Pratique de la data science

March 14, 2025

1 TP 2 - Clustering

Librairies à installer :

```
import yfinance as yf
import pandas as pd
from datetime import datetime
import numpy as np

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering, SpectralClustering
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression

from sklearn.manifold import TSNE
from scipy.spatial.distance import cdist
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
from scipy.spastial.distance import squareform, pdist
import glob
```

Objectif: Utiliser les méthodes de clustering sur les datasets créés en TP1.

Tout doit être codé sous forme de fonctions. L'objectif est de pouvoir comparer l'utilisation de chaque algorithme de clustering sur différents types de données (profils de risques, de performance...)

Pour chaque type de données étudié : Fournir la performance de chaque algorithme et des visualisations.

1.1 Financial profiles clustering

Objectif : Identifier les groupes d'entreprises aux caractéristiques financières similaires (Valorisation, performances financière...) $\implies$ Peut permettre de créer des portefeuilles diversifiés.

Algorithme : K-MEANS.

Fonctions:

- `StandarScaler()` pour standardiser les données.
- `KMeans()` : Regarder documentation sur sklearn.
- `TSNE()` pour permettre la visualisation des résultats.

Pipeline:

Pre processing: `def preprocess_for_financial_clustering()`

1. Charger les ratios financiers obtenus grâce au scrapping dans un dataframe.
2. Extraire dans un nouveau dataframe les données pertinentes pour une analyse de performances financières (ForwardPE, Beta, Price to Book, Return on equity....)
3. Nettoyer les NA (Basique: on supprime les entreprises qui présentent des NA sur ces données).
4. Standardiser les données.

Déterminer le nombre de clusters: `def elbow_method(data)`

1. Utiliser KMeans en faisant varier le nombre de clusters K.
2. Analyser l'inertie en fonction de K (`.inertia_`).
3. Plot les inerties et trouver le "coude".

KMeans et visualisation: `def do_kmeans_clustering()`

1. Appliquer Kmeans avec nombre de clusters décidé.
2. Rajouter la colonne "clusters" au dataframe.
3. Analyser les caractéristiques de chaque cluster et plot une représentation TSNE (Entreprises colorées par clusters).

1.2 Risk profiles clustering

Objectif : Identifier les groupes d'entreprises aux profils de risques (financiers et opérationnels) similaires (endettement, liquidité, rentabilité...) $\implies$ Peut permettre d'ajuster les allocations d'actifs.

Algorithme : Hierarchical Clustering.

Fonctions:

- `AgglomerativeClustering()` : Regarder documentation sur sklearn.
- `linkage` : Calcul des distances entre observations.
- `dendrogram` : Plot structure hiérarchique (cf documentation).
- **NB:** Linkage et AgglomerativeClustering() permettent tous les deux d'effectuer un clustering hierarchique mais présentent une légère différence:
  - `linkage` renvoie plutôt des distances $\implies$ Permet de tracer le dendogram.
  - `AgglomerativeClustering()` renvoie directement les clusters.

Pipeline:

Pre processing et détermination de nombres de clusters (cf Partie 1.2, en sélectionnant cette fois-ci les variables pertinentes pour une analyse des profils de risques)

Hierchical clustering (`def do_hierarchical_clustering()`) et visualisation :

1. Appliquer hierarchical clustering (`AgglomerativeClustering(...)`
2. Rajouter colonne "clusters" et analyser charactéristiques.
3. Plot le dendogram (`def plot_dendrogram()`)
   - Calcul des distances (`linked = linkage(data, method='ward')`
   - plot ces distances (`dendogramm(...)`

1.3 Daily returns correlations clustering

Objectif : Regrouper les entreprises en fonction de la similarité de leurs évolutions boursières $\Longrightarrow$ Peut permettre de mieux diversifier son portefeuille.

Algorithme : Hierarchical Clustering.

Fonctions :

- `filepaths = glob.glob("companieshistoric/*.csv")` : Permet d'accéder à plusieurs fichiers d'un répertoir (*.csv signifie que l'on prend tous les fichiers csv)

Pipeline:

Préparation des données (Rendements des entreprises):

1. Créer un dictionnaire (vide) qui va contenir les rendements de chaque entreprise.
2. Pour chaque entreprise, charger la colonne "Daily Return" et l'ajouter au dictionnaire (**clé** : company_name, **valeur** : colonne daily_return).
3. Transformer le dictionnaire en dataframe "**returns_df**".
4. Nettoyer les valeurs manquantes (Basique: avec moyenne par exemple).

Hierachical clustering:

1. Extraire les corrélations entre les rendements des entreprises (`returns_df.corr()`)
2. Appliquer le hierachical clustering sur cette matrice de corrélations et plot le dendogram.

1.4 Evaluation et comparaison des algorithmes

1. Implémenter l'algorithme DBSCAN (`def do_hierarchical_clustering()`).
2. Utiliser les trois algorithmes (KMEANS, Hierarchical, DBSCAN) sur our les trois critères (finance, risk, rendements).
3. **Silhouette score:** Fournir un tableau avec silhouette score pour chaque algorithme sur chaque type de données. Analyser.