1- Problématique

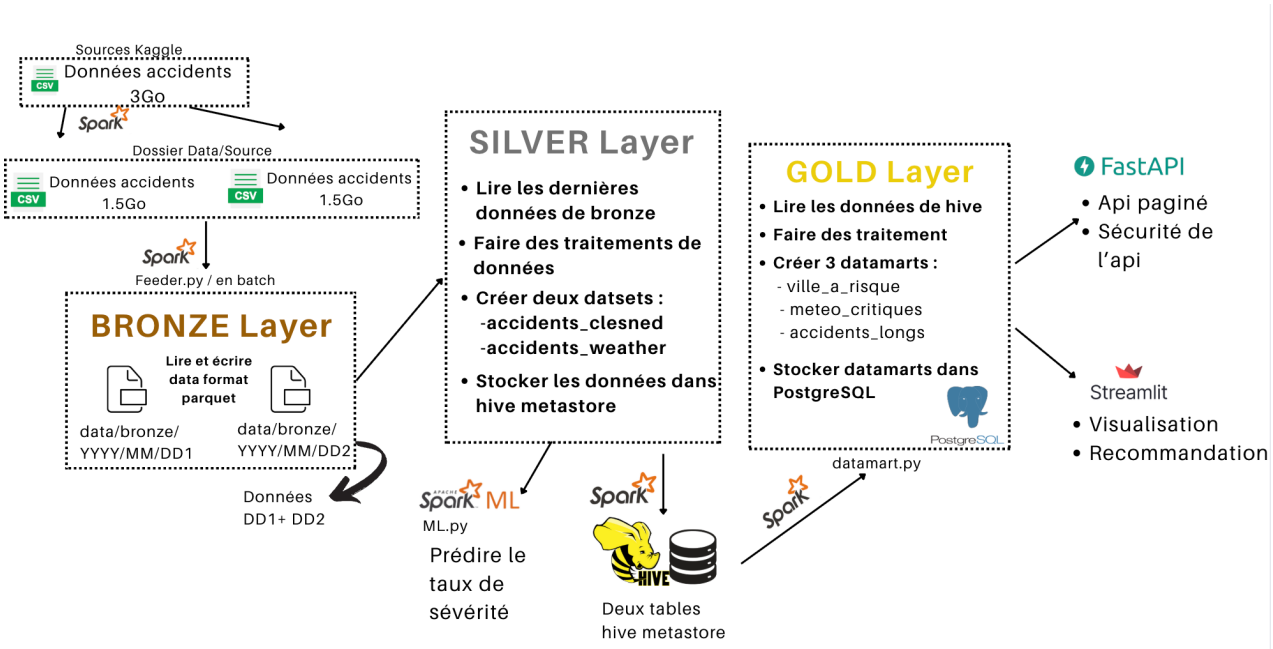
Les accidents de la route représentent un enjeu majeur pour la sécurité et la gestion des risques, nécessitant une exploitation efficace de données massives et hétérogènes. Dans ce contexte, la capacité à prédire la sévérité d'un accident à partir de données historiques (trafic, météo, localisation, etc.) offre des perspectives concrètes pour l'aide à la décision des assureurs, pouvoirs publics et services d'urgence.

2- Business Value

La prédiction de la sévérité des accidents de la route grâce à l'analyse de données massives (trafic, météo, géolocalisation) permet :

- Anticipation du risque : Identifier les zones et conditions météo les plus à risque pour orienter la prévention.
- Aide à la décision : Donner aux assureurs, pouvoirs publics et services d'urgence des informations exploitables en temps réel pour mieux cibler leurs actions.
- Optimisation des ressources : Prioriser l'allocation des secours, adapter la gestion du trafic et optimiser les interventions.
- Personnalisation des offres d'assurance : Ajuster les polices et tarifications selon le niveau de risque géolocalisé.
- Réduction des coûts : En diminuant la sinistralité et en intervenant plus efficacement, on limite les conséquences humaines et matérielles.

3- Architecture



4- Dossiers principaux

data/

- **bronze/** : fichiers Parquet partitionnés (données brutes, formaté par date).

- **source/** : fichiers CSV sources à ingérer.
- **seen_files.txt** : registre des fichiers déjà traités pour l'ingestion incrémentale.
- **(Autres fichiers)** : fichiers de configuration pour Hive Metastore et docker

notebooks/

- **logs/** : tous les fichiers de log générés par les scripts.
- **UTILS.py** : fonctions réutilisables (création SparkSession, gestion des fichiers lus, détection nouveaux fichiers, fusion...).
- **split_and_tag.py** : script pour diviser le gros fichier Kaggle en deux sous-fichiers CSV.
- **Feeder.py :**

    - Gère l'ingestion incrémentale des fichiers bruts depuis /data/source/ vers la couche Bronze.
    - Détecte automatiquement les nouveaux fichiers grâce à un registre seen_files.txt
    - Extraire la date depuis le nom du fichier et organise des données dans des dossiers partitionnés : data/bronze/YYYY/MM/DD/
    - Fusionne les nouvelles données avec les anciennes pour éviter toute perte.
    - Stocke les données au format Parquet
    - Chaque partition journalière est écrite avec **2 partitions Spark**
    - Journalisation détaillée dans un fichier log pour assurer le suivi et le débogage.

- **Preprocessor.py :**

    - traite le dernier dossier de la couche Bronze (/data/bronze/YYYY/MM/DD/), pour garantir que seul le batch de données le plus récent est pris en compte.
    - Les données sont lues au format Parquet, ce qui accélère le chargement et optimise le stockage.
    - Traitement automatique des données : conversion des timestamps, calcul de la durée d'accident, typage et gestion des colonnes numériques/catégorielles (remplacement des valeurs manquantes, cast, ajout de colonnes utiles)
    - Le DataFrame Spark est **persisté** en mémoire/disque (df.persist(StorageLevel.MEMORY_AND_DISK)) pour améliorer la performance lors des traitements.
    - Les données sont **repartitionnées en 2**
    - Deux datasets créés : un pour les accidents, un pour la météo, tous deux sauvegardés dans Hive (Silver Layer) sous forme de tables Parquet
    - Logging détaillé pour chaque étape (début, fin, succès Hive...)

- **ML.py :**

- Lit les jeux de données accidents et météo depuis Hive (Silver), puis les fusionne via la colonne ID
- Sélectionne et prépare les features numériques utiles à la prédiction de la sévérité.
- Supprime les lignes incomplètes pour garantir la qualité des données.
- Split les données en **train/test** (80/20).
- Crée un pipeline Spark ML :
    - **Assembler** des features,
    - **StandardScaler** (mise à l'échelle),
    - **RandomForestClassifier** (forêt aléatoire, 50 arbres).
- Entraîne le modèle sur le train et prédit sur le test.
- Évalue les performances avec 4 métriques principales (accuracy, precision, recall, f1).
- Loggue tout le processus

- **datamart.py :**

- Charge les données nettoyées (accidents + météo) depuis Hive et effectue une jointure sur la colonne ID.
- Crée 3 datamarts analytiques :
    - **villes_a_risque** : top villes avec le plus d'accidents graves (≥3), statistiques agrégées.
    - **meteo_critiques** : conditions météo associées à la gravité, fréquence, humidité, vent, précipitations.
    - **accidents_longs** : top 100 accidents de plus longue durée, avec infos météo associées.
- Les résultats sont écrits directement dans PostgreSQL (schéma datamarts) via le connecteur JDBC, en mode overwrite.
- Journalisation complète pour assurer la traçabilité et le suivi d'exécution.

- **Api.py :**

- Expose une API REST (FastAPI) pour accéder aux datamarts stockés dans PostgreSQL, avec authentification par jeton (API_TOKEN) pour sécuriser les endpoints.
- lire les tables postgresql via JDBC
- **Principales routes :**
    - `/health` : test simple pour vérifier si l'API fonctionne.
    - `/datamarts/{table_name}` : pagination sur une des 3 tables (`accidents_longs`, `meteo_critiques`, `villes_a_risque`), résultat en liste de dictionnaires.
    - `/datamarts/{table_name}/schema` : retourne le schéma de la table (nom/type de chaque colonne).
    - `/datamarts/{table_name}/count` : retourne le nombre de lignes total d'une table.
    - `/datamarts/{table_name}/search` : permet de chercher une valeur dans une colonne (filtre LIKE, pagination incluse).

- **Sécurité** : Chaque endpoint sensible requiert un token d'authentification HTTP Bearer.
- **Architecture réutilisable** : chaque opération de lecture est factorisée dans des fonctions, la gestion du SparkSession et la configuration sont mutualisées avec UTILS.py.

- **streamlit.py :**

- Dashboard développé avec Streamlit, connecté directement à la base PostgreSQL des datamarts.
- Top 10 villes à risque (accidents graves)
- Durée moyenne d'accident (top 10 villes)
- Top 10 conditions météo les plus dangereuses (par sévérité moyenne)
- Top 5 conditions météo les plus accidentogènes
- Distribution de la sévérité des accidents

5- Justificatif des paramètres dans spark-submit

Pour bronze on a mis :

```
spark-submit \
  --master yarn \
  --deploy-mode client \
  --conf spark.sql.shuffle.partitions=2 \
  --conf spark.driver.memory=2g \
  --conf spark.executor.memory=1g \
  --conf spark.executor.cores=2\
 datamart.py
```

-2 partitions chacune environ 460Mo => On a choisi **1Go** par exécuteur pour avoir une marge confortable et éviter les erreurs mémoire

- on a mis **2 cores** par exécuteur pour permettre le traitement parallèle des deux partitions

- conf spark.sql.shuffle.partitions=2 identique à ce que j'ai mis lors de l'écriture des fichiers (df ;repartition(2)

Pour Silver et Datamart_postgres :

-On a mis **2 cores** et 2**Go** de mémoire par executor parce qu'on a des jointures et des opérations de traitements de données comme la conversation de type qui consomme plus de RAM qu'une simple ingestion brute

Pour ML_severity :

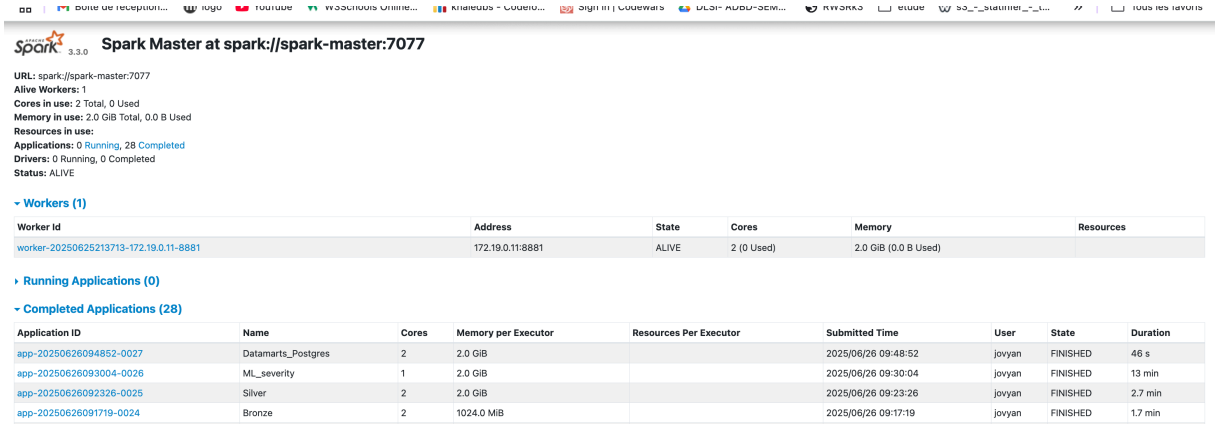
Pour l'entraînement du modèle, on a mis **1 seul core** par exécuteur, car les algorithmes de Machine Learning Spark, comme RandomForest, ne tirent pas toujours parti du parallélisme sur plusieurs cores lorsque la taille de données est modérée

-2 Go de mémoire à l'executor pour garantir que la phase d'entraînement du modèle (Random Forest) puisse charger en mémoire un volume suffisant de données, ainsi que tous les objets Spark nécessaires (features, labels, pipeline, modèle, etc).

6- Arborescence dossier après le feeder

data	aujourd'hui à 12:23	--
bronze	avant-hier à 10:44	--
2025	avant-hier à 10:44	--
01	aujourd'hui à 12:30	--
01	avant-hier à 10:42	--
_SUCCESS	avant-hier à 10:42	Zéro octet
part-00000-c39c184b-d37a-...185a10c1-c000.snappy.parquet	avant-hier à 10:42	218,1 Mo
part-00001-c39c184b-d37a-...185a10c1-c000.snappy.parquet	avant-hier à 10:42	218,3 Mo
02	aujourd'hui à 12:31	--
_SUCCESS	aujourd'hui à 12:31	Zéro octet
part-00000-b5493f3d-60be-...afcda5fc-c000.snappy.parquet	aujourd'hui à 12:31	464,1 Mo
part-00001-b5493f3d-60be-...afcda5fc-c000.snappy.parquet	aujourd'hui à 12:31	464,1 Mo

7- Spark UI



Spark Master at spark://spark-master:7077

URL: spark://spark-master:7077
Alive Workers: 1
Cores in use: 2 Total, 0 Used
Memory in use: 2.0 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 28 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (1)

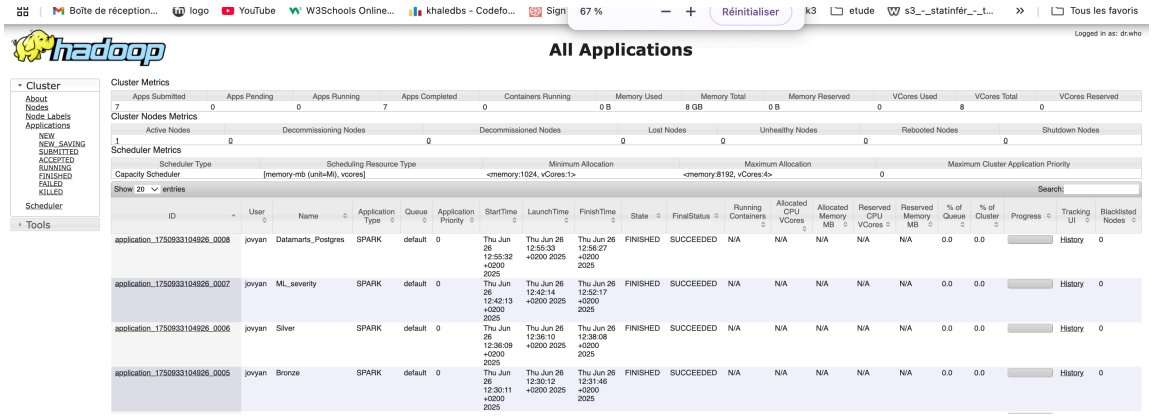
Worker Id	Address	State	Cores	Memory	Resources
worker-20250625213713-172.19.0.11-8881	172.19.0.11:8881	ALIVE	2 (0 Used)	2.0 GiB (0.0 B Used)	

Running Applications (0)

Completed Applications (28)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20250626094852-0027	Datamarts_Postgres	2	2.0 GiB		2025/06/26 09:48:52	jovyan	FINISHED	46 s
app-20250626093004-0026	ML_severity	1	2.0 GiB		2025/06/26 09:30:04	jovyan	FINISHED	13 min
app-20250626092326-0025	Silver	2	2.0 GiB		2025/06/26 09:23:26	jovyan	FINISHED	2.7 min
app-20250626091719-0024	Bronze	2	1024.0 MiB		2025/06/26 09:17:19	jovyan	FINISHED	1.7 min

8- Yarn UI



All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
7	0	0	7	0	0 B	8 GB	0 B	0	8	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
1	0	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Reserved CPU VCores	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
application_1750933104926_0008	jovyan	Datamarts_Postgres	SPARK	default	0	Thu Jun 26 12:55:32 +0200 2025	Thu Jun 26 12:55:33 +0200 2025	Thu Jun 26 12:56:27 +0200 2025	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1750933104926_0007	jovyan	ML_severity	SPARK	default	0	Thu Jun 26 12:42:13 +0200 2025	Thu Jun 26 12:42:14 +0200 2025	Thu Jun 26 12:52:17 +0200 2025	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1750933104926_0006	jovyan	Silver	SPARK	default	0	Thu Jun 26 12:36:09 +0200 2025	Thu Jun 26 12:36:10 +0200 2025	Thu Jun 26 12:38:08 +0200 2025	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1750933104926_0005	jovyan	Bronze	SPARK	default	0	Thu Jun 26 12:30:11 +0200 2025	Thu Jun 26 12:30:12 +0200 2025	Thu Jun 26 12:31:46 +0200 2025	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0

9- Hive (entrepôt de données)

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.3.0
      /_/

Using Python version 3.10.6 (main, Aug 22 2022 20:27:34)
Spark context Web UI available at http://c7d69d253ad8:4040
Spark context available as 'sc' (master = local[*], app id = local-1750764166838).
SparkSession available as 'spark'.
>>> from pyspark.sql import SparkSession
>>> spark = (
...     SparkSession.builder
...     .appName("CheckMetastore")
...     .master("spark://spark-master:7077")
...     .enableHiveSupport()
...     .getOrCreate()
... )
"SHOW DATABASES").show()
25/06/24 11:22:57 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
>>> spark.sql("SHOW DATABASES").show()
+---------+
|namespace|
+---------+
|  default|
|   silver|
+---------+
```

```
SyntaxError: invalid syntax
>>> spark.sql("USE silver")
DataFrame[]
>>>
>>> spark.sql("SHOW TABLES").show()
+---------+-----------------+-----------+
|namespace|        tableName|isTemporary|
+---------+-----------------+-----------+
|   silver|accidents_cleaned|      false|
|   silver|accidents_weather|      false|
+---------+-----------------+-----------+

>>>
>>> spark.sql("select * from accidents_cleaned").show(5)
+---------+-------+--------+-------------------+-------------------+---------+-------------------+-------+-------+-----------+-----------+---------
---+-----+-------+-----------+----------------+
|       ID| Source|Severity|         Start_Time|           End_Time|Start_Lat|          Start_Lng|End_Lat|End_Lng|Distance_mi|       City|      Cou
nty|State|Zipcode|is_accident|Duration_Minutes|
+---------+-------+--------+-------------------+-------------------+---------+-------------------+-------+-------+-----------+-----------+---------
---+-----+-------+-----------+----------------+
| A-667842|Source2|       4|2022-03-22 22:02:33|2022-03-22 22:46:23|39.628876|        -106.066376|   null|   null|        0.0|     Dillon|      Sum
mit|   CO|  80435|          1|            44.0|
|A-1470683|Source2|       3|2020-05-15 12:56:46|2020-05-15 15:50:08|28.036955|         -82.104012|   null|   null|        0.0| Plant City|Hillsboro
ugh|   FL|  33563|          1|           173.0|
| A-309318|Source2|       3|2016-11-03 18:00:19|2016-11-03 18:45:01|47.626503|-122.32861299999999|   null|   null|       0.01|    Seattle|        K
ing|   WA|  98102|          1|            45.0|
|A-1306350|Source2|       2|2020-09-20 08:48:18|2020-09-20 09:52:56|37.961433|        -121.245865|   null|   null|        0.0|   Stockton| San Joaq
uin|   CA|  95205|          1|            65.0|
|A-1338245|Source2|       3|2020-10-13 14:30:00|2020-10-13 15:14:42|39.072834|         -94.531265|   null|   null|        0.0|Kansas City|     Jack
son|   MO|  64128|          1|            45.0|
+---------+-------+--------+-------------------+-------------------+---------+-------------------+-------+-------+-----------+-----------+---------
---+-----+-------+-----------+----------------+
only showing top 5 rows
```

```
>>> spark.sql("select * from accidents_weather").show(5)
+---------+-------------------+-------------+-----------+------------+-----------+-------------+--------------+--------------+----------------+---
--------------+
|       ID|  Weather_Timestamp|Temperature_F|Wind_Chill_F|Humidity_pct|Pressure_in|Visibility_mi|Wind_Direction|Wind_Speed_mph|Precipitation_in|Wea
ther_Condition|
+---------+-------------------+-------------+-----------+------------+-----------+-------------+--------------+--------------+----------------+---
--------------+
| A-667842|2022-03-22 21:53:00|         13.0|        0.0|        81.0|      20.82|         10.0|           NNE|          10.0|            0.01|
 Mostly Cloudy|
|A-1470683|2020-05-15 12:50:00|         86.0|       86.0|        51.0|      29.76|         10.0|             E|          16.0|             0.0|
 Mostly Cloudy|
| A-309318|2016-11-03 17:53:00|         60.1|        0.0|        72.0|      30.17|         10.0|          Calm|           0.0|             0.0|
         Clear|
|A-1306350|2020-09-20 08:55:00|         69.0|       69.0|        65.0|      29.86|          8.0|            SW|           3.0|             0.0|
          Fair|
|A-1338245|2020-10-13 14:54:00|         75.0|       75.0|        23.0|      29.27|         10.0|            NE|           5.0|             0.0|
          Fair|
+---------+-------------------+-------------+-----------+------------+-----------+-------------+--------------+--------------+----------------+---
--------------+
only showing top 5 rows
```

10- PostgreSQL (datamarts)

```
PROBLEMS 2    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

(base) hadilneji@Host-003 Bigdata 3 % psql -h localhost -U spark -d datamarts -p 5433

Password for user spark:
psql (17.5, server 15.13 (Debian 15.13-1.pgdg120+1))
Type "help" for help.

datamarts=# SELECT * FROM villes_a_risque LIMIT 3;
    City     | State | Nombre_Accidents_Graves |  Severite_Moyenne  | Duree_Moyenne_Accident
-------------+-------+-------------------------+--------------------+------------------------
 Los Angeles | CA    |                   35866 |  3.026180784029443 |      68.40868789382702
 Dallas      | TX    |                   34142 |  3.033214222951204 |      95.63894909495636
 Atlanta     | GA    |                   33842 | 3.0837420956208264 |      71.68657289758289
(3 rows)

Cancel request sent
datamarts=# datamarts=# SELECT * FROM meteo_critiques  LIMIT 3;
   Weather_Condition    |    Moy_Severity    | Nb_Accidents |    Humidite_Moy    |  Vitesse_Vent_Moy  | Precipitations_Moy
------------------------+--------------------+--------------+--------------------+--------------------+-------------------
 Light Blowing Snow     | 3.6666666666666665 |            3 | 58.666666666666664 | 15.333333333333334 |                  0
 Patches of Fog / Windy |  3.142857142857143 |            7 |  86.14285714285714 |                 24 |                  0
 Partial Fog / Windy    |                  3 |            1 |                 81 |                 24 |                  0
(3 rows)

datamarts=# SELECT * FROM accidents_longs  LIMIT 3;
    ID     |    City     | State |     Start_Time      | Duration_Minutes | Severity | Weather_Condition | Wind_Speed_mph | Precipitation_in
-----------+-------------+-------+---------------------+------------------+----------+-------------------+----------------+-----------------
 A-4810425 | Wilmington  | DE    | 2016-10-21 07:26:00 |          2812939 |        2 | Overcast          |            8.1 |                0
 A-5053641 | Wilmington  | DE    | 2016-10-21 07:26:00 |          2812939 |        2 | Overcast          |            8.1 |                0
 A-5399002 | Southampton | NY    | 2018-04-19 09:24:00 |          2236406 |        2 | Overcast          |            5.8 |                0
(3 rows)

datamarts=#
```

Loggers

Feeder log

```
2025-06-26 10:21:17,564 [INFO] Closing down clientserver connection
2025-06-26 10:30:06,703 [INFO] Logs seront écrits dans /home/jovyan/work/logs/feeder.log
2025-06-26 10:30:28,215 [INFO] ➡️ Écriture dans /data/bronze/2025/01/02
2025-06-26 10:31:46,311 [INFO] Fichier traité : 02-01-2025.csv | Temps d'écriture : 83.32 s
2025-06-26 10:31:46,646 [INFO] ⏱️ Temps total d'exécution du pipeline : 99.94 sec
2025-06-26 10:31:46,650 [INFO] Closing down clientserver connection
```

Preprocessor log

```
2025-06-26 10:36:05,413 [INFO] Logs seront écrits dans /home/jovyan/work/logs/preprocessor.log
2025-06-26 10:36:05,415 [INFO] Dossier parquet le plus récent : /data/bronze/2025/01/02
2025-06-26 10:36:23,191 [INFO] ✅ DataFrame persisted en MEMORY_AND_DISK
2025-06-26 10:37:47,662 [INFO] Nombre de lignes dans le batch : 7728394
2025-06-26 10:38:02,418 [INFO] ✅ Table accidents sauvegardée dans Hive : silver.accidents_cleaned
2025-06-26 10:38:08,706 [INFO] ✅ Table météo sauvegardée dans Hive : silver.accidents_weather
2025-06-26 10:38:08,707 [INFO] ⏱️ Temps de traitement de la dernière partition : 108.17 sec
2025-06-26 10:38:09,235 [INFO] ⏱️ Temps total d'exécution du preprocessor : 123.82 sec
2025-06-26 10:38:09,236 [INFO] Closing down clientserver connection
```

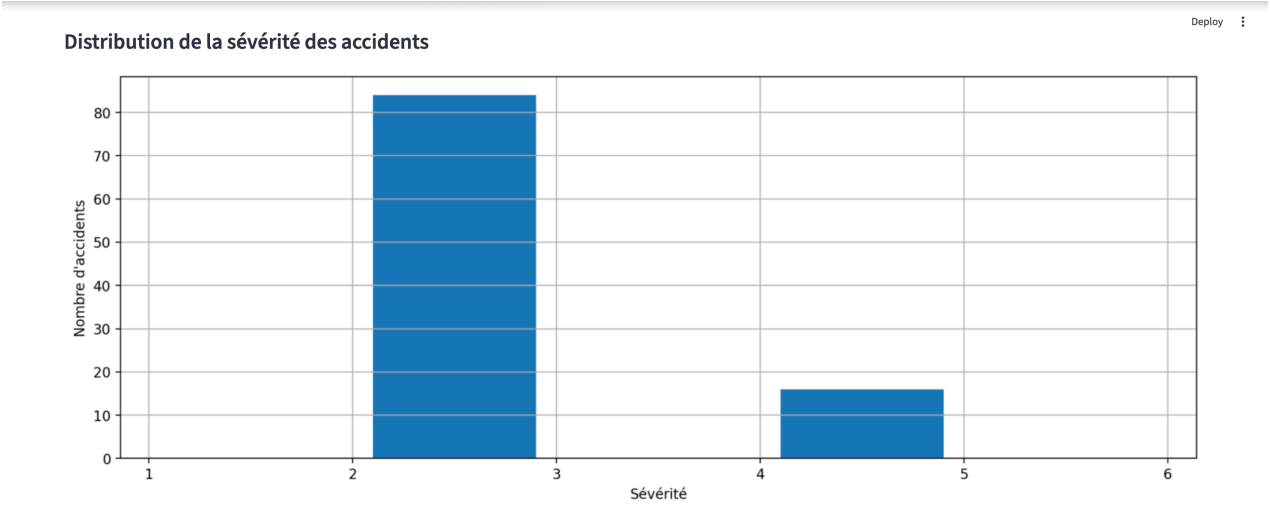
Datamarts log

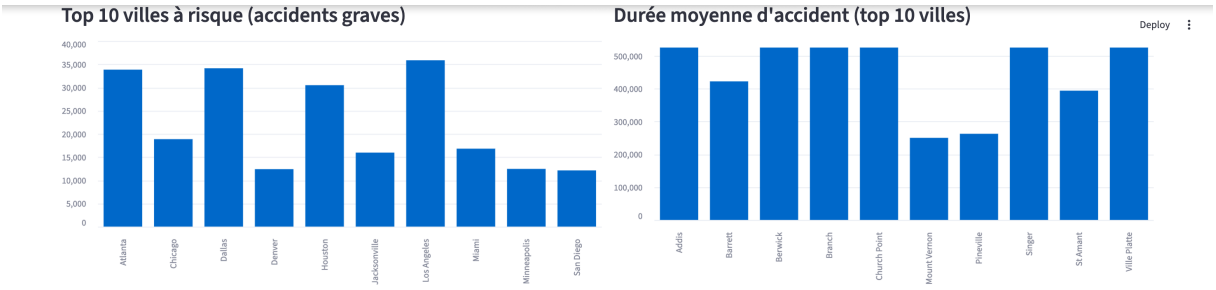
```
2025-06-26 10:55:28,092 [INFO] Logs seront écrits dans /home/jovyan/work/logs/datamarts.log
2025-06-26 10:55:28,093 [INFO] Les datamarts seront stockés dans la base Postgres : jdbc:postgresql://postgres:5432/datamarts
2025-06-26 10:55:45,097 [INFO] Chargement des données depuis Hive...
2025-06-26 10:55:46,556 [INFO] Jointure accidents + météo OK.
2025-06-26 10:55:46,598 [INFO] Datamart 1 'villes_a_risque' prêt.
2025-06-26 10:55:46,614 [INFO] Datamart 2 'meteo_critiques' prêt.
2025-06-26 10:55:46,624 [INFO] Datamart 3 'accidents_longs' prêt.
2025-06-26 10:55:46,625 [INFO] ➡️ Écriture du datamart 'villes_a_risque' dans jdbc:postgresql://postgres:5432/datamarts ...
2025-06-26 10:56:04,582 [INFO] ✅ Datamart 'villes_a_risque' écrit avec succès.
2025-06-26 10:56:04,585 [INFO] ➡️ Écriture du datamart 'meteo_critiques' dans jdbc:postgresql://postgres:5432/datamarts ...
2025-06-26 10:56:14,853 [INFO] ✅ Datamart 'meteo_critiques' écrit avec succès.
2025-06-26 10:56:14,854 [INFO] ➡️ Écriture du datamart 'accidents_longs' dans jdbc:postgresql://postgres:5432/datamarts ...
2025-06-26 10:56:26,750 [INFO] ✅ Datamart 'accidents_longs' écrit avec succès.
2025-06-26 10:56:27,123 [INFO] ⏱️ Temps total d'exécution du pipeline datamarts : 59.03 sec
2025-06-26 10:56:27,125 [INFO] Closing down clientserver connection
```

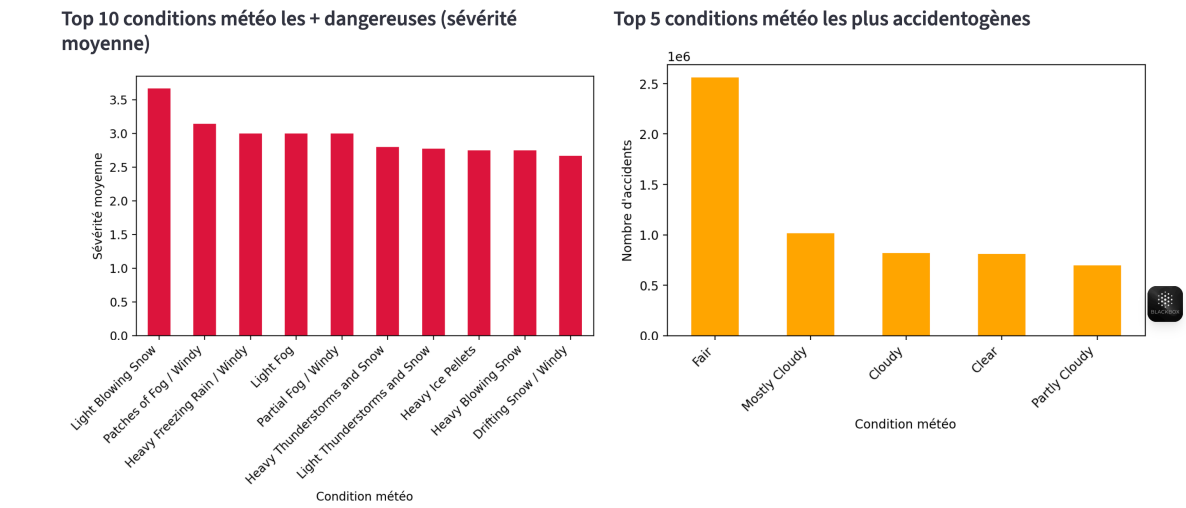
ML log

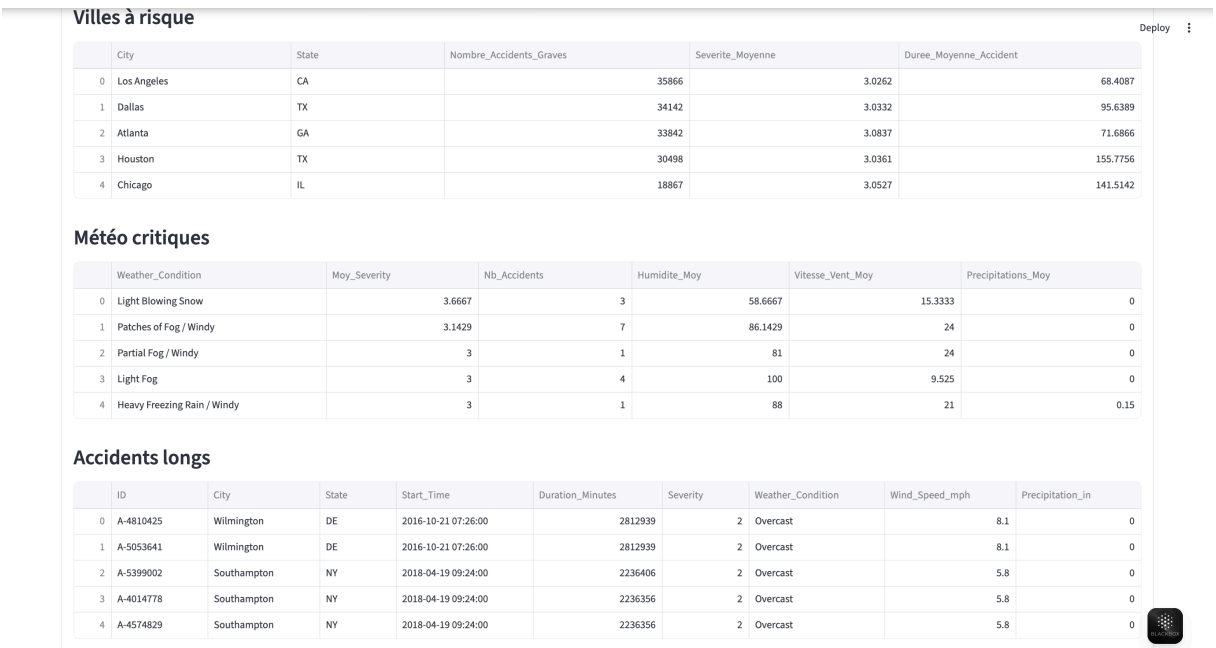
```
2025-06-26 10:49:34,766 [INFO] Accuracy: 0.797
2025-06-26 10:50:23,266 [INFO] Weightedprecision: 0.634
2025-06-26 10:51:07,781 [INFO] Weightedrecall: 0.797
2025-06-26 10:51:51,179 [INFO] F1: 0.706
2025-06-26 10:52:17,071 [INFO] Temps d'exécution total (sec): 607.68
2025-06-26 10:52:17,080 [INFO] Closing down clientserver connection
```

11-Visualisation









	City	State	Nombre_Accidents_Graves	Severite_Moyenne	Duree_Moyenne_Accident
0	Los Angeles	CA	35866	3.0262	68.4087
1	Dallas	TX	34142	3.0332	95.6389
2	Atlanta	GA	33842	3.0837	71.6866
3	Houston	TX	30498	3.0361	155.7756
4	Chicago	IL	18867	3.0527	141.5142

Météo critiques

	Weather_Condition	Moy_Severity	Nb_Accidents	Humidite_Moy	Vitesse_Vent_Moy	Precipitations_Moy
0	Light Blowing Snow	3.6667	3	58.6667	15.3333	0
1	Patches of Fog / Windy	3.1429	7	86.1429	24	0
2	Partial Fog / Windy	3	1	81	24	0
3	Light Fog	3	4	100	9.525	0
4	Heavy Freezing Rain / Windy	3	1	88	21	0.15

Accidents longs

	ID	City	State	Start_Time	Duration_Minutes	Severity	Weather_Condition	Wind_Speed_mph	Precipitation_in
0	A-4810425	Wilmington	DE	2016-10-21 07:26:00	2812939	2	Overcast	8.1	0
1	A-5053641	Wilmington	DE	2016-10-21 07:26:00	2812939	2	Overcast	8.1	0
2	A-5399002	Southampton	NY	2018-04-19 09:24:00	2236406	2	Overcast	5.8	0
3	A-4014778	Southampton	NY	2018-04-19 09:24:00	2236356	2	Overcast	5.8	0
4	A-4574829	Southampton	NY	2018-04-19 09:24:00	2236356	2	Overcast	5.8	0

12- Recommandation

- Les dashboards montrent que certaines villes enregistrent un nombre élevé d'accidents graves et une durée moyenne d'accident importante.
  Recommandation : Lancer des campagnes de sensibilisation ciblées et

améliorer la signalisation dans ces villes, surtout aux périodes où la sinistralité est la plus forte.

- Les graphiques mettent en évidence les conditions météo les plus dangereuses et accidentogènes (comme le brouillard, la neige ou le vent fort).
  Recommandation : Alerter les usagers et adapter la gestion du trafic lors de ces conditions : messages d'alerte, limitation temporaire de vitesse, déploiement de patrouilles sur les axes concernés.