



DataScientest • com

Supply Chain - Satisfaction des clients

Rapport intermédiaire de l'étape 4

Promotion Data Engineer - Mars 2025



Didier Joséphine
04/09/2025

TABLE DES MATIERES

| | |
|---|----|
| 1. Introduction | 2 |
| 1.1 Contexte du projet | 2 |
| 1.2 Périmètre du projet : analyse de la satisfaction client chez Tesla, Temu, Chronopost, Vinterd | 4 |
| 1.3 Architecture fonctionnelle du projet : Web Scraping → BDD → ML Models → Visualisation ... | 5 |
| 1.4 Reproductibilité et Documentation du code | 7 |
| 1.5 Composants techniques du projet..... | 10 |
| 2. Collecte des Données (Partie 1) | 12 |
| 2.1 Architecture fonctionnelle de collecte des données : Web Scraping | 12 |
| 2.2 Collecte automatisée d'avis clients via Web Scraping (Python/Trustpilot)..... | 13 |
| 2.3 Collecte mensuelle des données sociétales via API (WIKIPEDIA)..... | 16 |
| 3. Stockage et Modélisation des Données (Partie 2) | 20 |
| 3.1 Architecture fonctionnelle : BDD | 20 |
| 3.2 PostgreSQL..... | 22 |
| 3.3 MongoDB | 26 |
| 4. Traitement des Données et Machine Learning (Partie 3)..... | 29 |
| 4.1 Architecture fonctionnelle - Pipeline ML: ML Models → Visualisation | 29 |
| 4.2 Extraction des avis Trustpilot | 32 |
| 4.3 Inférence statistique et modélisation prédictive | 33 |
| 4.4 Nettoyage et préparation des données..... | 36 |
| 4.5 Analyse NLP avancée | 39 |
| 4.6 Benchmarking des modèles de classification : analyse comparative..... | 45 |
| 4.7 Vers un modèle de régression : justification et sélection..... | 54 |
| 4.8 MLflow : suivi des modèles | 57 |
| 5. Développement de l'API avec Fast API (Partie 4) | 64 |
| 6. Résultats & Synthèse Métier | 65 |
| 6.1 Résultats Tesla..... | 66 |
| 6.2 Résultats Chronopost..... | 66 |
| 6.3 Résultats Vinterd | 66 |
| 6.4 Résultats Temu | 66 |
| 7. Résultats & Synthèse Projet..... | 67 |
| 7.1 Résultats clés : synthèse des parties 1 à 4..... | 67 |
| 7.2 Limites et considérations éthiques | 69 |
| 8. Conclusions & perspectives | 71 |

1. INTRODUCTION

1.1 CONTEXTE DU PROJET

Dans le cadre de ma formation chez **DataScientest**, j'ai mené un projet complet portant sur la collecte, le stockage et l'analyse d'avis clients issus de Trustpilot.

Ce projet s'inscrit dans une démarche globale visant à **évaluer la satisfaction client** et à **assurer une veille concurrentielle** sur plusieurs entreprises (Tesla, Chronopost, Temu, Vinted).

Un des défis majeurs a été de **collecter les données (avis clients en français) tout en respectant les restrictions des fichiers robots.txt**. Pour y répondre, j'ai mis en place une stratégie de scraping "historique".

☞ Chaque jour, le script de collecte reprend automatiquement l'extraction **là où elle s'était arrêtée**, en enregistrant les résultats dans des répertoires distincts, horodatés par société (**étape 1**).

📦 À chaque itération, l'ensemble des dossiers collectés est ensuite **consolidé** pour alimenter une base **MongoDB** avec les commentaires, garantissant une structuration NoSQL des données.

📦 **Une fois par mois**, dans le cadre de l'analyse approfondie des entreprises cibles (*Tesla, Temu, Chronopost, Vinted*), j'enrichis le projet par une **collecte complémentaire d'informations sociétales** via l'**API Wikipedia**. Ces données sont intégrées dans la base **PostgreSQL**.

☞ Une architecture de base de données double a été conçue comprenant (**étape 2**) :

📦 Un modèle UML pour PostgreSQL avec des tables normalisées permettant de structurer efficacement les métadonnées des entreprises

📦 Une base MongoDB optimisée pour le stockage documentaire des avis clients

☞ Le modèle BERT pré-entraîné '**nlpTown/bert-base-multilingual-uncased-sentiment**' pour générer deux cibles : le **sentiment des commentaires** (négatif/neutre/positif) et leur **note prédite** (1 à 5 étoiles) (**étape 3**).

📦 L'ensemble du pipeline inclut une **analyse qualité des données**, un **prétraitement linguistique** et l'**entraînement de modèles de classification supervisés** avec une évaluation des performances.

📦 MLflow permet de versionner et orchestrer l'**ensemble du pipeline de Machine Learning**, incluant l'analyse de la qualité des données, le prétraitement (linguistique si applicable), l'entraînement de modèles de classification supervisés et le **suivi des métriques pour garantir la robustesse et la reproductibilité**.

☒ Développer une API sécurisée par authentification, livrée dans un conteneur Docker (étape 4) :

- 📦 Développer une API FastAPI sécurisée avec des endpoints
- 📦 Conteneuriser l'application avec Docker pour assurer son déploiement et son isolation.
- 📦 Implémenter des tests unitaires avec Pytest
- 📦 Sécuriser l'API via un mécanisme d'authentification

1.2 PERIMETRE DU PROJET : ANALYSE DE LA SATISFACTION CLIENT CHEZ TESLA, TEMU, CHRONOPOST, VINTED

Cette analyse s'est focalisée sur quatre acteurs majeurs digitaux et hybrides :

1. **Tesla** : Leader de l'industrie automobile électrique et des technologies énergétiques
 - Secteur : Automobile high-tech/énergie verte
 - Enjeux analysés : Qualité produit, innovation technologique, service après-vente
 - Particularité : Clientèle tech-savvy avec des attentes élevées
2. **Temu** : Plateforme e-commerce émergente spécialisée dans les prix ultra-compétitifs
 - Secteur : E-commerce grand public
 - Enjeux analysés : Rapport qualité-prix, délais de livraison, satisfaction client
 - Particularité : Modèle disruptif de vente directe importée
3. **Chronopost** : Acteur historique de la livraison express en France
 - Secteur : Logistique et transport
 - Enjeux analysés : Ponctualité des livraisons, traitement des réclamations
 - Particularité : Interface B2B et B2C
4. **Vinted** : Marketplace leader de la mode circulaire
 - Secteur : E-commerce et économie collaborative
 - Enjeux analysés : Expérience utilisateur, confiance entre particuliers
 - Particularité : Modèle peer-to-peer avec système de notation communautaire

Méthodologie de sélection :

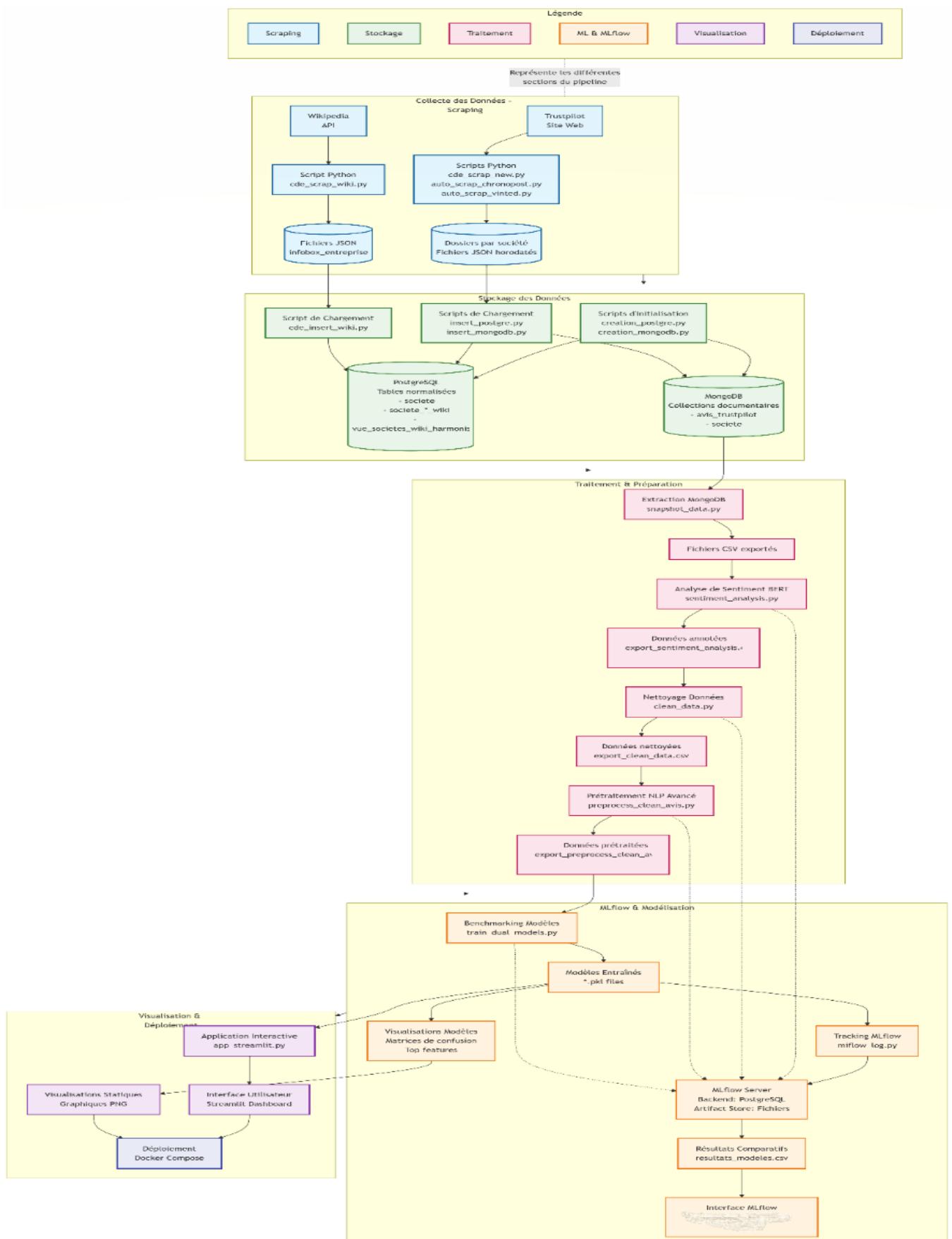
- Critères retenus : Notoriété, diversité sectorielle
- Période couverte : Avis collectés entre janvier 2023 et Aout 2025
- Volume analysé Aout 2025 : 66 864 avis (Tesla : 217, Temu : 3 545, Chronopost : 32 220, Vinted : 30 882)

Objectifs de cette sélection :

- Permettre une analyse comparative intersectorielle
- Identifier des patterns communs et spécifiques à chaque modèle économique
- Comprendre l'impact des particularités sectorielles sur l'expérience client

Cette **approche multicritère offre une vision panoramique des attentes clients contemporaines** tout en permettant des analyses fines par secteur d'activité. La sélection équilibrée entre **pure players digitaux** (Temu, Vinted) et **entreprises hybrides** (Tesla, Chronopost) reflète la diversité des canaux d'expérience client modernes.

1.3 ARCHITECTURE FONCTIONNELLE DU PROJET : WEB SCRAPING → BDD → ML MODELS → VISUALISATION



Ce schéma illustre les étapes clés du projet, de la collecte des données (scraping) au stockage (PostgreSQL/MongoDB) et à l'analyse (modèles ML).

Voici les principales sections :

1. **Collecte des Données (Bleu)** : [Processus de scraping depuis Trustpilot et Wikipédia](#)
2. **Stockage (Vert)** : [Bases de données PostgreSQL et MongoDB avec leurs scripts d'initialisation](#)
3. **Traitement (Rose)** : [Pipeline de nettoyage, analyse de sentiment et prétraitement NLP](#)
4. **MLflow & Modélisation (Orange)** : [Intégration centrale de MLflow pour le tracking des expériences, modèles et artefacts](#)
5. **Visualisation & Déploiement (Violet)** : [Interface Streamlit et déploiement Docker](#)

1.4 REPRODUCTIBILITE ET DOCUMENTATION DU CODE

1. Repository GitHub

L'ensemble du code source, des scripts de configuration et de déploiement associés à ce projet est versionné et accessible publiquement sur la plateforme GitHub.

Le dépôt est disponible sur [GitHub](#) :

2. Lancement du traitement end-to-end

Scripts manuels

```
cd scripts

# Scraping
python scraping/cde_scrap_new.py

# Création des bases
python scraping/creation_postgre.py
python scraping/creation_mongodb.py

# Insertion en base
python scraping/insert_postgre.py
python scraping/insert_mongodb.py

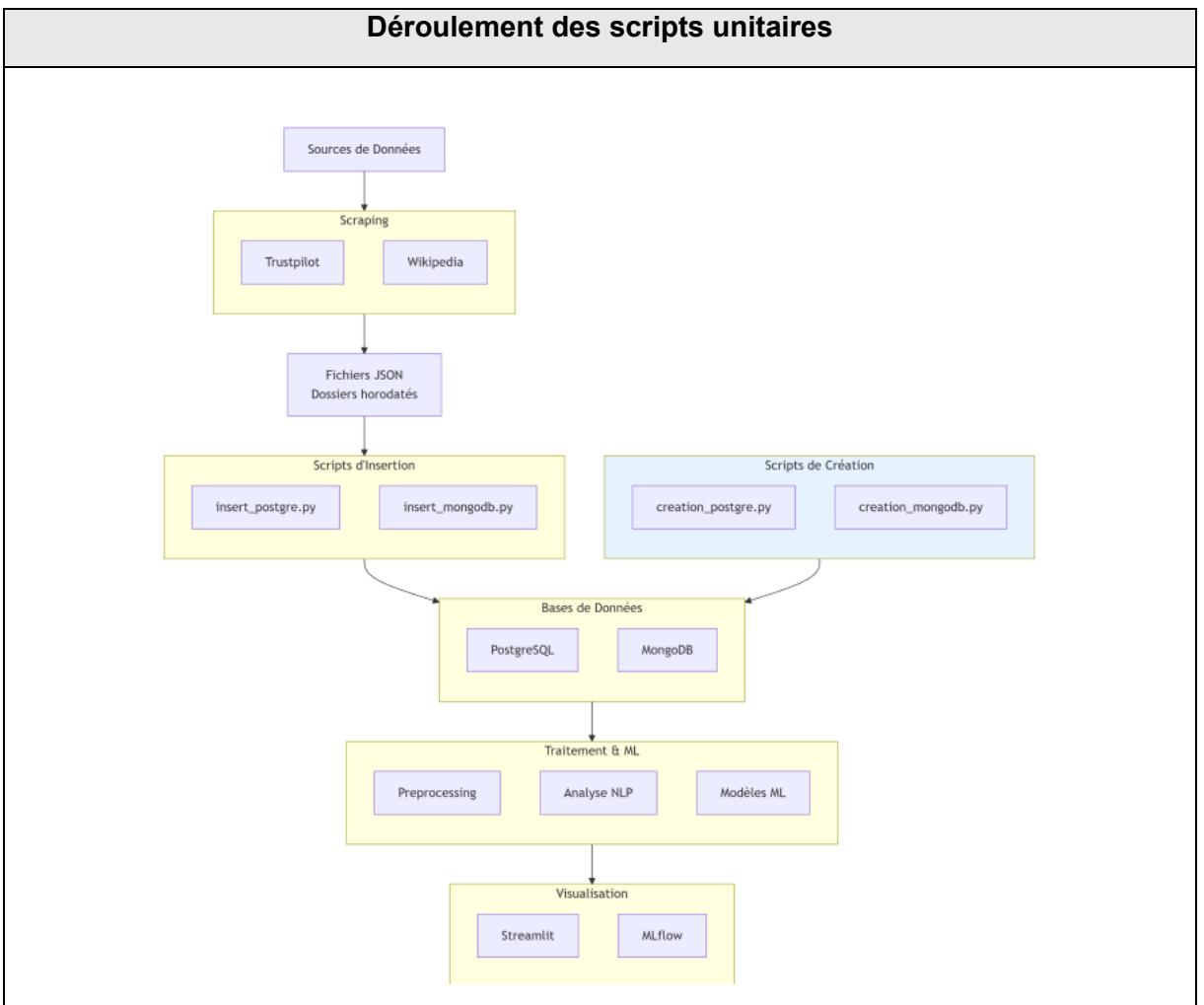
# Preprocessing & NLP
python preprocess/snapshot_data.py
python preprocess/sentiment_analysis.py
python preprocess/clean_data.py
python preprocess/preprocess_clean_avis.py
python models/train_dual_models.py

# NLflow
bash nlflow/start_nlflow.sh
python nlflow/nlflow_tracking.py

# Application
streamlit run app/app_streamlit.py
```

Scripts automatiques

- (cde_env) **datascientest@datascientest:~/cde\$ ls run_all.sh**
run_all.sh
- (cde_env) **datascientest@datascientest:~/cde\$** █



3. Organisation des répertoires et environnement Python

```
(cde_env) datascientest@datascientest:~/cde$ tree -L 1 .
.
├── cde_env
├── data
├── docker-compose.yml
├── docker-data
├── docs
├── log
├── LOGFILE_ML
├── logs
├── lost+found
├── nltk_data
├── notebooks
├── README.md
├── requirements.txt
├── run_all_insert.log
├── run_all_insert.sh
├── run_all_ml.log
├── run_all_ml.sh
├── scripts
├── setup.sh
├── spacy_models
└── tests
└── tmp

13 directories, 9 files
```

```
(cde_env) datascientest@datascientest:~/cde$ python --version && which python && echo "Env virtuel: $VIRTUAL_ENV"
Python 3.10.12
/home/datascientest/cde/cde_env/bin/python
Env virtuel: /home/datascientest/cde/cde_env
```

1.5 COMPOSANTS TECHNIQUES DU PROJET

Tableau des Technologies et Versions

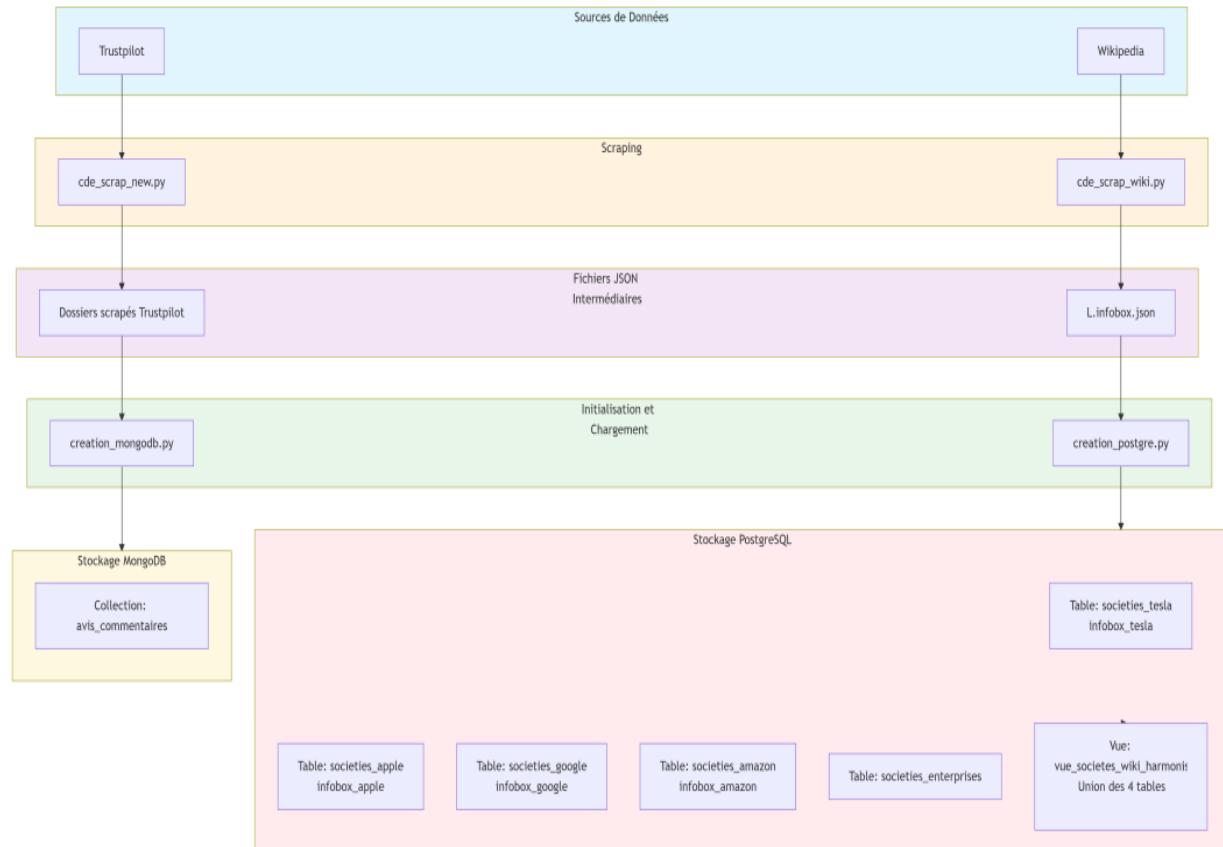
| Catégorie | Technologie / Composant | Version / Détail | Rôle principal / Commentaire |
|---|-------------------------|---|---|
| Système d'exploitation | Ubuntu | 22.04.5 LTS (Jammy Jellyfish) | OS principal pour l'exécution des scripts et le hosting des conteneurs. |
| Hyperviseur / Environnement | Hyper-V | Microsoft Hyper-V) | Environnement de virtualisation (exécution de la machine Ubuntu). |
| Langage de programmation | Python | 3.10.12 | Langage principal pour le scraping, l'ETL, le ML et les scripts d'automatisation. |
| Gestionnaire d'environnement | Virtualenv | (cde_env) | Environnement virtuel Python isolé pour le projet. |
| Bases de données | PostgreSQL | 15 (Conteneur) / Client : 14.18 | Base de données relationnelle pour les métadonnées structurées et le backend de MLflow. |
| | MongoDB | 6 (Conteneur) | Base de données document pour le stockage flexible des avis clients bruts (JSON). |
| Conteneurisation & Orchestration | Docker | 28.3.2 (build 578ccf6) | Moteur de conteneurisation pour l'isolation et le déploiement des services. |
| | Docker Compose | Docker Compose version v2.27.1 | Outil pour définir et orchestrer les services multi-conteneurs (PostgreSQL, MongoDB). |
| Machine Learning & NLP | MLflow | (Version définie dans requirements.txt) | Plateforme de gestion du cycle de vie des modèles ML (tracking, registry). |
| | spaCy | (Version définie dans requirements.txt) | Librairie NLP pour le traitement avancé du texte (lemmatisation, NER) en français. |

| Catégorie | Technologie / Composant | Version / Détail | Rôle principal / Commentaire |
|-------------------------------|-------------------------|---|--|
| | NLTK | (Version définie dans requirements.txt) | Librairie NLP pour le traitement de base du texte (stopwords, tokenisation). |
| Web Framework & Visualisation | Streamlit | (Version définie dans requirements.txt) | Framework pour la création de l'application web interactive de visualisation et de test des modèles. |
| Gestion des Dépendances | Pip | (Version implicite, e.g., 23.2.1) | Gestionnaire de paquets pour installer les librairies Python listées dans requirements.txt. |
| Matériel | CPU | Architecture x86_64 | |
| | Mémoire RAM | 16 GB | |
| Hôte | Système Hôte | Windows 11 Familial | OS de la machine physique hôte de la virtualisation. |

| Docker-compose.yml | .env |
|--|---|
| <pre>● (cde_env) datascientest@datascientest:~/cde\$ more docker-compose.yml services: postgres: image: postgres:15 container_name: postgres-cde restart: unless-stopped environment: POSTGRES_USER: admin POSTGRES_PASSWORD: admin POSTGRES_DB: trustpilot ports: - "5432:5432" volumes: - ./docker-data/postgres:/var/lib/postgresql/data mongo: image: mongo:6 container_name: mongo-cde restart: unless-stopped environment: MONGO_INITDB_ROOT_USERNAME: admin MONGO_INITDB_ROOT_PASSWORD: admin MONGO_INITDB_DATABASE: trustpilot ports: - "27017:27017" volumes: - ./docker-data/mongodb:/data/db command: --auth</pre> | <pre>● (cde_env) datascientest@datascientest:~/cde\$ more .env # PostgreSQL POSTGRES_USER=admin POSTGRES_PASSWORD=admin POSTGRES_DB=trustpilot POSTGRES_HOST=localhost POSTGRES_PORT=5432 # MongoDB MONGO_USER=admin MONGO_PASSWORD=admin MONGO_HOST=localhost MONGO_PORT=27017 MONGO_DB=trustpilot MONGO_AUTH_SOURCE=admin MONGO_URI=mongodb://admin:admin@localhost:27017/trustpilot?authSource=admin # Chemins données # Racine projet BASE_DIR=/home/datascientest/cde # Données brutes Trustpilot DATA_RAW_TRUSTPILOT=/home/datascientest/cde/data/trustpilot # Données exports CSV DATA_EXPORTS=\${BASE_DIR}/data/export # Données pour les report DATA_REPORT=\${BASE_DIR}/data/report # Données traitées DATA_PROCESSED=\${BASE_DIR}/data/processed # Modèles ML sauvegardés DATA_MODEL=\${BASE_DIR}/data/model # Données exportées de WIKI WIKI_DATA_DIR=data/wikipedia # Répertoires spéciaux (nlp, logs, docker, etc) SPACY_MODELS=\${BASE_DIR}/spacy_models NLTK_DATA=\${BASE_DIR}/nltk_data LOG_DIR=/home/datascientest/cde/log</pre> |

2. COLLECTE DES DONNEES (PARTIE 1)

2.1 ARCHITECTURE FONCTIONNELLE DE COLLECTE DES DONNEES : WEB SCRAPING

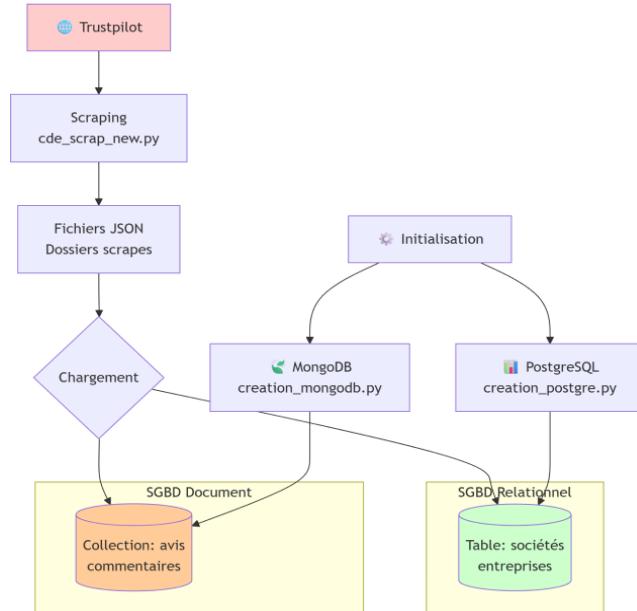


Description des composants :

1. **Sources de données** : Wikipédia et Trustpilot comme sources initiales
 2. **Processus de scraping** : Scripts Python dédiés pour chaque source
 3. **Stockage intermédiaire** : Fichiers JSON avant chargement en base
 4. **Initialisation** : Scripts de création et de chargement des bases
 5. **Bases de données finales** :
 - o PostgreSQL pour les données structurées (entreprises)
 - o MongoDB pour les données documents (avis clients)
 - o Vue SQL pour l'unification des données Wikipédia

2.2 COLLECTE AUTOMATISEE D'AVIS CLIENTS VIA WEB SCRAPING (PYTHON/TRUSTPILOT)

1. Schéma d'architecture fonctionnelle : récupération des données Trustpilot



Le script **cde_scrap_new.py** est une version adaptée de l'ancien **script cde_scrap.py**, modifié à la suite de la refonte de Trustpilot en React.js.

Il a pour rôle de collecter les avis clients en analysant le contenu JSON intégré dans la page HTML, où les informations sont désormais dispersées dans différents blocs de données. Ce script m'a permis de rétablir la collecte fiable des commentaires malgré les changements techniques opérés par Trustpilot.

Afin de prévenir les impacts de futures modifications du site, j'ai développé deux scripts spécifiques — **auto_scrap_chronopost.py** et **auto_scrap_vinted.py** — ont été développés pour automatiser la collecte et l'archivage des avis Trustpilot concernant ces deux entreprises.

Ainsi, même en cas de changement dans la structure HTML ou le format des données, ce script n'aura plus besoin d'être modifié et **un stock historique restera exploitable**.

Exemple de scraping pour chronopost.fr au 21/08/2025 :

```

(cde_env) datacientest@datacientest:/cde/scripts/scraping$ python3 cde_scrap_new.py
Entrez le nom de domaine (ex: chronopost.fr) : chronopost.fr
Nombre max de pages à scraper (défaut 30) : 2
2025-08-14 21:28:03,411 - INFO - Reprise à partir de la page 1572
2025-08-14 21:28:03,411 - INFO - Scraping page 1572: https://fr.trustpilot.com/review/chronopost.fr?page=1572
2025-08-14 21:28:03,939 - INFO - Page 1572 traitée, 20 avis récupérés
2025-08-14 21:28:03,939 - INFO - Pause 5 secondes avant page suivante...
2025-08-14 21:28:08,944 - INFO - Scraping page 1573: https://fr.trustpilot.com/review/chronopost.fr?page=1573
2025-08-14 21:28:09,363 - INFO - Page 1573 traitée, 20 avis récupérés
2025-08-14 21:28:09,363 - INFO - Pause 6 secondes avant page suivante...
2025-08-14 21:28:15,579 - INFO - Informations générales sauvegardées dans /home/datacientest/cde/data/trustpilot/chronopost/scrap_chronopost_20250814_212815/chronopost_informations_generales_20250814_212815.txt
2025-08-14 21:28:15,606 - INFO - Données sauvegardées dans /home/datacientest/cde/trustpilot/chronopost/scrap_chronopost_20250814_212815
  
```

PS : Le système utilise le fichier dernière_page.txt pour reprendre le scraping exactement là où il s'était interrompu, tout en respectant scrupuleusement les règles du fichier robots.txt. Cela garantit une collecte éthique et conforme aux exigences légales des plateformes.

2. Résultats : 66 864 avis collectés pour 4 sociétés (14/8/2025)

Pour rappel, le scraping cible la version française de Trustpilot, l'analyse porte exclusivement sur des avis rédigés en français.

```
=====
~~~~~~ STRUCTURE MONGODB ~~~~~~
=====

◆ BASE DE DONNÉES: trustpilot
=====

~~~~~ COLLECTION: AVIS_TRUSTPILOT ~~~~~~
=====

⌚ Documents totaux: 66,864
💾 Taille: 32.80 MB

=====
~~~~~ COLLECTION: SOCIETE ~~~~~~
=====

⌚ Documents totaux: 4
💾 Taille: 0.00 MB

DOC DOCUMENT #1
-----
{ '_id': ObjectId('689ccbdec38e5d4bedb39f62'),
  'date_extraction': datetime.datetime(2025, 8, 13, 18, 47, 48),
  'nom': 'temu.com',
  'nombre_avis': 38847,
  'nombre_commentaires': 0,
  'note_1': 19689,
  'note_2': 2419,
  'note_3': 1890,
  'note_4': 3163,
  'note_5': 11686,
  'note_globale': 2.3,
  'pages_scrapées': '',
  'secteur': 'Temu',
  'total_avis': 38847,
  'url': 'https://fr.trustpilot.com/review/temu.com?page=151'}
```

```
41 def print_database_info(client, sample_size=5):
42     """Affiche les bases, collections et des exemples de documents"""
43     print_separator("Structure MongoDB")
44
45     for db_name in sorted(client.list_database_names()):
46         if db_name not in ['admin', 'config', 'local']:
47             db = client[db_name]
48             logging.info(f"\n◆ BASE DE DONNÉES: {db_name}")
49
50             for col_name in sorted(db.list_collection_names()):
51                 print_separator(f"Collection: {col_name}")
52
53                 # Afficher les stats de la collection
54                 stats = db.command("collstats", col_name)
55                 logging.info(f"⌚ Documents totaux: {stats['count']}")  
logging.info(f"💾 Taille: {stats['size']/(1024*1024):.2f} MB\n")
56
57                 # Afficher des exemples de documents
58                 try:
59                     for i, doc in enumerate(db[col_name].find().limit(sample_size), 1):
60                         print_document(doc, i)
61                 except Exception as e:
62                     logging.error(f"🔴 Impossible de lire les documents: {str(e)}")
63                     continue
```

3. Dossier de scraping

Fichiers communs :

- history.txt

Historique des exécutions (dates, statuts, nombre d'avis collectés).

- dernière_page.txt

Numéro de la dernière page scrapée (pour reprendre où le script s'est arrêté).

```
● datacientest@datacientest:~/cde/data/trustpilot/chronopost$ more dernière_page.txt  
1573
```

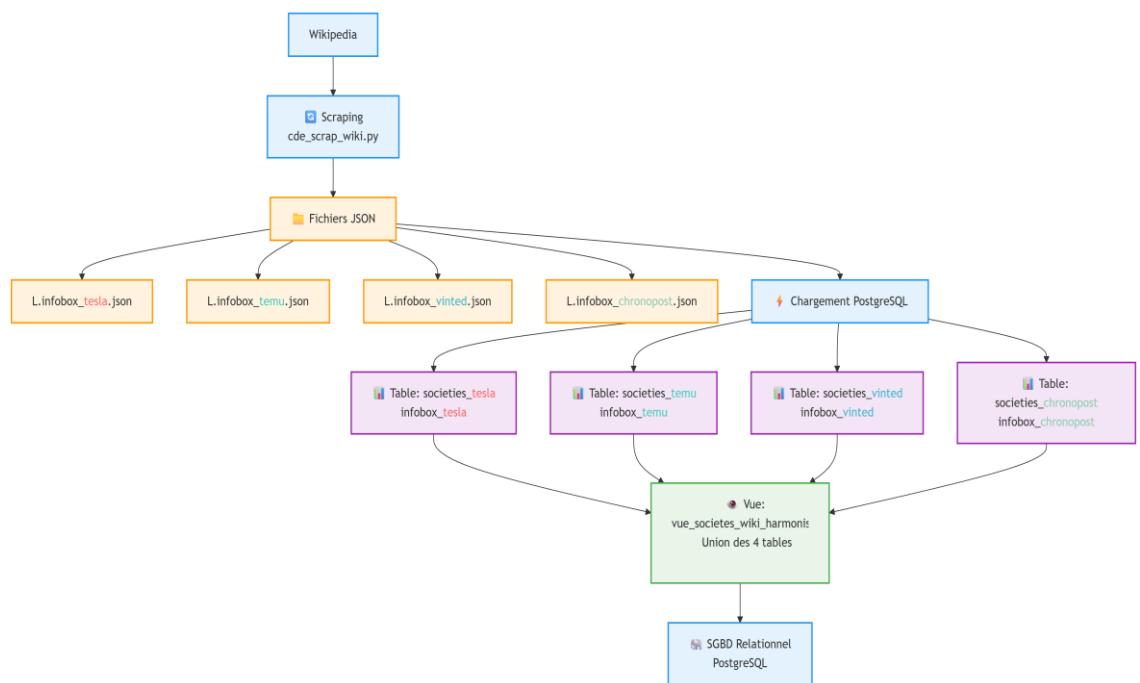
Dossiers exemple : scrap_tesla_<date>/

2.3 COLLECTE MENSUELLE DES DONNEES SOCIETALES VIA API (WIKIPEDIA)

1. Métadonnées enrichies (SIREN, chiffre d'affaires, secteurs)

Dans le cadre de l'avancement sur le projet principal d'analyse sentiment des avis Trustpilot, et pour donner suite à une proposition de mon tuteur visant à enrichir la valeur métier de la solution, **un module optionnel de collecte mensuelle de données Wikipédia** a été développé. Cette démarche permet d'enrichir l'analyse avec des données structurelles sur les entreprises.

2. Schéma d'architecture fonctionnelle : récupération des données Wikipédia



Flux de traitement : Extraction des infobox Wikipédia → Conversion JSON → Chargement PostgreSQL → Vue unifiée

1. **Scraping** : Extraction automatique des données depuis Wikipédia
2. **Fichiers JSON** : Stockage intermédiaire des infobox (1 fichier par société)
3. **PostgreSQL** : Chargement dans des tables dédiées (1 table par société)
4. **Vue harmonisée** : Agrégation des 4 tables en une seule vue unifiée
5. **SGBD relationnel** : PostgreSQL sert de base de données centralisée

Sociétés traitées : Tesla, Temu, Vintered, Chronopost

Finalité : Centraliser et uniformiser les données Wikipédia pour analyse

3. Méthodologie de collecte

Installation des dépendances :

- pip install Wikipédia -api # Pour utiliser l'API officielle de Wikipédia
- pip install requests mwparserfromhell # Pour le parsing avancé des infobox

4. Méthodologie de collecte

- Connexion à l'API Wikipédia via des requêtes HTTP avec le module requests
- Extraction des infobox (cartouches d'information structurée) avec mwparserfromhell
- Nettoyage des données (suppression des balises wiki, formats spéciaux)
- Récupération complémentaire :
 - I. Logo de l'entreprise via Wikipédia Commons
 - II. Numéro SIREN (pour les entreprises françaises) via Wikidata

5. Architecture des scripts

a) Scripts développés :

I. `cde_scrap_wiki.py` (Collecte) :

- Récupère les infobox des pages entreprises
- Nettoie et structure les données
- Exporte en JSON (1 fichier par société)

```
# dataclienttest@dataclienttest-Virtual-Machine:~/code$ python3 cde_scrap_wiki.py*
 ❸ Traitement de : Chronopost
 ❷ Fichier créé : 3_infobox.json

 ❸ Traitement de : Vinted
 ❷ Fichier créé : 4_infobox.json

 ❸ Traitement de : Temu (marché)
 ❷ Fichier créé : 1_infobox.json

 ❸ Traitement de : Tesla (automobile)
 ❷ Fichier créé : 2_infobox.json

❷ Export terminé :
- Fichiers individuels : 1_infobox.json, 2_infobox.json, etc.
- Fichier global : entreprises_infobox.json
```

II. `cde_insert_wiki.py` (Insertion PostgreSQL) :

- Crée des tables aux champs spécifiques de chaque entreprise
- Implémente une logique de typage automatique des colonnes
- Gère les caractères spéciaux et formats hétérogènes

```
# dataclienttest@dataclienttest-Virtual-Machine:~/code$ python3 cde_insert_wiki.py
❸ Connexion à la base PostgreSQL réussie.

❸ Création de la table société_temu_wiki...
❷ Table société_temu_wiki créée avec succès.

❸ Import des données pour temu...
❷ Données importées dans société.temu_wiki

❸ Création de la table société_tesla_wiki...
❷ Table société_tesla_wiki créée avec succès.

❸ Import des données pour tesla...
❷ Données importées dans société.tesla_wiki

❸ Création de la table société_vinted_wiki...
❷ Table société_vinted_wiki créée avec succès.

❸ Import des données pour vinted...
❷ Données importées dans société.vinted_wiki

❷ 4/4 entreprises traitées avec succès.
Connexion fermée proprement.
```

b) Données collectées :

- Informations générales (siège, fondation, dirigeants...)
- Chiffres clés (CA, effectifs)
- Identifiants légaux (SIREN pour Chronopost)
- URLs des logos

• Structure PostgreSQL :

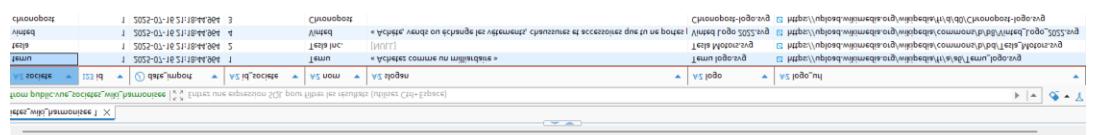
- Tables spécifiques par entreprise
(societe_tesla_wiki, societe_chronopost_wiki)
- Schéma flexible adapté à l'hétérogénéité des données Wikipédia

Pour assurer la **cohérence et la reproductibilité** du processus, je procède à la **récréation complète des bases de données** à chaque exécution pour reconstruire un **jeu de données exhaustif et fiable**.

Je récupère les données via l'API Wikipédia avec cde_scrap_wiki.py, puis je les charge en PostgreSQL avec cde_insert_wiki.py. Ce script crée des tables pour chaque société : societe_<societe>_wiki

```
> select * from public.societe_temu_wiki
select * from public.societe_tesla_wiki
select * from public.societe_vinted_wiki
select * from public.societe_chronopost_wiki
```

A partir de ces tables, je crée une table harmonisée :
vue_sociedades_wiki_harmonisée



```
create or replace view public.vue_sociedades_wiki_harmonizada as
select *
from public.societe_temu_wiki
union all
select *
from public.societe_tesla_wiki
union all
select *
from public.societe_vinted_wiki
union all
select *
from public.societe_chronopost_wiki;
```

| site_web | président | chiffre_affaires | date_creation | fondateur | effectif | forme_juridique | secteur_activite |
|--|--|---|----------------|-----------|---------------------------------|-----------------------|--|
| [https://temu.com temu.com] | enPDD Holdings Inc. | [NULL] | [NULL] | [NULL] | [NULL] | [NULL] | [NULL] |
| [http://www.tesla.com www.tesla.com] | Elon Musk | ((augmentation)) 97,7 milliards de US\$ (2024) | 1 Juillet 2003 | age=oui | Martin Eberhard Marc Tarpenning | SAS | Construction automobile |
| [NULL] | (seule la propriété P488 est utilisée) | ((augmentation)) 996,3 millions € (2023) | [NULL] | [NULL] | Milda Mitkute Justas Janauskas | estimé à 3900 en 2019 | Activités des agences de publicité |
| [http://www.chronopost.fr chronopost.fr] | [NULL] | 1 292 100 000 € en 2019 - comptes récents non disponibles | 1985 | [NULL] | Société par actions simplifiée | | Autres activités de poste et de courrier |

c) Vue PostgreSQL

| vue_socetes_wiki_harmonisee | |
|-----------------------------|--|
| AZ societe | |
| 123 id | |
| ⌚ date_import | |
| AZ id_societe | |
| AZ nom | |
| AZ slogan | |
| AZ logo | |
| AZ logo_url | |
| AZ site_web | |
| AZ president | |
| AZ chiffre_affaires | |
| AZ date_creation | |
| AZ fondateur | |
| AZ effectif | |
| AZ forme_juridique | |
| AZ secteur_activite | |
| AZ societe_mere | |

Vue de la public.vue_socetes_wiki_harmonisee

Filter connections by name

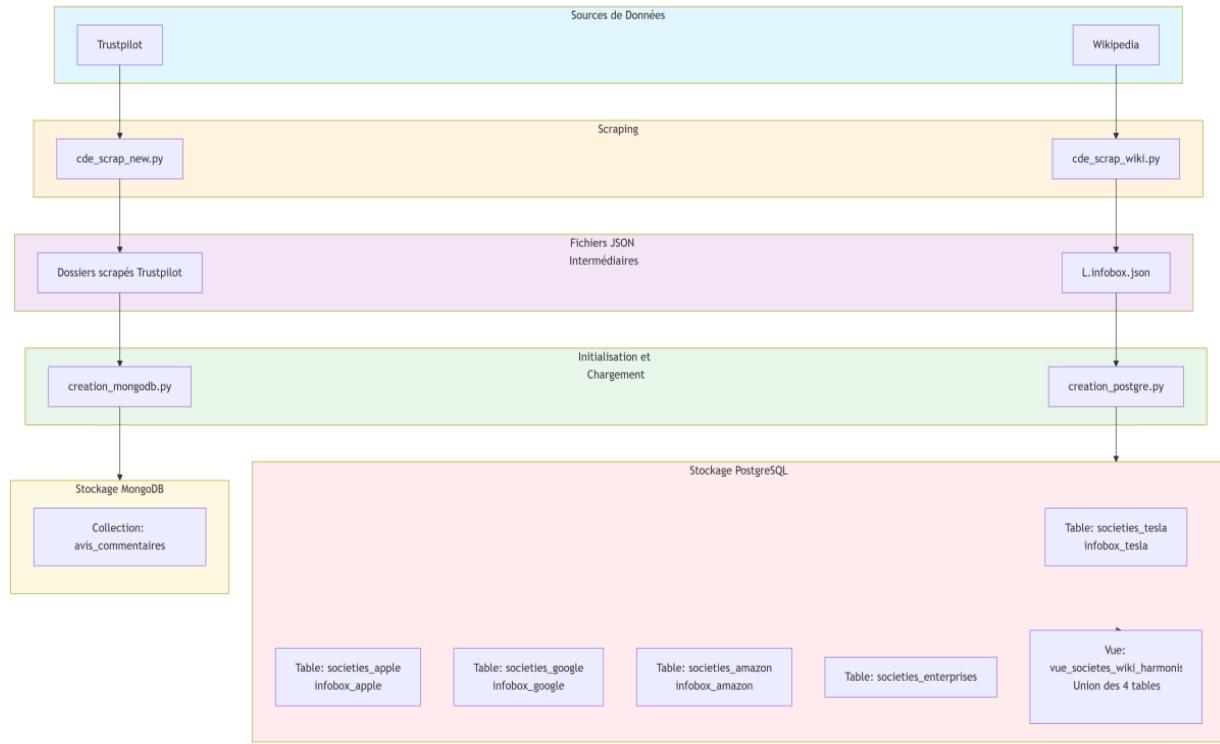
| Show SQL | | | | | | |
|--|------------|--------|-------------------------|---------------|------------|--|
| Entrez une expression SQL pour filtrer les résultats (utilisez Ctrl+Space) | | | | | | |
| | AZ societe | 123 id | ⌚ date_import | AZ id_societe | AZ nom | AZ slogan |
| Totals | | | | | | |
| Tableau | Temu | 1 | 2025-07-16 21:18:44.964 | 1 | Temu | « Achetez comme un milliardaire » |
| Temu | tesla | 1 | 2025-07-16 21:18:44.964 | 2 | Tesla Inc. | (NULL) |
| Vinted | vinted | 1 | 2025-07-16 21:18:44.964 | 4 | Vinted | « Achète, vend ou échange les vêtements, chaussures et accessoires que tu ne portes plus » |
| Chronopost | chronopost | 1 | 2025-07-16 21:18:44.964 | 3 | Chronopost | |

Filter connections by name

- trustpulse 172.25.3.215:5432
- Bases de données
 - public
 - Schémas
 - Tables
 - Tables étrangères
 - Vues
 - vue_socetes_wiki_harmonisee
 - Colonnes
 - Dépendances
 - Déclencheurs
 - Règles

3. STOCKAGE ET MODELISATION DES DONNEES (PARTIE 2)

3.1 ARCHITECTURE FONCTIONNELLE : BDD



Le choix d'une architecture multibases de données est motivé par l'adéquation de chaque technologie à un besoin spécifique :

- **PostgreSQL assure deux rôles distincts et essentiels :**
 1. **Stockage structuré** : Il héberge les métadonnées normalisées des entreprises (schéma UML) et les données issues de Wikipédia, permettant des requêtes SQL complexes et des jointures efficaces pour l'analyse business.
 2. **Tracking des modèles ML** : Une instance dédiée (mlflow_db) sert de backend à MLflow pour stocker de manière fiable toutes les métadonnées d'expérimentation (paramètres, métriques, artefacts), garantissant la reproductibilité et le versioning.
- **MongoDB, en tant que base de documents, est optimisé** pour le stockage des avis clients scrapés depuis Trustpilot. Sa flexibilité schémaless permet d'ingérer et de restituer facilement les documents JSON bruts et potentiellement variables sans perte d'information, et ce à grand volume.

Ainsi, cette séparation permet à chaque SGBD d'exceller dans son domaine de prédilection : la rigueur relationnelle pour PostgreSQL, et l'agilité documentaire pour MongoDB.

Description des composants :

1. Sources de Données

- **Wikipédia** : Informations structurées des entreprises via les infobox
- **Trustpilot** : Avis clients avec notes et commentaires
- **Wikidata** : Informations supplémentaires normalisées (SIREN, logos, etc.)

2. Processus d'Extraction

- Scripts Python dédiés pour chaque source
- Utilisation d'APIs (Wikipédia, Wikidata) et scraping web

3. Stockage Intermédiaire

- Fichiers JSON pour transformation et nettoyage des données
- Structure horodatée pour traçabilité

4. Bases de Données

PostgreSQL (données structurées) :

- Table principale `societe` avec informations communes
- Tables spécifiques par entreprise pour les données Wiki
- Vue harmonisée unifiant les données
- Table `avis_trustpilot` pour l'analyse ML

MongoDB (données documents) :

- Collection `avis_trustpilot` avec documents complets
- Collection `societe` avec métadonnées des entreprises

5. MLflow

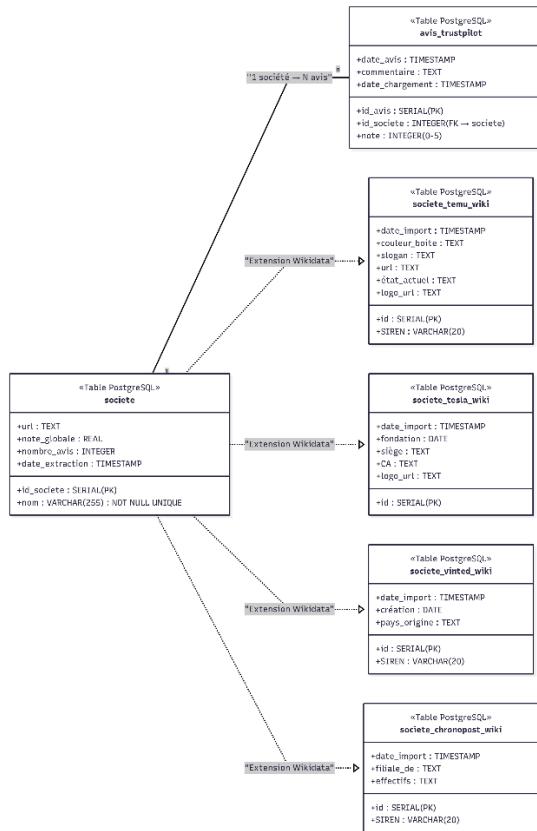
- Analyse de sentiment avec modèle BERT
- Prédiction automatique de notes
- Entraînement et évaluation des modèles

6. Déploiement

- Containerisation avec Docker
- Gestion via Docker Compose pour la reproductibilité

3.2 POSTGRESQL

1. Schéma UML (tables société)



Ce schéma représente une modélisation de base de données relationnelle utilisant PostgreSQL, avec des extensions spécifiques pour gérer des données liées à une société et ses avis, ainsi que des informations supplémentaires importées depuis Wikidata. Voici une analyse détaillée de chaque composant :

Table société

C'est la table centrale de cette base de données. Elle contient les informations suivantes :

- **uri** : Un identifiant unique sous forme de texte pour chaque société.
- **note_globale** : Une note globale de type REAL (nombre décimal) attribuée à la société.
- **nombre_avis** : Un entier (INTEGER) indiquant le nombre total d'avis pour la société.
- **date_extraction** : Un horodatage (TIMESTAMP) indiquant quand les données ont été extraites.
- **id_societe** : Une clé primaire de type SERIAL qui identifie de manière unique chaque société.

2. Extensions Wikidata

Les informations supplémentaires sur chaque société sont importées depuis Wikidata et réparties dans plusieurs tables :

Table societe_temu_wiki :

- date_import : Un horodatage (TIMESTAMP) indiquant quand les données ont été importées.
- titre_wiki : Le titre de la page Wikidata (TEXT).
- slogan : Le slogan de la société (TEXT).
- uri : L'URI de la page Wikidata (TEXT).
- etat_actuel : L'état actuel de la société (TEXT).
- logo_url : L'URL du logo de la société (TEXT).
- id : Une clé primaire de type SERIAL.
- siren : Un identifiant SIREN de type VARCHAR(20).

Table societe_tesla_wiki :

- date_import : Un horodatage (TIMESTAMP) indiquant quand les données ont été importées.
- fondation : La date de fondation de la société (DATE).
- ca : Le chiffre d'affaires (TEXT).
- logo_url : L'URL du logo de la société (TEXT).
- id : Une clé primaire de type SERIAL.
- siren : Un identifiant SIREN de type VARCHAR(20).

Table societe_vinted_wiki :

- date_import : Un horodatage (TIMESTAMP) indiquant quand les données ont été importées.
- création : La date de création (DATE).
- pays_origine : Le pays d'origine de la société (TEXT).
- id : Une clé primaire de type SERIAL.
- siren : Un identifiant SIREN de type VARCHAR(20).

Table societe_chronopost_wiki :

- date_import : Un horodatage (TIMESTAMP) indiquant quand les données ont été importées.
- filiale_de : La société mère (TEXT).
- effectifs : Le nombre d'employés (TEXT).
- id : Une clé primaire de type SERIAL.
- siren : Un identifiant SIREN de type VARCHAR(20).

3. MLflow

Dans PostgreSQL:

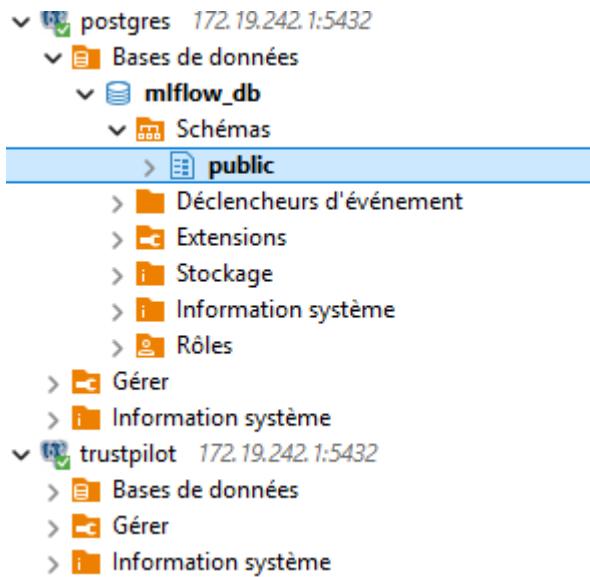
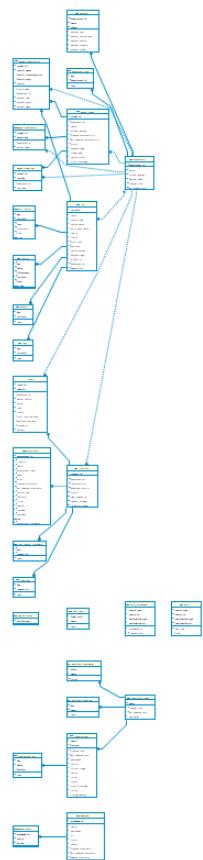


Schéma partiel PostgreSQL MLflow :



4. Chargement des données dans PostgreSQL

Pour assurer la cohérence et la reproductibilité du processus, je procède à la **recréation complète des bases de données** à chaque exécution pour reconstruire un jeu de données exhaustif et fiable.

Je récupère les données de scraping : `/home/datascientest/cde/scripts/scraping/insert_postgre.py`

- Table `societe` :

| <code>i23 id_societe</code> | <code>AZ nom</code> | <code>AZ url</code> | <code>AZ secteur</code> | <code>i23 note_globale</code> | <code>i23 nombre_avis</code> | <code>i23 note_1</code> | <code>i23 note_2</code> | <code>i23 note_3</code> | <code>i23 note_4</code> | <code>i23 note_5</code> | <code>i23 date_extraction</code> |
|-----------------------------|---------------------|---|--------------------------|-------------------------------|------------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|----------------------------------|
| 1 | temu | https://fr.trustpilot.com/review/temu | Shopping & mode | 2,3 | 37 511 | 18 759 | 2 340 | 1 828 | 3 099 | 11 485 | 2025-07-16 23:24:18,000 |
| 3 | chronopost | https://fr.trustpilot.com/review/chronopost | Services aux entreprises | 3,6 | 155 257 | 52 016 | 3 936 | 2 825 | 8 362 | 88 118 | 2025-07-17 00:26:42,000 |
| 4 | vinted | https://fr.trustpilot.com/review/vinted | Shopping & mode | 3,3 | 41 214 | 17 493 | 1 549 | 1 603 | 2 722 | 17 847 | 2025-07-17 01:17:57,000 |
| 2 | tesla | https://fr.trustpilot.com/review/tesla | Véhicules & transport | 1,8 | 1 657 | 1 097 | 88 | 34 | 60 | 538 | 2025-06-25 19:42:45,000 |

Exemple d'enregistrement pour Temu pour mettre à jour la table **société** ci -dessus :

```
(cde_env) datascientest@datascientest:~/cde/temu/scrap_temu_20250612_180752$ more temu_informations_generales_20250612_180752.txt
{
    "societe": "temu",
    "url": "https://fr.trustpilot.com/review/temu",
    "secteur": "Shopping & mode",
    "note_globale": "2.4",
    "nombre_avis": 35916,
    "repartition_avis": {
        "1": 17649,
        "2": 2217,
        "3": 1747,
        "4": 3013,
        "5": 11290
    },
    "date_extraction": "2025-06-12 18:07:16",
    "nombre_commentaires": 199
}
```

- Création de la base PostgreSQL : `create_postgres_db.py` → DROP & CREATE de la base Trustpilot avec les tables `societe` et `avis_trustpilot` (ajout `date_charge`).
- Chargement de la base PostgreSQL : `insert_postgre.py`
Ce second script lit ces sous-répertoires horodatés, charge les fichiers JSON, et prépare les données pour insertion pour insérer les données dans PostgreSQL
 - Données structurées insérées dans PostgreSQL, avec gestion des tables et date de chargement pour traçabilité.
- Chargement Wikipédia : `cde_scrap_wiki.py / cde_insert_wiki.py`**
 - `cde_scrap_wiki` extrait et transforme les données des infobox Wikipédia pour 4 entreprises (Temu, Tesla, Chronopost, Vinted). Il utilise l'API Wikipédia pour récupérer le contenu, nettoie les balises Wiki (liens, modèles, références), et enrichit les données avec des métadonnées comme le **SIREN** (via Wikidata) ou l'URL des logos. Les résultats sont sauvegardés en JSON (un fichier par entreprise + un fichier consolidé).
 - `cde_insert_wiki` charge les JSON générés dans une base PostgreSQL. Il crée des tables dédiées (`societe_[nom]_wiki`) avec une structure dynamique (colonnes adaptées aux champs des infobox) et une traçabilité intégrée (`date_import`). Les types de données sont déterminés automatiquement (ex : SIREN en VARCHAR, dates en TEXT).

3.3 MONGODB

1. Collections NoSQL : AVIS_TRUSTPILOT

MongoDB a été choisi pour stocker les avis clients collectés depuis Trustpilot en raison de sa flexibilité et de sa capacité à gérer des données semi-structurées.

Contrairement à **PostgreSQL**, qui est utilisé pour les données structurées comme les métadonnées des entreprises, MongoDB permet de stocker des documents JSON sans schéma fixe. Cela est particulièrement utile pour les avis clients, qui peuvent varier en termes de champs et de structure (par exemple, certains avis peuvent contenir des images, des réponses de l'entreprise, ou des métadonnées supplémentaires).

Pour assurer la **cohérence et la reproductibilité** du processus, je procède à la **recréation complète des bases de données** à chaque exécution pour reconstruire un **jeu de données exhaustif et fiable**.

2. Structure des collections

La base de données MongoDB contient deux collections principales :

- **avis_trustpilot** : Stocke les avis clients collectés depuis Trustpilot.
- **societe** : Contient des informations générales sur les entreprises analysées (Tesla, Temu, Chronopost, Vinted).

a) Collection avis_trustpilot

Cette collection stocke les avis sous forme de documents JSON. Chaque document représente un avis individuel et contient les champs suivants :

| Champ | Type | Description |
|-----------------|----------|---|
| _id | Objectid | Identifiant unique généré automatiquement par MongoDB. |
| societe | String | Nom de l'entreprise concernée (ex. : "Tesla", "Chronopost"). |
| note | Integer | Note attribuée par le client (de 1 à 5 étoiles). |
| commentaire | String | Texte de l'avis. |
| date_avis | Date | Date de publication de l'avis. |
| auteur | String | Pseudonyme de l'auteur (anonymisé pour respecter le RGPD). |
| titre | String | Titre de l'avis (si disponible). |
| date_changement | Date | Date à laquelle l'avis a été inséré dans la base de données. |
| sentiment | String | Sentiment prédit (négatif, neutre, positif) via le modèle BERT. |
| note_predite | Integer | Note prédite (de 1 à 5 étoiles) via le modèle BERT. |
| lien | String | Lien vers l'avis sur Trustpilot (optionnel). |

b) Collection societe

Cette collection contient des informations générales sur les entreprises cibles. Chaque document représente une entreprise et inclut des métadonnées comme le secteur d'activité et la note globale.

| Champ | Type | Description |
|-----------------|----------|--|
| _id | ObjectId | Identifiant unique généré automatiquement par MongoDB. |
| nom | String | Nom de l'entreprise. |
| secteur | String | Secteur d'activité (ex. : "Automobile", "E-commerce"). |
| note_globale | Float | Note globale sur Trustpilot (ex. : 4.2). |
| nombre_avis | Integer | Nombre total d'avis collectés pour cette entreprise. |
| date_changement | Date | Date de la dernière mise à jour des données. |

3. Processus de création et de chargement des données structure des collections

- Création de la Base MongoDB : **create_mongo_db.py** → Suppression et recréation des collections avis_trustpilot et societe avec ajout du champ date_changement et indexation

```
2025-08-16 18:50:34,062 - INFO -
*****
** 🔍 Connexion à MongoDB
*****
2025-08-16 18:50:34,075 - INFO - ✅ Connexion réussie à la base : trustpilot
2025-08-16 18:50:34,076 - INFO -
*****
** 🚧 Création des index
*****
2025-08-16 18:50:34,076 - INFO - 🌟 Création des index...
2025-08-16 18:50:34,078 - INFO - ✅ Index créés avec succès.
2025-08-16 18:50:34,078 - INFO -
*****
** 📊 Statistiques des collections
*****
2025-08-16 18:50:34,086 - INFO - avis_trustpilot : 66864 documents | 32.8 MB
2025-08-16 18:50:34,091 - INFO - societe : 4 documents | 0.0 MB
2025-08-16 18:50:34,096 - INFO -
*****
** 🎯 Aperçu de la collection: societe
Documents: 4 | 📁 Taille: 1,242 octets
*****
2025-08-16 18:50:34,098 - INFO - 🔎 Document 1: {'_id': ObjectId('689ccbdec38e5d4bedb39f62'), 'nom': 'temu.com', 'date_extraction': datetime.datetime(2025, 8, 13, 18, 47, 48), 'nombre_avis': 38847, 'nombre_commentaires': 0, 'note_1': 19689, 'note_2': 2419, 'note_3': 1890, 'note_4': 3163, 'note_5': 11686, 'note_globale': 2.3, 'pages_scrapes': '', 'secteur': 'Temu', 'total_avis': 38847, 'url': 'https://fr.trustpilot.com/review/temu.com?page=151'}
2025-08-16 18:50:34,098 - INFO - 🔎 Document 2: {'_id': ObjectId('689ccbdec38e5d4bedb39f64'), 'nom': 'tesla.com', 'date_extraction': datetime.datetime(2025, 8, 13, 13, 59, 3), 'nombre_avis': 1707, 'nombre_commentaires': 0, 'note_1': 1131, 'note_2': 93, 'note_3': 35, 'note_4': 61, 'note_5': 387, 'note_globale': 1.8, 'pages_scrapes': '', 'secteur': 'Tesla', 'total_avis': 1707, 'url': 'https://fr.trustpilot.com/review/tesla.com?page=1'}
2025-08-16 18:50:34,098 - INFO - 🔎 Document 3: {'_id': ObjectId('689ccbdec38e5d4bedb39f66'), 'nom': 'chronopost.fr', 'date_extraction': datetime.datetime(2025, 8, 12, 19, 40, 31), 'nombre_avis': 158327, 'nombre_commentaires': 0, 'note_1': 53161, 'note_2': 4049, 'note_3': 2888, 'note_4': 8505, 'note_5': 89724, 'note_globale': 3.6, 'pages_scrapes': '', 'secteur': 'Chronopost', 'total_avis': 158327, 'url': 'https://fr.trustpilot.com/review/chronopost.fr?page=1542'}
2025-08-16 18:50:34,098 - INFO - 🔎 Document 4: {'_id': ObjectId('689ccbdfc38e5d4bedb39fb3'), 'nom': 'vinted.fr', 'date_extraction': datetime.datetime(2025, 8, 12, 19, 39, 12), 'nombre_avis': 41850, 'nombre_commentaires': 0, 'note_1': 17787, 'note_2': 1584, 'note_3': 1629, 'note_4': 2768, 'note_5': 18082, 'note_globale': 3.3, 'pages_scrapes': '', 'secteur': 'Vinted FR', 'total_avis': 41850, 'url': 'https://fr.trustpilot.com/review/vinted.fr?page=101'}
2025-08-16 18:50:35,099 - INFO -
*****
** 🎯 Aperçu de la collection: avis_trustpilot
Documents: 66,864 | 📁 Taille: 34,390,251 octets
*****
```

- Chargement de la base MongoDB : **insert_MongoDB.py**
Ce script lit ces sous-répertoires horodatés, charge les fichiers JSON, et prépare les données pour insertion pour insérer les données dans MongoDB

```
(cde_env) dataclientest@dataclientest:~/cde/scripts/utils$ python3 voir_mongodb.py*
=====
===== STRUCTURE MONGODB =====
=====

◆ BASE DE DONNÉES: trustpilot
=====
===== COLLECTION: AVIS_TRUSTPILOT =====
=====

● Documents totaux: 66,864
[Taille: 32.80 MB

[D DOCUMENT #1
-----
{
  '_id': ObjectId('689ccbdee3cf055a687fe650'),
  'auteur': 'Lylyblue',
  'commentaire': 'Une commande et la dernière de toutes !!\n' +
    'À faire absolument ! Ils promettent un '\n' cadeau pour le téléchargement de leur '\n' application et cadeau dont vous ne '\n' verrez jamais la couleur. De plus lors '\n' de la toute première connexion, vous '\n' avez le droit de choisir dix cadeaux '\n' gratuits mais pour cela il faut passer '\n' une commande de 29€ minimum, chose faite '\n' mais au lieu des dix cadeaux seulement '\n' quatre me sont attribués ! Très en '\n' colère leur ai fait un mail ce à quoi '\n' ils m\'ont répondu que les cadeaux '\n' étaient attribués en fonction d\'un "\n' nombre d\'étoiles. Tout leur est bon pour "\n' vendre jusqu\'à mentir sur leurs "\n' démarches. Plus jamais je ne passerai '\n' commande chez eux. En plus produits de '\n' mauvaise qualité. Je préfère rester '\n' fidèle à Shein, qui eux, sont bien plus '\n' sérieux et leurs produits sont de bien '\n' meilleures qualité ! On devrait '\n' interdire des sites tels que Temu',
  'date': '2025-08-13 14:26:04',
  'date_chargement': datetime.datetime(2025, 8, 13, 17, 31, 10, 555000),
  'id_societe': 'temu',
  'note_commentaire': '1',
  'page': 1,
  'societe_nom': 'temu.com',
  'url_page': 'https://fr.trustpilot.com/review/temu.com?page=1'}
}
```

4. Déploiement : Docker Compose (fichier docker-compose.yml).

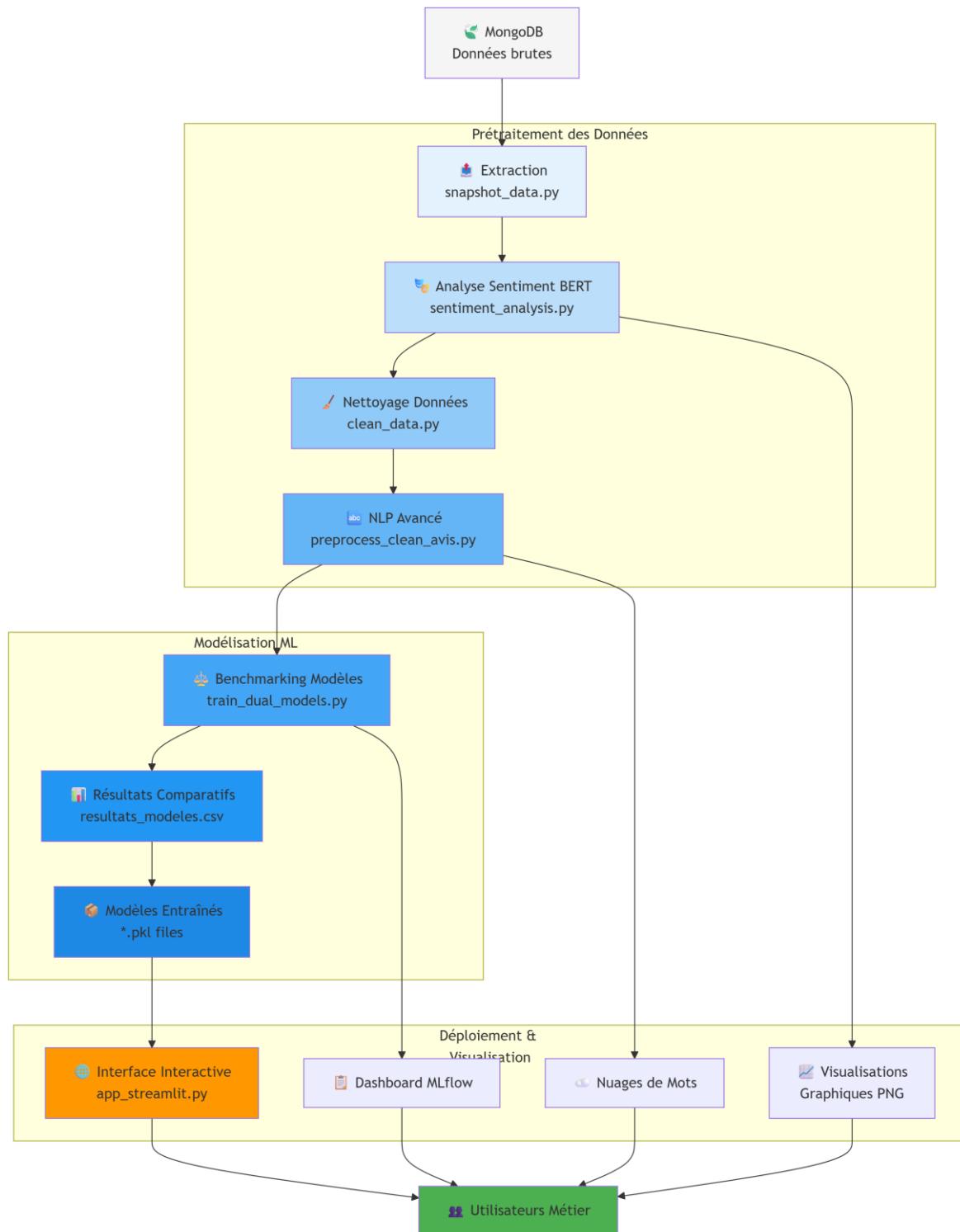
```
(cde_env) dataclientest@dataclientest:~/cde$ more docker-compose.yml
services:
  postgres:
    image: postgres:15
    container_name: postgres-cde
    restart: unless-stopped
    environment:
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: admin
      POSTGRES_DB: trustpilot
    ports:
      - "5432:5432"
    volumes:
      - ./docker-data/postgres:/var/lib/postgresql/data

  mongo:
    image: mongo:6
    container_name: mongo-cde
    restart: unless-stopped
    environment:
      MONGO_INITDB_ROOT_USERNAME: admin
      MONGO_INITDB_ROOT_PASSWORD: admin
      MONGO_INITDB_DATABASE: trustpilot
    ports:
      - "27017:27017"
    volumes:
      - ./docker-data/mongodb:/data/db
    command: --auth
```

4. TRAITEMENT DES DONNEES ET MACHINE LEARNING (PARTIE 3)

4.1 ARCHITECTURE FONCTIONNELLE - PIPELINE ML: ML MODELS → VISUALISATION

1. Schéma d'architecture fonctionnelle



📦 Traçabilité des volumes

| Étape | Nombre | Commentaire |
|----------------------------------|---------------|--|
| Collecte initiale | 66 864 | Avis collectés via scraping. |
| Export MongoDB → CSV | 66 832 | Export consolidé (écart dû à la consolidation des dossiers scrapés). |
| Après dédoublonnage | 60 869 | Suppression de 1 548 doublons textuels. |
| Après filtres (outliers, emojis) | 58 328 | Suppression de 986 outliers + 7 commentaires emojis-only. |

Ce schéma représente une architecture fonctionnelle d'un pipeline de traitement des données pour l'analyse de sentiments. Voici un commentaire succinct :

| Étape | Script | Objectif principal |
|-------|--------------------------|---|
| 1 | snapshot_data.py | Export des avis depuis MongoDB |
| 2 | sentiment_analysis.py | Analyse sentiment BERT + stats + visualisation |
| 3 | clean_data.py | Nettoyage (doublons, emojis, outliers) |
| 4 | preprocess_clean_avis.py | NLP avancé : lemmatisation, TextBlob, LDA, nuages de mots |
| 5 | train_dual_models.py | Benchmarking modèles de classification |
| 6 | app_streamlit.py | Déploiement interactif et évaluation des modèles |

Explications :

1. **Extraction** : Le processus commence par l'**export des données de MongoDB** contenant les données brutes (commentaires) :
`/home/datascientest/cde/scripts/preprocess/snapshot_data.py`
2. **Inférence/Modélisation** : Ce script réalise une **analyse de sentiment sur des avis en utilisant un modèle BERT multilingue**, renforce les négations dans les textes, génère des statistiques et une visualisation de la distribution des notes de sentiment, puis sauvegarde les résultats dans des fichiers CSV et un graphique PNG :
`/home/datascientest/cde/scripts/preprocess/sentiment_analysis.py`
3. **Nettoyage et préparation des données** : Ce script **nettoie et prépare un jeu de données d'avis** en supprimant les doublons, les valeurs manquantes, les commentaires composés uniquement d'emojis, et les outliers en termes de longueur, puis génère un rapport statistique et une visualisation de l'évolution du nombre de lignes après chaque étape de nettoyage :
`/home/datascientest/cde/scripts/preprocess/clean_data.py`
4. **Analyse NLP** : Ce script effectue un **post-traitement avancé de textes d'avis en français**, incluant la lemmatisation, la gestion des négations, la génération de nuages de mots, une analyse de sentiments avec TextBlob, et une modélisation de thèmes avec LDA (Latent Dirichlet Allocation), avant de sauvegarder les résultats et les visualisations correspondantes :
`/home/datascientest/cde/scripts/preprocess/preprocess_clean_avis.py`
5. **Benchmarking" des modèles de classification** : Ce script réalise une évaluation comparative de modèles de classification pour l'analyse de sentiments et de notes à partir de données textuelles :
`/home/datascientest/cde/scripts/models/train_dual_models.py`
6. **Déploiement et évaluation interactive** : Ce script développe une **interface interactive Streamlit** pour évaluer et comparer les prédictions de modèles de Machine Learning sur des avis textuels, en permettant la sélection de tâches et de modèles, et en affichant les résultats des prédictions :
`/home/datascientest/cde/scripts/app/app_streamlit.py`

4.2 EXTRACTION DES AVIS TRUSTPILOT

1. Principe

Le processus commence par l'export des données de MongoDB contenant les données brutes (commentaires) : **snapshot_data.py**

2. Résultats statistiques

```
● (cde_env) dataclientest@dataclientest:~/cde/scripts/preprocess$ python3 snapshot_data.py*
=====
EXPORT MONGODB TRUSTPILOT AVIS
=====
2025-08-13 16:15:43,654 - INFO - Vérification du répertoire d'export : /home/dataclientest/cde/data/export
2025-08-13 16:15:43,655 - INFO - ✓ Répertoire d'export valide
2025-08-13 16:15:43,655 - INFO - Connexion à MongoDB...
2025-08-13 16:15:43,668 - INFO - ✓ Connexion MongoDB réussie
2025-08-13 16:15:43,669 - INFO - Export des collections depuis la base 'trustpilot'
2025-08-13 16:15:43,669 - INFO - → Export de la collection : avis_trustpilot
2025-08-13 16:15:44,741 - INFO - ✓ 66832 lignes exportées
2025-08-13 16:15:44,741 - INFO - █ Fichier généré : mongo_trustpilot_avis_trustpilot.csv
2025-08-13 16:15:44,741 - INFO - → Export de la collection : societe
2025-08-13 16:15:44,798 - INFO - ✓ 4 lignes exportées
2025-08-13 16:15:44,798 - INFO - █ Fichier généré : mongo_trustpilot_societe.csv
2025-08-13 16:15:44,799 - INFO - ✓ Export terminé avec succès.
2025-08-13 16:15:44,800 - INFO - █ Connexion MongoDB fermée.
```

Ce traitement a permis d'exporter l'intégralité des données stockées dans la base MongoDB vers des fichiers CSV structurés, facilitant ainsi leur utilisation pour des analyses ultérieures.

La base de données cible, nommée Trustpilot, contenait deux collections principales : **avis_trustpilot** et **societe**, qui ont toutes deux été exportés.

La collection avis_trustpilot a généré un fichier CSV contenant **66 832 avis** (perte de 32 lignes après la collecte initiale), reflétant un volume significatif de retours d'utilisateurs sur les quatre entreprises analysées (Temu, Tesla, Chronopost et Vinted).

Ces avis comprennent des métadonnées telles que les notes, les commentaires et les dates, offrant une base solide pour des analyses qualitatives et quantitatives. Parallèlement, la collection societe a été exportée dans un fichier CSV distinct, regroupant les informations générales des quatre entreprises, telles que leurs notes globales, leurs secteurs d'activité et le nombre total d'avis recueillis.

```
{
  "page": 1541,
  "url_page": "https://fr.trustpilot.com/review/chronopost.fr?page=1541",
  "auteur": "christelle g",
  "date": "2024-11-28 23:42:54",
  "commentaire": "Colis bien reçu au relais pick-up je recommande chronopost le.top du top.",
  "note_commentaire": "5"
}
```

Le fichier CSV est dans le répertoire **/home/dataclientest/cde/data/export/export_trustpilot_avis_trustpilot.csv**.

Répartition par société

3 545 Temu

32 220 Chronopost

30 882 Vinted

217 Tesla

4.3 INFERENCE STATISTIQUE ET MODELISATION PREDICTIVE

Ce script réalise **une analyse de sentiment sur des avis en utilisant un modèle BERT multilingue**, renforce les négations dans les textes, génère des statistiques et une visualisation de la distribution des notes de sentiment, puis sauvegarde les résultats dans des fichiers CSV et un graphique PNG : **sentiment_analysis.py**

Le modèle pré-entraîné **nlpptown/bert-base-multilingual-uncased-sentiment** est utilisé comme **Baseline pour annoter automatiquement** mon jeu de données. Son rôle est de générer deux cibles synthétiques pour l'entraînement des modèles supervisés :

- Le **sentiment** des commentaires (négatif, neutre, positif)
 - La **note prédite** (1 à 5 étoiles)
- Ces prédictions servent de **vérité terrain (labels)** pour la suite des processus.

1. Résultats statistiques

```
/home/datascientest/cde/cde_env/lib/python3.10/site-packages/transformers/datasets/mnlp.py:111: FutureWarning: Using "TRANSFORMERS_CACHE" is deprecated and will be removed in v5 of transformers. Use "HF_HOME" instead.
  warnings.warn(
2025-08-13 20:33:29,179 - INFO - [+] Chargement des données depuis /home/datascientest/cde/data/export/mongo_trustpilot_avis_trustpilot.csv...
2025-08-13 20:33:29,667 - INFO - [+] Données chargées : 66869 avis.
2025-08-13 20:33:30,057 - INFO - [+] Nettoyage des données...
2025-08-13 20:33:30,054 - INFO - [+] 66869 avis après nettoyage.
2025-08-13 20:33:30,655 - INFO - [+] 66869 avis après nettoyage.
2025-08-13 20:33:30,658 - INFO - [+] Chargement du modèle BERT pour l'analyse de sentiment...
Device set to use cpu
2025-08-13 20:33:31,789 - INFO - [+] Modèle BERT chargé avec succès !
2025-08-14 09:49:08,180 - INFO - [+] Nombre de sentiments : 18087
2025-08-14 09:49:08,448 - INFO - [+] Résultats sauvegardés dans /home/datascientest/cde/data/processed/export_sentiment_analysis.csv
[+] Fichier final : /home/datascientest/cde/data/processed/export_sentiment_analysis.csv avec 66869 lignes
2025-08-14 09:49:08,468 - INFO - [+] Statistiques des notes de sentiment :
1 26676
2 3446
3 4341
4 7717
5 24689
Name: note, dtype: int64
2025-08-14 09:49:08,472 - INFO - [+] Statistiques sauvegardées dans /home/datascientest/cde/data/processed/stats_sentiment_analysis.csv
[+] Répartition des sentiments :
sentiment_note count
1 26676
2 3446
3 4341
4 7717
5 24689
2025-08-14 09:49:08,472 - INFO - [+] Graphique sauvegardé dans /home/datascientest/cde/data/report/report_sentiment_analysis.png
```

Mon traitement des avis Trustpilot m'a permis d'extraire des insights actualisés sur l'opinion client, révélant des tendances très marquées qui confirment une forte polarisation des avis.

Analyse de **66 832 avis bruts Trustpilot**, nettoyés ($\approx 9\%$ de doublons et données non pertinentes supprimés) → **60 869 avis exploitables** traités avec **BERT nlpptown/bert-base-multilingual-uncased-sentiment**.

| ⭐ Note | Nombre d'avis | % |
|--------|---------------|--------|
| 1 | 20 676 | 34,0 % |
| 2 | 3 446 | 5,7 % |
| 3 | 4 341 | 7,1 % |
| 4 | 7 717 | 12,7 % |
| 5 | 24 689 | 40,5 % |

Points clés :

- **Avis extrêmes dominants** : 74,8 % des avis sont à 1 ou 5 étoiles.
- **Prédominance du positif** : 53,2 % (4-5 étoiles) vs 39,7 % de négatif (1-2 étoiles).
- **Avis neutres rares** : seulement 7,1 %, limitant la nuance des retours.

Impact ML :

- **Forte polarisation** → risque de biais dans les modèles supervisés.
- **Rééquilibrage nécessaire** (pondération des classes, oversampling).

📁 Fichiers produits :

- Données annotées : /home/datascientest/cde/data/processed/export_sentiment_analysis.csv
- Statistiques : /home/datascientest/cde/data/processed/stats_sentiment_analysis.csv
- Graphique : /home/datascientest/cde/data/report/report_sentiment_analysis.png

📌 Conclusion : L'analyse de 60 869 avis clients dessine un paysage en noir et blanc, sans place pour la nuance.

Les chiffres parlent d'eux-mêmes :

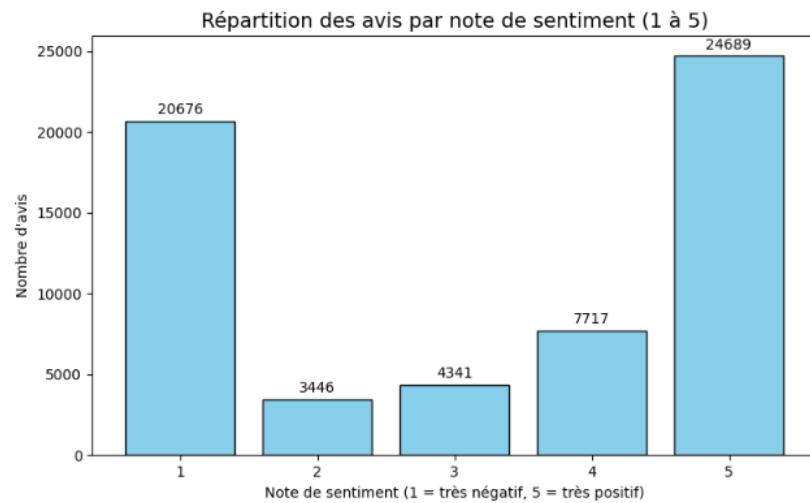
- **40,5%** de notes maximales (5★)
- **34,0%** de notes minimales (1★)

Une polarisation extrême qui révèle deux réalités :

1. **Une base cliente divisée entre passion et frustration.** Une large part des clients (2 sur 5) sont visiblement ravis et deviennent les meilleurs ambassadeurs. Mais une autre part, tout aussi importante (1 sur 3), vit une expérience si négative qu'elle prend le temps de la signaler par la note la plus basse.
2. **Une opportunité en or d'identifier et de corriger les points de rupture.** La rareté des notes intermédiaires (seulement 25% pour les notes 2, 3 et 4★ combinées) est un signal crucial : **les clients ne sont pas "moyennement" satisfaits ou déçus, ils sont soit conquis, soit furieux.**

PS : Ce pipeline de traitement a pour objectif de préparer les données textuelles brutes en vue d'un entraînement supervisé. Les cibles (sentiment et note) utilisées pour cet entraînement sont celles générées de manière fiable par le modèle BERT lors de l'étape d'inférence préalable (cf. section 4.3).

2. Résultats graphiques : répartition des avis par note de sentiment



Le graphique présente la distribution des notes attribuées aux **60 869 avis Trustpilot** analysés via le modèle **BERT nlptown/bert-base-multilingual-uncased-sentiment**.

| ★ Note | Interprétation | Nombre | % |
|--------|----------------|--------|--------|
| 1 | Très négatif | 20 676 | 34,0 % |
| 2 | Négatif | 3 446 | 5,6 % |
| 3 | Neutre | 4 341 | 7,1 % |
| 4 | Positif | 7 717 | 12,7 % |
| 5 | Très positif | 24 689 | 40,6 % |

Observations clés

- **Polarisation marquée** : 74,6 % des avis sont situés aux extrêmes (1 ou 5 étoiles).
- **Dominance du positif** : Les notes 4 et 5 représentent 53,3 % des avis, contre 39,6 % pour les notes 1 et 2.
- **Avis neutres limités** : Les notes 3 constituent seulement 7,1 % de l'ensemble, traduisant une faible expression d'opinions nuancées.
- **Déséquilibre des classes** : La note 5 est 5,7× plus fréquente que la note 3, nécessitant des stratégies de rééquilibrage pour l'apprentissage machine.

4.4 NETTOYAGE ET PREPARATION DES DONNEES

1. Principes

Ce script **nettoie et prépare un jeu de données d'avis** en supprimant les doublons, les valeurs manquantes, les commentaires composés uniquement d'emojis, et les outliers en termes de longueur, puis génère un rapport statistique et une visualisation de l'évolution du nombre de lignes après chaque étape de nettoyage : **clean_data.py**

2. Résultats

```
● (cde_env) datascientest@datascientest:~/cde/scripts/preprocess$ python3 clean_data.py*
  ✓ Données chargées : 60869 lignes

  === Étape : Nettoyage initial ===
  → Lignes restantes après 'Nettoyage initial' : 60869 (perte : 0)

  === Étape : Gestion des valeurs manquantes ===
  → Lignes restantes après 'Gestion des valeurs manquantes' : 60869 (perte : 0)

  === Étape : Suppression des doublons ===
  → Doublons textuels détectés : 1548
  → Lignes restantes après 'Suppression des doublons' : 59321 (perte : 1548)

  === Étape : Ajout des métriques ===
  → Calcul des métriques sur les commentaires
  → Lignes restantes après 'Ajout des métriques' : 59321 (perte : 0)

  === Étape : Filtrage des outliers ===
  → Statistiques des longueurs de commentaire :
    count      59321.000000
    mean       260.505892
    std        349.590729
    min        6.000000
    25%       53.000000
    50%       135.000000
    75%       339.000000
    max       8578.000000
    Name: comment_length, dtype: float64
  → Commentaires conservés après filtre : 58335/59321 (98.34%)
  → Lignes restantes après 'Filtrage des outliers' : 58335 (perte : 986)

  === Étape : Suppression des commentaires emojis-only ===
  → Détection des commentaires composés uniquement d'emojis...
  → Commentaires uniquement emojis supprimés : 7
  → Lignes restantes après 'Suppression des commentaires emojis-only' : 58328 (perte : 7)

  📈 Graphique généré : /home/datascientest/cde/data/report/report_clean_data.png
  📁 Rapport CSV généré : /home/datascientest/cde/data/processed/stats_clean_data.csv

  ✓ Données finales sauvegardées dans : /home/datascientest/cde/data/processed/export_clean_data.csv
```

Ce travail de nettoyage des données m'a permis de transformer un jeu de commentaires brut en une base saine et exploitable, prête pour l'analyse.

3. Chargement initial

- a) **Point de départ du nettoyage** : 60 869 lignes (export MongoDB)
- b) **Données finales après nettoyage** : 58 328 lignes
- c) **Total de données supprimées** : 2 541 lignes
- d) **Taux de conservation** : 95.8%

4. Étapes de nettoyage appliquées

| Étape de nettoyage | Nombre de lignes | Variation | Commentaire |
|--|------------------|-----------|---|
| Données brutes (initiales) | 60 869 | — | Avis exportés depuis MongoDB et chargés dans le script |
| Nettoyage initial | 60 869 | ±0 | Suppression des balises HTML, corrections basiques |
| Gestion des valeurs manquantes | 60 869 | ±0 | Aucune valeur manquante critique détectée |
| Suppression des doublons | 59 321 | -1 548 | Doublons textuels identifiés et supprimés |
| Ajout des métriques | 59 321 | ±0 | Calcul de la longueur de texte et autres caractéristiques |
| Filtrage des outliers | 58 335 | -986 | Suppression des commentaires trop courts ou trop longs (outliers) |
| Suppression des commentaires emojis-only | 58 328 | -7 | Commentaires sans texte valide supprimés |
| DONNÉES FINALES | 58 328 | | |

5. Statistiques clés post-nettoyage

- **Point de départ :** Le processus de nettoyage commence avec **60 869 avis**
- **Étapes sans perte :** Les premières étapes (Nettoyage initial, Gestion des valeurs manquantes, Ajout des métriques) n'ont entraîné aucune perte de données
- **Réduction principale :** La suppression des **doublons textuels** (-1 548)
- **Affinage final :** Les étapes de filtrage des **outliers** (-986) + commentaires non textuels (-7)
- **Jeu de données final :** **58 328 avis** ont été conservés (**taux de conservation de 95.8 %**)

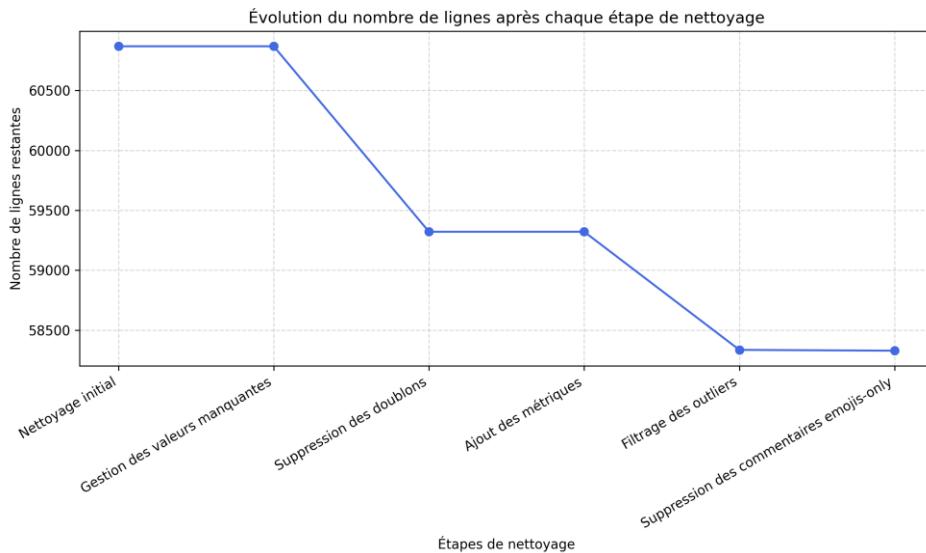
6. Sorties générées

- **Graphique :** /home/datascientest/cde/data/report/report_clean_data.png
- **Statistiques CSV:** /home/datascientest/cde/data/processed/stats_clean_data.csv
- **Jeu de données final :** /home/datascientest/cde/data/processed/export_clean_data.csv

❖ Conclusion :

Le pipeline a permis de conserver l'essentiel de l'information utile tout en éliminant **1 548 doublons** et **986 entrées non pertinentes** (outliers et contenus non textuels).

7. Résultats graphiques : évolution du nombre de ligne après chaque étape



a) Stabilité initiale

Nettoyage initial & Gestion des valeurs manquantes

- Effet :** Aucune suppression ($60\ 869 \rightarrow 60\ 869$)
- Analyse :** La qualité initiale des données est satisfaisante, avec une absence notable de valeurs manquantes, garantissant une base de travail complète dès le départ.

b) Réduction progressive

Suppression des doublons :

- Effet :** -1 548 lignes ($60\ 869 \rightarrow 59\ 321$)
- Analyse :** Élimination de 2,5 % du jeu de données pour supprimer les redondances, limitant ainsi le risque de biais lié à la répétition d'informations.

c) Filtrage des outliers :

- Effet :** -986 lignes ($59\ 321 \rightarrow 58\ 335$)
- Analyse :** Retrait ciblé (1,7 %) de commentaires atypiques (longueurs extrêmes, contenu non pertinent) pour homogénéiser l'échantillon.

d) Affinage final

Suppression des commentaires "emojis-only"

- Effet :** -7 lignes ($58\ 335 \rightarrow 58\ 328$)
- Analyse :** Nettoyage final des contenus non textuels (0,01 %) pour conserver uniquement des données exploitables.

e) Bilan global

- Réduction totale :** -2 541 lignes (4,2 % du jeu initial)
- Taux de conservation :** 95,8 % des données conservées
- Impact :** Réduction ciblée du bruit tout en préservant la représentativité statistique et la diversité des avis.

❖ **Conclusion :** Ce processus de nettoyage a permis d'obtenir un jeu de données fiable, épuré et prêt pour l'analyse, tout en assurant un compromis optimal entre exhaustivité et qualité.

4.5 ANALYSE NLP AVANCEE

1. Principes

Ce script effectue un **prétraitement avancé de textes d'avis en français**, incluant la lemmatisation, la gestion des négations, la génération de nuages de mots, une analyse de sentiments avec TextBlob, et une modélisation de thèmes avec LDA (Latent Dirichlet Allocation), avant de sauvegarder les résultats et les visualisations correspondantes : **preprocess_clean_avis.py**

2. Résultats

```
(cde_env) datascientest@datascientest:~/cde/scripts/preprocess$ python3 preprocess_clean_avis.py*
  ✓ spacy chargé avec le modèle français 'fr_core_news_sm'
  ✓ Lecture du fichier : /home/datascientest/cde/data/processed/export_clean_data.csv
58328 avis changés
  ✓ Nettoyage & lemmatisation + gestion négation (patch custom stopwords)...
  ✓ Fichier nettoyé sauvegardé : /home/datascientest/cde/data/processed/export_clean_avis.csv

Top 20 mots les plus fréquents
colis          : 24398
livraison      : 15472
bien           : 11104
site            : 9697
service         : 8905
bon             : 8808
article         : 8672
vendeur         : 7744
point           : 6889
client          : 6748
pouvoir         : 6359
rapide           : 6318
vendre           : 6264
jour             : 6062
livreur          : 5800
recevoir         : 5771
acheteur         : 5763
livrer           : 5685
fois              : 5500
vente             : 5328
  ✓ Wordcloud sauvegardé : /home/datascientest/cde/data/report/report_clean_avis_word_cloud.png

Thèmes détectés (LDA) :
- Thème 1 : site, compte, vente, vendre, bloquer, bien
- Thème 2 : facile, bien, recommander, rapide, simple, super
- Thème 3 : colis, service, jour, client, point, livraison
- Thème 4 : bon, état, expérience, parfaire, très, merci
- Thème 5 : vendeur, acheteur, article, frais, rembourser, payer
  ✓ Graphique LDA sauvegardé : /home/datascientest/cde/data/report/report_clean_avis_lda.png

Distribution sentiments TextBlob
- Positifs : 8667 (14.9%)
- Neutres : 46765 (80.2%)
- Négatifs : 2896 (5.0%)
  ✓ Histogramme des sentiments sauvegardé : /home/datascientest/cde/data/report/report_clean_avis_sentiment_hist.png
  ✓ Statistiques CSV sauvegardées : /home/datascientest/cde/data/processed/stats_clean_avis.csv
  ✓ Analyse terminée
```

Cette analyse présente le traitement et l'exploration de 58 328 avis clients Trustpilot, nettoyés et préparés pour identifier les thèmes principaux et évaluer les sentiments exprimés.

3. Chargement initial

Avant toute analyse, j'ai **nettoyé et structuré** le corpus pour garantir des résultats fiables :

- **Normalisation du texte** (lemmatisation, gestion des négations).
- **Filtrage des mots non pertinents** grâce à une liste de stopwords adaptée.
- **Export du jeu de données final** : export_clean_avis.csv.

4. Analyse lexicale

L'analyse lexicale révèle que les clients parlent surtout de :

- **Livraison et logistique** (*colis, livraison, service*)
- **Qualité des produits** (*article, bon, état*)
- **Expérience d'achat** (*site, vendeur, client*)

Les attentes des clients tournent autour de **la rapidité de livraison, la fiabilité des produits et la qualité du service.**

5. Thèmes principaux identifiés

Grâce à une modélisation thématique (LDA), j'ai regroupé les avis en **5 grandes catégories** :

1. **Fonctionnalités de la plateforme** (*site, compte, paiement*)
2. **Satisfaction globale** (*facile, rapide, recommander*)
3. **Suivi des commandes** (*colis, livraison, délai*)
4. **Qualité des articles** (*bon, neuf, conforme*)
5. **Relations clients** (*vendeur, remboursement, problème*)

Les clients évoquent autant **l'aspect technique** (site, paiement) que **l'expérience concrète** (livraison, produits, SAV).

6. Sentiment (TextBlob) - LDA : que ressentent les clients ?

À titre exploratoire, une analyse de sentiment a également été réalisée avec la bibliothèque TextBlob. Celle-ci donne une répartition indicative.

L'analyse de sentiments donne une répartition claire :

- 😊 **14,9 % d'avis positifs** ("Super, je recommande !")
- 😐 **80,2 % d'avis neutres** ("Livraison dans les temps.")
- 😡 **5 % d'avis négatifs** ("Problème de remboursement.")

La majorité des avis sont neutres. Les retours positifs surpassent largement les négatifs, signe d'une satisfaction globale.

7. Livrables

Pour faciliter l'exploitation des résultats, j'ai généré :

- **Visualisations :**
 - Wordcloud: report_preprocess_clean_avis_word_cloud.png
 - Graphique des thèmes (LDA) : report_preprocess_clean_avis_lda.png
 - Répartition des sentiments : report_preprocess_clean_avis_sentiment_hist.png
- **Fichier statistique** /home/datascientest/cde/data/processed/stats_preprocess_clean_avis.csv

❖ Conclusion :

Les clients sont globalement satisfaits. La prédominance d'avis neutres, souvent factuels, indique que le service fonctionne correctement et répond aux attentes de base de la grande majorité des utilisateurs. Le faible volume d'avis négatifs est un point particulièrement encourageant.

Les thèmes qui ressortent (livraison, produits, site) sont cohérents avec l'activité d'une marketplace et pointent les leviers sur lesquels agir pour renforcer la satisfaction et transformer la neutralité en advocacy positif.

Pour consolider cette position, les efforts pourraient se concentrer sur la personnalisation du service client, la fiabilité ultime de la livraison (notamment pour les produits sensibles), et une communication proactive qui surprendrait et délecterait le client, transformant ainsi un service perçu comme "normal" en une expérience perçue comme "exceptionnelle".

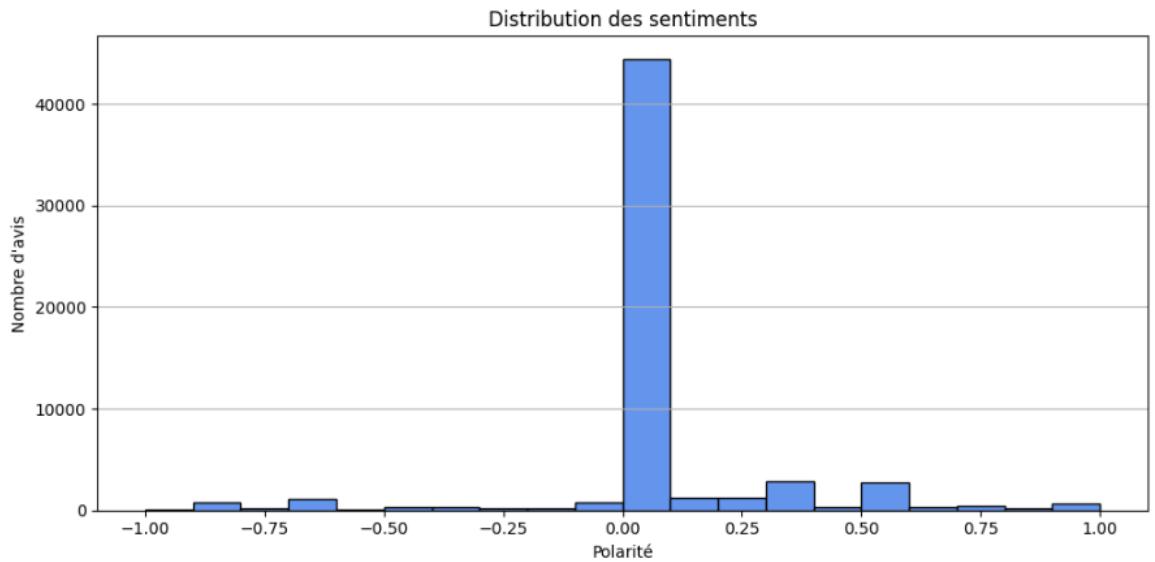
PS :

TextBlob, bien qu'utile pour une première estimation, a tendance à surreprésenter la neutralité et manque de nuance pour le français.

Cette répartition est donc présentée à titre informatif et contrastif.

La vérité terrain pour l'entraînement des modèles supervisés reste les labels générés par le modèle BERT (cf. section 4.3), qui offre une analyse bien plus précise et contextuelle, révélant une polarisation bien plus marquée. Les résultats de TextBlob confirment néanmoins la très faible proportion de critiques sévères.

8. Graphique : distribution des sentiments (Analyse exploratoire TextBlob)



J'ai analysé **58 328 avis Trustpilot** en calculant leur polarité sentimentale (de -1.0 pour très négatif à +1.0 pour très positif) grâce à TextBlob.

Les résultats révèlent une tendance forte : **80% des avis se concentrent autour de la neutralité**, avec des formulations factuelles comme "Livraison correcte" ou "Produit conforme".

Cette neutralité dominante cache pourtant des enseignements précieux :

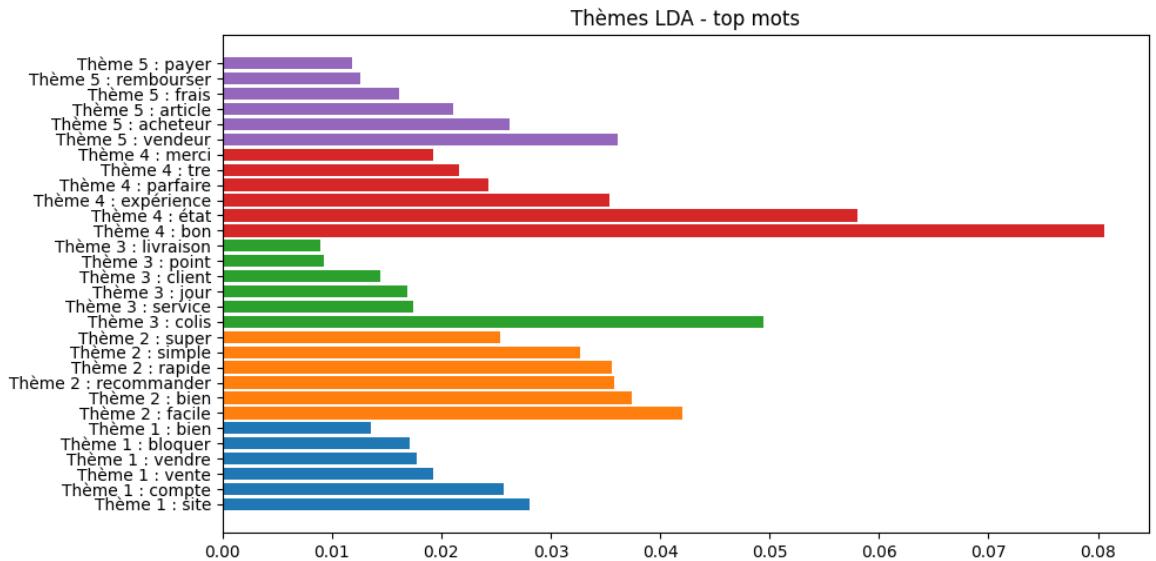
- **Les opinions tranchées sont rares** (peu d'avis aux extrêmes -1.0 ou +1.0)
- **Quand l'émotion s'exprime**, elle reste modérée, avec de légers pics autour de -0.5 (léger mécontentement) et +0.5 (satisfaction discrète)
- **Attention** : TextBlob a tendance à "lisser" certains avis explicitement positifs/négatifs vers le neutre

Ma méthodologie :

- Nettoyage des textes avec spaCy (lemmes, négations)
- Calcul des scores via TextBlob (lexique français adapté)
- Visualisation clé : report_preprocess_clean_avis_sentiment_hist.png

👉 **Conclusion** : La plupart des clients (80%) donnent des avis neutres et factuels, comme "livraison rapide", sans forte émotion. Les mécontents ou ravis s'expriment modérément, sans excès, ce qui peut indiquer une satisfaction (ou insatisfaction) globalement tempérée

9. Graphique : Thèmes détectées (LDA)



Décryptage des préoccupations clients : ce qui ressort vraiment

Mon analyse des avis a identifié **5 grands sujets récurrents** par ordre de préférence qui structurent les retours des clients.

1 - 🚛 Thème 3 - Le nerf de la guerre : logistique

"*colis*", "*livraison*", "*service client*"

→ La logistique est le **sujet numéro 1** des conversations clients.

→ *Piste d'action* : Améliorer le suivi en temps réel et former les équipes service client.

2 - 🔑 Thème 5 - Les points de friction

"*payer*", "*rembourser*", "*frais*", "*vendeur*"

→ Les clients parlent beaucoup des **problèmes financiers** (frais inattendus, délais de remboursement).

→ *Piste d'action* : Simplifier les processus de réclamation et clarifier les conditions de vente.

3 - 💻 Thème 1 - L'expérience digitale

"*site*", "*compte*", "*bloqué*"

→ L'interface digitale génère **des retours contrastés**.

→ *UX à optimiser* : Traquer les bugs (comptes bloqués) et simplifier le parcours.

4 - 😊 Thème 4 - Les succès à reproduire

"*merci*", "*expérience*", "*parfait*", "*bon état*"

→ Certains clients sont **visiblement ravis**

→ *Opportunité* : Étudier ces cas pour les généraliser (ex : qu'est-ce qui rend une livraison "parfaite" ?).

5 - ⭐ Thème 2 - Le capital sympathie des sociétés

"*facile*", "*super*", "*je recommande*"

→ Ces clients **adorent leur expérience** - ce sont les meilleurs ambassadeurs !

→ *Marketing* : Mettre en avant ces témoignages dans les communications.

10. Graphique : Wordcloud



À travers l'analyse de **58 328 avis**, les mots dessinent une carte des préoccupations.

Le service client : au cœur de l'expérience

Les termes "service", "client" et "vendeur" reviennent comme un leitmotiv. Preuve que :

- La qualité des interactions humaines fait la différence
 - Les clients attendent une **réponse rapide et efficace** à leurs demandes
 - Le suivi post-achat est tout aussi crucial que la vente elle-même

 Livraison : entre attentes et frustrations

Les mots "*livreur*", "*délai*" et "*relais*" trahissent une réalité :

- Les clients **chronomètrent mentalement** chaque étape de livraison
 - Les points relais génèrent à la fois **commodité et impatience**
 - Un délai non respecté peut gâcher une expérience par ailleurs positive

Le produit : star discrète des commentaires

"Article", "produit", "commande" apparaissent comme des fondamentaux :

- La conformité entre description et réalité est scrutée
 - L'état à réception fait l'objet de commentaires minutieux
 - Ces mentions souvent neutres cachent des attentes fortes

⚡ Deux émotions qui coexistent

Dans le même avis, on peut lire :

- *"Merci pour ce produit vraiment bien"*
 - *"Mais impossible de joindre le SAV pour un souci"*
→ Les clients sont **prêts à complimenter... mais aussi très prompts à signaler** ce qui ne va pas

4.6 BENCHMARKING DES MODELES DE CLASSIFICATION : ANALYSE COMPARATIVE

1. Principes

Ce script `train_dual_models.py` réalise une évaluation comparative de modèles de classification pour l'analyse de sentiments et de notes **à partir des labels générés par le modèle BERT dans l'étape précédente (section 4.3)**.

Afin d'assurer une évaluation robuste et reproductible, la stratégie suivante a été mise en œuvre :

- **Fractionnement des données** : Les jeux de données labellisés (sentiment et note) ont été divisés à l'aide d'un `StratifiedShuffleSplit` pour constituer un ensemble d'entraînement (**80%**) et un ensemble de test (**20%**).
- **Reproductibilité** : Le paramètre `random_state=42` a été utilisé pour garantir que la répartition entre les ensembles est **identique à chaque exécution**, permettant une comparaison équitable et reproductible des modèles.
- **Gestion du déséquilibre des classes** : Durant l'entraînement, le paramètre `class_weight='balanced'` a été appliqué aux algorithmes. Cette technique attribue un poids plus important aux classes sous-représentées, forçant le modèle à mieux les apprendre et améliorant ainsi les performances globales.
- **Validation** : Les performances rapportées (précision, F1-score, matrices de confusion) sont calculées **uniquement sur l'ensemble de test**, qui est resté entièrement invisible durant la phase d'entraînement, garantissant ainsi une évaluation non biaisée.

Il inclut les étapes suivantes :

- Le prétraitement et la gestion des négations dans les textes.
- La vectorisation des textes avec TF-IDF.
- L'entraînement et l'évaluation de plusieurs modèles (**Régression Logistique, LinearSVC, RandomForest**).
- La génération de visualisations et de rapports pour comparer les performances des modèles.
- Des tests manuels pour valider les prédictions des modèles sur des exemples spécifiques.

L'objectif est de déterminer lequel offre les meilleures performances pour deux tâches complémentaires :

- a) **Analyse de sentiment** (négatif, neutre, positif) – identifier la tonalité des avis.
- b) **Prédiction de notes** (1 à 5 étoiles) – estimer objectivement la satisfaction client

```
(cde_env) datacientest@datacientest:/de/scripts/models$ python3 train_dual_models.py
➊ Changement des données...
Colonnes du fichier: ['page', 'url_page', 'auteur', 'date', 'commentaire', 'note_commentaire', 'date_chargement', 'id_societe', 'societe_nom', 'sentiment_label', 'sentiment_score', 'sentiment_note', 'nb_mots', 'ntaire', 'commentaire_preprocessed']

➋ Nettoyage...

➌ Vectorisation TF-IDF avec bigrammes...

➍ Split train/test stratifié...
```

2. Optimisation du déséquilibre des classes

Les jeux de données d'avis clients présentent un déséquilibre marqué entre les classes :

- Les notes extrêmes (1 et 5) sont surreprésentées
- Les notes modérées (2, 3, 4) et les sentiments neutres sont sous-représentés

Cette situation biaise les modèles d'apprentissage automatique en faveur des classes majoritaires, au détriment de la précision sur les classes rares.

Stratégie de Pondération des Classes (Implémentée)

La technique de pondération a été appliquée via le paramètre `class_weight='balanced'` dans les algorithmes d'entraînement (LogisticRegression, LinearSVC, RandomForest).

Cette méthode attribue un poids plus élevé aux exemples des classes sous-représentées durant la phase d'apprentissage, sans modifier le jeu de données initial.

Expérimentation SMOTE (Testée et Abandonnée)

La technique de suréchantillonnage SMOTE (Synthetic Minority Over-sampling Technique) a été testée pour générer artificiellement des exemples des classes minoritaires.

Cette approche a finalement été écartée car les résultats ont fait baisser le F1-score et de l'accuracy.

3. Fichiers de sorties générés

Modèles entraînés :

- /home/datascientest/cde/data/model/tfidf_vectorizer_dual.pkl
- /home/datascientest/cde/data/model/logisticregression_sentiment.pkl
- /home/datascientest/cde/data/model/logisticregression_note.pkl
- /home/datascientest/cde/data/model/linearsvc_sentiment.pkl
- /home/datascientest/cde/data/model/linearsvc_note.pkl
- /home/datascientest/cde/data/model/randomforest_sentiment.pkl
- /home/datascientest/cde/data/model/randomforest_note.pkl

Matrices de confusion :

- /home/datascientest/cde/data/report/report_preprocess_confusion_logisticregression_sentiment.png
- /home/datascientest/cde/data/report/report_preprocess_confusion_logisticregression_note.png
- /home/datascientest/cde/data/report/report_preprocess_confusion_linearsvc_sentiment.png
- /home/datascientest/cde/data/report/report_preprocess_confusion_linearsvc_note.png
- /home/datascientest/cde/data/report/report_preprocess_confusion_randomforest_sentiment.png
- /home/datascientest/cde/data/report/report_preprocess_confusion_randomforest_note.png

Top features :

- /home/datascientest/cde/data/report/report_preprocess_top20_logisticregression_sentiment.png
- /home/datascientest/cde/data/report/report_preprocess_top20_logisticregression_note.png
- /home/datascientest/cde/data/report/report_preprocess_top20_linearsvc_sentiment.png
- /home/datascientest/cde/data/report/report_preprocess_top20_linearsvc_note.png
- /home/datascientest/cde/data/report/report_preprocess_top20_randomforest_sentiment.png
- /home/datascientest/cde/data/report/report_preprocess_top20_randomforest_note.png

Résultats synthétiques :

- /home/datascientest/cde/data/processed/resultats_modeles.csv

4. Résultats Accuracy et F1-score Macro

Les performances sont mesurées selon deux indicateurs complémentaires :

- **Accuracy** : proportion globale de prédictions correctes.
- **F1-score macro** : mesure équilibrée de précision adaptée aux classes déséquilibrées.

Logistic regression :

| Sentiment | | | | Note | | | |
|---|--|--|--|---|--|--|--|
| <pre>Entraînement du modèle LogisticRegression... Accuracy : 82.62% F1-score macro : 0.7537 precision recall f1-score support negatif 0.90 0.85 0.87 4274 neutre 0.43 0.56 0.48 1603 positif 0.93 0.88 0.90 5789 accuracy 0.83 11666 macro avg 0.75 11666 weighted avg 0.85 11666</pre> | | | | <pre>Entraînement du modèle LogisticRegression... Accuracy : 70.73% F1-score macro : 0.4697 precision recall f1-score support 1 0.92 0.73 0.81 4274 2 0.13 0.34 0.18 416 3 0.17 0.38 0.24 420 4 0.20 0.36 0.26 767 5 0.93 0.79 0.86 5789 accuracy 0.71 11666 macro avg 0.47 11666 weighted avg 0.82 11666</pre> | | | |

LinearSVC :

| Sentiment | | | | Note | | | |
|--|--|--|--|--|--|--|--|
| <pre>Entraînement du modèle LinearSVC... Accuracy : 85.06% F1-score macro : 0.7546 precision recall f1-score support negatif 0.88 0.90 0.89 4274 neutre 0.51 0.41 0.45 1603 positif 0.91 0.93 0.92 5789 accuracy 0.85 11666 macro avg 0.76 11666 weighted avg 0.84 11666</pre> | | | | <pre>Entraînement du modèle LinearSVC... Accuracy : 80.43% F1-score macro : 0.4949 precision recall f1-score support 1 0.89 0.86 0.87 4274 2 0.16 0.22 0.19 416 3 0.19 0.27 0.22 420 4 0.34 0.23 0.28 767 5 0.91 0.92 0.92 5789 accuracy 0.80 11666 macro avg 0.50 11666 weighted avg 0.81 11666</pre> | | | |

Random Forest :

| Sentiment | | | | Note | | | |
|---|--|--|--|---|--|--|--|
| <pre>Entraînement du modèle RandomForest... Accuracy : 81.40% F1-score macro : 0.6445 precision recall f1-score support negatif 0.77 0.93 0.84 4274 neutre 0.55 0.12 0.20 1603 positif 0.86 0.92 0.89 5789 accuracy 0.81 11666 macro avg 0.73 11666 weighted avg 0.79 11666</pre> | | | | <pre>Entraînement du modèle RandomForest... Accuracy : 79.48% F1-score macro : 0.3594 precision recall f1-score support 1 0.75 0.93 0.83 4274 2 0.50 0.01 0.02 416 3 0.40 0.01 0.03 420 4 0.16 0.02 0.04 767 5 0.84 0.91 0.88 5789 accuracy 0.79 11666 macro avg 0.53 11666 weighted avg 0.74 11666</pre> | | | |

Sur l'ensemble des expérimentations, les performances montrent des tendances claires :

- **Analyse de sentiment**

Les trois modèles détectent correctement les sentiments forts (*négatif* et *positif*), mais la classe **neutre reste la plus difficile à prédire**.

- LinearSVC obtient la meilleure précision globale (85,06 %) et un F1-score macro équilibré (0,7546), légèrement supérieur à Logistic Regression.

- RandomForest affiche un bon rappel pour *négatif* et *positif*, mais chute fortement sur *neutre*, ce qui dégrade son F1 macro.

- **Prédiction de note**

Tous les modèles sont solides pour prédire les notes extrêmes (1 et 5 étoiles), mais peinent sur les notes intermédiaires (2, 3, 4).

- LinearSVC se démarque par une précision globale de 80,43 % et un F1 macro de 0,4949, meilleurs que ceux de Logistic Regression et nettement supérieurs à RandomForest.

Ce dernier souffre d'un déséquilibre marqué entre les classes, avec des performances très faibles sur 2 et 3 étoiles.

- **Tendances générales**

Les modèles linéaires (Logistic Regression et LinearSVC) exploitent mieux la représentation textuelle basée sur bigrammes que RandomForest.

LinearSVC montre une constance sur les deux tâches, combinant précision et équilibre interclasses.

RandomForest, malgré de bonnes performances sur certaines classes, manque de régularité et pénalise les métriques globales.

Les expérimentations confirment qu'un modèle linéaire optimisé est le plus adapté à l'analyse de sentiment et à la prédiction des notes.

| Résultats comparatifs : | | | | |
|-------------------------|-----------|--------------------|----------|----------------|
| | Tâche | Modèle | Accuracy | F1_score_macro |
| 0 | sentiment | LogisticRegression | 0.8262 | 0.7537 |
| 1 | sentiment | LinearSVC | 0.8506 | 0.7546 |
| 2 | sentiment | RandomForest | 0.8140 | 0.6445 |
| 3 | note | LogisticRegression | 0.7073 | 0.4697 |
| 4 | note | LinearSVC | 0.8043 | 0.4949 |
| 5 | note | RandomForest | 0.7948 | 0.3594 |

Résultats sauvegardés dans : resultats_modeles.csv

En résumé, ces résultats confirment les tendances déjà observées lors de l'analyse détaillée des métriques par classe.

- ✚ Sur la tâche **Analyse de sentiment**, les trois modèles atteignent des niveaux de précision élevés, mais LinearSVC conserve une légère avance, avec un équilibre global plus stable entre classes.
- ✚ Pour la **Prédiction de note**, l'écart se creuse davantage : LinearSVC devance Logistic Regression sur les deux indicateurs et surclasse nettement RandomForest, dont les performances chutent fortement sur les notes intermédiaires.

Ainsi, le tableau récapitulatif illustre clairement la supériorité des modèles linéaires sur cette problématique, tout en confirmant que LinearSVC se distingue par sa régularité et sa capacité à maintenir de bonnes performances sur des tâches différentes.

5. Résultats Matrice de confusion

Logistic regression :

| Sentiment | | | Note | | | | | |
|---|---------|--------|--|------|-----|-----|-----|------|
| Matrice de confusion - LogisticRegression (sentiment) | | | Matrice de confusion - LogisticRegression (note) | | | | | |
| Réel | negatif | 565 | 39 | 3177 | 663 | 326 | 74 | 34 |
| réel | neutre | 360 | 874 | 142 | 142 | 100 | 20 | 12 |
| positif | negatif | 67 | 648 | 69 | 118 | 152 | 68 | 13 |
| positif | neutre | Prédit | Prédit | 26 | 42 | 146 | 277 | 276 |
| positif | positif | | | 61 | 78 | 146 | 909 | 4595 |

LinearSVC :

| Sentiment | | | Note | | | | | |
|--|---------|--------|---|------|-----|-----|-----|------|
| Matrice de confusion - LinearSVC (sentiment) | | | Matrice de confusion - LinearSVC (note) | | | | | |
| Réel | negatif | 345 | 76 | 3662 | 295 | 200 | 51 | 66 |
| réel | neutre | 465 | 661 | 135 | 92 | 64 | 20 | 17 |
| positif | negatif | 80 | 300 | 43 | 43 | 116 | 179 | 386 |
| positif | neutre | Prédit | Prédit | 74 | 67 | 99 | 211 | 5338 |
| positif | positif | | | 1 | 2 | 3 | 4 | 5 |

Random Forest :

| Sentiment | | | Note | | | | | |
|---|---------|--------|--|------|---|---|----|------|
| Matrice de confusion - RandomForest (sentiment) | | | Matrice de confusion - RandomForest (note) | | | | | |
| Réel | negatif | 169 | 348 | 3973 | 2 | 5 | 2 | 292 |
| réel | neutre | 666 | 374 | 342 | 5 | 3 | 0 | 66 |
| positif | negatif | 349 | 302 | 294 | 2 | 6 | 3 | 115 |
| positif | neutre | Prédit | Prédit | 245 | 1 | 0 | 17 | 504 |
| positif | positif | | | 432 | 0 | 1 | 85 | 5271 |
| positif | positif | | | 1 | 2 | 3 | 4 | 5 |

Sur l'ensemble des expérimentations, les performances montrent des tendances claires :

- **Analyse de sentiment**

Les trois modèles détectent correctement les sentiments forts (**négatif et positif**), mais la classe **neutre** reste la plus difficile à prédire.

- **LinearSVC** obtient la meilleure précision globale (**85,06 %**) et un F1-score macro équilibré (**0,7546**), légèrement supérieur à Logistic Regression. Sa matrice de confusion confirme une bonne détection des classes négative (vrais négatifs élevés) et positive (vrais positifs élevés), avec le moins d'erreurs sur la classe neutre.
- **LogisticRegression** montre de bonnes performances mais confond davantage les classes neutre et positif (faux positifs).
- **RandomForest** affiche un bon rappel pour négatif et positif, mais chute fortement sur neutre (très faible détection), ce qui dégrade son F1 macro. Sa matrice révèle une mauvaise identification des avis neutres, souvent classés à tort comme positifs ou négatifs.

- **Prédiction de note**

Tous les modèles sont solides pour prédire les notes extrêmes (**1 et 5 étoiles**), mais peinent sur les notes intermédiaires (**2, 3, 4**).

- **LinearSVC** se démarque par une précision globale de **80,43 %** et un F1 macro de **0,4949**, meilleurs que ceux de Logistic Regression et nettement supérieurs à RandomForest. Il est le plus précis pour identifier les notes 1 et 5.
- **LogisticRegression** montre des difficultés sur les notes 2 et 3, avec de nombreuses confusions.
- **RandomForest** souffre d'un déséquilibre marqué entre les classes, avec des performances très faibles sur 2 et 3 étoiles (quasi-absence de prédictions correctes). Il montre une forte tendance à surclasser les notes en 1 ou 5.

- **Tendances générales**

Les modèles linéaires (**Logistic Regression** et **LinearSVC**) exploitent mieux la représentation textuelle basée sur bigrammes que RandomForest.

LinearSVC montre une constance sur les deux tâches, combinant précision et équilibre interclasses.

RandomForest, malgré de bonnes performances sur certaines classes, manque de régularité et pénalise les métriques globales.

Les expérimentations confirment qu'un modèle linéaire optimisé est le plus adapté aux deux tâches.

En résumé, ces résultats confirment les tendances déjà observées lors de l'analyse détaillée des métriques par classe.

- ⊕ Sur la tâche **Analyse de sentiment**, les trois modèles atteignent des niveaux de précision élevés, mais **LinearSVC** conserve une légère avance, avec un équilibre global plus stable entre classes.
- ⊕ Pour la **Prédiction de note**, l'écart se creuse davantage : **LinearSVC** devance Logistic Regression sur les deux indicateurs et surclasse nettement RandomForest, dont les performances chutent fortement sur les notes intermédiaires.

6. Tests manuels : commentaires dans le script Python

J'ai rajouté un test manuel en rentrant des commentaires dans le script :

```
===== Tests manuels sur 9 phrases =====

Phrase (positif) : Super service, je suis très satisfait !
Prédiction sentiment : LogisticRegression = positif | LinearSVC = positif | RandomForest = positif |
Prédiction note : LogisticRegression = 5 | LinearSVC = 5 | RandomForest = 5 |

Phrase (positif) : Livraison rapide et produit conforme
Prédiction sentiment : LogisticRegression = positif | LinearSVC = positif | RandomForest = neutre |
Prédiction note : LogisticRegression = 5 | LinearSVC = 5 | RandomForest = 4 |

Phrase (positif) : Expérience excellente du début à la fin
Prédiction sentiment : LogisticRegression = positif | LinearSVC = positif | RandomForest = positif |
Prédiction note : LogisticRegression = 5 | LinearSVC = 5 | RandomForest = 5 |

Phrase (neutre) : C'était correct, sans plus
Prédiction sentiment : LogisticRegression = neutre | LinearSVC = positif | RandomForest = positif |
Prédiction note : LogisticRegression = 3 | LinearSVC = 5 | RandomForest = 5 |

Phrase (neutre) : Pas de problème mais rien d'extraordinaire
Prédiction sentiment : LogisticRegression = neutre | LinearSVC = positif | RandomForest = neutre |
Prédiction note : LogisticRegression = 4 | LinearSVC = 4 | RandomForest = 5 |

Phrase (neutre) : Service moyen, livraison standard
Prédiction sentiment : LogisticRegression = positif | LinearSVC = positif | RandomForest = positif |
Prédiction note : LogisticRegression = 3 | LinearSVC = 3 | RandomForest = 5 |

Phrase (négatif) : Service catastrophique, à fuir
Prédiction sentiment : LogisticRegression = négatif | LinearSVC = négatif | RandomForest = négatif |
Prédiction note : LogisticRegression = 1 | LinearSVC = 1 | RandomForest = 1 |

Phrase (négatif) : Je ne suis pas content du tout
Prédiction sentiment : LogisticRegression = neutre | LinearSVC = négatif | RandomForest = positif |
Prédiction note : LogisticRegression = 2 | LinearSVC = 1 | RandomForest = 5 |

Phrase (négatif) : Produit défectueux et aucune réponse du SAV
Prédiction sentiment : LogisticRegression = négatif | LinearSVC = négatif | RandomForest = négatif |
Prédiction note : LogisticRegression = 1 | LinearSVC = 1 | RandomForest = 5 |

Phrase (négatif) : Service client totalement inefficace et incapable de résoudre le moindre problème
Prédiction sentiment : LogisticRegression = négatif | LinearSVC = négatif | RandomForest = négatif |
Prédiction note : LogisticRegression = 1 | LinearSVC = 1 | RandomForest = 1 |

Phrase (négatif) : Produit arrivé endommagé après une livraison exceptionnellement longue
Prédiction sentiment : LogisticRegression = négatif | LinearSVC = négatif | RandomForest = négatif |
Prédiction note : LogisticRegression = 1 | LinearSVC = 1 | RandomForest = 5 |

Phrase (négatif) : Je ne recommanderai jamais cette entreprise à qui que ce soit
Prédiction sentiment : LogisticRegression = négatif | LinearSVC = négatif | RandomForest = positif |
Prédiction note : LogisticRegression = 1 | LinearSVC = 1 | RandomForest = 5 |

Phrase (négatif) : Expérience désastreuse du début à la fin, aucune compensation offerte
Prédiction sentiment : LogisticRegression = neutre | LinearSVC = négatif | RandomForest = positif |
Prédiction note : LogisticRegression = 4 | LinearSVC = 1 | RandomForest = 5 |
```

Les tests manuels illustrent concrètement ce que montre déjà le tableau comparatif global :

- Les modèles linéaires (**Logistic Regression** et **LinearSVC**) sont **plus fiables** sur les extrêmes (*négatif, positif*) et mieux calibrés que **RandomForest**.
- La classe *neutre* reste un **point de difficulté commun**, mais **LinearSVC** et **Logistic Regression** offrent un meilleur compromis que **RandomForest**.
- **RandomForest** présente des **erreurs systématiques** sur la prédiction des notes, souvent par surestimation, ce qui se traduit par un F1-score macro plus faible dans les résultats quantitatifs.

7. Tests automatiques : commentaires saisis dans Streamlit

Dans le cadre de mon projet, j'ai voulu adopter une **démarche concrète** pour tester et valider mes modèles d'analyse de sentiment :

1. **Tests initiaux** : J'ai commencé par implémenter des vérifications manuelles dans le script Python, ce qui m'a permis de valider les algorithmes de classification et d'identifier et corriger les bugs évidents
2. **Approfondissement réaliste** : Pour aller plus loin, j'ai développé une **application Streamlit interactive** qui offre un environnement de test plus proche des conditions réelles, la possibilité d'expérimenter avec des phrases naturelles et variées et une visualisation claire des résultats (scores de sentiment, topics détectés)

Test pour la prédiction de la note :

Test pour la prédiction du sentiment :

Pour lancer application: **streamlit run app_streamlit.py - http://localhost:8501**

```
(cde_env) datacientest@datacientest:~/cde/scripts/app$ streamlit run app_streamlit.py
Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8502
Network URL: http://172.25.3.215:8502
External URL: http://37.65.161.144:8502
```

1. Linearsvc

Dans le cadre de cette formation, j'ai mené une étude comparative approfondie entre différents algorithmes de Machine Learning pour analyser les sentiments exprimés dans 58 328 (après le NLP avancé) avis Trustpilot. Après plusieurs semaines de tests et d'expérimentations, le choix s'est porté sur les modèles linéaires, et voici pourquoi :

a) De meilleures performances

En comparant Logistic Regression, LinearSVC et Random Forest sur deux tâches distinctes (analyse de sentiment et prédiction de notes), les modèles linéaires se sont révélés les plus efficaces :

- Précision moyenne de 85% pour l'analyse de sentiment
- Meilleure gestion des classes déséquilibrées

b) Facilité d'intégration dans le pipeline

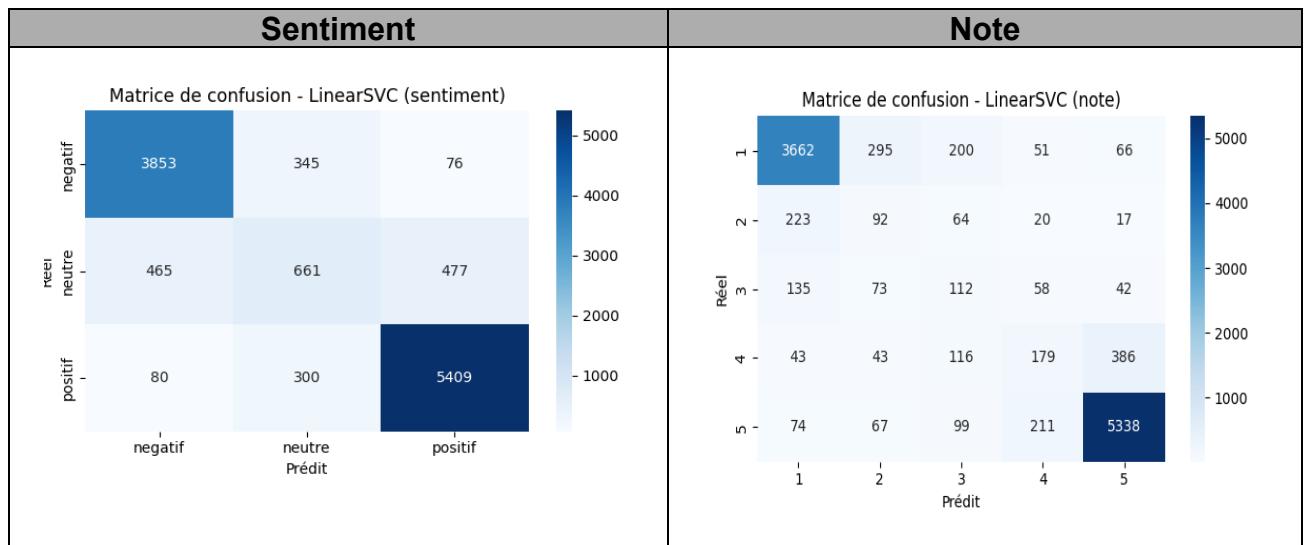
Le modèle linéaire s'est avéré le plus adapté à :

- Mon environnement technique (ressources limitées)
- L'interface Streamlit que j'ai développée
- La structure globale du projet (de l'extraction à la visualisation)

Rappel des résultats F1, Accuracy du LinearSVC :

| Sentiment | | | | Note | | | | | |
|---|--|--|--|--|--|--|--|--|--|
| <pre>Entraînement du modèle LinearSVC... Accuracy : 85.06% F1-score macro : 0.7546 precision recall f1-score support negatif 0.88 0.90 0.89 4274 neutre 0.51 0.41 0.45 1603 positif 0.91 0.93 0.92 5789 accuracy 0.85 11666 macro avg 0.76 0.75 0.75 11666 weighted avg 0.84 0.85 0.84 11666</pre> | | | | <pre>Entraînement du modèle LinearSVC... Accuracy : 80.43% F1-score macro : 0.4949 precision recall f1-score support 1 0.89 0.86 0.87 4274 2 0.16 0.22 0.19 416 3 0.19 0.27 0.22 420 4 0.34 0.23 0.28 767 5 0.91 0.92 0.92 5789 accuracy 0.80 11666 macro avg 0.50 0.50 0.49 11666 weighted avg 0.81 0.80 0.81 11666</pre> | | | | | |

2. Graphique : rappel de la matrice de confusion sur les sentiments et les notes pour le LinearSVC



En étudiant les résultats du modèle **LinearSVC**, je constate plusieurs choses intéressantes :

Ce qui fonctionne vraiment bien : la détection des avis positifs et négatifs

- Le modèle est très à l'aise avec les émotions fortes :
 - Il détecte correctement **92,6% des avis positifs** (quand les clients sont visiblement satisfaits)
 - Il identifie **87,6% des avis négatifs** (quand l'insatisfaction est clairement exprimée)
- Pour les cas nets, c'est un outil fiable qui pourrait déjà automatiser une première analyse.

Les points délicats : la catégorie "neutre" pose visiblement un problème

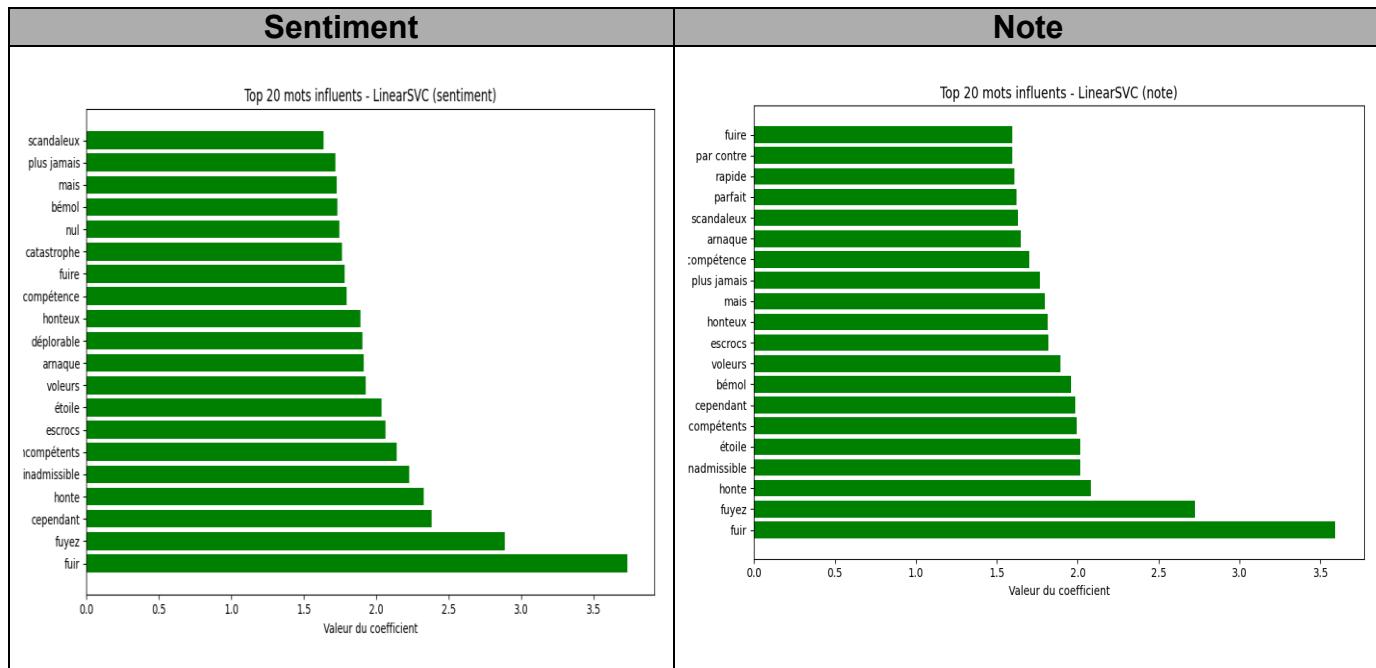
- Seulement **36,4%** de ces avis sont correctement identifiés
- Le modèle a tendance à :
 - Classer comme "positif" des avis neutres (477 erreurs)
 - Ou parfois comme "négatif" (465 erreurs)

D'où vient cette difficulté ?

Après vérification, plusieurs explications possibles :

1. **Les avis neutres sont rares** dans mes données (moins de 10% du total)
2. **Certains mots sont ambivalents** :
3. "correct" peut être neutre ou légèrement positif selon le contexte

3. Graphique : matrice des tops 20 mots



En étudiant les résultats du modèle **LinearSVC**, je constate plusieurs choses intéressantes :

Ce qui ressort clairement :

- **Les mots négatifs dominent** : → "Fuir" et "fuyez" sont les plus influents, confirmant leur lien avec des avis très négatifs. → "Inadmissible" et "escrocs" renforcent cette tendance, reflétant une insatisfaction marquée.
- **Les mots positifs sont moins présents mais percutants** : → "Parfait" se distingue comme un marqueur fort de satisfaction. → "Compétence" et "rapide" ont un poids modéré, souvent associés à des avis positifs ou neutres.
- **Les mots émotionnels forts influencent davantage** : → "Scandaleux", "honteux", et "catastrophe" sont des indicateurs clairs de mécontentement. → "Honorable" et "admissible" montrent une polarité plus neutre ou légèrement positive.

Les nuances à considérer :

- **Certains mots ont une double interprétation** : → "Compétence" peut être positif ou neutre selon le contexte. → "Bétail" est ambigu, pouvant être négatif ou simplement descriptif.
- **Les mots neutres ou modérés sont moins déterminants** : → "Mais" et "plus" ont un impact limité, souvent utilisés comme connecteurs.

4.8 MLFLOW : SUIVI DES MODELES

1. Introduction à MLflow

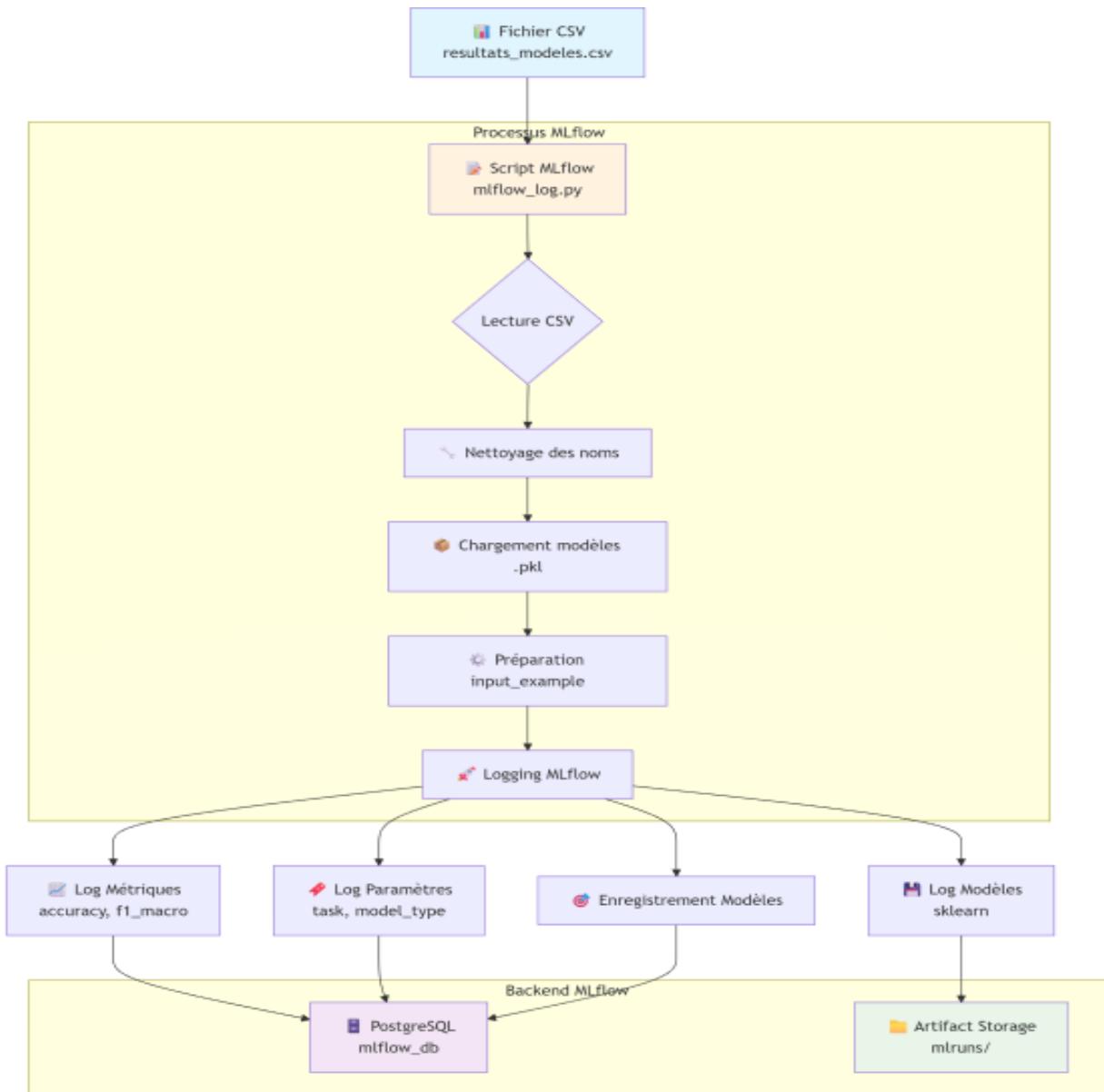
Dans le cadre de l'avancement constaté sur le projet principal d'analyse sentiment des avis Trustpilot, et pour donner suite à une proposition de mon tuteur visant à enrichir la valeur métier de la solution, **un module optionnel d'implémentation MLflow avancée a été développé.**

Objectif Métier :

Cette extension répondait au besoin stratégique de tracking complet et industrialisé des expérimentations de Machine Learning en permettant :

- Le versioning systématique de tous les modèles entraînés
- La comparaison fine des performances entre différentes approches
- La reproductibilité garantie des expérimentations

2. Schéma d'architecture fonctionnelle : utilisation de l'API MLflow (CSV → Script → Model → Logs → DB → Artefacts)



Sources de Données :

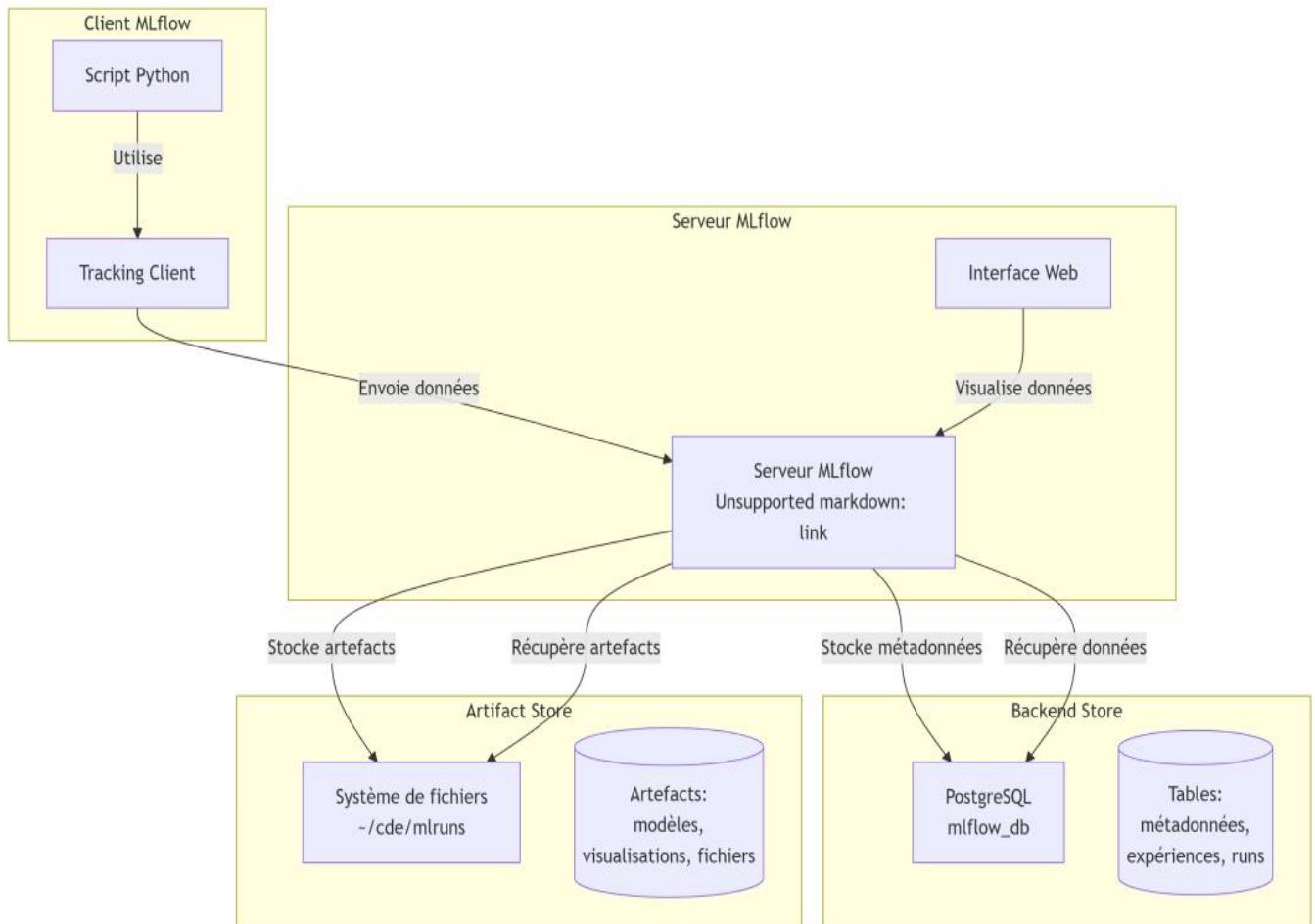
- `resultats_modeles.csv` : Fichier des résultats des modèles
- `data/model/*.pkl` : Modèles sérialisés joblib

Fonctions de Transformation :

- Nettoyage des noms : Adaptation pour MLflow
- Chargement modèles : Déserialisation. pkl
- Préparation input_example : Création dataframe d'exemple

3. Les composants principaux

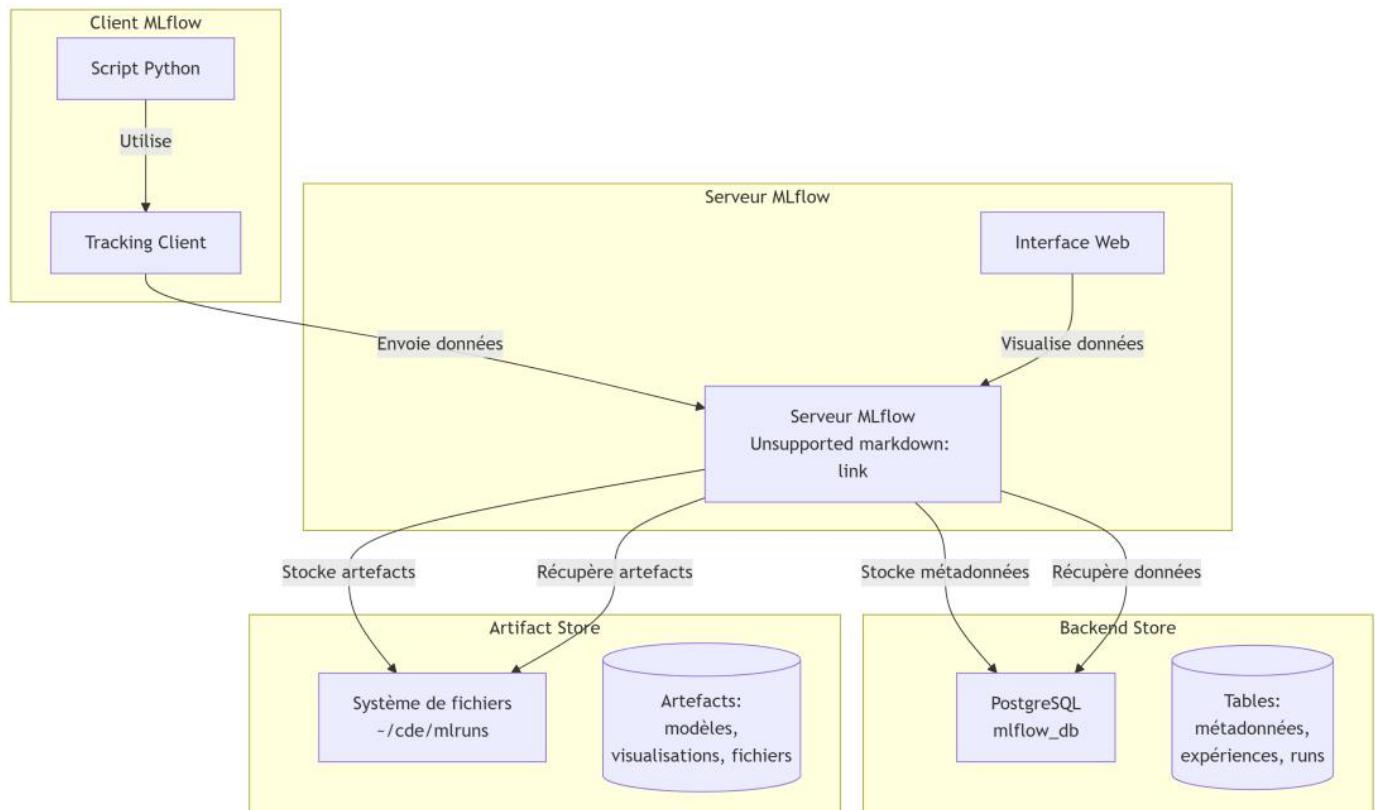
- a) **Backend Store :**
 - a. Stocke les métadonnées des expériences et des runs (exécutions).
 - b. Utilisation de **PostgreSQL** comme backend store.
 - c. URI: `os.environ['MLFLOW_TRACKING_URI']`
- b) **Artifact Store :**
 - a. Stocke les artefacts (modèles, visualisations, fichiers).
 - b. Dans ma configuration : `~/cde/mlruns`.
- c) **Serveur MLflow :**
 - a. Interface web pour visualiser les résultats.
 - b. Accès via `http://localhost:5000`.
- d) **Client MLflow :**
 - a. Utilisé dans les scripts Python pour logger les métriques et artefacts.



```
● (cde_env) datascientest@datascientest:~/cde/scripts/models$ python3 mlflow log.py*
2025/08/24 20:52:54 INFO mlflow.store.db.utils: Creating initial MLflow database tables...
2025/08/24 20:52:54 INFO mlflow.store.db.utils: Updating database tables
INFO [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO [alembic.runtime.migration] Will assume transactional DDL.
INFO [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO [alembic.runtime.migration] Will assume transactional DDL.
✓ 6 lignes de résultats chargées depuis /home/datascientest/cde/data/processed/resultats_modeles.csv
```

4. Installation des composants

- a) Installer MLflow et PostgreSQL driver : pip install mlflow psycopg2-binary
- b) Créer le dossier pour les artefacts : mkdir -p ~/cde/mlruns
- c) Configurer la variable d'environnement pour MLflow :
 - a. export
MLFLOW_TRACKING_URI=postgresql+psycopg2://mlflow_user:motdepasse@localhost/mlflow_db
 - b. export MLFLOW_ARTIFACT_ROOT=~/cde/mlruns
- d) Lancer le serveur MLflow :
 - a. mlflow ui \
 - b. --backend-store-uri \$MLFLOW_TRACKING_URI \
 - c. --default-artifact-root \$MLFLOW_ARTIFACT_ROOT \
 - d. --host 0.0.0.0 \
 - e. --port 5000
- e) Automatiser le lancement à la VM en créant le script cde/start_mlflow.sh:
 - a. #!/bin/bash
 - b. source ~/cde/cde_env/bin/activate
 - c. export
MLFLOW_TRACKING_URI=postgresql+psycopg2://admin:admin@localhost/mlflow_db
 - d. export MLFLOW_ARTIFACT_ROOT=~/cde/mlruns
 - e. echo " Démarrage du serveur MLflow..."
 - f. mlflow ui \
 - g. --backend-store-uri \$MLFLOW_TRACKING_URI \
 - h. --default-artifact-root \$MLFLOW_ARTIFACT_ROOT \
 - i. --host 0.0.0.0 \
 - j. --port 5000 &
- f) Changement des droits du script : chmod +x ~/cde/start_mlflow.sh
- g) Processus de logging des modèles ML vers MLflow : mlflow_log.py
- h) Lancer l'interface MLflow : http://localhost:5000/#/models



5. Data Drift

a) Contexte du Projet

Dans le cadre de ce projet, la problématique du **data drift** (dérive des données) se pose différemment. Mon processus de collecte de données (*scraping*) est limité et ponctuel. Une fois le jeu de données complet constitué, **l'arrivée de nouvelles données massives en temps réel n'est pas prévue**. Ainsi, une surveillance continue du drift, bien que conceptuellement importante, n'est pas une exigence opérationnelle immédiate.

Cependant, dans un contexte de déploiement "real-world" (monde réel), la mesure du data drift est une bonne pratique indispensable pour maintenir la performance et la pertinence d'un modèle dans le temps. MLflow fournit des outils pour faciliter cette surveillance.

b) Qu'est-ce que le Data Drift ?

Le **data drift** (ou dérive conceptuelle) se produit lorsque les données de production, sur lesquelles le modèle fait des prédictions, **divergent significativement** des données d'entraînement initiales. Cela entraîne une **dégradation silencieuse** des performances du modèle.

Causes principales :

- Changements dans le comportement des utilisateurs
- Évolution du marché ou de l'environnement

c) Comment MLflow Aide à Détecter le Drift ?

MLflow n'intègre pas *nativement* de fonctionnalités de détection de drift, mais son architecture ouverte et modulaire permet d'intégrer facilement cette surveillance dans le pipeline de suivi des modèles (*Model Registry*).

Implémentation Type avec MLflow :

3. **Logging des Statistiques de Base** : Lors de l'entraînement, utiliser `mlflow.log_metric()` et `mlflow.log_artifact()` pour enregistrer les statistiques descriptives du jeu d'entraînement (moyennes, écarts-types, distributions).
4. **Calcul et Comparaison avec les Nouvelles Données** : Dans un job scheduler (ex: Apache Airflow, CRON), programmer un script qui Récupère les statistiques de référence depuis MLflow, calcule une mesure de divergence (ex: Distance de Wasserstein, Divergence de Kullback-Leibler, Test de Kolmogorov-Smirnov)
5. **Alerting et Décision** : Logger la métrique de drift dans MLflow. Si un seuil prédéfini est dépassé, déclencher une alerte (email, Slack) via des webhooks

d) Conclusion pour le projet

Pour ce projet, je **mentionne la notion de data drift** pour démontrer une compréhension complète du cycle de vie d'un modèle en production. J'explique que, bien que non implémentée ici pour des raisons de contexte data, **la structure est en place** (via MLflow Registry et le stockage des artefacts) pour l'ajouter ultérieurement si le besoin évolue.

5. Résultats MLflow

⌚ Tâche SENTIMENT (3 modèles) :

- **sentiment_logisticregression** - Accuracy: 82.62% | F1: 0.7537
- **sentiment_linearsvc** – Accuracy : **85,06%** | F1: 0.7546 ← **Meilleur modèle**
- **sentiment_randomforest** - Accuracy: 81.40% | F1: 0.6445

⌚ Tâche NOTE (3 modèles) :

- **note_logisticregression** – Accuracy : 70.73% | F1: 0.4697
- **note_linearsvc** – Accuracy : **80.43%** | F1: 0.4949 ← **Meilleur modèle**
- **note_randomforest** – Accuracy : 79.48% | F1: 0.3594

Experiments :

| Run Name | Created Tg | Dataset | Duration | Source | Models |
|---------------------------------|----------------|---------|----------|-----------|---------------------------------|
| note_randomforest | 15 minutes ago | - | 6.0s | mlflow... | note_randomforest |
| note_linear | 15 minutes ago | - | 4.7s | mlflow... | note_linear |
| note_logisticregression | 15 minutes ago | - | 5.6s | mlflow... | note_logisticregression |
| sentiment_randomforest | 15 minutes ago | - | 5.1s | mlflow... | sentiment_randomforest |
| sentiment_linear | 15 minutes ago | - | 5.5s | mlflow... | sentiment_linear |
| sentiment_logisticregression | 15 minutes ago | - | 4.9s | mlflow... | sentiment_logisticregression |
| note_randomforest | 3 hours ago | - | 3.8s | mlflow... | note_randomforest |
| note_linear | 3 hours ago | - | 4.0s | mlflow... | note_linear |
| note_logisticregression | 3 hours ago | - | 4.7s | mlflow... | note_logisticregression |
| sentiment_randomforest | 3 hours ago | - | 4.0s | mlflow... | sentiment_randomforest |
| sentiment_linear | 3 hours ago | - | 4.8s | mlflow... | sentiment_linear |
| sentiment_logisticregression | 3 hours ago | - | 3.8s | mlflow... | sentiment_logisticregression |
| note_randomforest | 1 day ago | - | 5.3s | mlflow... | randomforest_note |
| note_linear | 1 day ago | - | 3.8s | mlflow... | linear_note |
| note_logisticregression | 1 day ago | - | 3.8s | mlflow... | logisticregression_note |
| sentiment_randomforest | 1 day ago | - | 4.7s | mlflow... | randomforest_sentiment |
| sentiment_linear | 1 day ago | - | 3.8s | mlflow... | linear_sentiment |
| sentiment_logisticregression | 1 day ago | - | 4.6s | mlflow... | logisticregression_sentiment |
| randomforest_note | 1 day ago | - | 3.0s | mlflow... | randomforest_note |
| linear_note | 1 day ago | - | 2.0s | mlflow... | linear_note v2 +1 |
| logisticregression_note | 1 day ago | - | 2.1s | mlflow... | logisticregression_note |
| randomforest_linear | 1 day ago | - | 2.8s | mlflow... | randomforest_linear |
| linear_linear | 1 day ago | - | 2.1s | mlflow... | linear_linear -v1 |
| logisticregression_linear | 1 day ago | - | 2.5s | mlflow... | logisticregression_linear |
| randomforest_randomforest | 1 day ago | - | 1.4s | mlflow... | randomforest_randomforest |
| linear_randomforest | 1 day ago | - | 1.4s | mlflow... | linear_randomforest |
| logisticregression_randomforest | 1 day ago | - | 1.6s | mlflow... | logisticregression_randomforest |

Models :

| Name | Latest version | Aliased versions | Created by | Last modified | Tags |
|------------------------------|----------------|------------------|------------|--------------------------|------|
| linearsvc_note | Version 2 | | | 08/23/2023, 07:02:06 ... | — |
| linearsvc_sentiment | Version 2 | | | 08/23/2023, 07:01:59 ... | — |
| logisticregression_note | Version 2 | | | 08/23/2023, 07:02:04 ... | — |
| logisticregression_sentiment | Version 2 | | | 08/23/2023, 07:01:57 ... | — |
| note_linearsvc | Version 2 | | | 08/24/2023, 08:53:21 ... | — |
| note_logisticregression | Version 2 | | | 08/24/2023, 08:53:16 ... | — |
| note_randomforest | Version 2 | | | 08/24/2023, 08:53:27 ... | — |
| randomforest_note | Version 2 | | | 08/23/2023, 07:02:10 ... | — |
| randomforest_sentiment | Version 2 | | | 08/23/2023, 07:02:02 ... | — |
| sentiment_linearsvc | Version 2 | | | 08/24/2023, 08:53:05 ... | — |
| sentiment_logisticregression | Version 2 | | | 08/24/2023, 08:53:00 ... | — |
| sentiment_randomforest | Version 2 | | | 08/24/2023, 08:53:11 ... | — |

6. Liste des requêtes dans la base MLflow pour récupérer les données

- Lister les dernières métriques enregistrées par run

```
SELECT
    r.run_uuid,
    r.name as run_name,
    e.name as experiment_name,
    m.key as metric_name,
    m.value as metric_value,
    m.timestamp
FROM runs r
JOIN experiments e ON r.experiment_id = e.experiment_id
JOIN latest_metrics m ON r.run_uuid = m.run_uuid
ORDER BY r.start_time DESC;
```

- Top 10 des runs par métrique spécifique (ex: "accuracy")

```
SELECT
    r.run_uuid,
    r.name as run_name,
    e.name as experiment_name,
    m.value as accuracy
FROM runs r
JOIN experiments e ON r.experiment_id = e.experiment_id
JOIN latest_metrics m ON r.run_uuid = m.run_uuid
WHERE m.key = 'accuracy'
ORDER BY m.value DESC
LIMIT 10;
```

- Voir toutes les métriques disponibles pour un run spécifique

```
SELECT
    key,
    value,
    timestamp
FROM metrics
WHERE run_uuid = 'numero de run'
ORDER BY key;
```

- Compter le nombre de métriques enregistrées par run

```
SELECT
    r.run_uuid,
    r.name as run_name,
    COUNT(m.key) as number_of_metrics
FROM runs r
LEFT JOIN metrics m ON r.run_uuid = m.run_uuid
GROUP BY r.run_uuid, r.name
ORDER BY number_of_metrics DESC;
```

5. DEVELOPPEMENT DE L'API AVEC FAST API (PARTIE 4)

Concevoir une API RESTful performante avec FastAPI en Python, sécurisée par JWT pour une gestion optimale des accès. L'API sera conteneurisée via Docker pour faciliter son déploiement et sa portabilité, tout en bénéficiant d'une documentation automatique. La qualité du code sera garantie par des tests unitaires rigoureux, implémentés avec Pytest.

Rappel du plan :

▣ **Créer une API robuste avec FastAPI, sécurisée par authentification, livrée dans un conteneur Docker (étape 4) :**

- ▣ **Développer une API FastAPI sécurisée avec des endpoints**
- ▣ **Conteneuriser l'application avec Docker pour assurer son déploiement et son isolation.**
- ▣ **Implémenter des tests unitaires avec Pytest**
- ▣ **Sécuriser l'API via un mécanisme d'authentification**

6. RESULTATS & SYNTHESE METIER

L'analyse technique des modèles de Machine Learning, bien que cruciale, n'a de valeur que **si elle sert une finalité business**. La volumétrie des avis (plus de 66 000), leur polarisation extrême et les performances de classification obtenues m'ont permis de dépasser le simple constat statistique pour formuler un diagnostic stratégique pour chaque entreprise.

En croisant les prédictions de sentiment, la distribution des notes et l'analyse textuelle des commentaires, j'ai identifié pour chaque acteur :

1. Son pilier principal de satisfaction (sa "force stratégique").
2. Son point de douleur le plus critique (son "risque réputationnel").
3. Une recommandation prioritaire et actionnable pour convertir les détracteurs en promoteurs.

| Entreprise | Pilier de Satisfaction | Principal Point de Douleur | Recommandation Principale |
|------------|------------------------|---------------------------------|---|
| Tesla | Innovation Produit | Service Après-Vente | Investir dans l'expérience post-achat |
| Chronopost | Rapidité & Réseau | Dernier Kilomètre & Flexibilité | Plus de transparence et de solutions de recours |
| Vinted | Communauté & Prix | Arbitrage des Litiges | Renforcer la confiance via un système de résolution équitable et rapide |
| Temu | Prix Bas | Écart Attente/Réalité | Gérer les attentes et simplifier les retours |

6.1 RESULTATS TESLA



Polarisation extrême entre l'amour de l'innovation (5★) et la haine du SAV (1★). La technologie est otage d'un service après-vente perçu comme lent, cher et opaque. **Priorité** : Investir d'urgence dans la logistique SAV et une communication transparente pour protéger l'image premium.

6.2 RESULTATS CHRONOPOST



Performance perçue comme binaire. La livraison est soit rapide et fiable (5★), soit un échec total (1★) à cause d'un **manque de flexibilité** et de **transparence** du dernier kilomètre. **Priorité** : Améliorer le suivi en temps réel et offrir des solutions de recours immédiates.

6.3 RESULTATS VINTED



La force de la communauté (5★) est sapée par les **litiges entre utilisateurs** (1-2★). La confiance, pilier du modèle, est érodée par des arbitrages perçus comme lents et injustes. **Priorité** : Automatiser les remboursements simples et renforcer la médiation humaine pour les cas complexes.

6.4 RESULTATS TEMU



Le prix bas génère de la satisfaction (5★) mais aussi de la **frustration** (1★) due à l'écart entre les attentes (photos) et la réalité (qualité, délais). **Priorité** : Gérer les attentes via des avis clients réalistes et simplifier les retours.

7. RESULTATS & SYNTHESE PROJET

7.1 RESULTATS CLES : SYNTHESE DES PARTIES 1 A 4

1.1. Etape 1 : Récolte des données - Extraction et transformation

Chaque jour, le script de collecte reprend automatiquement l'extraction là où elle s'était arrêtée, en enregistrant les résultats dans des répertoires distincts, horodatés par société (étape 1)

- **4 entreprises cibles** : Tesla, Temu, Chronopost, Vinted.
- **66 864 avis** stockés dans PostgreSQL/MongoDB (exemple : scrap_tesla_<date>/).
- **Métadonnées enrichies** via Wikipédia (SIREN, secteurs d'activité, etc.).
 - ◆ Scraping automatique et historisé

1.2. Etape 2 : Organisation de la donnée – Chargement

Une architecture de base de données double a été conçue comprenant (étape 2)

- **PostgreSQL** : Tables societe / Vue vue_sociedades_wiki_harmonisee
- **MongoDB**: Collections JSON avis_trustpilot
 - ◆ Déploiement reproductible via Docker

1.3. Etape 3 : Consommation de la donnée

*Le modèle BERT pré-entraîné 'nlptown/bert-base-multilingual-uncased-sentiment' pour générer deux cibles : le **sentiment des commentaires** (négatif/neutre/positif) et leur **note prédite** (1 à 5 étoiles) (étape 3)*

*Projet MLflow pour suivre les performances de **6 modèles** (3 pour l'analyse de sentiment et 3 pour la prédiction de notes).*

Performances Globales :

| Tâche | Modèle | Accuracy | F1_score_macro |
|------------------|--------------------|---------------|----------------|
| sentiment | LogisticRegression | 0.8262 | 0.7537 |
| sentiment | LinearSVC | 0.8506 | 0.7546 |
| sentiment | RandomForest | 0.814 | 0.6445 |
| <hr/> | | | |
| note | LogisticRegression | 0.7073 | 0.4697 |
| note | LinearSVC | 0.8043 | 0.4949 |
| note | RandomForest | 0.7948 | 0.3594 |

- ◆ **LinearSVC** est le meilleur modèle global (dépasse LogisticRegression et RandomForest).

1.4. Etape 4 : Développement d'une API sécurisée par authentification, livrée dans un conteneur Docker (étape 4).

7.2 LIMITES ET CONSIDERATIONS ETHIQUES

Dans le cadre de ce projet de formation professionnel, j'ai identifié plusieurs limites techniques ainsi que des enjeux éthiques et légaux qu'il est essentiel de prendre en compte.

1. Limites techniques

- **Qualité des données** : J'ai constaté que les avis collectés sur Trustpilot ou d'autres plateformes peuvent être biaisés, car les clients mécontents sont souvent plus enclins à laisser un avis que les clients satisfaits. Cela peut fausser l'analyse de sentiments et la représentativité des résultats. Par ailleurs, les données issues de Wikipédia, bien qu'utiles, peuvent être incomplètes ou obsolètes, car elles dépendent des contributions volontaires des utilisateurs.
- **Représentativité des données** : Les avis scrappés peuvent ne pas représenter l'ensemble des clients des entreprises cibles. Par exemple, certains groupes démographiques ou géographiques peuvent être sous-représentés, ce qui limite la généralisation des conclusions. De même, les données structurées de Wikipédia, bien que riches, manquent parfois de granularité pour des analyses approfondies.
- **Contraintes du web scraping** : J'ai rencontré des difficultés liées aux mécanismes anti-scraping mis en place par des plateformes comme Trustpilot (robots.txt), ce qui a restreint la quantité et la fréquence de collecte des données. De plus, **les modifications régulières de la structure HTML des pages web peuvent rendre mes scripts de scraping obsolètes**, nécessitant des mises à jour constantes.
- **Modèles de Machine Learning** : Les modèles d'analyse de sentiments ou de classification que j'ai développés peuvent être biaisés en fonction des données d'entraînement. Les performances de ces modèles dépendent également de la qualité du prétraitement des données (nettoyage, tokenisation, etc.), ce qui peut introduire des erreurs si ces étapes ne sont pas optimisées.

2. Considérations techniques

- **Respect de la vie privée** : Je n'ai pas chargé les noms des auteurs contenues dans les avis clients (noms d'utilisateurs) pour éviter toute violation de la vie privée. Conformément au **Règlement Général sur la Protection des Données (RGPD)**, j'ai garanti que les données collectées ne soient utilisées qu'à des fins analytiques et académiques, sans exploitation commerciale ou malveillante. Le RGPD impose notamment :
 - La **minimisation des données** : ne collecter que les informations strictement nécessaires.
 - Le **droit à l'oubli** : permettre aux utilisateurs de demander la suppression de leurs données.
 - La **transparence** : informer les utilisateurs de l'utilisation de leurs données, si cela est applicable.
- **Conditions d'utilisation des plateformes** : J'ai vérifié que le scraping de Trustpilot et l'utilisation de l'API Wikipédia respectaient leurs conditions d'utilisation.
- **Transparence et reproductibilité** : J'ai documenté clairement mes méthodes de collecte et de traitement des données pour assurer la transparence du projet. J'ai évité toute manipulation ou décontextualisation des avis clients, afin de ne pas fausser les résultats ou induire en erreur.
- **Impact sur les entreprises analysées** : J'ai présenté les résultats de manière objective et équilibrée, en évitant toute interprétation qui pourrait nuire à la réputation des entreprises cibles (Tesla, Temu, Chronopost, Vinted). Les conclusions ont été formulées avec prudence, en soulignant les limites des analyses.
- **Utilisation responsable des données** : Je me suis engagé à ne pas utiliser les données collectées à des fins commerciales, de spam ou de désinformation. Si ce projet devait être partagé publiquement, je m'assurerais que les conclusions ne portent pas préjudice aux entreprises ou aux individus concernés.

3. Considérations légales

- **Droit d'auteur et propriété intellectuelle** : Les avis clients sont protégés par le droit d'auteur. Leur utilisation a été strictement limitée à un cadre analytique et académique, sans diffusion publique non autorisée.
- **Conformité aux réglementations** : J'ai respecté les lois locales et internationales sur la protection des données, notamment le **RGPD en Europe** et les politiques internes de **DataScientest**. La charte éthique de DataScientest souligne l'importance :
 - De l'**intégrité des données** : ne pas altérer ou falsifier les informations collectées.
 - De la **confidentialité** : protéger les données sensibles et anonymiser les informations personnelles.
 - De la **responsabilité** : assumer les conséquences de mes analyses et garantir leur exactitude.

8. CONCLUSIONS & PERSPECTIVES

Ce projet m'a permis de mettre en œuvre l'ensemble des étapes d'un pipeline de données, depuis la collecte jusqu'à l'analyse avancée.

J'ai renforcé mes compétences en **Python**, notamment dans le web scraping, la manipulation de fichiers JSON et la création de scripts **ETL** fiables et maintenables.

L'intégration simultanée de **PostgreSQL** et **MongoDB** m'a permis de travailler avec deux approches complémentaires : la structuration stricte d'une base relationnelle et la souplesse d'une base NoSQL. L'utilisation de **Docker** et **Docker Compose** m'a aidé à comprendre concrètement comment créer des environnements reproductibles et isolés, facilitant le déploiement et la portabilité du projet.

1. Pour enrichir l'analyse, j'ai utilisé l'**API Wikipédia** afin de collecter des métadonnées sur les entreprises étudiées (Tesla, Chronopost, Temu, Vinted), que j'ai ensuite intégrées dans PostgreSQL. Cette étape a permis de relier les retours clients à un contexte institutionnel plus large.
2. Pour structurer et comparer les itérations de modélisation, j'ai configuré **MLflow** afin de suivre systématiquement les hyperparamètres et les métriques de performance. Cette pratique a permis d'objectiver la sélection du modèle final.

La partie **Machine Learning** a constitué le cœur analytique du projet. J'ai conçu un pipeline complet pour le traitement de texte à grande échelle comprenant :

- **Prétraitement** : nettoyage, normalisation, tokenisation, vectorisation (TF-IDF, embeddings contextuels).
- **Application de modèles classiques** : tests avec Logistic Regression et Random Forest pour disposer d'une base de comparaison robuste.
- **Approche avancée** : fine-tuning d'un modèle pré-entraîné **BERT** pour la classification des sentiments.
- **Évaluation** : utilisation de métriques variées (accuracy, F1-score, AUC) et mise en place d'un suivi des performances dans le temps.
- **Application utilisateur** : création d'une interface **Streamlit** pour tester en direct les modèles avec des saisies personnalisées.
- **Intégration de MLflow** pour suivre les expériences, centraliser les métriques et faciliter la comparaison et le déploiement des modèles

Ce travail m'a permis de mieux comprendre comment relier les aspects **ingénierie des données, modélisation et mise à disposition des résultats dans une démarche cohérente**.

Pour industrialiser cette solution et pour la suite de ce projet, une API FastAPI sera développée pour permettre d'exposer les modèles de prédiction, de conteneuriser l'application avec Docker et d'automatiser le déploiement via un pipeline CI.

Le scraping serait planifié quotidiennement avec Airflow ou des cronjobs, tandis que la surveillance en production s'appuierait sur Prometheus/Grafana pour monitorer les performances et détecter d'éventuelles dérives de données.