## Automatisation **de la production des datapacks et gap analysis**

Le nouveau processus intègre une **automatisation complète** de la production des datapacks au format investisseur ainsi que des rapports de **gap analysis**. Cette automatisation repose sur une chaîne de traitement structurée, modulaire et réutilisable, qui permet de :

**Étapes automatisées :**

1. **Création des jeux de données historiques** : récupération des données N et N-1 à partir des sources identifiées.
2. **Nettoyage et contrôle qualité des données** via le module DataQuality, avec application de seuils définis par famille d’indicateurs.
3. **Calcul des indicateurs** (PD obligor, PD contrat, LGD, taux de défaut, back-in-bonis, taux de recouvrement, etc.) à l’aide de la classe Indicator et ses dérivés.
4. **Génération automatique du datapack**, structuré par :
   * Entités (SME, LC, VLC, Common)
   * Rating system
   * Indicateur
5. **Production de la Gap Analysis** :
   * Comparaison automatisée entre les indicateurs N et N-1.
   * Génération de fichiers de sortie normalisés (.csv) incluant les écarts (absolus et relatifs), les nouvelles populations, et les disparitions.
6. **Dépôt dans un dossier de sortie nommé dynamiquement** selon le pattern suivant : *name\_datapack\_YYYYMMDD\_HHMM*

Cette automatisation repose sur un orchestrateur centralisé : la **classe Process**.

**Classe Process : orchestration de bout-en-bout**

La classe Process a été développée pour orchestrer toutes les étapes nécessaires à la génération d’un datapack complet. Elle repose sur la méthode principale run\_process() qui exécute, dans l’ordre, les étapes suivantes :

**Étape 1 : Collecte des données — classe Data**

Cette étape permet de charger toutes les données nécessaires à partir d’un dossier géré (managed folder) ou de datasets sur S3.

**Méthodes principales :**

* **load\_data\_from\_folder()** : charge les données selon quatre paramètres :
  + source\_type : "folder" ou "dataset"
  + source\_names : liste des datasets à charger (si dataset)
  + folder\_id : identifiant du dossier Dataiku
  + mapping\_file\_name : fichier Excel de mapping
  + **Sortie** : dictionnaire de DataFrames
* **collect\_data()** : applique le mapping pour structurer les données sous forme de DataFrames PySpark selon la configuration du fichier de mapping.
* **retrieve\_data()** : récupère les fichiers CSV à partir d’un dossier cible et les charge sous forme de dictionnaire de DataFrames.

**Étape 2 : Préparation des données**

Les données collectées sont organisées par système de rating (SME, LC, VLC, etc.), ce qui facilite les traitements par périmètre. Exemple :

python

CopierModifier

data\_origination\_df = {

'sme': data\_instance["pd"].df\_selected["origination\_data"].filter(col("rt\_system\_cn") == "PME"),

'lc' : data\_instance["pd"].df\_selected["origination\_data"].filter(col("rt\_system\_cn") == "LC")

}

**Étape 3 : Analyse qualité — classe DataQuality**

Cette classe effectue plusieurs contrôles statistiques sur les DataFrames :

* **check\_missing\_values()** : taux de valeurs manquantes par colonne
* **check\_duplicates()** : détection des doublons
* **check\_outliers()** : identification des outliers selon des seuils définis
* **sanity\_check()** : visualisations statistiques pour vérification humaine (distribution des ratings, évolution du nombre de contrats, etc.)
* **generate\_dq\_report()** : regroupe tous les contrôles dans un rapport de qualité global

**Étape 4 : Calcul des indicateurs — classes Indicator, Datapack et fichier indicator\_config**

Le calcul repose sur un fichier de configuration indicator\_config qui précise :

* enabled : activer/désactiver l’indicateur
* class : classe Python utilisée
* name, id\_indicator, tab : métadonnées
* rating\_system : périmètres concernés
* data et full\_df : sources de données nécessaires

**Méthodes principales :**

* **get\_indicator()** : applique le calcul de chaque indicateur
* **generate\_indicator()** : génère les fichiers CSV dans le dossier de sortie, organisés par système de rating

**Étape 5 : Génération du Datapack et Gap Analysis**

Grâce à la classe Datapack :

* **Les résultats d’indicateurs** sont utilisés pour alimenter les modèles Excel à l’aide de la méthode :
  + **fill\_model\_excel()** : alimente les plages nommées du template et sauvegarde une copie horodatée.
* **La gap analysis** est produite automatiquement en comparant les résultats N et N-1 à travers un module dédié.