



# BIG DATA

Deploy :

## Simulateur de tarifs Taxi New-Yorkais

Prédez le coût total d'une course en fonction du trajet et de l'heure.

<b>Paramètres du trajet</b>	
<b>Où et Combien ?</b>	<b>Quand ? (Par défaut : Maintenant)</b>
Quartier de départ	Date de la course
Auburndale	2026/02/11
Quartier d'arrivée	Heure de la course
Allerton/Pelham Gardens	19:34
Nombre de passagers	<small>Astuce : La distance impacte le prix de base, mais l'heure et les quartiers déterminent les suppléments. (heures de pointe, frais de congestion, etc.).</small>
3	
Distance estimée (en miles)	
2.50	
<b>Calculer le prix estimé</b>	

Estimation du Total Amount  
**26.26 \$**

Estimation basée sur les données historiques de 2024. Inclut les frais de base et taxes.

14/02/2026

Amine AIT MOUSSA  
Siham DAANOUNI  
Haoxin LIU  
Wangjiachen ZHAO  
ING3 IAC

Professeur : Rakib SHEIKH

# SOMMAIRE

<b>1. Introduction.....</b>	<b>3</b>
<b>2. Infrastructure et Flux d'Ingestion.....</b>	<b>3</b>
2.1 Environnement Conteneurisé.....	3
2.2 Collecte et Ingestion Multi-Branche (Exercices 1 et 2).....	3
2.3 Gestion des volumes massifs.....	4
<b>3. Modélisation du Data Warehouse (Exercice 3).....</b>	<b>4</b>
3.1 Choix du schéma en étoile.....	4
3.2 Architecture et Intégrité.....	4
<b>4. Visualisation de Données (Exercice 4).....</b>	<b>5</b>
<b>5. Machine Learning Qualité Logicielle (Exercice 5).....</b>	<b>6</b>
5.1 Modélisation et Résultats.....	6
5.2 Industrialisation et Tests.....	6
<b>6. Orchestration (Bonus Exercice 6).....</b>	<b>7</b>
6.1 Automatisation via Apache Airflow.....	7
6.2 Monitoring "CAC 40".....	8
<b>7. Conclusion.....</b>	<b>9</b>

# 1. Introduction

Notre projet du module Big Data s'inscrit dans la mise en œuvre d'une architecture complète, visant à traiter et analyser les données officielles de transport de la ville de New York. L'objectif est de construire un pipeline industriel capable de transformer des données brutes massives en informations exploitables via un dashboard décisionnel et un service de prédiction.

L'architecture repose sur une segmentation en trois couches :

**Data Lake (Couche Bronze/Silver)** : Utilisation de MinIO pour le stockage objet des fichiers Parquet bruts.

**ETL & Data Warehouse (Couche Gold)** : Emploi d'Apache Spark (Scala) pour le nettoyage et l'ingestion dans PostgreSQL, structuré en schéma en étoile.

**Service de Prédiction** : Module de Machine Learning en Python pour l'estimation des tarifs, validé par des tests unitaires et la norme PEP 8.

Ce rapport détaille les défis de traitement de volumes importants et les performances de notre modèle (RMSE de 4,25), dépassant les exigences initiales (RMSE < 10).

# 2. Infrastructure et Flux d'Ingestion

## 2.1 Environnement Conteneurisé

L'ensemble de l'infrastructure est orchestrée via **Docker Compose** pour garantir la portabilité. MinIO simule un Data Lake industriel pour le stockage des fichiers Parquet. PostgreSQL pour un serveur de base de données relationnelle faisant office de Data Warehouse.

## 2.2 Collecte et Ingestion Multi-Branche (Exercices 1 et 2)

Le processus a été automatisé en Scala pour assurer un flux direct depuis la source vers le Cloud local.

Nous commençons avec une acquisition des données au format Parquet depuis le site de l'État de New York directement vers le bucket nyc-raw de MinIO. Nous enchaînons ensuite avec un traitement Spark (Data Ingestion). Un job Spark qui traite les données brutes avec une logique de "branching":

**Branche 1** : Sauvegarde des données nettoyées en format Parquet dans MinIO pour l'entraînement ML.

**Branche 2** : Ingestion vers le Data Warehouse PostgreSQL après transformation en mémoire.

```
aminedesel@aminedesel-B550-AORUS-ELITE-AX-V2:~/Desktop/BigData_projet/ex02_data_ingestion$ docker exec -it postgres-db psql -U myuser -d taxidb -c "SELECT count(*) FROM dwh.fact_trip;"  
count  
-----  
8479488  
(1 row)
```

## 2.3 Gestion des volumes massifs

Le défi majeur a été l'ingestion du dataset 2024. L'utilisation de Spark a permis de partitionner les données, évitant les erreurs de mémoire (OOM) rencontrées avec des librairies classiques.

# 3. Modélisation du Data Warehouse (Exercice 3)

## 3.1 Choix du schéma en étoile

Nous avons implémenté un schéma en étoile dans PostgreSQL. Bien que le contrat initial suggérait un modèle en flocon, nous avons privilégié l'étoile pour la performance et la simplicité. Cela implique moins de jointures complexes pour les requêtes analytiques, ainsi qu'une compatibilité directe et performante avec les outils de Data Visualization.

## 3.2 Architecture et Intégrité

Le Datamart s'articule autour d'une table de faits centrale (fact\_trip) et de tables de dimensions (dim\_date, dim\_location, dim\_rate\_code, dim\_payment\_type). L'intégrité est garantie par des scripts SQL de création de contraintes et un mécanisme de "lookup" lors de l'ingestion Spark.

```

CREATE SCHEMA IF NOT EXISTS dwh;

-- Dimensions
CREATE TABLE IF NOT EXISTS dwh.dim_vendor (
    vendor_key    SERIAL PRIMARY KEY,
    vendor_id     INT UNIQUE NOT NULL
);

CREATE TABLE IF NOT EXISTS dwh.dim_rate_code (
    rate_code_key SERIAL PRIMARY KEY,
    rate_code_id  INT UNIQUE NOT NULL
);

CREATE TABLE IF NOT EXISTS dwh.dim_payment_type (
    payment_type_key SERIAL PRIMARY KEY,
    payment_type_id  INT UNIQUE NOT NULL
);

CREATE TABLE IF NOT EXISTS dwh.dim_location (
    location_key SERIAL PRIMARY KEY,
    location_id   INT UNIQUE NOT NULL
);

CREATE TABLE IF NOT EXISTS dwh.dim_datetime (
    datetime_key SERIAL PRIMARY KEY,
    ts           TIMESTAMP UNIQUE NOT NULL,
    date         DATE NOT NULL,
    year         INT NOT NULL,
    month        INT NOT NULL,
    day          INT NOT NULL,
    hour         INT NOT NULL,
    dow          INT NOT NULL
);

-- Fact table: 1 row = 1 trip
CREATE TABLE IF NOT EXISTS dwh.fact_trip (
    trip_key      SERIAL PRIMARY KEY,
    pickup_datetime_key INT NOT NULL,
    dropoff_datetime_key INT NOT NULL,
    vendor_key     INT,

```

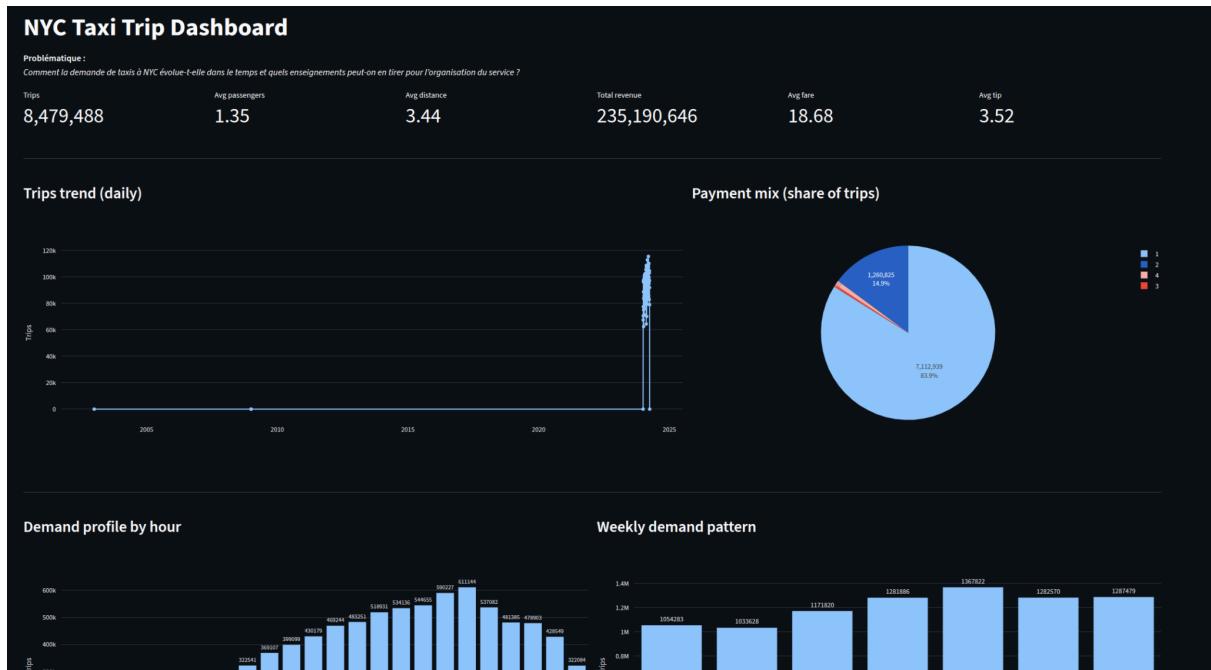
## 4. Visualisation de Données (Exercice 4)

Une fois les données structurées dans le Data Warehouse, nous avons développé des outils de restitution pour répondre aux besoins métiers.

**Analyse Exploratoire (EDA)** : Utilisation de dashboard.py pour visualiser les volumes de courses, les revenus totaux et les comportements de paiement des usagers.

**Indicateurs clés (KPIs)** : Affichage en temps réel de la distance moyenne des trajets et des zones de prise en charge les plus rentables.

**Accessibilité** : Le choix de Streamlit permet une interaction dynamique avec la base PostgreSQL, facilitant la lecture des données pour des utilisateurs non-techniques



## 5. Machine Learning Qualité Logicielle (Exercice 5)

### 5.1 Modélisation et Résultats

Pour prédire le total\_amount, nous avons utilisé l'algorithme HistGradientBoostingRegressor. Notre modèle affiche un RMSE de 4,25, soit une précision bien supérieure à l'objectif de 10 fixé par le cahier des charges. Les principaux intérêts de ce modèle sont la gestion native des catégories et la rapidité sur de gros volumes de données.

### 5.2 Industrialisation et Tests

Le développement respecte les standards de production:

**Qualité** : Audit via flake8 et documentation au format NumpyDoc.

**Tests** : Suite de tests unitaires validant l'intégrité des données à l'entrée de l'entraînement et lors de l'inférence.

```
ta_projet/ex05_ml_prediction_service$ python -m src.train
[EX5] RMSE: 4.2550
[EX5] Saved model -> artifacts/model.joblib
[EX5] Saved spec -> artifacts/feature_spec.json
```

```
(.venv) aminedesel@aminedesel-B550-AORUS-ELITE-AX-V2:~/Desktop/BigData_projet/ex05_ml_prediction_service$ flake8 --max-line-length=100 src/
(.venv) aminedesel@aminedesel-B550-AORUS-ELITE-AX-V2:~/Desktop/BigData_projet/ex05_ml_prediction_service$ export MINIO_BUCKET_CLEAN="nyc-cleaned"
export YEAR="2024"
export MONTH="01"
python -m pytest tests/test_pipeline.py
===== test session starts =====
platform linux -- Python 3.11.14, pytest-9.0.2, pluggy-1.6.0
rootdir: /home/aminedesel/Desktop/BigData_projet/ex05_ml_prediction_service
configfile: pytest.ini
collected 3 items

tests/test_pipeline.py ... [100%]

===== 3 passed in 32.63s =====
```

The screenshot shows a web-based application titled "Simulateur de tarifs Taxi New-Yorkais". The interface is dark-themed. At the top, there is a header with a car icon and the title. Below the header, a sub-header reads "Prédisez le coût total d'une course en fonction du trajet et de l'heure.". The main form is titled "Paramètres du trajet". It contains several input fields: "Où et Combien ?" (Quartier de départ: Auburndale, Quartier d'arrivée: Allerton/Pelham Gardens), "Quand ? (Par défaut : Maintenant)" (Date de la course: 2026/02/11, Heure de la course: 19:34), "Nombre de passagers" (3), and "Distance estimée (en miles)" (2.50). A note below the distance field states: "Astuce : La distance impacte le prix de base, mais l'heure et les quartiers déterminent les suppléments (heures de pointe, frais de congestion, etc.)". At the bottom of the form is a button labeled "Calculer le prix estimé". Below the form, the estimated total amount is displayed as "Estimation du Total Amount: 26.26 \$". A small note at the bottom says "Estimation basée sur les données historiques de 2024. Inclut les frais de base et taxes." On the right side of the screen, there is a "Deploy" button with a three-dot menu icon.

## 6.Orchestration (Bonus Exercice 6)

### 6.1 Automatisation via Apache Airflow

Pour transformer ces scripts en un pipeline industriel, nous avons déployé Apache Airflow. L'ensemble du workflow est géré par un DAG (Directed Acyclic Graph) :

**Séquençage** : La collecte MinIO déclenche le job Spark, qui lui-même précède l'ingestion SQL et l'entraînement ML.

**Robustesse** : Airflow gère les reprises sur erreur (retries) en cas de défaillance réseau lors du téléchargement des fichiers Parquet.

## 6.2 Monitoring "CAC 40"

L'interface d'Airflow permet un suivi en temps réel de l'état de santé du pipeline, offrant une visibilité complète sur les temps d'exécution et les éventuels goulots d'étranglement.

The screenshot shows the Airflow web interface at version v2.9.3. The top navigation bar includes links for Dags, Cluster Activity, Datasets, Security, Browse, Admin, and Docs. The main content area is titled 'DAGs' and displays a single DAG named 'ex6\_pipeline'. The DAG has 6 tasks: ex1\_data\_mirrivel, ex2\_q1\_data\_ingestion, ex3\_q1\_create\_tables, ex2\_q2\_data\_ingestion, ex3\_q2\_insert\_reference\_data, and ex4\_dashboard. All tasks are currently running, indicated by green circles. The 'Last Run' was on 2026-02-12, 22:59:50. Below the DAG list, a message indicates 'Showing 1-1 of 1 DAG'.

The screenshot shows the detailed view for the 'ex6\_pipeline' DAG. The top navigation bar is identical to the previous screenshot. The main content area is titled 'DAG: ex6\_pipeline'. It features a timeline chart showing task execution from 2026-02-12 00:00:00 to 2026-02-13 00:04:48. The chart uses color coding: red for failed tasks and green for successful ones. A legend at the bottom of the chart defines the colors for various states: deferred, failed, queued, removed, restarting, running, scheduled, shutdown, skipped, success, up\_for\_reschedule, up\_for\_retry, upstream\_failed, and no\_status. To the right of the chart, there is a 'DAG Run Summary' table with the following data:

Total Runs Displayed	17
Total success	9
Total failed	8
First Run Start	2026-02-12, 15:12:22 UTC
Last Run Start	2026-02-13, 20:29:50 UTC
Max Run Duration	00:07:37
Mean Run Duration	00:04:48
Min Run Duration	00:00:05
Total Tasks	7

## 7. Conclusion

Ce projet démontre la robustesse d'une stack Big Data moderne : Spark pour la puissance de calcul, MinIO/PostgreSQL pour le stockage hybride, et Airflow pour l'orchestration globale. Le respect des normes logicielles et l'atteinte d'un RMSE de 4,25 confirment la viabilité de la solution pour un déploiement en production. L'interface Streamlit finale transforme cette infrastructure technique en un véritable outil d'aide à la décision.