

NHÓM 22

PROJECT - PYTHON

# COFFEE SHOP

Lưu Nguyễn Ngọc Hân - 23280055

Vũ Thị Thanh Hà - 23280053

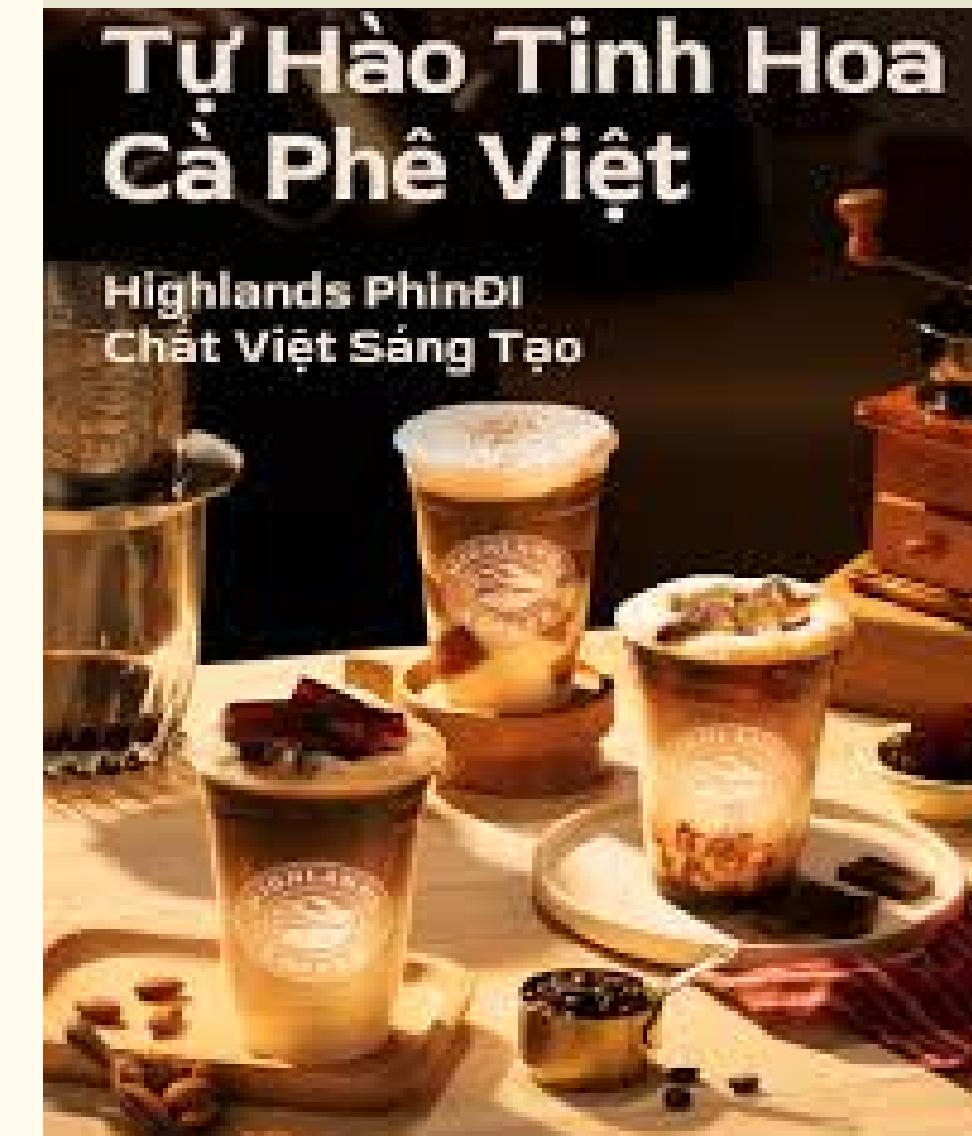
Lê Ái Như - 23280024

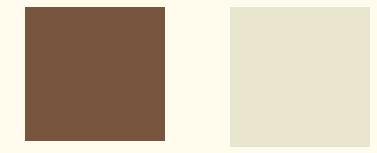
> 10 DECEMBER, 2025



- ▶ Giới thiệu bài toán
- ▶ Giới thiệu dữ liệu
- ▶ Giới thiệu các Class và Phương thức
- ▶ Thư viện mô hình máy học và tối ưu tham số
- ▶ Phân tích kết quả thí nghiệm
- ▶ Kết luận

# MỤC LỤC





# GIỚI THIỆU BÀI TOÁN

## 1. Bối cảnh:

- Dự đoán phân khúc khách hàng của Highlands Coffee

## 2. Mục tiêu:

- Kinh doanh
- Kỹ thuật

## 3. Phạm vi:

- Cơ bản, không bao gồm triển khai hệ thống, tối ưu nâng cao.



# Nguồn dữ liệu

Business Intelligence 9



## Cấu trúc dữ liệu

- Thông tin khách hàng
- Hành vi và nhu cầu
- Nhận biết và mức độ sử dụng thương hiệu
- Chỉ số và hình ảnh thương hiệu
- Chi tiêu và tần suất
- Phân khúc và nhãn mục tiêu

## Tiền xử lý dữ liệu

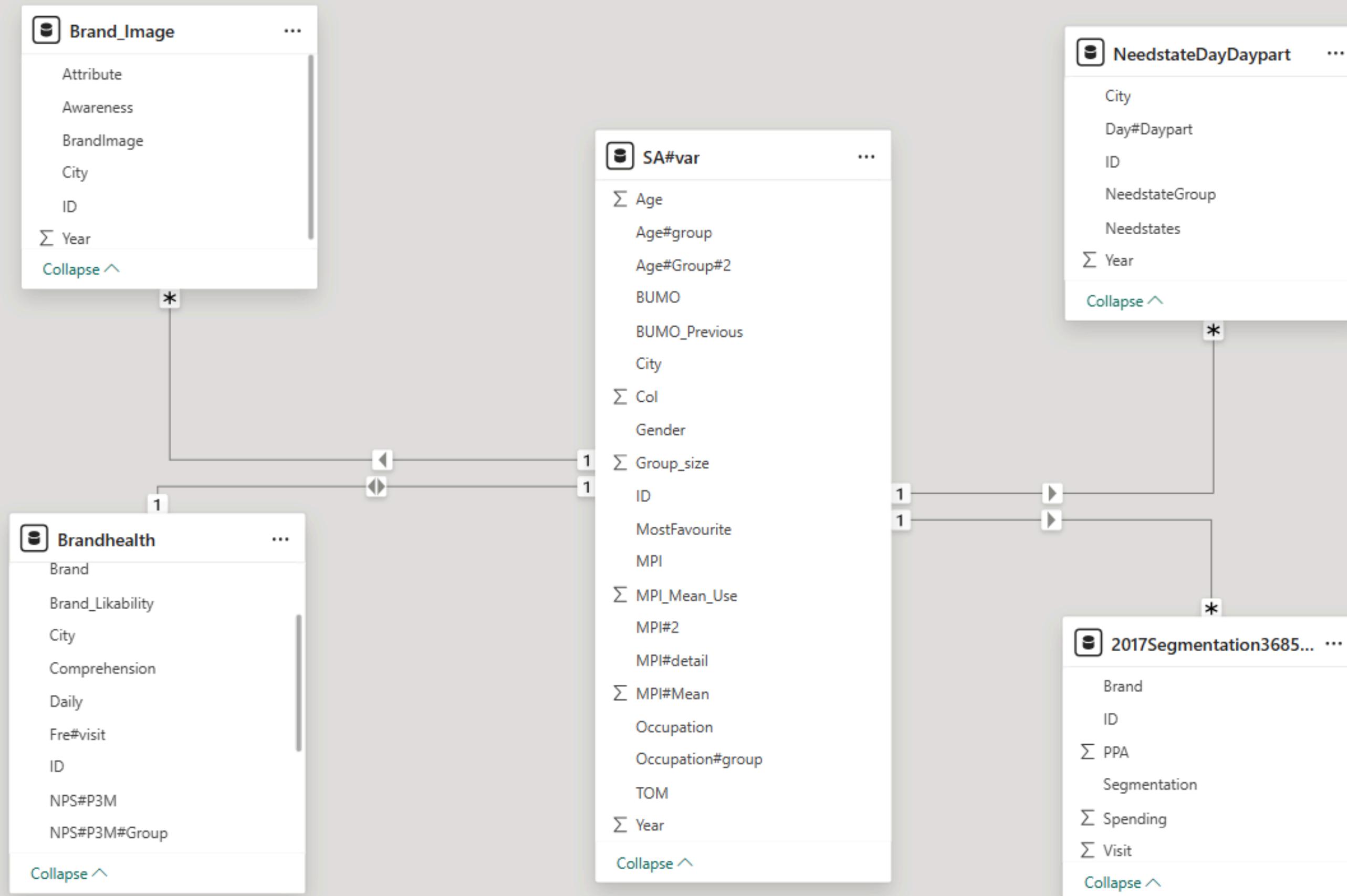
- CustomerSegmentation\_2017
- Brand\_Image
- BrandHealth
- SA#Var
- Needstate



GIỚI  
THIỆU  
DỮ  
LIỆU



# LIÊN HỆ GIỮA CÁC BẢNG DỮ LIỆU





PROJECT - PYTHON

## Kiến trúc code

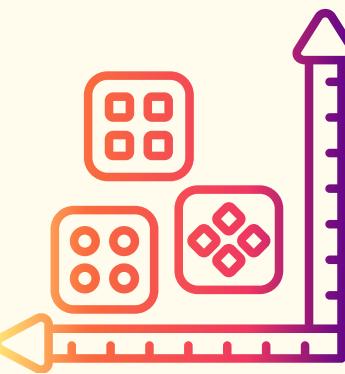
- Khối dữ liệu (data/)
- Khối tiền xử lý (src/preprocessing/)
- Khối mô hình (src/models/)
- Khối trực quan hóa(src/visualization/)
- Quy trình pipeline tổng quát

## Class

- Class DataLoader
- Class DataCleaner
- Class FeatureEngineering
- Class LogTransformer
- Class EncoderConfig
- Class FeatureEncoder



# GIỚI THIỆU CLASS VÀ PHƯƠNG THỨC

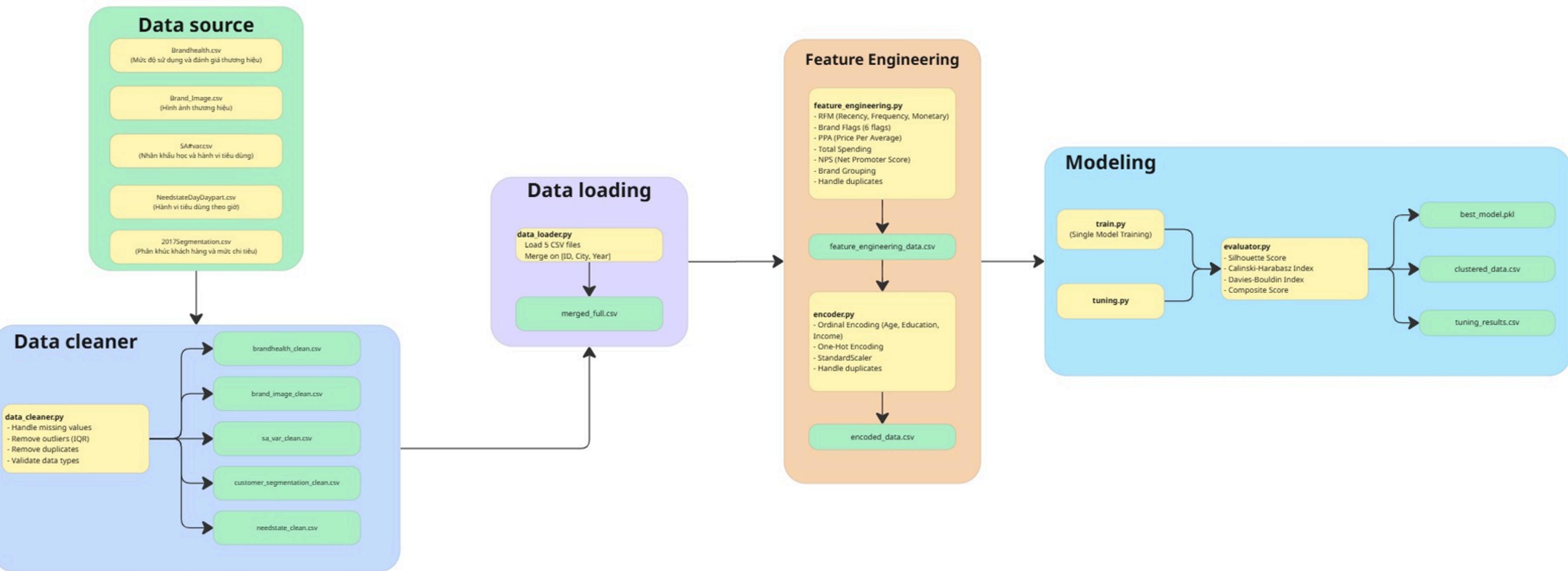


# Luồng hoạt động

- Workflow xử lý

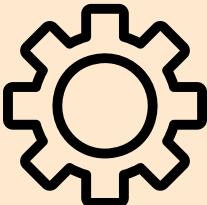


PROJECT - PYTHON



# DATALOADER

đọc dữ liệu từ thư mục **data/raw/**, quản lý nhiều bảng dữ liệu và cung cấp các phương thức hợp nhất các bảng.



## `__init__()`

- Khởi tạo lớp với đường dẫn dữ liệu
- Thiết lập danh sách tên file và separator tương ứng
- Tạo các cấu trúc lưu trữ (`self.data` và `self.merged_df`)



## `_read_csv(file_name, key)`

- Đọc file CSV từ thư mục dữ liệu **thô/cleaned**.
- Kiểm tra file có tồn tại, nạp nội dung bằng `pandas.read_csv()`
- Ghi log quá trình đọc và quản lý dữ liệu trong lớp.



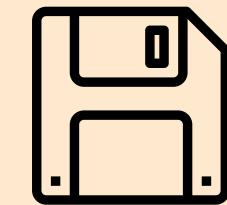
## `load_all()`

- Ghi log bắt đầu quá trình tải dữ liệu.
- Duyệt qua danh sách file trong `self.file_names` và gọi `_read_csv()` cho từng file.
- Lưu toàn bộ DataFrame vào `self.data`.
- Ghi log số lượng file đã tải thành công và trả về toàn bộ dữ liệu.



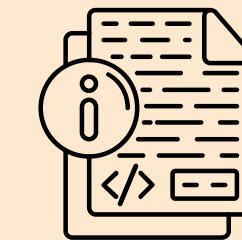
## `merge_all()`

- Lọc bảng **SA\_var** chỉ lấy dữ liệu năm 2017 để đồng nhất với các bảng còn lại.
- Kiểm tra sự tồn tại của các cột bắt buộc: **ID, City, Year**.
- Loại bỏ dòng **trùng lặp** trong các bảng trước khi merge.
- Loại các cột **PPA, Spending** trong Brandhealth vì sẽ lấy bản chuẩn từ Segmentation.
- Merge tuần tự theo chiến lược.
- Làm sạch dữ liệu sau Merge.
- Lưu DataFrame cuối cùng vào `self.merged_df`.



## `save_merged(out_path=None)`

- Kiểm tra xem đã có dữ liệu merge trong `self.merged_df` hay chưa.
- Tạo thư mục đầu ra nếu chưa tồn tại.
- Ghi toàn bộ bảng dữ liệu đã merge xuống file CSV.
- Ghi log quá trình lưu file.



## `get_merged_info()`

- Kiểm tra sự tồn tại của `self.merged_df`, sau đó trả về kích thước của DataFrame dưới dạng shape.

# DATACLEANER

làm sạch và tiền xử lý từng bảng dữ liệu khảo sát.

## `__init__()`

Khởi tạo đối tượng DataCleaner.

## `_normalize_string_columns(df, cols)`

Chuẩn hóa các cột dạng chuỗi trong DataFrame bằng cách loại bỏ khoảng trắng ở đầu và cuối.

## `_winsorize_series`

### `(s, lower_q=0.01, upper_q=0.99)`

Thực hiện **Winsorization** cho một biến số (Series) bằng cách giới hạn giá trị của nó trong khoảng giữa hai quantile. Mục tiêu là giảm ảnh hưởng của các outlier mà vẫn giữ nguyên số lượng mẫu.

## `_map_income_band_from_mean(x)`

Phương thức giúp chuyển đổi dữ liệu thu nhập liên tục thành các “**income band**” để dễ phân tích, phân khúc và trực quan hóa.

## `_fix_brand_spelling(value)`

Chuẩn hóa và sửa lỗi chính tả phổ biến trong tên thương hiệu (**brand**).

## `clean_customer_segmentation(df)`

Làm sạch bảng dữ liệu  
**Customer Segmentation**

## `clean_brand_image(df)`

Làm sạch bảng dữ liệu **Brand Image**  
→ chuẩn bị dữ liệu trước khi phân tích nhận diện thương hiệu hoặc hợp nhất với các bảng khác

## `clean_brandhealth` (`df, brand_image_df=None`)

Làm sạch toàn bộ bảng **Brand Health** với nhiều bước xử lý phức tạp.  
→ Đây là phương thức **quan trọng nhất** trong pipeline làm sạch Brand Health.

## `clean_sa_var(df)`

Làm sạch và chuẩn hóa bảng **SA Variables**.

## `clean_needstate(df)`

Làm sạch và chuẩn hóa bảng **Needstate**.

## `clean_all(dfs)`

Thực thi toàn bộ quy trình làm sạch cho tất cả các datasets: **Customer Segmentation, Brand Image, Brand Health, SA Variables** và **Needstate**.  
→ Đóng vai trò **điều phối, tự động gọi các phương thức làm sạch** tương ứng.

## `save_cleaned` (`dfs_clean, base_dir`)

Lưu toàn bộ các DataFrames sạch ra file CSV trong thư mục chỉ định.  
→ đảm bảo thư mục tồn tại và tự động tạo nếu chưa có.

# FEATUREENGINEERING / tạo ra các biến dẫn xuất cho việc phân cụm

## `__init__(df)`

Khởi tạo đối tượng **FeatureEngineering** với dữ liệu đầu vào và cấu hình sẵn các danh sách brand được giữ lại cùng bộ ánh xạ flag cho Highlands Coffee.

## `_add_brand_grouping()`

Nhóm các cột liên quan đến brand (**TOM, BUMO, BUMO\_Previous, MostFavourite**) thành các nhóm brand, giữ nguyên các brand có trong danh sách **self.keep** và gán các giá trị còn lại vào nhóm "**Others**".

## `_add_brand_flags()`

Tạo các **cột cờ (flag)** cho Highlands Coffee dựa trên các cột gốc trong **flag\_map**. Nếu giá trị trong cột gốc bằng "Highlands Coffee" thì cột flag sẽ nhận giá trị **1**, ngược lại là **0**.

## `_add_brand_loyalty()`

Tính chỉ số trung thành thương hiệu (**Brand\_Loyalty**) từ 0–4 dựa trên sự trùng khớp giữa các cột nhóm thương hiệu (**TOM\_Group, BUMO\_Group, BUMO\_Previous\_Group, MostFavourite\_Group**) và các giá trị gốc. Mỗi điều kiện trùng khớp cộng 1 điểm.

## `_add_brand_switcher()`

Xác định **hành vi đổi thương hiệu** bằng cách kiểm tra sự khác nhau giữa **BUMO\_Group & BUMO\_Previous\_Group**. Nếu khác nhau, giá trị = **1**, nếu giống nhau, giá trị = **0**.

## `_add_funnel_depth()`

Tính chỉ số **Funnel\_Depth** (0–7) cho mỗi khách hàng bằng cách tổng hợp các flag về **Awareness, Trial, P3M, P1M, Weekly, Daily** liên quan đến Highlands Coffee.  
→ Giá trị **càng cao** thể hiện mức độ gắn bó và trải nghiệm thương hiệu sâu hơn.

## `_add_awareness_usage_gap()`

Tính khoảng cách giữa mức độ nhận biết thương hiệu và **mức độ dùng thử (Awareness – Trial)** cho Highlands Coffee.  
Giá trị **dương** → biết mà chưa thử.  
Giá trị **âm** → Awareness = Trial

## `_add_need_is_drinking()`

Tạo cột chỉ báo (**Need\_is\_Drinking**) xác định khách hàng có nhu cầu liên quan đến uống đồ uống hay không. Needstates chứa từ "Drinking" giá trị là **1**, ngược lại là **0**.

## `build_all()`

Thực hiện tuần tự tất cả các bước **feature engineering** trên DataFrame, bao gồm: **brand grouping, brand flags, brand loyalty, brand switcher, funnel depth, awareness-usage gap, và Need\_is\_Drinking**.

# LOGTRANSFORMER CLASS

## Công dụng:

- Chuyển các giá trị âm (thường là -1 biểu thị missing) thành 0.
- Áp dụng log1p transformation để **giảm skew và tránh log(0)**.
- Hỗ trợ **inverse\_transform** để hồi lại giá trị gốc.

## Phương thức:

### **fit(X, y=None)**

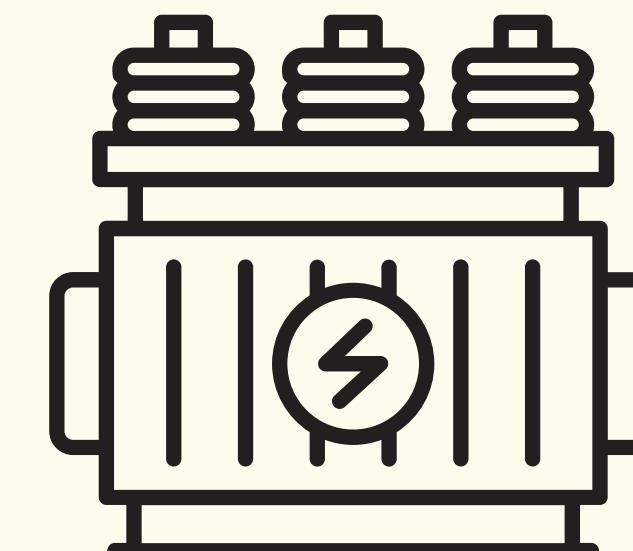
Không làm gì, chỉ trả về chính instance.

### **transform(X)**

Thực hiện **log1p transformation**.

### **inverse\_transform(X)**

Hồi lại dữ liệu gốc từ log1p.



# ENCODERCONFIG CLASS

## Công dụng:

Lưu cấu hình để khi tạo FeatureEncoder, các tham số scale, encode và xử lý unknown được áp dụng nhất quán.

## Phương thức:

### scaler\_type (str)

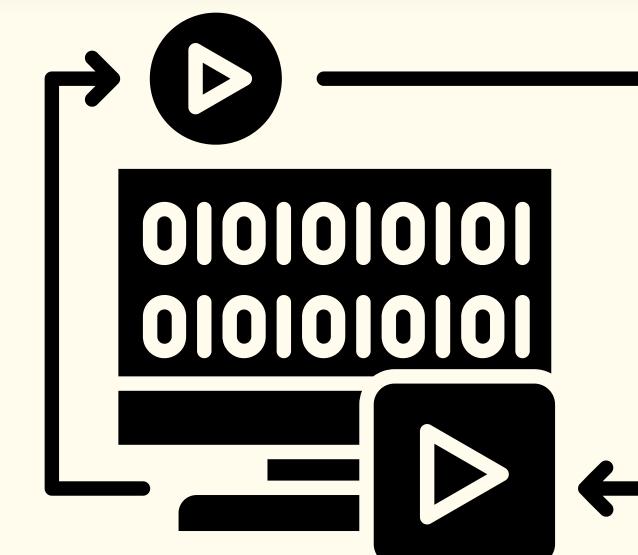
Kiểu scaler dùng cho biến numeric, có thể là 'standard' (z-score) hoặc 'minmax' (0-1). Mặc định là '**standard**'.

### handle\_unknown (str)

Cách xử lý giá trị chưa xuất hiện trong dữ liệu categorical.  
Mặc định là '**ignore**'

### sparse\_output (bool)

Xác định output của encoder có phải sparse matrix hay không.  
Mặc định là **False**.



# FEATUREENCODER

mã hóa cho toàn bộ dữ liệu sau feature engineering, chuẩn bị đầu vào cho các thuật toán phân cụm.

## `__init__()`

- Tạo một instance của **FeatureEncoder**.
- Thiết lập cấu hình encoder dựa trên **EncoderConfig**.
- Khởi tạo các thuộc tính nội bộ `preprocessor_` và `feature_names_` bằng `None`.

## `_get_ordinal_categories()`

- Xác định thứ tự các category cho từng biến ordinal, đảm bảo encoding phản ánh đúng **quan hệ thứ tự tự nhiên**.
- Được sử dụng trong pipeline khi fit `OrdinalEncoder`.

## `_build_preprocessor()`

- Xây dựng một **ColumnTransformer** kết hợp các pipeline xử lý cho từng nhóm cột.
- Chuẩn bị dữ liệu đầu vào cho các thuật toán clustering.

## `_prepare_dataframe(df)`

Chuẩn bị DataFrame đầu vào bằng cách loại bỏ các cột không cần thiết, chuẩn bị cho bước encoding.

## `_extract_feature_names()`

Trích xuất danh sách tên feature sau khi dữ liệu đã được transform bằng **preprocessor\_ (ColumnTransformer)**.

## `fit_transform(df)`

Fit encoder trên DataFrame đầu vào và transform dữ liệu sang dạng numeric chuẩn hóa.

## `transform(df)`

Sử dụng encoder đã fit để **transform** dữ liệu mới, áp dụng các bước encoding và scaling giống như dữ liệu huấn luyện

## `save(filepath)`

Lưu toàn bộ object FeatureEncoder, bao gồm cả `preprocessor_` và `feature_names_`, để tái sử dụng cho các dữ liệu mới mà không cần fit lại.

## `load(filepath)`

Tải lại FeatureEncoder đã lưu từ file .pkl, cho phép sử dụng encoder mà không cần fit lại.

## `get_feature_names()`

Trả về danh sách tên các feature sau khi dữ liệu đã được encode.



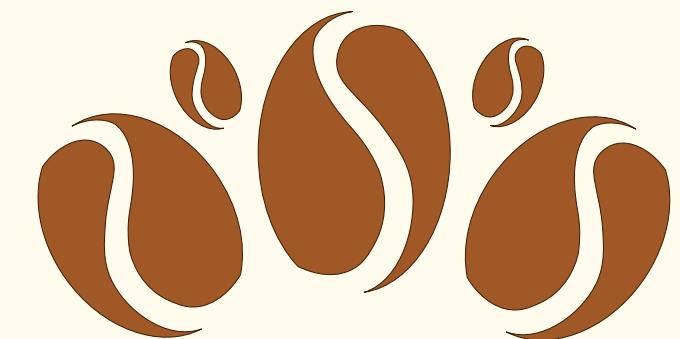
# TRAININGCONFIG CLASS

## Công dụng:

Cấu hình này chuẩn hóa tất cả thông số cần thiết cho huấn luyện mô hình clustering, bao gồm:

- dữ liệu,
- loại model,
- số cụm,
- seed,
- các feature,
- đường dẫn lưu kết quả
- model,
- hyperparameters cố định,

→ Giúp code gọn gàng, dễ bảo trì và tái sử dụng.



# MODELTRAINER

huấn luyện các mô hình phân cụm,  
đánh giá kết quả và lưu model đã huấn luyện.

## **`__init__()`**

Khởi tạo class ModelTrainer.

## **`train_model()`**

Huấn luyện mô hình clustering.

## **`load_model(path)`**

Tải mô hình clustering từ file pickle.

## **`load_data()`**

Dùng để tải dữ liệu đã encode từ CSV.

## **`build_model(n_clusters=None)`**

Khởi tạo mô hình clustering.

## **`evaluate()`**

Đánh giá mô hình clustering hiện tại.

## **`save_model(path=None)`**

Lưu mô hình clustering đã  
huấn luyện ra file.

## **`get_cluster_labels()`**

Lấy nhãn của các cụm từ mô hình  
clustering.

## **`save_labels(output_path)`**

Lưu nhãn các cụm ra file CSV.



# CLUSTERINGEVALUATOR

đánh giá chất lượng mô hình phân cụm.

## `__init__()`

Khởi tạo class ClusteringEvaluator.

## `evaluate_once()`

Đánh giá một mô hình phân cụm duy nhất theo bộ ba chỉ số

**Silhouette Score,**  
**Calinski–Harabasz Score,**  
**Davies–Bouldin Index**

## `evaluate_many()`

Đánh giá nhiều mô hình phân cụm cùng lúc dựa trên tập nhãn đầu ra của từng mô hình.

## `save_results()`

Lưu toàn bộ kết quả đánh giá clustering vào một file CSV.

## `plot_scores()`

Trực quan hóa kết quả đánh giá mô hình phân cụm.

## `plot_all_scores()`

Trực quan hóa đồng thời ba chỉ số trong cùng một biểu đồ dạng subplots.

## `get_best_model()`

Lựa chọn mô hình phân cụm tốt nhất dựa trên một metric đánh giá cụ thể.

## `clear_results()`

Xóa toàn bộ kết quả đánh giá đã được lưu trong evaluator.



# TUNINGCONFIG

Lớp cấu hình dành cho module Hyperparameter Tuning.

## Thuộc tính:

### **data\_path (str):**

đường dẫn đến file dữ liệu  
đã encode.

### **results\_path (str)**

đường dẫn lưu kết quả  
tuning dạng CSV.

### **random\_state (int)**

seed đảm bảo tính tái lập.

### **metric\_selection (str)**

chọn mô hình tốt nhất

## Công dụng:

Lưu trữ toàn bộ cấu hình cần thiết cho quá trình tuning, được truyền  
vào class ModelTuner để **điều khiển quá trình chạy thử** nhiều  
mô hình/hyperparameter khác nhau và chọn ra mô hình tối ưu.



# HYPERPARAMETER TUNER

tự động thử nghiệm nhiều bộ  
siêu tham số khác nhau  
cho các mô hình phân cụm.

## \_\_init\_\_()

Khởi tạo đối tượng **HyperparameterTuner**

## load\_data()

Dùng để tải dữ liệu.

## \_generate\_param\_grid (**model\_type**, **param\_grid**)

- Sinh toàn bộ tổ hợp tham số từ **param\_grid**.
- Gán thêm trường **model\_type** cho mỗi bộ tham số.

## run\_grid\_search

### (**model\_type**, **param\_grid**)

Grid Search toàn diện cho một thuật toán phân cụm với tập tham số cho trước.

## run\_all\_models()

Chạy toàn bộ quá trình **Grid Search** cho tất cả các thuật toán phân cụm.

## \_calculate\_composite\_score (**df: pd.DataFrame**) -> **pd.Series**

Tính toán **composite score** dựa trên 3 metrics đã chuẩn hóa.

## get\_summary()

Tổng hợp và sắp xếp kết quả của toàn bộ quá trình dò tìm siêu tham số.

## save\_results()

Lưu toàn bộ kết quả quá trình tìm **hyperparameter** vào file và in ra các mô hình tốt nhất.

## save\_best\_model\_and\_df()

Lưu mô hình tốt nhất và DataFrame kèm nhãn phân cụm.



# THƯ VIỆN MÔ HÌNH MÁY HỌC VÀ TỐI ƯU THAM SỐ



## Thư viện mô hình máy học

- scikit - learn
- HDBSCAN
- NumPy
- Pandas

## Các mô hình sử dụng

- KMeans Clustering
- Gaussian Mixture Model (GMM)
- DBSCAN (Density-Based Spatial Clustering)
- HDBSCAN (Hierarchical DBSCAN)

## Thư viện tối ưu tham số

- GridSearchCV
- Không gian tìm kiếm hyperparameter
- Thư viện hỗ trợ khác
  - joblib
  - itertools
  - tqdm

# PHÂN TÍCH KẾT QUẢ THÍ NGHIỆM



Kết quả mô hình tốt nhất

Metric	Giá trị
Model	KMeans
Số cụm (n_clusters)	3
Silhouette Score	0.2544
Calinski-Harabasz Index	1269.22
Davies-Bouldin Index	1.5338
Composite Score	0.7018

# PHÂN TÍCH KẾT QUẢ THÍ NGHIỆM



	Hành vi	Thái độ	Vị trí	Insight
<b>Cluster 0</b> Nhóm trung niên ghé nhiều - chi ít	<ul style="list-style-type: none"><li>Tần suất ghé <b>cao nhất</b></li><li>PPA thấp</li><li>Tổng chi thấp</li><li>Gia trị đơn hàng nhỏ</li></ul>	<ul style="list-style-type: none"><li>Mức độ hài lòng <b>kém</b></li><li>Ưa chuộng quán địa phương</li><li>Ít khả năng giới thiệu</li></ul>	Cần Thơ	Tập trung vào <b>tăng engagement</b> và tạo giá trị cảm nhận.
<b>Cluster 1</b> Nhóm trẻ chi tiêu cao hài lòng cao	<ul style="list-style-type: none"><li>Tần suất ghé thấp</li><li>PPA <b>cao nhất</b></li><li>Tổng chi <b>thấp nhất</b></li><li>Mua combo hoặc sản phẩm cao cấp</li></ul>	<ul style="list-style-type: none"><li>Mức độ hài lòng cao</li><li>Thương hiệu <b>yêu thích</b></li><li>Khả năng giới thiệu cao</li><li>Khách hàng trung thành</li></ul>	Hà Nội	Phân khúc " <b>vàng</b> " cần ưu tiên trong marketing và loyalty program
<b>Cluster 2</b> Tuổi trung bình chi tiêu thấp hài lòng cao	<ul style="list-style-type: none"><li>Tần suất ghé tb - thấp</li><li>PPA thấp</li><li>Tổng chi thấp</li><li>Sinh viên hoặc nhóm thu nhập hạn chế</li></ul>	<ul style="list-style-type: none"><li>Mức độ hài lòng <b>cao nhất</b></li><li>Vẫn thích Independent Cafe</li><li>Có khả năng thành khách hàng trung thành</li></ul>	TP.HCM	Chiến lược <b>giá (pricing)</b> và chương trình <b>khuyến mãi</b> phù hợp.



Coffee ☕

# KẾT LUẬN

## Tóm tắt

- Tổng kết

## Hướng phát triển

- Mở rộng dữ liệu
- Thủ mô hình khác
- Tối ưu thêm



# THANK YOU

PROJECT - PYTHON

