Voici les services qu'ils proposent :

- **Livres de recettes** : Chefclub publie des livres de recettes qui regorgent d'idées créatives pour préparer des plats délicieux et surprenants.³ [Chefclub Shop]
- **Vidéos en ligne** : Chefclub propose une vaste bibliothèque de vidéos en ligne, accessibles sur leur site web et leurs plateformes de médias sociaux. Ces vidéos présentent des recettes étape par étape, mettant en valeur les astuces et les techniques de cuisine.⁴ [Main YouTube Channel] ⁵ [Main Facebook Page]
- **Tutoriels** : Chefclub offre des tutoriels détaillés pour guider les cuisiniers de tous niveaux. Que ce soit pour apprendre des techniques de base ou pour réaliser des recettes plus avancées, les tutoriels sont conçus pour aider les amateurs de cuisine à améliorer leurs compétences.
- **Application mobile** : Chefclub propose une application mobile conviviale qui permet aux utilisateurs d'accéder à des recettes, des vidéos, des astuces et des fonctionnalités interactives, le tout depuis leur appareil mobile.⁶ [Chefclub Android Mobile App]

² www.chefclub.tv/fr/s/maroc-discovery/.

³ boutique.chefclub.tv.

⁴ www.youtube.com/channel/UCQ_FrN6FNViZVjxHUVf6_xA.

⁵ www.facebook.com/Chefclub.tv.

⁶ play.google.com/store/apps/details?id=com.chefclub.App&hl=fr&gl=US.

- **Ateliers de cuisine** : Chefclub organise des ateliers de cuisine dans différentes villes, offrant aux participants l'opportunité d'apprendre des techniques culinaires auprès de chefs expérimentés. Ces ateliers permettent également aux participants de cuisiner et de déguster des plats ensemble.
- **Événements** : Chefclub organise régulièrement des événements spéciaux, tels que des démonstrations culinaires en direct, des dégustations de recettes exclusives et des rencontres avec l'équipe de Chefclub. Ces événements offrent aux fans et aux clients de Chefclub la possibilité de se connecter avec la communauté et de découvrir de nouvelles inspirations culinaires.

1.2.3 Valeurs de l'entreprise

L'une des pierres angulaires de Chefclub réside dans ses valeurs d'entreprise, qui guident leurs actions au quotidien. Ces valeurs sont fondamentales pour la réussite de l'entreprise et pour la création d'une expérience culinaire unique et engageante pour leur communauté. Voici en détail les valeurs et la mission de l'entreprise :

- **Leur Mission** : *Réunir tout le monde en cuisine* : Chefclub s'engage à rassembler les personnes autour de la cuisine. Leur mission est de créer des contenus inspirants et accessibles qui encouragent les gens à partager des moments conviviaux et créatifs dans leur cuisine. En réunissant tout le monde en cuisine, Chefclub souhaite stimuler la créativité culinaire et apporter de la joie et du plaisir à leurs utilisateurs.



FIG. 1.3 : Chefclub Mission

- **Performance** : La performance est une valeur centrale chez Chefclub. Avec 100 millions de followers, 1 milliard de vues par mois et 700 000 livres vendus, leur succès n'est pas le fruit du hasard ou de la chance. Il repose sur le travail acharné de leurs collaborateurs, qui mettent tous les jours leurs efforts et leur expertise au service de la réussite de la marque. Chez Chefclub, la performance est synonyme d'excellence et de constante recherche d'amélioration.
- **Humilité** : Chez Chefclub, ils croient en l'importance de l'humilité dans leur approche. Ils prennent des décisions en écoutant attentivement leur communauté et en s'appuyant sur une analyse approfondie des données. L'humilité signifie également valoriser l'expertise de chacun, co-construire les projets, accepter le compromis, remettre en question leurs propres idées, faire confiance aux autres et à eux-mêmes, et avancer ensemble dans la même direction.

- **Soin** : Ils accordent une importance capitale au soin dans tout ce qu'ils font. L'obsession du détail, la précision, la minutie et le sens du beau caractérisent leur approche. Ce souci extrême qu'ils portent à leur travail est le reflet du soin qu'ils ont pour leur communauté. Leur objectif principal est d'offrir à leurs utilisateurs une expérience d'une qualité exceptionnelle, afin de leur faire vivre et ressentir des émotions positives à travers leurs contenus culinaires.

Ces valeurs de **Chefclub**, leur mission inspirante, leur quête de performance, leur humilité et leur soin, façonnent la culture de l'entreprise et sont les fondations sur lesquelles repose leur engagement envers leur communauté et leur succès continu.

1.2.4 Histoire de Chefclub

L'histoire de Chefclub commence en février 2016, lorsque les trois frères fondateurs, Thomas, Axel et Jonathan, décident de réenchanter le placard et de s'amuser avec des ingrédients ordinaires pour créer des recettes extraordinaires. Ils transforment la cuisine de leur appartement familial en le tout premier studio Chefclub. Jonathan a l'idée de se filmer en train de préparer un simple croque-monsieur, et la vidéo devient instantanément virale avec des centaines de milliers de vues.

Encouragés par ce succès, ils continuent à proposer des vidéos de cuisine divertissantes qui conquièrent rapidement un large public. En avril 2016, Chefclub définit sa raison d'être : réenchanter le placard et permettre à chacun de prendre du plaisir aux fourneaux. Les vidéos deviennent de plus en plus spectaculaires et Chefclub gagne rapidement en popularité dans le monde entier.

En février 2017, Chefclub compte déjà 10 millions d'abonnés et l'équipe doit annexer le salon de l'appartement pour faire face à son expansion. En avril 2017, l'équipe compte déjà 15 membres. Au fil des années, Chefclub continue de croître et en février 2018, ils lancent Chefclub Light Fun, proposant des recettes gourmandes avec une approche diététique. En mai 2018, l'équipe déménage dans de nouveaux locaux rénovés pour accompagner leur ambition.

Chefclub élargit son influence en lançant de nouveaux livres, des coffrets thématiques et des produits dérivés. En novembre 2021, Chefclub compte plus de 100 membres dans son équipe et dépasse les 100 millions de followers sur les réseaux sociaux. Leurs vidéos ont généré plus de 2 milliards de vues, et ils proposent plus de 10 000 recettes sur leur site.



FIG. 1.4 : Chefclub Team - Highlights from 2019

En mars 2021, Chefclub étend son influence aux écoles en proposant des kits créatifs pour éveiller et émerveiller les élèves. Ils lancent également leur propre application mobile, offrant aux utilisateurs la possibilité de retrouver leurs recettes préférées, de participer à des challenges amusants et d'échanger avec la communauté.

En mai 2022, Chefclub publie le livre Super-Légumes, mettant en avant les pouvoirs cachés des légumes à travers des recettes saisonnières. L'équipe travaille sans relâche pour répondre aux attentes de leur communauté grandissante et continue d'innover.

Chefclub prévoit de lancer une série de masterclasses en ligne en collaboration avec des chefs renommés et des experts culinaires. Ils se lancent également dans l'univers de la télévision avec une émission de cuisine captivante en préparation. En parallèle, Chefclub s'engage dans une démarche écoresponsable en collaborant avec des producteurs locaux et en encourageant l'utilisation d'ingrédients bio.

Avec leur succès croissant, Chefclub ne perd pas de vue sa mission originelle de réenchanter la cuisine et de rendre le plaisir de cuisiner accessible à tous. Ils continuent de créer une véritable communauté de passionnés de cuisine et de partager leur savoir et leur passion à travers leurs vidéos, leurs livres et leurs produits. Pour plus de détails sur l'histoire de la croissance de Chefclub, vous pouvez visiter leur site web⁷ [Chefclub-Story]



FIG. 1.5 : Chefclub Slogant

1.2.5 Chiffres clés de Chefclub

Chefclub, bénéficie d'une impressionnante portée mondiale avec des chiffres qui témoignent de son succès. Chaque mois, Chefclub enregistre environ 2,5 milliards de vues à travers le monde, démontrant l'engouement international pour leurs contenus.

En France, Chefclub attire également une audience considérable, avec pas moins de 390 millions de vues mensuelles. Ce nombre témoigne de la popularité de l'organisme dans le pays, captivant les spectateurs français par leurs créations culinaires innovantes.

Outre les frontières françaises, Chefclub continue d'étendre son influence aux États-Unis, où il génère environ 1 milliard de vues chaque mois. Ces chiffres impressionnantes mettent en évidence

⁷ www.chefclub.tv/fr/story/.

l'attrait croissant des Américains pour les recettes uniques et divertissantes proposées par Chefclub.

De plus, Chefclub peut se targuer d'avoir une base de fans solide, comptant pas moins de 150 millions de followers à travers le monde. Ces adeptes fidèles témoignent de l'impact durable que l'organisme a eu sur les amateurs de cuisine et de divertissement.

En termes de production, Chefclub ne se contente pas d'un seul programme, mais propose une diversité de contenu. Avec pas moins de 15 programmes à son actif, l'organisme offre une variété d'expériences culinaires pour satisfaire tous les goûts et toutes les envies.

Ces chiffres remarquables attestent du rayonnement mondial de Chefclub et de sa capacité à engager et inspirer un large public à travers ses vidéos créatives et captivantes.

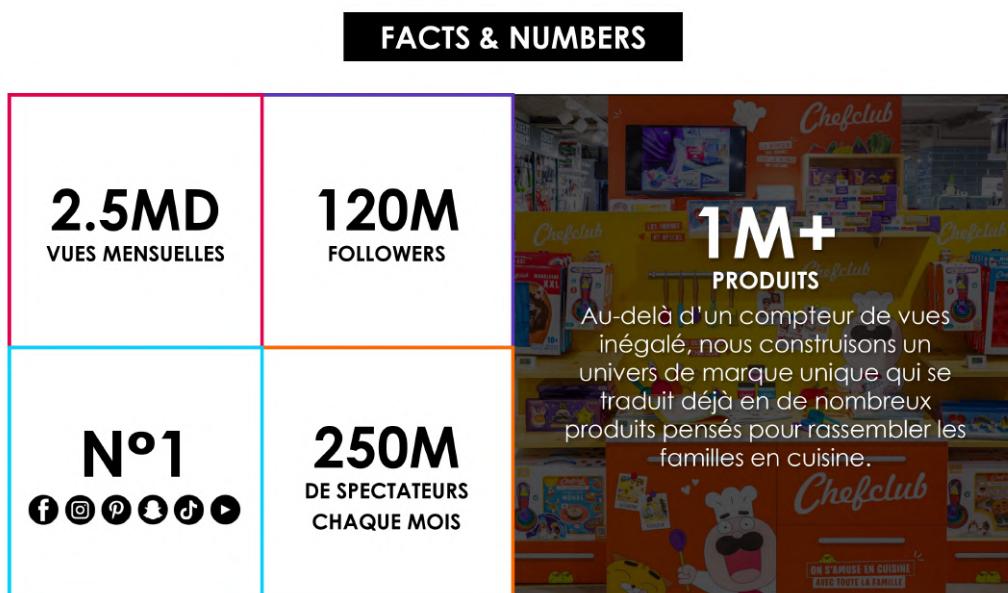


FIG. 1.6 : Chefclub Performance

1.2.6 Reconnaissances et réalisations

Chefclub a été récompensé à plusieurs reprises pour son travail remarquable dans le domaine culinaire. Voici un article qui détaille davantage leurs réalisations⁸ [réalisations-Chefclub]

Voici quelques-unes de leurs distinctions les plus notables :

- **Grand Prix Stratégies du Brand Content (2018)** : Chefclub a remporté ce prestigieux prix qui récompense les meilleures stratégies de contenu de marque. Cette reconnaissance souligne l'approche créative et innovante de Chefclub pour engager son audience à travers ses recettes et ses vidéos.
- **Tech For Good Awards - Meilleure startup de l'année (2019)** : Chefclub a été honoré en tant que meilleure startup de l'année lors des Tech For Good Awards. Ce prix met en évidence l'impact positif de Chefclub dans le domaine culinaire en utilisant la technologie de manière responsable et durable.

⁸ www.frenchweb.fr/chefclub/.

- **Entreprise innovante - BFM Business (2019)** : Chefclub a été sélectionné comme l'une des entreprises les plus innovantes de France par BFM Business. Cette reconnaissance témoigne de l'approche novatrice de Chefclub pour inspirer et accompagner les amateurs de cuisine, en repoussant les limites de la créativité culinaire.

Ces distinctions témoignent du succès et de l'impact de Chefclub dans l'industrie culinaire. En continuant à innover et à offrir des services de qualité, Chefclub s'est positionné comme un acteur majeur de la scène gastronomique, tout en étant reconnu pour son engagement envers la créativité et la responsabilité.

1.3 Cadre général du Projet

Ce projet vise à améliorer l'infrastructure Data de Chefclub afin d'optimiser la collecte, le stockage et l'analyse des données générées sur les réseaux sociaux. L'objectif est de soutenir les objectifs business de l'entreprise en fournissant des insights pertinents pour prendre des décisions éclairées et améliorer la performance sur les plateformes de médias sociaux.

1.3.1 Problématique

La problématique abordée dans le cadre de ce projet de fin d'études est la suivante : *Comment concevoir une solution répondant aux besoins de Chefclub en termes de centralisation des données, de prise de décisions data driven, de maximisation des revenus et de présence sur les SMPs, tout en tenant compte des besoins spécifiques de l'équipe Data ?*

Cette problématique peut être traduite par les points suivants :

- **Centralisation des données et prise de décisions basée sur les données** : Chefclub souhaite centraliser ses données provenant des réseaux sociaux (SMPs) pour faciliter l'accès et la collaboration en interne. Cela permettra de regrouper les informations pertinentes dans des feuilles de calcul Google Sheets pour une vue d'ensemble cohérente. L'objectif est d'utiliser ces données pour prendre des décisions éclairées et optimiser les performances de Chefclub sur les SMPs.
- **Maximisation des revenus et de la présence sur les SMPs** : Chefclub vise à maximiser ses revenus et sa présence sur les SMPs en exploitant le potentiel des données. L'utilisation de ces informations permettra d'identifier de nouvelles opportunités de croissance, d'optimiser les performances et d'augmenter les revenus de Chefclub. La solution proposée devra efficacement exploiter les données pour soutenir ces objectifs.
- **Besoins spécifiques de l'équipe Data** : L'équipe Data de Chefclub a des besoins spécifiques à prendre en compte dans la solution. Elle souhaite migrer vers une infrastructure Cloud plus stable pour économiser du temps et des ressources. L'accès régulier aux données analytiques des chaînes YouTube est nécessaire pour évaluer les performances et ajuster les actions. Une analyse approfondie des données, en se concentrant sur Facebook, est essentielle pour extraire des insights pertinents et soutenir la prise de décision stratégique. En outre, l'équipe souhaite développer un modèle de Machine Learning pour prédire les performances futures des posts Facebook, basé sur les données historiques, afin d'anticiper les tendances. La solution proposée doit répondre à ces besoins spécifiques tout en centralisant les données, en prenant des décisions basées sur les données et en maximisant les revenus et la présence sur les SMPs pour Chefclub.

1.3.2 Objectifs du projet

Les objectifs du projet sont les suivants :

- **Migration vers le Cloud de GCP** : Effectuer la migration des services Airbyte et Airflow de Chefclub depuis l'infrastructure locale vers le Cloud de Google Cloud Platform (GCP) en utilisant Kubernetes (K8s). Cette migration vise à améliorer la performance et la stabilité de l'infrastructure data de Chefclub.
- **Collecte automatisée des données des réseaux sociaux** : Mettre en place un processus automatisé de collecte des données provenant des plateformes de médias sociaux telles que YouTube. Cela garantira que les données utilisées pour les décisions et le suivi des performances sont fiables et à jour.
- **Transformation des données** : Développer des processus d'extraction, de chargement et de transformation (ELT) pour nettoyer, préparer et intégrer les données collectées dans le data warehouse BigQuery.
- **Tableaux de bord interactifs** : Concevoir et mettre en œuvre des tableaux de bord interactifs alimentés automatiquement avec des données mises à jour quotidiennement. Ces tableaux de bord fourniront une visualisation approfondie des performances sur les réseaux sociaux pour une meilleure compréhension et une prise de décision éclairée.
- **Analyse des données** : Utiliser des techniques d'analyse de données pour identifier des tendances, des modèles et des insights pertinents à partir des données sociales. Cette analyse approfondie permettra d'améliorer les stratégies de Chefclub et de prendre des décisions plus éclairées.
- **Modèle d'apprentissage automatique** : Développer un modèle d'apprentissage automatique basé sur les données historiques pour prédire les performances futures sur les réseaux sociaux. Ce modèle permettra une planification plus précise et une prise de décision basée sur des prévisions fiables.

Ces objectifs globaux visent à fournir à Chefclub une infrastructure de données optimisée, automatisée et évolutive, permettant une analyse approfondie et des décisions basées sur des données fiables et à jour. La réalisation de ces objectifs contribuera à l'amélioration des performances et à la maximisation des revenus de Chefclub sur les réseaux sociaux.

1.3.3 Périmètre du projet

Le périmètre du projet comprend les éléments suivants :

1. **Collecte automatisée des données des plateformes de médias sociaux** : Intégration des API des principales plateformes de médias sociaux pour extraire les statistiques d'engagement et les performances des publications et des pages de Chefclub.
2. **Intégration de sources de données supplémentaires** : Ajout des données provenant des autres SMPs pour obtenir une vue complète des performances de Chefclub sur différents réseaux sociaux.
3. **Stockage des données dans BigQuery et GCS** : Centralisation des données collectées dans BigQuery, le data warehouse de Google Cloud Storage, pour un stockage évolutif et un accès facile aux données.

4. Développement de processus ETL : Mise en place de processus ETL (Extract, Transform, Load) pour nettoyer, transformer et charger les données collectées dans BigQuery, garantissant ainsi leur qualité et leur préparation pour l'analyse.

5. Conception de tableaux de bord interactifs : Création de tableaux de bord interactifs permettant de visualiser les performances de Chefclub sur les réseaux sociaux, en fournissant des métriques clés telles que les vues et les revenus.

6. Utilisation de techniques d'analyse : Application de techniques d'analyse pour extraire des informations pertinentes à partir des données historiques, afin de comprendre le comportement de l'audience de Chefclub et d'identifier les contenus performants.

7. Développement d'un modèle d'apprentissage automatique : Création d'un modèle d'apprentissage automatique utilisant les données historiques pour prédire les performances futures de Chefclub sur les réseaux sociaux, permettant ainsi de prendre des décisions stratégiques basées sur des informations prédictives.

8. Automatisation de l'ingestion des données : Mise en place de processus automatisés pour collecter et traiter les données sociales, réduisant ainsi le temps nécessaire et permettant une analyse approfondie.

9. Migration vers le Cloud GCP : Migration des services locaux d'Airflow et Airbyte vers le Cloud de Google (GCP) en utilisant Kubernetes, assurant ainsi une infrastructure data plus stable.

10. Maintenance de l'infrastructure Data : Surveillance, résolution des problèmes techniques, mises à jour régulières et sauvegardes pour garantir la disponibilité, l'intégrité et la sécurité des données de Chefclub.

En résumé, le périmètre du projet comprend l'automatisation de l'ingestion des données, la migration d'Airflow et Airbyte vers le Cloud GCP avec Kubernetes, la création de tableaux de bord interactifs, l'analyse des données, le développement d'un modèle d'apprentissage automatique, ainsi que la maintenance de l'infrastructure data. Ces initiatives permettront à Chefclub de collecter, stocker, traiter et analyser efficacement les données provenant des plateformes de médias sociaux et d'autres sources, tout en assurant une disponibilité, une performance et une sécurité optimales.

1.3.4 Impact du projet

Le projet aura un impact significatif sur l'évolution de l'entreprise Chefclub, en apportant les avantages suivants :

- **Prise de décisions éclairées :** L'automatisation de l'ingestion des données et les tableaux de bord interactifs permettront au département financier de Chefclub de surveiller les données des médias sociaux. Cela facilitera la génération de rapports financiers précis et offrira une vision claire des performances financières. L'analyse avancée permettra également d'estimer les revenus futurs, aidant ainsi les décideurs dans leur planification financière.

- **Réduction du temps d'ingénierie :** En automatisant les processus de collecte, de transformation et de chargement des données, le projet permettra de gagner du temps pour l'équipe d'ingénierie. Cela permettra de réaffecter les ressources techniques à des projets à plus forte valeur ajoutée, tels que le développement de nouvelles fonctionnalités et l'analyse approfondie des données. L'entreprise pourra ainsi optimiser l'utilisation de ses compétences techniques et accélérer la réalisation de ses projets.

- **Optimisation de la planification et de la performance marketing :** L'analyse appro-

fondée des données sociales et l'utilisation de modèles d'apprentissage automatique permettront à Chefclub d'obtenir des insights précieux pour optimiser ses activités marketing. Ces insights aideront à identifier les tendances, les modèles et les facteurs clés de succès sur les plateformes de médias sociaux, ce qui permettra à l'entreprise d'ajuster ses stratégies et ses campagnes en temps réel. Ainsi, Chefclub pourra améliorer sa visibilité en ligne, augmenter l'engagement des utilisateurs et stimuler sa croissance.

En résumé, le projet aura un impact majeur sur Chefclub en permettant une prise de décisions éclairées grâce à des rapports financiers précis, en réduisant le temps d'ingénierie pour des tâches de moindre valeur ajoutée, en optimisant la planification et la performance marketing pour soutenir la croissance de l'entreprise sur les plateformes de médias sociaux.

1.4 Conduite de projet

Pour la réalisation de notre projet, nous avons décidé d'adopter la méthode **Scrum** de l'approche **Agile**. Cette décision découle de plusieurs raisons, dont la principale est que nous n'avions pas de conception initiale complète de la solution. Étant donné cette incertitude et la nécessité de tester différentes approches, il était clair que l'approche traditionnelle en cascade (*Waterfall*) n'était pas adaptée. Dans notre contexte, l'approche itérative et agile de Scrum s'est avérée plus appropriée. Elle nous permet de progresser par étapes, d'itérer régulièrement et de recueillir des retours d'expérience précieux. Cela nous permet d'ajuster notre approche en temps réel et de répondre aux besoins évolutifs de Chefclub de manière flexible et efficace. Ainsi, en adoptant Scrum, nous pouvons garantir une meilleure **collaboration** entre les membres de l'équipe, une **livraison itérative** des fonctionnalités et une **satisfaction continue** des besoins de Chefclub tout au long du projet.

1.4.1 Processus du développement

Le processus de développement utilisant la méthode Scrum offre un cadre itératif et incrémental pour la construction du produit. Il s'appuie sur des principes clés visant à maximiser la collaboration, l'adaptabilité et la livraison régulière de logiciels fonctionnels. Dans le cadre de notre projet, nous avons adopté la méthode Scrum pour guider notre processus de développement. Voici les éléments clés de notre approche :

1.4.1.1 Sprints et itérations

Nous avons structuré notre projet en sprints de durée fixe, généralement d'une à deux semaines. Chaque sprint représente une itération pendant laquelle nous nous concentrerons sur la réalisation de fonctionnalités spécifiques. La durée relativement courte des sprints nous permet de nous adapter rapidement aux changements et d'obtenir des résultats tangibles à intervalles réguliers.

1.4.1.2 Réunions Scrum

Au cours de chaque sprint, nous avons organisé des réunions Scrum essentielles pour favoriser la transparence et la collaboration au sein de l'équipe. Voici les réunions clés que nous avons tenues :

- **Planification de sprint** : Cette réunion marque le début de chaque sprint. Le chef de l'équipe data et l'un des fondateurs de l'entreprise se réunissent pour déterminer les objectifs du sprint

et sélectionner les fonctionnalités à développer. Nous évaluons également les efforts requis pour chaque fonctionnalité et répartissons les tâches entre les membres de l'équipe.

- **Réunion quotidienne (Daily Stand-up)** : Chaque jour, nous avons tenu une réunion de stand-up pour partager nos avancées, discuter des défis rencontrés et coordonner nos efforts. Ces réunions rapides et structurées nous ont permis de rester synchronisés et de détecter rapidement d'éventuels problèmes nécessitant une attention particulière.
- **Démo du sprint** : À la fin de chaque sprint, nous organisons une démonstration pour présenter les fonctionnalités développées pendant la période. Cette réunion nous permet d'évaluer le travail accompli, de recevoir les retours du management et de valider les résultats obtenus. Les démonstrations de sprint favorisent une communication ouverte et régulière avec les parties prenantes du projet.
- **Rétrospective de sprint** : Après chaque démo de sprint, nous tenons une réunion de rétrospective pour évaluer notre performance et identifier les améliorations potentielles. Nous discutons des aspects positifs du sprint, des difficultés rencontrées et des actions à entreprendre pour optimiser notre processus. Cette rétroaction continue nous permet de tirer des leçons précieuses et d'ajuster notre approche pour les sprints suivants.

Ces réunions Scrum ont été cruciales pour maintenir une dynamique de travail efficace, favoriser la communication et l'adaptabilité, et garantir la réalisation continue des objectifs du projet. La figure 1.7 illustre la méthodologie Agile Scrum utilisée par l'équipe Data de Chefclub pour accomplir ses tâches.

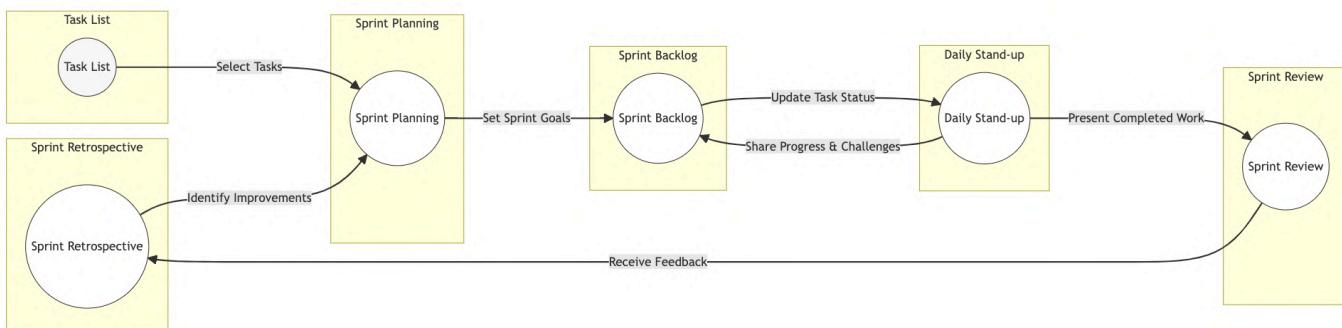


FIG. 1.7 : Chefclub Data Team Agile Scrum Process

1.4.1.3 Avantages de la méthode Scrum

L'utilisation de la méthode Scrum dans notre processus de développement présente plusieurs **avantages significatifs**. Tout d'abord, elle favorise la **collaboration** et la **communication régulière** entre tous les membres de l'équipe, ce qui permet une meilleure compréhension des besoins et une résolution rapide des problèmes. De plus, la méthode Scrum permet une **adaptabilité accrue** grâce à ses **itérations courtes**, ce qui nous a permis de nous ajuster rapidement aux changements de priorités et d'exigences. La méthode Scrum nous a également aidés à **livrer des fonctionnalités opérationnelles à intervalles réguliers**, offrant ainsi une **valeur ajoutée** aux parties prenantes et leur permettant de fournir un **feedback continu**.

En résumé, notre processus de développement s'est appuyé sur la méthode Scrum, en utilisant des **sprints**, des **réunions Scrum**, des **artefacts visuels** et une **collaboration étroite** avec l'équipe.

Chefclub. Cette approche **itérative et incrémentale** a permis d'obtenir des **résultats concrets** tout en favorisant la **flexibilité, la collaboration et la transparence**.

1.4.2 Planification du projet

La **planification** joue un rôle essentiel dans la réussite d'un projet. Elle consiste à anticiper et organiser le déroulement du projet tout au long de ses différentes phases, en suivant un cycle approprié. L'objectif principal de la planification est d'optimiser la gestion du **temps** et des **ressources** disponibles. Pour ce faire, le projet est décomposé en **tâches**, qui sont ensuite **estimées** en termes de charge, puis réparties entre les membres de l'équipe. Cette **estimation des charges** nous permet d'évaluer les besoins en ressources et d'établir une **date prévisionnelle de fin** de projet. Le respect du **planning établi** est crucial, car il conditionne la réussite globale du projet.

Dans le cadre de notre projet, nous avons accordé une attention particulière à sa **planification** en utilisant plusieurs outils. Nous avons employé le **diagramme de Gantt** pour obtenir une vue d'ensemble des différentes tâches composant le projet. De plus, nous avons utilisé **Google Calendar** pour gérer efficacement le temps alloué à chaque tâche. Nous avons également mis en place une solution interne appelée **Chefclub Timesheet**, qui nous permet de remplir une feuille de suivi. Cette approche facilite la gestion des ressources humaines par Chefclub pour ce projet, en garantissant un suivi précis des efforts et des contributions de chaque membre de l'équipe. La Figure 1.8 suivante présente un résumé de notre planification de projet :

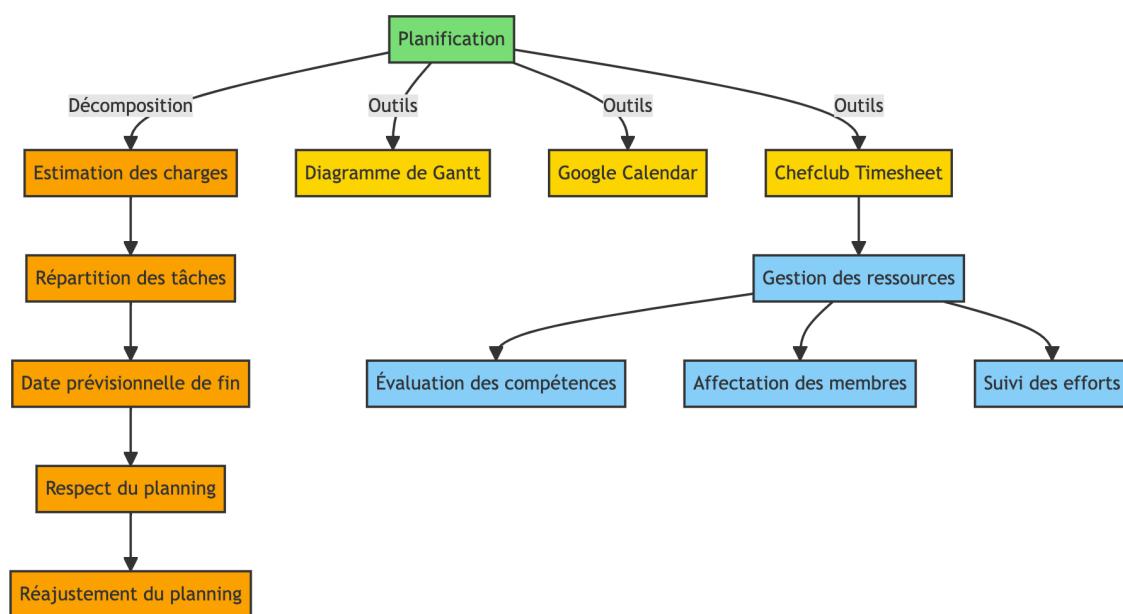


FIG. 1.8 : Project Planning Workflow

1.4.2.1 Diagramme de Gantt

Pour visualiser et suivre le déroulement du projet, nous avons utilisé un **diagramme de Gantt**. Ce diagramme nous permet de représenter graphiquement la séquence des tâches, leurs dépendances et leur durée respective. Grâce à cette visualisation, nous pouvons facilement identifier les étapes clés du projet, les jalons importants et les éventuels retards ou chevauchements de tâches. Le diagramme de Gantt nous aide à planifier et à coordonner les différentes activités du projet de manière efficace, en nous permettant de voir l'avancement global et de prendre les mesures nécessaires en cas de

déviations par rapport au planning initial. Ci-dessous, la figure 1.9 illustre le **diagramme de Gantt** pour le projet :

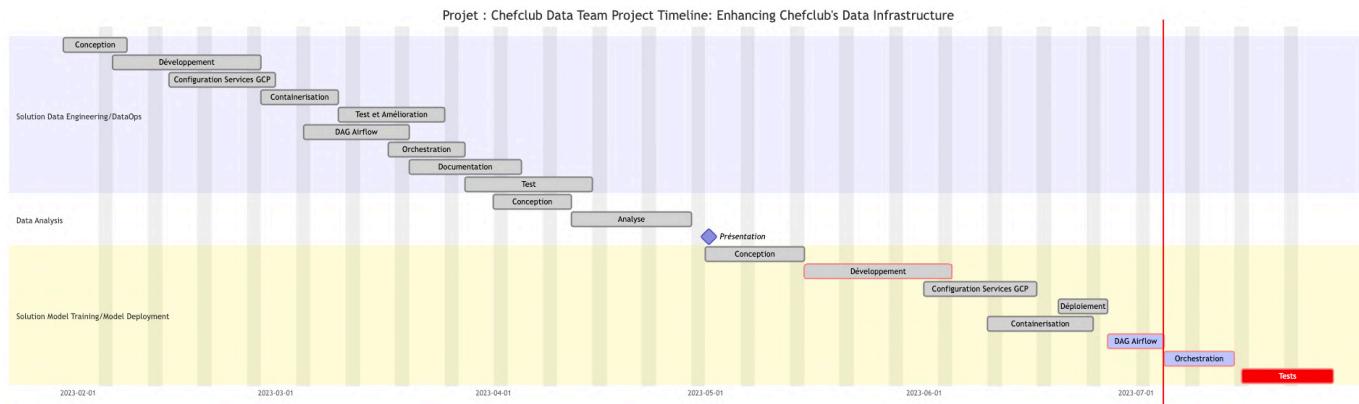


FIG. 1.9 : Project Gantt Chart

Ce diagramme illustre de manière visuelle les différentes phases du projet, les tâches associées à chaque phase, ainsi que les dépendances entre les tâches. Il permet à toute l'équipe de projet de comprendre rapidement les étapes clés et de suivre le progrès global du projet. En cas de retard ou de modification des tâches, le diagramme de Gantt facilite l'identification des ajustements nécessaires pour maintenir le projet sur la bonne voie. Voici une description des principales tâches et objectifs du projet :

1. Solution Data Engineering/DataOps

- Conception :
 - Concevoir l'architecture initiale de la solution répondant aux besoins identifiés.
- Développement :
 - Utilisation et test de l'API YouTube :
 - * Explorer l'API YouTube Analytics et effectuer des tests de rapport.
 - Gestion de l'Oauth :
 - * Mettre en place la gestion des mécanismes de sécurité pour accéder aux données historiques.
 - Extraction des données :
 - * Développer un code dans un environnement virtuel (VENV) pour extraire les données analytiques.
- Configuration Services GCP :
 - Configuration de Google Cloud Storage (GCS) :
 - * Configurer Google Cloud Storage comme zone de staging pour les données extraites.
 - Configuration de Google BigQuery :
 - * Configurer Google BigQuery pour le traitement et la transformation des données.
- Containerisation :
 - Utiliser Docker pour containeriser le code local permettant de récupérer les données analytiques.

- Test et Amélioration :
 - Test du code containerisé :
 - * Tester le code sur différents types de rapports afin de vérifier sa validité.
 - Amélioration du code containerisé.
- DAG Airflow :
 - Développer un DAG Airflow qui s'exécute quotidiennement et utilise le code containerisé.
- Orchestration :
 - Mettre en place l'orchestration du code containerisé avec Airflow et Kubernetes.
- Documentation :
 - Documenter sur GitHub le code développé et assurer sa simplicité de reproduction.
- Test :
 - Vérifier la validité de la solution et la crédibilité des rapport.

2. Data Analysis

- Conception :
 - Concevoir l'architecture initiale de la solution répondant aux besoins identifiés :
 - * Définition et choix des technologies.
 - * Choix des données à analyser : Comparaison des différentes tables sur BigQuery et les données de Facebook A/B Test.
 - * Conception de l'environnement de travail d'analyse : Paramétrage du VENV et installation des différentes librairies nécessaires.
- Analyse :
 - Collecte des données :
 - * Organisation des données dans le Google Cloud Storage.
 - Nettoyage et préparation des données :
 - * Prétraitement des données, création de nouvelles features.
 - Exploration des données :
 - * Comprendre à la fois le business et les données.
 - Analyse des données :
 - * Donner un aspect visuel aux données.
 - Génération d'insights :
 - * Interprétation des résultats des graphes obtenus.
- Présentation :
 - Préparation de la présentation :
 - * Organisation des insights dans un Notion pour une présentation simple et claire.
 - Présentation devant l'équipe.

3. Solution Model Training/Model Deployment

- Conception :

- Concevoir une architecture initiale répondant aux besoins de création et de déploiement d'un modèle d'apprentissage automatique.
- Développement :
 - Élaborer le code d'entraînement du modèle.
 - Conteneuriser le code d'entraînement du modèle.
- Configuration Services GCP :
 - Configurer Google Cloud Storage (GCS) comme zone de stockage temporaire pour le modèle entraîné.
 - Configurer BigQuery pour récupérer régulièrement les tables de performances Facebook nécessaires à l'entraînement du modèle.
 - Configurer la Cloud Function comme environnement d'inférence du modèle.
 - Configurer Slack pour la communication avec la Cloud Function.
- Déploiement :
 - Effectuer des inférences du modèle à partir de Slack.
- Containerisation :
 - Utiliser Docker pour encapsuler le code local permettant de récupérer les données analytiques.
- DAG Airflow :
 - Créer un DAG Airflow qui s'exécute quotidiennement pour entraîner le modèle.
- Orchestration :
 - Mettre en place l'orchestration du code conteneurisé avec Airflow et Kubernetes.
- Tests :
 - Évaluer les performances du modèle :
 - * Tester les performances du modèle et évaluer sa précision.
 - Vérifier l'inférence du modèle depuis Slack et la mise à jour quotidienne :
 - * Tester le bon fonctionnement de l'inférence du modèle via Slack et sa mise à jour régulière.

1.4.2.2 Google Calendar

Google Calendar est un outil essentiel que nous avons utilisé pour la gestion régulière du temps alloué aux tâches du projet. Il nous permet de créer des événements spécifiques pour chaque tâche et de les assigner aux membres de l'équipe concernés. Cela facilite la visualisation des horaires et des délais de chaque tâche de manière claire et organisée. De plus, Google Calendar offre des fonctionnalités de rappel et de notification qui nous aident à respecter les échéances et à maintenir une communication fluide entre les membres de l'équipe. Grâce à Google Calendar, nous pouvons mieux gérer notre emploi du temps et éviter les conflits de planification, ce qui contribue à maintenir le projet sur la bonne voie.

Ci-dessous, vous trouverez un aperçu de Google Calendar utilisé dans notre projet :



FIG. 1.10 : Google Calendar - Saad LABRIJI's View

1.4.2.3 Chefclub Timesheet

Pour assurer une gestion efficace des ressources humaines dans notre projet, nous avons utilisé Chefclub Timesheet. Cet outil nous permet de remplir une feuille de suivi détaillée pour chaque membre de l'équipe, où nous enregistrons les heures travaillées, les tâches effectuées et les charges associées. Cela nous offre une vue précise de l'effort et du temps consacrés à chaque tâche, ainsi qu'un suivi de la progression individuelle des membres de l'équipe. Grâce à Chefclub Timesheet, nous pouvons facilement identifier d'éventuels déséquilibres de charge de travail et prendre les mesures nécessaires pour les résoudre, garantissant ainsi une utilisation efficace des ressources humaines dans le projet.

Ci-dessous, vous trouverez un aperçu de Chefclub Timesheet utilisé dans notre projet :

Mon, 1 May 23	TECH DATA	Status:	46	50	53	54	55	56	15	58	59	60	61	62	65	66
2023-07-03-5306	Team:		Inactive	Active	Active	Active	Active	Active	Active	Active	Active	Active	Active	Active	Active	Active
Copy cell A2 below to autofill (CMD + C, CMD + SHIFT + V)																
2023-05-01-5306	Day	Employee	Total	EVENTS OTHER	STAFF FORREPRENTS	CORPORATE_GM	MANAGEMENT	OFF_ABSENCE	TACHE_ADMIN	ORIGINAL_OPERATIONS	BRAND_PARTNERSHIP	LICENCE	MEDIA	RETAIL	ECOMMERCE	SNSP_OPERATIONS
2023-05-01-5306	Mon, 1 May 23	53067 Max Perdrigeat	100					100	100	35	35					15
	Tue, 2 May 23	53176 Saad Labrij	100						12.5	67.5						
	Wed, 3 May 23	53067 Max Perdrigeat	100						10							
	Thu, 4 May 23	53176 Saad Labrij	100						6.25	93.75						
	Fri, 5 May 23	53067 Max Perdrigeat	100						10	65						
	Fri, 5 May 23	53176 Saad Labrij	100						6.25	85.75						8
	Mon, 8 May 23	53067 Max Perdrigeat	100						10	65						25
	Mon, 8 May 23	53176 Saad Labrij	100						6.25	88.75						5
	Tue, 9 May 23	53067 Max Perdrigeat	100							20						
	Tue, 9 May 23	53176 Saad Labrij	100							6.25	85.75					
	Wed, 10 May 23	53067 Max Perdrigeat	100							10	55					
	Wed, 10 May 23	53176 Saad Labrij	100							12.5	87.5					
	Thu, 11 May 23	53067 Max Perdrigeat	100							15	65					
	Thu, 11 May 23	53176 Saad Labrij	100							6.25	93.75					
	Fri, 12 May 23	53067 Max Perdrigeat	100							10						20
	Fri, 12 May 23	53176 Saad Labrij	100							6.25	93.75					80
	Mon, 15 May 23	53067 Max Perdrigeat	100								5					95
	Mon, 15 May 23	53176 Saad Labrij	100							6.25	93.75					

FIG. 1.11 : Chefclub Timesheet - Data Team View

1.5 Conclusion

En conclusion, ce chapitre a fourni une vue d'ensemble du projet en mettant en avant l'organisme d'accueil, Chefclub, et en exposant les motivations et les problématiques du projet. Nous avons également présenté en détail la conduite du projet, en mettant l'accent sur le processus de développement et la planification. Ces informations permettent au lecteur de mieux appréhender les enjeux et les objectifs du projet, ainsi que les différentes étapes de sa réalisation. Dans le prochain chapitre, nous approfondirons l'état de l'art dans le domaine.

Chapitre 2

Etat de l'art : Contexte et Technologies Associées

Le chapitre 2 de ce mémoire se concentre sur le *contexte* et les *technologies* associées au projet. Son objectif principal est de fournir un aperçu de l'état de l'art des sciences de données en abordant les différentes disciplines qui les composent, ainsi que les synergies qui en découlent. De plus, je présenterai les concepts et les technologies clés qui sont essentiels à la réalisation de notre projet. Grâce à ce chapitre, le lecteur acquerront une compréhension approfondie du paysage actuel des sciences de données, ainsi que du milieu et des outils utilisés pour accomplir nos tâches et atteindre les objectifs clés énoncés dans le premier chapitre.

2.1 Introduction

Pendant mon stage, j'ai eu l'opportunité d'explorer différents aspects des **sciences de données**, ce qui m'a permis d'acquérir une vision globale de la chaîne de valeur des données et de comprendre les composantes clés de ce domaine en plein essor. Dans cette partie de mon rapport, je vais *présenter l'état de l'art des sciences de données*, en mettant l'accent sur les concepts, les techniques et les outils spécifiques aux tâches que j'ai réalisées au cours de mon stage.

Les données sont désormais une ressource stratégique pour les entreprises. Elles fournissent des informations précieuses pour prendre des décisions éclairées, optimiser les processus opérationnels et mieux comprendre les besoins des clients. En exploitant efficacement les données, les entreprises peuvent bénéficier d'un avantage concurrentiel significatif en identifiant de nouvelles opportunités, en améliorant leur prise de décision et en proposant des produits adaptés au marché.

Dans la section suivante, j'examinerai les principaux disciplines des sciences des données qui ont émergé grâce aux avancées technologiques. Ces domaines sont spécifiquement conçus pour répondre aux divers besoins liés à la gestion et à l'exploitation des données.

2.2 Sciences des Données

La *Sciences des Données*, également connue sous le nom de *Data Science* (DS), est un domaine interdisciplinaire qui combine des concepts et des techniques issus de divers domaines tels que les mathématiques, la statistique, l'informatique et le business. Elle se concentre sur l'extraction de connaissances et des informations à partir de grandes quantités de données et de créer des modèles prédictifs en utilisant des méthodes analytiques et des outils technologiques avancés.

Les sciences des données jouent un rôle essentiel dans la résolution de problèmes complexes et dans la prise de décisions éclairées au sein des entreprises. Elles permettent d'explorer, d'analyser et d'interpréter les données pour découvrir des modèles, des tendances et des relations cachées. Grâce à ces informations, les entreprises peuvent prendre des décisions stratégiques, améliorer leurs processus et optimiser leurs performances.

La figure 2.1 ci-dessous met en évidence les mots clés de ce domaine. Il est très probable qu'un de ces mots clés soit mentionné lorsqu'on aborde le domaine de la Data Science.



FIG. 2.1 : Data Science Word Cloud Illustration

Le Word Cloud (Fig. 2.1) est une technique utilisée en DS, plus précisément dans le traitement automatique du langage naturel (Natural Language Processing, NLP). Elle permet de visualiser les

mots les plus *importants* et *fréquents* présents dans un texte ou un corpus de données. Dans notre contexte, le nuage de mots permet de mettre en évidence les termes clés liés à notre sujet d'étude. Les mots les plus significatifs sont représentés en *plus grand* et sont *plus visibles*, offrant ainsi une représentation visuelle de l'importance des différents concepts.

2.2.1 Disciplines des Sciences des Données

Le domaine des sciences des données englobe différentes disciplines qui jouent un rôle essentiel dans l'exploitation et l'analyse des données. Chaque discipline contribue à une étape spécifique du processus, depuis la collecte des données jusqu'à la mise en production des modèles Machine Learning. L'ingénierie de sciences de données peut avoir plusieurs casquette et travailler sur plusieurs tâches et disciplines qui peuvent être regroupées de la façon illustrée dans la figure 2.2 suivante. Cette figure met en évidence les **différentes disciplines** et **concepts clés** qui composent le domaine des sciences des données et montre leur interconnexion pour la réalisation d'un projet en Data Science.

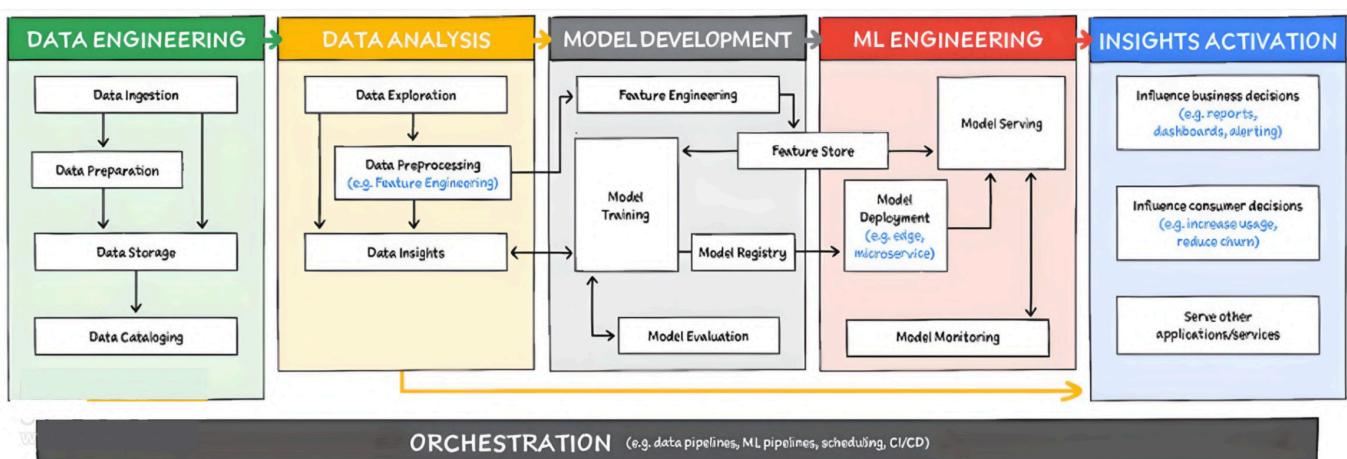


FIG. 2.2 : Synergy of Data Science Domains

Chacune de ces disciplines, présentes dans la figure 2.2, joue un rôle crucial dans le processus global d'exploitation des données, contribuant ainsi à l'amélioration des performances des entreprises et à la prise de décisions éclairées. Dans ce rapport, je propose de regrouper ces disciplines de la manière suivante pour des raisons de *synergie* et de *complémentarité* :

- **Data Engineering et DataOps** : Collecte, stockage, nettoyage, transformation, automatisation et orchestration des données.
- **Data Analysis et Insights Activation** : Exploration, compréhension, identification de tendances et de modèles, traduction en actions.
- **Model Development et ML Engineering** : Création, entraînement et déploiement de modèles d'apprentissage automatique.

Note : Ces disciplines se chevauchent souvent dans les sciences des données. Les *Data Engineer* et les *Data Scientist* peuvent assumer plusieurs rôles, en particulier dans les petites équipes. La répartition des disciplines dépend de la structure et des ressources de l'organisation.

2.2.1.1 Data Engineering et DataOps

Le *Data Engineering* englobe les processus essentiels de collecte, stockage, préparation et gestion des données. Il implique la *création de pipelines de données* pour l'ingestion fluide et fiable de diverses sources de données, ainsi que la *mise en place d'infrastructures solides* pour capturer, stocker et sécuriser efficacement les données. Les data engineers travaillent sur la transformation des données, en les nettoyant, les normalisant et les préparant pour une analyse ultérieure, garantissant ainsi leur cohérence, exhaustivité et disponibilité. Ils accordent une grande importance à la qualité des données, mettant en place des mécanismes de surveillance et de gestion des erreurs afin de garantir que les données respectent les normes et les exigences de l'entreprise, assurant ainsi des analyses précises et fiables.

la figure suivante (Fig. 2.3) illustre les différentes étapes du cycle de vie du Data Engineering, ainsi que les concepts et les mots clés associés à cette discipline.

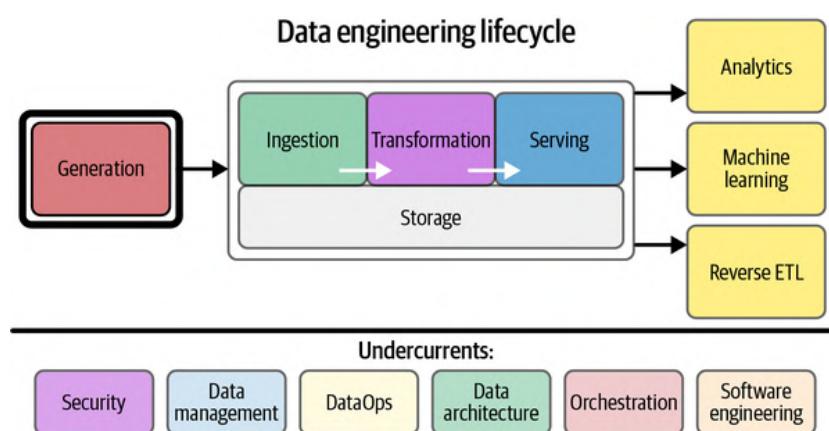


FIG. 2.3 : Data Engineering - The Lifecycle and Key Keywords

La Data Engineering englobe plusieurs étapes clés dans son cycle de vie, notamment l'ingestion, la transformation, le stockage et le service des données. Chacune de ces étapes joue un rôle essentiel pour garantir la disponibilité et la qualité des données tout au long du processus d'analyse.

Note : Le *DataOps* est une approche moderne de gestion des données qui met l'accent sur l'automatisation, la collaboration et l'intégration continues. Cette méthodologie vise à optimiser les processus de collecte, de traitement et d'analyse des données.

2.2.1.2 Data Analysis et Insights Activation

L'analyse des données (*Data Analysis*) et l'activation des insights (*Insights Activation*) sont deux aspects essentiels de l'exploitation des données pour prendre des décisions éclairées et créer de la valeur. Ils constituent des éléments clés de la science des données, en fournissant des informations précieuses pour des décisions stratégiques et éclairées.

Les *Data Analysts* analyse des données implique l'exploration approfondie et l'interprétation des données à l'aide de techniques statistiques et mathématiques en utilisant l'exploration de données (Exploratory Data Analysis, EDA). Les analystes de données identifient des **modèles** et des **tendances** significatifs, offrant ainsi des **insights exploitables**.

L'activation des insights consiste à **communiquer** et à **exploiter** les résultats de l'analyse des données. Cela implique de traduire les insights en actions concrètes, en recommandations stratégiques

et en améliorations opérationnelles, guidant ainsi les décisions basées sur les données.

Dans un projet de Science des Données, l'analyse des données et l'activation des insights aident à prendre des décisions stratégiques basées sur des preuves tangibles et faciles à communiquer.

2.2.1.3 Model Development et ML Engineering

Dans le domaine de la science des données, la *création de modèles de Machine Learning (ML)* est une étape clé dans la réalisation de projets et constitue le cœur du métier de Data Scientist. Ce processus implique l'utilisation de techniques statistiques et de Machine Learning pour construire des modèles capables d'apprendre à partir des données disponibles, de prendre des décisions ou de faire des prédictions sur de nouvelles observations. La création d'un modèle de ML comprend plusieurs étapes, telles que la *préparation des données*, l'*exploration et la transformation des caractéristiques (Feature Engineering)*, la *sélection d'un algorithme approprié*, la division des données en ensembles d'*entraînement et de test*, ainsi que l'*ajustement des hyperparamètres du modèle*. Une fois que le modèle est construit, il est évalué en utilisant différentes mesures de performance pour estimer sa capacité à généraliser sur de nouvelles données. Les projets de science des données nécessitent une approche itérative, où les modèles sont constamment améliorés et optimisés en fonction des résultats d'évaluation et des besoins spécifiques du projet. La figure suivante représente les étapes clés de la création d'un modèle ML : **Prétraitement des données**, **Entraînement du modèle** et **Évaluation du modèle**.

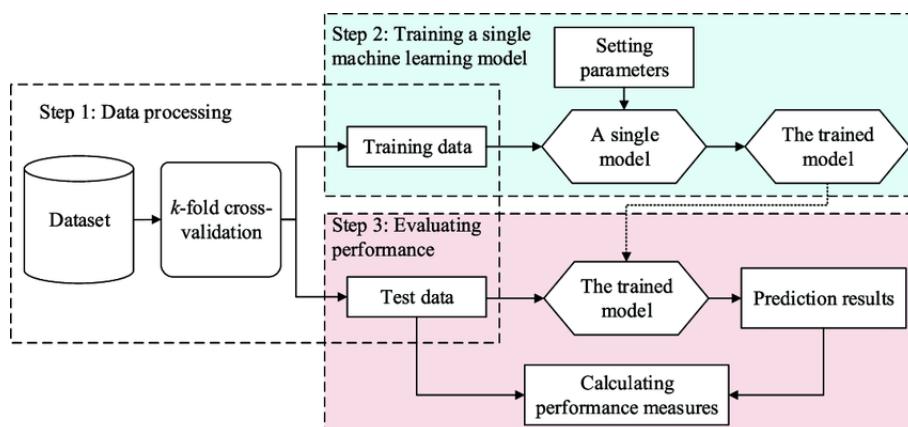


FIG. 2.4 : Machine Learning Model Creation Pipeline

Note : Cette figure (Fig. 2.4) présente les grandes lignes de la création d'un modèle ML. En réalité, pour chaque projet, ces étapes peuvent différer et nécessiter plus de détails.

La création et l'entraînement d'un modèle de ML sont des **processus itératifs** comprenant le data wrangling, l'entraînement du modèle, la validation et les ajustements si nécessaire. Le data wrangling consiste à préparer les données, tandis que l'entraînement du modèle implique l'utilisation d'un algorithme d'apprentissage pour ajuster ses paramètres (*hyperparameter tuning*). La validation est effectuée avec des données distinctes pour évaluer la précision du modèle. Si les performances sont insatisfaisantes, des ajustements peuvent être apportés au modèle, au processus de data wrangling ou de nouvelles techniques peuvent être explorées. Cette approche itérative est cruciale pour obtenir des résultats précis et fiables avec les modèles de ML.

Le schéma suivant (Fig. 2.5) illustre le cycle de vie du Machine Learning Engineering, ainsi que les différents mots clés qui englobent cette discipline.

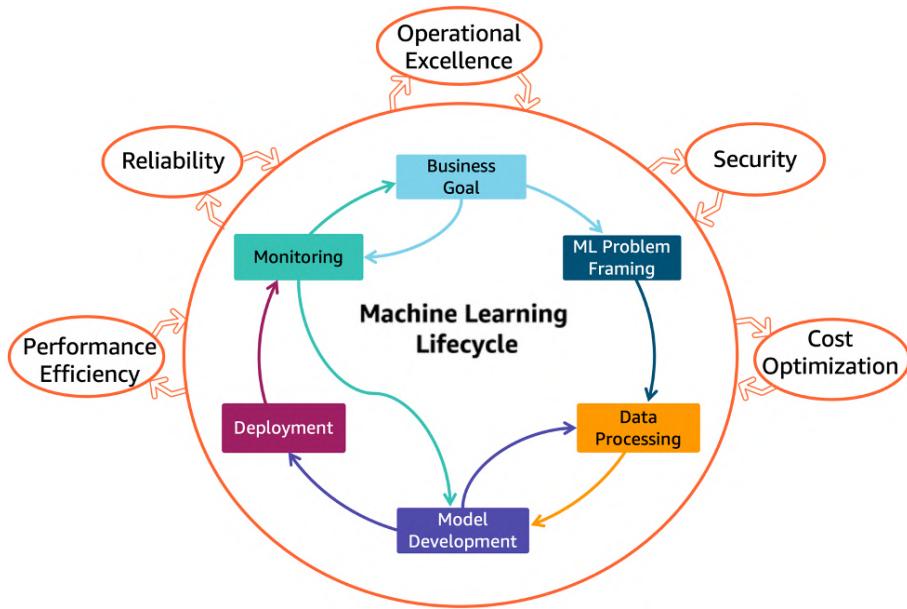


FIG. 2.5 : Machine Learning Engineering - The Lifecycle and Key Keywords

L'une des disciplines clés d'un projet en sciences des données est le déploiement des modèles ML, et c'est le rôle des ingénieurs en apprentissage automatique (ML Engineers) de s'en charger. Ils sont responsables de l'optimisation, de la mise à l'échelle et du déploiement de ces modèles afin de garantir leur fonctionnement fiable, efficace et sécurisé dans des environnements réels. Ils utilisent des techniques du MLOps (Fig. 2.6) pour cela. Les ML Engineers surveillent en permanence les performances des modèles déployés pour s'assurer de l'atteinte des objectifs métier. Ils prennent en compte les contraintes techniques et opérationnelles, tout en se concentrant sur le développement et l'optimisation des modèles, tout en assurant la sécurité des données et des informations sensibles.

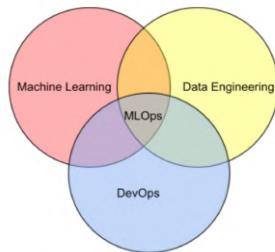


FIG. 2.6 : MLOps : The Intersection of ML, DevOps, and Data Engineering

Le MLOps (Fig. 2.6) intègre le développement logiciel et les opérations dans le cycle de vie des modèles d'apprentissage automatique (ML). Il inclut la gestion des versions, des pipelines d'apprentissage, le suivi des performances en production, la gestion des configurations, des données et des infrastructures, ainsi que la surveillance et la gestion des incidents. Son objectif est d'**améliorer la reproductibilité, la fiabilité et l'efficacité du déploiement des modèles ML**, en favorisant la collaboration entre les équipes de développement, d'exploitation et de sciences des données.

2.2.2 Techniques et Concepts Clés

Dans cette section, je vais présenter les *concepts et les techniques clés* liés à mon projet de fin d'études dans le domaine des *sciences de données*. Les sujets seront abordés de manière

parallèle à la dernière section et se déclinent comme suit :

- Concepts et Techniques Généraux en développement
- Concepts et Techniques Data Engineering et DataOps
- Concepts et Techniques Data Analysis et Insights Activation
- Concepts et Techniques Model Development et ML Engineering

2.2.2.1 Concepts et Techniques Généraux en Développement

2.2.2.1.1 Python et VENV

Python est un langage de programmation polyvalent et puissant. Il est largement utilisé dans le développement d'applications, la science des données et bien d'autres domaines.

VENV est un outil intégré à *Python* qui permet de créer des **environnements virtuels isolés**. Un environnement virtuel est une installation de *Python* indépendante qui permet d'isoler les dépendances et les packages d'un projet spécifique. Cela permet de travailler sur différents projets avec des versions de packages différentes sans qu'ils entrent en conflit les uns avec les autres, comme illustré dans la figure 2.7.

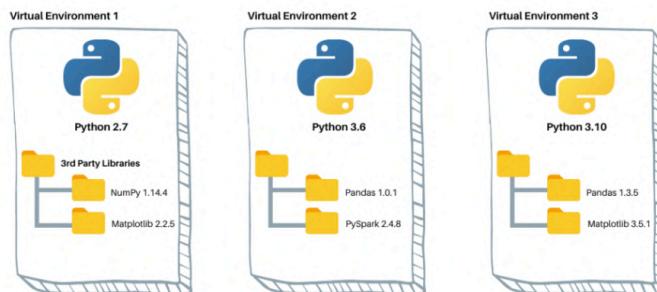


FIG. 2.7 : Illustration of a Python virtual environment (VENV) example

En combinant l'utilisation de *VENV* et *Python*, les développeurs peuvent créer des **environnements de développement Python isolés** pour chaque projet, assurant ainsi une **gestion plus efficace des dépendances** et une meilleure **portabilité du code**. Cette approche favorise la collaboration et **facilite le déploiement d'applications** *Python* dans différents environnements.

Pour plus de détails, vous pouvez consulter la documentation¹ sur les environnements virtuels *Python* (*VENV*) [**Python-VENV**].

2.2.2.1.2 API

Une *API (Application Programming Interface)* est un ensemble de règles et de protocoles qui permettent à différents logiciels de communiquer entre eux. Elle fournit une interface standardisée pour l'échange de données et l'accès aux fonctionnalités d'une application ou d'un service. Les API facilitent la communication entre les services.

Une *API REST (Representational State Transfer)* est un type spécifique d'API qui utilise le protocole HTTP pour la communication. Elle repose sur l'idée de représenter les ressources sous forme d'URL. La figure suivante (Figure 2.8) montre le fonctionnement d'une API REST.

¹docs.python.org/fr/3/library/venv.html

WHAT IS A REST API?

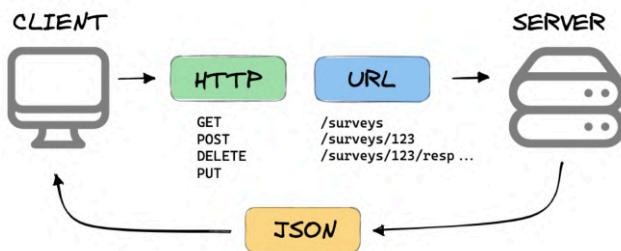


FIG. 2.8 : Functioning of a REST API

2.2.2.1.3 OAuth (Open Authorization)

OAuth est un framework d'autorisation qui permet aux utilisateurs de donner accès à leurs ressources sur un site web à une autre application sans partager leurs identifiants de connexion. Le processus implique un propriétaire de ressources (utilisateur), un client (application demandant l'accès), un serveur d'autorisation et un serveur de ressources. Le client s'enregistre auprès du serveur d'autorisation, redirige l'utilisateur vers celui-ci pour obtenir son consentement, puis échange un code d'autorisation contre un *jeton d'accès* (*Access Token*). Le client utilise ensuite ce jeton pour accéder aux ressources sur le serveur de ressources. L'*Access Token* a une durée de vie limitée et peut être rafraîchi à l'aide d'un *jeton de rafraîchissement* (*Refresh Token*). **OAuth offre une méthode sécurisée d'autorisation d'accès aux API tout en préservant la confidentialité des utilisateurs.**

La figure 2.9 ci-dessous explique le concept d'authentification OAuth.

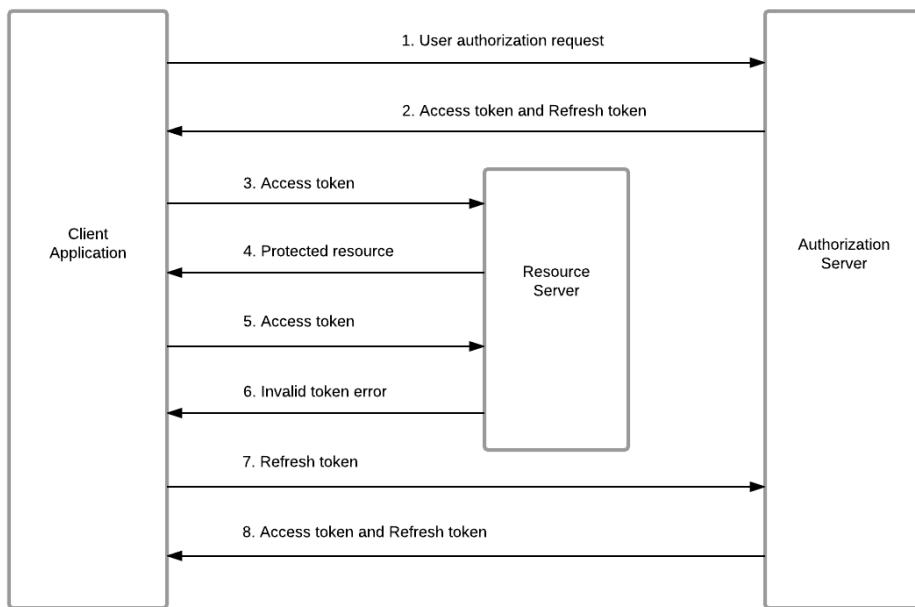


FIG. 2.9 : OAuth Access and Refresh Token Flow Diagram

Pour plus de détails, vous pouvez consulter cet article ² [OAuth-Best Practices]

² stateful.com/blog/oauth-refresh-token-best-practices.

2.2.2.1.4 Contrôle de version

Le contrôle de version (*version control*) est un système qui gère les modifications apportées aux fichiers au fil du temps, permettant à plusieurs personnes de travailler simultanément sur un projet tout en suivant toutes les modifications. Il facilite la collaboration en équipe dans un environnement de production. Les types de contrôle de version incluent :

- **Contrôle de version local (*Local Version Control - LVC*)** : Chaque développeur dispose d'une copie locale des fichiers du projet sur son propre ordinateur. Ils peuvent apporter des modifications, ajouter de nouvelles fonctionnalités ou corriger des erreurs sans affecter le code des autres développeurs. Cette approche est idéale pour les projets individuels ou les petits groupes de travail.
- **Contrôle de version centralisé (*Centralized Version Control - CVC*)** : Un référentiel central unique (*central repository*) stocke tous les fichiers et leurs versions. Les développeurs extraient les fichiers du référentiel, effectuent des modifications et valident ensuite ces modifications dans le référentiel central. Cette approche facilite la gestion des modifications et assure une cohérence du code pour tous les membres de l'équipe. Cependant, elle nécessite une connexion au référentiel central pour effectuer des opérations de contrôle de version.
- **Contrôle de version distribué (*Distributed Version Control - DVC*)** : Chaque développeur possède son propre référentiel local (*local repository*) contenant l'historique complet du projet. Les développeurs peuvent apporter des modifications à leur référentiel local de manière indépendante, puis synchroniser ces modifications avec les autres en poussant (*pushing*) ou en tirant (*pulling*) les modifications. Cette approche permet aux développeurs de travailler hors ligne et de fusionner leurs modifications avec d'autres référentiels plus tard. Elle offre une plus grande flexibilité et évite les problèmes de dépendance à un référentiel central.

La figure suivante 2.10 illustre un exemple de *DVC*. Chaque développeur dispose de son propre repository local, tandis qu'un repository central partagé est également présent. Cela permet aux développeurs d'accéder et de poursuivre leur travail même en cas de problèmes ou de pannes sur les postes de travail individuels.

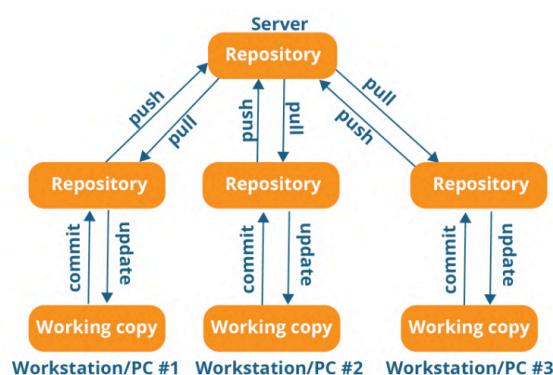


FIG. 2.10 : Illustration of Distributed Version Control System

Le *Version Control* offre plusieurs avantages, tels que la possibilité de team collaboration, de version tracking, de faire du code backup et de faciliter l'identification et la résolution des problèmes liés aux modifications du code source.

2.2.2.2 Concepts et Techniques Data Engineering et DataOps

2.2.2.2.1 Orchestration et Planification

Dans le contexte de la Data Engineering, l'*orchestration* et la *planification* du code jouent un rôle crucial pour assurer l'exécution fiable et efficace des processus ETL (Fig. 2.13). Ces concepts permettent de gérer la séquence d'exécution des tâches, la gestion des dépendances entre celles-ci et l'automatisation du flux de travail.

- **Orchestration** : L'*orchestration* du code se réfère à la coordination des différentes tâches et étapes d'un processus ETL, en veillant à ce qu'elles s'exécutent dans l'ordre approprié et selon les dépendances prédéfinies. Cela permet d'automatiser le flux de travail, de réduire les erreurs humaines et d'améliorer l'efficacité globale du processus.

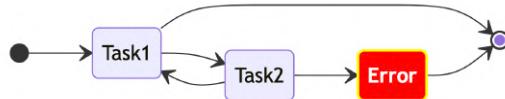


FIG. 2.11 : Orchestration Engine

Dans un environnement d'orchestration du code, chaque tâche est représentée par un morceau de code ou un script qui effectue une opération spécifique, comme l'extraction des données, la transformation ou le chargement. Ces tâches sont organisées en un graphe de dépendances, où les dépendances indiquent l'ordre d'exécution. Par exemple, une tâche de transformation peut dépendre des données extraites d'une autre tâche. L'orchestrateur veille à ce que ces dépendances soient respectées lors de l'exécution des tâches.

- **Planification** : La *planification* du code est étroitement liée à l'orchestration du code et se concentre sur la gestion du calendrier d'exécution des tâches. Elle permet de spécifier quand et à quelle fréquence chaque tâche doit être exécutée. Par exemple, une tâche d'extraction peut être planifiée pour s'exécuter chaque jour à minuit, tandis qu'une tâche de transformation peut être planifiée pour s'exécuter toutes les heures.

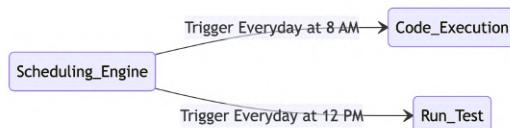


FIG. 2.12 : Scheduling Engine

La planification du code prend également en compte d'autres aspects, tels que la gestion des retards (delays), les horaires spécifiques, les conditions de déclenchement (triggers) et la gestion des erreurs. Elle offre une flexibilité pour ajuster et optimiser le calendrier d'exécution des tâches en fonction des contraintes et des besoins du projet.

2.2.2.2.2 ETL dans la Data Engineering

Les processus ETL (Extract, Transform, Load) jouent un rôle essentiel dans la Data Engineering. Ils permettent d'extraire, de transformer et de charger des données provenant de différentes sources vers une destination spécifique, telle qu'une base de données ou un entrepôt de données.

Les étapes clés du processus ETL sont les suivantes :

1. **Extraction (Extract)** : Cette étape consiste à récupérer les données à partir de sources hétérogènes telles que des bases de données, des fichiers plats ou des services web, etc.
2. **Transformation (Transform)** : Pendant cette étape, des opérations et des règles sont appliquées pour nettoyer, filtrer, agréger ou enrichir les données extraites.
3. **Chargement (Load)** : L'étape de chargement consiste à intégrer les données transformées dans une destination spécifique, généralement une base de données ou un entrepôt de données.

La figure 2.13 ci-dessous illustre un exemple de pipeline ETL :

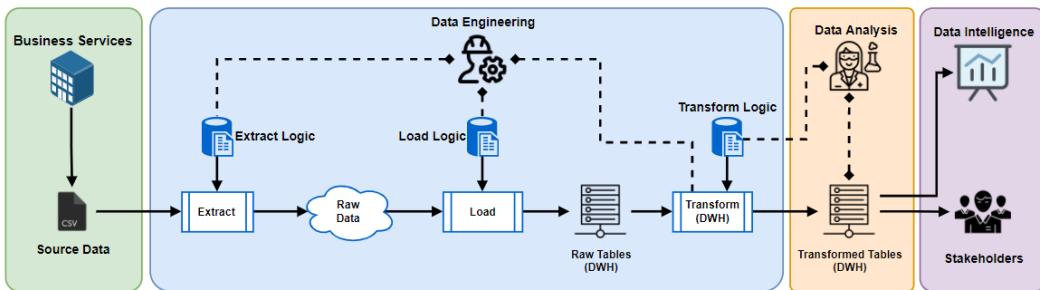


FIG. 2.13 : Example of an ETL Pipeline

La figure 2.13 représente un pipeline ETL typique, où les données sont extraites de différentes sources de services métier. Ensuite, ces données subissent une transformation à l'aide d'une logique de transformation pour les préparer à être chargées dans l'entrepôt de données. Une fois chargées, les données deviennent disponibles pour l'analyse et la génération d'intelligence au profit des parties prenantes (stakeholders).

Note 1 : La *staging area* est une zone de préparation temporaire . Elle permet d'effectuer des transformations et des manipulations sur les données avant leur intégration finale.

Note 2 : Les *ELTs (Extract, Load, Transform)* sont une approche alternative aux ETLs dans la préparation des données pour le DWH. Contrairement aux ETLs, les ELTs chargent d'abord les données brutes dans le DWH, puis effectuent les transformations ultérieurement. Cela permet d'exploiter la puissance de traitement de le DWH pour des volumes importants de données ou des transformations complexes.

Il serait encore plus intéressant d'*orchestrer* et de *planifier* ces ETLs/ELTs afin de garantir la périodique récupération et la transformation régulière des données, je présenterai dans la section 2.3.4 un outil capable de réaliser cela, appelé *Apache Airflow*.

2.2.2.3 Concepts et Techniques Data Analysis et Insights Activation

2.2.2.3.1 Boxplot

Un *boxplot (diagramme en boîte)*, également connu sous le nom de *box-and-whisker plot*, est un outil graphique utilisé pour représenter la distribution d'un ensemble de données numériques. Il fournit des informations sur la médiane (*median*), les quartiles (*quartiles*), les valeurs aberrantes (*outliers*) et l'étendue des données (*data range*). Le boxplot se compose d'une boîte et de deux moustaches (*whiskers*). La boîte représente le premier quartile, la médiane et le troisième quartile, tandis que les moustaches indiquent l'étendue des données sans les valeurs aberrantes.

Le boxplot est utile pour **comparer la distribution de plusieurs ensembles de données**

(datasets), visualiser la dispersion des données et détecter les valeurs aberrantes (*outliers*). La figure suivante (Figure 2.14) présente un exemple de boxplot sur une distribution normale (*normal distribution*).

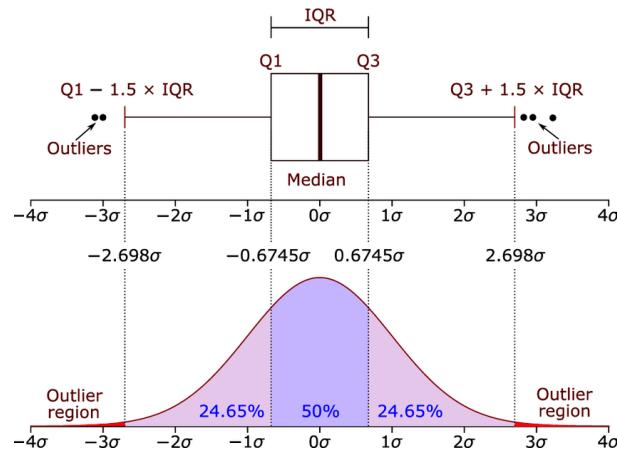


FIG. 2.14 : Boxplot on a Normal Distribution

Note : une *Distribution Normale* est une distribution statistique continue symétrique, elle est caractérisée par sa forme en cloche déterminée par la **moyenne** et l'**écart-type**.

2.2.2.3.2 Z-score et Outlier Detection

La détection de valeurs aberrantes est une technique essentielle dans l'analyse de données. Elle permet d'identifier les observations qui s'écartent significativement du reste de l'échantillon. Une méthode couramment utilisée pour détecter les valeurs aberrantes est le score Z.

Le score Z, également appelé score standard, mesure l'écart d'une observation par rapport à la moyenne de l'échantillon, en prenant en compte l'écart-type. Une valeur Z élevée indique que l'observation est éloignée de la moyenne et peut être considérée comme une valeur aberrante potentielle. La formule du score Z est donnée par :

$$Z = \frac{x - \mu}{\sigma}$$

où x est la valeur de l'observation, μ est la moyenne de l'échantillon et σ est l'écart-type de l'échantillon.

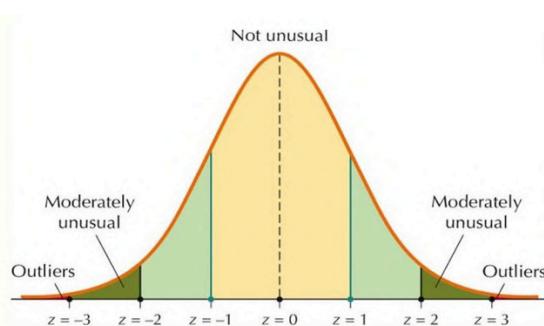


FIG. 2.15 : Detecting outliers with z-Scores

En utilisant le score Z, il est possible de définir un seuil au-delà duquel une observation est considérée comme une valeur aberrante. Par exemple, dans la figure 2.15, nous fixons un seuil de 3

pour le score Z, toutes les observations ayant un score Z supérieur à 3 seront considérées comme des outliers. Cette approche permet d'**automatiser le processus de détection d'outliers**.

2.2.2.4 Concepts et Techniques Model Development et ML Engineering

2.2.2.4.1 Le Machine Learning : Types et Utilisations

Le *Machine Learning*, également connu sous le nom d'*Apprentissage Automatique*, est un domaine de recherche en intelligence artificielle qui se concentre sur le développement d'algorithmes et de modèles capables d'apprendre à partir de données et de prendre des décisions autonomes sans être explicitement programmés. Il repose sur le principe que les systèmes informatiques peuvent analyser des ensembles de données, découvrir des motifs, et généraliser ces connaissances pour faire des prédictions ou prendre des décisions sur de nouvelles données.

Les principaux types d'apprentissage automatique sont les suivants :

- **L'apprentissage supervisé (Supervised Learning)** : Ce type d'apprentissage implique l'utilisation d'un ensemble de données d'entraînement étiquetées, où chaque exemple est associé à une étiquette prédéfinie. Les algorithmes d'apprentissage supervisé apprennent à partir de ces exemples pour créer un modèle capable de prédire les étiquettes de nouvelles données. On distingue les problèmes de régression, où le modèle cherche à prédire une valeur continue, et les problèmes de classification, où le modèle cherche à prédire une classe ou une catégorie spécifique.
- **L'apprentissage non supervisé (Unsupervised Learning)** : Contrairement à l'apprentissage supervisé, l'apprentissage non supervisé ne nécessite pas d'étiquettes dans les données d'entraînement. Les algorithmes d'apprentissage non supervisé explorent les *structures et les relations intrinsèques des données*, en identifiant des motifs, des clusters ou des associations significatives. Cela permet de découvrir des informations cachées, de réduire la dimensionnalité des données ou de segmenter les données en groupes similaires.
- **L'apprentissage par renforcement (Reinforcement Learning)** : L'apprentissage par renforcement se base sur l'idée d'un agent qui interagit avec un environnement dynamique. L'agent prend des actions dans cet environnement et reçoit des récompenses ou des sanctions en fonction de la qualité de ses actions. L'objectif de l'apprentissage par renforcement est d'apprendre une politique d'action qui maximise la somme des récompenses à long terme. Cela implique d'équilibrer l'exploration de nouvelles actions et l'exploitation des actions connues pour optimiser les performances.

Pour plus de détails sur Le Machine Learning et les algorithmes utilisés, vous pouvez consulter la documentation officielle de scikit-learn³ [Scikit-Learn-docs]

Le ML est utilisé dans divers domaines comme la vision par ordinateur, le traitement du langage naturel (Fig 2.1), l'analyse de données, la robotique et la bioinformatique. Grâce à des techniques avancées telles que les *réseaux de neurones* et les *algorithmes d'optimisation*, il résout des problèmes complexes, découvre des informations précieuses et prend des décisions intelligentes basées sur les données. La figure suivante 2.16 donne un aperçu des différentes techniques et algorithmes du Machine Learning.

³ scikit-learn.org.

Note : *Scikit-Learn* est une bibliothèque libre Python destinée à l'apprentissage automatique.

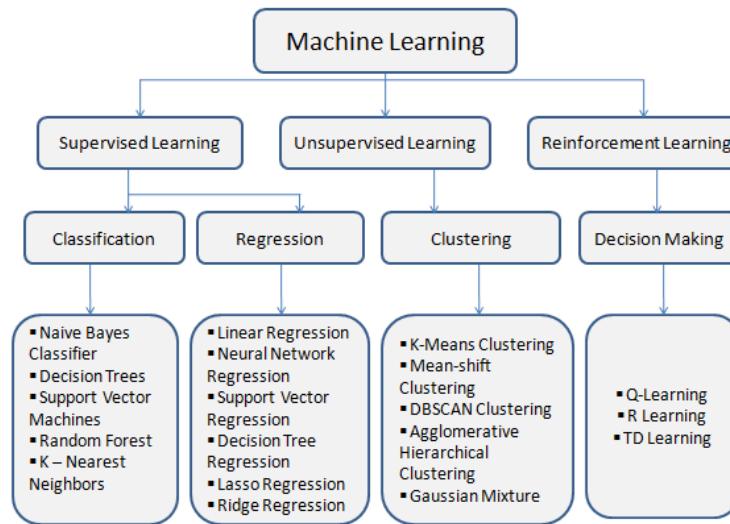


FIG. 2.16 : Comprehensive Overview of Machine Learning Techniques and Algorithms

2.2.2.4.2 Choix de l'algorithme de ML

Lors du choix d'un modèle d'apprentissage automatique (ML), il est important de prendre en compte certains aspects pour garantir des résultats fiables et précis. Deux facteurs clés à considérer sont la Matrice de Corrélation et la Distribution des Variables.

- Matrice de Corrélation (Correlation Matrix)** : Il est important d'examiner la corrélation linéaire entre les différentes variables du *Dataset*. La corrélation est une mesure statistique qui évalue la relation linéaire entre deux variables. Une corrélation élevée ou faible peut fournir des indications sur la pertinence des *features* pour le modèle et les interactions potentielles entre elles. (voir Fig 2.17)

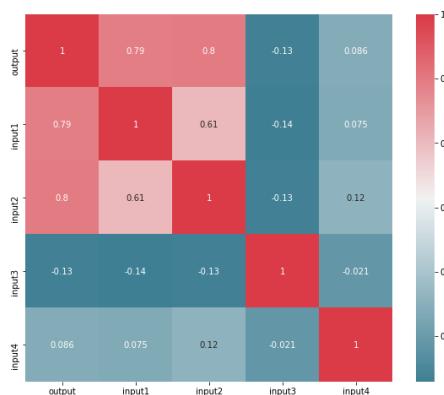


FIG. 2.17 : Correlation Matrix example

- Distribution des Variables (Variable Distribution)** : Comprendre la distribution des variables du *Dataset* est important. Cela permet de déterminer si les variables suivent une distribution normale ou si elles présentent des biais ou des valeurs aberrantes. Une distribution inadéquate peut nécessiter des prétraitements ou des transformations pour garantir des résultats optimaux du modèle.

Pour plus de détails sur les **Distributions de Probabilité**, vous pouvez consulter cet article⁴ [[probability-distribution](#)]

2.2.2.4.3 Feature Engineering

Le Feature engineering (ingénierie des caractéristiques) joue un rôle important dans la construction de modèles ML performants. Il englobe plusieurs aspects, dont deux en particulier : *Domain-driven feature* et la *Variable Distribution Optimization*.

- **Domain-driven feature** : Cette approche consiste à créer de nouvelles caractéristiques directement liées à l'expertise du domaine et aux objectifs spécifiques du problème. Ce processus permet d'extraire des caractéristiques significatives des données qui correspondent au contexte du projet. Par exemple, dans le domaine de la vente au détail, il peut s'agir de caractéristiques telles que le montant moyen des achats par client ou la fréquence d'achat d'un produit particulier.
- **Variable Distribution Optimization** : Cette étape implique l'utilisation de techniques de transformation telles que la normalisation, la mise à l'échelle ou la gestion des valeurs aberrantes afin de garantir que les variables présentent des propriétés statistiques appropriées. Cela permet d'améliorer la capacité du modèle à capturer des motifs et à effectuer des prédictions précises.

En combinant ces techniques avec d'autres disponibles dans cet article⁵ [[feature-engineering](#)], on obtient un ensemble de variables pertinentes et préparées de manière appropriée pour l'algorithme du modèle. Ces caractéristiques permettent au modèle de comprendre les schémas sous-jacents des données et d'effectuer des prédictions précises. **Le Feature engineering est une étape essentielle pour maximiser les performances et la pertinence des modèles ML.**

2.2.2.4.4 Train-Val-Test Split

Pour évaluer correctement les performances d'un modèle ML, il est courant de diviser les données en ensembles d'*entraînement*, de *validation* et de *test*.

- **Ensemble d'entraînement (Training Set)** : Cet ensemble est utilisé pour entraîner le modèle. Il contient les données sur lesquelles le modèle apprendra les relations existantes entre les caractéristiques (features) et les étiquettes (labels) fournies.
- **Ensemble de test (Test Set)** : Cet ensemble est utilisé pour évaluer les performances finales du modèle. Il permet d'estimer l'exactitude du modèle sur des données qu'il n'a jamais vues auparavant. Il évalue également la capacité du modèle à généraliser à de nouvelles instances.
- **Ensemble de validation (Validation Set)** : Cet ensemble est utilisé pour affiner les hyperparamètres et la configuration du modèle. Il sert à évaluer les performances du modèle sur des données indépendantes, mais n'est pas utilisé directement pour l'entraînement du modèle.

Voici une illustration du découpage Train-Test-Val dans la figure 2.18.

⁴ bf.refer.org/peche/chap2/chap24.html.

⁵ towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114.

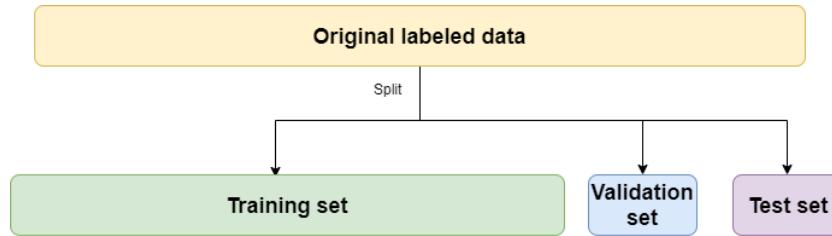


FIG. 2.18 : Test, training and validation sets

La taille relative de chaque ensemble peut varier en fonction de la quantité de données disponibles et de la complexité du modèle. Une répartition typique pourrait être de **70% pour l'ensemble d'entraînement, 15% pour l'ensemble de validation et 15% pour l'ensemble de test**. Cependant, il n'existe pas de règles strictes à ce sujet et la répartition peut être adaptée en fonction des besoins spécifiques du projet.

2.2.2.4.5 Évaluation des performances d'un modèle de Régression en ML

La régression est un type d'algorithme d'apprentissage supervisé qui consiste à prédire une variable dépendante continue en fonction d'une ou de plusieurs variables indépendantes. Afin d'évaluer les performances d'un modèle de régression, diverses mesures sont couramment utilisées :

- **MSE** (Erreur quadratique moyenne) : Cette métrique est la plus couramment utilisée dans les problèmes de régression. Elle mesure la moyenne des différences au carré entre les valeurs réelles et prédictives. Le MSE attribue le même poids à toutes les erreurs, qu'elles soient positives ou négatives. La formule du MSE est la suivante :

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

où n est le nombre d'observations, y_i est la valeur réelle et \hat{y}_i est la valeur prédictive.

- **RMSE** (Erreur quadratique moyenne de la racine) : Le RMSE est la racine carrée du MSE, ce qui donne l'erreur dans les mêmes unités que la variable dépendante. C'est une métrique populaire car elle est facile à interpréter et à comprendre. La formule du RMSE est la suivante :

$$\text{RMSE} = \sqrt{\text{MSE}}$$

- **MAE** (Erreur absolue moyenne) : Le MAE est similaire au MSE, mais au lieu de mettre les erreurs au carré, il prend la valeur absolue des erreurs. Le MAE est utile lorsque les valeurs aberrantes ont un impact élevé sur les performances du modèle. La formule du MAE est la suivante :

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

- **R2** (Coefficient de détermination) : Le R2 est une mesure statistique de l'adéquation du modèle de régression aux données. Il mesure la proportion de la variance de la variable dépendante expliquée par les variables indépendantes. Le R2 varie de 0 à 1 et une valeur plus élevée indique un meilleur ajustement. La formule du R2 est la suivante :

$$R^2 = 1 - \frac{SS_r}{SS_t}$$

où SS_r est la somme des carrés des résidus et SS_t est la somme totale des carrés.

La Table 2.1 suivante présente les **avantages** et **inconvénients** des différentes métriques de régression.

Métrique	Avantages	Inconvénients
MSE/RMSE	Accentue les fortes erreurs, régulière, optimisable	Sensible aux outliers, utilise des valeurs au carré
R2	Indique la proportion de la variance expliquée	Ne mesure pas l'erreur absolue, dépendant de l'échelle des données
MAE	Homogène, interprétable	Sensible aux outliers, ne tient pas compte des différences entre erreurs

TAB. 2.1 : Advantages and disadvantages of regression metrics

Pour plus de détails sur ces métriques de performance, vous pouvez consulter⁶ [regression-perf]

La figure suivante 2.19 présente un exemple d'Overfitting, d'Underfitting et de Good-Fit dans un exemple de série temporelle.

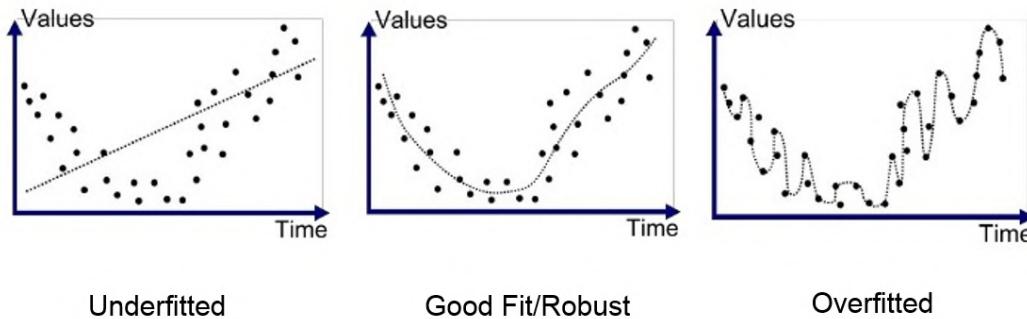


FIG. 2.19 : Time Series Regression : Underfitting Vs. Good Fit Vs. Overfitting

Note : Alors que la *Régression* est un concept plus général qui peut s'appliquer à différents types de données, le *Time Series Analysis (analyse de séries temporelles)* est un domaine spécifique qui traite des dépendances temporelles dans les données. Les techniques de régression peuvent être utilisées dans l'analyse de séries temporelles pour modéliser la relation entre les variables, et des approches de Machine Learning peuvent être employées pour analyser et prédire les données de séries temporelles.

L'objectif de l'évaluation des performances du modèle est de quantifier sa capacité à généraliser sur de nouvelles données, ainsi que de surveiller et prévenir les phénomènes d'*Overfitting* (Surajustement) et d'*Underfitting* (Sous-Ajustement).

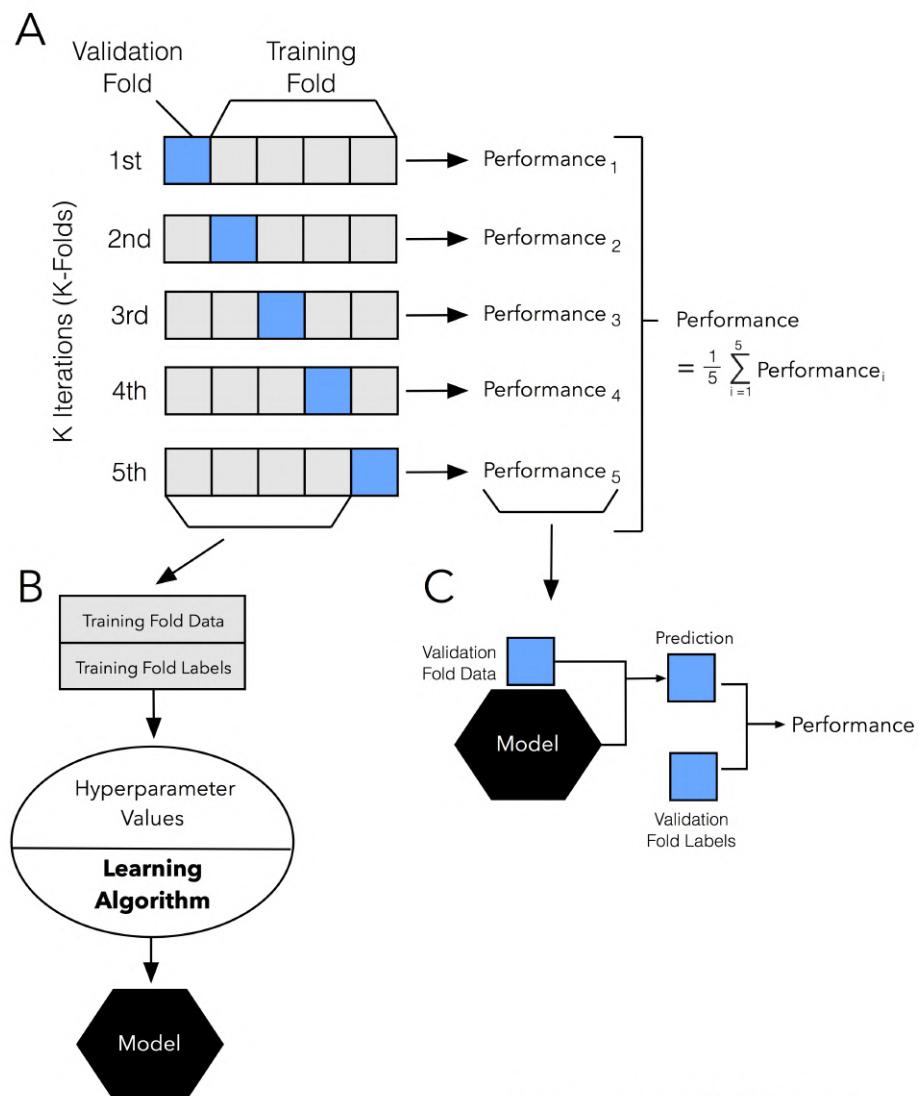
2.2.2.4.6 Recherche des hyperparamètres optimaux avec GridSearchCV

GridSearchCV est un outil utilisé pour l'optimisation des hyperparamètres en ML. Il automatise

⁶ kobia.fr/regression-metricsquelle-metrique-choisir/.

le processus de recherche de la meilleure combinaison d'hyperparamètres en évaluant de manière exhaustive les performances du modèle à l'aide de la cross-validation (validation croisée).

Note : La *Cross-Validation* est une technique utilisée en ML pour estimer comment un modèle entraîné se comportera sur des données inconnues (voir section ??). Elle implique de diviser les données disponibles en plusieurs *subsets*, d'entraîner le modèle sur certains subsets et d'évaluer ses performances sur le *subset restant*. Ce processus est répété plusieurs fois et les résultats sont ensuite moyennés pour obtenir une estimation globale des performances. La validation croisée aide à **évaluer la capacité de généralisation d'un modèle et est utile pour la sélection de modèle et l'optimisation des *hyperparamètres***.



This work by Sebastian Raschka is licensed under a Creative Commons Attribution 4.0 International License.

FIG. 2.20 : GridSearch Cross Validation Approach

Comme illustré dans la figure 2.20, une fois que le *Search Space* (l'ensemble des combinaisons d'hyperparamètres à tester) est défini, le processus de *Cross Validation* est répété pour chaque combinaison d'hyperparamètres de cet espace. Ensuite, la moyenne des performances sur les validations croisées est calculée. Cela permet **d'identifier la meilleure combinaison d'hyperparamètres qui maximise les performances du modèle** en fonction des hyperparamètres choisies.

2.2.2.4.7 Export d'un Modèle Scikit-Learn (Model Persistence)

L'exportation d'un modèle *Scikit-Learn* permet de sauvegarder l'*objet modèle entraîné* dans un fichier *pkl*, ce qui permet de le charger ultérieurement et de l'utiliser pour effectuer des prédictions sur des nouvelles données. Avant d'utiliser le modèle, il faut s'assurer de configurer l'environnement et d'effectuer les étapes de preprocessing nécessaires sur les données avant d'utiliser le modèle. Voici comment exporter un modèle entraîné :

```

1 # importing the pickle module
2 import pickle
3
4 # Export the trained model object to a pkl file
5 with open('trained_model.pkl', 'wb') as file:
6     pickle.dump(trained_model, file)
7
8 # Load the trained model from the pkl file
9 with open('trained_model.pkl', 'rb') as file:
10    trained_model = pickle.load(file)
11
12 # Predict using the loaded model
13 y_pred = trained_model.predict(x_test)

```

Listing 2.1: Model Persistence: Exporting and Loading a Scikit-Learn Model using Pickle

Pour plus de détails sur l'export de modèles avec Pickle, vous pouvez consulter la documentation suivante ⁷ [**model-persistence**]

2.3 Outils Clés

2.3.1 Le Cloud Computing

Le *Cloud Computing*, ou informatique en nuage, est un modèle de prestation de services informatiques via Internet offrant aux utilisateurs un *accès à des ressources partagées* telles que des serveurs, des bases de données, des réseaux, des logiciels et des services. Il permet aux entreprises de bénéficier de flexibilité, évolutivité et disponibilité sans gérer l'infrastructure sous-jacente, ce qui élimine les coûts élevés d'investissement en équipements informatiques et facilite la mise en place de solutions technologiques. La figure 2.22 présente une architecture simplifiée du cloud computing, avec le front-end représentant les interfaces et applications utilisées par les clients, et le back-end correspondant aux ressources gérées par les fournisseurs de services (fig. 2.21) pour fournir les services de cloud computing.



FIG. 2.21 : The Big Three Cloud Computing Providers

⁷ scikit-learn.org/stable/model_persistence.html.

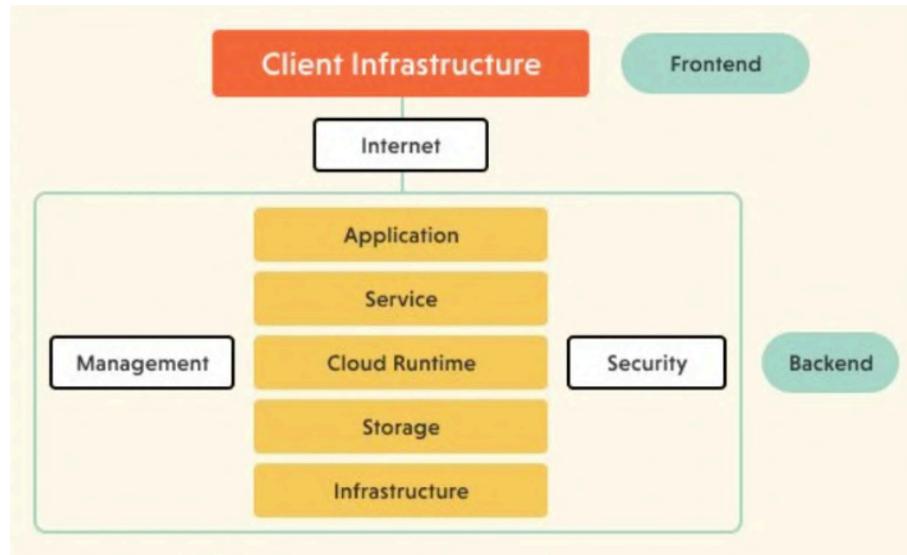


FIG. 2.22 : Cloud Computing Architecture

2.3.1.1 Le Cloud GCP

Le cloud *GCP (Google Cloud Platform)* (Fig. 2.23) est une plateforme de *Cloud Computing* fournie par Google. Elle propose un ensemble complet de services et d'outils permettant de développer, déployer et gérer des applications et des services dans le cloud. Le cloud GCP est conçu pour offrir une évolutivité, une performance et une fiabilité élevées, ainsi qu'une intégration étroite avec les autres produits et services de Google.

Le cloud GCP *GCP (Google Cloud Platform)* (Fig. 2.23) est une plateforme de *Cloud Computing* fournie par Google. Il offre un éventail complet de services et d'outils pour le **développement**, le **déploiement** et la **gestion d'applications** et de services dans le cloud. Avec une évolutivité, des performances et une fiabilité exceptionnelles, ainsi qu'une intégration étroite avec d'autres produits et services de Google, le cloud GCP permet aux développeurs et aux entreprises de bénéficier d'une infrastructure puissante et flexible. Des services tels que le stockage de données, l'analyse de données, l'intelligence artificielle et le Machine Learning, les outils de développement, la sécurité et la mise en réseau sont disponibles, permettant la création de solutions personnalisées. Le cloud GCP se distingue également par sa facilité d'intégration avec des technologies populaires telles que *Kubernetes*, *Docker* et *Jenkins*, simplifiant ainsi la création et le déploiement d'applications modernes.



FIG. 2.23 : GCP - Logo

GCP propose plusieurs services pour mener des Sciences de Données dans diverses disciplines (voir Figure 2.2). La Figure suivante (2.24) présente les différents services que l'on peut utiliser pour chaque discipline des Sciences de Données.

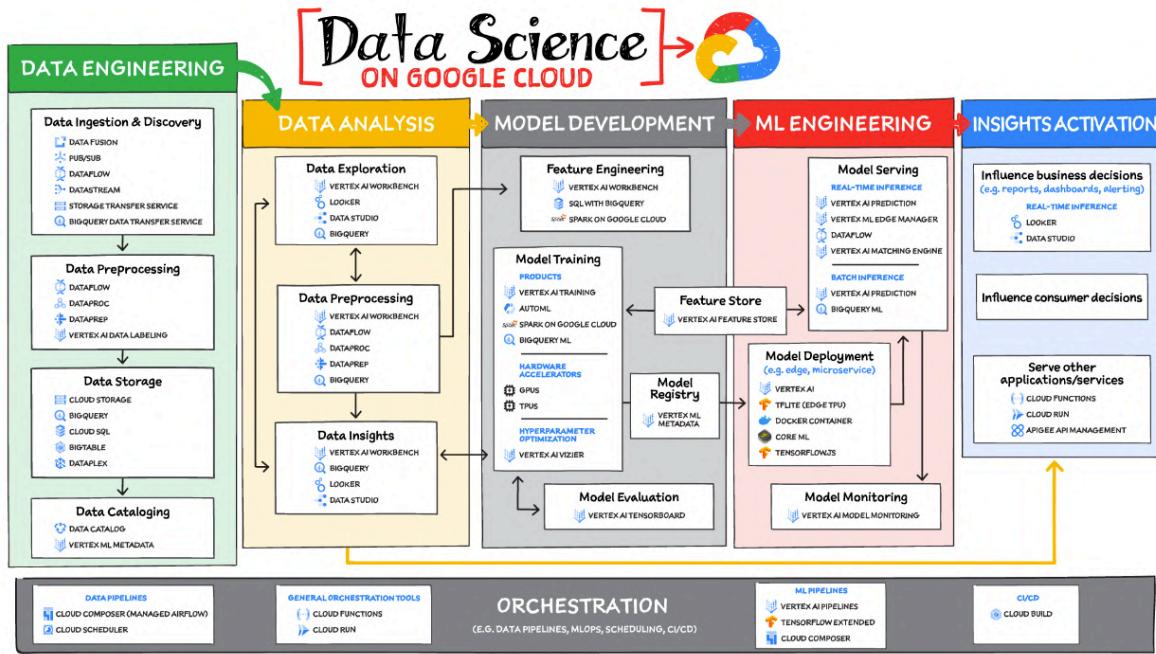


FIG. 2.24 : Data Science tools on Google Cloud Platform

Voici quelques services clés du cloud GCP que j'ai utilisé tout au long du projet. Dans la partie développement, j'introduirai d'autres services, mais voici les principaux éléments à retenir.

2.3.1.1.1 BigQuery (Fig. 2.25) est un service de data warehouse entièrement géré et hautement évolutif proposé par le cloud GCP. Il permet de stocker, d'interroger et d'analyser de grandes quantités de données structurées sans nécessiter de configuration ou de gestion complexe de l'infrastructure sous-jacente. Voici quelques caractéristiques clés de BigQuery :

- **Data warehouse** : BigQuery peut être utilisé comme une solution de data warehouse, permettant d'organiser et de structurer les données.
- **Moteur de requêtes** : BigQuery offre une puissance de calcul élevée pour exécuter rapidement des requêtes sur de grandes quantités de données.
- **Stockage** : Il permet de stocker de vastes volumes de données dans des tables structurées (tables matérialisées) ou d'utiliser des sources de données externes.

Pour plus de détails, vous pouvez consulter la documentation officielle de BigQuery⁸ [BQ-docs]



FIG. 2.25 : BigQuery - Logo

Note : Un *data warehouse* (*DWH*) est un système de gestion de données conçu pour l'analyse et le reporting. Il centralise, consolide et optimise les données pour les requêtes analytiques, facilitant ainsi les analyses approfondies des utilisateurs finaux. Il est essentiel pour la gestion et l'exploitation de données à grande échelle.

⁸ cloud.google.com/bigquery.

2.3.1.1.2 Google Cloud Storage (Fig. 2.26) est un service de stockage d'objets offert par le cloud GCP. Il permet de stocker et de récupérer de grandes quantités de données non structurées, telles que des fichiers, des images, des vidéos, etc. Voici quelques caractéristiques clés de Google Cloud Storage :

- **Scalabilité** : Il offre une évolutivité élevée pour stocker des données de différentes tailles.
- **Durabilité** : Google Cloud Storage assure une durabilité élevée des données, avec des mécanismes de réplication et de sauvegarde intégrés.
- **Accès facile et contrôlé** : Il permet de définir des politiques d'accès faciles et de contrôler les autorisations pour assurer la sécurité des données.



FIG. 2.26 : Google Cloud Storage - Logo

Pour plus de détails, vous pouvez consulter la documentation officielle de GCS⁹ [[GCS-docs](#)]

2.3.1.1.3 Looker Studio (Fig. 2.27a) est un outil de visualisation de données et de business intelligence proposé par le cloud GCP. Il permet d'explorer, d'analyser et de présenter les données de manière interactive, offrant des insights précieux pour la prise de décisions. Voici quelques caractéristiques clés de Looker :

- **Exploration des données** : Looker facilite l'exploration des données en permettant de créer des requêtes interactives pour obtenir des informations spécifiques.
- **Tableaux de bord interactifs** : Il permet de créer des tableaux de bord interactifs personnalisés, regroupant différentes visualisations pour une vue d'ensemble des données.
- **Intégration avec d'autres outils** : Looker peut être intégré avec d'autres outils et plateformes pour enrichir l'analyse des données, tels que BigQuery, Google Cloud Storage, etc.

Pour plus de détails, vous pouvez consulter la documentation officielle de Looker Studio¹⁰ [[Looker-docs](#)]



Looker Studio



Google
Sheets

(a) Looker Studio - Logo (b) Google Sheets - Logo

FIG. 2.27 : Logos of Looker Studio and Google Sheets

⁹ cloud.google.com/storage.

¹⁰ cloud.google.com/looker.

2.3.2 Docker

Docker (Fig. 2.28) est une plate-forme logicielle permettant la conception, le test et le déploiement rapides d'applications. Grâce à des conteneurs normalisés, Docker regroupe tous les éléments nécessaires au fonctionnement des applications (bibliothèques, outils système, code et environnement d'exécution). Cela facilite le déploiement et la mise à l'échelle des applications dans n'importe quel environnement, en garantissant l'exécution correcte du code. En somme, Docker simplifie le cycle de vie des applications en offrant une solution pratique et fiable pour leur déploiement.

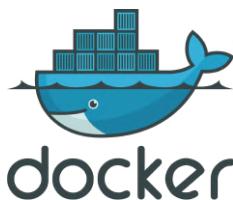


FIG. 2.28 : Docker - Logo

Docker se distingue des *Machines Virtuelles* en utilisant une approche de conteneurisation légère et efficace. Contrairement aux VM qui virtualisent tout un système d'exploitation, Docker utilise des conteneurs qui partagent le même noyau du système d'exploitation de l'hôte (voir Fig. 2.29). Les conteneurs sont plus légers, plus rapides à démarrer et utilisent efficacement les ressources. Il fonctionne en utilisant des images de conteneur préconfigurées pour créer des instances exécutables de conteneurs. Les conteneurs sont isolés les uns des autres et peuvent être gérés facilement via une interface en ligne de commande.

Note : Un conteneur Docker, également appelé *Docker Container*, est une instance exécutable d'une image Docker. Il s'agit d'un environnement isolé et portable qui permet de lancer une application. Une image Docker, ou *Docker Image*, quant à elle, est un modèle immuable contenant tous les fichiers et configurations nécessaires pour exécuter ladite application dans un conteneur.

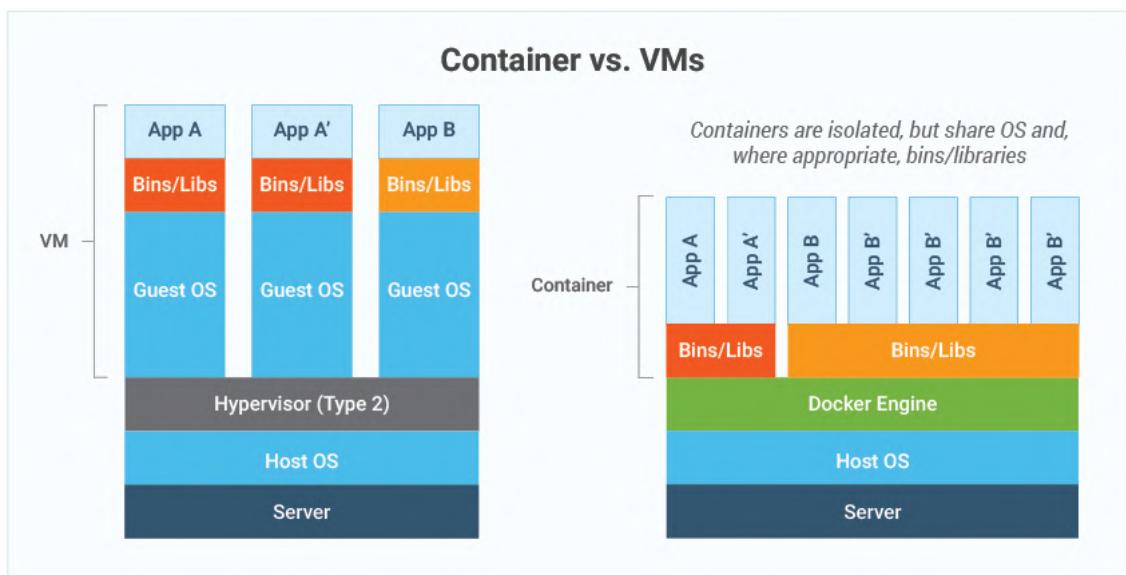


FIG. 2.29 : Containers Vs. VMs - Logo

Pour plus de détails, vous pouvez consulter la documentation officielle de Docker¹¹ [[Docker-docs](https://docs.docker.com/get-started)]

¹¹ <https://docs.docker.com/get-started>.

2.3.3 Kubernetes

Kubernetes - K8S (Fig. 2.30) est une plate-forme open-source extensible et portable pour la gestion de charges de travail et de services conteneurisés. Elle permet le déploiement, la gestion et la mise à l'échelle automatisés et efficaces d'applications conteneurisées. En fournissant des fonctionnalités avancées telles que la planification automatisée, la répartition de charge, la gestion des ressources et la tolérance aux pannes, Kubernetes simplifie la gestion des environnements complexes de conteneurs. Cela permet aux développeurs et aux administrateurs système de se concentrer sur le développement d'applications plutôt que sur les détails de l'infrastructure sous-jacente.



FIG. 2.30 : Kubernetes - Logo

Pour plus de détails sur le *Kubernetes*, vous pouvez consulter cette documentation¹² [[kubernetes](https://kubernetes.io/fr/docs/home/)]

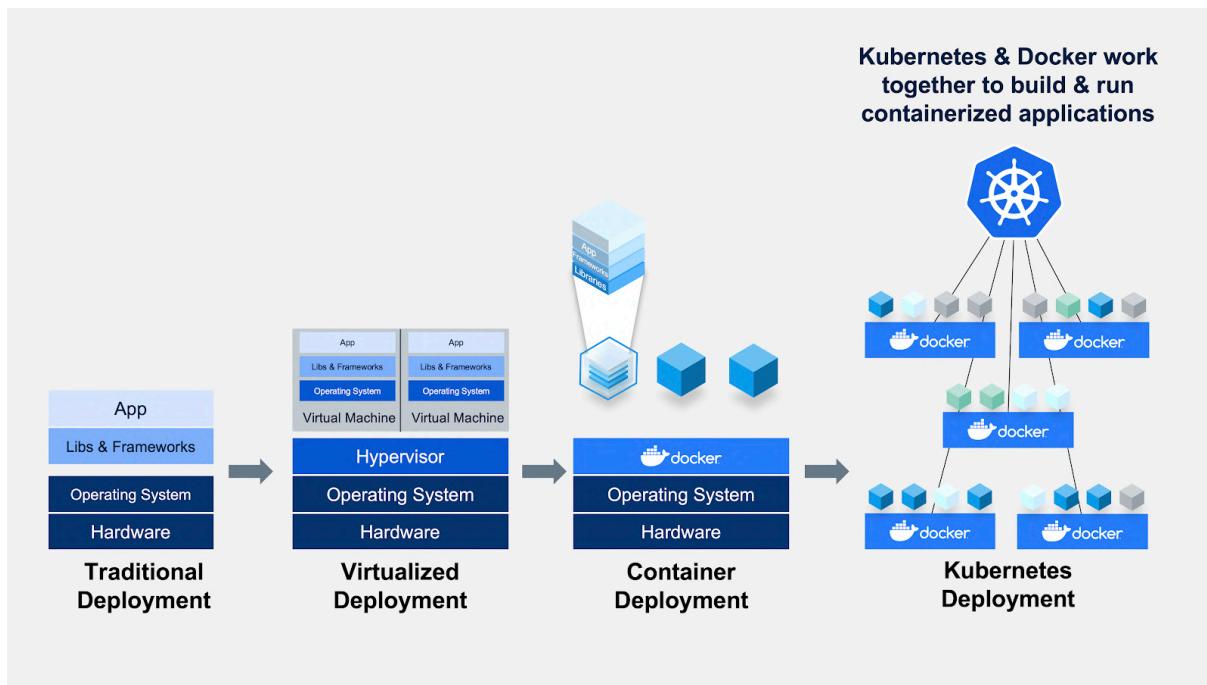


FIG. 2.31 : Containerized Application Deployment with Docker and Kubernetes

La figure 2.31 présente une architecture où Docker est utilisé pour créer des conteneurs, tandis que Kubernetes est utilisé pour gérer la création, la déploiement et la gestion de ces conteneurs. En effet, **Docker et Kubernetes offrent une solution complète pour le développement et le déploiement d'applications conteneurisées** de manière automatisée, efficace et fiable.

2.3.4 Apache Airflow

L'outil *Apache Airflow* (que j'appellerai tout au long de ce document **Airflow**) est une plateforme open source utilisée pour l'orchestration et la planification des tâches. Il offre une flexibilité et une

¹² kubernetes.io/fr/docs/home/.

extensibilité pour créer, ordonner et gérer des workflows (flux de travail) complexes. Pour plus de détails, vous pouvez consulter la documentation officielle d'Airflow¹³ [Apache-Airflow-docs]



FIG. 2.32 : Apache Airflow - Logo

Airflow est un outil capable d'effectuer l'*orchestration* et la *planification* (voir section 2.2.2.2.1) des ETL (voir section 2.2.2.2.2) de manière efficace.

Airflow adopte une approche basée sur les DAG (Directed Acyclic Graphs) (Fig. 2.33), où chaque tâche est représentée par un nœud dans le graphe et les dépendances entre les tâches sont représentées par des arêtes dirigées. Cette approche permet de définir de manière visuelle et déclarative les flux de travail, offrant ainsi une vue claire des dépendances et de l'ordre d'exécution. La figure suivante illustre un exemple de DAG pour un processus ETL :

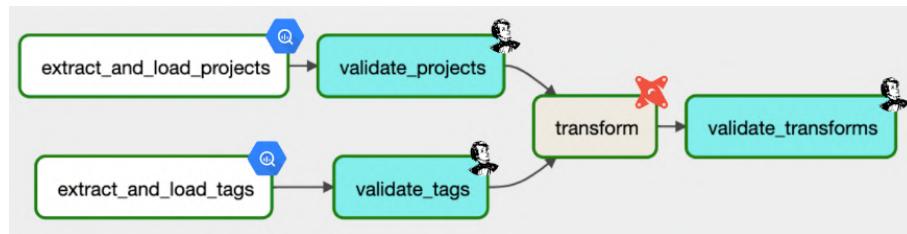


FIG. 2.33 : ETL Workflow DAG in Airflow

2.3.4.1 Les Opérateurs dans Airflow

Les opérateurs jouent un rôle essentiel dans Airflow en représentant les tâches individuelles au sein d'un workflow. Par exemple, dans la Figure 2.33, la tâche *transform* utilise l'opérateur DBT pour la transformation des données. Airflow propose plusieurs opérateurs intégrés, parmi lesquels :

- **BashOperator** : permet d'exécuter des commandes ou des scripts bash.
- **PythonOperator** : permet d'exécuter du code Python.
- **DockerOperator** : permet de créer des conteneurs et d'exécuter du code à l'intérieur.
- **KubernetesPodOperator** : permet d'exécuter des tâches dans des pods Kubernetes. Cet opérateur crée un pod Kubernetes et y exécute un conteneur spécifié..

Chaque opérateur a un objectif spécifique et peut être utilisé pour effectuer diverses actions dans un workflow Airflow. L'opérateur **KubernetesPodOperator** est conçu pour faciliter l'exécution de tâches à l'intérieur de conteneurs Docker dans un cluster Kubernetes. Il crée et gère un pod Kubernetes contenant un conteneur Docker basé sur une image spécifiée, exécute une commande à l'intérieur du conteneur et gère les résultats et les erreurs de la tâche.

Pour plus de détails sur le *KubernetesPodOperator*, vous pouvez consulter cette documentation¹⁴ [KubernetesPodOperator]

¹³ airflow.apache.org/docs/apache-airflow/stable/index.html.

¹⁴ cloud.google.com/composer/docs/how-to/using/using-kubernetes-pod-operator.

```

1 # Create the DAG object
2 dag = DAG('kubernetes_pod_example',
3            start_date=datetime(2023, 6, 28),
4            default_args=default_args,
5            owner='LABRIJI Saad')
6
7 # Define a task using the KubernetesPodOperator
8 task1 = KubernetesPodOperator(
9     dag=dag,                                     # Assign the DAG to the task
10    task_id='task1',                             # Unique ID for the task
11    name='example-pod',                          # Name of the pod
12    namespace='default',                         # Namespace where the pod will run
13    image='ubuntu:latest',                       # Docker image to use for the pod
14    cmd=['echo'],                                # Command to execute inside the pod
15    arguments=['Hello, Airflow!'],                # Arguments for the command
16    get_logs=True                               # Retrieve the logs of the pod
17 )
18
19 # Set the task dependencies (none in this case)
20 task1

```

Listing 2.2: KubernetesPodOperator Example in Airflow

Le code ci-dessous crée un *DAG* dans *Airflow* avec une tâche qui utilise le *KubernetesPodOperator* pour exécuter un conteneur *Docker* basé sur l'image 'ubuntu :latest' dans *Kubernetes*. La tâche utilise la commande 'echo' pour afficher un message et récupère les logs du pod. Cela permet d'automatiser l'exécution du conteneur dans Kubernetes.

2.3.5 GitHub

GitHub (Fig 2.34) est une plateforme web basée sur *Git*, un système de DVC (voir section 2.2.2.1.4 et figure 2.10). Il offre des fonctionnalités pour faciliter le développement collaboratif :

- **Merging** : Il permet d'avoir plusieurs lignes de développement indépendantes et de fusionner les modifications effectuées.
- **Pull Requests** : Cette fonctionnalité permet aux développeurs de proposer des modifications, de les examiner et de les incorporer dans la branche principale du projet.
- **Issue Tracking** : GitHub propose un système de suivi des problèmes (issues) qui permet de gérer les tâches, les bugs et les demandes de fonctionnalités.
- **Code Reviews** : GitHub facilite l'examen des modifications proposées par les membres de l'équipe et permet de fournir des commentaires pour améliorer la qualité du code.



FIG. 2.34 : Github - Logo

GitHub associe la puissance du contrôle de version distribué de **Git** avec des fonctionnalités de collaboration basées sur le web. Cela permet aux équipes de travailler efficacement ensemble, de gérer les modifications du code de manière efficace et de garantir la qualité du code dans un environnement de **production**.

2.4 Conclusion

En conclusion, ce deuxième chapitre a permis de présenter l'état de l'art des sciences de données en mettant l'accent sur les concepts, les techniques et les outils spécifiques aux tâches réalisées durant mon stage. J'ai examiné en détail les principales disciplines des sciences des données, ainsi que leurs synergies. De plus, j'ai introduit des concepts clés ainsi que les différents outils et techniques importantes et que j'ai eu l'occasion de découvrir et utiliser lors de mon stage à Cherfclub.

À la lumière de ces connaissances, nous sommes désormais prêts à passer à la prochaine étape du rapport. Dans la partie suivante, j'aborderai la phase d'analyse et conception de la solution proposée, en mettant en œuvre les concepts et les techniques des sciences des données pour résoudre les problématiques identifiées.

Chapitre 3

Analyse et conception

Ce chapitre a pour objectif de fournir une analyse approfondie et une conception détaillée de la solution proposée. Les points principaux abordés sont les suivants :

- Analyse des besoins de Chefclub en matière de Sciences des Données.
- Présentation détaillée de l'infrastructure de data de Chefclub.
- Analyse approfondie de l'existant, incluant les technologies utilisées, les fonctionnalités disponibles et les processus impliqués.
- Conception de la solution proposée pour répondre aux besoins identifiés et qui s'intègre dans l'infrastructure existante.

3.1 Introduction

Le troisième chapitre de ce rapport est consacré à l'analyse et à la conception de la solution proposée pour Chefclub. Dans un premier temps, je présenterai les besoins de Chefclub en matière de Science des Données. Ensuite, j'examinerai l'infrastructure data existante de Chefclub, en mettant en évidence les défis auxquels ils sont confrontés. À travers l'analyse des technologies utilisées, des fonctionnalités disponibles et des processus impliqués, j'identifierai les problématiques qui doivent être résolues en relation avec les besoins identifiés. Enfin, je présenterai les solutions conçues pour répondre à ces besoins, tout en s'intégrant harmonieusement dans l'infrastructure de données de Chefclub. Ce chapitre fournira une vision claire de la situation actuelle, ainsi qu'une feuille de route détaillée pour la mise en œuvre de la solution proposée.

3.2 Analyse des besoins

Dans cette section d'analyse des besoins, je vais examiner les différentes exigences de Chefclub ainsi que celles de son équipe data. Comprendre ces besoins est essentiel pour justifier la pertinence de notre projet. Je vais explorer les *objectifs de l'entreprise et comment ils se traduisent par des besoins spécifiques pour l'équipe data*. De plus, je vais identifier les acteurs clés du système qui seront impliqués dans la mise en œuvre de notre projet.

3.2.1 Besoins opérationnels

Chefclub a identifié plusieurs besoins fondamentaux qui motivent notre projet. Voici les principaux besoins de l'entreprise :

- **Centralisation des données** : Chefclub souhaite disposer de toutes les données des réseaux sociaux de Chefclub, appelés SMPs, accessibles aux différentes équipes internes. Cela comprend les équipes de la finance, les équipes de distribution de contenus, et autres. L'objectif est de regrouper toutes les informations pertinentes dans des feuilles de calcul Google Sheets, afin de faciliter l'accès et la collaboration entre les différentes équipes.
- **Prise de décisions basée sur les données** : Chefclub vise à adopter une approche data-driven en débloquant davantage d'informations sur le contenu publié sur les SMPs. L'entreprise souhaite utiliser les données disponibles pour prendre des décisions plus éclairées et orienter ses stratégies marketing et de distribution de contenus.
- **Maximisation des revenus et de la présence sur les SMPs** : L'utilisation du potentiel de la data est essentielle pour améliorer les performances de Chefclub sur les SMPs. L'entreprise souhaite exploiter les informations recueillies pour identifier de nouvelles opportunités de croissance, optimiser les performances et augmenter ses revenus.

3.2.2 Besoins de l'équipe Data

L'équipe data de Chefclub a également des besoins spécifiques qui doivent être pris en compte dans la conception de notre projet. Voici les principaux besoins de l'équipe :

- **Infrastructure stable** : L'équipe data rencontre des difficultés liées à la gestion des services locaux sur la machine locale. Elle souhaite migrer les services vers une infrastructure Cloud plus stable afin de libérer du temps et des ressources pour se concentrer sur des tâches à plus forte valeur ajoutée, telles que l'analyse des données et l'ajout de fonctionnalités.
- **Récupération régulière des données analytiques des chaînes YouTube** : L'équipe a besoin d'accéder régulièrement aux données analytiques des différentes chaînes YouTube de Chefclub. Cela permettra de suivre les performances, d'évaluer l'impact des stratégies et d'ajuster les actions en conséquence, dans le but d'informer les autres équipes.
- **Analyse approfondie des données** : L'équipe souhaite analyser en profondeur les données collectées à partir des SMPs, en mettant l'accent sur les données de Facebook, étant donné que cette plateforme représente une part importante des revenus. L'objectif est d'extraire des insights pertinents et des tendances significatives pour aider à la prise de décision stratégique.
- **Modèle de Machine Learning pour les posts Facebook** : L'équipe souhaite développer et d'intégrer un modèle d'apprentissage automatique qui permettra de prédire les performances futures des posts Facebook en se basant sur les données historiques. Cela permettra à Chefclub d'anticiper les tendances et de prendre des mesures proactives pour améliorer ses résultats.

3.2.3 Besoins non fonctionnels

En plus des besoins fonctionnels définis dans les sections 3.2.1 et 3.2.2, il est important de prendre en compte les besoins non fonctionnels (contraintes à respecter) afin de garantir que la solution développée répond aux exigences de Chefclub. Voici les contraintes identifiées :

- **Facilité de maintenance** : Le code doit être conçu en suivant les meilleures pratiques, afin de faciliter la maintenance et les mises à jour ultérieures.
- **Convivialité** : Les tableaux de bord et les interfaces utilisateur doivent être conviviaux et intuitifs, facilitant ainsi l'accès aux informations et la prise de décision.
- **Disponibilité** : Il est essentiel de garantir la disponibilité des données et des systèmes, afin de permettre une analyse en temps réel et une prise de décision continue.
- **Sécurité** : Des mesures de sécurité appropriées doivent être mises en place pour protéger les données contre les accès non autorisés, les violations de la confidentialité et les atteintes à l'intégrité.
- **Performance** : Le système doit être optimisé pour offrir des temps de réponse rapides et permettre une analyse efficace des données, même avec des volumes importants.
- **Fiabilité** : Il est essentiel de garantir la fiabilité du système en réduisant au minimum les temps d'arrêt et en mettant en place des mécanismes de récupération en cas de défaillance.
- **Scalabilité** : L'infrastructure data doit être capable de s'adapter à une croissance future en gérant efficacement les volumes croissants de données.

3.2.4 Les acteurs du système

Voici les acteurs du système impliqués dans le projet Chefclub :

- **Équipe data** : Les membres de l'équipe data de Chefclub sont responsables de la conception, du développement, de la mise en œuvre et de la maintenance de l'infrastructure data. Voici les membres de l'équipe data :
 - **PERDRIGEAT Max** *Data Manager and Business Analyst*
 - **LABRIJI Saad** *Data Engineering / Data Science intern*
- **Équipes internes** : Les différentes équipes (équipe de la distribution de contenu, de la finance, les dirigeants de l'entreprise, etc.) qui utiliseront les tableaux de bord interactifs et les données Google Sheets sont les utilisateurs finaux qui bénéficieront de ces données récupérées à partir des SMPs.
- **Équipes Tech Freelance** : soutient l'équipe data en prenant part aux décisions, en proposant des révisions de la conception et en apportant de l'aide en cas de blocage dans des fonctionnalités techniques.

3.3 Infrastructure Data de Chefclub

Dans cette section, je vais présenter en détail l'infrastructure data de Chefclub. **Il est important de souligner que les besoins identifiés dans la section 3.2 sont étroitement liés à cette infrastructure**, Leur alignement cohérent est crucial, car notre conception doit respecter et intégrer les solutions proposées avec l'infrastructure existante.

L'infrastructure data de Chefclub, repose sur une combinaison d'une **machine locale** (appelée *Cheesecake*) et des **services cloud de GCP**, formant une architecture *Hybrid* (combinaison de ressources cloud et locales). La machine locale utilisée est un serveur de stockage de type NAS, qui est une machine Linux spécialement conçue pour le stockage de données. L'infrastructure data de Chefclub comprend plusieurs services clés, notamment le **stockage des données**, la **collecte et l'ingestion des données**, le **traitement et la préparation des données**, ainsi que **l'analyse et la visualisation des données**.

3.3.1 Stockage des données

Pour répondre aux besoins de stockage des données, Chefclub utilise une combinaison de solutions, offrant ainsi une approche hybride pour répondre à différents besoins et scénarios d'utilisation. Les principales solutions de stockage utilisées sont les suivantes :

- **Google Cloud Storage (GCS)** : Chefclub utilise GCS comme solution de stockage pour héberger ses données brutes. En se référant à la figure 2.13, GCS joue le rôle en tant que zone de staging (contient les données brutes) où les réponses des API, généralement au format *CSV*, *AVRO* ou *JSON*, sont stockées en vue de transformations ultérieures. Il est également possible d'accéder directement aux données brutes via GCS à partir du Data Warehouse. Pour plus de détails sur ces formats de fichiers, vous pouvez consulter cet article ¹ [data-file-formats]

¹ blog.ippon.fr/2020/03/02/de-limportance-du-format-de-la-donnee-pratique-partie-2-2/.

- Entrepôt de données BQ (BigQuery)** : Chefclub utilise BigQuery (Fig. 2.25) comme entrepôt de données (DWH) pour stocker et organiser les données structurées. Dans notre cas, BigQuery contient les tables transformées et nettoyées. Il est directement connecté à Google Cloud Storage (GCS) et a accès à toutes les données brutes récupérées via les APIs des SMPs. En se référant à la figure 2.13 du dernier chapitre, BigQuery contient les données qui seront analysées et exploitées pour la réalisation de modèles ML.
- Serveur NAS et Amazon S3** : Chefclub dispose d'un serveur de stockage de type NAS local. Ce serveur NAS est utilisé pour stocker des données non structurées telles que le catalogue des vidéos produites par Chefclub sur les SMPs. Ces données nécessitent un accès rapide et sont fréquemment utilisées par les équipes de Chefclub. Le serveur NAS assure un stockage local pour garantir la disponibilité et la sécurité de ces données. De plus, pour assurer la disponibilité et la sécurité des données, Chefclub a mis en place une stratégie de réplication 3.1a) vers le service de stockage cloud AWS S3 (Fig. 3.1b). Cette réplication permet de créer une copie des données stockées dans le NAS, ce qui réduit les risques de perte de données et garantissant une continuité opérationnelle optimale. Pour plus de détails sur les serveurs NAS, vous pouvez consulter l'article suivant ²[NAS-server].

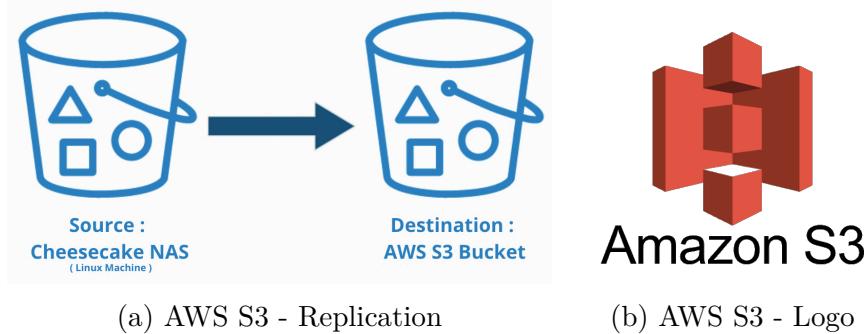


FIG. 3.1 : AWS S3 - Replication and Logo

- PostgreSQL dans Cloud SQL** : Chefclub utilise une base de données relationnelle PostgreSQL (voir Fig. 3.2) hébergée dans le service Cloud SQL de GCP. Cette base de données est utilisée pour stocker les données après leur transformation, provenant des SMPs tels que Facebook et Snapchat, récupérées par Chefclub.

RECIPES		
Recipes	+ Add	Change
<hr/>		
SOCIALMEDIA		
Comments	+ Add	Change
Facebook country days	+ Add	Change
Facebook page days	+ Add	Change
Facebook post days	+ Add	Change
Feed logs	+ Add	Change

FIG. 3.2 : Table Inventory from PostgreSQL Hosted on CloudSQL

Cette combinaison de solutions de stockage permet à Chefclub de tirer parti des avantages spécifiques de chaque solution, en garantissant la disponibilité, la performance et la sécurité des données en fonction de leurs besoins et de leurs cas d'utilisation spécifiques.

²1min30.com/dictionnaire-du-web/serveur-nas

3.3.2 Collecte et ingestion des données

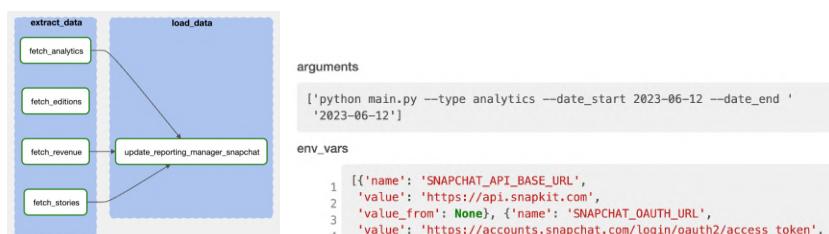
La collecte et l'ingestion des données sont des étapes cruciales dans l'infrastructure data de Chefclub. En effet, on utilise les outils et services suivants pour assurer ces fonctionnalités :

- **Airbyte** : Chefclub utilise les connecteurs Airbyte (voir Fig. 3.3) pour faciliter l'intégration des données provenant de sources externes. Airbyte est une plateforme open source qui offre une bibliothèque de connecteurs prêts à l'emploi pour diverses sources de données, telles que des bases de données, des API, des services cloud, et bien d'autres encore. Grâce à ces connecteurs, Chefclub peut extraire les données des SMPs et les charger dans BigQuery, en utilisant GCS comme zone de staging. Cela permet à Chefclub d'effectuer des transformations ultérieures sur les données brutes stockées dans GCS avant de les charger dans le DWH BigQuery.



FIG. 3.3 : Airbyte - Logo

- **Python personnalisé + Airflow** : Chefclub a développé en interne une solution appelée *Reporting Manager* qui gère les opérations d'extraction et de chargement EL (Extract, Load). Elle agit en tant qu'API (voir Fig. 3.4b) qui récupère les données des SMPs sélectionnés, les sérialise et les charge dans le stockage cloud (Fig. 2.26). Cette opération est déclenchée et planifiée grâce à Airflow. Les tâches de la figure 3.4a, identifiées sous le nom de *fetch*, utilisent le code personnalisé *Reporting Manager* pour effectuer ces opérations. L'objectif du Reporting Manager est de récupérer les autres données non prises en charge par les connecteurs d'Airbyte.



(a) Snapchat DAG (b) Utilizing the Reporting Manager API

FIG. 3.4 : Workflow and Fetch Analytics data Task

La combinaison d'*Airflow* et des connecteurs *Airbyte* avec le *Code Custom (Reporting Manager)* permet à Chefclub d'automatiser et de simplifier le processus de collecte et d'intégration des données. Cela permet d'obtenir efficacement et de manière fiable des données provenant de diverses sources (principalement les SMPs), garantissant ainsi la disponibilité des informations nécessaires pour les étapes ultérieures de traitement et d'analyse.

Note : Au début de mon stage, *Airbyte* et *Airflow* résidaient sur le serveur local NAS, mais nous les avons migrés vers le Cloud GCP. Les raisons de cette migration seront abordées dans la section 3.3.5.

3.3.3 Traitement et Préparation des données

Une fois les données collectées et intégrées, Chefclub les traite et les prépare pour l'analyse ultérieure. Dans ce domaine, Chefclub s'appuie principalement sur le service BQ Datawarehouse de GCP (Fig. 2.25), chefclub utilise aussi du code custom (Reporting Manager) pour faire l'ingestion et la préparation des données stockés sur le GCS.

- **BQ DDL + Airflow** : Chefclub utilise BQ non seulement comme un entrepôt de données (DWH), mais également comme une puissante plateforme de traitement et de préparation des données. L'un des avantages de BQ réside dans sa flexibilité SQL, qui permet d'effectuer des opérations de langage de définition de données (DDL - Data Definition Language) et de langage de manipulation de données (DML - Data Manipulation Language), ainsi que la création de tables matérialisées. En complément de BQ, Chefclub utilise Airflow pour planifier et orchestrer ces opérations. Airflow dispose d'opérateurs spécifiques pour BigQuery, ce qui facilite l'exécution de tâches programmées pour la transformation et la préparation des données. Cela permet d'optimiser la préparation des données pour les étapes ultérieures d'analyse et de visualisation. La figure suivante illustre un exemple d'utilisation d'Airflow qui utilise un opérateur BigQuery pour lancer des requêtes SQL de DDL afin de créer des tables temporaires et les remplir. Ensuite, une requête upsert est utilisée pour joindre la table matérialisée avec la table temporaire.

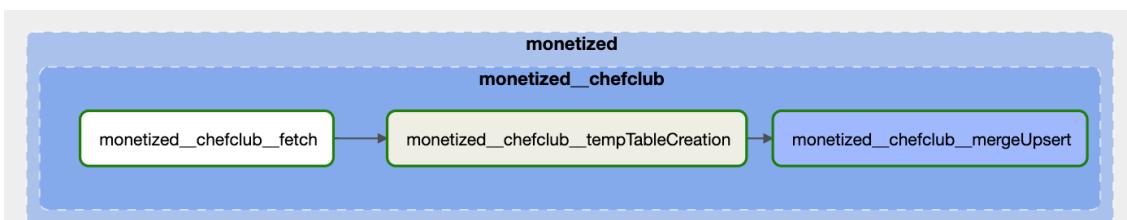


FIG. 3.5 : Data Preparation in BigQuery Using DDL with Airflow

Note : le *DDL* est utilisé pour créer et organiser la structure des bases de données. Le *DML* est utilisé pour manipuler les données à l'intérieur de ces bases.

La figure suivante (Fig. 3.6) montre un exemple de requête *upsert* planifiée avec Airflow pour remplir les tables internes de Chefclub et éviter les doublons de données.

```

1 ✓MERGE
2   `chefclub-158403.social_media.dim_page_perf_day_yt_monetized_` T
3   USING
4   `chefclub-158403.airflow_tables_temp.temp--chefclub--time-based-day-monetized--2023-06-18` S
5   ON
6   T.RUN_START_DATE = S.RUN_START_DATE AND T.RUN_CHANNEL_ID = S.RUN_CHANNEL_ID
7   WHEN MATCHED THEN
8 ✓UPDATE SET
9     T.monetizedPlaybacks = S.monetizedPlaybacks,
10    T.playbackBasedCpm = S.playbackBasedCpm,
11    T.adImpressions = S.adImpressions,
12    T.cpm = S.cpm
13    WHEN NOT MATCHED BY TARGET THEN
14    INSERT
15      (FETCHED_AT_UTC , RUN_CHANNEL_ID, playbackBasedCpm ,adImpressions, day, cpm )
16 ✓VALUES
17      ( S.FETCHED_AT_UTC, S.RUN_CHANNEL_ID, S.playbackBasedCpm, S.adImpressions, S.day , S.cpm );

```

FIG. 3.6 : Example of Upsert Queries in BigQuery planified with Airflow

Note : le *Upserting* est une opération combinant l'*insertion* et la *mise à jour* dans une table. Cela permet de mettre à jour les enregistrements existants et d'insérer de nouveaux enregistrements si nécessaire. Ceci permet d'assurer la synchronisation des données.

- **Reporting Manager + Airflow :** Après avoir récupéré les données sérialisées depuis le GCS (voir Section 3.3.2, page 53), celles-ci sont transformées avant d'être intégrées dans PostgreSQL, qui se trouve dans Cloud SQL (Fig. 3.2). L'exemple d'utilisation du Reporting Manager pour la transformation et l'intégration dans PostgreSQL est illustré dans la figure suivante (voir Fig. 3.7).

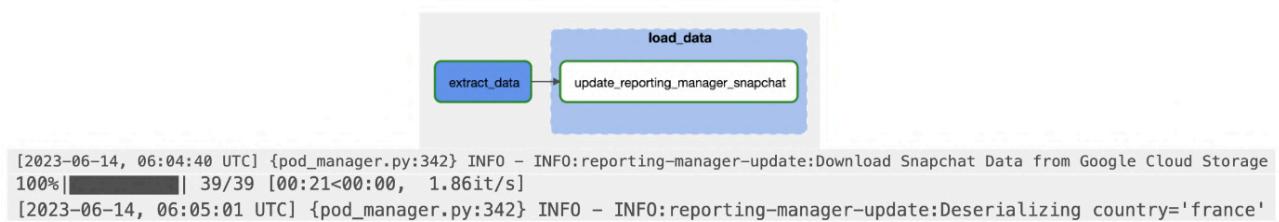


FIG. 3.7 : Data Transformation Flow : Transforming and Uploading Data from GCS to CloudSQL

Ainsi, grâce à la combinaison du Data Warehouse BQ, du code personnalisé Reporting Manager et d'Airflow, Chefclub optimise le traitement et la préparation des données en planifiant et en orchestrant les tâches de manière efficace.

3.3.4 Analyse et visualisation des données

Une fois les données préparées, Chefclub les analyse et les visualise pour obtenir des informations exploitables. Voici les outils et services clés utilisés par Chefclub dans cette phase :

- **Google Sheets et Connected Sheets :** Chefclub utilise Google Sheets (Fig. 2.27b) et sa fonction Connected Sheets pour partager des tableaux avec les équipes. Cela permet une connectivité en temps réel avec BQ pour récupérer les dernières données et effectuer des analyses en temps réel. Google Sheets offre une grande flexibilité avec l'utilisation de formules pour les calculs, l'ajout de tableaux et la génération simplifiée de graphiques. Son interface conviviale en fait un outil idéal pour les équipes de la finance et de la distribution de contenu.
- **Looker Studio :** Chefclub utilise Looker Studio (Fig. 2.27a), une plateforme de visualisation de données, pour créer des tableaux de bord interactifs et des visualisations personnalisées. Grâce à Looker Studio connecté à BigQuery DWH, Chefclub peut accéder aux données en temps réel et leur donner vie en les présentant de manière visuelle et facilement compréhensible. Les différentes équipes de Chefclub ont la possibilité d'explorer les données, de générer des rapports et d'obtenir des informations précieuses grâce à des visualisations interactives et fluides. Un exemple concret d'utilisation de Looker Studio chez Chefclub est présenté dans la figure suivante (voir Fig. 3.8).

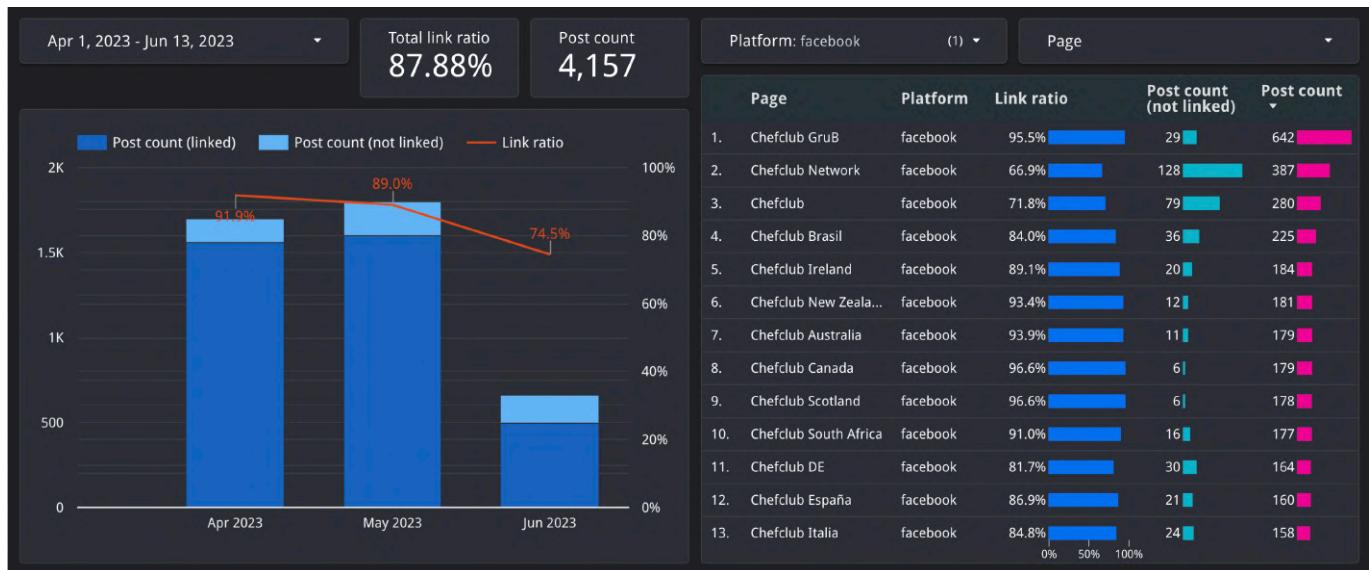


FIG. 3.8 : Chefclub Looker Dashboard Example

Il est important de noter que la solution que nous développons est étroitement liée à l'infrastructure de données de Chefclub, qui doit être stable. Dans cette sous-section 3.3.5, je détaillerai les choix effectués concernant l'infrastructure data afin d'améliorer sa stabilité, ainsi que l'impact de ces choix. Une fois que les besoins de stabilité seront satisfaits, je pourrai ensuite définir la conception qui s'intègre à cette nouvelle infrastructure.

3.3.5 Migration de Airbyte et Airflow vers GCP

3.3.5.1 Limitations du système actuel

L'infrastructure data de Chefclub présente des limitations significatives découlant de l'utilisation inappropriée du NAS. Ce dernier, qui est principalement destiné au stockage, est utilisé pour exécuter des tâches volumineuses dans les DAG Airflow.

Voici une liste des limitations identifiées, accompagnées des besoins en sciences de données correspondants :

- **Problèmes de performance :**
 - Les DAG volumineux rencontrent des problèmes de performance en raison des ressources de calcul limitées du NAS.
 - Les tâches échouent fréquemment, entraînant des retards dans la récupération des données, parfois pendant plusieurs heures.
- **Version obsolète d'Airflow :**
 - L'utilisation de la version 1.7 d'Airflow nécessite une maintenance et des optimisations importantes.
 - La mise à niveau vers la version 2.6 d'Airflow permettrait de bénéficier de fonctionnalités améliorées de surveillance et d'une interface utilisateur plus conviviale.
- **Temps d'ingénierie consacré à la maintenance :**

- L'équipe data, qui est en effectif réduit, consacre énormément de temps à résoudre les erreurs dans les DAG. La complexité de l'ancienne structure des DAG rend difficile la compréhension, la maintenance et la personnalisation des workflows.
- La personnalisation des DAG est très difficile en raison de la complexité de l'ancienne structure, alors que de meilleures pratiques pour le développement des DAG Airflow existent désormais. L'adoption de ces meilleures pratiques faciliterait le développement, la maintenance et l'évolutivité des workflows.

3.3.5.2 Décisions de migration

Suite à des discussions entre les équipes et les dirigeants de Chefclub, et en prenant en compte les arguments mentionnés dans la sous-section 3.3.5.1, il a été décidé de migrer vers la version maintenue par Google Cloud Platform (GCP) pour *Airbyte* et *Airflow*. Cette décision revêt une importance capitale, comme le montre la figure 3.9, qui met en évidence les principaux critères pris en compte lors du choix d'amélioration de l'infrastructure data de Chefclub.

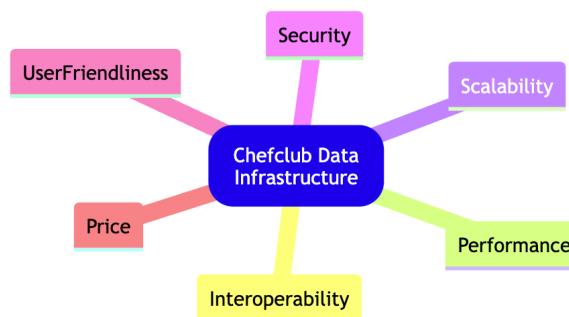


FIG. 3.9 : Principaux critères de sélection des technologies et des outils pour l'infrastructure de données de Chefclub

3.3.5.3 Avantages de la migration

La migration vers GCP pour *Airflow* et *Airbyte* présente plusieurs avantages importants :

- 1. Performances améliorées et gestion des ETL complexes** : Airflow local a des difficultés à gérer les ETL utilisant des images dockerisées, tandis qu'Airflow dans GCP avec Kubernetes offre de meilleures performances et une meilleure évolutivité. Chefclub dispose déjà d'un cluster dans GCP utilisé par d'autres services, ce qui signifie qu'il n'y a pas eu d'impact sur la facturation en échange des fonctionnalités débloquées.
- 2. Évolutivité et gestion simplifiée des ressources** : GCP et Kubernetes permettent de gérer efficacement les charges de travail en ajustant facilement la capacité des clusters pour optimiser l'utilisation des ressources. Kubernetes offre également des capacités de répartition de charge (*load balancing*) sur le cluster, garantissant ainsi une meilleure gestion de celui-ci.
- 3. Intégration native avec les services GCP** : Airflow dans GCP s'intègre facilement avec les services GCP tels que BigQuery DWH, ce qui simplifie la création de pipelines d'ETL complexes sans se soucier de la gestion des autorisations.

4. **Gestion des erreurs et surveillance améliorées** : Airflow dans GCP propose des fonctionnalités avancées pour la gestion des erreurs, la surveillance des tâches et la génération d'alertes en cas d'échec. De plus, le monitoring de GCP est plus avancé et offre de meilleurs rapports en cas de problèmes.
5. **Sécurité et fiabilité** : GCP propose des mesures de sécurité robustes pour protéger les flux de travail Airflow et les données, avec des fonctionnalités de stockage sécurisé et de gestion des identités et des accès (IAM).
6. **Service managé par Google et mises à jour faciles** : les services Airflow et Airbyte sont gérés par Google, ce qui signifie que les correctifs et les mises à jour sont pris en charge.

En résumé, migrer vers *Airbyte* et *Airflow* dans GCP avec *Kubernetes* offre des performances améliorées, une évolutivité accrue, une intégration native avec les services GCP, une gestion avancée des erreurs, une sécurité renforcée et une facilité de déploiement, tout en maintenant une facturation constante, conforme aux critères définis dans la figure 3.9.

Pour plus de détails sur la migration vers le Cloud GCP, vous pouvez consulter la documentation officielle ³ [[migration-to-GCP](https://cloud.google.com/architecture/migration-to-gcp-getting-started/)]

3.3.5.4 Résumé de l'infrastructure data de Chefclub

La Figure suivante (Fig. 3.10) résume l'infrastructure de data de Chefclub après migration, et illustre les différentes connexions entre les services.

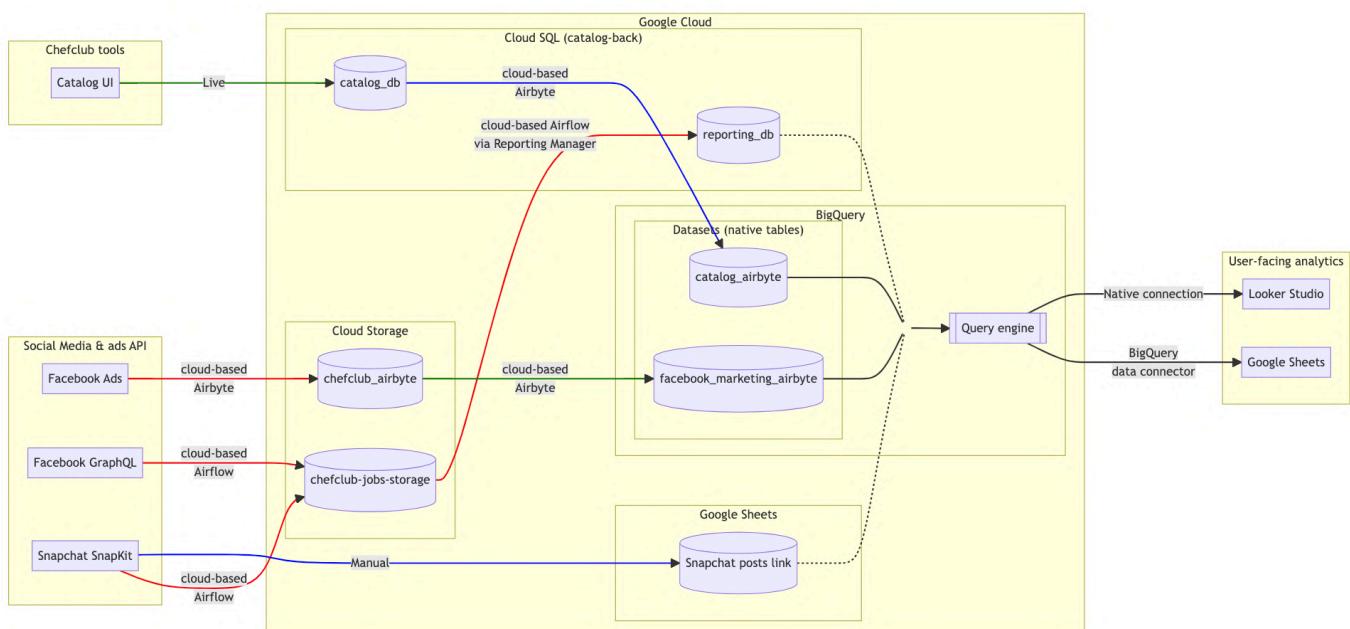


FIG. 3.10 : Overview of Chefclub's Data Infrastructure (Post-Migration) - June 2023

³ cloud.google.com/architecture/migration-to-gcp-getting-started/.

3.4 Conception de la solution

L'objectif de cette section est de présenter la conception de la solution répondant aux besoins énoncés dans la section précédente 3.2, tout en prenant en considération les besoins non fonctionnels mentionnés dans la sous-section 3.2.3.

3.4.1 Conception, Workflow et Présentation de la Solution

Maintenant que l'infrastructure répond aux besoins de l'équipe data en termes de stabilité, je vais aborder la conception de la solution proposée pour répondre aux besoins définis dans la section 3.2. Cette conception sera définie en fonction des trois projets/tâches majeurs sur lesquels j'ai eu l'occasion de travailler, qui sont :

1. **Conception de la solution Data Engineering/DataOps** : La solution proposée consiste à récupérer régulièrement les données analytiques des chaînes YouTube. Elle se compose de trois parties principales : l'extraction des données à partir de l'API YT Analytics, le chargement et la transformation des données dans Google Cloud Storage et Google BigQuery, et enfin l'utilisation des données pour générer des rapports et des analyses. Cette solution utilise des outils tels que *Python*, *Airflow*, *Docker*, *Kubernetes*, *Google Cloud Storage*, *Google BigQuery* et *Looker Studio*. Elle permet d'assurer une récupération régulière des données, garantissant ainsi des rapports et des analyses basés sur des données actualisées et cohérentes.
2. **Workflow et outils pour l'Analyse de données** : Ce workflow proposé permet d'analyser les données des publications sur Facebook pour Chefclub. Il utilise des outils tels que *Python*, *Pandas*, *Jupyter Notebook*, *Plotly*, *SQL*, *Notion* et *Github*. Cette solution aide à explorer les données, à découvrir des insights et à optimiser les performances de la plateforme.
3. **Conception de la solution Model Training/Model Deployment** : La solution permet de prédire les performances futures des publications sur Facebook pour Chefclub. Elle utilise principalement *Python*, *Airflow*, *Docker*, *Kubernetes*, *Google Cloud Storage*, *Google BigQuery*, *Scikit-learn*, *Cloud Function* et l'API *Slack*. Les étapes comprennent l'extraction des données, l'entraînement du modèle, la sauvegarde du modèle, l'inférence et la mise à disposition des résultats. Cela permet aux utilisateurs d'obtenir une estimation des performances attendues des publications via *Slack*.

3.4.1.1 Conception de la solution DATA Engineering/DataOps

3.4.1.1.1 Vue d'ensemble de la solution

La Figure 3.11 présente une vue d'ensemble de la solution proposée.

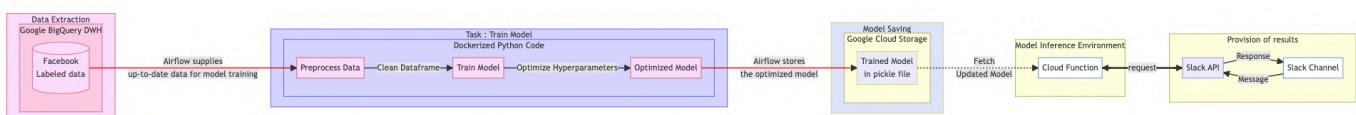


FIG. 3.11 : YouTube Analytics Data Retrieval Solution

3.4.1.1.2 Outils et technologies utilisés

La solution utilise les outils et technologies suivants :

1. **YT Analytics API** : Cette API, fournie par GCP, permet d'accéder aux données analytiques historiques des chaînes YouTube.
2. **Python** : Langage de programmation utilisé pour le développement de notre processus d'ELT (Extract, Load, Transform).
3. **Airflow** : Plateforme utilisée pour orchestrer et planifier l'exécution de nos jobs.
4. **Docker** : Permet de containeriser le code responsable de l'authentification et de l'orchestration depuis l'API.
5. **Kubernetes** : Plateforme d'orchestration de conteneurs utilisée pour assurer une gestion optimale des conteneurs créés pour récupérer les données.
6. **Google Cloud Storage** : Zone de staging où les données récupérées depuis l'API sont stockées.
7. **Google BigQuery** : Permet de traiter et de transformer les données, et contient les tables matérialisées.
8. **Looker Studio** : Plateforme d'analyse de données et de création de rapports utilisée pour visualiser les données.
9. **SQL** : Langage utilisé pour effectuer des transformations sur les données et les tables dans BigQuery.

3.4.1.1.3 Description de la solution

La solution proposée vise à répondre au besoin de récupération régulière des données analytiques des chaînes YouTube. Pour cela, nous avons conçu une solution en trois parties principales, intégrant les outils et technologies mentionnés ci-dessus. La Figure 3.11 illustre cette architecture.

- **Extract** : Cette partie consiste à extraire les données de la YT Analytics API à l'aide d'un code Python containerisé avec Docker. Ce code gère l'authentification et récupère les données au format JSON depuis l'API. Ensuite, les données sont transformées en format AVRO. Airflow prend en charge l'orchestration et la planification de cette étape, garantissant une exécution régulière et fiable de l'extraction des données.
- **Load and Transform** : Une fois les données extraites, elles sont stockées au format AVRO dans Google Cloud Storage, qui sert de zone de staging. À l'aide d'Airflow, un processus de mise à jour et d'insertion (Upsert) est effectué. Une table temporaire est créée dans Google BigQuery pour stocker les données récupérées quotidiennement, puis celles-ci sont fusionnées (upsert) avec la table matérialisée utilisée en production. Cela garantit que les données sont toujours à jour et cohérentes.
- **Serve** : Dans cette partie, les données sont prêtes à être utilisées pour générer des rapports et réaliser des analyses. Les données sont extraites de la table matérialisée dans Google BigQuery. Chaque jour, après la récupération des données stockées dans BigQuery, des requêtes SQL sont exécutées sur les tables matérialisées pour s'adapter aux différents rapports générés sur Google Sheets. De plus, ces données sont utilisées pour alimenter les graphiques interactifs dans Looker.

Studio. Ainsi, les utilisateurs ont la possibilité d'explorer et de visualiser les données de manière conviviale et efficace.

En résumé, la solution repose sur l'utilisation de l'API YT Analytics pour extraire les données, Python et Docker pour la gestion du code, Airflow pour l'orchestration des tâches, Google Cloud Storage comme zone de staging, Google BigQuery pour le traitement et la transformation des données, et enfin Looker Studio pour l'analyse et la visualisation des données. Cette architecture assure une récupération régulière des données, garantissant ainsi des rapports et des analyses basés sur des données actualisées et cohérentes. tout en assurant l'intégration avec l'infra existante.

3.4.1.1.4 Justification des choix

Voici les explications concernant les choix effectués dans la conception de la solution proposée (Figure 3.11) :

- **Utilisation de l'API YT Analytics** : L'utilisation de l'API YT Analytics est justifiée par le fait qu'elle est la seule API fournie par Google permettant d'accéder aux données analytiques historiques des chaînes YouTube. Elle est sécurisée par le mécanisme OAuth pour garantir la confidentialité et l'intégrité des données.
- **Transformation de la réponse JSON en AVRO** : La transformation des données JSON en format AVRO est choisie pour plusieurs raisons. Tout d'abord, AVRO est un format tabulaire compatible avec les outils et technologies que nous utilisons, notamment Google BigQuery. De plus, il offre une bonne interprétabilité des données grâce à la possibilité de stocker les métadonnées sur les informations. AVRO est également un format compact et rapide, ce qui facilite le stockage et le traitement ultérieur des données. Enfin, la compatibilité avec BigQuery simplifie les opérations de chargement et de transformation des données.
- **Création d'un code Python containerisé pour la récupération de données** : Le choix d'utiliser un code Python containerisé avec Docker garantit la portabilité et la reproductibilité de notre solution. En encapsulant le code et ses dépendances dans un conteneur, nous pouvons le déployer et l'exécuter facilement sur différentes plateformes, indépendamment des configurations spécifiques de l'environnement. Étant donné l'absence de connecteurs open-source sur Airbyte permettant d'accéder directement aux données de l'API YT Analytics, il était nécessaire de développer notre propre code pour récupérer les données.
- **Utilisation de l'Upsert et la création de tables temporaires** : L'utilisation de l'Upsert entre la table temporaire et la table principale dans Google BigQuery garantit la cohérence des données. En créant une table temporaire pour stocker les données récupérées quotidiennement, nous évitons les conflits de mise à jour directe sur la table principale. Cela nous permet de fusionner les données de manière contrôlée et de maintenir l'intégrité des données existantes. Les tables temporaires simplifient également le processus de chargement et de transformation des données en offrant une étape intermédiaire pour préparer les données avant de les insérer dans la table principale.
- **Utilisation de GitHub pour la gestion du code** : Afin de faciliter la collaboration au sein de l'équipe et de gérer efficacement le code, nous avons utilisé GitHub. GitHub nous a permis de versionner le code et de travailler simultanément sur différentes branches. Cela nous a également offert un historique complet des modifications et a facilité la résolution des conflits lors des mises à jour du code.

Ces choix de conception ont été faits en prenant en compte les besoins spécifiques du projet, tels que l'accès aux données analytiques de YouTube, la sécurité des données, la compatibilité avec les outils existants, la portabilité de la solution et la cohérence des données.

3.4.1.2 Workflow et outils pour la Data Analysis

3.4.1.2.1 Vue d'ensemble Workflow de l'analyse des données Facebook

Dans cette section, je présenterai le workflow utilisé pour répondre aux besoins de Chefclub en matière d'analyse des données des publications sur Facebook. Étant donné que Facebook est la plateforme générant le plus de revenus pour Chefclub sur les réseaux sociaux, l'équipe data a pour objectif d'explorer ces données afin de découvrir des insights. La Figure 3.12 présente une vue d'ensemble du workflow utilisé pour l'analyse des données des publications sur Facebook.

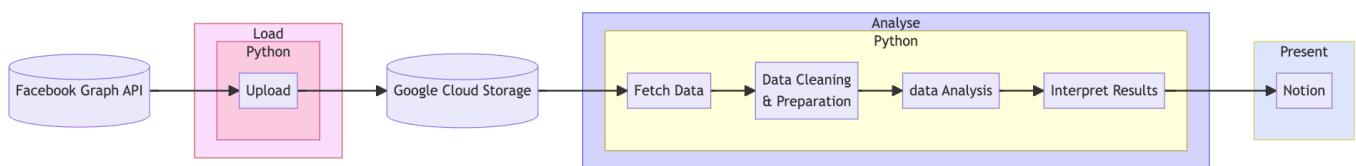


FIG. 3.12 : Data Analysis Workflow for Facebook Posts

3.4.1.2.2 Outils et technologies utilisés

La solution repose sur les outils et technologies suivants :

1. **Python** : J'utilise le langage de programmation Python pour le traitement et l'analyse des données tabulaires.
2. **Pandas** : Cette bibliothèque Python est utilisée pour le traitement des DataFrames, ce qui facilite les opérations sur les données.
3. **NumPy** : J'utilise la bibliothèque NumPy pour effectuer des opérations mathématiques et manipuler des tableaux multidimensionnels.
4. **Jupyter Notebook** : C'est notre environnement de travail pour l'analyse des données, offrant une interface interactive pour l'exploration et la visualisation des données.
5. **Plotly** : Cette bibliothèque Python nous permet de générer des graphiques interactifs et est utilisée pour la visualisation des données.
6. **SQL** : J'utilise le langage SQL pour effectuer des agrégations et des transformations sur les données tabulaires.
7. **Notion** : J'utilise Notion pour regrouper et organiser les informations pertinentes obtenues lors de l'analyse des données.
8. **GitHub** : J'utilise GitHub pour sauvegarder, versionner et partager les notebooks utilisés par l'équipe data.

3.4.1.2.3 Justification des choix

Pour cette solution, nous utilisons GitHub pour collaborer efficacement et assurer la sécurité, la sauvegarde et le partage des notebooks. Notre environnement de développement utilise un environnement virtuel (venv) avec Python, SQL, pandas, et NumPy pour le traitement et la manipulation des données. Jupyter Notebook est également installé dans cet environnement virtuel, offrant une plateforme pratique pour l'exploration et l'analyse interactives des données. Pour la visualisation, nous utilisons Plotly en raison de sa flexibilité et de ses fonctionnalités avancées pour créer des graphiques interactifs et expressifs. Voici les explications concernant les choix effectués dans la conception de la solution proposée (Figure 3.12) :

- **Environnement de développement local** : Pour permettre une exploration efficace des données, j'ai choisi de mettre en place un environnement de développement local. J'ai utilisé un environnement virtuel (venv) qui intègre Python, Jupyter Notebook, Pandas, NumPy et Plotly. Cela nous a permis de disposer d'un environnement de travail cohérent, prêt à être utilisé pour l'analyse et la manipulation des données.
- **Plotly pour les graphes interactifs** : L'objectif de notre analyse était de découvrir des insights à partir des données. J'ai donc choisi d'utiliser Plotly pour la visualisation des données. Plotly offre des fonctionnalités avancées pour créer des graphiques interactifs et expressifs, ce qui nous a permis d'explorer les données sous différents angles et de mettre en évidence les tendances et les relations.
- **Stockage des données dans Google Cloud Storage (GCS)** : Pour organiser les données et assurer un suivi de l'historique, j'ai récupéré les données du test A/B de Meta et les ai stockées dans Google Cloud Storage (GCS). Cela nous a permis de centraliser les données et de les rendre facilement accessibles pour les analyses futures. GCS offre également des fonctionnalités de sécurité et de sauvegarde, garantissant la disponibilité et l'intégrité des données.
- **Utilisation de GitHub pour la gestion du code** : Afin de faciliter la collaboration au sein de l'équipe et de gérer efficacement le code, nous avons utilisé GitHub. GitHub nous a permis de versionner le code et de travailler simultanément sur différentes branches. Cela nous a également offert un historique complet des modifications et a facilité la résolution des conflits lors des mises à jour du code.

Ces choix ont permis à l'équipe data d'explorer les données de manière efficace, en utilisant des outils adaptés à l'analyse et à la visualisation. L'utilisation de l'environnement de développement local, de Plotly pour les graphes interactifs, de GCS pour le stockage des données et de GitHub pour la gestion du code a facilité la collaboration et la traçabilité des travaux effectués. Cela a permis à l'équipe d'obtenir des insights précieux à partir des données de Chefclub sur Facebook et d'optimiser les performances de la plateforme.

3.4.1.3 Conception de la solution Model Training/Model Deployment

3.4.1.3.1 Vue d'ensemble de la solution

La Figure 3.13 présente une vue d'ensemble de la solution proposée.

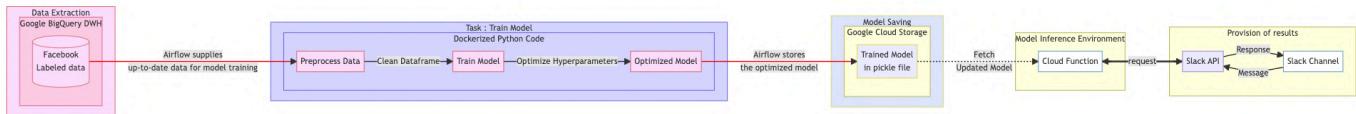


FIG. 3.13 : Model Training/Model Deployment Solution

La Figure 3.14 suivante montre le diagramme d’interaction qui complète la conception de la Figure 3.13. Il illustre le mécanisme par lequel un utilisateur demande l’inférence du modèle via Slack.

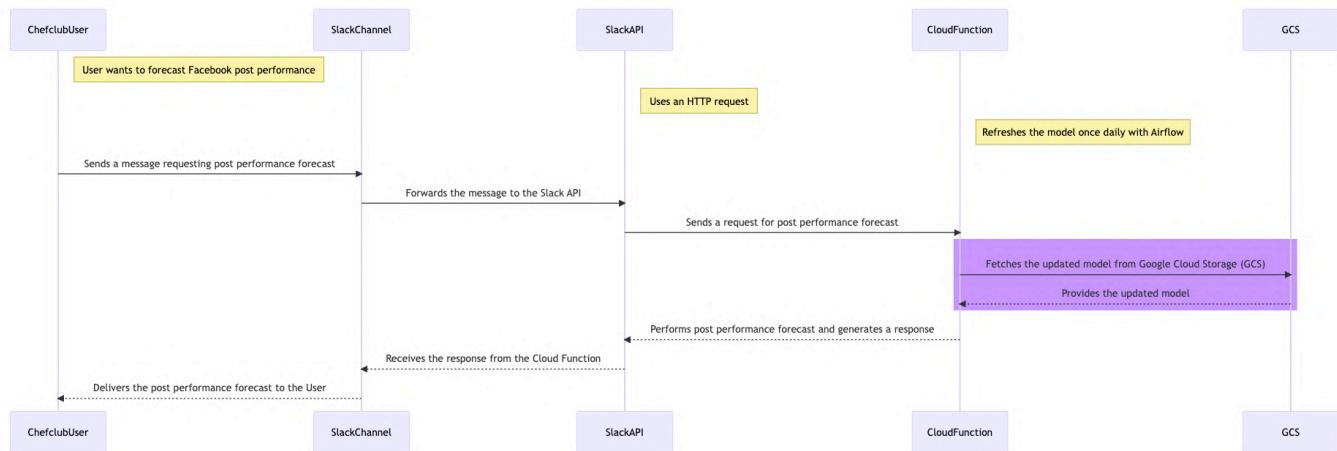


FIG. 3.14 : Sequence Diagram : Model Inference via Slack

Note : Le type de modèle utilisé pour cette solution est un **modèle de régression**, car nous prédisons le nombre estimé de vues pour le post en question.

3.4.1.3.2 Description de la solution

La solution proposée vise à prédire les performances futures des publications sur Facebook en utilisant des données historiques, afin d’anticiper les résultats d’une publication et de prendre des mesures proactives pour les améliorer. L’utilisateur n’a qu’à fournir les données internes relatives à la publication, et il recevra une estimation des performances attendues. Pour répondre à ce besoin, une architecture a été conçue et intégrée de manière transparente dans l’infrastructure de données de Chefclub, comme illustré dans la Figure 3.13. Cette architecture se compose des éléments suivants :

- **Data Extraction** : Cette étape est orchestrée par Airflow. Chaque jour, les données de performances de Facebook sont extraites et mises à jour quotidiennement. Ces données sont utilisées pour entraîner et améliorer les performances du modèle.
- **Model Training** : Une fois les données historiques récupérées, elles passent par un processus de nettoyage et de prétraitement afin d’être prêtes à être utilisées pour l’entraînement. Pendant cette phase, nous optimisons les hyperparamètres pour trouver les meilleurs hyperparamètres du modèle.
- **Model Saving** : Cette partie est également orchestrée par Airflow. Une fois le modèle entraîné, sa version est sauvegardée dans le GCS (Google Cloud Storage).

- **Model Inference Environment** : Cet environnement fournit les ressources nécessaires pour exécuter le modèle de prédiction. Il s'agit d'une fonction cloud qui récupère la dernière version du modèle entraîné, peut recevoir une requête pour effectuer une prédiction et renvoyer une réponse.
- **Provision of results** : Cette partie de l'architecture permet aux utilisateurs de fournir les données internes relatives à un post spécifique et d'obtenir une réponse détaillée sur les performances attendues. Par exemple, ils peuvent envoyer un message sur Slack dans un canal spécifique. L'API Slack interroge la fonction de prédiction, qui est une fonction cloud, et génère une réponse sur Slack.

3.4.1.3.3 Outils et technologies utilisés

La solution repose sur les outils et technologies suivants :

1. **Python** : Langage de programmation utilisé pour développer notre processus d'ELT (Extract, Load, Transform) et entraîner le modèle.
2. **Airflow** : Plateforme utilisée pour orchestrer et planifier l'exécution quotidienne du job d'entraînement du modèle.
3. **Docker** : Permet de containeriser le code responsable de l'entraînement du modèle.
4. **Kubernetes** : Plateforme d'orchestration de conteneurs utilisée pour assurer une gestion optimale des conteneurs créés lors de l'entraînement du modèle.
5. **Google Cloud Storage** : Utilisé pour stocker quotidiennement le fichier pickle du modèle entraîné.
6. **Google BigQuery** : Source des tables utilisées pour l'entraînement du modèle.
7. **Pandas** : Bibliothèque Python utilisée pour le traitement des DataFrames et la préparation des données.
8. **NumPy** : Utilisé pour effectuer des opérations mathématiques et manipuler les données lors des transformations.
9. **Jupyter Notebook** : Environnement de travail utilisé pour développer le code qui sera mis en production.
10. **Scikit-learn** : Bibliothèque Python utilisée pour entraîner le modèle destiné à la production.
11. **Cloud Function** : Environnement déployé et géré par Google, permettant l'exécution de fonctions, notamment celle chargée de la prédiction.
12. **Slack API** : Cette API nous permet de communiquer avec le modèle et l'utilisateur qui envoie sa requête sur un canal Slack.

3.4.1.3.4 Justification des choix

Voici les explications justifiant les choix effectués dans la conception de la solution proposée (Figure 3.13) :

- **Utilisation de Python pour le développement du modèle et la création d'ETL :** Python est un langage de programmation extrêmement populaire dans le domaine du Machine Learning, offrant une grande flexibilité et une variété de modèles préconstruits disponibles via des bibliothèques telles que Scikit-learn. De plus, Python est le langage principal utilisé par l'équipe data pour le développement des DAGs dans Airflow, ce qui en fait un choix naturel pour la création de modèles et d'ETLs.
- **Adoption d'Airflow pour le model training :** Airflow est déjà présent dans l'infrastructure data existante et facilite la gestion de l'orchestration et de la planification des jobs de récupération de données, ainsi que des workflows ETL. En tirant parti de cette plateforme, il est possible d'appliquer le même concept à la phase d'entraînement des modèles.
- **Containerisation du code Python pour le model training :** Nous avons choisi d'utiliser Docker pour containeriser le code responsable de l'entraînement du modèle. Cela permet d'isoler le code, ses dépendances et ses configurations, facilitant ainsi le déploiement et l'exécution dans différents environnements sans se soucier des différences de configuration. Cette approche cohérente avec les ETLs existants de l'infrastructure garantit une meilleure gestion et une plus grande flexibilité lors de l'exécution du processus d'entraînement du modèle.
- **Stockage du modèle dans Google Cloud Storage (GCS) :** Nous avons choisi d'utiliser GCS pour stocker quotidiennement la version du modèle entraîné. Étant donné que notre modèle est de petite taille, nous avons opté pour cette solution simple et efficace plutôt que d'introduire des services de type model registry supplémentaires. GCS offre une fiabilité, une disponibilité et une intégration transparente avec d'autres services Google Cloud, ce qui facilite la récupération du modèle entraîné lorsqu'il est nécessaire d'effectuer des prédictions.
- **Choix de scikit-learn (Sklearn) :** Nous avons choisi d'utiliser scikit-learn (Sklearn) comme bibliothèque principale pour l'entraînement du modèle destiné à la production. Sklearn est une bibliothèque Python puissante et largement utilisée dans le domaine du Machine Learning. Elle offre une grande variété d'algorithmes d'apprentissage automatique, des fonctionnalités d'optimisation des hyperparamètres et des outils pour évaluer les performances du modèle. En utilisant scikit-learn, nous avons pu développer, entraîner et évaluer notre modèle de prédiction de manière efficace et robuste.
- **Utilisation de Jupyter Notebook :** Nous avons opté pour l'utilisation de Jupyter Notebook comme environnement de travail pour le développement du code destiné à la production. Jupyter Notebook est un environnement interactif populaire permettant le développement et l'exécution de code Python. Il offre une interface conviviale pour l'écriture, le test et la visualisation du code, ainsi que la possibilité d'intégrer des visualisations et des explications textuelles. En utilisant Jupyter Notebook, nous avons pu itérer rapidement sur notre solution, expérimenter différents modèles et techniques, et faciliter la collaboration au sein de l'équipe data.
- **Utilisation de Cloud Function :** Nous avons choisi d'utiliser Cloud Function, un service serverless proposé par Google Cloud, pour exécuter la fonction de prédiction. Cloud Function nous a permis de déployer facilement une fonction de prédiction qui peut être invoquée à la

demande, sans se soucier de la gestion de l'infrastructure. Cela nous a offert une mise à l'échelle automatique en fonction du volume de requêtes, une haute disponibilité et des temps de réponse rapides pour les prédictions. De plus, en utilisant Cloud Function, nous avons pu bénéficier de l'intégration étroite avec les autres services Google Cloud, tels que Google Cloud Storage, ce qui facilite la récupération du modèle entraîné.

- **Intégration de l'API Slack :** L'intégration de l'API Slack nous a permis de communiquer facilement avec le modèle et les utilisateurs. Les utilisateurs peuvent envoyer leurs données de publication via Slack, ce qui facilite l'interaction avec la solution de prédiction. De plus, les résultats de prédiction sont renvoyés directement dans le canal Slack, offrant ainsi une expérience utilisateur fluide et pratique.

En résumé, les choix de conception de la solution Model Training/Model Deployment comprennent l'utilisation de Python pour le développement du modèle et des ETL, l'adoption d'Airflow pour l'orchestration des jobs de récupération de données, le model training avec le code conteneurisé, le stockage du modèle dans Google Cloud Storage, l'utilisation de scikit-learn pour l'entraînement du modèle, Jupyter Notebook comme environnement de développement et de test, Cloud Function comme environnement d'inférence pour l'exécution de la fonction de prédiction, et l'intégration de l'API Slack pour servir les utilisateurs des résultats des prédictions.

3.5 Conclusion

Ce chapitre a permis de détailler l'analyse et la conception de la solution, en mettant en évidence les workflows des différentes tâches et projets auxquels j'ai participé. Dans un premier temps, j'ai présenté les besoins fonctionnels et non fonctionnels de l'entreprise en sciences des données, en les ayant également formulés pour l'équipe data. Ensuite, j'ai exposé l'infrastructure data actuelle de Chefclub, en soulignant les améliorations apportées à sa stabilité et en fournissant un schéma récapitulatif de l'ensemble de cette infrastructure. J'ai ensuite défini la conception et le workflow des solutions et des tâches que j'ai réalisés. Toutes ces étapes ont permis de cadrer le projet de manière cohérente et de garantir que toutes les exigences ont été prises en compte lors de la conception de la solution.

Chapitre 4

Implémentation et validation de la solution

Dans ce dernier chapitre, j'aborde le développement et la mise en production des solutions proposées. L'objectif est de déployer les différentes phases de développement et de test afin de concrétiser les concepts définis précédemment lors de la phase d'analyse et de conception, tout en garantissant qu'ils répondent aux besoins identifiés.

4.1 Introduction

Après avoir examiné les besoins et proposé une solution détaillée pour y répondre, ce chapitre explique le développement à travers les différentes phases, les tests de vérification, ainsi que l'assurance que les solutions sont opérationnelles et répondent aux besoins identifiés, tout en s'intégrant correctement à l'infrastructure data de Chefclub existante.

4.2 Développement et Test de la solution Data Engineering / DataOps

4.2.1 Développement et Test du Code Python qui récupère les données de l'API

Dans cette première partie du chapitre, je me concentre sur le développement de la solution Data Engineering / DataOps en suivant la conception et les outils définis dans le chapitre précédent. La figure 4.1 suivante rappelle la conception de la solution que nous avons mise en place lors de l'analyse et la conception :

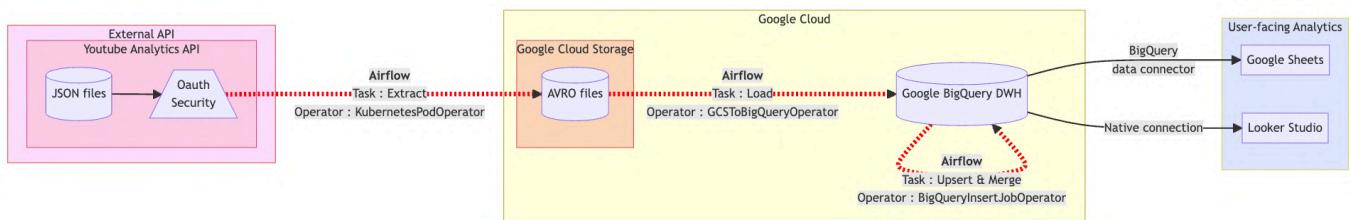


FIG. 4.1 : YouTube Analytics Data Retrieval Solution (Reminder)

4.2.1.1 Overview sur le fonctionnement final du code Python développé

Cette section vise à présenter au lecteur le résultat final du code Python développé, ainsi que ses inputs et outputs. Ce code représente le cœur de la solution. Les sections suivantes détailleront les composantes, les fonctionnalités et l'évolution du code afin de le rendre industrialisable et prêt pour la production.

Le code développé est responsable de la récupération des données à partir de l'API *YouTube Analytics* et de leur téléchargement dans un *bucket* de stockage *Google Cloud Storage* au format *AVRO*. Le code peut être configuré pour récupérer des données sur une période spécifique et pour une chaîne *YouTube* donnée, et générer des rapports personnalisés en fonction des métriques/rapports choisis.

Les paramètres d'entrée du code sont spécifiés à travers le script `main.py`, qui requiert les arguments suivants :

- `-r/--report-id` : l'identifiant du rapport personnalisé à générer.
- `-c/--channel-id` : l'identifiant de la chaîne *YouTube* pour laquelle récupérer les données.
- `-s/--start-date` : la date de début de la période de récupération des données au format *AAAA-MM-JJ*.

- **-e/--end-date** : la date de fin de la période de récupération des données au format *AAAA-MM-JJ*.
- **-d/--destination** : le dossier dans le bucket *GCS* où télécharger les données récupérées.
- **-a/--to-avro** : lorsque défini sur **True**, le fichier est uploadé au format *AVRO*; sinon, il est téléchargé au format *JSON*.

Voici la syntaxe pour exécuter le script :

```
1 $ python main.py -c <channel_id> -r <report_id> -s <start_date> -e <end_date> -d
<folder> --to-avro
```

Listing 4.1: CLI Python command to run the code

Le résultat du job YT est la récupération des données au format **AVRO/JSON**, qui sont téléchargées dans le bucket *GCS* spécifié. Des rapports personnalisés peuvent être configurés dans le répertoire *config/*.

Voici comment mon code est organisé, comme illustré dans la figure 4.2 ci-dessous. Je fournirai plus de détails sur ce code dans les sections suivantes.

```
airflow-manager
├── jobs
│   ├── youtube-fetch
│   │   ├── Dockerfile
│   │   ├── .dockerignore (used to ignore unnecessary files and folders when building the Docker image)
│   │   ├── docker-compose.yml
│   │   ├── Makefile (provides shortcut commands used in the project)
│   │   ├── .gitignore (used to ignore sensitive files and unnecessary folders in Git)
│   │   ├── README.md (provides a brief introduction to the project)
│   │   └── requirements.txt (contains the necessary requirements to run the code)
│   └── tools/
│       └── channel-credentials-getter.py (a tool used to generate refresh tokens for channels)
└── src/
    ├── client.py (contains code for authentication and fetching data from the YouTube Analytics API)
    ├── config.py (contains functions used to load file contents)
    ├── main.py (contains the main function to run the code)
    ├── utils.py (contains functions used to convert the API response and upload it to GCS)
    └── sample_secrets/ (contains informative data on how to set up the secrets folder)
        ├── data-credentials
        │   └── credentials.json (contains the service account credentials needed to upload files to GCS)
        ├── yt-channels-credentials
        │   └── channels_credentials.json (contains the channels refresh tokens generated with the channel-credentials)
        ├── yt-client-credentials
        │   └── client_credentials.json (contains the credentials generated in the YouTube Analytics API with 2auth)
        └── config/
            └── reports.yaml (used to add custom reports)
    └── secrets/ (contains the credentials needed to access the GCS bucket where the data is uploaded)
```

FIG. 4.2 : Folder Architecture of **youtube-fetch** in VENV Environment

4.2.1.2 Utilisation de l'API YouTube Analytics

Pour accéder à nos données analytiques historiques, j'utilise l'API YouTube Analytics, cette API est proposé dans GCP permettant aux développeurs d'accéder et de récupérer des données analytiques sur l'activité des utilisateurs, les performances des vidéos et les revenus publicitaires sur YouTube.

La principale chose à faire pour réussir correctement à utiliser cette API, c'est definir les rapports à récupérer. Dans notre cas j'ai personnalisé les rapports en fonction de nos besoins. Il y a la possibilité de demander les données en fonction de la variable temporelle (c'est une dimension), on peut aussi appliquer des filtres. Voici un exemple de code Python utilisé pour demander un rapport à l'API :

```

1 def fetch_youtube_analytics_data( youtube_analytics_service,
2                                     report_config_parameters,
3                                     start_date, end_date,
4                                     raise_error_on_empty_api_response: bool = False,
5                                     ):
6     print(
7         f"""
8             - Date Range: {start_date} to {end_date}
9             - Metrics: {report_config_parameters.get('metrics', '')}
10            - Dimensions: {report_config_parameters.get('dimensions', '')}
11            - Filters: {report_config_parameters.get('filters', '')}
12            - Sort: {report_config_parameters.get('sort', '')}
13            - Max Results: {report_config_parameters.get('maxResults', '')}
14            - Start Index: {report_config_parameters.get('startIndex', '')}
15            - Include Historical Channel Data: {report_config_parameters.get('
16                includeHistoricalChannelData', '')}
17
18        """
19    )
20
21    try:
22        request = youtube_analytics_service.reports().query(
23            ids="channel==MINE",
24            dimensions=report_config_parameters.get("dimensions", None),
25            startDate=start_date,
26            endDate=end_date,
27            maxResults=report_config_parameters.get("maxResults", None),
28            metrics=", ".join(report_config_parameters.get("metrics", "")),
29            sort=report_config_parameters.get("sort", None),
30            startIndex=report_config_parameters.get("startIndex", None),
31            filters=report_config_parameters.get("filters", None),
32            includeHistoricalChannelData=report_config_parameters.get(
33                "includeHistoricalChannelData", None
34            ),
35        )
36
37        response = request.execute()
38        if "error" in response:
39            error = response["error"]
40            raise Exception(f"Failed to fetch data. Error message: {error['message']}")
41
42    if len(response["rows"]) == 0 and bool(raise_error_on_empty_api_response):
43        raise Exception("Failed to fetch data. Error message: Empty Data!")
44    except googleapiclient.errors.HttpError as e:
45        raise Exception(f"Failed to fetch data. Error message: {e}")
46
47    return response

```

Listing 4.2: Python code to fetch YouTube analytics data

La fonction `fetch_youtube_analytics_data()` envoie une requête à l'API YT Analytics en utilisant les paramètres de configuration du rapport, ainsi que les dates de début et de fin. Elle renvoie la réponse de l'API, et si une erreur se produit ou si la réponse est vide, une exception est levée.

Pour plus d'informations sur les rapports qu'on peut demander à l'API YT Analytics, vous pouvez consulter la documentation suivante :¹ [YT-Analytics-docs]

¹developers.google.com/youtube/analytics

4.2.1.3 Extraction des données analytiques avec Python

Pour extraire les données analytiques en appelant l’API YouTube Analytics. Le processus comprend plusieurs étapes : l’installation des packages requis, l’authentification avec les identifiants, la définition du type de rapport souhaité, puis l’appel de l’API.

J’ai la possibilité de demander différents types de rapports à l’API afin d’accéder aux données analytiques. Par exemple, je peux demander un rapport sur la monétisation de la chaîne. La figure 4.20 suivante présente un exemple de rapport que j’utilise en production. Ce rapport interroge l’API pour obtenir les données de monétisation.

```
time-based-day-monetized:
parameters:
metrics:
- estimatedRevenue
- estimatedAdRevenue
- grossRevenue
- estimatedRedPartnerRevenue
- monetizedPlaybacks
- playbackBasedCpm
- adImpressions
- cpm
dimensions: day
filters:
```

FIG. 4.3 : Configuration File - YT Analytics Reports

Je transmets ce rapport en incluant d’autres paramètres tels que l’intervalle de temps, l’identifiant de la chaîne et les informations d’authentification. En retour, l’API me renvoie une réponse au format JSON, représentée sous forme de dictionnaire. Vous pouvez trouver un exemple de code dans la figure suivante, qui illustre comment j’effectue cette demande à l’API en utilisant Python :

```
1 # Load client yt_channel_credentials
2 yt_channel_credentials = load_credentials(CHANNEL_CREDENTIALS_PATH)
3 yt_client_credentials = load_credentials(CLIENT_CREDENTIALS_PATH)
4 gcs_credentials = service_account.Credentials.from_service_account_file(
DATA_CREDENTIALS_PATH)
5
6 # Authenticate with the YouTube API
7 youtube_analytics_service = authenticate(channel_id,
8                                         yt_channel_credentials,
9                                         yt_client_credentials)
10
11 # Get report parameters
12 report_params = load_report_params(REPORT_PATH, report_id)
13
14 # Fetch data from YouTube Analytics API
15 youtube_analytics_response = fetch_youtube_analytics_data(
16                                         youtube_analytics_service,
17                                         report_params,
18                                         start_date,
19                                         end_date)
```

Listing 4.3: Custom Python code for retrieving YouTube analytics data using YAML Reports

La fonction `authenticate()` permet l'authentification et rendre un service authentifié en utilisant les arguments suivants : `channel_id` (identifiant de la chaîne), `yt_channel_credentials` (les informations d'identification de la chaîne) et `yt_client_credentials` (l'identifiant du compte de service ayant créé les informations d'identification de la chaîne). Après l'authentification, il est possible de demander un rapport en utilisant la fonction `fetch_youtube_analytics_data()`.

4.2.1.4 Transformation de la réponse JSON en AVRO

Dans cette partie, je vais expliquer comment je transforme et adapte la réponse brute de l'API en utilisant le format AVRO. J'effectue plusieurs étapes pour cela. Tout d'abord, je commence par le parsing du JSON afin d'extraire les champs qui m'intéressent. En fonction des besoins spécifiques, je sélectionne les données pertinentes pour la transformation. Ensuite, j'ajoute des informations supplémentaires à la réponse JSON. Ces informations incluent des champs tels que `FETCHED_AT_UTC`, `CHANNEL_ID`, `RUN_START_DATE` et `RUN_END_DATE`. Ces champs sont nécessaires pour des analyses ultérieures et surtout fournir plus de contexte aux données et permettre de faire l'opération **Upsert** (Voir sous-section 4.2.2.5)

Une fois que j'ai effectué toutes les transformations et ajouté les champs nécessaires, je procède à la création du fichier AVRO. AVRO (Voir l'exemple dans la figure 4.4) est un format de données efficace et compact, adapté au stockage et au traitement de données volumineuses. Je stocke le fichier AVRO en vue d'un envoi vers une Staging Area comme GCS (Google Cloud Storage).

Voici le code utilisé pour effectuer cette tâche :

```

1  def convert_yt_data(data: dict, channel_id: str, start_date: str, end_date: str)
2      -> str:
3      execution_date_UTC = get_utc_timestamp()
4      file_path = f"{channel_id}.avro"
5      schema_fields = [
6          {'name': 'FETCHED_AT_UTC', 'type': 'string'},
7          {'name': 'RUN_CHANNEL_ID', 'type': 'string'},
8          {'name': 'RUN_START_DATE', 'type': ['null', 'string']},
9          {'name': 'RUN_END_DATE', 'type': ['null', 'string']}
10     ]
11     schema_fields.extend(
12         {'name': column['name'], 'type': ['null', 'int'] if column['dataType'] == 'INTEGER' else ['null', 'float'] if
13             column['dataType'] == 'FLOAT' else 'string'} for column in data['
14             columnHeaders']
15     )
16     schema = avro.schema.parse(json.dumps({'type': 'record', 'name': 'YoutubeData', 'fields': schema_fields}))
17     with open(file_path, 'wb') as f:
18         writer = DataFileWriter(f, DatumWriter(), schema)
19         for row in data.get('rows', []):
20             record = {'FETCHED_AT_UTC': execution_date_UTC, 'RUN_START_DATE':
21             start_date, 'RUN_END_DATE': end_date,
22                         'RUN_CHANNEL_ID': channel_id}
23             for i, value in enumerate(row):
24                 column_name = data['columnHeaders'][i]['name']
25                 if value is not None:
26                     record[column_name] = float(value) if data['columnHeaders'][i][
27                     'dataType'] == 'FLOAT' else value
28             writer.append(record)

```

```
25     writer.close()
26     return file_path
```

Listing 4.4: Code to convert YouTube Analytics data to Avro format and write it to a file

Ce fichier *AVRO* doit être téléchargé vers Google Cloud Storage. Avant cela, il est nécessaire d'organiser les fichiers afin de faciliter leur utilisation pour les analyses. La section suivante décrit en détail ce processus.

4.2.1.5 Configuration de Cloud Storage Storage comme Staging Area

L'organisation des données revêt une grande importance dans mon processus de développement. J'ai choisi une structure d'organisation suivante : *report-id/year-month-day/channel-id.avro*. Cette organisation me permet d'ajouter de nouvelles chaînes et de générer différents rapports sans impacter la structure existante. De plus, cette configuration répond aux meilleures pratiques en matière de récupération de données depuis mon Data Warehouse (DWH). En utilisant Google Cloud Storage (GCS) comme zone de stockage temporaire, je peux assurer la disponibilité et l'évolutivité nécessaires pour gérer mes données.

La figure 4.4 suivante montre un exemple de fichier AVRO après transformation :

	FETCHED_AT_UTC	RUN_CHANNEL_ID	RUN_START_DATE	RUN_END_DATE	day	creatorContentType	estimatedAdRevenue
1	2023-06-07 16:58:47	UCQ_I	vf6_xA	2022-01-28	2022-01-28	shorts	1.183
2	2023-06-07 16:58:47	UCQ_I	vf6_xA	2022-01-28	2022-01-28	videoOnDemand	86.322

FIG. 4.4 : Example AVRO File Containing API Response

La figure 4.5 suivante présente l'organisation des données sur Google Cloud Storage (GCS), où chaque fichier est identifié par le rapport auquel il appartient, la date des données (dimension temporelle) et l'identifiant du canal associé.

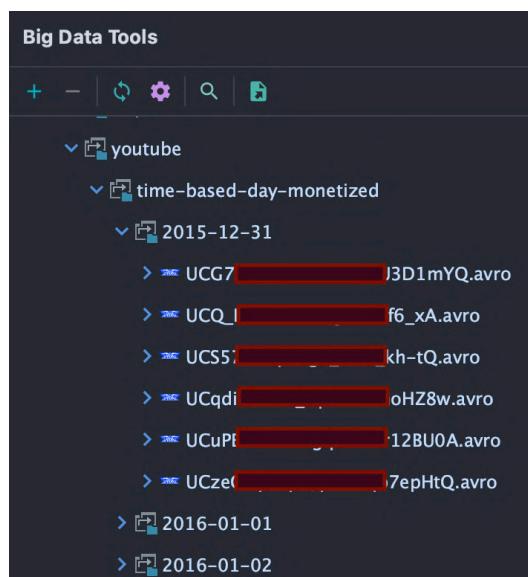


FIG. 4.5 : Organization of Analytics Files on GCS Bucket

4.2.1.6 Test et Validation du Code Python avec VENV

L'environnement local VENV (Virtual Environment) est utilisé pour le développement et les tests locaux (Voir paragraphe 2.2.2.1.1). J'ai configuré l'environnement VENV avec les bibliothèques et les packages nécessaires, ce qui me permet de développer et de tester mon code de manière indépendante de l'environnement global.

Note : Avant de dockeriser mon code, il est essentiel d'effectuer des tests dans l'environnement VENV. Pour cela, j'ai organisé mon environnement VENV comme illustré dans la figure 4.2

Après avoir activé l'environnement VENV, j'ai exécuté un test qui est visible dans la figure suivante. En réponse, j'obtiens le lien où le fichier AVRO est stocké sur GCS :

```
(venv) + youtube-fetch git:(master) x python src/main.py -c UCQ_Fr[REDACTED] -r top-videos-US -s 2021-01-01 -e 2021-12-30 -d top-videos-US-2021 --to-avro
  - Date Range: 2021-01-01 to 2021-12-30
  - Metrics: ['views', 'estimatedMinutesWatched', 'likes', 'subscribersGained']
  - Dimensions: video
  - Filters: country==US
  - Sort: -views
  - Max Results: 200
  - Start Index:
  - Include Historical Channel Data:

File uploaded to gs://business-analysis-3/top-videos-US-2021/UCQ_Fr[REDACTED].avro
-> Done for channel 'UCQ_FrN6FNViZVjxHUVf6_xA' channel
```

FIG. 4.6 : Python Code testing in virtual environment (VENV)

4.2.2 Développement, Test et Orchestration du DAG Airflow

4.2.2.1 Conteneurisation, Upload vers l'Artifact Registry et Test du code avec Docker

4.2.2.1.1 Containerisation avec Docker et Upload vers l'Artifact Registry

Le code développé va être exécuté en production, il est donc nécessaire de le containeriser dans une image Docker et de le sauvegarder dans le *Google Cloud Artifact Registry*. Pour cela, je crée une image Docker à l'aide d'un *Dockerfile*, qui regroupe toutes les dépendances du code dans l'image. Voici un exemple de Dockerfile utilisé dans notre cas :



FIG. 4.7 : Used Dockerfile to build the YouTube Analytics Code Image

Le Dockerfile spécifie les étapes nécessaires pour construire l'image Docker, notamment l'installation des dépendances, la copie des fichiers de code source, la définition des variables d'environnement, etc.

Dans le but de simplifier la création de l'image Docker, en particulier lors de la phase de test de la solution, j'ai mis en place une commande Make. Cette commande permet de générer automatiquement

l'image Docker en exécutant les instructions du Dockerfile. Voici un exemple de commande Make utilisée :

```
52 ► build-image: # builds the image: youtube-analytics-image:dev
53   docker build --no-cache -t ${IMAGE_NAME}:dev --build-arg GIT_HASH=${GIT_HASH} --build-arg GIT_VERSION=$(git_version) .
54   @echo
55   @echo "Successfully built the image: ${IMAGE_NAME}:dev"
56
```

FIG. 4.8 : Make command to build the Docker image

La commande Make simplifie le processus de création de l'image Docker en automatisant les étapes nécessaires. Elle permet de générer rapidement l'image Docker à des fins de test et de déploiement.

De plus, pour faciliter le transfert de l'image vers le Google Cloud Artifact Registry, j'utilise la commande Make suivante. J'ajoute également un TAG à l'image en utilisant le hachage GitHub du commit actuel pour l'identifier l'image en production.

```
70 ► push-dev-image-with-git-hash: # pushes the image: youtube-analytics-image:dev in Google Artifacts Registry and returns the tag
71   docker tag ${IMAGE_NAME}:dev ${REGISTRY_YOUTUBE_IMAGE_NAME}:$(GIT_HASH)
72   docker push ${REGISTRY_YOUTUBE_IMAGE_NAME}:$(GIT_HASH)
73   @echo
74   @echo ">> Here is the used image tag: $(GIT_HASH) ( update your DAGs! 😊 )"
```

FIG. 4.9 : Make command to push the image to Google Artifact Registry

En utilisant cette commande, je peux pousser l'image Docker vers le Artifact Registry de Google et l'associer à un hachage GitHub spécifique pour un suivi plus précis lors du déploiement en production.

La figure suivante montre l'image YouTube Analytics dans le Google Artifact Registry :

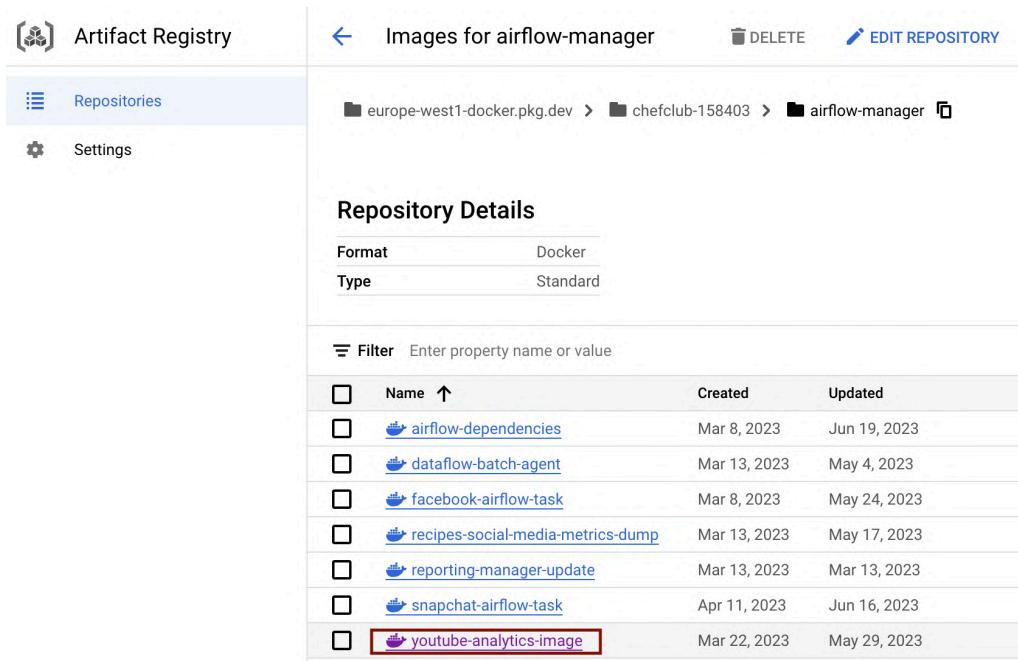


FIG. 4.10 : YouTube Analytics image in the Google Artifact Registry

Note : Maintenant que mon image Docker est stockée dans le **Google Artifact Registry**, mon

encadrant la récupère sur sa machine, la teste, puis vérifie sa validité dans un autre environnement. La section suivante explique comment nous validons ce code sur Docker.

4.2.2.1.2 Test de la solution sur un environnement Docker

Pour effectuer des tests appropriés avec Docker, j'ai créé les commandes Make suivantes qui permettent d'automatiser la création de l'image et son test. Après avoir testé la solution, j'ai poussé l'image correspondante dans le Google Artifact Registry afin qu'elle puisse être testée par les autres membres de l'équipe. La figure 4.11 suivante montre les commandes Make :

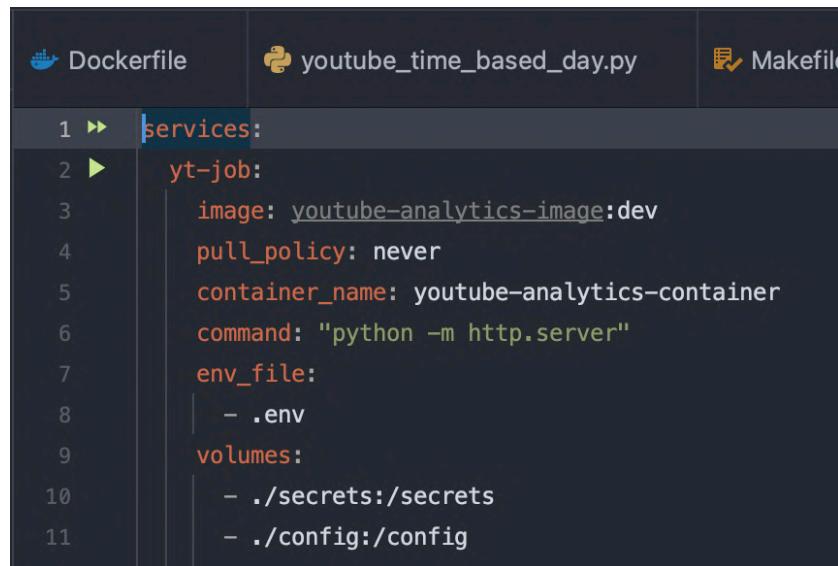
```

52 ► build-image: # builds the image: youtube-analytics-image:dev
53     @docker build --no-cache -t $(IMAGE_NAME):dev --build-arg GIT_HASH=$(GIT_HASH) --build-arg GIT_VERSION=$(git_version) .
54     @echo
55     @echo "Successfully built the image: $(IMAGE_NAME):dev"
56
57 ► run-container: # creates the container youtube-analytics-container
58     @docker-compose -p $(SERVICE_NAME) up -d
59     @echo
60     @echo "Successfully run : $(SERVICE_NAME)"
61
62 ► open-shell: # returns the shell of the container: youtube-analytics-container
63     @docker exec -it $(CONTAINER_NAME) bash
64
65 ► stop-container: # stops the container: youtube-analytics-container
66     @docker-compose -p $(SERVICE_NAME) down
67     @echo
68     @echo "Successfully stopped the container $(SERVICE_NAME)"
69
70 ► push-dev-image-with-git-hash: # pushes the image: youtube-analytics-image:dev in Google Artifacts Registry and returns the tag
71     @docker tag $(IMAGE_NAME):dev $(REGISTRY_YOUTUBE_IMAGE_NAME):$(GIT_HASH)
72     @docker push $(REGISTRY_YOUTUBE_IMAGE_NAME):$(GIT_HASH)
73     @echo
74     @echo ">> Here is the used image tag: $(GIT_HASH) ( update your DAGs! 😊 )"

```

FIG. 4.11 : Make commands for **image build**, **container run**, and **image push**

Pour l'exécution de l'image et la création du conteneur Docker, j'utilise un fichier docker-compose qui construit le conteneur et monte en volume les secrets avec les fichiers de configuration. La figure 4.12 suivante montre le fichier docker-compose utilisé :



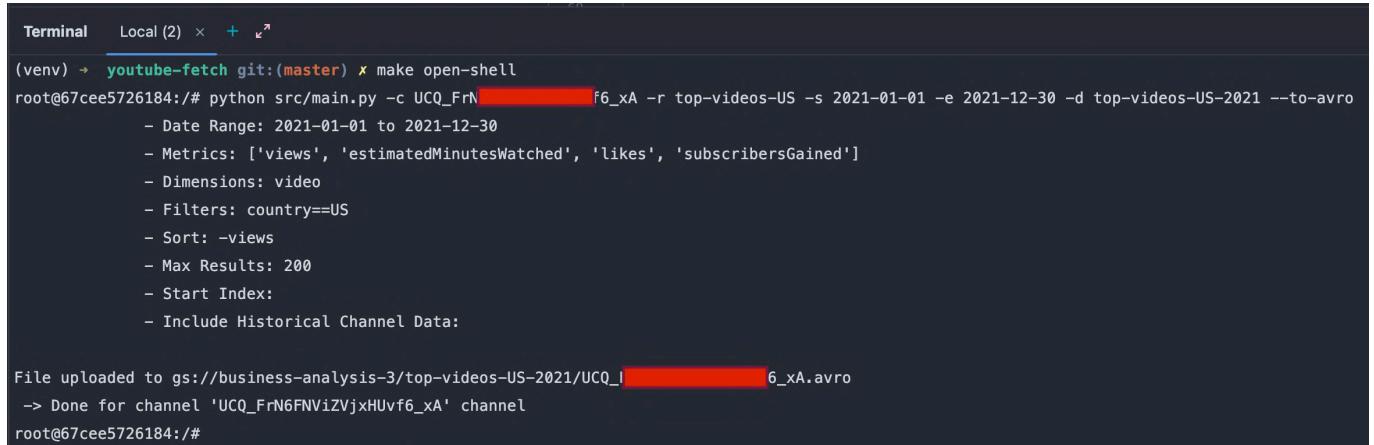
```

Dockerfile          youtube_time_based_day.py      Makefile
1 ► services:
2 ►   yt-job:
3       image: youtube-analytics-image:dev
4       pull_policy: never
5       container_name: youtube-analytics-container
6       command: "python -m http.server"
7       env_file:
8           - .env
9       volumes:
10      - ./secrets:/secrets
11      - ./config:/config

```

FIG. 4.12 : Docker Compose file for local container build

Après la construction du conteneur, j'accède au conteneur avec la commande **Make shell**, puis j'exécute mon test. En fin de compte, je reçois le message indiquant que le fichier a été stocké avec succès sur GCS. La figure 4.13 suivante l'illustre :



```
Terminal Local (2) × + ↗
(venv) → youtube-fetch git:(master) ✘ make open-shell
root@67cee5726184:/# python src/main.py -c UCQ_FrN[REDACTED]f6_xA -r top-videos-US -s 2021-01-01 -e 2021-12-30 -d top-videos-US-2021 --to-avro
  - Date Range: 2021-01-01 to 2021-12-30
  - Metrics: ['views', 'estimatedMinutesWatched', 'likes', 'subscribersGained']
  - Dimensions: video
  - Filters: country==US
  - Sort: -views
  - Max Results: 200
  - Start Index:
  - Include Historical Channel Data:

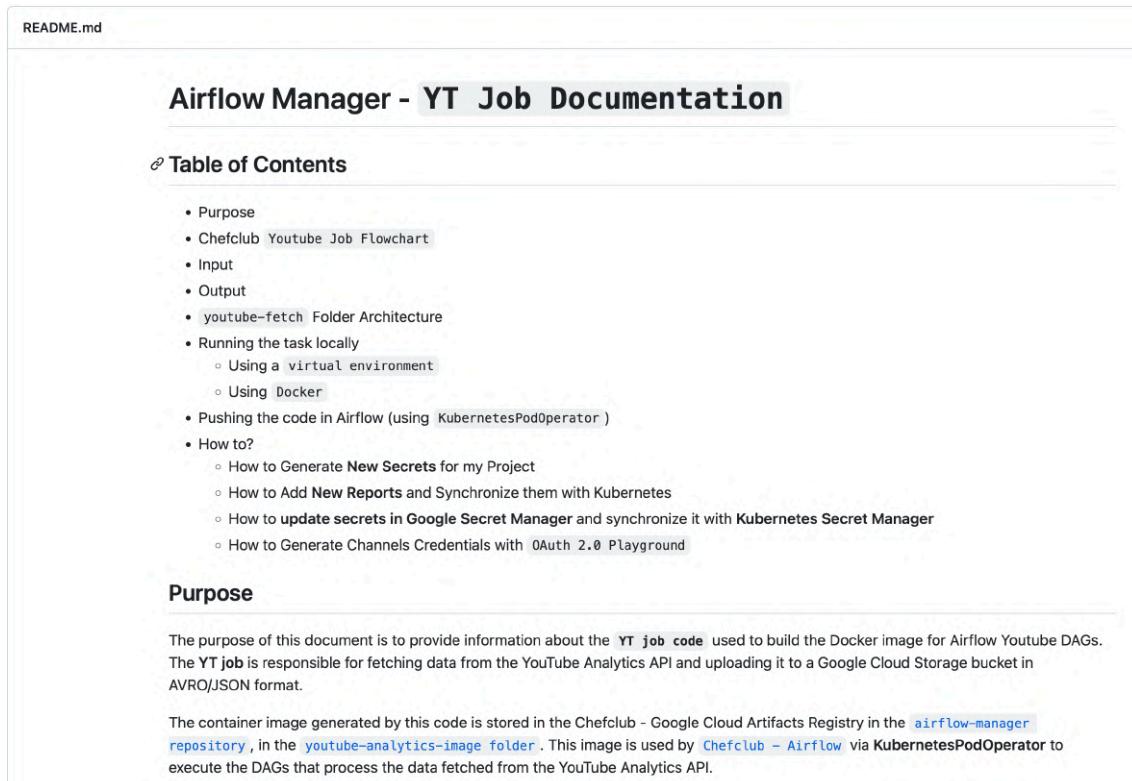
File uploaded to gs://business-analysis-3/top-videos-US-2021/UCQ_I[REDACTED]6_xA.avro
-> Done for channel 'UCQ_FrN6FNVizVjxHUVf6_xA' channel
root@67cee5726184:/#
```

FIG. 4.13 : Testing the custom Python Code inside the container

4.2.2.1.3 Documentation du code sur GitHub

Afin de faciliter la compréhension et la collaboration autour de notre solution, j'ai documenté le code sur GitHub de Chefclub. La documentation comprend une description détaillée de chaque composant ainsi que des instructions d'installation.

La figure 4.14 suivante illustre la documentation du code dans le fichier **README.md** :



README.md

Airflow Manager - YT Job Documentation

Table of Contents

- Purpose
- Chefclub Youtube Job Flowchart
- Input
- Output
- youtube-fetch Folder Architecture
- Running the task locally
 - Using a virtual environment
 - Using Docker
- Pushing the code in Airflow (using KubernetesPodOperator)
- How to?
 - How to Generate New Secrets for my Project
 - How to Add New Reports and Synchronize them with Kubernetes
 - How to update secrets in Google Secret Manager and synchronize it with Kubernetes Secret Manager
 - How to Generate Channels Credentials with OAuth 2.0 Playground

Purpose

The purpose of this document is to provide information about the **YT job code** used to build the Docker image for Airflow Youtube DAGs. The **YT job** is responsible for fetching data from the YouTube Analytics API and uploading it to a Google Cloud Storage bucket in AVRO/JSON format.

The container image generated by this code is stored in the Chefclub - Google Cloud Artifacts Registry in the [airflow-manager repository](#), in the [youtube-analytics-image folder](#). This image is used by [Chefclub - Airflow](#) via [KubernetesPodOperator](#) to execute the DAGs that process the data fetched from the YouTube Analytics API.

FIG. 4.14 : YT Job Code Documentation

La documentation sur GitHub fournit aux membres de l'équipe et aux utilisateurs une référence complète sur la structure et le fonctionnement de notre solution. Cela facilite la maintenance, les futures améliorations et la collaboration dans un environnement de développement partagé.

4.2.2.2 Configuration de Google BigQuery pour le traitement des données

La configuration de Google BigQuery pour le traitement des données dans notre projet implique plusieurs aspects. Tout d'abord, j'organise les tables dans des datasets spécifiques pour regrouper les données liées. De plus, j'utilise des tables matérialisées natives pour optimiser les requêtes en partitionnant les données selon des critères pertinents, tels que le rapport : **time-based-day-non-monitized**.

La figure 4.15 suivante illustre un exemple de table de performance des chaînes utilisant une table native dans BigQuery, où j'ai personnalisé les tables et les colonnes pour répondre à nos besoins spécifiques.

The screenshot shows the Google Cloud BigQuery interface. On the left, the 'Explorer' sidebar lists a dataset named 'chefclub-1' containing various tables like 'airflow_tables_temp', 'production', and 'social_media'. In the main area, a query titled 'Untitled 2' is displayed with the following SQL code:

```
1 SELECT
2   *
3   FROM
4   `chefclub-1.dim_page_perf_day_yt_no...ed_` AS T
5   ORDER BY
6   RUN_START_DATE DESC;
```

Below the code, the 'Query results' section shows a table with five rows of data:

Row	FETCHED_AT_UTC	RUN_CHANNEL_ID	RUN_START_DATE	RUN_END_DATE
1	2023-07-06 19:31:25	UCn5...JSRcaA	2023-07-04	2023-07-04
2	2023-07-06 19:39:01	UCrl...luNjXHpg	2023-07-04	2023-07-04
3	2023-07-06 19:31:26	UCA...rjaOhw	2023-07-04	2023-07-04
4	2023-07-06 19:31:26	UCYV...UldPf8WA	2023-07-04	2023-07-04
5	2023-07-06 19:38:30	UCE...oTJuKA	2023-07-04	2023-07-04

FIG. 4.15 : Example of Performance Channel Table in BigQuery

Cette configuration me permet d'optimiser les performances des requêtes et de manipuler les données de manière efficace dans le cadre de notre projet.

4.2.2.3 Gestion améliorée de l'OAuth pour l'accès aux données de l'API

La gestion de l'authentification est d'une importance cruciale, particulièrement lorsqu'il s'agit d'automatiser et d'orchestrer des tâches sans intervention humaine. Dans notre cas, l'API YouTube Analytics est protégée par OAuth. Afin de renforcer davantage la robustesse de notre solution et d'assurer sa cohérence, nous avons mis en place un système amélioré de gestion des secrets intégré au Google Secret Manager, une solution de stockage sécurisé fournie par GCP. La figure 4.16 suivante illustre le stockage des secrets dans le Google Secret Manager de GCP :

The screenshot shows the Google Secret Manager interface. On the left, there's a sidebar with 'Security' selected. The main area shows a secret named 'YT-CHANNELS-CREDENTIALS' with its project ID 'projects/812058810488/secrets/YT-CHANNELS-CREDENTIALS'. The 'VERSIONS' tab is active, displaying a table with one row:

	Version	Aliases	Status	Encryption	Created on	Actions
	26	-	Enabled	Google-managed	6/30/23, 10:14 AM	⋮

FIG. 4.16 : Secrets YouTube dans Google Secret Manager

Dans ce contexte, j'ai créé des secrets contenant les tokens de rafraîchissement des chaînes, ce qui me permet d'accéder de manière sécurisée et continue aux données historiques. Pour simplifier l'orchestration de ma solution avec Airflow et faciliter son industrialisation, j'ai regroupé tous ces secrets dans un fichier YAML. Cette approche centralise la gestion des secrets et m'offre une flexibilité pour les modifier selon les besoins.

En cas de problème avec l'un des tokens, il suffit de mettre à jour le fichier dans le Secret Manager, puis de lancer l'orchestration. Cette méthode assure une transition fluide et sans interruption du système, en garantissant un accès ininterrompu aux données historiques. J'ai également créé une commande Make pour simplifier le processus de changement de secrets sur Google Secret Manager. La figure 4.17 suivante illustre cette commande :

```

41 ► update-yt-report-in-GSM: # updates Report in google secret manager
42   @gcloud secrets versions add YT-CONFIG-REPORT --data-file=config/reports.yaml
43   @echo
44   @echo "Successfully updated the report"
45

```

FIG. 4.17 : Command to Update Secrets in Google Secret Manager with Make

La figure suivante 4.18 illustre un exemple de fichier YAML utilisé pour stocker les secrets, démontrant ainsi comment notre solution facilite la gestion des secrets et l'orchestration du processus.

```

- channel_id: UCYWcDa0A2gI8[REDACTED]
  channel_name: 'Chefclub Cuisine du monde'
  credentials:
    created_at: May 19, 2023 10:44 AM
    refresh_token: 1//03bzUhWoL1u_PcgYTARAAGAMSNwF-L9IrldLM[REDACTED]
  published_at: January 06, 2021 11:14 AM

- channel_id: UCn5l9v1z_5Lxah[REDACTED]
  channel_name: 'Chefclub patisserie'
  credentials:
    created_at: May 19, 2023 10:45 AM
    refresh_token: 1//03mgvioVDNM_5CgYTARAAGAMSNwF-L9IrA83[REDACTED]
  published_at: January 06, 2021 11:31 AM

```

FIG. 4.18 : Exemple de fichier YAML pour les secrets YouTube

4.2.2.4 Développement du DAG

Dans le cadre de la solution Data Engineering / DataOps, j'utilise **Apache Airflow** pour l'orchestration des tâches. Un DAG (Directed Acyclic Graph) est utilisé pour définir et gérer le flux de travail de mon système.

4.2.2.4.1 Paramétrage du DAG

Le développement du DAG consiste à définir les différentes étapes et dépendances des tâches qui doivent être exécutées. Chaque tâche est représentée par un opérateur dans Airflow, et le DAG permet de spécifier l'ordre d'exécution de ces tâches.

Dans un premier temps, je paramètre le DAG. En effet, le DAG fait appel à l'image stockée dans l'Artifact Registry pour exécuter du code. Je définis également les secrets. Les secrets seront montés en tant que volumes pour exécuter le code. Cela me permet d'avoir une image indépendante des secrets, facilitant ainsi leur modification lorsque des échanges de secrets sont nécessaires. Voici le code que j'utilise pour le paramétrage du DAG :

```
1      # Image used in YouTube Dags
2      REGISTRY_IMAGE_NAME = (
3          "europe-west1-docker.pkg.dev"
4          "/chefclub-xxxx"
5          "/airflow-manager"
6          "/youtube-analytics-image:"
7      )
8      REGISTRY_IMAGE_TAG = "xxxxxxxx"
9
10     # Secret sync from DATASCIENCE_GSA_CREDENTIALS in Google Secret Manager
11     datascience_gsa_credentials = Secret(
12         deploy_type="volume",
13         deploy_target="secrets/data-credentials/",
14         secret="datascience-gsa-credentials",
15         key="credentials.json",
16     )
17
18     # Secret sync from YT CHANNELS CREDENTIALS in Google Secret Manager
19     yt_channels_credentials = Secret(
20         deploy_type="volume",
21         deploy_target="secrets/yt-channels-credentials/",
22         secret="yt-channels-credentials",
23         key="channels_credentials.yml",
24     )
```

Listing 4.5: Configuration of the DAG with Secrets and Custom Image in the Artifact Registry

Une fois que mon DAG a accès à l'image dans l'Artifact Registry, je configure également d'autres paramètres, tels que l'accès à Google Cloud Storage et à BigQuery pour créer les tables temporaires. De plus, je définis les rapports à récupérer et l'intervalle de temps. Voici un extrait du code permettant de réaliser ces configurations :

```
1      # Data date interval
2      START_DATE = "{{ macros.ds_add(ds, -1) }}"
3      END_DATE = START_DATE
4
5      # BigQuery configuration
```

```

6      BQ_PROJECT = "chefclub-xxxxx"
7      BQ_TEMP_DATASET = "airflow_tables_temp"
8
9      # Define the default arguments for Airflow Dag
10     default_args = {
11         "owner": "airflow",
12         "concurrency": 50,
13         "max_active_tis_per_dag": 20,
14         "max_active_runs": 20,
15         "depends_on_past": False,
16         "start_date": datetime(2016, 1, 2, 19, 30, tzinfo=timezone.utc),
17         "on_failure_callback": slack_fail_alert}
18
19     with DAG(
20         dag_id="youtube_time-based-day",
21         description="DAG to fetch list of Youtube Analytics data for time
22         -based reports",
23         tags=["social_media", "youtube", "content_type"],
24         schedule_interval=timedelta(days=1),
25         default_args=default_args,
26     ) as dag:

```

Listing 4.6: Configuration of the DAG parameters - continued

En effet, c'est une partie du code utilisé en production chez Chefclub. Ce code permet de définir un DAG qui s'exécute **chaque jour à 19h30 UTC**, avec une date de début le 2 janvier 2016 (date de création de Chefclub). Dans la partie suivante, je présenterai le paramétrage des tâches, qui correspond aux actions effectuées par le DAG lors de chaque exécution.

4.2.2.4.2 Paramétrage des Tasks

L'objectif ici est d'itérer sur toute les chaînes et exécuté le code qui va récupérer les informations des chaînes, et lancer l'extraction des données avec l'image stocké sur l'Artifact Registry. Pour se faire en itérant sur toutes les chaînes (fichier de configuration YAML qui contient ces données), on utilise le **KubernetesPodOperator** pour se faire, je rappelle que définir une task sur Airflow, c'est instancier un opérateur, voici le code capable de paramétriser cet opérateur :

```

1      # Define the destination path for the YouTube data
2      destination = f"youtube/{REPORT_NAME}/dt={START_DATE}"
3      fetch_youtube_data = KubernetesPodOperator(
4          **container_constraints(memory=384),
5          image=f"{REGISTRY_IMAGE_NAME}{REGISTRY_IMAGE_TAG}",
6          task_id=FETCH_TASK_ID, # Define The Task Id
7          cmds=["bash", "-cx"], # Execute the command inside the Container
8          arguments=[
9              "python src/main.py"
10             f" -c {channel_id}" # Specify the channel ID
11             f" -r {REPORT_NAME}" # Specify the report name
12             f" -s {START_DATE}" # Specify the start date
13             f" -e {END_DATE}" # Specify the end date
14             f" -d {destination}" # Specify the destination for the data
15             " --to-avro"], # Convert the data to Avro format
16
17             kubernetes_conn_id=None,
18             **YOUTUBE_DAG_CONFIG, # Dag Configuration
19             retry_delay=timedelta(seconds=30),
20             retries=3)

```

Listing 4.7: Configuration of the KubernetesPodOperator

A cette étape, le DAGs a récupérer les données et les stocké dans GCS en format AVRO, maintenant il faut stocké ces données dans une table temporaire dans BigQuery, pour ce faire on utilise l'opérateur `GCSToBigQueryOperator` comme suivant :

```

1 DESTINATION_TABLE_NAME = (
2     REPORT_BIGQUERY_UPSERT_DESTINATION.get("table_name", "")
3 )
4 TEMPORARY_DESTINATION_TABLE_NAME = (
5     f"temp--{channel_name}--{REPORT_NAME}--{START_DATE}"
6         )
7 # Task to transfer the fetched data (saved in GCS)
8 # to a temporary table in BigQuery
9 gcs_to_bq = GCSToBigQueryOperator(
10     dag=dag,
11     task_id=f"{REPORT_NAME_CLEANED}__{channel_name}__tempTableCreation",
12     bucket=YOUTUBE_DAG_CONFIG["env_vars"]["BUCKET_NAME"],
13     source_objects=[f"{destination}/{channel_id}.avro"],
14     destination_project_dataset_table=(
15         f"{BQ_PROJECT}.{BQ_TEMP_DATASET}."
16         f"{TEMPORARY_DESTINATION_TABLE_NAME}"),
17     source_format="AVRO",
18     autodetect=True,
19     create_disposition="CREATE_IF_NEEDED",
20     ignore_unknown_values=True,
21     write_disposition="WRITE_TRUNCATE", # WRITE_APPEND : to append data
22     gcp_conn_id="google_cloud_default",
23     max_active_tis_per_dag=1,
24     deferrable=True,
25     retry_delay=timedelta(seconds=30),
26     retries=5)
```

Listing 4.8: Configuration of the GCSToBigQueryOperator

Maintenant que les données sont stockées dans la table temporaire, je dois effectuer l'opération de fusion-insertion (**Merge Upsert**). Pour cela, j'utilise l'opérateur `BigQueryInsertJobOperator`. Voici le code de paramétrage de cette tâche de **Merge Upsert** :

```

1 # Task to merge and upsert data from a temporary table
2 # into the native BigQuery table.
3 upsert_table = BigQueryInsertJobOperator(
4     task_id=f"{REPORT_NAME_CLEANED}__{channel_name}__mergeUpsert",
5     configuration={
6         "query": {"query": build_query_to_upsert_table(
7             bq_project=BQ_PROJECT,
8             bq_dataset=DESTINATION_DATASET_NAME,
9             bq_temp_dataset=BQ_TEMP_DATASET,
10            temporary_destination_table_name=(
11                TEMPORARY_DESTINATION_TABLE_NAME
12            ),
13            destination_table_name=DESTINATION_TABLE_NAME,
14            report_id=REPORT_NAME,
15        ), "useLegacySql": False}}},
16     location="eu",
```

```

17     deferrable=True,
18     task_concurrency=1,
19     retry_delay=timedelta(seconds=30),
20     retries=5)

```

Listing 4.9: Configuration of the BigQueryInsertJobOperator

Note : Ce code utilise la fonction `build_query_to_upsert_table()` qui génère la requête pour effectuer la fusion (Merge Upinsert). La section 4.2.2.5 suivante explique la requête.

4.2.2.5 Transfert des données de GCS vers BigQuery avec le mécanisme d'Upsert

Maintenant que mes données sont disponibles sur GCS, mon objectif est de les stocker dans une table native dans BigQuery en utilisant la technique Upsert. Voici les étapes à suivre : Tout d'abord, je vais créer une table temporaire dans BigQuery pour charger le fichier AVRO. Cette table temporaire sera utilisée pour l'opération Upsert. Voici un exemple de table temporaire générée à partir d'un fichier AVRO :

Row	FETCHED_AT_UTC	RUN_CHANNEL_ID	RUN_START_DATE	RUN_END_DATE
1	2023-07-03 19:32:53	UCQ_FrN6...	2023-07-01	2023-07-01

FIG. 4.19 : BigQuery Temp Tables used for Upinsert

Il est important de noter que ces tables temporaires sont automatiquement supprimées après 2 jours de leur création, ce qui permet de maintenir l'environnement propre et évite l'accumulation de tables inutilisées.

Une fois que j'ai ma table temporaire, je peux effectuer l'opération Upinsert avec ma table native dans BigQuery en utilisant la requête SQL suivante :

```

1      MERGE
2          `{{bq_project}}.{{bq_dataset}}.{{destination_table_name}}` T
3      USING
4          `{{bq_project}}.{{bq_temp_dataset}}.{{temporary_destination_table_name}}` S
5      ON
6          T.RUN_START_DATE = S.RUN_START_DATE
7      AND
8          T.RUN_CHANNEL_ID = S.RUN_CHANNEL_ID
9      WHEN MATCHED THEN UPDATE SET
10         T.column1 = S.column1, T.column2 = S.column2, ...
11     WHEN NOT MATCHED BY TARGET THEN
12         INSERT (key_column, column1, column2, ...)
13         VALUES (S.key_column, S.column1, S.column2, ...)

```

Listing 4.10: Example of Upinsert Operation using MERGE in BigQuery

Voici à quoi ressemble la table BigQuery native finale :

Chapitre 4. Implémentation et validation de la solution

The screenshot shows the BigQuery interface. On the left, there's an 'Explorer' sidebar with a search bar and a list of datasets and tables under 'chefclub-'. The main area is titled 'Untitled 2' and contains a single query:

```
1 SELECT * FROM `chefclub-...social_media.dim_page_perf_day_yt_non_monetized`
```

Below the query is a 'Query results' table with the following columns: JOB INFORMATION, RESULTS, JSON, EXECUTION DETAILS, and EXECUTION GRAPH. The 'RESULTS' tab is selected. The table has 11 rows, each representing a run with details like 'FETCHED_AT_UTC', 'RUN_CHANNEL_ID', 'RUN_START_DATE', and 'RUN_END_DATE'.

Row	FETCHED_AT_UTC	RUN_CHANNEL_ID	RUN_START_DATE	RUN_END_DATE
1	2023-07-04 19:32:58	UC6Uq0V9HZ...	2023-07-02	2023-07-02
2	2023-07-04 19:31:45	UCAbJyKJ_xV...	2023-07-02	2023-07-02
3	2023-07-04 19:31:44	UCDneLZVnbt...	2023-07-02	2023-07-02
4	2023-07-04 19:32:15	UCE_I57aXpG0...	2023-07-02	2023-07-02
5	2023-07-04 19:33:15	UCG7baC-AN8...	2023-07-02	2023-07-02
6	2023-07-04 19:32:56	UCHBQWuBWj...	2023-07-02	2023-07-02
7	2023-07-04 19:32:25	UCQ_Frn6FNV...	2023-07-02	2023-07-02
8	2023-07-04 19:31:41	UCS57SkOy7X...	2023-07-02	2023-07-02
9	2023-07-04 19:32:23	UCV1m3ef2Z1...	2023-07-02	2023-07-02
10	2023-07-04 19:31:44	UCYWc0Da0A2...	2023-07-02	2023-07-02
11	2023-07-04 19:32:17	UCaebTJT-hG5...	2023-07-02	2023-07-02

FIG. 4.20 : BigQuery Native Production Table for YouTube Analytics

4.2.2.6 Orchestration du DAG avec Airflow et Kubernetes

Pour l'orchestration de mon code, j'utilise Airflow conjointement avec Kubernetes. Airflow est responsable de la gestion des flux de travail et de l'ordonnancement des tâches, tandis que Kubernetes s'occupe de l'exécution et de la mise à l'échelle des conteneurs.

Maintenant que mon code du DAG est prêt à être utilisé, je dois l'envoyer à Airflow version GCP. Pour ce faire, je pousse le code vers un dépôt GitHub. Airflow scanne ensuite le code et affiche le DAG sur l'interface, comme illustré dans la figure 4.21 ci-dessous :

DAGs

The screenshot shows the Airflow 'DAGs' page. It lists several DAGs with their status (All 5, Active 5, Paused 0), owner (airflow), and last run information. One DAG, 'youtube_time-based-day', is highlighted with a red box.

DAG	Owner	Last Run	Next Run
facebook	airflow	2023-07-04, 06:30:00	2023-07-05, 06:30:00
snapshot-analytics-stories	airflow	2023-07-04, 08:00:00	2023-07-05, 08:00:00
snapshot-revenue	airflow	2023-07-04, 06:00:00	2023-07-05, 06:00:00
update_catalog_bigquery_views	airflow	2023-07-05, 08:00:00	2023-07-05, 12:00:00
youtube_time-based-day	airflow	2023-07-03, 19:30:00	2023-07-04, 19:30:00

FIG. 4.21 : Airflow User Interface - YouTube time based DAG

Comme vous pouvez le voir, le DAG YouTube `monitized` est présent sur l'interface utilisateur d'Airflow, et j'ai les paramètres que j'ai codés, tels que l'exécution quotidienne à 19h30 UTC.

On trouve également une multitude d'informations de suivi des DAGs, ainsi qu'une représentation graphique des DAGs. La figure 4.22 suivante montre le graphique des tâches du DAG «monitized», qui présente les trois tâches :

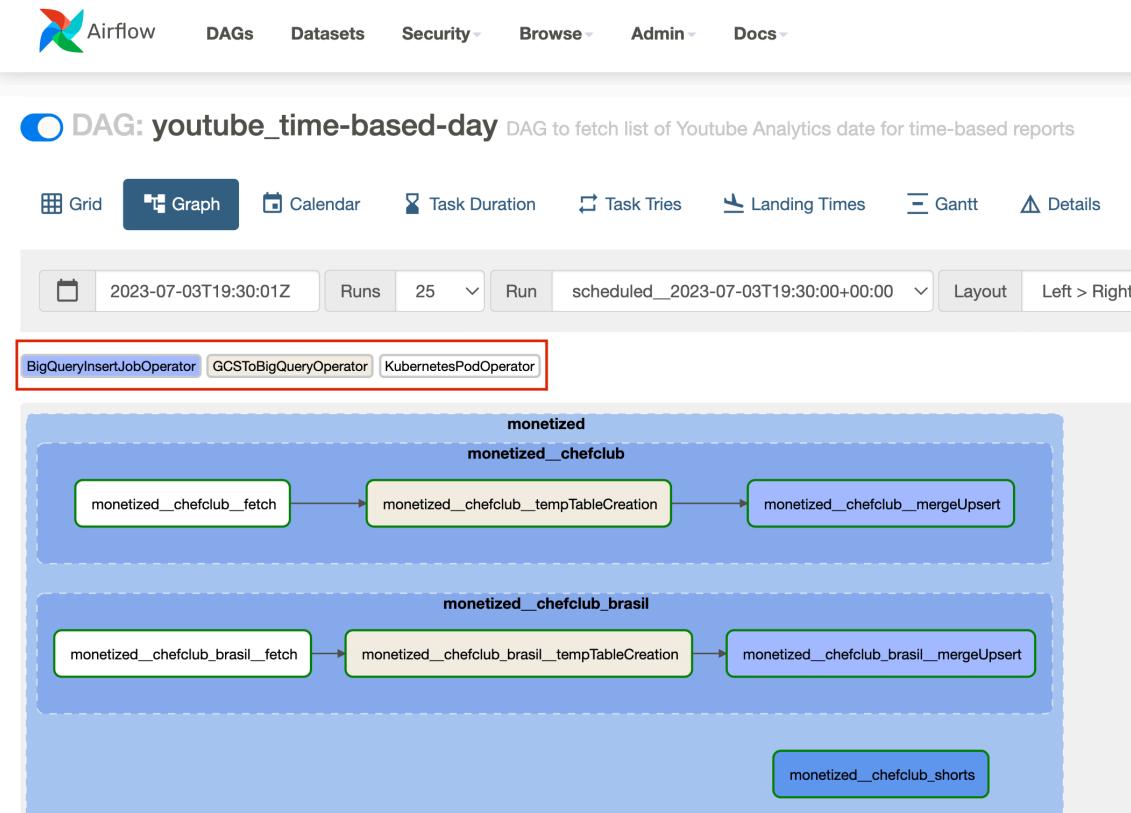


FIG. 4.22 : Task Graph of the monetized DAG in Airflow

Note : En annexe, vous trouverez le code du DAG YouTubeAnalyticsTimeBasedDAG qui est responsable de la génération de ces DAGs.

4.2.2.7 Test et Validation du DAG sur Airflow

L'environnement DAG avec Airflow Kubernetes est utilisé en production pour l'orchestration de mon travail. Airflow offre plusieurs outils intégrés qui permettent de surveiller les jobs en cours. Cela me permet d'être notifié en cas d'erreur. J'ai également ajouté une notification Airflow sur Slack qui m'envoie un message en cas d'erreur du DAG, comme illustré dans la figure suivante :

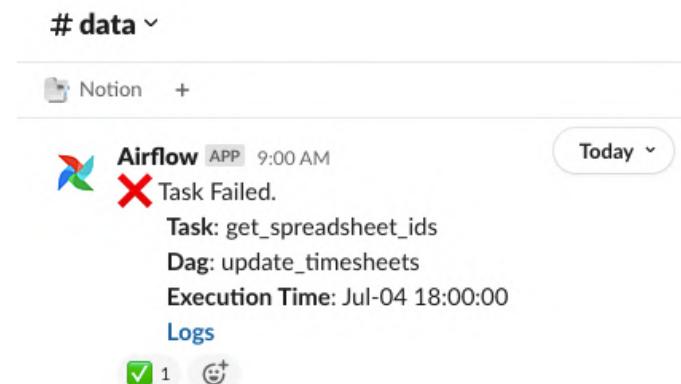


FIG. 4.23 : Airflow Error Notification on Slack

En plus de cela, j'ai accès aux logs, à la grille d'exécution, aux tentatives de tâches, etc. La figure

4.24 suivante montre la grille d'exécution sur Airflow :

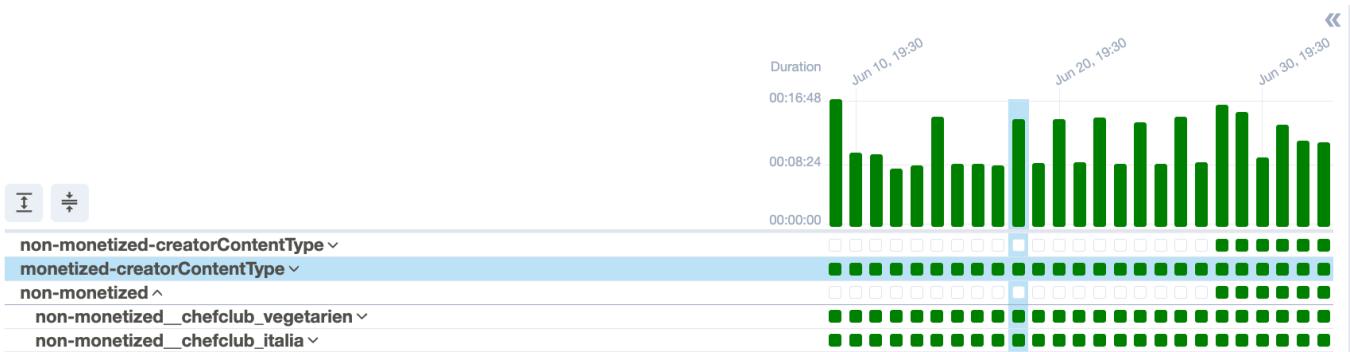


FIG. 4.24 : Execution Grid on Airflow - YouTube Job

Voici dans la figure 4.25 un graphique dans Airflow qui présente la durée d'exécution des tâches :

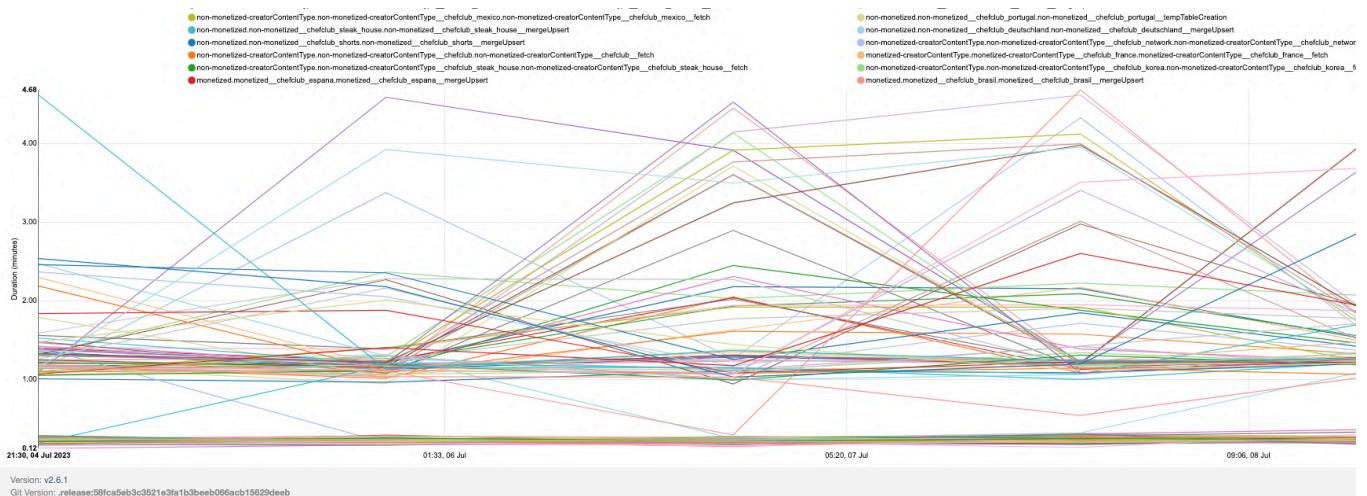


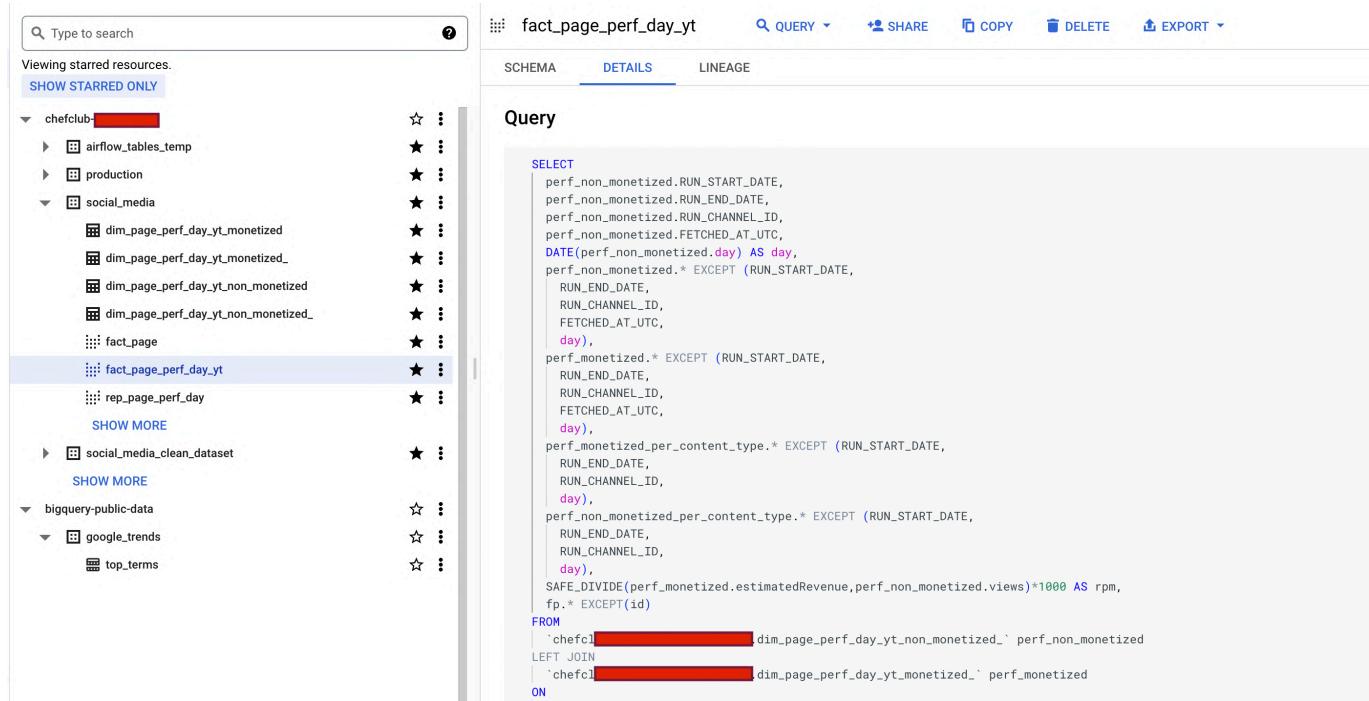
FIG. 4.25 : Task duration Graph on Airflow - YouTube Job

Dans cet environnement, je bénéficie d'une visibilité accrue sur l'exécution du DAG grâce à un système de surveillance avancé. Je peux facilement surveiller l'état des Tasks en temps réel, recevoir des alertes sur Slack en cas de problème et consulter logs détaillés pour comprendre ce qui s'est passé lors de l'exécution. Ce qui permet de prendre rapidement des mesures pour résoudre les problèmes et maintenir les Jobs. C'est grâce à ces fonctionnalités que Airflow se révèle être un outil efficace pour la gestion et l'orchestration des workflows !

4.2.2.8 Crédit de rapports sur Google Sheets et Looker

Note : Le code développé offre principalement de la flexibilité. En d'autres termes, n'importe quel utilisateur peut l'utiliser pour personnaliser la recherche des rapports en définissant les paramètres dans le fichier de configuration. Il suffit ensuite d'ajouter ce code au DAG tel qu'ilustré dans la figure `YouTubeAnalyticsTimeBasedDAG` en Annexe A.1 et de lancer l'orchestration. Airflow se chargera automatiquement de récupérer les données spécifiées. Ensuite, il ne reste plus qu'à lier ces données avec *Looker Studio* ou *Google Sheets*, comme je le montre ici pour le rapport financier dans cette sous-section.

Maintenant que mon code est capable d'ajouter quotidiennement les données analytiques de performance des chaînes YouTube, j'utilise ces données pour alimenter des rapports sur Google Sheets. Pour ce faire, j'effectue des agrégations et des jointures sur les tables de rapports que je récupère, ainsi que sur les données internes des tables. La figure suivante présente un exemple de requête SQL que j'exécute automatiquement chaque matin pour rafraîchir les tables avec les données de la veille :



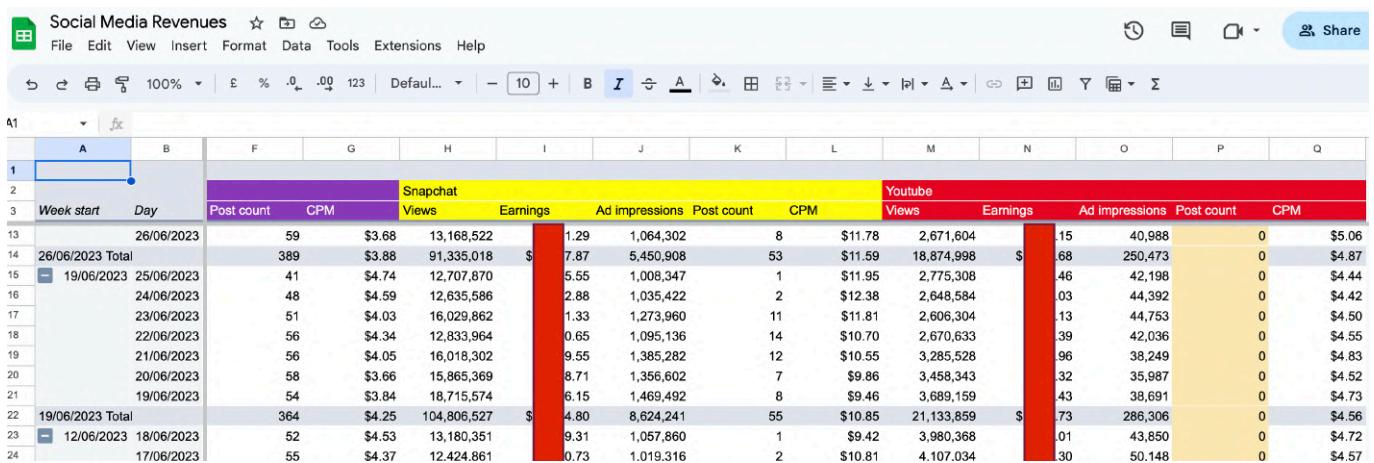
```

SELECT
    perf_non_monetized.RUN_START_DATE,
    perf_non_monetized.RUN_END_DATE,
    perf_non_monetized.RUN_CHANNEL_ID,
    perf_non_monetized.FETCHED_AT_UTC,
    DATE(perf_non_monetized.day) AS day,
    perf_non_monetized.* EXCEPT (RUN_START_DATE,
    RUN_END_DATE,
    RUN_CHANNEL_ID,
    FETCHED_AT_UTC,
    day),
    perf_monetized.* EXCEPT (RUN_START_DATE,
    RUN_END_DATE,
    RUN_CHANNEL_ID,
    day),
    perf_monetized_per_content_type.* EXCEPT (RUN_START_DATE,
    RUN_END_DATE,
    RUN_CHANNEL_ID,
    day),
    perf_non_monetized_per_content_type.* EXCEPT (RUN_START_DATE,
    RUN_END_DATE,
    RUN_CHANNEL_ID,
    day),
    SAFE_DIVIDE(perf_monetized.estimatedRevenue,perf_non_monetized.views)*1000 AS rpm,
    fp.* EXCEPT(id)
FROM
    `chefclub[REDACTED]`.dim_page_perf_day_yt_non_monetized` perf_non_monetized
LEFT JOIN
    `chefclub[REDACTED].dim_page_perf_day_yt_monetized` perf_monetized
ON

```

FIG. 4.26 : YouTube analytics query for performing **table joins** in BigQuery

Grâce à la table résultante (Fig. 4.26) et au connecteur *Data connector* entre **BigQuery** et **Google Sheets**, on génère des rapports similaires à celui présenté dans la figure 4.27 suivante :



				Snapshot				Youtube							
				Post count	CPM	Views	Earnings	Ad Impressions	Post count	CPM	Views	Earnings	Ad Impressions	Post count	CPM
13		26/06/2023		59	\$3.68	13,168,522	\$1.29	1,064,302	8	\$11.78	2,671,604	\$1.15	40,988	0	\$5.06
14		26/06/2023 Total		389	\$3.88	91,335,018	\$7.87	5,450,908	53	\$11.59	18,874,998	\$6.68	250,473	0	\$4.87
15		19/06/2023 25/06/2023		41	\$4.74	12,707,870	5.55	1,008,347	1	\$11.95	2,775,308	.46	42,198	0	\$4.44
16		24/06/2023		48	\$4.59	12,635,586	2.88	1,035,422	2	\$12.38	2,648,584	.03	44,392	0	\$4.42
17		23/06/2023		51	\$4.03	16,029,862	1.33	1,273,960	11	\$11.81	2,606,304	.13	44,753	0	\$4.50
18		22/06/2023		56	\$4.34	12,833,964	0.65	1,095,136	14	\$10.70	2,670,633	.39	42,036	0	\$4.55
19		21/06/2023		56	\$4.05	16,018,302	9.55	1,385,282	12	\$10.55	3,285,528	.96	38,249	0	\$4.83
20		20/06/2023		58	\$3.66	15,865,369	8.71	1,356,602	7	\$9.86	3,458,343	.32	35,987	0	\$4.52
21		19/06/2023		54	\$3.84	18,715,574	6.15	1,469,492	8	\$9.46	3,689,159	.43	38,691	0	\$4.73
22		19/06/2023 Total		364	\$4.25	104,806,527	\$4.80	8,624,241	55	\$10.85	21,133,859	\$7.73	286,306	0	\$4.56
23		12/06/2023 18/06/2023		52	\$4.53	13,180,351	9.31	1,057,860	1	\$9.42	3,980,368	.01	43,850	0	\$4.72
24		17/06/2023		55	\$4.37	12,424,861	0.73	1,019,316	2	\$10.81	4,107,034	.30	50,148	0	\$4.57

FIG. 4.27 : Financial Report Sample

De plus, grâce à la connexion native entre **Looker Studio** et **Google Sheets**, nous pouvons générer des rapports interactifs Looker similaires à celui illustré dans la figure 4.28, qui représente un exemple de travail sur Youtube dans Looker Studio.

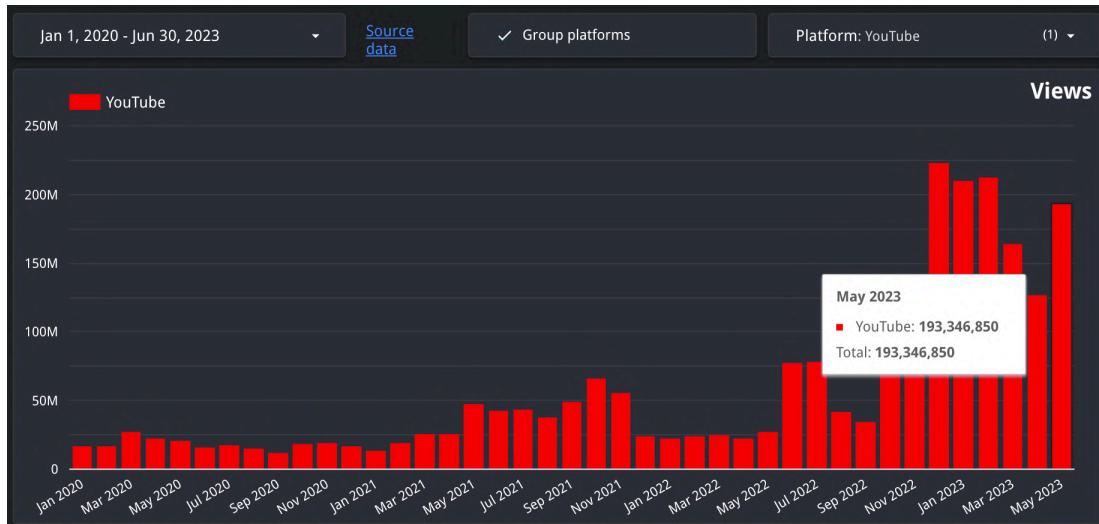


FIG. 4.28 : Looker Studio UI - YouTube Analytics interactive Report

4.2.2.9 Test de la crédibilité des données

Afin de garantir la crédibilité des données manipulées par notre solution, j'ai effectué plusieurs tests au sein de notre équipe Data. Tout d'abord, j'ai comparé les résultats affichés sur les tables générées par notre solution avec les résultats obtenus dans YT Studio. La figure 4.29 suivante illustre un exemple de l'interface de YT Studio :

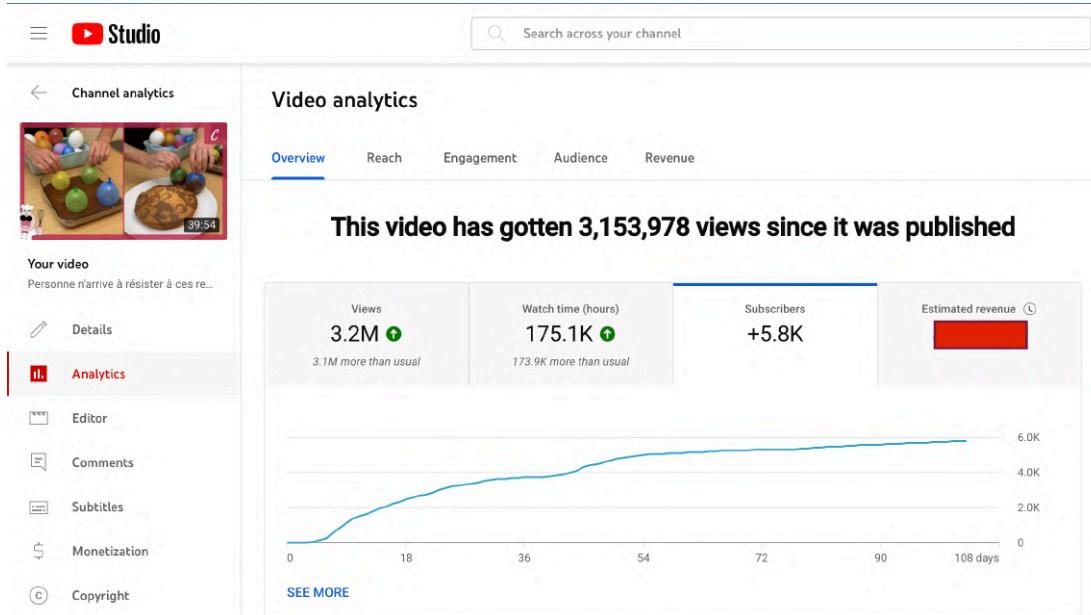


FIG. 4.29 : Example of YT Studio Interface

En effectuant cette comparaison, j'ai vérifié la cohérence entre les résultats obtenus par ma solution et ceux affichés dans YT Studio.

Un autre test que j'ai réalisé est la demande de la marge d'erreur entre les rapports financiers établis par les équipes de la finance et les résultats obtenus sur les tables générées par ma solution.

En effectuant ces tests de crédibilité des données, je me suis assuré de la fiabilité de ma solution de Data Engineering/DataOps.

4.3 Analyse des performances des Posts sur Facebook

Dans cette partie, je vais me concentrer sur l'analyse des données historiques de revenus/vues en fonction du nombre de publications des pages **Chefclub** (<https://www.facebook.com/Chefclub.tv>) et **Chefclub Network** (<https://www.facebook.com/chefclubnetworkofficial>) sur Facebook. Mon objectif principal est de comprendre comment les différentes publications influencent mes performances économiques globales. Pour cela, je vais comparer mes performances en termes de revenus et de vues afin de déterminer quel type de publication est le plus performant. De plus, je vais également évaluer la fréquence optimale de publication par semaine pour maximiser les revenus. la figure 4.30 suivante rappel la conception de la solution que nous avons mis en place lors de l'analyse et la conception :

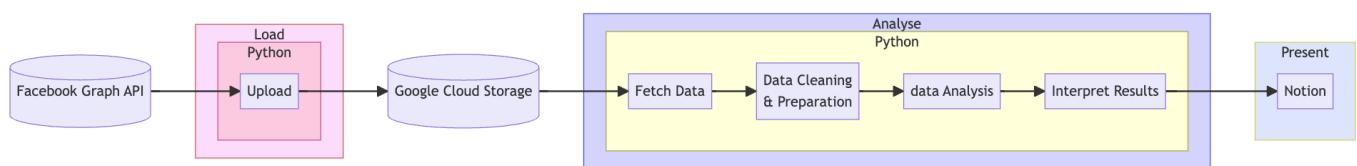


FIG. 4.30 : Data Analysis Workflow for Facebook Posts (Reminder)

Note : Cette section revêt une importance capitale dans le développement de notre modèle, comme décrit dans la section 4.3.4, ainsi que pour l'ingénierie des caractéristiques (*Feature Engineering*) dans la partie 4.3.4.3.2. Elle joue également un rôle clé dans la présentation des insights et des découvertes majeures, telles qu'abordées dans la section 4.3.3.

4.3.1 Contexte, Terminologie et Illustration

4.3.1.1 Définition des termes

Afin d'établir une compréhension commune, voici les termes clés que j'utilise tout au long de notre analyse :

- **Post** : Il s'agit des vidéos produites par Chefclub et publiées sur Facebook. Les posts se présentent sous deux formes : les «Tested-Posts» et les «Untested-Posts».
 - **Tested-Post** : Ce sont les publications qui ont été soumises au test A/B de Meta. Elles comprennent une série de 2 à 4 vidéos variantes.
 - **Untested-Post** : Il s'agit des publications *normales* qui n'ont pas été soumises au test A/B de Meta. Elles sont publiées telles quelles, sans variations.
- **Video** : Il s'agit des vidéos affichées sur le mur Facebook, qu'elles proviennent des **Tested-Posts** ou des **Untested-Posts**.

4.3.1.2 Mise en situation

Chefclub utilise la plateforme Facebook pour diffuser son contenu et interagir avec son audience. Deux types de publications sont réalisés : les **Tested-Posts** et les **Untested-Posts**. Les

Tested-Posts consistent en une série de 2 à 4 vidéos variantes soumises à un *test A/B par Meta*. L'objectif de ce test est de déterminer la variation qui génère les meilleurs résultats en fonction d'objectifs spécifiques, tels que le nombre de vues ou les revenus.

Pour mener à bien ce processus de test, Chefclub sélectionne des critères (dans notre cas, les revenus) et publie les variantes correspondantes. Ces variantes sont ensuite présentées à différents groupes d'utilisateurs dans le cadre d'un test A/B. La durée de ce test est préalablement définie. Une fois le test terminé, il devient possible de classer les variantes en fonction de leurs performances par rapport à l'objectif spécifique fixé initialement (par exemple, les revenus).

En résumé, l'A/B test de Meta permet de classifier une série de 2 à 4 variantes d'un post en fonction de leurs performances. Ce test est réalisé sur une période donnée et est présenté à différents groupes d'utilisateurs. Le variant gagnant, c'est-à-dire celui qui obtient les meilleurs résultats, sera alors publié.

4.3.1.2.1 Exemple de Tested-Posts et Untested-Posts

L'exemple 4.31 suivant illustre les notions de **Tested-Post**, **Untested-Post** et de **Video** :

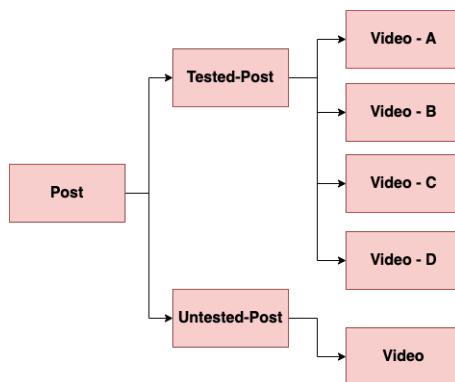


FIG. 4.31 : Flow diagram showing the relationships between Posts and Videos

L'exemple 4.32 suivant montre comment un post est publié (qu'il soit de type **Tested** ou **Untested**) :

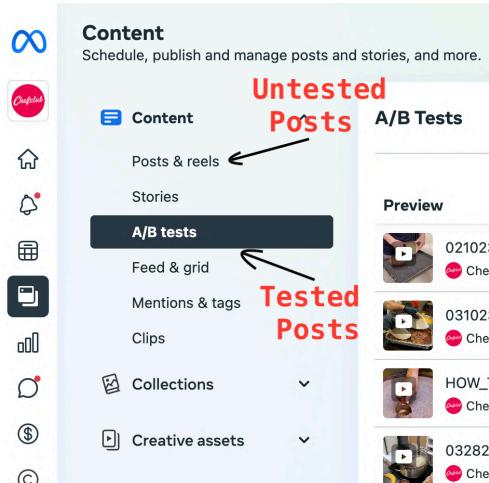


FIG. 4.32 : Example of posting a (Tested or Untested) post.

L'exemple 4.33 suivant illustre les notions de **Tested-Post** et de vidéos soumises au test A/B de

Meta :

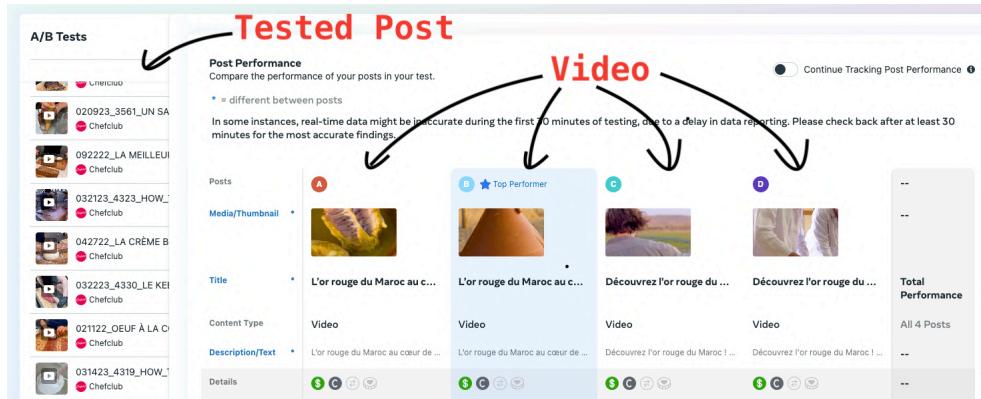


FIG. 4.33 : Example of a *Tested-Post* and *Video* submitted to Meta's A/B testing

4.3.1.3 Explication technique de la problématique

La problématique que je dois résoudre concerne le choix du type de publication à diffuser et la fréquence optimale de publication par semaine. Doit-on privilégier les *Tested-Posts* ou les *Untested-Posts*? Il est important de prendre en compte que les *Tested-Posts* demandent plus d'efforts et de ressources, car ils nécessitent la création de 2 à 4 variantes différentes pour chaque publication.

4.3.2 Développement et Analyse des données

4.3.2.1 Sélection et Présentation des données pour l'analyse

4.3.2.1.1 Sélection de données pour l'analyse

Pour récupérer les données nécessaires à cette analyse, deux choix se sont présentés :

1. Utiliser les données collectées par Airflow à partir de l'API Facebook Graph et stockées dans une base de données PostgreSQL sur le Cloud SQL de Google Cloud Platform (GCP).
2. Récupérer l'ensemble des données via des fichiers CSV depuis l'interface A/B test de Meta.

Après considération, j'ai opté pour la deuxième option. Les fichiers CSV offraient une plus grande quantité de détails par rapport aux tables stockées dans la base de données SQL, qui étaient relativement réduites.

Note : Il est important de noter que la réponse de l'API ne contient que les vidéos (variantes testées et non testées). Le premier défi consiste donc à identifier les *Untested-Posts* (publications non testées) et les *Tested-Posts* (publications testées) à partir de ces vidéos. L'objectif est de comparer les performances des *Tested-Posts* par rapport aux *Untested-Posts*.

4.3.2.1.2 Intervalle de temps des données

J'ai sélectionné l'intervalle de temps **Q1-2023** et les **deux premières semaines d'avril** pour l'analyse. Cette décision est motivée par le fait que l'algorithme de Facebook évolue à une fréquence accélérée, ce qui nécessite de réduire l'intervalle de temps pour tenir compte des changements de comportement dans les données historiques par rapport aux données plus récentes.

Note : Veuillez noter que cette analyse a été réalisée pendant la première semaine de mai, lorsque les publications des deux premières semaines d'avril avaient atteint le nombre maximal de vues. Il est essentiel de noter que *la durée de vie d'un post, en termes de vues, est à deux semaines*.

4.3.2.1.3 Présentation des données brutes

Les données brutes utilisées dans cette analyse comprennent un ensemble d'informations sur les publications et les performances des pages **Chefclub** et **Chefclub Network** sur les réseaux sociaux. Dans notre cas, ces données couvrent la période du premier trimestre de 2023 et les deux premières semaines d'avril.

L'ensemble de données contient plus de 290 colonnes, qui fournissent des détails sur divers aspects des publications, notamment les dates de publication, les types de publications, les audiences cibles, les interactions des utilisateurs, les revenus générés, etc.

Pour vous donner un aperçu des données brutes, voici une représentation visuelle (Fig. 4.34 et Fig.4.36) montrant un extrait du fichier CSV et son affichage sous forme de DataFrame dans Pandas :

	Post ID	Page ID	Page name	Title	Description	Duration (sec)	Publish time	Caption type	Permalink
1	"Post ID",""Page ID"",""Page name"",""Title",""Description,""Duration (sec)"",""Publish time"",""Caption type"",""Permalink"								
2	6933807859977101	169373263087295	Chefclub	"Le saumon poché"	"Le saumon poché ! et ses légumes qui ne boivent				
3	6934172149940672	169373263087295	Chefclub	"L'apéro crostì italiano"	"L'apéro crostì italiano ! Mamma mia				
4	6934172189940668	169373263087295	Chefclub	"L'apéro crostì italiano"	"L'apéro crostì italiano ! Mamma mia				
5	6934172196607334	169373263087295	Chefclub	"L'apéro crostì italiano"	"L'apéro crostì italiano ! Mamma mia				
6	6934172206607333	169373263087295	Chefclub	"L'apéro crostì italiano"	"L'apéro crostì italiano ! Mamma mia				
7	6934884509869436	169373263087295	Chefclub	"3 recettes de gratin à tester pour le dîner"	"3 recettes de gratin à tester pour le dîner"				
8	6934936226530931	169373263087295	Chefclub	"Croque madame au poulet"	"Croque madame au poulet ! Plus besoin				
9	6934936276530926	169373263087295	Chefclub	"Croque madame au poulet"	"Croque madame au poulet ! Plus besoin				
10	6934936326530921	169373263087295	Chefclub	"Croque madame au poulet"	"Croque madame au poulet ! Plus besoin				
11	6934936329864254	169373263087295	Chefclub	"Croque madame au poulet"	"Croque madame au poulet ! Plus besoin				

FIG. 4.34 : Overview of raw data in the CSV file

Entrée [5]:	1	post_perf_lt.head(3)					
Out[5]:							
	Post ID	Page ID	Page name	Title	Description	Duration (sec)	Publish time
0	6798052036886018	169373263087295	Chefclub	Le donut Scream	Le donut Scream ! Revivez la dernière sortie c...	198	03/31/2023 09:03
1	6797575543600334	169373263087295	Chefclub	La baguette cocotte à la poêle	La baguette cocotte à la poêle ! et ses croust...	182	03/31/2023 05:03
2	6797571253600763	169373263087295	Chefclub	Les pommes de terre de Pâques	Les pommes de terre de Pâques ! On part à la c...	58	03/31/2023 04:03

FIG. 4.35 : Overview of raw data in a Pandas DataFrame

4.3.2.2 Préparation des données

Maintenant que j'ai réussi à définir l'objectif, les données à utiliser et l'intervalle de temps pertinent, je dois préparer ces données afin de pouvoir les utiliser dans nos graphiques visuels. **Il est important de noter que l'API de Facebook ne fournit que des vidéos et ne les étiquette pas en fonction de leur provenance (post testé ou non testé).** Nous devons donc trouver un moyen de labéliser nos données.

4.3.2.2.1 Data preprocessing

Dans cette phase, j'ai effectué plusieurs tâches visant à améliorer, restructurer et corriger les données. Les étapes suivantes ont été suivies :

- Nettoyage des noms de colonnes (*data cleaning*) : J'ai vérifié et nettoyé les noms des colonnes afin d'assurer leur cohérence et leur lisibilité. J'ai corrigé toute irrégularité ou erreur dans les noms des colonnes.
- Réorganisation des colonnes et suppression des colonnes non nécessaires (*data wrangling*) : J'ai réorganisé les colonnes pour les aligner selon un ordre logique et cohérent. J'ai supprimé les colonnes qui n'étaient pas pertinentes pour l'analyse, afin de simplifier le jeu de données.
- Conversion des types de données des colonnes (*data type handling*) : J'ai vérifié les types de données des différentes colonnes et les ai modifiés si nécessaire. Cela garantit que chaque colonne est attribuée au type de données approprié, facilitant ainsi les opérations ultérieures sur les données.

Le code 4.36 suivant présente le code utilisé pour effectuer le prétraitement des données :

Analysing Facebook performance table post_perf_lt

Cleaning the Column names

```
Entrée [8]: 1 def rename_column(column_name):
2     """
3         Convert column name to desired format.
4     """
5     column_name = column_name.strip().lower().replace(' ', '_').replace(',', '').replace('(', '').replace(')', '')
6     if column_name.endswith('_'):
7         column_name = column_name[:-1]
8     return column_name
9
10 # apply the function to all column names and assign the new names to the dataframe
11 post_perf_lt.columns = [rename_column(col) for col in post_perf_lt.columns]
```

Reordering Columns and Dropping unnecessary ones

```
Entrée [9]: 1 # Define the columns to keep
2 columns_to_keep = ['publish_time', 'page_name', 'title', 'estimated_earnings_usd', '3-second_video_views']
3 # Filter the dataframe and store the result in a new variable
4 post_perf_lt = post_perf_lt[columns_to_keep]
```

Converting the columns datatypes

```
Entrée [10]: 1 # Convert 'dbdate' columns to 'datetime' with Paris timezone
2 post_perf_lt[['publish_time']] = post_perf_lt[['publish_time']].apply(lambda x: pd.to_datetime(x))
3 post_perf_lt['publish_time'] = post_perf_lt['publish_time'].dt.floor('D')
```

FIG. 4.36 : Data Preprocessing in Jupyter Notebook

4.3.2.2.2 Data labelling

Les données fournies ne comportaient pas d'étiquettes (*label*) indiquant si elles avaient été testées ou non. Pour résoudre ce Challenge, j'ai effectué une opération de regroupement (**group by**) sur les colonnes ***title*** et ***publish_time***. J'ai utilisé cette méthode pour détecter les variants des posts testés et ajouter des étiquettes aux données en fonction de leurs similitudes et différences.

Explication : En d'autres termes, si je trouve dans mon ensemble de données plus de deux vidéos avec le même titre et la même date de création, cela signifie qu'il s'agit d'un post testé. Sinon, il s'agit d'un post non testé. Cette approche nous permettra de labéliser nos données en fonction de ces critères.

Note : Une information importante à souligner est l'ajout d'une nouvelle *feature* : ***total_estimated_earnings*** a été ajoutée à toutes les vidéos. Cette *feature* a été incluse pour toutes les vidéos et calculée selon la formule mathématique suivante :

$$\text{total_estimated_earnings} = \begin{cases} \text{estimated_earnings}, & \text{si le } post \text{ n'est pas testé} \\ \sum \text{variant}(\text{estimated_earnings}), & \text{si le } post \text{ est testé} \end{cases}$$

Remarque et Justification : À la fin du *Test A/B*, la variante gagnante en termes de revenu estimée sera sélectionnée pour publication. Je compare actuellement les *Untested-Posts* avec les *Tested-Posts*, en particulier en comparant les *Untested-Posts* avec le *Winner-Variant*. J'ai introduit la feature ***total_estimated_earnings*** pour prendre en compte les revenus des autres variantes qui ont achevé la phase de test dans le cadre du Test A/B. Ainsi, je pourrai comparer les performances des publications testées et des publications non testées en utilisant la métrique ***total_estimated_earnings***, tout en tenant compte des revenus des variantes qui ont terminé la phase de test. Par le même analogie j'ai ajouter ***total_test-3-second_video_views***

Voici le code Python utilisé pour la labellisation :

```

1 # Group the data by title and publish time
2 for (title, publish_time), group in post_perf_lt.groupby(["title", "publish_time"]):
3
4     # Sort the group by estimated earnings in descending order
5     sorted_group_by_earnings = group.sort_values("estimated_earnings_usd", ascending=False)
6
7     # Sort the group by 3-second video views in descending order
8     sorted_group_by_views = group.sort_values("3-second_video_views", ascending=False)
9
10    # Marking "tested" and "non-tested" posts
11    if len(group) > 1:
12        post_perf_lt.loc[sorted_group_by_earnings.index, "is_tested"] = True
13    else:
14        post_perf_lt.loc[sorted_group_by_earnings.index, "is_tested"] = False
15
16    # Add a new column called "test_group_size" that shows the total number of
17    # elements in the group
18    post_perf_lt.loc[sorted_group_by_earnings.index, "test_group_size"] = len(group)
19
20    # Add a new column called "rank_earnings" that indicates the ranking of each
21    # element in the group
22    post_perf_lt.loc[sorted_group_by_earnings.index, "rank_earnings"] =
23    sorted_group_by_earnings[

```

```

20     "estimated_earnings_usd"].rank(method="min", ascending=False)
21
22 # Add a new column called "rank_views" that indicates the ranking of each element
23 # in the group
23 post_perf_lt.loc[sorted_group_by_views.index, "rank_views"] =
24 sorted_group_by_views["3-second_video_views"].rank(
25     method="min", ascending=False)
26
26 ### Calculating "total_estimated_earnings_usd" based on "3-second_video_views"
27 for the group of tested posts
27 post_perf_lt.loc[sorted_group_by_views.index, "total_test-3-second_video_views"] =
28 sorted_group_by_views[
28     "3-second_video_views"].sum()
29
30 ### Calculating "total_estimated_earnings_usd" based on "estimated_earnings_usd"
31 for the group of tested posts
31 post_perf_lt.loc[sorted_group_by_earnings.index, "total_estimated_earnings_usd"] =
32 sorted_group_by_earnings[
32     "estimated_earnings_usd"].sum()
33

```

Listing 4.11: Data Labeling - Tested and Untested Posts

4.3.2.2.3 Data transformation et agrégation

Dans cette partie, je réalise l'agrégation et la transformation de mes données. L'idée est d'abord de filtrer les posts testés et les posts non testés à partir du dataframe qui contient les vidéos labélisées. Cependant, avant cela, je divise le dataset par page, car chaque page a son propre auditoire et son propre comportement. Voici le code que j'utilise pour effectuer le splitting :

Splitting dataframes by page page

```

Entrée [14]: 1 # Split the dataset into two based on the values in the 'page_name' column
2 post_perf_lt_chefclub = post_perf_lt[post_perf_lt['page_name'] == 'Chefclub']
3 post_perf_lt_chefclub_network = post_perf_lt[post_perf_lt['page_name'] == 'Chefclub Network']

```

FIG. 4.37 : Splitting the dataframe by page in Jupyter Notebook

Rappel : Parmi les posts testés, seul le variant gagnant sera publié (dans notre cas, le variant qui a obtenu le plus de vues). Ainsi, je compare les performances lifetime des variantes gagnantes avec celles des posts non testés.

Note : Pour le reste de cette analyse dans ce rapport, je ne traiterai que les données de la page *Chefclub Network*.

Pour ce faire, le code suivant permet d'extraire du dataset trois jeux de données :

- **Posts dataset** : contient les variantes gagnantes et les posts non testés.
- **Tested-Posts dataset** : contient uniquement les variantes gagnantes.
- **Untested-Posts dataset** : contient uniquement les posts non testés.

Ces jeux de données sont indexés par la variable temporelle *publish-time*. La figure suivante montre le code :

Chapitre 4. Implémentation et validation de la solution

```
Filtering Posts , Tested Posts , Non-Tested Posts and Videos datasets
```

Entrée [19]:

```
1 # Posts
2 post_perf_lt_chefclub_network_posts = post_perf_lt_chefclub_network[(post_perf_lt_chefclub_network["rank_views"] == 1) & (post_perf_lt_chefclub_network["is_tested"] == True)]
3 post_perf_lt_chefclub_network_posts = post_perf_lt_chefclub_network_posts.set_index("publish_time", inplace=True)
4 post_perf_lt_chefclub_network_posts.rename(columns={"total_test-3-second_video_views": "3-second_video_posts", "total_estimated_earnings_usd": "estimated_earnings_usd_posts"}, inplace=True)
```

Entrée [20]:

```
1 # Tested Posts
2 post_perf_lt_chefclub_network_tested_posts = post_perf_lt_chefclub_network[(post_perf_lt_chefclub_network["is_tested"] == True) & (post_perf_lt_chefclub_network["rank_views"] == 1)]
3 post_perf_lt_chefclub_network_tested_posts = post_perf_lt_chefclub_network_tested_posts.set_index("publish_time", inplace=True)
4 post_perf_lt_chefclub_network_tested_posts = post_perf_lt_chefclub_network_tested_posts.rename(columns={"3-second_video_views": "3-second_video_views_tested_posts", "estimated_earnings_usd": "estimated_earnings_usd_tested_posts"}, inplace=True)
```

Entrée [21]:

```
1 # Non-Tested Posts
2 post_perf_lt_chefclub_network_non_tested = post_perf_lt_chefclub_network[post_perf_lt_chefclub_network["is_tested"] == False]
3 post_perf_lt_chefclub_network_non_tested = post_perf_lt_chefclub_network_non_tested.set_index("publish_time", inplace=True)
4 post_perf_lt_chefclub_network_non_tested.rename(columns={"3-second_video_views": "3-second_video_views_non-tested_posts", "estimated_earnings_usd": "estimated_earnings_non-tested_posts"}, inplace=True)
```

Entrée [22]:

```
1 # Videos
2 post_perf_lt_chefclub_network_all_videos = post_perf_lt_chefclub_network.set_index("publish_time", inplace=True)
3 post_perf_lt_chefclub_network_all_videos = post_perf_lt_chefclub_network_all_videos.rename(columns={"3-second_video_views": "3-second_video_views_videos", "estimated_earnings_usd": "estimated_earnings_videos"}, inplace=True)
```

FIG. 4.38 : Dataset extraction by filtering the original dataset

Le code suivant permet de calculer le nombre de vues par jour en fonction de la nature du post :

Entrée [15]:

```
1 # Filter the dataframe to include only rows where rank_earnings=1 and is_tested=true
2 post_perf_lt_chefclub_network_tested_winners = post_perf_lt_chefclub_network[(post_perf_lt_chefclub_network["rank_earnings"] == 1) & (post_perf_lt_chefclub_network["is_tested"] == True)]
3
4 # Calculate the total number of 3-second views per day for the filtered dataframe
5 views_per_day_tested_winners = \
6     post_perf_lt_chefclub_network_tested_winners.groupby(
7         post_perf_lt_chefclub_network_tested_winners["publish_time"].dt.date)[
8             "3-second_video_views"].sum().reset_index()
```

Entrée [16]:

```
1 post_perf_lt_chefclub_network_untested = post_perf_lt_chefclub_network[(post_perf_lt_chefclub_network["rank_earnings"] == 1) & (post_perf_lt_chefclub_network["is_tested"] == False)]
2
3 # Calculate the total number of 3-second views per day for the untested dataframe
4 views_per_day_untested = \
5     post_perf_lt_chefclub_network_untested.groupby(post_perf_lt_chefclub_network_untested["publish_time"].dt.date)[
6         "3-second_video_views"].sum().reset_index()
```

FIG. 4.39 : Views per day per post nature

Note : J'arrive maintenant à la partie la plus importante de mon code. Je dois comparer des variables discrètes en fonction du temps, en utilisant un intervalle de temps d'une semaine. Ce choix est plus parlant pour les équipes de Chefclub. La comparaison du nombre de posts publiés par semaine est plus simple à gérer que le nombre de posts par jour.

Le code suivant permet de calculer la **somme**, la **moyenne**, la **médiane** et le **nombre de posts par semaine** :

```
Resample data to weekly frequency and compute various statistics for each column

Entrée [23]: 1 # Define the aggregation metrics to compute
2 agg_metrics = ['sum', 'mean', 'median', 'count']
3 # Resample the four dataframes by day and compute the aggregation metrics
4 weekly_tested_posts_chefclub_network = post_perf_lt_chefclub_network_tested_posts.resample('W').agg(agg_metrics)
5 weekly_non_tested_chefclub_network = post_perf_lt_chefclub_network_non_tested.resample('W').agg(agg_metrics)
6 weekly_all_videos_chefclub_network = post_perf_lt_chefclub_network_all_videos.resample('W').agg(agg_metrics)
7 weekly_posts_chefclub_network = post_perf_lt_chefclub_network_posts.resample('W').agg(agg_metrics)

Entrée [24]: 1 # Merge the four resampled dataframes into a single dataframe using their index
2 weekly_summary_chefclub_network = (
3     weekly_tested_posts_chefclub_network
4     .merge(weekly_non_tested_chefclub_network, left_index=True, right_index=True)
5     .merge(weekly_all_videos_chefclub_network, left_index=True, right_index=True)
6     .merge(weekly_posts_chefclub_network, left_index=True, right_index=True)
7 )

Entrée [25]: 1 # Calculate the percentage of tested posts out of the total posts for each day and add it as a new column
2 weekly_summary_chefclub_network['tested_posts_ratio'] = weekly_summary_chefclub_network.apply(
3     lambda row: row['3-second_video_views_tested_posts']['count'] / row['3-second_video_posts', 'count']) * 100,
4

Entrée [26]: 1 weekly_summary_chefclub_network.head(3)

Out[26]:
   3-second_video_views_tested_posts  estimated_earnings_usd_tested_posts  3-second_video_views_non-tested_posts  estimated_earnings_non
   sum      mean      median    count    sum      mean      median    count    sum      mean      median    count    sum      mean      mean
publish_time
2023-01-08  15387808  2.198258e+06  1774418.0       7  5276.15  753.735714  147.17       7  103631970  3.701142e+06  354034.5     28  9190.79  328.2425  134.
2023-01-15  32484093  2.165606e+06  1240253.0      15  2859.30  190.620000  86.94      15  16054587  7.297540e+05  387814.0     22  7294.74  364.7370  195.
2023-01-22  36282540  2.418836e+06  528479.0      15  7050.86  470.057333  38.56      15  10674781  7.116521e+05  252948.0     15  4492.29  299.4860  130.
```

FIG. 4.40 : Mean, sum, and count per week

Dans la section suivante, j'expliquerai comment j'utiliserai ce dataframe présenté dans la figure 4.40 pour obtenir des insights.

4.3.2.3 Analyse et Interprétation des données

Maintenant que j'ai préparé mon jeu de données, je vais aborder l'analyse et la visualisation.

La première chose que j'ai examinée est l'évolution temporelle des publications testées (Tested-winners Posts) par rapport aux publications non testées (Untested Posts). La figure suivante présente ce graphe :

Total 3-second video views of `Chefclub Network` posts published on the same day

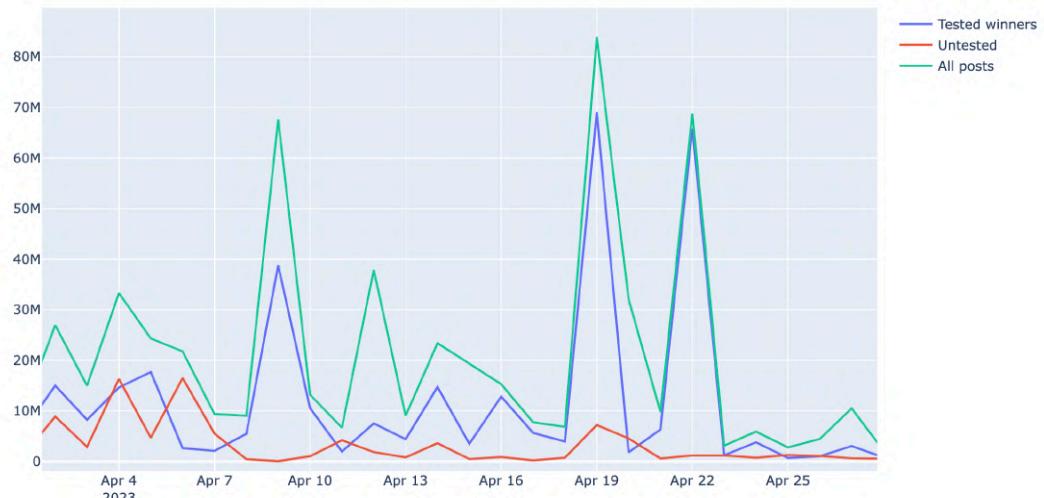


FIG. 4.41 : Lineplot -Total Views per Day by Post Nature

Cette figure montre l'évolution de la somme cumulative des publications testées et non testées au fil du temps. La courbe bleue représente les publications testées gagnantes (*Tested Winners*), qui est au-dessus de la courbe rouge représentant les publications non testées (*Untested Posts*). La courbe verte est la somme des deux courbes précédentes. À première vue, on peut émettre l'hypothèse que les publications testées génèrent plus de vues que les publications non testées. Cependant, pour vérifier cette hypothèse, il est nécessaire d'approfondir l'analyse.

La figure suivante présente le nombre de publications par semaine pour les différents types de publications :

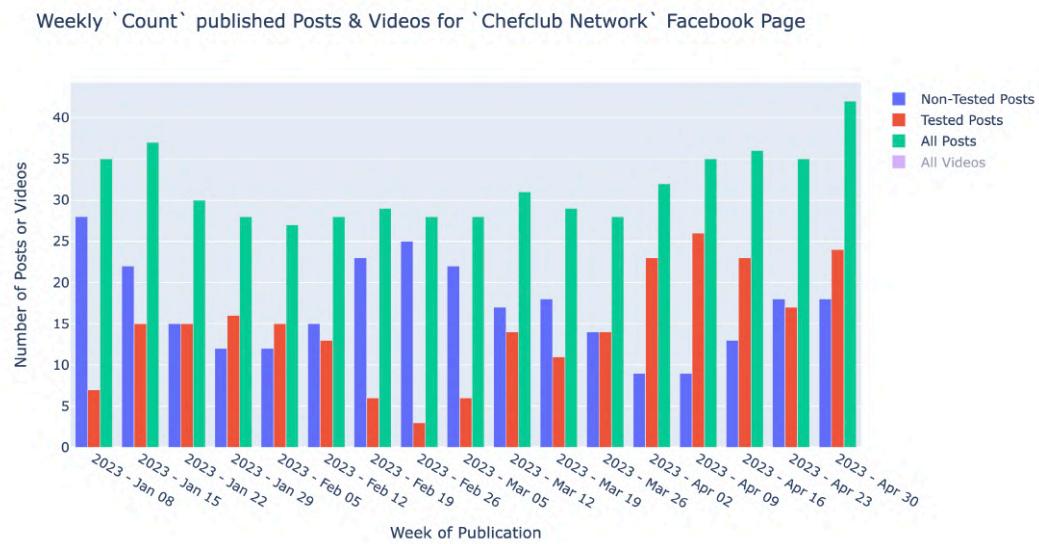


FIG. 4.42 : Histplot - Post Count by Week by Post Nature

Dans cette figure, on constate que Chefclub a adopté différentes stratégies de publication pour la page ChefClub Network. En effet, le nombre de publications par semaine est d'environ 30. De plus, le ratio de publications testées varie, ce qui est une bonne nouvelle pour moi. Je vais comparer le ratio des publications testées en fonction de la somme, de la moyenne et de la médiane.

Ensuite, je représente graphiquement la somme, la moyenne et la médiane par semaine pour les différents types de publications :

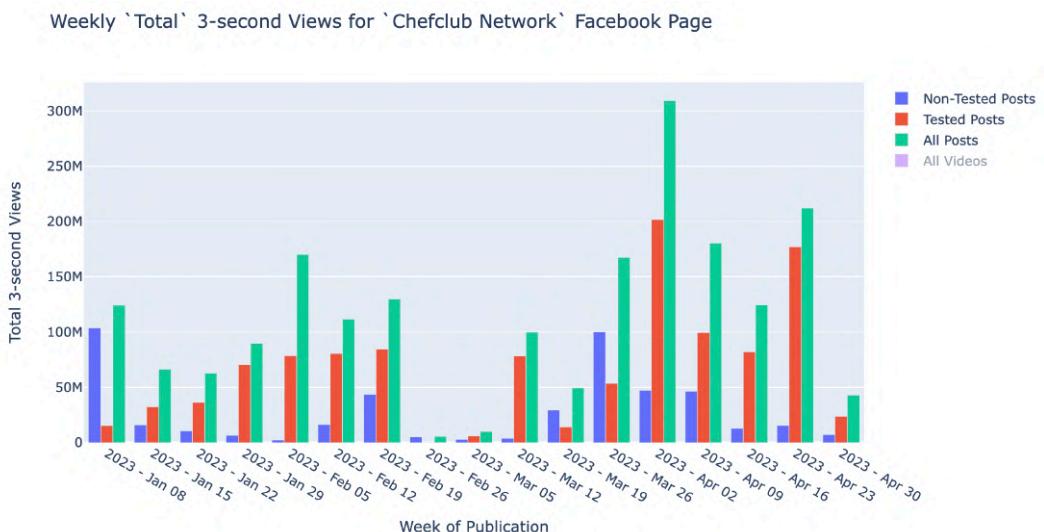


FIG. 4.43 : Histplot - Total Views per Week by Post Nature

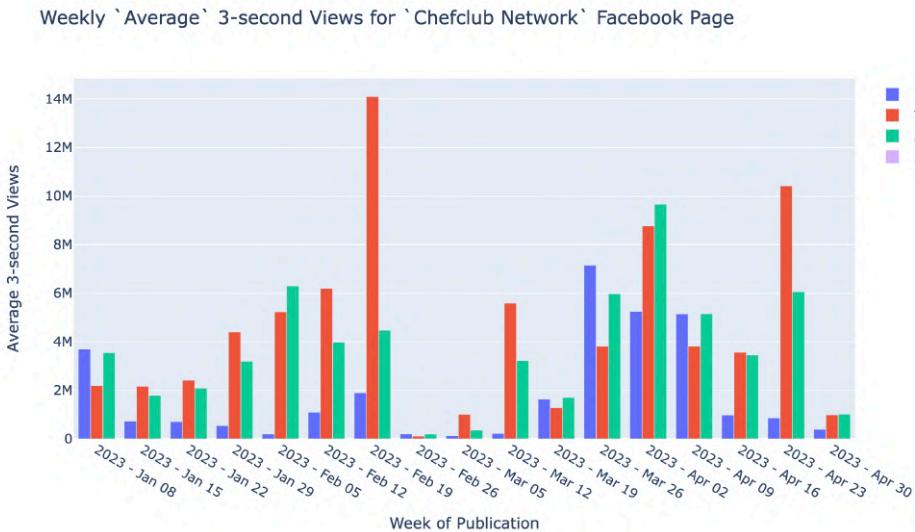


FIG. 4.44 : Histplot - Average Views per Week by Post Nature

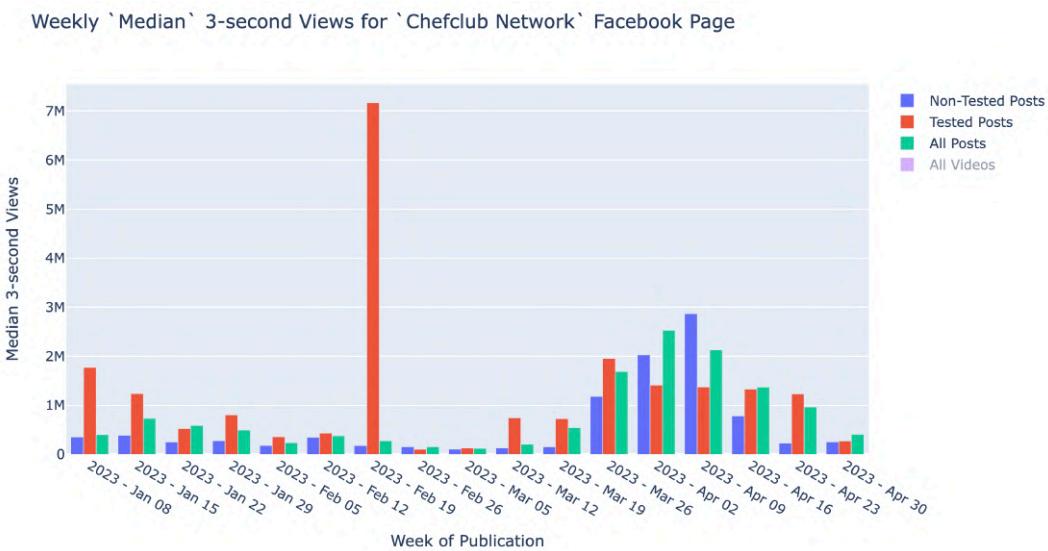


FIG. 4.45 : Histplot - Median Views per Week by Post Nature

Ces graphiques illustrent respectivement la somme, la moyenne et la médiane des vues par semaine pour les différents types de publications.

D'après ces figures, on peut confirmer que les publications testées génèrent généralement plus de vues en termes de somme, d'average et de médiane par rapport aux publications non testées, à l'exception de quelques semaines particulières.

Note : J'ai également réalisé ces graphiques en fonction des revenus pour les deux pages Facebook, **Chefclub** et **Chefclub Network**. Il est à noter que les graphiques des *revenus* et des *vues* reflètent les mêmes informations, car les variables discrètes *vues* et *revenus* sont fortement corrélées.

Maintenant, il serait important de regrouper toutes ces informations dans un seul graphique résumant ces données. Le graphique suivant présente cette synthèse :

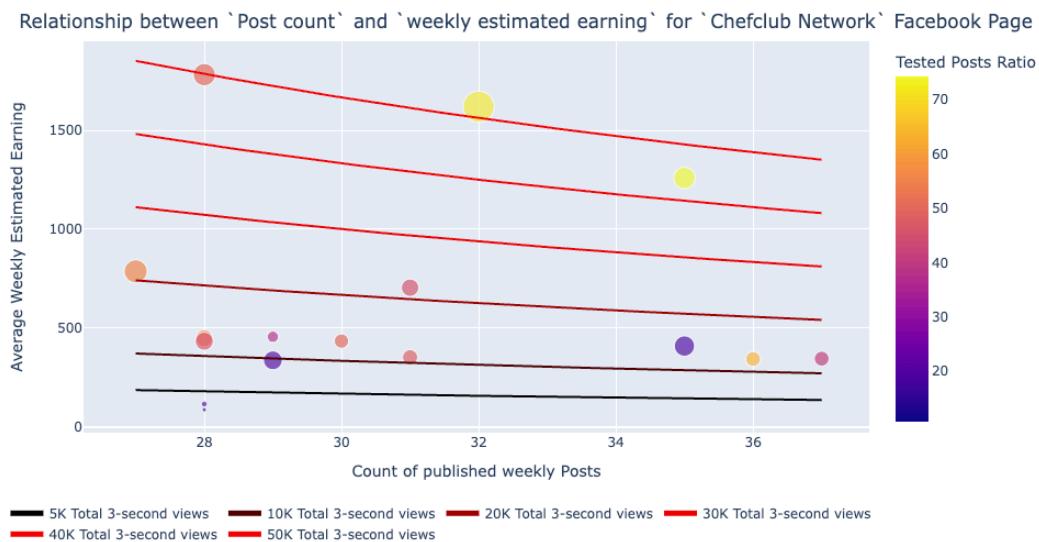


FIG. 4.46 : Scatter plot - Post Count and Weekly Estimated Earnings

La figure représentée dans la figure 4.47 fournit des informations sur la relation entre le nombre de publications et les gains hebdomadaires estimés. Les caractéristiques clés de la figure sont les suivantes :

- L'axe des x représente le **nombre de publications par semaine** (nombre de publications des Tested-posts et des Untested-Posts).
- L'axe des y représente la **moyenne des gains hebdomadaires estimés**.
- La période considérée s'étend du **premier trimestre de 2023** aux **deux premières semaines d'avril 2023**.
- Chaque point sur le graphique correspond à une valeur spécifique du nombre de publications et de la moyenne des gains hebdomadaires estimés.
- Les **lignes pleines** du graphique représentent les **courbes d'iso-gains** (ISO-EARNING), où chaque point sur la même courbe génère le même niveau de gains, par exemple, 5 000 € ou 50 000 € par semaine.
- La taille de chaque point représente la **somme des vues de 3 secondes générées au cours de cette semaine**.

Des détails supplémentaires à noter à propos de la figure sont :

- Les couleurs plus claires des points indiquent un pourcentage plus élevé de tests effectués.
- Les couleurs plus claires des courbes ISO-EARNING correspondent à des gains générés plus élevés.

Dans l'ensemble, cette représentation graphique offre une analyse claire et complète de la relation entre le nombre de publications, les gains hebdomadaires estimés et l'impact des vues.

4.3.3 Recommandations et Conclusion

Après avoir analysé les données et examiné les graphiques présentés, plusieurs recommandations peuvent être formulées pour améliorer les performances de la page **Chefclub Network**. Voici quelques suggestions basées sur les observations faites :

1. Augmenter le ratio des publications testées : Les publications testées ont montré une tendance positive en termes de vues et de revenus. Il est recommandé d'augmenter le ratio des publications testées par rapport aux publications non testées afin d'exploiter davantage cette opportunité de génération de vues et de revenus. Les points de couleur claire dans le graphique 4.47 indiquent les performances supérieures des publications testées.

2. Optimiser le moment des publications : Étant donné que le nombre de publications par semaine varie, il est important d'analyser les résultats en fonction du moment de publication. Identifier les moments où les publications testées ont générée des vues et des revenus plus élevés peut aider à planifier des publications stratégiques pour maximiser l'engagement des utilisateurs. Cette analyse peut être réalisée en utilisant un modèle prédictif des performances, comme décrit dans la section 4.3.4.

3. Tester de nouvelles approches entre les publications testées et non testées : Outre l'augmentation du ratio des publications testées, il serait intéressant d'explorer de nouvelles approches. Par exemple, tester un ratio de plus de 90% de publications testées avec un nombre de publications compris entre 32 et 35 par semaine. Le graphique suivant illustre cette zone prometteuse :

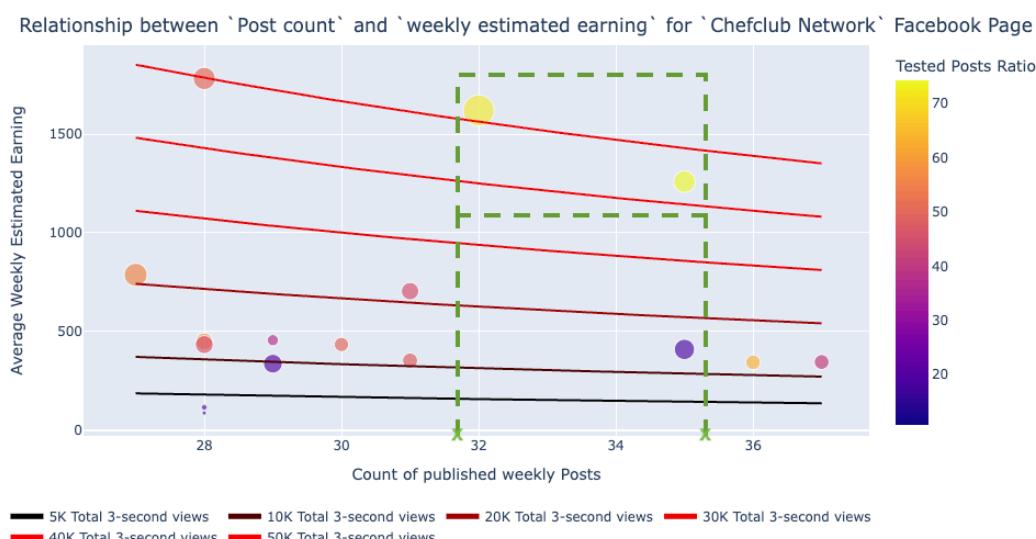


FIG. 4.47 : Optimal Post Count Strategy for Maximizing Weekly Estimated Earnings

Voici le code que j'ai utilisé pour générer ce graphique 4.47 avec Plotly :

```

1 import numpy as np
2 import plotly.express as px
3 import plotly.graph_objects as go
4
5 # Generate x values
6 x = np.linspace(posts_count.min(), posts_count.max(), 100)
7 # Generate y values for each iso line
8 K_values = [5000, 10000, 20000, 30000, 40000, 50000]

```

```

9 iso_lines = [K / x for K in K_values]
10 # Create traces for iso lines
11 iso_line_traces = [
12     go.Scatter(x=x, y=line, mode='lines', line=dict(color=f'rgb({i * 80},0,0)'), name
13     =f'{K // 1000}K Total 3-second views')
14     for i, line in enumerate(iso_lines)
15 ]
16 # Create scatter plot
17 fig = px.scatter(
18     x=posts_count,
19     y=average_weekly_posts,
20     color=tested_posts_ratio,
21     size=post_3_sec_views,
22     symbol=date_list,
23     color_discrete_sequence=["red", "green", "blue"],
24     symbol_map={"A": "square", "B": "circle"},
25     labels={'x': "Post Count", 'y': "Average Weekly Estimated Earning", "color": "Tested Posts Ratio", "size": "Total 3-sec Views"}
26 )
27 # Add iso line traces to the scatter plot
28 fig.add_traces(iso_line_traces)
29 # Customize figure layout
30 fig.update_layout(
31     title={
32         'text': "Relationship between 'Post count' and 'weekly estimated earning' for
33         'Chefclub Network' Facebook Page",
34         'y': 0.95,
35         'x': 0.5,
36         'xanchor': 'center',
37         'yanchor': 'top'
38     },
39     legend=dict(x=-0.1, y=-0.8, yanchor="bottom", orientation="h"),
40     margin=dict(t=50, b=50, l=50, r=50)
41 )
42 fig.show()
43

```

Listing 4.12: Post Count vs. Weekly Estimated Earnings with Iso-Gains Lines - Chefclub Network

4. Le monitoring : Il est important de rester vigilant et de surveiller régulièrement les performances de la page Chefclub Network. Les algorithmes de Facebook évoluent fréquemment, et il est nécessaire de s'adapter aux changements. De plus, le type de contenu publié peut également avoir un impact significatif sur les performances. Un monitoring constant permettra d'ajuster les stratégies en fonction des changements et des nouvelles tendances.

En conclusion, l'analyse approfondie des données et des graphiques a permis de mieux comprendre la relation entre les publications, les vues et les revenus sur la page Chefclub Network. En suivant les recommandations mentionnées, notamment en augmentant le ratio des publications testées, en optimisant le moment des publications et en explorant de nouvelles approches, il est possible d'optimiser les performances de la page Chefclub Network et de continuer à attirer et fidéliser une audience engagée. Le monitoring constant est également crucial pour rester à jour avec les évolutions de l'algorithme de Facebook et pour ajuster les stratégies en conséquence.

4.3.4 Développement et Test du Code Python pour l’Entraînement du Modèle

Dans cette partie, je me concentre sur le développement de la solution pour l’entraînement et le déploiement du modèle, en suivant la conception et les outils définis dans le chapitre précédent. La Figure 4.48 rappelle la conception de la solution que j’ai mise en place lors de l’analyse et de la conception.

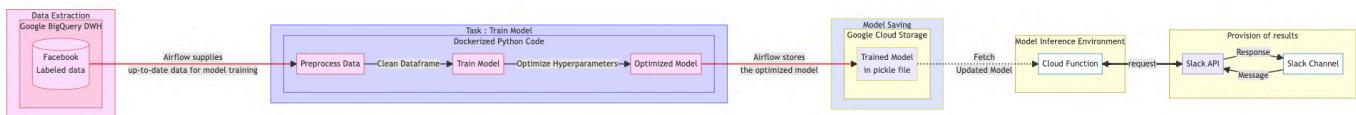


FIG. 4.48 : Solution d’Entraînement/Déploiement du Modèle (Rappel)

La Figure 4.49 rappelle le diagramme d’interaction qui complète la conception de la Figure 4.48. Il représente le mécanisme par lequel un utilisateur demande l’inférence du modèle via Slack.

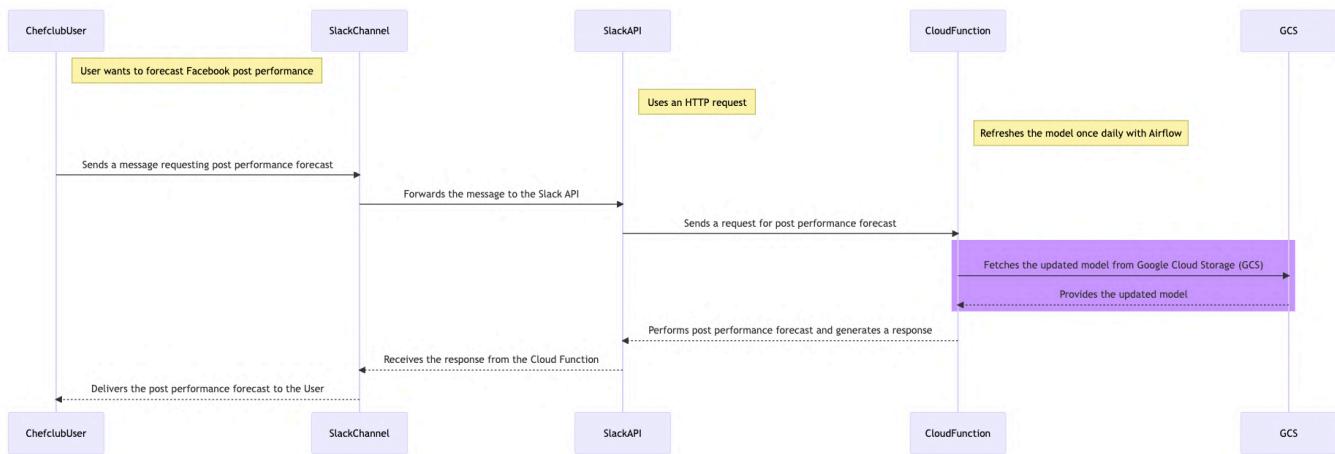


FIG. 4.49 : Sequence Diagram : Model Inference via Slack

4.3.4.1 Présentation du fonctionnement final du code Python développé

Cette section vise à présenter au lecteur le résultat final du code Python développé, ainsi que ses entrées et sorties. Ce code représente le cœur de la solution pour le Model Training. Les sections suivantes détailleront les composantes, les fonctionnalités et l’évolution du code afin de le rendre industrialisable et prêt pour la production.

Le code développé est responsable de l’entraînement d’un modèle sur une période donnée en utilisant les données de performances des posts sur Facebook. Le code télécharge les données à partir de BigQuery, entraîne un modèle d’apprentissage automatique et télécharge le modèle enregistré au format pickle vers Google Cloud Storage (GCS).

Les paramètres d’entrée du code sont spécifiés à travers le script `main.py`, qui requiert les arguments suivants :

- `-p/--page-name` : le nom de la page *Facebook* pour laquelle récupérer les données.

- **-s/--start-date** : la date de début de la période de d'entraînement des données au format *AAAA-MM-JJ*.
- **-e/--end-date** : la date de fin de la période d'entraînement des données au format *AAAA-MM-JJ*.

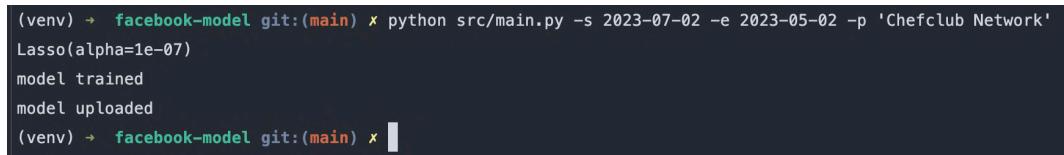
Voici la syntaxe pour exécuter le script :

```
1 $ python main.py -p <page_name> -s <start_date> -e <end_date>
```

Listing 4.13: CLI Python command to run the Model Training code

Le résultat du code est l'entraînement du modèle sur la période de **<start-date>** à **<end-date>** pour la page **<page-id>**.

la figure 4.50 suivante montre un exemple d'utilisation de la commande pour entraîner un modèle sur les données de la page *Chefclub Network* :

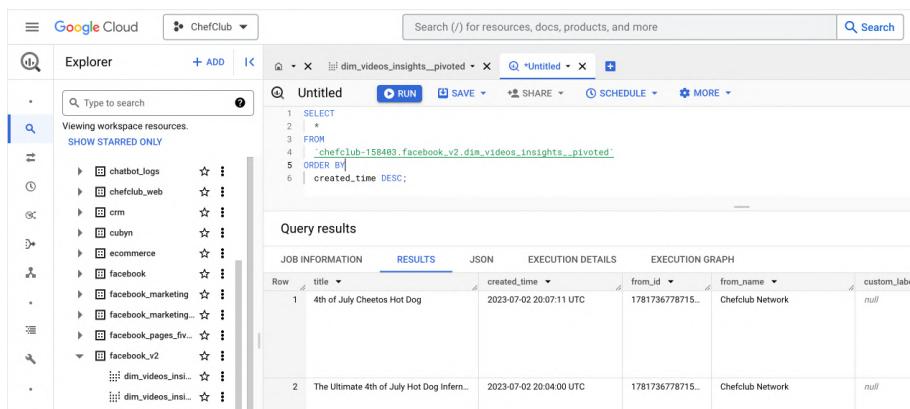


```
(venv) → facebook-model git:(main) ✘ python src/main.py -s 2023-07-02 -e 2023-05-02 -p 'Chefclub Network'
Lasso(alpha=1e-07)
model trained
model uploaded
(venv) → facebook-model git:(main) ✘
```

FIG. 4.50 : Example of Model Training with CLI

4.3.4.2 Récupération des données de performance des posts sur Facebook

Pour récupérer les données de performance des posts sur Facebook, j'utilise BigQuery Data Warehouse. Tout comme pour le Job YouTube, j'ai mis en place un Job spécifique à Facebook qui permet de récupérer quotidiennement les données de performance des posts sur Facebook. La table utilisée ressemble à celle illustrée dans la figure 4.51 suivante :



Row	title	created_time	from_id	from_name	custom_label
1	4th of July Cheetos Hot Dog	2023-07-02 20:07:11 UTC	1781736778715...	Chefclub Network	null
2	The Ultimate 4th of July Hot Dog Infer...	2023-07-02 20:04:00 UTC	1781736778715...	Chefclub Network	null

FIG. 4.51 : Example of a performance table for Facebook posts in BigQuery

Cette table est mise à jour quotidiennement, ce qui me permet de maintenir la fraîcheur des données. L'objectif principal est de récupérer les nouvelles données quotidiennes et de les utiliser pour entraîner un nouveau modèle ayant pour objectif de prévoir les performances futures des posts sur Facebook.

Voici le code 4.52 que j'utilise pour extraire les données de cette table et les stocker dans un DataFrame pandas :

```
# Define the initial date to be used in the query
DATASET_NAME = 'facebook_v2'
TABLE_NAME = 'dim_videos_insights_pivoted'

def run(start_date, end_date, page_name):
    # Initialize a BigQuery client using the service account JSON file
    client = bigquery.Client.from_service_account_json(DATA_CREDENTIALS_PATH)

    # Load the BigQuery data with the specified parameters
    sql_dim_videos_insights_pivoted = load_bigquery_data(dataset_name=DATASET_NAME,
                                                          table_name=TABLE_NAME,
                                                          start_date=start_date,
                                                          end_date=end_date,
                                                          page_name=page_name)

    # Query the BigQuery using the provided SQL and retrieve the results as a DataFrame
    df = client.query(sql_dim_videos_insights_pivoted).to_dataframe()
```

FIG. 4.52 : Loading the BigQuery table into a *DataFrame*

Grâce à ce code 4.52, je suis en mesure d'extraire les données nécessaires de la table BigQuery et de les manipuler efficacement en utilisant un DataFrame pour entraîner le modèle. Cette approche me permet de tirer parti des données historiques pour identifier les schémas et les tendances, et de les utiliser ensuite pour faire des prévisions précises sur les performances futures des posts sur Facebook. En utilisant les nouvelles données quotidiennes pour mettre à jour le modèle, je garantis sa pertinence et sa capacité à s'adapter aux changements récents dans les comportements des utilisateurs et les algorithmes de Facebook.

4.3.4.3 Développement du modèle à partir des données Facebook

Maintenant que j'ai la table de données récupérer a partir de `bigrquery`, et puisqu'il faut automatiser l'entraînement du model, il faut créer un code capable d'entraîner un model à chaque fois sur la table. pour se faire on suit cet enchainement :

4.3.4.3.1 Préparation des données :

Cette première étape consiste à préparer les données récupérées à partir de Facebook. Tout d'abord, je réalise le nettoyage des données en renommant les colonnes du dataframme et en effectuant les castings nécessaires pour les types de données. De plus, je réalise la labellisation des données. Vous pouvez consulter le code utilisé dans l'annexe (voir Figure A.1).

Ensuite, je procède à la détection des valeurs aberrantes (outliers) en utilisant le code suivant :

```
1  def flag_outliers(df, column_name, frequency, threshold):
2      # Calculate the z-scores using a rolling window
3      print(f"Number of total posts: {df.shape[0]}")
4      window_size = frequency
5      df['z_score'] = df[column_name].rolling(window_size).apply(
6          lambda x: (x[-1] - np.mean(x)) / np.std(x), raw=True)
7      # Flag the records based on the z-score values
8      df['is_outlier'] = np.logical_or(df['z_score'] > threshold, df['z_score'] < -threshold)
9      num_outliers = df['is_outlier'].sum()
```

```

10     # Print the number of detected outliers
11     print(f"Number of (deleted) outliers: {num_outliers}")
12     # Drop the outliers
13     df = df[~df['is_outlier']]
14     # Remove the intermediate 'z_score' column
15     df.drop('z_score', axis=1, inplace=True)
16
17     return df

```

Listing 4.14: Outlier Detection and Removal Python Code

Ce code est basé sur la **détection des outliers à l'aide du Z-score**, comme expliqué dans le chapitre 2 de l'état de l'art (Voir le paragraphe 2.2.2.3.2). La figure 4.53 suivante illustre le processus mis en place pour la détection des outliers.

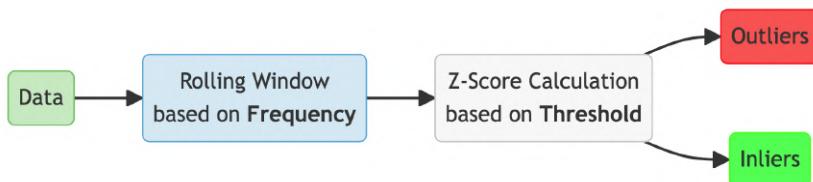


FIG. 4.53 : Z-Score Outlier Detection Chart

Note : Notez que dans ce contexte, la **Frequency** représente la taille de la fenêtre, c'est-à-dire le nombre de jours considéré.

Voici comment j'utilise ce code pour détecter et supprimer les *relative outliers* (*valeurs aberrantes relatives*) à une semaine (7 jours) :

```

# Flaging the Outliers
chefclub_us_post_perf_lt = flag_outliers(chefclub_us_post_perf_lt, "total_3-second_video_views", 7, 1.5)

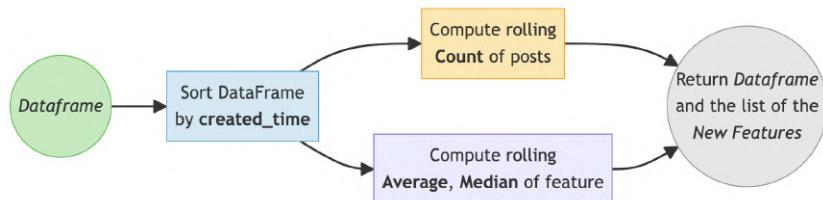
```

FIG. 4.54 : Flaging and Deleting the Outliers with Z-Score

En effet, j'élimine les posts ayant un score supérieur à 1.5 pour chaque semaine. Cette approche permet de supprimer les éléments aberrants (outliers) qui pourraient confondre le modèle et dégrader ses performances.

4.3.4.3.2 Feature Engineering :

L'objectif de cette étape est de créer de nouvelles caractéristiques auxquelles nous avons accès avant la publication du post sur Facebook. Il s'agit de l'historique des performances des posts sur la page en question. Voici le code que j'utilise pour ajouter les données historiques : <code>. Ce code permet de calculer la *médiane*, la *moyenne* et le *nombre de posts* effectués sur une fréquence (en jours). La figure 4.55 suivante explique le processus :


 FIG. 4.55 : The process of adding **historical features** to the Pandas DataFrame

Voici dans cette figure 4.56 comment j'utilise ce code pour ajouter mes nouvelles caractéristiques : je liste les caractéristiques pour lesquelles je souhaite obtenir l'historique, puis je calcule leur historique sur les 7 derniers jours.

```

# Add Historical Features
historical_features_list = ['3-second_video_views',
                            '60-second_video_views_by_returning_viewers',
                            'estimated_earnings_by_video_usd',
                            'overall_negative_feedback_per_total_3-second_video_views',
                            'seconds_viewed_by_returning_viewers_per_seconds_viewed_from_recommendations',
                            'overall_negative_feedback_per_total_3-second_video_views'
                           ]

historical_features_names, feature = [], []
for feature in historical_features_list:
    chefclub_us_post_perf_lt, feature = add_historical_features(chefclub_us_post_perf_lt, 7, feature)
    historical_features_names += feature

historical_features_names = list(dict.fromkeys(historical_features_names))
    
```

 FIG. 4.56 : Adding **historical features** to the Pandas DataFrame

Note : Il est important que le code utilise les données des 7 derniers jours sans inclure le jour actuel. Le calcul des caractéristiques historiques se fait sur une fenêtre de rolling de 7 jours, excluant ainsi le jour courant. Cela est crucial pour éviter toute contamination de notre modèle.

En plus des caractéristiques historiques, j'ajoute d'autres caractéristiques, telles que l'heure de la publication et le numéro du jour de la publication. Le code 4.57 suivant illustre la fonction que j'utilise :

```

63 # Function to add Temporal Features
64 def add_temporal_features(df):
65     df.index = pd.to_datetime(df.index)
66     df['hour'] = df.index.hour
67     df['day_of_week'] = df.index.dayofweek
68     return df
    
```

 FIG. 4.57 : Adding **temporal features** to the Pandas DataFrame

4.3.4.3.3 Transformation des Features et la Target :

La figure suivante présente la distribution de la variable Target :

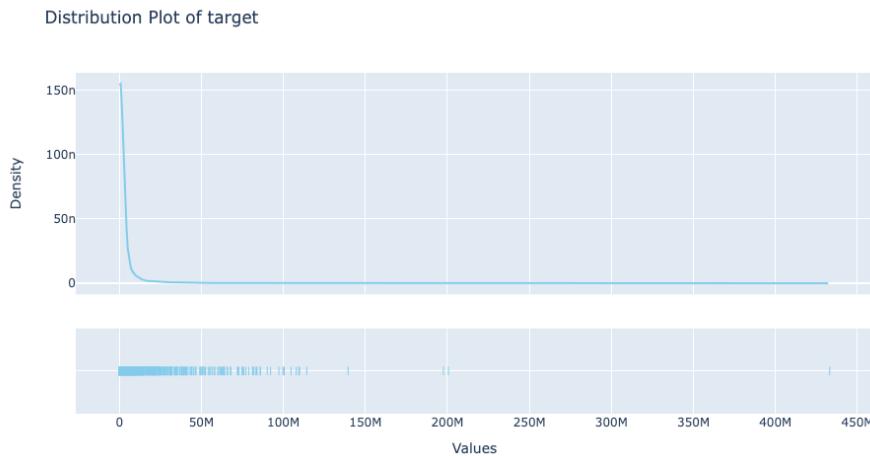


FIG. 4.58 : Analysis of the Target Variable Distribution

En réalité, cette Target cible présente une distribution asymétrique à droite (Right Skewed), ce qui pose problème pour les modèles de régression classiques. L'idéal serait de corriger cette distribution en utilisant une fonction bijective afin de se rapprocher d'une distribution plus classique qui respecte les hypothèses d'un modèle de régression. Dans mon cas, j'ai utilisé la fonction $x \rightarrow \log(x)$ pour corriger la variable cible, ce qui a donné la nouvelle distribution (qui se rapproche d'une distribution normale) :

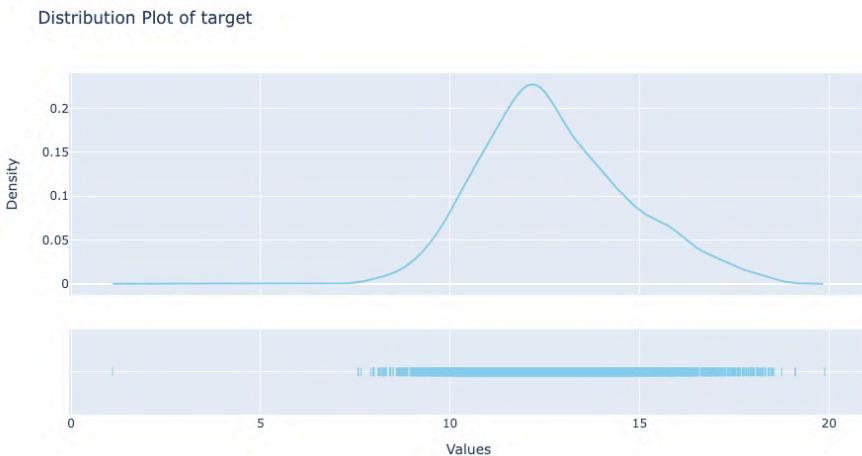


FIG. 4.59 : Correction of the Target Variable Distribution

Comme il est visible dans la figure 4.59, la distribution a été corrigée. J'ai appliqué la même transformation aux autres *Features*. L'objectif de cette correction est de rendre la distribution aussi proche que possible des distributions classiques.

4.3.4.3.4 Sélection des Features :

Lors de cette étape, je procède à la sélection des Features qui seront utilisées pour entraîner le modèle. Pour cela, j'utilise une matrice de corrélation qui permet d'évaluer les relations linéaires entre la Target et les différentes Features. La figure suivante 4.60 présente un exemple de matrice de corrélation utilisée dans ce processus :

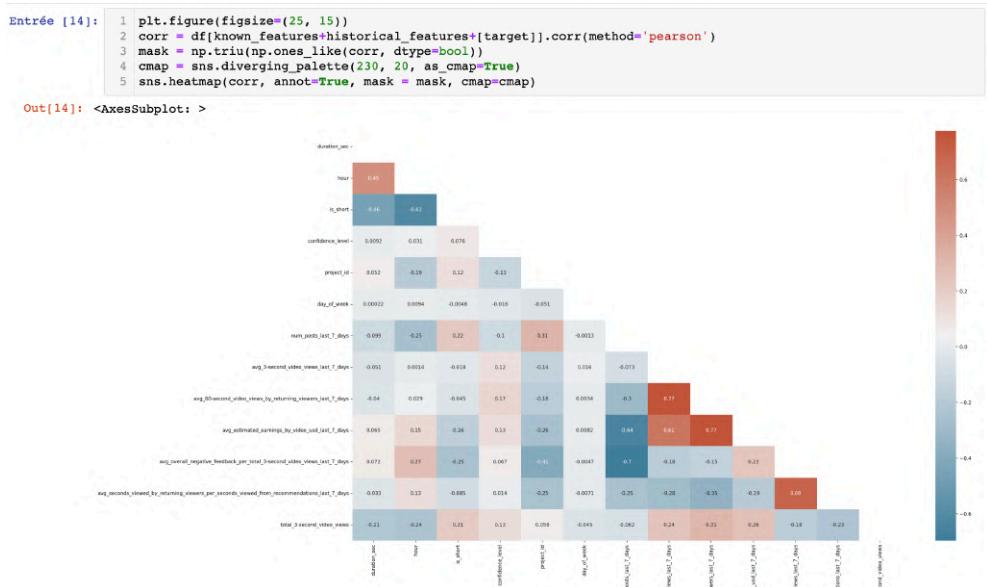


FIG. 4.60 : Feature Selection with the Correlation Matrix

Cette matrice permet d'afficher et de classer la corrélation linéaire entre la Target et les Features. J'ai choisi les caractéristiques qui présentent une corrélation élevée avec la variable cible et qui sont peu indépendantes entre elles. De plus, j'également ajouté d'autres caractéristiques qui n'ont pas de corrélation linéaire évidente avec la variable cible, mais qui sont pertinentes en fonction de l'expérience business dans le sujet traité. Cette approche combine à la fois des **aspects statistiques** et des **connaissances métier** pour une sélection optimale des Features.

4.3.4.3.5 Entraînement et Validation des performances du modèle :

Le but de cette section est de trouver le meilleur modèle qui correspond à nos données. J'ai entraîné plusieurs modèles et évalué leurs performances en utilisant les fonctions de coût RMSE, MAE et MSE. Notre objectif principal est de trouver le modèle qui minimise la RMSE. Voici la liste des modèles que j'ai utilisés :

```

Entrée [21]: 1 # Define models
2 models = [
3     RANSACRegressor(),
4     LinearRegression(),
5     DecisionTreeRegressor(),
6     BaggingRegressor(),
7     RandomForestRegressor(n_estimators=100,
8                           criterion='absolute_error',
9                           max_depth=None,
10                          min_samples_split=2,
11                          min_samples_leaf=1,
12                          min_weight_fraction_leaf=0.0,
13                          max_features='auto',
14                          max_leaf_nodes=None,
15                          random_state=None,
16                          n_jobs=None,
17                          verbose=0,
18                          warm_start=False),
19     SVR(),
20     KNeighborsRegressor(n_neighbors=5),
21     Ridge(alpha=0.0001),
22     Lasso(alpha=0.0001),
23     GradientBoostingRegressor(),
24     ExtraTreesRegressor()
25 ]

```

FIG. 4.61 : Models Used for Model Training

En divisant les données à l'aide de la méthode **Train-Val-Test Split** mentionnée dans l'état de l'art, j'ai obtenu les résultats suivants :

```

1 # Evaluate models and store results in a DataFrame
2 results = []
3 for model in models:
4     model_name = type(model).__name__
5     model.fit(x_train, y_train)
6     scores = evaluate_model(model, x_test, y_test, evaluation_metrics)
7     results.append({'Model Name': model_name, **scores})
8 results_df = pd.DataFrame(results)
9
10 # Rename the <lambda> column to RMSE
11 results_df = results_df.rename(columns={"<lambda>": "RMSE"})
12 results_df

```

	Model Name	RMSE	mean_squared_error	mean_absolute_error	r2_score
1	LinearRegression	0.827670	0.685038	0.675715	0.245505
7	Ridge	0.827670	0.685038	0.675715	0.245505
8	Lasso	0.827870	0.685370	0.675837	0.245139
9	GradientBoostingRegressor	0.853683	0.728775	0.709341	0.197333
4	RandomForestRegressor	0.903014	0.815434	0.741924	0.101887
5	SVR	0.910499	0.829008	0.756056	0.086937
10	ExtraTreesRegressor	0.945841	0.894615	0.763890	0.014678
6	KNeighborsRegressor	0.987628	0.975409	0.787788	-0.074308
3	BaggingRegressor	0.988754	0.977635	0.816588	-0.076759
0	RANSACRegressor	1.139567	1.298612	0.963962	-0.430281
2	DecisionTreeRegressor	1.374295	1.888688	1.160023	-1.080186

FIG. 4.62 : Lasso Model Performance on test data

De plus, j'ai créé un graphique interactif avec Plotly pour visualiser les performances du modèle. Ce graphique présente les prédictions du modèle Y_{pred} par rapport aux valeurs réelles Y_{test} . Les meilleures prédictions sont représentées par des points proches de la droite verte $Y_{\text{test}} = Y_{\text{pred}}$. Voici le graphique :

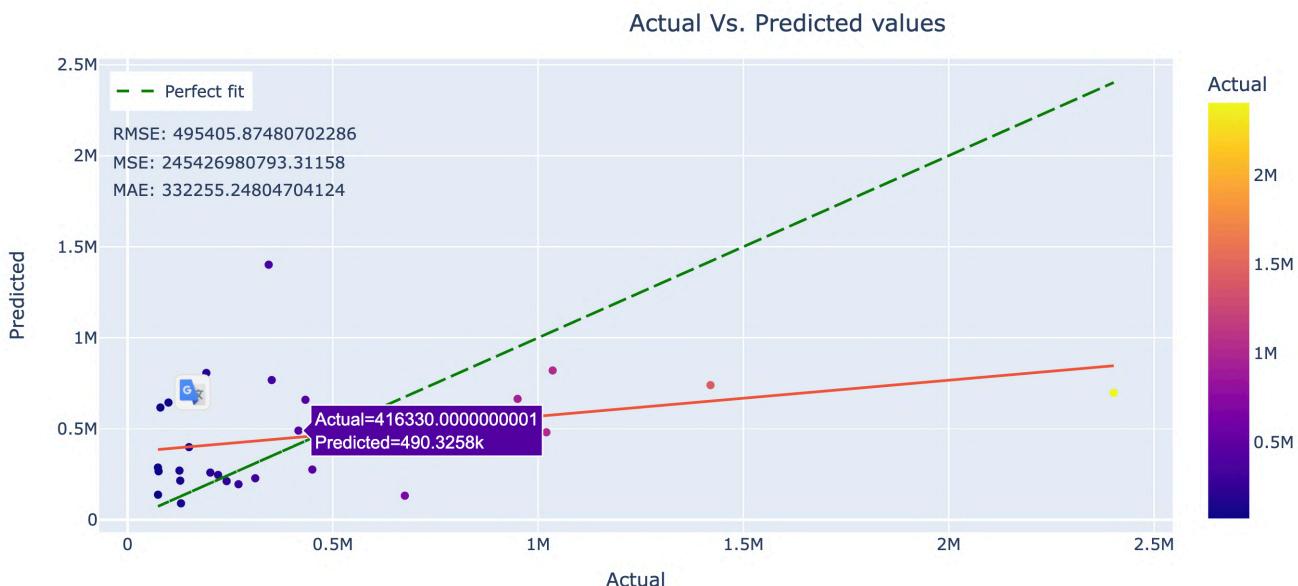


FIG. 4.63 : Lasso Model : Y_{test} (actual) Vs. Y_{pred} (predicted)

Note importante : Les figures 4.62 et 4.63 présentent les performances du modèle **Lasso**. Cependant, ces performances relativement moyennes sont attribuables au manque de variables explicatives. Il est en effet difficile de trouver des caractéristiques qui peuvent nous aider à prédire les performances d'un post sur Facebook, étant donné l'évolution rapide de l'algorithme de Facebook et le manque d'informations détaillées sur les publications. Il est donc essentiel de noter que ce modèle doit être amélioré davantage.

Parmi les modèles testés, trois se sont démarqués : **LinearRegression**, **Ridge** et **Lasso**. Après

avoir utilisé la technique de validation croisée **Rolling Cross Validation**, le modèle **Lasso** s'est avéré être le meilleur. Afin de continuer à améliorer ce modèle, nous allons utiliser la méthode **GridSearchCV**.

Voici le code utilisé pour effectué la **Rolling Cross Validation**

```

1  from sklearn.model_selection import TimeSeriesSplit
2  from sklearn.metrics import mean_squared_error
3
4  def rolling_cross_validation(model, X, y):
5      # Initialize the rolling cross-validation
6      rolling_cv = TimeSeriesSplit(n_splits=5)
7      rmse_scores = [] # Perform rolling cross-validation
8      for train_index, test_index in rolling_cv.split(X):
9          X_train, X_test = X[train_index], X[test_index]
10         y_train, y_test = y[train_index], y[test_index]
11         # Fit the model on the training data
12         model.fit(X_train, y_train)
13         # Predict on the testing data
14         y_pred = model.predict(X_test)
15         # Calculate the RMSE
16         rmse = np.sqrt(mean_squared_error(y_test, y_pred))
17         rmse_scores.append(rmse)
18         # Print the RMSE
19         # Calculate the average RMSE across all iterations
20         avg_rmse = np.mean(rmse_scores)
21     return avg_rmse
22

```

Listing 4.15: Rolling Cross-Validation with RMSE

4.3.4.3.6 Ajustement des hyperparamètres :

Maintenant que j'identifie le modèle **Lasso** comme étant le meilleur performer sur mes données, je procède à l'ajustement des hyperparamètres de ce modèle. Pour ce faire, j'utilise la méthode **GridSearchCV**, qui me permet de définir un espace de recherche pour les hyperparamètres. Dans mon cas, l'hyperparamètre que j'ajuste est *Alpha*, qui contrôle la régularisation du modèle **Lasso**.

Je définis un intervalle pour la valeur d'*Alpha* et lance l'entraînement du modèle avec différentes valeurs d'*Alpha* à l'aide de **GridSearchCV**. À la fin du processus, **GridSearchCV** me retourne la meilleure valeur d'*Alpha* qui optimise les performances du modèle. Le processus d'ajustement des hyperparamètres est illustré dans la figure suivante :

```

Entrée [15]: 1 X, y = x_train, y_train
2 n_samples, n_features = 10, 5
3 # Define a wider range of alpha values
4 alphas = np.linspace(0.0000001, 0.0000000001, 500)
5 parameters = {'alpha': alphas}
6 # Define the model/estimator
7 model = Lasso()
8 # Define the grid search
9 Lasso_reg = GridSearchCV(model,
10                         parameters,
11                         scoring='neg_root_mean_squared_error',
12                         cv=5)
13 # Fit the grid search
14 Lasso_reg.fit(X, y)
15 # Best estimator
16 print(Lasso_reg.best_estimator_)
17 # Best model
18 best_model = Lasso_reg.best_estimator_
19 best_model.fit(X, y)

Lasso(alpha=1e-07)

```

FIG. 4.64 : Optimizing Lasso Hyperparameters with GridSearchCV

En utilisant GridSearchCV, j'explore différentes combinaisons d'hyperparamètres dans l'espace de recherche défini pour trouver la meilleure configuration. Cela me permet d'améliorer davantage les performances du modèle `Lasso` en trouvant les valeurs d'hyperparamètres les plus appropriées pour mon dataset.

4.3.4.3.7 Exportation du modèle :

Une fois que le modèle a été entraîné et que les hyperparamètres ont été ajustés, je procède à l'exportation du modèle. Cette étape consiste à sauvegarder les poids, les paramètres et les métainformations du modèle dans un format compatible pour une utilisation ultérieure. J'utilise dans ce cas le module *Pickle* pour effectuer cette sauvegarde. Voici le code que j'ai utilisé pour exporter le modèle en utilisant *Pickle* :

```

1 # Get the best model from Lasso_GridSearchCV
2 best_model = Lasso_GridSearchCV.best_estimator_
3 # Fit the best model on the training data
4 best_model.fit(X_train, y_train)
5 # Save the trained model to a file
6 pickle.dump(best_model, open('model/winner_model.pkl', 'wb'))
7

```

Listing 4.16: Exporting the trained Lasso Model with Pickle

L'exportation du modèle nous permet de le déployer dans d'autres environnements ou de l'utiliser pour effectuer des prédictions sur de nouvelles données. Le fichier *Pickle* contient toutes les informations nécessaires pour l'utilisation ultérieure du modèle, et c'est ce fichier que je vais charger dans le *Model Registry* afin de pouvoir effectuer des inférences avec le modèle.

4.3.4.4 Configuration de Cloud Storage comme Model Registry

Bien que le Vertex AI Model Registry de Google soit le service le plus adapté pour le stockage et la gestion des modèles, dans notre cas, les modèles sont relativement simples et de taille très réduite. Leur gestion est donc plus simple. Nous utilisons donc Google Cloud Storage comme service

de gestion des modèles créés (un modèle par page). Le code suivant illustre comment j'effectue cette opération :

```

1   from google.cloud import storage
2   from google.oauth2 import service_account
3
4   def upload_or_replace_blob(bucket_name,
5                               source_file_name,
6                               destination_blob_name,
7                               service_account_key_path):
8       # Load the service account credentials
9       credentials = service_account.Credentials.from_service_account_file(
10          service_account_key_path)
11       # Create a storage client using the credentials
12       storage_client = storage.Client(credentials=credentials)
13       # Get the specified bucket
14       bucket = storage_client.bucket(bucket_name)
15       # Get or create the blob with the specified name in the bucket
16       blob = bucket.blob(destination_blob_name)
17       # If the blob already exists, delete it before uploading the new file
18       if blob.exists():
19           blob.delete()
20       # Upload the file to the blob
21       blob.upload_from_filename(source_file_name)

```

Listing 4.17: Cloud Storage Blob Uploader with Replacement Function

Dans ce code, la fonction `upload_or_replace_blob()` remplace le modèle s'il existe déjà. Cela permet à chaque exécution du code de mettre à jour le modèle. La figure 4.68 montre le bucket où les modèles sont stockés. Voici comment on appelle cette fonction avec nos propres arguments (Voir Fig. 4.65) :

```

# Upload the model to GCS
upload_or_replace_blob(bucket_name=BUCKET_NAME,
                       source_file_name=MODEL_PATH,
                       destination_blob_name=MODEL_PATH,
                       service_account_key_path=DATA_CREDENTIALS_PATH)
print("model uploaded")

```

FIG. 4.65 : Production Use of `upload_or_replace_blob()` Function

4.3.4.5 Test et Validation du Code Python avec VENV

Comme pour le Job Youtube, Avant de dockeriser mon code et de le mettre en production, il est crucial de réaliser des tests dans l'environnement VENV. Pour ce faire, j'ai organisé mon environnement VENV selon l'architecture illustrée dans la figure suivante :

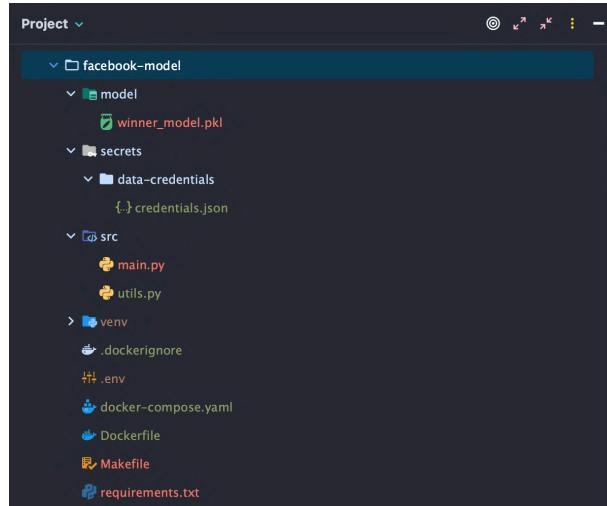


FIG. 4.66 : Folder Architecture of Model Training in VENV Environment

Une fois l'environnement VENV activé, j'ai procédé à un test qui est visible dans la figure suivante. En réponse, j'ai obtenu la confirmation que le modèle est bien stocké dans GCS.

```
(venv) → facebook-model git:(main) ✘ source venv/bin/activate
(venv) → facebook-model git:(main) ✘ make train-model
python src/main.py -s 2023-07-02 -e 2023-05-02 -p 'Chefclub Network'
Lasso(alpha=1e-07)
model trained
model uploaded
Successfully trained the model
(venv) → facebook-model git:(main) ✘
```

FIG. 4.67 : Model Training Evaluation in VENV Environment

Dans la figure 4.68 suivante, vous pouvez voir que le modèle a été correctement stocké dans GCS :

Name	Size	Type	Created	Storage class	Last modified
winner_model.pkl	3 KB	application/octet-stream	Jul 8, 2023, 11:13:38 AM	Standard	Jul 8, 2023, 11:13:38 AM

FIG. 4.68 : Updated Pickle Model in Google Cloud Storage

4.3.5 Développement et Test de l'environnement pour la Serverless Inference

Dans cette section, je vais aborder le développement et les tests de l'environnement nécessaire à l'inférence sans serveur (serverless). L'objectif est de créer une fonction Cloud sur Google Cloud Platform (GCP) qui permettra d'effectuer l'inférence à l'aide du modèle préalablement entraîné. Dans un deuxiéme temps, je vais entamé le process d'automatisation de la création du model.

4.3.5.1 Développement local de la Cloud Function

Pour commencer, j'ai développé localement la fonction Cloud appelée `predict()` qui sera responsable de l'inférence. J'utilise un framework de développement pour faciliter la création de cette fonction. Cette étape m'a permis de valider le bon fonctionnement de la fonction avant de la déployer sur GCP. Pour ce faire, j'ai organisé mon environnement VENV selon l'architecture illustrée dans la figure 4.69 ci-dessous :

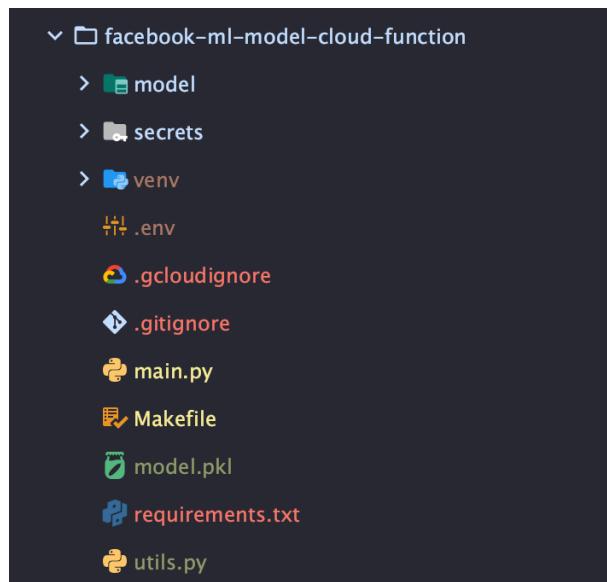


FIG. 4.69 : Folder Architecture of Model Deployment in VENV Environment

Voici la fonction responsable de la prédiction :

```

1  @functions_framework.http
2  def predict(request: flask.Request) -> flask.Response:
3      content_type = request.headers.get("content-type")
4
5      allowed_content_types = [
6          "application/json",
7          "text/plain",
8          "application/x-www-form-urlencoded"]
9
10     if content_type not in allowed_content_types:
11         raise ValueError(f"Unknown content type: {content_type}")
12
13     if content_type == "application/json":
14         json_data = request.get_json()
15     elif content_type == "text/plain":
16         json_data = json.loads(request.data)
17     else:
  
```

```

16         json_data = json.loads(request.form.get("text"))
17
18     df = process_data(json_data)
19
20     # Make predictions using the loaded model
21     y_pred = pkl_model.predict(df)
22     prediction = calculate_prediction(y_pred)
23
24     return flask.Response(prediction, mimetype="text/plain")

```

Listing 4.18: Code for the `predict()` function

Il s'agit d'une fonction classique qui utilise le décorateur `@functions_framework.http` pour simuler l'environnement Cloud Function en local. Je peux maintenant effectuer l'inférence local du modèle en utilisant la fonction `predict()`.

4.3.5.1.1 Inférence locale du modèle en utilisant la fonction `predict()` Pour effectuer des tests d'inférence locale, j'utilise deux terminaux : un pour lancer le serveur et un deuxième terminal (client) pour effectuer la demande. Dans le premier terminal, j'exécute la commande pour lancer le serveur (voir la figure 4.70 ci-dessous).

```

Terminal Local (3) × Local (2) × + ↗
→ facebook-ml-model-cloud-function git:(main) ✘ make run-server
Server Running
. venv/bin/activate && functions-framework --target=predict --source=main.py

```

FIG. 4.70 : Local execution of the `predict()` function using the `functions_framework`

Une fois le serveur lancé, dans le deuxième terminal, j'effectue une requête HTTP POST en envoyant les données de test (`X_test`) au format JSON. Je reçois ensuite la prédiction en réponse (voir la figure 4.71 ci-dessous).

```

(venv) → facebook-ml-model-cloud-function git:(main) ✘ make test-local-1
curl -w "\n" -H "Content-Type: application/json" -X POST -d '[{"publish_time": "2023-01-01T00:00:00Z", "day_of_week": "5", "avg_3-second_video_views_last_7_days": "13.78073798014", "video_usd_last_7_days": "2.426594545656942", "avg_overall_negative_feedback_percent": "0.0001", "viewed_from_recommendations_last_7_days": "-2.730848097906935"}]' localhost:8080
261863.03
(venv) → facebook-ml-model-cloud-function git:(main) ✘

```

FIG. 4.71 : Example of local model inference

Cette approche me permet de vérifier le bon fonctionnement de la fonction `predict()` localement avant de la déployer sur la plateforme GCP.

4.3.5.2 Déploiement et Inference de la Cloud Function sur GCP

Maintenant que la fonction `predict()` fonctionne localement, je procède au déploiement de cette fonction sur GCP. J'utilise la commande illustrée dans la figure 4.72 pour effectuer le déploiement.

```

Terminal Local (3) × Local (2) × + ↵
(venv) → facebook-ml-model-cloud-function git:(main) ✘ make deploy-GCP
gcloud functions deploy test-ml2 --gen2 --runtime=python311 --region=europe-west6 --source=. --entry-point=predict --trigger-http --allow-unauthenticated
Preparing function...done.
↑ Deploying function...
↑ [Build] Creating build... Logs are available at [https://console.cloud.google.com/cloud-build/builds;region=europe-west6/fa85f4e3-4152-4469-866b-e1a57ec57826?project=812058810488]
. [Service]
. [ArtifactRegistry]
. [Healthcheck]
. [Triggercheck]

```

FIG. 4.72 : Deployment of the Cloud Function using the Make command

Une fois le déploiement terminé, la fonction est prête à être utilisée pour l'inférence, comme indiqué dans la figure 4.73 ci-dessous.

Environment	Name	Last deployed	Region	Recommendation	Trigger	Runtime	Memory allocated	Executed function	Actions
2nd gen	test-ml2	Jul 8, 2023, 5:31:31 PM	europe-west6		HTTP	Python 3.11	256 MB	predict	⋮

FIG. 4.73 : Cloud Function deployed on GCP

Un exemple d'inférence avec une requête HTTP effectuée avec la fonction `predict()` déployée sur le cloud GCP est présenté dans la figure 4.74.

```

Terminal Local (3) × Local (2) × + ↵
(venv) → facebook-ml-model-cloud-function git:(main) ✘ make test-gcp
curl -w "\n" -H "Content-Type: application/json" -X POST -d '[{"publish_time": "2023-05-07 13:05:33", "day_of_week": "1", "avg_3-second_video_views_last_7_days": "13.780737980146222", "avg_60-second_video_usd_last_7_days": "2.426594545656942", "avg_overall_negative_feedback_per_total_3-second_viewed_from_recommendations_last_7_days": "-2.730848097906935"}]' https://europe-west6-ml2-test-ml2.1235017.46
(venv) → facebook-ml-model-cloud-function git:(main) ✘

```

FIG. 4.74 : Example of model inference using the deployed `predict()` function on GCP

4.3.5.3 Intégration de l'API Slack avec la Cloud Function

Maintenant que la fonction a été déployée, testée et renvoie les résultats de l'inférence du modèle, je vais l'intégrer avec Slack. Cette intégration me permet d'envoyer des requêtes à la fonction `predict()` via Slack.

Pour cela, j'ai créé une Application sur Slack et configuré le point d'accès (endpoint) (le même endpoint caché montré dans la Figure 4.74). Une fois l'application configurée, j'ai créé une commande slash (slash command) comme suit :

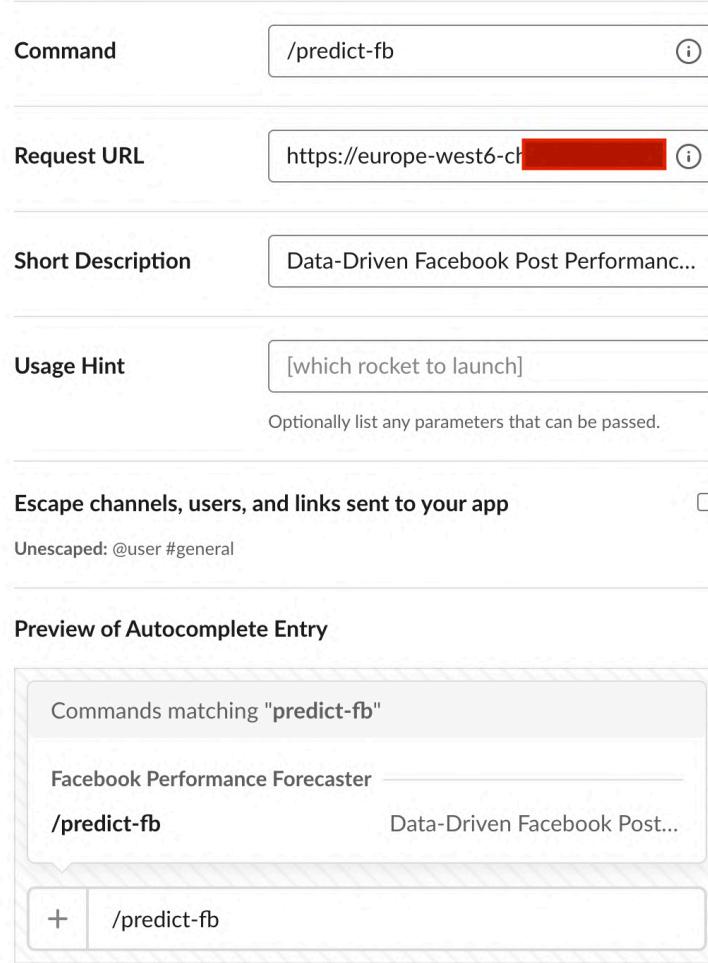


FIG. 4.75 : Slack API Slash Command Configuration

Maintenant, il suffit d'installer l'application sur Slack, d'accorder les autorisations appropriées, et notre fonction est prête à être invoquée depuis Slack.

4.3.5.4 Test de l'inférence du modèle à partir de Slack

Je procède maintenant à un test pour vérifier le bon fonctionnement de l'inférence du modèle à partir de Slack. J'envoie une requête contenant le JSON 4.76 suivant :

```
/predict-fb [{"publish_time": "2023-05-07 13:05:00", "duration_sec": "60", "is_tested": "True", "hour": "2", "is_short": "True", "confidence_level": "3", "day_of_week": "1", "avg_3-second_video_views_last_7_days": "13.780737980146222", "avg_60-second_video_views_by_returning_viewers_last_7_days": "7.225888513590066", "avg_estimated_earnings_by_video_usd_last_7_days": "2.426594545656942", "avg_overall_negative_feedback_per_total_3-second_video_views_last_7_days": "-1.257916966573267", "avg_seconds_viewed_by_returning_viewers_per_seconds_viewed_from_recommendations_last_7_days": "-2.730848097906935"}]]
```

FIG. 4.76 : Model inference request from Slack

Après avoir envoyé la requête, voici la réponse (Fig. 4.77) retournée par l'application **Facebook Performance Forecaster** :

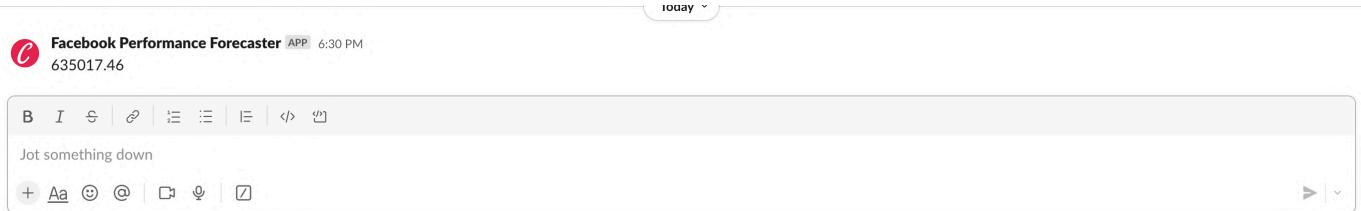


FIG. 4.77 : Response from the Cloud Function

Note : Pour améliorer la formulation du résultat et fournir plus de contexte à l'utilisateur, il est important de préciser le modèle utilisé, sa version et les données d'entrée. De plus, le format JSON peut ne pas être convivial pour l'utilisateur. Il serait préférable de trouver une solution plus conviviale pour permettre à l'utilisateur de saisir ses données de test. Une solution temporaire a été présentée en utilisant Google Sheets. L'utilisateur peut saisir ses données dans un tableau et la commande nécessaire se génère automatiquement. La figure 4.78 illustre ce processus :

	A	B	C	E
1	This Google Sheets aims at providing a tool for users to format their messages to send to a Slack bot /predict-fb for predicting Facebook post performance on Chefclub Network			
2	Before starting, make sure to disable smart quotes for those on MacOS (System Settings > Keyboard > Text Input section > Edit Input Sources > Disable Smart Quotes			
3				
4		1. Changed values in yellow below 2. Send this generated message to any conversation in Slack --> 3. See number of lifetime views predicted by model		
5	duration_sec	200	Seconds	
6	is_tested	FALSE	Facebook A/B post testing	
7	hour	2	Hour of posting (Paris Time)	
8	is_short	FALSE	Calculated (strict comparison to 60 s)	
9	confidence_level	3 - TOP	From video project main variant (not necessarily result on specific variant)	
10	day_of_week	3 - Wednesday	Day of posting (Paris Time)	
11	avg_3-second_video_views_last_7_days	965,113	From tables	
12	avg_60-second_video_views_by_returning_viewers_last_7_days	1,380	From tables	
13	avg_estimated_earnings_by_video_usd_last_7_days	11.36	From tables	
14	avg_overall_negative_feedback_per_total_3-second_video_views_last_7_days	0.28	From tables	
15	avg_seconds_viewed_by_returning_viewers_per_seconds_viewed_from_recommendations_last_7_days	0.07	From tables	

FIG. 4.78 : Slack - Message formatter for prediction bot

Cette approche permet à l'utilisateur de saisir facilement ses données de test dans un environnement familier, puis d'utiliser la commande générée pour effectuer les prédictions à l'aide du modèle spécifié. Cela facilite l'utilisation du bot de prédiction et assure une meilleure expérience utilisateur.

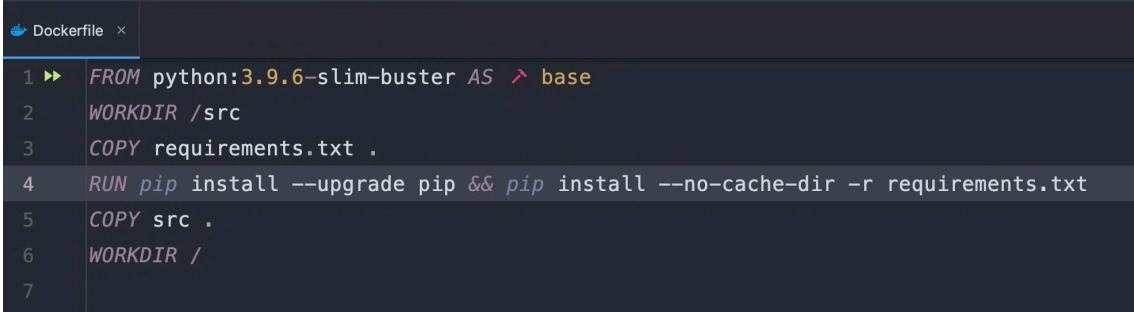
4.3.6 Développement, Test et Orchestration de la Task *Model Train* dans le DAG Facebook

Maintenant que le code responsable est pres et a été testé avec VENV, et que la possibilité d'inference du modèle à partir de slack, maintenant il faut orchestrer le code pour pouvoir Update le modèle de façon quotidienne, il faut aussi s'assurer que la cloud function à toujours accès au dernier modèle ajouté. Par analogie avec le Job Youtube, je vais contourner le code et orchestrer avec Airflow en utilisant le KubernetesPodOperator :

4.3.6.1 Containerization, Test du code avec Docker, suivi de son upload dans l’Artifact Registry

4.3.6.1.1 Conteneurisation et push vers l’Artifact Registry

Comme pour le Job Youtube, le code développé va être exécuté en production. Il est donc nécessaire de le containeriser dans une image Docker et de le sauvegarder dans le Google Cloud Artifact Registry. Pour ce faire, je crée une image Docker à l'aide d'un Dockerfile, qui regroupe toutes les dépendances du code dans l'image. La figure suivante 4.79 montre le Dockerfile utilisé dans notre cas :



```

Dockerfile ×
1 ► FROM python:3.9.6-slim-buster AS base
2 WORKDIR /src
3 COPY requirements.txt .
4 RUN pip install --upgrade pip && pip install --no-cache-dir -r requirements.txt
5 COPY src .
6 WORKDIR /
7

```

FIG. 4.79 : Dockerfile used to build the Model Training image

L'image Docker est ensuite créée en utilisant ce Dockerfile, et voici à quoi elle ressemble (Figure 4.80) :

	Name	Tag	Status	Created	Size	Actions
□	facebook-model-image 4b5cb7659a27	dev	In use	10 minutes ago	970.82 MB	▶ ⚙️ 🗑️

FIG. 4.80 : Model Training Docker image in Docker UI

Pour pousser l'image Docker vers l'Artifact Registry de Google et l'associer à un hachage spécifique de GitHub pour un suivi précis lors du déploiement en production, une commande Make est utilisée. Voici la commande correspondante (Figure 4.81) :

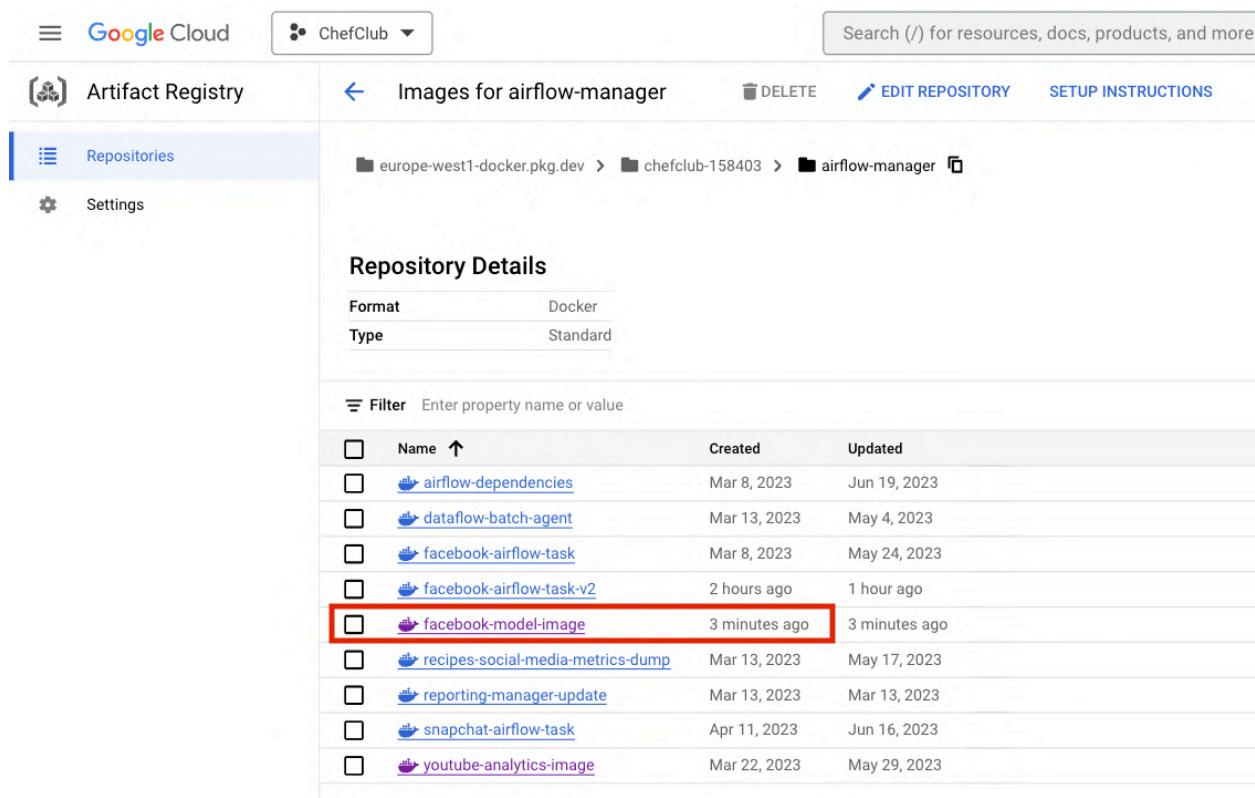
```

(venv) ➔ facebook-model git:(main) ✘ make push-dev-image-with-git-hash
The push refers to repository [europe-west1-docker.pkg.dev/chefclub-158403/airflow-manager/facebook-model-image]
a60469430b53: Pushed

```

FIG. 4.81 : Make command to push Docker image to Artifact Registry

L'image d'entraînement du modèle est ensuite stockée dans le Google Artifact Registry, comme le montre la figure suivante (Figure 4.82) :



The screenshot shows the Google Cloud Artifact Registry interface. The left sidebar has 'Artifact Registry' selected under 'Repositories'. The main area shows a breadcrumb path: europe-west1-docker.pkg.dev > chefclub-158403 > airflow-manager. The title is 'Images for airflow-manager'. Below this is a table titled 'Repository Details' with columns 'Format' (Docker) and 'Type' (Standard). A 'Filter' input field is present. The table lists several Docker images, with the last one, 'facebook-model-image', highlighted by a red box.

Name	Created	Updated
airflow-dependencies	Mar 8, 2023	Jun 19, 2023
dataflow-batch-agent	Mar 13, 2023	May 4, 2023
facebook-airflow-task	Mar 8, 2023	May 24, 2023
facebook-airflow-task-v2	2 hours ago	1 hour ago
facebook-model-image	3 minutes ago	3 minutes ago
recipes-social-media-metrics-dump	Mar 13, 2023	May 17, 2023
reporting-manager-update	Mar 13, 2023	Mar 13, 2023
snapchat-airflow-task	Apr 11, 2023	Jun 16, 2023
youtube-analytics-image	Mar 22, 2023	May 29, 2023

FIG. 4.82 : Model Training image in Google Artifact Registry

4.3.6.1.2 Test et validation du code Python avec Docker

Pour s'assurer que le code fonctionne sur Docker, comme expliqué dans la section 4.2.2.1.2 dédiée au Job Youtube, j'entre dans le conteneur à l'aide de la commande Make et j'exécute le code d'entraînement du modèle. Une fois l'exécution terminée, je reçois la confirmation que le modèle a été correctement entraîné et poussé vers GCS. La figure 4.83 illustre ce processus.

```
(venv) → facebook-model git:(main) ✘ make run-container
[+] Running 1/0
  # Container facebook-model-container  Running
  0.0s

Successfully run : facebook-model
(venv) → facebook-model git:(main) ✘ make open-shell
root@c95327734fff:/# python src/main.py -s 2023-07-02 -e 2023-05-02 -p 'Chefclub Network'
Lasso(alpha=1e-07)
model trained
model uploaded
root@c95327734fff:/#
```

FIG. 4.83 : Testing Model Training Python Code with Docker

4.3.6.2 Intégration avec le DAG Facebook existant

L'objectif de cette section est d'intégrer le code au DAG Facebook existant, c'est-à-dire compléter l'orchestration sur Airflow afin que le modèle se mette à jour directement une fois les données récupérées. Ainsi, le modèle sera basé sur les données les plus récentes.

La figure 4.84 suivante présente le DAG Facebook actuel avant l'ajout de la tâche d'entraînement du modèle :

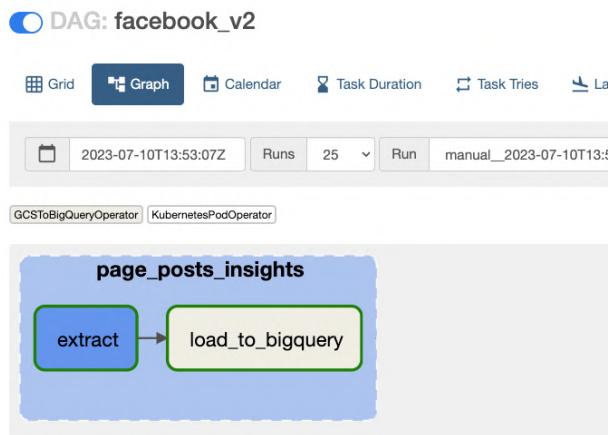


FIG. 4.84 : Current Facebook DAG before adding the Model Training Task

Voici à quoi ressemble le nouveau DAG après l'ajout de la tâche d'entraînement du modèle :

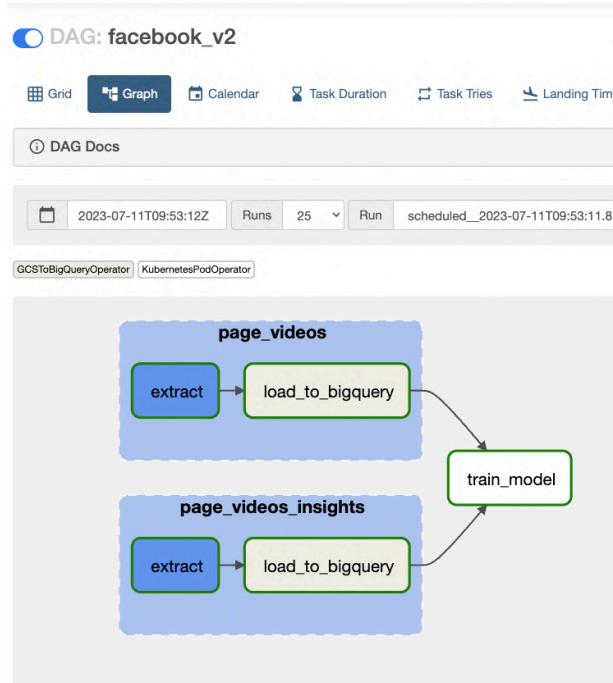


FIG. 4.85 : Updated Facebook DAG after adding the Model Training Task

Voici le code Python que j'ai ajouté pour créer la nouvelle tâche Train Model :

```

1 def train_model() -> KubernetesPodOperator:
2     train_model_task = KubernetesPodOperator(
3         task_id='train_model',
4         image=f'{REGISTRY_ML_IMAGE_NAME}{REGISTRY_ML_IMAGE_TAG}',
5         cmds=["bash", "-cx"],
6         arguments=[
7             "python src/main.py",
8             f"-p {PAGE_NAME}",
9             f"-s {START_DATE}",
10            f"-e {END_DATE}"
11        ],
12        container_constraints={"memory": 384},
13        retries=4,

```

```

14         retry_delay=timedelta(minutes=1),
15         kubernetes_conn_id=None, # Remove this line if not required
16         execution_timeout=timedelta(minutes=30),
17         max_active_tis_per_dag=5,
18         secrets=[datascience_gsa_credentials],
19         env_vars={
20             "DATA_CREDENTIALS_PATH": (f"/{datascience_gsa_credentials.
deploy_target}" +
21                                         f"{datascience_gsa_credentials.key}"),
22             "DATASET_NAME": "facebook_v2",
23             "MODEL_PATH": "model/winner_model.pkl",
24             "BUCKET_NAME": "business-analysis-xxx"
25         },
26     )
27
28     return train_model_task

```

Listing 4.19: Python Code for Model Training Task in Airflow

Voici à quoi ressemble le graphique d'exécution de cette nouvelle tâche sur Airflow (voir la figure 4.86) :

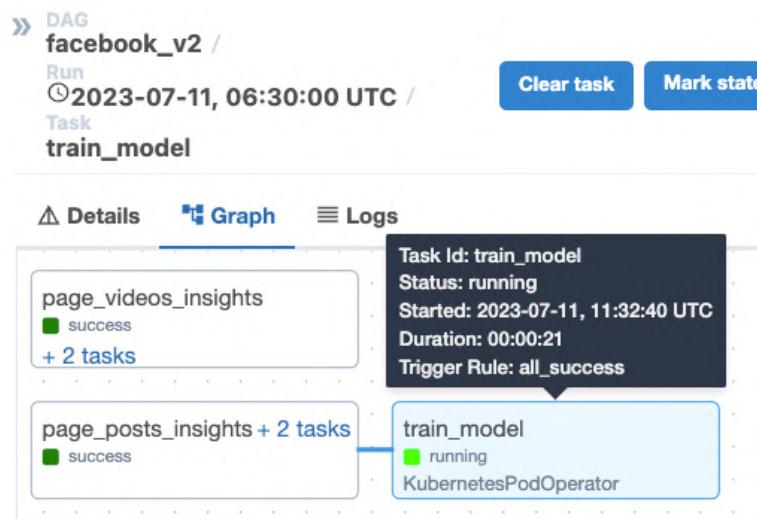


FIG. 4.86 : Running the Model Training Task on Airflow

4.3.6.3 Test et validation de la Task

Afin de tester et valider la nouvelle tâche ajoutée au DAG sur Airflow, il est nécessaire de vérifier que le modèle est automatiquement mis à jour après la récupération des données. Cela signifie qu'à chaque exécution du DAG, un nouveau fichier Pickle est ajouté sur GCS. Pour effectuer cette vérification, il suffit de se rendre sur GCS et de consulter la date de création la plus récente, comme illustré dans la figure 4.87 ci-dessous.

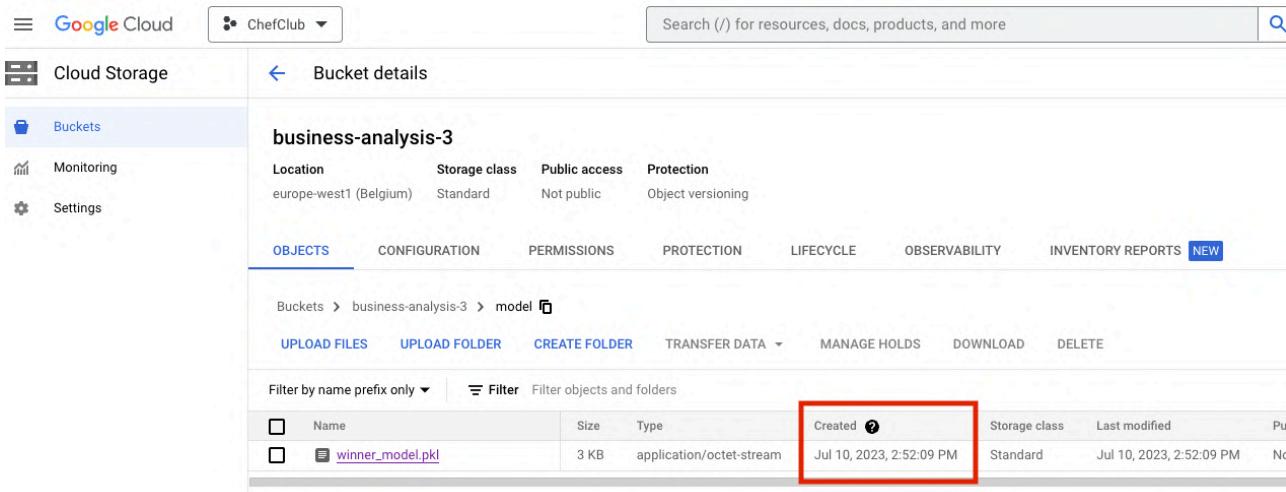


FIG. 4.87 : Updated Model on Google Cloud Storage

4.4 Conclusion

En conclusion, ce chapitre a abordé l'implémentation et la validation des solutions proposées pour répondre aux besoins de Chefclub en utilisant les sciences des données. J'ai suivi les phases de développement et de test afin de concrétiser les concepts définis lors de l'analyse et de la conception. L'objectif était de garantir que les solutions développées répondent aux besoins identifiés et s'intègrent de manière harmonieuse à l'infrastructure de données existante de Chefclub.

Cette implémentation réussie marque une étape clé dans l'évolution de Chefclub, en offrant des solutions pratiques et adaptées à la gestion des données. Elle permettra une utilisation plus efficace des ressources et contribuera à améliorer les performances opérationnelles de l'entreprise.

Le prochain chapitre sera consacré à la conclusion générale de mon projet de fin d'études, dans lequel je récapitulerai les principales réalisations, les enseignements tirés et les perspectives d'avenir.

Conclusion et perspectives

Conclusion générale

Je me suis présenté dans ce rapport sur mon projet de fin d'études réalisé au sein de Chefclub, intitulé « *"Mise en place d'infrastructures pour la collecte, le stockage et l'analyse des données Chefclub"* ».

En combinant la puissance du cloud **GCP** avec des outils et des techniques de science des données, j'ai réussi à déployer et mettre en production une architecture permettant de récupérer périodiquement les données analytiques de **YouTube**. Ces données se sont révélées utiles aux équipes de la finance de Chefclub. J'ai également analysé les performances des publications sur **Facebook** et développé une architecture *end-to-end* automatisant l'entraînement des modèles et la prédiction des performances des posts sur Facebook à partir des performances historiques. J'ai intégré cette solution aux équipes de distribution de Chefclub via **Slack**. Cependant, il reste encore des possibilités d'amélioration, notamment en ajoutant des fonctionnalités de monitoring du modèle et en étendant le déploiement des modèles vers d'autres réseaux sociaux.

Les difficultés rencontrées au cours du projet sont principalement liées à la gestion et à l'automatisation de la récupération de données à partir des API, ainsi qu'à la difficulté d'améliorer les performances des modèles en raison du manque de variables explicatives. De plus, la création des codes **Python dockerisés** est étroitement liée à l'utilisation d'**Airflow**. Ainsi, une compréhension approfondie du fonctionnement de cet outil s'est révélée indispensable pour concevoir une architecture adaptée à nos besoins.

Il conviendrait également d'explorer comment effectuer la transformation des données avec un outil plus approprié comme **DBT** et d'établir une liaison avec **BigQuery**. L'implémentation de la **CI/CD** avec **GitHub Actions** pourrait faciliter les tests et le déploiement des images *dockerisées*, plutôt que d'utiliser des commandes *Make*. Il serait également intéressant d'explorer des modèles de *Deep Learning* plus complexes, susceptibles d'améliorer les performances des prédictions.

Je tiens à souligner que ce stage représente pour moi une expérience riche en termes de connaissances, de savoir-faire et de savoir-être. Il a renforcé et complété mes acquis théoriques et techniques des trois années d'études à l'INPT.

En guise de conclusion, je ne peux qu'être fier et satisfait du bilan positif de mon stage de fin d'études au sein de Chefclub pour ce projet. Il m'a donné l'occasion de travailler sur des sujets tendance tels que le *cloud computing* avec la puissance des sciences des données, de mettre en pratique mes connaissances théoriques, de travailler sur une architecture *end-to-end* en data et surtout d'acquérir de nouvelles compétences. Je suis convaincu que cette expérience aura un impact positif considérable sur mon avenir professionnel en tant qu'ingénieur en science des données.

Bibliographie

- [1] Apache Airflow. Official Website. Retrieved from <https://airflow.apache.org/>
- [2] Docker. Official Website. Retrieved from <https://www.docker.com/>
- [3] Scikit-learn. Official Website. Retrieved from <https://scikit-learn.org/>
- [4] Google Cloud Platform. Official Website. Retrieved from <https://cloud.google.com/>
- [5] Python Programming Language. Official Website. Retrieved from <https://www.python.org/>
- [6] GitHub. Official Website. Retrieved from <https://github.com/>
- [7] YouTube Analytics API. Official Documentation. Retrieved from <https://developers.google.com/youtube/analytics/>
- [8] YAML. Official Website. Retrieved from <https://yaml.org/>
- [9] Slack API. Official Documentation. Retrieved from <https://api.slack.com/>
- [10] Python Pickle Module. Official Documentation. Retrieved from <https://docs.python.org/3/library/pickle.html>
- [11] Python Virtual Environments (venv). Official Documentation. Retrieved from <https://docs.python.org/3/library/venv.html>
- [12] Mermaid. Official Website. Retrieved from <https://mermaid-js.github.io/mermaid/>

Annexes

Annexe A

Codes

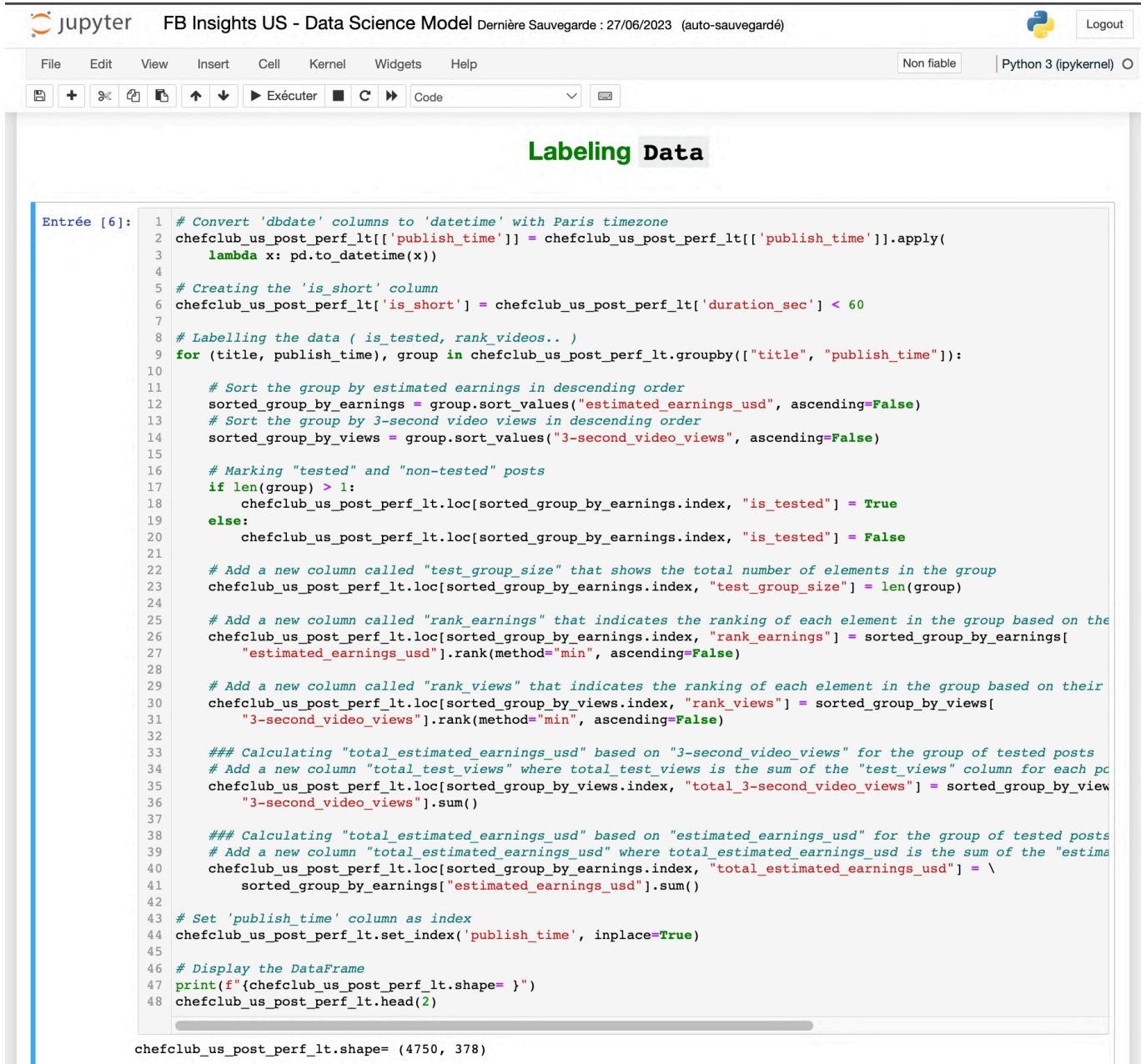
A.1 YouTubeAnalyticsTimeBasedDAG Airflow DAG

```
1
2 # Get the current Kubernetes namespace Airflow is running in
3 NAMESPACE = conf.get("kubernetes", "NAMESPACE")
4 DAG_FOLDER_PATH = os.path.dirname(os.path.realpath(__file__))
5
6 # Load and Extract the channel IDs
7 with open(f"{DAG_FOLDER_PATH}/config/youtube/channels.yaml",
8             "r", encoding="utf-8") as f:
9     channels_file = yaml.safe_load(f)
10 CHANNELS = [
11     {
12         "channel_id": channel["channel_id"],
13         "channel_name": channel["channel_name"],
14         "monetization": channel["monetization"],
15     }
16     for channel in channels_file
17 ]
18
19 # Configuration of the DAG
20 YOUTUBE_DAG_CONFIG = {
21     # Configuration details here...
22 }
23
24 # Data date interval
25 START_DATE = "{{ macros.ds_add(ds, -1) }}"
26 END_DATE = START_DATE
27
28 # BigQuery configuration
29 BQ_PROJECT = "chefclub-xxx"
30 BQ_TEMP_DATASET = "airflow_xxxx_temp"
31
32 with DAG(
33     dag_id="youtube_time-based-day",
34     description="DAG to fetch list of Youtube Analytics data for time-based
35     reports",
36     tags=["social_media", "youtube", "content_type"],
37     schedule_interval=timedelta(days=1),
38     default_args=default_args,
```

```
38 ) as dag:
39     for report in REPORTS:
40         REPORT_NAME = report.get("name", "")
41         REPORT_NAME_CLEANED = REPORT_NAME.replace("time-based-day-", "")
42         REPORT_MONETIZED_CHANNELS_ONLY = report.get("monetized_channels_only")
43         REPORT_BIGQUERY_UPSERT_DESTINATION = report.get("bigquery_upsert_destination"
44 , {})
45         with TaskGroup(group_id=f"{REPORT_NAME_CLEANED}"):
46             for channel in CHANNELS:
47                 channel_id = channel["channel_id"]
48                 channel_name = re.sub(
49                     r" +",
50                     "_",
51                     re.sub(r"[^a-zA-Z0-9 ]", "", channel["channel_name"].lower()),
52                 )
53                 if not REPORT_MONETIZED_CHANNELS_ONLY or bool(channel["monetization"]
54 ]):
55                 with TaskGroup(group_id=f"{REPORT_NAME_CLEANED}_{channel_name}"):
56                     :
57                         FETCH_TASK_ID_FULL = f"{REPORT_NAME_CLEANED}_{channel_name}"
58
59 _fetch"
60
61
62                     if REPORT_BIGQUERY_UPSERT_DESTINATION != {}:
63                         DESTINATION_DATASET_NAME =
64 REPORT_BIGQUERY_UPSERT_DESTINATION.get("dataset_name", "")
65                         DESTINATION_TABLE_NAME =
66 REPORT_BIGQUERY_UPSERT_DESTINATION.get("table_name", "")
67                         TEMPORARY_DESTINATION_TABLE_NAME = f"temp--{channel_name}
68 }--{REPORT_NAME}--{START_DATE}"
69                         # Task to transfer the fetched data (saved in GCS) to a
70 temporary table in BigQuery
71                         gcs_to_bq = GCSToBigQueryOperator(
72                             # Operator details here...
73                         )
74
75                         # Task to merge and upsert data from a temporary table
76 into the native BigQuery table
77                         upsert_table = BigQueryInsertJobOperator(
78                             # Operator details here...
79                         )
80
81                         fetch_youtube_data.set_downstream(gcs_to_bq)
82                         gcs_to_bq.set_downstream(upsert_table)
```

Listing A.1: YouTubeAnalyticsTimeBasedDAG Airflow DAG (Non-Confidential Version)

A.2 Facebook Post Performance Analysis



The screenshot shows a Jupyter Notebook interface with the title "FB Insights US - Data Science Model". The notebook has a single cell labeled "Entrée [6]:" containing Python code for data refinement and labeling. The code is color-coded for syntax highlighting. The cell content includes operations like converting columns to datetime, creating an "is_short" column, labeling data based on title and publish time, and adding new columns for test_group_size, rank_earnings, rank_views, total_3-second_video_views, and total_estimated_earnings_usd. The code concludes by setting the publish_time column as the index and printing the DataFrame shape.

```

Entrée [6]:
1 # Convert 'dbdate' columns to 'datetime' with Paris timezone
2 chefclub_us_post_perf_lt[['publish_time']] = chefclub_us_post_perf_lt[['publish_time']].apply(
3     lambda x: pd.to_datetime(x))
4
5 # Creating the 'is_short' column
6 chefclub_us_post_perf_lt['is_short'] = chefclub_us_post_perf_lt['duration_sec'] < 60
7
8 # Labelling the data ( is_tested, rank_videos... )
9 for (title, publish_time), group in chefclub_us_post_perf_lt.groupby(["title", "publish_time"]):
10
11     # Sort the group by estimated earnings in descending order
12     sorted_group_by_earnings = group.sort_values("estimated_earnings_usd", ascending=False)
13     # Sort the group by 3-second video views in descending order
14     sorted_group_by_views = group.sort_values("3-second_video_views", ascending=False)
15
16     # Marking "tested" and "non-tested" posts
17     if len(group) > 1:
18         chefclub_us_post_perf_lt.loc[sorted_group_by_earnings.index, "is_tested"] = True
19     else:
20         chefclub_us_post_perf_lt.loc[sorted_group_by_earnings.index, "is_tested"] = False
21
22     # Add a new column called "test_group_size" that shows the total number of elements in the group
23     chefclub_us_post_perf_lt.loc[sorted_group_by_earnings.index, "test_group_size"] = len(group)
24
25     # Add a new column called "rank_earnings" that indicates the ranking of each element in the group based on the
26     # estimated_earnings_usd
27     chefclub_us_post_perf_lt.loc[sorted_group_by_earnings.index, "rank_earnings"] = sorted_group_by_earnings[
28         "estimated_earnings_usd"].rank(method="min", ascending=False)
29
30     # Add a new column called "rank_views" that indicates the ranking of each element in the group based on their
31     # 3-second_video_views
32     chefclub_us_post_perf_lt.loc[sorted_group_by_views.index, "rank_views"] = sorted_group_by_views[
33         "3-second_video_views"].rank(method="min", ascending=False)
34
35     ### Calculating "total_estimated_earnings_usd" based on "3-second_video_views" for the group of tested posts
36     # Add a new column "total_test_views" where total_test_views is the sum of the "test_views" column for each post
37     chefclub_us_post_perf_lt.loc[sorted_group_by_views.index, "total_3-second_video_views"] = sorted_group_by_views[
38         "3-second_video_views"].sum()
39
40     ### Calculating "total_estimated_earnings_usd" based on "estimated_earnings_usd" for the group of tested posts
41     # Add a new column "total_estimated_earnings_usd" where total_estimated_earnings_usd is the sum of the "estimated_earnings_usd"
42     # for each post
43     chefclub_us_post_perf_lt.loc[sorted_group_by_earnings.index, "total_estimated_earnings_usd"] = \
44         sorted_group_by_earnings["estimated_earnings_usd"].sum()
45
46     # Set 'publish_time' column as index
47     chefclub_us_post_perf_lt.set_index('publish_time', inplace=True)
48
49     # Display the DataFrame
50     print(f"{chefclub_us_post_perf_lt.shape= }")
51     chefclub_us_post_perf_lt.head(2)
52
53
54 chefclub_us_post_perf_lt.shape= (4750, 378)

```

FIG. A.1 : Data Refinement and Labeling for Facebook Post Performance Analysis