Schéma du pipeline de données

Tu peux le mettre tel quel dans ton rapport (ou le transformer en diagramme Draw.io / PowerPoint).

Vue globale (données → analyse → interfaces)

```
flowchart LR

A[📥 tweets.csv (raw)\ndata/raw/tweets.csv] --> B[🧹 Préparation\n(tweets_prepared.csv)]
B --> C[🧠 Enrichissement NLP + LLM\n(tweets_enriched.csv)]
C --> D1[📞 Espace Agent\n1_Agent.py]
C --> D2[📊 Espace Manager\n2_Manager.py]
C --> D3[🔎 Espace Analyste\n3_Analyst.py]
A --> D4[🌐 Fil public\n5_Public_Feed.py]

subgraph Pré-traitement
B
end

subgraph Enrichissement
C
end

subgraph Interfaces Streamlit
D1
D2
D3
D4
end
```

Vue détaillée du pipeline technique

```
flowchart TD

RAW[📥 Import CSV\n tweets.csv] --> |Admin - Upload|
INGEST[Ingestion & validation\n(enrichment_pipeline.py)]
INGEST --> CLEAN[Nettoyage & structuration\ncleaning_service.py\n+
```

```
tweets_prepared.csv]
CLEAN --> NLP[NLP basique\nnlp_basic_service.py]
NLP --> LLM[Appels LLM Mistral\nllm_client.py + classification.py]
LLM --> POSTPROC[Post-traitement\n(consolidation règles + LLM)]
POSTPROC --> ENRICH[📁 Export enrichi\n tweets_enriched.csv]
ENRICH --> KPIS[KPIs & stats\nkpis_analyst.py, report_builder.py]
ENRICH --> UI_AGENT[UI Agent\n1_Agent.py]
ENRICH --> UI_MANAGER[UI Manager\n2_Manager.py]
ENRICH --> UI_ANALYST[UI Analyste\n3_Analyst.py]
RAW --> UI_PUBLIC[UI Fil public\n5_Public_Feed.py]
```

2. Doc résumé & précis – Ce que fait chaque partie

2.1. Les fichiers de données CSV

data/raw/tweets.csv

- **Source brute** : export Twitter (tous les tweets, retweets, réponses, citations, officiels & clients).
- Colonnes principales :
  - `id`, `created_at`, `full_text`
  - `screen_name`, `name`
  - `in_reply_to`, `retweeted_status`, `quoted_status`
  - `favorite_count`, `retweet_count`, `reply_count`, `views_count`
- Utilisations :
  - **Entrée du pipeline** (Admin → lancement pipeline)
  - **Fil public** (5_Public_Feed.py) → timeline type Twitter

data/processed/tweets_prepared.csv

- Fichier **nettoyé & structuré**, résultat du pré-traitement.
- Colonnes ajoutées / modifiées (exemple) :
  - `text_clean` : texte nettoyé (sans URL, hashtags, mentions, etc.)
  - `tweet_type` : `original` / `reply` / `retweet` / `quote`
  - `author_type` : `official` (Free, Freebox, etc.) ou `client`
  - `keep_for_analysis` : booléen → tweet retenu pour l'analyse SAV

- Objectif :
    - Transformer le brut en **données prêtes pour le NLP + LLM**.
    - Ne rien supprimer physiquement, mais marquer ce qui est utile.

data/processed/tweets_enriched.csv (ou tweets_enriched_fixed.csv)

- Fichier **final**, exploité par les interfaces internes.
- Contient toutes les colonnes de `tweets_prepared.csv` **+ colonnes d'analyse** :
    - `final_intent` :
        - `complaint`, `question`, `suggestion`, `thanks`, `other`
        - avec correction de l'ironie (beaucoup de faux "thanks" → reclassés en `complaint`)
    - `final_sentiment` :
        - `negative`, `neutral`, `positive`
        - ajusté pour les cas de sarcasme et de menace de résiliation
    - `final_sarcasm` : `True` / `False`
    - `final_complaint_type` :
        - (ex.) `fibre`, `mobile`, `facturation`, `SAV`, `débit`, etc.
    - `final_priority` :
        - `critical`, `high`, `medium`, `low`, `none`
    - `nlp_*` ou autres colonnes techniques provenant des règles (détection de panne, urgence, résiliation…)
- Utilisations :
    - Vue Agent (liste de tickets à traiter)
    - Vue Manager (KPI, volumes, priorités, types de plaintes)
    - Vue Analyste (analyses détaillées, rapports)

2.2. Modules backend (pipeline)

backend/data_pipeline/enrichment_pipeline.py

- **Cerveau du pipeline**.
- Orchestration :
    - Prend `tweets.csv`
    - Appelle les services :
        - ingestion / cleaning

        - NLP basique
        - LLM (classification)
        - postprocessing
    - Sauvegarde :
        - `tweets_prepared.csv`
        - `tweets_enriched.csv`
    - Retourne aussi des **KPIs globaux** au front Admin.

◇ *backend/data_pipeline/cleaning_service.py*

- S'occupe du **nettoyage** :
    - enlève URL, mentions, hashtags
    - normalise le texte (`text_clean`)
- Détecte le **type de tweet** :
    - `retweet` si `retweeted_status` ou `RT @...`
    - `quote` si `quoted_status`
    - `reply` si `in_reply_to` non vide
    - sinon : `original`
- Détermine `author_type` (client vs compte officiel).

◇ *backend/data_pipeline/nlp_basic_service.py*

- Analyse **NLP classique** (sans LLM) :
    - détecte mots liés à la panne, au mécontentement, à la résiliation…
    - calcule un **sentiment baseline** (simplifié)
    - repère certains signaux d'urgence (durée qui s'allonge, "depuis 3 jours…", etc.)
- Sert de **filet de sécurité** : les résultats peuvent corriger ou renforcer le LLM.

◇ *backend/data_pipeline/llm_client.py*

- Interface avec l'API **Mistral** :
    - charge la clé API dans `.env`
    - envoie le texte + quelques métadonnées au modèle (ex. `mistral-small-latest`)

- récupère : intent, plainte, sarcasme, sentiment, priorité, risque…

◇ *backend/data_pipeline/classification.py*

- Utilise `llm_client` + les infos `nlp_basic_service`.
- Construit les colonnes finales :
  - `final_intent`
  - `final_complaint_type`
  - `final_priority`
  - `final_sentiment`
  - `final_sarcasm`
  - `nlp_has_resiliation_risk`, etc.
- Intègre les corrections métier (ex : ironie “Super le service après-vente” → plainte négative).

◇ *backend/analytics/kpis_analyst.py & backend/analytics/report_builder.py*

- Calculent les **statistiques** et **KPI** :
  - volumes, nombre de plaintes, part de négatif, part de sarcasme, etc.
- `report_builder.py` produit un **rapport texte** synthétique (utilisé par Analyst & Admin).

2.3. Les pages Streamlit (interfaces)

Home.py

- Page d’accueil + **login**.
- Authentifie l’utilisateur (en fonction d’un JSON / store interne).
- Redirige selon le rôle :
  - Agent
  - Manager
  - Analyste
  - Admin
  -

        - accès au fil public

pages/1_Agent.py

- Interface pour les **agents SAV** :
    - liste des tweets/“tickets” à traiter (principalement les `complaints`)
    - filtres par :
        - priorité,
        - type de plainte,
        - sentiment,
    - possibilité de changer le **statut du ticket** : nouveau, `en cours`, `résolu`.
- Utilise `tweets_enriched.csv` comme source.

pages/2_Manager.py

- Vue **Manager** :
    - KPI globaux :
        - nombre de plaintes,
        - plaintes `high/critical`,
        - risque de résiliation
        - part de tweets négatifs…
    - graphiques :
        - répartition des intents
        - types de plaintes
        - priorités
        - évolution du volume et du sentiment dans le temps
    - liste de suggestions / feedbacks positifs pour des actions d’amélioration.

pages/3_Analyst.py

- Vue **Analyste**, la plus détaillée :
    - Volume brut / préparé / enrichi
    - Tweets comptes Free vs clients
    - Structure du flux (retweets, réponses, citations)
    - Répartition des intents / plaintes / priorités

- Sentiment, sarcasme, résiliation
- Évolution temporelle (courbes)
- Génération d'un **rapport téléchargeable**.

🗂️ *pages/4_Admin_Upload_CSV.py*

- Interface **Admin** :
  - vérifie la présence des fichiers :
    - `tweets.csv`
    - `tweets_prepared.csv`
    - `tweets_enriched.csv`
  - cas gérés :
    - tous présents → pas besoin de relancer, on affiche juste KPIs + rapport
    - aucun présent → demande d'upload du CSV brut
    - brut présent mais pas enrichi → propose de **lancer le pipeline**
  - permet de **relancer le pipeline complet** :
    - nettoyage, NLP, LLM
    - puis affichage synthétique des résultats.

🗂️ *pages/5_Public_Feed.py*

- Fil **public type Twitter** :
  - source : `tweets.csv` (brut)
  - affiche uniquement :
    - tweets originaux
    - retweets
    - citations
  - les **replies** apparaissent comme **commentaires** lorsque l'utilisateur clique sur 💬.
  - barre de recherche : par texte + auteur
  - pagination
  - visuel spécifique :
    - retweet :
      - " 🔁 @X a retweeté"
      -
        - tweet original

        - citation :
            - commentaire de l’utilisateur
            -
                - bloc “tweet cité”

2.4. Rôles et usage des fichiers

- **`tweets.csv`** → matière brute, utilisée :
    - pour le pipeline
    - pour la page publique
- **`tweets_prepared.csv`** → étape intermédiaire, utile pour :
    - déboguer / montrer le travail de nettoyage
    - expliquer le pré-traitement dans le rapport
- **`tweets_enriched.csv`** → cœur de l’exploitation métier :
    - agents, managers, analystes
    - KPI, dashboards, rapport