



RAPPORT BIG DATA

Réalisé par :

Edner GOMA

Herman SESSOU

Nour BEN MILED

Thibault GARCIA-MEGEVAND

Encadré par :

Professeur: Rakib SHEIKH

2025-2026

Sommaire

Introduction.....	3
I/ Architecture globale.....	4
II/ Collecte et prétraitement des données.....	5
1. Collecte des données.....	5
2. Prétraitement.....	6
3. Datawarehouse et modélisation.....	7
III/ Visualisation.....	9
IV/ Machine learning.....	12
1. Données utilisées.....	12
2. Modèle.....	12
3. Séparation & entraînement.....	13
4. Évaluation des performances.....	13
5. Qualité du code et test unitaires.....	15
Conclusion.....	16
Bibliographie.....	17
Annexe.....	18

Introduction

Au fil des siècles, la quantité des données produites ne cesse d'augmenter, tant en volume qu'en diversité. Savoir les exploiter constitue un enjeu majeur pour les grandes entreprises. Capables de gérer l'ensemble du cycle de vie de l'information, la conception d'architectures de données représente un enjeu technique et méthodologique important. C'est dans ce contexte que s'inscrit le projet que nous allons présenter dans ce rapport.

L'objectif de ce projet est de concevoir et de déployer une architecture Big Data complète de type CAC 40, allant de la collecte des données à des modèles de machine learning, en passant par l'analyse décisionnelle. Le projet s'appuie sur les données des courses de taxis jaunes de la ville de New York, fournies sous forme de fichiers Parquet mensuels. Ces données seront d'abord stockées dans un Data Lake représenté par MinIO, puis nettoyées et transformées à l'aide du moteur de calcul distribué Apache Spark. Les données traitées seront ensuite intégrées dans une base PostgreSQL, jouant le rôle de Data Warehouse, et organisées selon un modèle multidimensionnel optimisé pour l'analyse analytique.

L'enjeu de notre rapport sera le suivi de l'évolution des taxis jaunes de la ville de New York en analysant l'influence de facteurs tels que la localisation géographique, les distances parcourues et les montants des courses, la qualité du service, etc.

Dans ce document, nous commencerons par présenter l'architecture globale du projet. Nous détaillerons ensuite les différentes étapes de collecte et de prétraitement des données. La troisième partie sera consacrée à la modélisation du Data Warehouse ainsi qu'à l'intégration des données. Enfin, nous aborderons l'aspect visualisation et présenterons les différents modèles de machine learning mis en œuvre.

I/ Architecture globale

L'architecture dans ce projet est une architecture qui s'inspire des architectures utilisées par les grandes entreprises. Elle repose sur une pipeline de données de type ETL (voir Figure 1) couvrant l'ensemble du cycle. Cette pipeline est conçue de manière à garantir la scalabilité, la maintenabilité et la reproductibilité de la chaîne des traitements, pour être conforme aux pratiques courantes en ingénierie des données et aux exigences du projet.

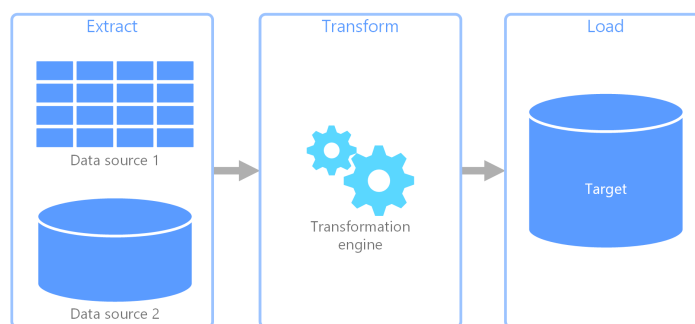


Figure 1: Architecture simplifiée du pipeline ETL

Pour cela, nous utilisons les outils suivants :

- **MinIO** : Data Lake utilisé pour stocker les données brutes dans le bucket *nyc-raw*, ainsi que les données nettoyées dans le bucket *nyc-clean*.
- **Apache Spark** : moteur de calcul distribué dédié au nettoyage des données et à leur ingestion vers le Data Warehouse. Dans notre contexte, il est utilisé avec le langage **Scala**.
- **PostgreSQL** : système de gestion de base de données relationnelle assurant le rôle de Data Warehouse pour le stockage des données structurées.
- **Docker** : l'ensemble du pipeline est déployé au sein d'un environnement conteneurisé afin de garantir la portabilité et la reproductibilité de l'infrastructure.

II/ Collecte et prétraitement des données

1. Collecte des données

Cette première étape consiste à récupérer les fichiers sources depuis le site officiel du gouvernement de New York ([nyc-gov](https://www.nyc.gov)).

Dans notre cas, nous avons choisi de récupérer le fichier parquet Yellow Taxi Trip Records qui correspond aux courses de taxis jaunes de New York. Ce fichier contient des informations sur les trajets, incluant horaires de prise en charge et de dépose, zones géographiques, distances parcourues, tarifs appliqués et pourboires.

Afin de mettre en place un Data Lake local, le service MinIO a été déployé à l'aide de Docker Compose. MinIO est un système de stockage objet compatible avec l'API Amazon S3, largement utilisé dans les architectures Big Data modernes. Il permet de stocker des fichiers sous forme d'objets dans des conteneurs appelés *buckets*.

MinIO est un Data Lake dans le cadre de notre projet, dans celui-ci deux buckets ont été créés afin de structurer le stockage des données :

- le bucket **nyc-raw**, destiné à stocker les données brutes,
- le bucket **nyc-clean**, destiné à accueillir les données nettoyées.

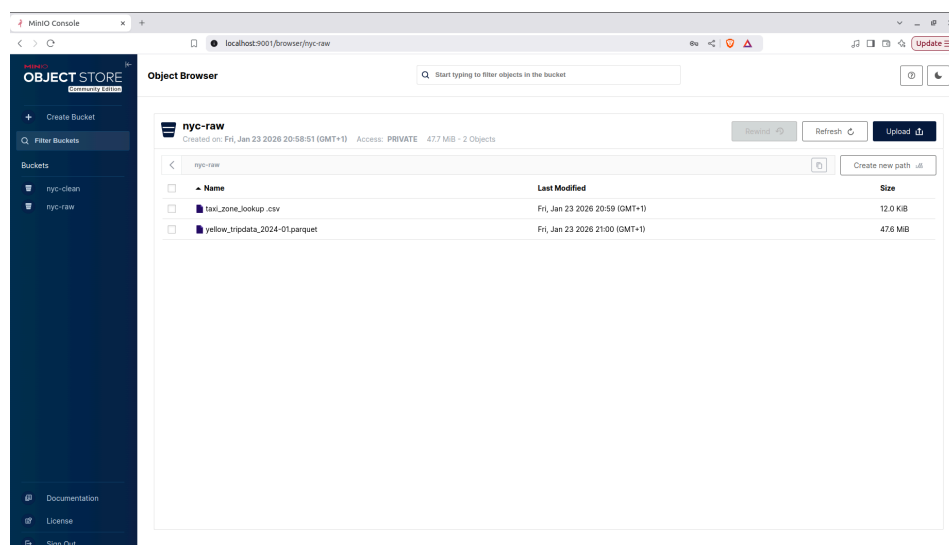


Figure 2 – Interface de gestion des Buckets

Une fois les données insérées dans le bucket nyc-raw, nous passons à la phase de nettoyage des données.

2. Prétraitement

L'objectif de cette partie est d'orchestrer l'ingestion des données brutes de taxis, de garantir leur qualité par un nettoyage rigoureux, et de distribuer les résultats vers deux destinations distinctes afin de servir des usages différents (Data Lake et Data Warehouse).

Architecture du flux de données :

La pipeline suit une architecture de branchement après une étape commune de validation. Le flux s'exécute selon les étapes suivantes :

1. **Ingestion (Extract)** : Lecture des fichiers Parquet depuis le bucket MinIO nyc-raw via le connecteur S3A.
2. **Transformation & Nettoyage** : Filtrage des données aberrantes et enrichissement (calcul de la durée).
3. **Branchement (Load)** :
 - **Branche 1 (Stockage Froid/Data Lake)** : Écriture des données complètes et propres dans le bucket MinIO nyc-clean au format Parquet.
 - **Branche 2 (Stockage Chaud/Data Warehouse)** : Sélection des colonnes métier et insertion dans la base relationnelle PostgreSQL pour l'analyse BI.

Règles de gestion et Nettoyage:

Afin de garantir la fiabilité des analyses, nous avons appliqué une série de filtres stricts sur le DataFrame brut :

- **Intégrité temporelle** : Suppression des courses dont les horodatages (pickup/dropoff) sont nuls.

- **Cohérence métier** : Suppression des distances nulles, des montants de course négatifs ou nuls, et des nombres de passagers incohérents.
- **Enrichissement** : Création de la variable trip_duration_min (différence entre l'arrivée et le départ). Une validation supplémentaire (duration > 0) est appliquée pour éliminer les erreurs de saisie temporelle.

3. Datawarehouse et modélisation

Nous avons choisi une modélisation en étoile pour le DWH (Datawarehouse), avec une table de faits centrale **t_taxi_jaune** qui représentent les trajets des taxis jaunes et des tables de dimensions décrivant le contexte d'analyse. Ce choix s'explique par la nature analytique des données et de la problématique que nous avons choisi d'explorer. La table des faits **t_taxi_jaune** regroupe les mesures quantitatives telles que la distance parcourue, le montant total d'une course, le nombre de passagers et le montant des pourboires, tandis que la dimension géographique (table **t_zone**), provenant du fichier de correspondance des zones [Taxi Zone Lookup Table](#) , permet d'avoir des informations tel que le nom sur les zones de prise en charge et de dépose.

La dimension temporelle **t_dimension_temps** complète le modèle en facilitant les analyses calendaires grâce à des colonnes comme l'année, le mois ou le jour de la semaine. Cette table a été trouvée sur internet. Ce modèle en étoile (**figure 3**) améliore la lisibilité du schéma, limite la complexité et optimise les performances des requêtes analytiques, ce qui le rend facilement exploitable pour l'outil de reporting (Power BI) que nous allons utiliser.

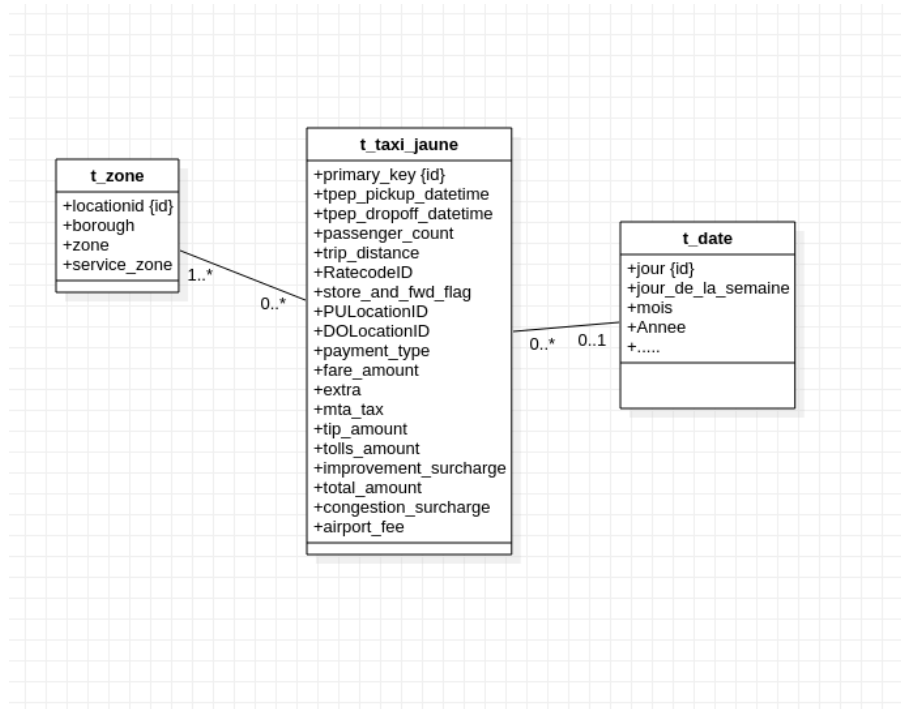


Figure 3 : Schéma de la modélisation en étoile du Data Warehouse

Après la modélisation de notre base de données, nous avons procédé à la création des différentes tables dans le datawarehouse. Pour la création des ces tables, nous avons utilisé un script sql de création, en ce qui concerne l’insertion nous avons utilisé un script insertion.sql . La figure 4 illustre la base de données postgresql que nous utilisons. On y retrouve deux schémas : un schéma config qui regroupe la table temporelle et un autre schéma pour le reste des tables.

	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	improvement_surcharge	total_amount	congestion_surcharge	airport_fee
1	2024-01-13 03:18:09	2024-01-13 03:24:37	1	2.31	1	N	48	68	1	11.4								
2	2024-01-13 03:52:58	2024-01-13 04:01:18	1	2.61	1	N	231	164	1	13.5								
3	2024-01-13 03:26:02	2024-01-13 03:34:43	1	1.79	1	N	90	233	1	11.4								
4	2024-01-13 03:53:44	2024-01-13 04:10:56	1	6.58	1	N	141	244	1	27.5								
5	2024-01-13 03:56:28	2024-01-13 04:14:53	1	3.4	1	N	79	246	1	18.4								
6	2024-01-13 03:54:24	2024-01-13 04:03:58	1	2.85	1	N	140	74	2	14.2								
7	2024-01-13 03:06:55	2024-01-13 03:50:08	3	16.25	3	N	48	1	1	88.1								
8	2024-01-13 03:22:26	2024-01-13 03:30:50	2	1.35	1	N	249	144	1	10								
9	2024-01-13 03:21:19	2024-01-13 03:46:54	1	5.72	1	N	164	25	1	28.9								
10	2024-01-13 03:13:35	2024-01-13 03:40:25	1	8.4	1	N	90	181	1	35.2								
11	2024-01-13 03:22:52	2024-01-13 03:45:47	1	11.83	1	N	161	138	1	46.4								
12	2024-01-13 03:20:52	2024-01-13 03:54:01	1	7.08	1	N	107	188	1	37.3								
13	2024-01-13 03:47:40	2024-01-13 04:08:34	3	4.43	1	N	211	225	1	24.7								
14	2024-01-13 03:31:39	2024-01-13 03:34:13	1	0.5	1	N	48	48	1	5.1								
15	2024-01-13 03:46:31	2024-01-13 03:58:40	1	2.87	1	N	263	164	1	14.2								
16	2024-01-13 03:47:31	2024-01-13 04:48:06	1	30.6	1	N	230	44	1	116.4								
17	2024-01-13 03:40:35	2024-01-13 03:45:55	1	1.68	1	N	79	137	1	8.6								
18	2024-01-13 03:10:37	2024-01-13 03:22:17	1	2	1	N	141	161	3	12.1								
19	2024-01-13 03:39:41	2024-01-13 03:59:26	1	3.61	1	N	48	145	4	21.9								
20	2024-01-13 03:39:41	2024-01-13 03:59:26	1	3.61	1	N	48	146	4	21.9								

Figure 4: Capture d’écran de la base de données

Le détail de chaque colonne de la table **t_taxi_jaune** est présenté en annexe, en fin du rapport. Celui-ci apporte des explications sur le contenu de chacune des colonnes.

III/ Visualisation

Pour cette partie, nous avons créé un rapport pour visualiser les données à l'aide de Power BI. Le rapport s'intitule "visualisation_data_taxi_jaune" il repose principalement sur les données du mois de Janvier qui est un mois caractérisé par un retour de vacances et une relance de l'activité après les fêtes de fin d'année.

Nous allons présenter brièvement les différentes pages qui le composent ainsi que les indicateurs clés pour s'assurer du suivi des taxis jaunes. Pour se mettre dans le contexte, à cette époque la population de New York était estimée à 19,87 millions.

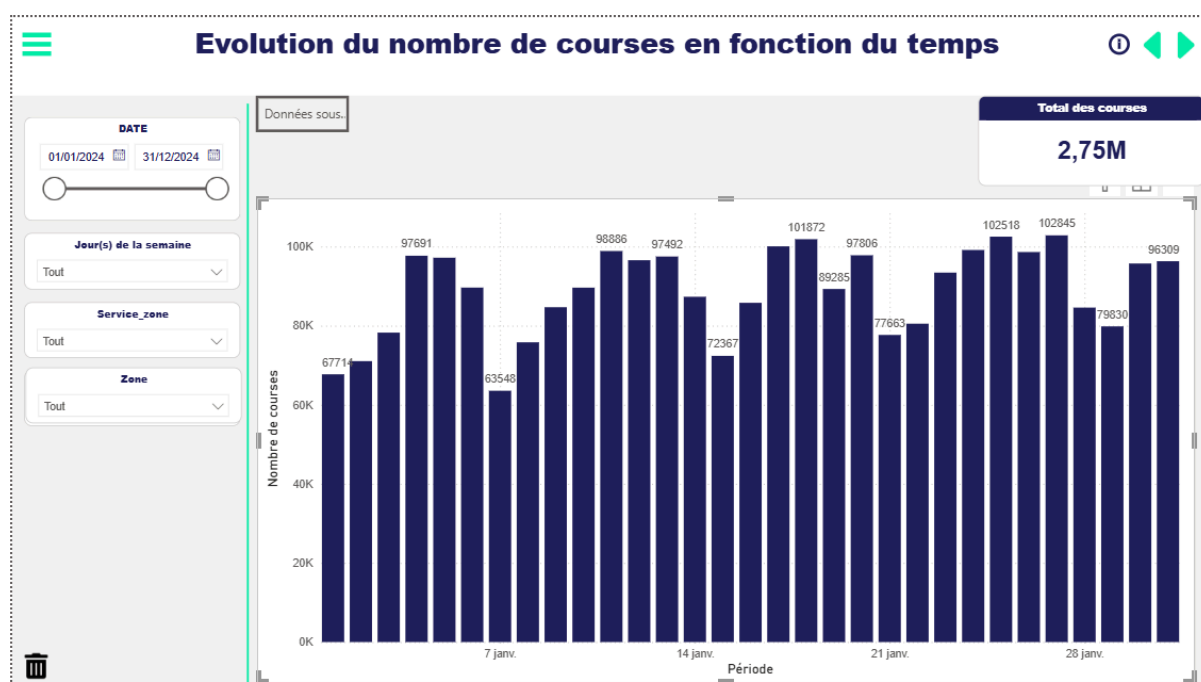


Figure 5: Evolution du nombre de courses en fonction du temps

La figure 5 présente l'évolution quotidienne du nombre de courses sur le mois de Janvier 2024. On observe une activité globalement stable autour de 45 000 à 55 000

courses par jour, avec des variations modérées selon les dates. Le volume total atteint 2,75 million de courses, traduisant une forte demande sur l'ensemble de la période.

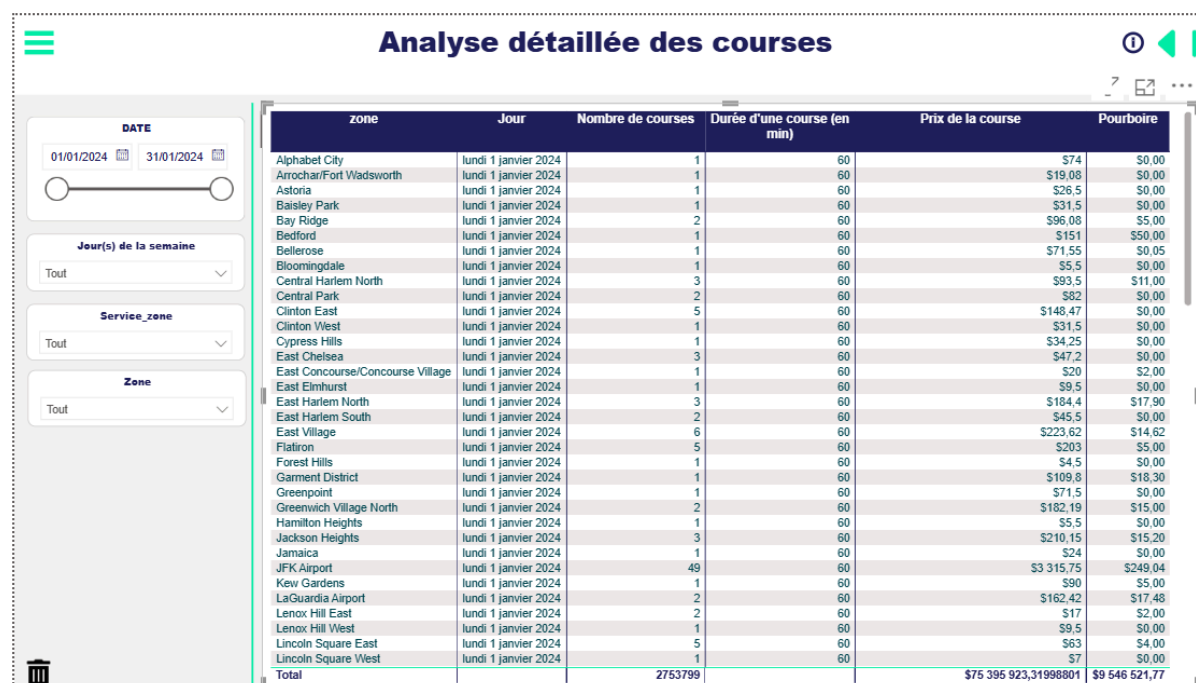


Figure 6: Analyse détaillée des courses

La figure 6 quant à elle présente une vue détaillée du nombre de courses pour la période sélectionnée, en les déclinant par jour et par zone géographique. Elle met en évidence le nombre de prises en charge, la durée des courses, ainsi que le prix associé pour chaque course.

L'analyse montre une hétérogénéité significative entre les zones, tant en termes de volume de courses que de prix moyens. Certaines zones présentent des trajets qui mettent plus de temps que d'autres, mais on peut aussi voir que le prix d'une course ne se base pas que sur la distance, plusieurs facteurs entrent en compte.

Ce niveau de granularité permet d'identifier précisément les zones à forte activité ou à forte rentabilité et constitue une base pertinente pour des analyses plus avancées, notamment la modélisation prédictive des prix et l'étude de l'impact des facteurs spatiaux et temporels sur l'activité des taxis.

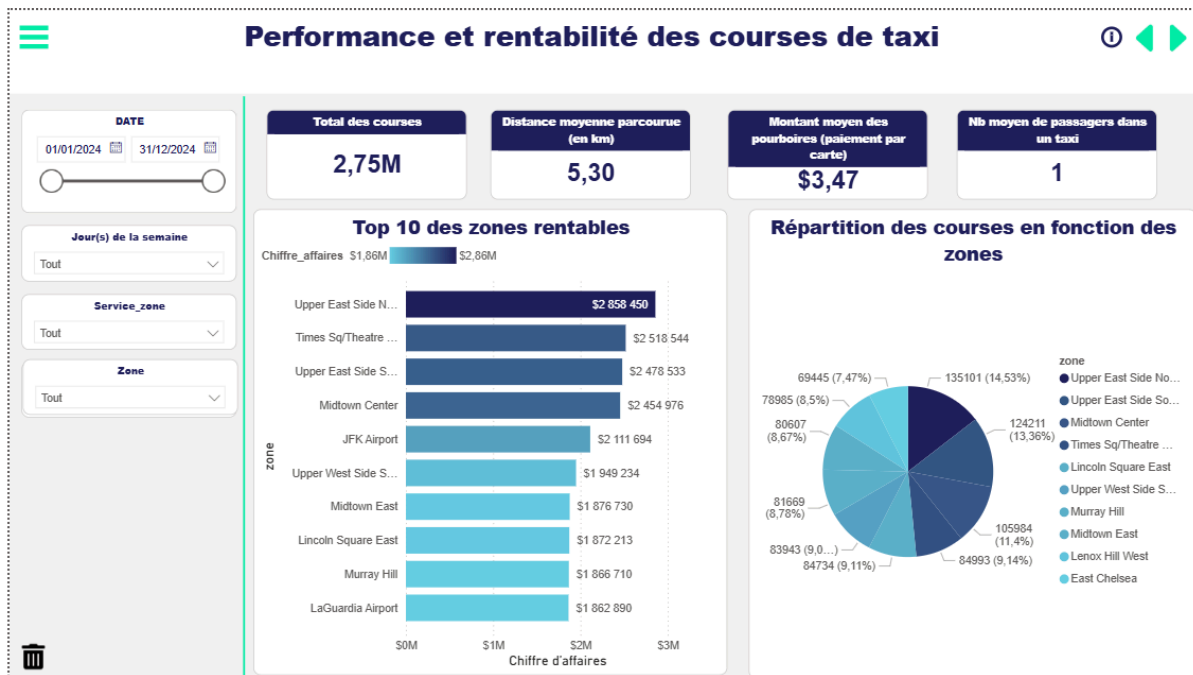


Figure 7: Performance et rentabilité des courses de taxi

Le graphique ci-dessus de synthèse présente les principaux indicateurs de performance de l'activité des taxis sur la période étudiée. Au cours du mois de janvier 2024, le nombre total de courses des taxis jaunes est d'environ 2,75 millions, avec une distance moyenne de 5,30 km, un pourboire moyen de 3,47 dollars et un passager par course en moyenne, indiquant une majorité de trajets individuels.

L'analyse met en évidence une forte concentration du **chiffre d'affaires** dans certaines zones centrales, notamment **Upper East Side**, **Times Square/Theatre District** et **Midtown Center**. Cela s'explique par la forte concentration de population dans ces différentes zones. La répartition des courses par zone confirme cette concentration tout en révélant une diversité géographique de l'activité.

Le **chiffre d'affaires** est calculé comme la différence entre le montant total des courses et le montant total des péages (**Chiffre d'affaires = SUM(total_amount) - SUM(tolls_amount)**), représentant le revenu réellement généré par les courses avant déduction fiscale.

IV/ Machine learning

L'objectif de cette partie est de concevoir un modèle de machine learning permettant de prédire le prix total d'une course de taxi à partir des données des taxis jaunes de la ville de New York. Cette étape vise à introduire les principes fondamentaux du MLOps, notamment la séparation entre les phases d'entraînement et d'inférence, l'industrialisation du code et la mise en place de tests unitaires.

1. Données utilisées

Les données proviennent des fichiers de trajets des taxis jaunes de New York, initialement stockés sous forme de fichiers Parquet dans un Data Lake (Minio).

La variable cible du modèle est le **prix total de la course** (*total_amount*), correspondant au montant global facturé au client.

Les principales variables explicatives retenues sont :

- la distance parcourue (*trip_distance*)
- les composantes tarifaires, incluant le tarif de base (*fare_amount*), le pourboire (*tip_amount*), les péages (*tolls_amount*) et les frais aéroportuaires (*airport_fee*)
- les informations temporelles de prise en charge, telles que la date et l'heure du début de la course (*tpep_pickup_datetime*), à partir desquelles sont extraites des variables temporelles (heure, jour de la semaine, etc.).

2. Modèle

Le modèle retenu pour le rendu final est un **Random Forest Regressor**. Ce choix s'explique par sa robustesse face au bruit, sa capacité à modéliser des relations non linéaires et son bon compromis entre performance et généralisation.

L'entraînement du modèle est réalisé à l'aide d'un script Python dédié (`train.py`), sans recourir à des notebooks, conformément aux contraintes de l'énoncé. Les données sont divisées en un ensemble d'entraînement et un ensemble de tests afin d'évaluer les performances du modèle de manière objective.

3. Séparation & entraînement

Dans une logique inspirée des pratiques MLOps, les phases d'entraînement et d'inférence sont clairement séparées :

- Le script **`train.py`** entraîne le modèle et le sauvegarde sous forme sérialisée à l'aide de la bibliothèque **`joblib`** ;
- Le script **`predict.py`** charge le modèle sauvegardé et génère des prédictions sur de nouvelles données, sans réentraîner le modèle.

Cette organisation permet de simuler un déploiement réel du modèle dans un environnement de production.

4. Évaluation des performances

Les performances du modèle ont été évaluées à l'aide des métriques MAE (Mean Absolute Error) et RMSE (Root Mean Squared Error), permettant de mesurer l'écart entre les valeurs réelles et les valeurs prédites. Les résultats obtenus sont les suivants :

```
RMSE (Random Forest) : 3.68  
MAE (Random Forest) : 0.16
```

Ces valeurs indiquent que le modèle présente une bonne précision, avec une erreur moyenne largement inférieure au seuil attendu de 10, conformément aux exigences de l'exercice.

En complément, une analyse de corrélation a été réalisée afin d'étudier les relations entre les variables explicatives et la variable cible. La matrice de corrélation met en évidence une **très** forte corrélation positive entre **`fare_amount`** et **`total_amount`** (~ 0.98), confirmant que le tarif de base constitue le principal facteur explicatif du prix

total. Les composantes tarifaires telles que les pourboires, les péages et les frais aéroport présentent également des corrélations significatives. En revanche, les variables temporelles (heure, jour de la semaine, mois) montrent des corrélations plus faibles, traduisant un impact plus modéré et souvent non linéaire sur le prix, bien capté par le modèle Random Forest.

Ces résultats confirment la pertinence des variables sélectionnées et la capacité du modèle à fournir des prédictions globalement satisfaisantes.

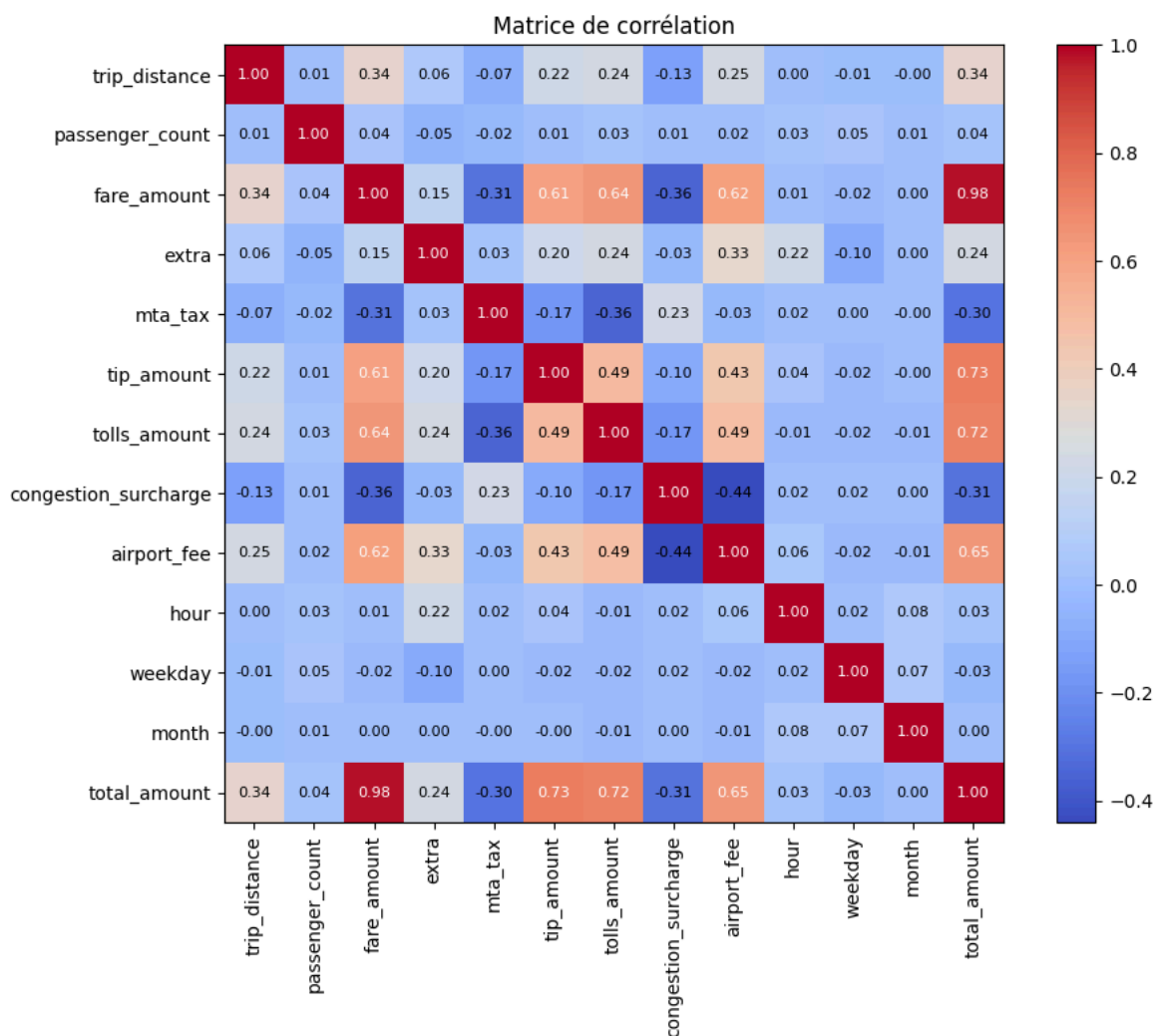


Figure 8: Matrice de Corrélation

5. Qualité du code et test unitaires

La qualité du code a été vérifiée à l'aide de flake8, garantissant le respect de la norme PEP 8. Le code est documenté avec pyment selon la convention NumpyDoc, afin d'en améliorer la lisibilité et la maintenabilité. Des tests unitaires ont été mis en place pour valider les données en entrée ainsi que le bon fonctionnement du modèle lors des phases d'entraînement et d'inférence, assurant ainsi la fiabilité du pipeline de machine learning.

Conclusion

Ce projet a permis de concevoir et de déployer une architecture Big Data complète de type "CAC 40", couvrant l'intégralité du cycle de vie de la donnée, depuis sa collecte brute jusqu'à sa valorisation par le Machine Learning. En nous appuyant sur les données des taxis jaunes de New York, nous avons mis en place une infrastructure robuste et scalable utilisant des technologies de pointe telles que MinIO pour le Data Lake, Apache Spark pour le traitement distribué et PostgreSQL pour le Data Warehouse.

La mise en œuvre d'une pipeline ETL rigoureuse a été l'un des piliers de ce travail. Le nettoyage systématique des données aberrantes et l'enrichissement des variables ont garanti la fiabilité des analyses ultérieures. Grâce à une modélisation en étoile optimisée pour l'analyse décisionnelle, nous avons pu transformer des données brutes massives en indicateurs clairs, révélant une activité intense et concentrée dans les zones stratégiques de Manhattan, comme l'Upper East Side et Times Square.

Enfin, le volet Machine Learning a démontré la valeur prédictive de notre architecture. L'utilisation du modèle Random Forest Regressor a permis de prédire avec une grande précision le prix des courses, validant ainsi la pertinence des facteurs temporels et géographiques sélectionnés. Au-delà des résultats techniques, ce projet a permis de mettre en pratique des principes fondamentaux du MLOps et de l'ingénierie logicielle, tels que l'industrialisation du code, la documentation et les tests unitaires, assurant ainsi la maintenabilité et la qualité de la solution produite.

Bibliographie

[TLC Trip Record Data - TLC](#)

Figure 1:

<https://learn.microsoft.com/fr-fr/azure/architecture/data-guide/relational-data/etl>

Annexe

This data dictionary describes yellow taxi trip data.

For data dictionaries involving other trip types, and metadata like the TLC Taxi Zones, please visit http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml.

Field Name	Description
VendorID	A code indicating the TPEP provider that provided the record. 1 = Creative Mobile Technologies, LLC 2 = Curb Mobility, LLC 6 = Myle Technologies Inc 7 = Helix
tpep_pickup_datetime	The date and time when the meter was engaged.
tpep_dropoff_datetime	The date and time when the meter was disengaged.
passenger_count	The number of passengers in the vehicle.
trip_distance	The elapsed trip distance in miles reported by the taximeter.
RatecodeID	The final rate code in effect at the end of the trip. 1 = Standard rate 2 = JFK 3 = Newark 4 = Nassau or Westchester 5 = Negotiated fare 6 = Group ride 99 = Null/unknown
store_and_fwd_flag	This flag indicates whether the trip record was held in vehicle memory before sending to the vendor, aka "store and forward," because the vehicle did not have a connection to the server. Y = store and forward trip N = not a store and forward trip
PULocationID	TLC Taxi Zone in which the taximeter was engaged.
DOLocationID	TLC Taxi Zone in which the taximeter was disengaged.
payment_type	A numeric code signifying how the passenger paid for the trip. 0 = Flex Fare trip 1 = Credit card 2 = Cash 3 = No charge 4 = Dispute 5 = Unknown 6 = Voided trip
fare_amount	The time-and-distance fare calculated by the meter. For additional information on the following columns, see https://www.nyc.gov/site/tlc/passengers/taxi-fare.page
extra	Miscellaneous extras and surcharges.
mta_tax	Tax that is automatically triggered based on the metered rate in use.
tip_amount	Tip amount – This field is automatically populated for credit card tips. Cash tips are not included.
tolls_amount	Total amount of all tolls paid in trip.
improvement_surcharge	Improvement surcharge assessed trips at the flag drop. The improvement surcharge began being levied in 2015.
total_amount	The total amount charged to passengers. Does not include cash tips.
congestion_surcharge	Total amount collected in trip for NYS congestion surcharge.
airport_fee	For pick up only at LaGuardia and John F. Kennedy Airports.
cbd_congestion_fee	Per-trip charge for MTA's Congestion Relief Zone starting Jan. 5, 2025.