**📊 Traitement des données**

L'efficacité d’un modèle de Machine Learning dépend directement de la qualité des données utilisées en entrée. Avant toute tentative de prédiction, une étape essentielle consiste donc à nettoyer, structurer et filtrer les données de manière rigoureuse. Cette section détaille l’ensemble du processus de traitement appliqué au fichier **BI\_Prices\_Data.xlsx**, fourni par le Becquerel Institute.

**🔹 Phase de nettoyage des données**

Le fichier source contient une feuille nommée **Component Prices Data**, qui recense plusieurs catégories de composants photovoltaïques, chacun accompagné de sa description, son type (cellule ou module), sa fréquence d’actualisation (souvent hebdomadaire), son prix unitaire (en $/Wp ou RMB/Wp), ainsi que sa source (EnergyTrend, PVinsights, etc.). La première étape a consisté à importer cette feuille en utilisant la bibliothèque Python pandas.

Le fichier comportant un volume important de données, il a été nécessaire de cibler uniquement les composants pertinents. Deux types ont été retenus pour l’analyse : les cellules **“n-type TOPCon M10”** et les modules **“n-type TOPCon”**. Un filtrage a permis de ne conserver que les lignes contenant ces libellés, en ignorant la casse et en éliminant les enregistrements incomplets.

Par la suite, seules les colonnes utiles ont été conservées : la date (Date), le type de composant (Item), la description (Description) et le prix (Base Price). Les données ont été transformées sous forme de tableau croisé (pivot) pour obtenir une ligne par date et deux colonnes distinctes : l’une pour les **prix des cellules n-type TOPCon M10**, l’autre pour les **prix des modules n-type TOPCon**.

Un formatage a été appliqué pour assurer la cohérence des types : la colonne Date a été convertie au format datetime, les prix ont été forcés au format numérique (float) et l’ensemble a été trié par ordre chronologique. Des valeurs manquantes ont été identifiées, notamment dues au fait que certains composants ne sont pas disponibles tous les jours. Selon les cas, ces valeurs ont été interpolées, comblées par des moyennes glissantes ou supprimées si trop rares.

Enfin, un nettoyage final a permis de garantir l’unicité des lignes, l’alignement parfait des deux séries temporelles et l’homogénéité des unités. Le jeu de données ainsi préparé a été exporté dans un fichier Excel nommé **Prix\_TOPCon.xlsx**, prêt à être utilisé pour l’entraînement de modèles prédictifs.

Ce jeu de données propre permet désormais une analyse temporelle hebdomadaire, une comparaison entre cellules et modules **n-type TOPCon**, et une préparation optimale pour l’application de méthodes de séries temporelles.

**🔹 Filtrage des données selon la source EnergyTrend**

Bien que le fichier source rassemble des données issues de plusieurs fournisseurs, il a été décidé de se concentrer uniquement sur la **source EnergyTrend**. Ce choix repose sur la qualité, la régularité et la cohérence des données produites par cette source, et vise à éviter toute disparité méthodologique liée à des fournisseurs différents.

Ce filtrage permet d’éviter les biais introduits par des modes de collecte hétérogènes, de limiter les discontinuités dans les séries, et de garantir une homogénéité des unités, des fréquences et des formats.

Le jeu de données utilisé est toujours celui issu de la feuille Component Prices Data, contenant les colonnes Date, Item, Description, Base Price et Source. Après avoir réalisé un nettoyage préliminaire (conversion des dates, suppression des doublons et des valeurs manquantes), un filtrage a été appliqué à la colonne Source :

python

CopierModifier

df\_energytrend = df[df['Source'].str.lower() == 'energytrend']

Cette opération permet de ne conserver que les lignes dont la source est bien EnergyTrend, quel que soit le format de casse utilisé (ex. : "ENERGYTREND", "Energytrend", etc.).

Une fois le filtrage effectué, les données ont été triées par date et réindexées afin de garantir une série temporelle continue et exploitable :

python

CopierModifier

df\_energytrend = df\_energytrend.sort\_values(by='Date').reset\_index(drop=True)

Le résultat final est un sous-ensemble propre, complet et homogène des prix des composants photovoltaïques issus exclusivement de la source **EnergyTrend**, prêt à être utilisé pour les phases suivantes d’analyse exploratoire et de modélisation prédictive.

**📏 Normalisation des données**

Une fois les données nettoyées, structurées et filtrées, une étape essentielle avant l’entraînement des modèles de Machine Learning consiste à **normaliser les variables numériques**. Cette opération vise à ramener toutes les données à une **échelle comparable**, afin d’éviter que certaines variables, par leur magnitude, n’aient une influence excessive sur l'apprentissage des modèles.

En particulier, dans le cadre de ce mémoire, les variables comme :

* les **prix des composants photovoltaïques** (exprimés en $/Wp ou RMB/Wp),
* les **cours journaliers des matières premières** (en $/tonne, €/g, etc.),
* les **taux de change**,
* ou encore les **coûts logistiques** (transport maritime en $/conteneur),

peuvent présenter des **ordres de grandeur très différents**. Cette disparité peut induire un **biais** lors de l'entraînement, en particulier pour les algorithmes sensibles à l’échelle des variables (régression linéaire, k-plus proches voisins, réseaux de neurones, etc.).

**🔹 Méthode utilisée**

Deux méthodes de mise à l’échelle sont communément utilisées :

1. **Standardisation (Z-score)** :  
   Transforme chaque valeur en soustrayant la moyenne et en divisant par l’écart-type :

xstandardiseˊ=x−μσx\_{\text{standardisé}} = \frac{x - \mu}{\sigma}xstandardiseˊ​=σx−μ​

1. **Min-Max Scaling** :  
   Ramène les valeurs dans un intervalle fixe (souvent [0, 1]) :

xnorm=x−xmin⁡xmax⁡−xmin⁡x\_{\text{norm}} = \frac{x - x\_{\min}}{x\_{\max} - x\_{\min}}xnorm​=xmax​−xmin​x−xmin​​

Dans ce travail, la **standardisation** a été préférée, car elle permet de conserver la structure de distribution (surtout utile en cas de données ayant une distribution normale ou quasi-normale).

**🔹 Application dans le code**

L’opération a été réalisée à l’aide de la bibliothèque scikit-learn :

python

CopierModifier

from sklearn.preprocessing import StandardScaler

# Sélection des colonnes à normaliser

features = ['Prix Cellule', 'Prix Module', 'Prix Silicium', 'Prix Aluminium', 'Taux de Change']

scaler = StandardScaler()

scaled\_features = scaler.fit\_transform(df[features])

# Création d’un nouveau DataFrame avec les données standardisées

df\_scaled = pd.DataFrame(scaled\_features, columns=features)

df\_scaled['Date'] = df['Date'].values

**✅ Résultat**

Les variables sont désormais sur une **échelle standardisée**, avec une **moyenne nulle** et un **écart-type unitaire**. Cela garantit que :

* aucune variable ne domine l’apprentissage,
* les modèles convergent plus rapidement,
* les résultats sont plus stables et interprétables.

Cette phase de normalisation constitue la dernière étape du prétraitement des données avant leur injection dans les modèles de prédiction.