Pour la transformation des données après avoir terminé le scraping et enregistré les données dans un fichier CSV, voici les étapes importantes à suivre :

**1. Nettoyage des données :**

* **Suppression ou gestion des valeurs manquantes** : Identifier les lignes où certaines valeurs sont manquantes (par exemple, si une cryptomonnaie n’a pas de prix ou de volume d’échange). Tu peux soit :
  + Remplir les valeurs manquantes avec une valeur par défaut (par exemple, 0 pour le volume d’échange).
  + Supprimer les lignes ou colonnes ayant trop de valeurs manquantes.
  + Utiliser une technique comme **l’imputation** pour estimer les valeurs manquantes (par exemple, utiliser la moyenne ou la médiane des autres valeurs similaires).
* **Correction des types de données** : Assure-toi que chaque colonne a le bon type de données (par exemple, les prix et les volumes doivent être de type float, les dates de type datetime, etc.).

python

Copier le code

df['price'] = pd.to\_numeric(df['price'], errors='coerce')

df['timestamp'] = pd.to\_datetime(df['timestamp'])

* **Traitement des valeurs aberrantes (outliers)** : Identifie les valeurs qui semblent anormalement élevées ou faibles, comme un prix de 0 ou un volume de millions pour une cryptomonnaie peu connue. Tu peux :
  + Utiliser des statistiques (comme l’écart-type) pour détecter les valeurs anormales.
  + Remplacer ou exclure ces valeurs.

**2. Normalisation ou standardisation :**

* **Normalisation des colonnes numériques** : Si tu veux comparer les cryptomonnaies entre elles, tu peux normaliser les colonnes comme le prix, la capitalisation boursière et le volume d’échange pour les ramener à une même échelle (par exemple, entre 0 et 1).

python

Copier le code

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

df[['price', 'var\_cap', 'var\_volume\_day']] = scaler.fit\_transform(df[['price', 'var\_cap', 'var\_volume\_day']])

**3. Enrichissement des données :**

* **Calcul des variations de prix** : Ajoute des colonnes pour calculer les variations du prix sur des périodes spécifiques (1 jour, 1 semaine, 1 mois) si elles ne sont pas déjà présentes dans ton jeu de données. Par exemple, tu peux calculer les variations en pourcentage :

python

Copier le code

df['variation\_7\_days'] = (df['price'] - df['price'].shift(7)) / df['price'].shift(7) \* 100

df['variation\_30\_days'] = (df['price'] - df['price'].shift(30)) / df['price'].shift(30) \* 100

* **Indicateurs financiers supplémentaires** :
  + **Moyenne mobile** : Calcule une moyenne mobile pour le prix sur une période donnée (par exemple, 7 jours, 30 jours) pour lisser les fluctuations.

python

Copier le code

df['moving\_avg\_7d'] = df['price'].rolling(window=7).mean()

df['moving\_avg\_30d'] = df['price'].rolling(window=30).mean()

* **Volatilité** : Calcule la volatilité des prix (écart-type des variations de prix sur une période donnée).

python

Copier le code

df['volatility\_7d'] = df['price'].rolling(window=7).std()

**4. Ajout de nouvelles colonnes :**

* **Date et heure** : Si ton timestamp est en format datetime, extrais les composants (jour, mois, année, heure) pour faciliter les analyses temporelles.

python

Copier le code

df['year'] = df['timestamp'].dt.year

df['month'] = df['timestamp'].dt.month

df['day'] = df['timestamp'].dt.day

* **Classement** : Classe les cryptomonnaies en fonction de la capitalisation boursière, du prix ou du volume d’échange.

python

Copier le code

df['rank\_by\_market\_cap'] = df['var\_cap'].rank(ascending=False)

**5. Agrégation des données (facultatif) :**

* **Groupement des données** : Si tu collectes des données plusieurs fois par jour, tu peux agréger les résultats pour obtenir une vue quotidienne, hebdomadaire, ou mensuelle. Par exemple, tu pourrais regrouper les données par jour et calculer les moyennes ou les médianes pour chaque journée :

python

Copier le code

daily\_data = df.groupby(df['timestamp'].dt.date).agg({

'price': 'mean',

'var\_cap': 'mean',

'var\_volume\_day': 'mean'

})

**6. Sauvegarde des données transformées :**

* Une fois les transformations effectuées, tu peux sauvegarder les données enrichies dans un nouveau fichier CSV ou les stocker dans une base de données pour une analyse plus poussée.

python

Copier le code

df.to\_csv('transformed\_crypto\_data.csv', index=False)

**7. Visualisation des tendances :**

* Tu peux utiliser des bibliothèques comme **Matplotlib**, **Seaborn** ou **Plotly** pour visualiser les tendances des prix, la capitalisation boursière, ou la volatilité sur plusieurs périodes.

**8. Intégration dans un pipeline :**

* Après la transformation des données, tu peux intégrer le processus dans un pipeline automatisé avec **Apache NiFi** ou **Airflow** pour que tout soit exécuté régulièrement et automatiquement.

Ces étapes vont non seulement enrichir tes données, mais également te permettre d'obtenir des informations exploitables à partir de tes données de cryptomonnaies. Cela rendra ton projet plus impressionnant et utile pour le montrer en tant qu'ingénieur de données.