Tuần 4 - Ước tính giá nhà và Trợ lý AI định giá bất động sản.

Time-Series Team

Ngày 1 tháng 11 năm 2025

Dự án Hệ thống dự đoán giá nhà gồm các nội dung chính:

- ***Đặt vấn đề***
- ***Mở rộng của nhóm***
- ***Giới thiệu chi tiết mở rộng của nhóm***

Mục lục

I. Đặt vấn đề

Trong tài liệu gốc của dự án *House Price Prediction – Advanced Regression Techniques*, pipeline được thiết kế theo hướng tuyến tính: từ giai đoạn **khám phá dữ liệu (EDA)**, **làm sạch**, **tiền xử lý**, đến **huấn luyện các mô hình hồi quy tuyến tính và phi tuyến cơ bản** như Linear, Polynomial, Ridge và Lasso Regression. Quy trình này phù hợp cho mục tiêu học tập, nhưng khi triển khai trong thực tế, nhóm nhận thấy tồn tại nhiều hạn chế:

- **Thiếu cơ chế chọn đặc trưng có hệ thống**: pipeline gốc sử dụng toàn bộ tập biến, dễ dẫn đến đa cộng tuyến và giảm hiệu suất.
- **Chưa kiểm soát overfitting ở cấp độ đặc trưng**: chỉ áp dụng điều chuẩn ở mức mô hình, không sàng lọc sớm các biến nhiễu.
- **Không đánh giá ảnh hưởng của PCA hoặc giảm chiều dữ liệu**.
- **Thiếu khả năng diễn giải**: pipeline dừng lại ở so sánh chỉ số R^2 và $RMSE$, chưa áp dụng các phương pháp giải thích mô hình.
- **Không có hướng mở rộng ứng dụng thực tế**: chưa kết nối mô hình với hệ thống định giá tự động.

1 Cải tiến trong pipeline mới

Pipeline mới được nhóm đề xuất theo định hướng **kết hợp giữa khả năng tự động hóa, phân tích đa chiều và giải thích mô hình**. Các cải tiến chính bao gồm:

(a) **Chọn đặc trưng đa hướng**: kết hợp *Genetic Algorithm (GA)* để tìm tập biến tối ưu và *Correlation/F-test* để đánh giá thống kê truyền thống.

(b) **Phân nhánh huấn luyện với và không có PCA**: giúp so sánh tác động của giảm chiều dữ liệu đến hiệu quả mô hình và khả năng diễn giải.

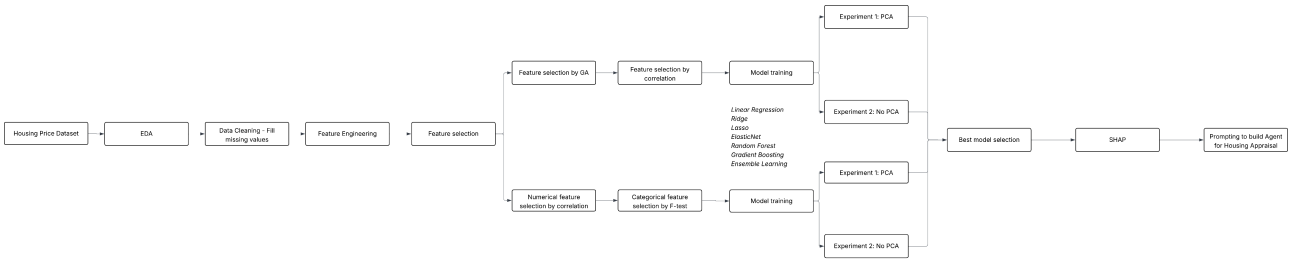
(c) **Mở rộng phạm vi mô hình**: bổ sung các thuật toán nâng cao như *ElasticNet, Random Forest, Gradient Boosting, Ensemble Learning.*

(d) **Giải thích bằng SHAP**: định lượng mức đóng góp của từng đặc trưng vào kết quả dự đoán, giúp mô hình minh bạch và đáng tin cậy.

(e) **Hướng đến ứng dụng thực tế**: tích hợp bước *Prompting to build Agent for Housing Appraisal*, tạo nền tảng cho hệ thống định giá nhà tự động.

2 Tóm tắt sự khác biệt

Thành phần	**Pipeline gốc (AIO)**	**Pipeline mới của nhóm**
Feature Selection	Thủ công	GA, Correlation, F-test
Feature Reduction	Không xét	PCA song song với dữ liệu gốc
Model Scope	Linear, Ridge, Lasso	+ ElasticNet, RF, GB, Ensemble
Interpretability	Không có	SHAP để giải thích kết quả
Application	Notebook mô phỏng	Agent định giá tự động



Hình 1: Pipeline của nhóm

Pipeline mới 4 không chỉ cải thiện độ chính xác, mà còn tăng tính minh bạch và khả năng ứng dụng trong thực tế, hướng tới mục tiêu **xây dựng mô hình định giá nhà thông minh, có thể diễn giải và mở rộng thành công cụ hỗ trợ ra quyết định**.

II. Hướng phát triển mở rộng

1 Xử lý giá trị khuyết (N/A)

Giá trị khuyết thiếu là một thách thức phổ biến trong học máy và phân tích dữ liệu. Chúng xảy ra khi một số điểm dữ liệu bị thiếu đối với các biến cụ thể trong một tập dữ liệu. Những khoảng trống thông tin này có thể ở dạng ô trống, giá trị null hoặc các ký hiệu đặc biệt như "NA", "NaN" hoặc "không xác định". Nếu không được xử lý đúng cách, giá trị bị thiếu có thể gây ảnh hưởng đến độ chính xác và độ tin cậy của mô hình. Chúng có thể làm giảm kích thước mẫu, gây ra sai lệch và gây khó khăn cho việc áp dụng một số kỹ thuật phân tích yêu cầu dữ liệu đầy đủ. Việc xử lý hiệu quả các giá trị bị thiếu là rất quan trọng để đảm bảo các mô hình học máy của chúng ta tạo ra kết quả chính xác và khách quan.

Cũng giống như mọi loại dataset khác, dataset trong dự án House Price Prediction cũng tồn tại những giá trị khuyết thiếu như vậy:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleTyp
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	2008	WI
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	5	2007	WI
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	9	2008	WI
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	2006	WI
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	12	2008	WI

5 rows × 81 columns

1.1 Lý do đằng sau các giá trị khuyết thiếu trong tập dữ liệu

Dữ liệu có thể bị thiếu trong một tập dữ liệu vì nhiều lý do, và việc hiểu rõ nguyên nhân rất quan trọng để lựa chọn cách xử lý hiệu quả nhất. Các lý do phổ biến gây ra tình trạng thiếu dữ liệu bao gồm:

- Thu thập dữ liệu không thành công hoặc có lỗi trong quá trình truyền dữ liệu.
- Những sai sót như nhập dữ liệu không chính xác hoặc sơ suất trong quá trình xử lý dữ liệu.
- Thiếu thông tin nhạy cảm hoặc thông tin cá nhân do chính sách bảo mật.
- Lỗi xảy ra trong quá trình chuẩn bị dữ liệu.

Bằng cách xác định lý do đằng sau việc dữ liệu bị thiếu, chúng ta có thể đánh giá tốt hơn tác động của nó, liệu nó có gây ra sai lệch hay ảnh hưởng đến quá trình phân tích hay không, và lựa chọn phương pháp xử lý phù hợp như quy kết hoặc loại bỏ.

1.2 Tầm quan trọng của việc xử lý các giá trị khuyết thiếu

Việc xử lý các giá trị bị thiếu rất quan trọng để đảm bảo tính chính xác và độ tin cậy của phân tích dữ liệu và các mô hình học máy. Các lý do chính bao gồm:

- **Cải thiện độ chính xác của mô hình:** Xử lý các giá trị bị thiếu giúp tránh dự đoán không chính xác và tăng hiệu suất của mô hình.
- **Tăng sức mạnh thống kê:** Việc thay thế hoặc loại bỏ dữ liệu bị thiếu cho phép sử dụng nhiều kỹ thuật phân tích hơn, đồng thời duy trì quy mô mẫu.
- **Ngăn ngừa sai lệch:** Xử lý đúng cách đảm bảo dữ liệu bị thiếu không gây ra sai lệch hệ thống, dẫn đến kết quả đáng tin cậy hơn.
- **Quyết định đáng tin cậy:** Một tập dữ liệu sạch sẽ dẫn đến những quyết định sáng suốt và đáng tin cậy hơn dựa trên những hiểu biết chính xác.

1.3 Phương pháp xác định dữ liệu khuyết thiếu

Việc phát hiện và quản lý dữ liệu bị thiếu rất quan trọng đối với việc phân tích dữ liệu. Trong phần này chỉ nêu một số hàm tiêu biểu để tham khảo khi làm việc với Pandas:

- **.isnull()** : Xác định các giá trị nào bị thiếu trong một Series hoặc DataFrame.
- **.notnull()**: ngược lại với .isnull(), nó trả về True cho các giá trị không bị thiếu và False cho các giá trị bị thiếu.
- **.info()**: display tóm tắt về kiểu dữ liệu, mức sử dụng bộ nhớ và số lượng giá trị bị thiếu trong một DataFrame.
- **.isna()** : Trả về True cho dữ liệu thiếu và False cho dữ liệu không thiếu.
- **dropna()**: Xóa các hàng hoặc cột có giá trị thiếu
- **fillna()** : Điền các giá trị còn thiếu bằng một giá trị cụ thể (mean, median,...)
- **drop_duplicates()**: Xóa các hàng trùng lặp dựa trên các cột được chỉ định.
- **unique()**: Tìm các giá trị duy nhất trong một Series hoặc DataFrame.
- **replace()**: Thay thế các giá trị được chỉ định trong DataFrame

1.4 Xử lý giá trị khuyết thiếu trong project

Trước tiên, chia dataset thành hai nhóm: categorical feature và numerical column. Sau đó xử lý cho từng loại.

Với nhóm categorical, một số đặc trưng bị thiếu, thực ra có nghĩa là "không tồn tại", chứ không phải "thiếu thông tin", ví dụ PoolQC (chất lượng hồ bơi) → nhà không có hồ bơi, ... Với nhóm này, sẽ đc thay thế bằng "None". Các cột categorical còn lại → điền giá trị phổ biến nhất.

Với nhóm numerical, những cột dạng số như diện tích tầng hầm hoặc số gara thường bị thiếu khi... nhà không có phần đó...Vì vậy, điền 0 là hợp lý nhất. LotFrontage (chiều rộng mặt tiền) thay đổi mạnh tùy khu vực, vì vậy không nên điền toàn cục. Thay vào đó, tính median riêng cho từng Neighborhood để phản ánh đặc trưng địa lý. Các numerical còn lại → điền median toàn cột

2 Kỹ thuật đặc trưng (Feature Engineering)

Để nâng cao khả năng dự đoán của mô hình hồi quy, nhóm đã tiến hành tạo ra một tập hợp các **đặc trưng dẫn xuất (derived features)** nhằm phản ánh tốt hơn các yếu tố cấu trúc, tiện ích và chất lượng của từng căn nhà. Những đặc trưng mới này được xây dựng từ các biến gốc trong dữ liệu, giúp mô hình nắm bắt các mối quan hệ phi tuyến và đa chiều giữa các yếu tố vật lý – kinh tế của bất động sản. Phần này mô tả chi tiết từng nhóm đặc trưng được xây dựng.

2.1 Tổng diện tích sử dụng (Total Square Footage)

Biến `TotalSF` biểu thị tổng diện tích sử dụng của căn nhà, được tính bằng tổng diện tích tầng hầm, tầng 1 và tầng 2:

$$\text{TotalSF} = \text{TotalBsmtSF} + \text{1stFlrSF} + \text{2ndFlrSF}$$

Đây là thước đo tổng thể về không gian sinh hoạt, thường là một trong những yếu tố chính ảnh hưởng trực tiếp đến giá trị thị trường của căn nhà.

2.2 Tổng số phòng tắm (Total Bathrooms)

Để chuẩn hóa giá trị giữa phòng tắm đầy đủ và phòng tắm bán phần, nhóm tính tổng số phòng tắm theo công thức có trọng số:

$$\text{TotalBathrooms} = \text{FullBath} + 0.5 \times \text{HalfBath} + \text{BsmtFullBath} + 0.5 \times \text{BsmtHalfBath}$$

Cách tính này giúp đảm bảo phòng tắm bán phần (Half Bath) được tính hợp lý trong tổng diện tích tiện nghi sinh hoạt.

2.3 Đặc trưng về tuổi nhà và cải tạo (Age Features)

Hai biến mới được tạo ra để thể hiện độ tuổi của căn nhà và khoảng thời gian kể từ lần cải tạo gần nhất:

$$\text{HouseAge} = \text{YrSold} - \text{YearBuilt}$$

$$\text{YearsSinceRemod} = \text{YrSold} - \text{YearRemodAdd}$$

`HouseAge` cho biết số năm kể từ khi căn nhà được xây dựng, trong khi `YearsSinceRemod` giúp mô hình hiểu rõ hơn về mức độ cập nhật và tình trạng bảo trì của tài sản.

2.4 Diện tích hiên và ban công (Porch Features)

Không gian ngoài trời là yếu tố quan trọng trong việc đánh giá giá trị cảm nhận của căn nhà. Biến `TotalPorchSF` được xây dựng bằng cách tổng hợp diện tích từ các loại hiên và ban công khác nhau:

$$\text{TotalPorchSF} = \text{OpenPorchSF} + \text{3SsnPorch} + \text{EnclosedPorch} + \text{ScreenPorch} + \text{WoodDeckSF}$$

Chỉ số này đại diện cho mức độ tiện nghi và không gian thư giãn ngoài trời của ngôi nhà.

2.5 Tương tác giữa chất lượng và tình trạng (Quality Interaction Feature)

Để thể hiện mối quan hệ giữa chất lượng xây dựng và tình trạng bảo trì, nhóm tạo biến `OverallGrade`:

$$\text{OverallGrade} = \text{OverallQual} \times \text{OverallCond}$$

Biến này giúp mô hình nhận biết rằng một căn nhà có vật liệu tốt và được bảo trì tốt thường có giá trị cao hơn nhiều so với các căn cùng hạng mục nhưng xuống cấp.

2.6 Tuổi của gara (Garage Age)

Tuổi của gara được tính bằng:

$$\text{GarageAge} = \text{YrSold} - \text{GarageYrBlt}$$

Các giá trị khuyết (do không có gara) được điền bằng 0. Biến này cho phép mô hình phân biệt giữa các căn có gara mới và các gara đã cũ, từ đó đánh giá chính xác hơn giá trị tiện ích.

2.7 Biến nhị phân về sự tồn tại (Binary Features)

Để giúp mô hình nhận biết sự có mặt của các đặc trưng quan trọng, nhóm tạo ra các biến nhị phân như sau:

$$\text{HasBasement} = \begin{cases} 1 & \text{nếu TotalBsmtSF} > 0 \\ 0 & \text{ngược lại} \end{cases} \quad \text{HasGarage} = \begin{cases} 1 & \text{nếu GarageArea} > 0 \\ 0 & \text{ngược lại} \end{cases}$$

$$\text{HasFireplace} = \begin{cases} 1 & \text{nếu Fireplaces} > 0 \\ 0 & \text{ngược lại} \end{cases} \quad \text{HasPool} = \begin{cases} 1 & \text{nếu PoolArea} > 0 \\ 0 & \text{ngược lại} \end{cases}$$

Những biến này giúp mô hình phân biệt rõ ràng giữa các căn có hoặc không có các tiện ích quan trọng như tầng hầm, gara, lò sưởi hay hồ bơi.

2.8 Tỷ lệ diện tích (Ratio Features)

Hai biến tỷ lệ được tạo ra nhằm chuẩn hóa diện tích sinh hoạt và diện tích gara theo kích thước lô đất:

$$\text{LivLotRatio} = \frac{\text{GrLivArea}}{\text{LotArea}}, \qquad \text{GarageLotRatio} = \frac{\text{GarageArea}}{\text{LotArea}}$$

Các chỉ số này giúp mô hình đánh giá mức độ sử dụng đất hiệu quả của từng căn nhà, đồng thời giảm thiểu sự sai lệch do quy mô lô đất khác nhau.

2.9 Tổng kết

Quá trình xây dựng đặc trưng này đã giúp cải thiện đáng kể khả năng học và giải thích của mô hình hồi quy. Những đặc trưng mới không chỉ phản ánh không gian và tiện ích, mà còn kết hợp cả yếu tố thời gian, chất lượng, và tỷ lệ sử dụng đất — tạo ra nền tảng mạnh mẽ cho việc huấn luyện và suy luận giá nhà chính xác hơn.

3 Lựa chọn đặc trưng (Feature Selection)

3.1 Chọn đặc trưng bằng hệ số tương quan và F-statistics

Sau khi tiền xử lý, tập dữ liệu còn lại chứa nhiều đặc trưng mô tả hình thái, vật liệu và điều kiện bán của ngôi nhà. Để tránh đa cộng tuyến và giảm chiều dữ liệu, nhóm thực hiện bước lựa chọn đặc trưng theo hai hướng song song: (i) lựa chọn biến số dựa trên hệ số tương quan Pearson, và (ii) lựa chọn biến phân loại dựa trên kiểm định F-statistics (ANOVA F-test).

(1) Lựa chọn đặc trưng số. Với mỗi biến số x_j, tính hệ số tương quan Pearson với biến mục tiêu y:

$$r_j = \frac{\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Giữ lại các biến có $|r_j| \geq 0.3$, biểu thị mối liên hệ mạnh với giá nhà. Các biến này thường gồm `OverallQual`, `GrLivArea`, `GarageCars`, `TotalBsmtSF`, v.v. Bước này loại bỏ các đặc trưng yếu và giảm đa cộng tuyến giữa các biến.

```python
# Compute the absolute correlation between each numerical feature and the target variable
correlation = numeric_df.corr()["SalePrice"].abs().sort_values(ascending=False)
print(correlation)

# Select features with correlation above the threshold (e.g., 0.1)
selected_numeric_stats = correlation[correlation >= 0.1].index.tolist()

# Remove the target variable itself from the selected feature list
selected_numeric_stats.remove("SalePrice")
print(selected_numeric_stats)
```

Code Listing 1: Feature selection for numerical variables using correlation analysis

(2) Lựa chọn đặc trưng phân loại. Với các biến phân loại, áp dụng mã hoá One-Hot và sử dụng kiểm định ANOVA F-test cho hồi quy:

$$F_j = \frac{\text{Var}_{\text{between}}(x_j)}{\text{Var}_{\text{within}}(x_j)}$$

Giữ lại các đặc trưng có $p < 0.05$ để đảm bảo sự khác biệt có ý nghĩa thống kê giữa các nhóm. Các biến như `Neighborhood`, `HouseStyle`, `SaleCondition` thường được chọn nhờ giá trị F cao.

```python
from sklearn.feature_selection import SelectKBest, f_regression

# Select top k categorical features using ANOVA F-test
selector = SelectKBest(score_func=f_regression, k=min(15, len(categorical_cols)))
X_cat_selected = selector.fit_transform(X_cat, y)

# Get mask of selected features
selected_cat_mask = selector.get_support()

# Extract feature names that passed the selection
selected_categorical_stats = [
    categorical_cols[i]
    for i in range(len(categorical_cols))
    if selected_cat_mask[i]
]

# Retrieve and sort F-statistic scores for the selected features
```

```python
feature_scores = selector.scores_
cat_feature_scores = [
    (categorical_cols[i], feature_scores[i])
    for i in range(len(categorical_cols))
    if selected_cat_mask[i]
]
cat_feature_scores.sort(key=lambda x: x[1], reverse=True)

# Extend the final feature list with the selected categorical features
selected_features_stats.extend(selected_categorical_stats)
```

Code Listing 2: Feature selection for categorical variables using F-statistics

(3) Tập đặc trưng cuối cùng. Hợp nhất hai tập con:

$$\mathcal{F}_{\text{final}} = \mathcal{F}_{\text{corr}} \cup \mathcal{F}_{\text{F-test}}$$

Tập $\mathcal{F}_{\text{final}}$ là đầu vào cho các bước PCA, GA, hoặc mô hình Ridge/Lasso sau này, đảm bảo mô hình vừa gọn nhẹ vừa dễ diễn giải.

3.2 Chọn đặc trưng bằng thuật toán di truyền (Genetic Algorithm)

Feature Selection hoạt động bằng cách loại bỏ các đặc trưng dư thừa và chỉ giữ lại những đặc trưng được coi là quan trọng liên quan đến biến đầu ra cần dự đoán. Bằng cách này, độ phức tạp của mô hình được giảm thiểu.

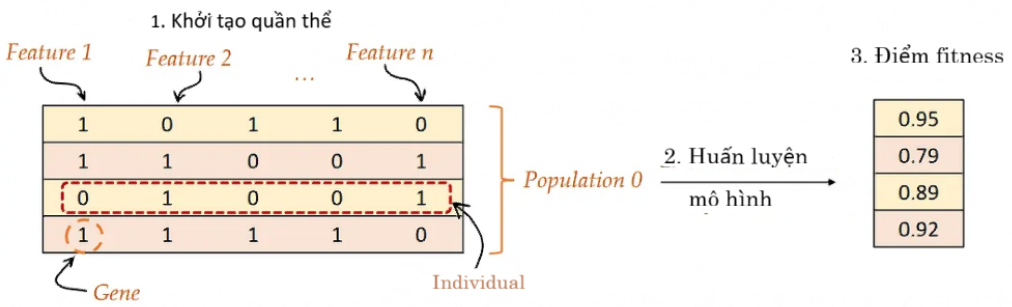
Trong phần này, nhóm mình áp dụng **Thuật toán di truyền (GA)** để lựa chọn đặc trưng. Cách này thuộc về loại lựa chọn đặc trưng bằng phương pháp Embedded methods. Phương pháp này chỉ việc thực hiện lựa chọn đặc trưng trong quá trình huấn luyện mô hình, tạo ra mô hình được huấn luyện cuối cùng làm nổi bật các đặc trưng hữu ích nhất.

Ở đây, **thuật toán di truyền** sẽ được nhóm sử dụng để **tìm kiếm tổ hợp đặc trưng tối ưu** bằng cách kết hợp các cha mẹ dựa trên **điểm số fitness** của chúng, Điểm fitness được xác định dựa trên **hiệu suất của mô hình hồi quy tuyến tính (Linear Regression)**, được đo bằng **hệ số xác định** R^2 khi mô hình được huấn luyện trên tập đặc trưng tương ứng.

3.2.1 Sơ lược về thuật toán di truyên(genetic algorithm)

Thuật toán di truyền gồm các bước cơ bản sau:

1. Tạo quần thể ban đầu:
2. Đào tạo một mô hình trên mọi cá thể từ quần thể và xác định hàm fitness.
3. Tính độ điểm số fitness của mỗi cá thể từ quần thể.



4. Chọn cha mẹ: Sử dụng hàm fitness để chọn cha mẹ tiềm năng từ quần thể để sinh con.
5. Sinh con: Tạo ra các cá thể mới bằng cách kết hợp hai cá thể bố mẹ được chọn thông qua phép lai ghép. Các cá thể con mới tạo thành một quần thể mới.
6. Đột biến: Sử dụng xác suất được chỉ định (5%), 5% gen của thế hệ con sẽ bị đột biến. Điều này được thực hiện để có khả năng đưa gen mới lạ vào quần thể.
7. Đánh giá cá thể con: Huấn luyện mô hình trên mọi cá thể trong quần thể và đánh giá mức độ phù hợp của chúng bằng cách sử dụng cùng hàm fitness như trên.
8. Chọn thế hệ tiếp theo: Chọn những cá thể có điểm số tốt nhất và thêm chúng vào quần thể thế hệ tiếp theo.
9. Điều kiện dừng: Xác định thời điểm dừng thuật toán dựa trên số thế hệ đã đặt hoặc khi đạt đến mức chất lượng giải pháp mong muốn.
10. Tiếp tục đánh giá, chọn lựa, lai ghép và đột biến đến khi thỏa mãn điều kiện dừng

3.2.2 Áp dụng thuật toán di truyền vào tập dữ liệu Sales Predict

Bộ dữ liệu dự đoán giá nhà sau khi nhóm thực hiện các bước xử lý như làm sạch, xử lý NaN, feature engineer chứa 92 đặc trưng và 1 biến dự đoán. Mục tiêu là đào tạo một mô hình có thể xác định các đặc điểm quan trọng nhất từ bộ dữ liệu.

```
for col in cat_cols:
    X[col] = le.fit_transform(X[col].astype(str))

print(f"Số lượng features ban đầu: {X.shape[1]}")
print(f"Shape của X: {X.shape}")
print(f"Shape của y: {y.shape}")

Số lượng features ban đầu: 92
Shape của X: (1460, 92)
Shape của y: (1460,)
```

Tiếp theo, nhóm định nghĩa các hàm cho từng phần của thuật toán di truyền như sau. Trong bước này, chúng tôi chọn N_POPULATION = 100, N_GENERATION = 30, crossover=0.5, xác suất cá thể đột biến = 0.05...

Thông qua quá trình huấn luyện, kết quả nhóm chọn lọc được 48/92 feature để thực hiện các bước training model phía sau.

Danh sách 48 features được chọn:

MSSubClass , LotArea , Street , LotShape , Neighborhood , HouseStyle , OverallQual , OverallCond , YearBuilt , RoofStyle , RoofMatl , Exterior1st , MasVnrType , MasVnrArea , ExterQual , Foundation , BsmtQual , BsmtCond ,BsmtExposure ,BsmtFinType1 ,BsmtFinSF2 ,HeatingQC ,CentralAir ,Electrical ,GrLivArea ,BsmtFullBath ,HalfBath ,BedroomAbvGr ,KitchenAbvGr ,KitchenQual ,Functional ,Fireplaces ,FireplaceQu ,GarageCars ,GarageCond ,PavedDrive ,WoodDeckSF ,EnclosedPorch ,ScreenPorch ,PoolArea ,SaleCondition ,TotalBathrooms ,TotalPorchSF ,GarageAge ,HasGarage ,HasFireplace ,HasPool ,GarageLotRatio.

4 Huấn luyện và so sánh mô hình

4.1 Giảm số chiều dữ liệu bằng phương pháp PCA

Trong thực tế, các biến ngẫu nhiên thường có số chiều và kích thước rất lớn, có thể lên tới hàng nghìn. Thêm vào đó, số lượng các điểm dữ liệu lớn gây khó khăn cho việc lưu trữ và tốc độ tính toán.

Giả sử cho p biến ngẫu nhiên. Liệu có cách nào để xây dựng được p biến mới không tương quan với nhau và được biểu diễn tuyến tính thông qua các biến cũ, đồng thời không làm mất mát dữ liệu ban đầu hoặc làm mất mát ít nhất?

Câu trả lời chính là **Thuật toán phân tích thành phần chính (Principal Component Analysis - PCA)**.

Động cơ (vấn đề đặt ra trong dự đoán giá nhà).

Sau bước mã hoá/khai phá đặc trưng (ví dụ One-Hot Encoding cho khu phố, chất lượng; tổng hợp diện tích), ta thường có hàng chục biến, trong đó nhiều biến *tương quan mạnh*. Điều này gây đa cộng tuyến cho mô hình tuyến tính, làm *tăng phương sai ước lượng*, dễ overfit và chậm huấn luyện. PCA được dùng như một phép *nén tuyến tính* để thay thế cụm biến tương quan bằng vài trục trực giao (PCs) giữ phần lớn phương sai.

Mục tiêu của PCA

Phân tích thành phần chính (PCA) là một kỹ thuật phân tích cấu trúc ma trận hiệp phương sai $\mathbf{\Sigma}$ của một tập hợp các biến X thông qua các tổ hợp tuyến tính của các biến đó. Mục tiêu chính của PCA bao gồm:

- Giảm số chiều của dữ liệu.
- Thay vì giữ lại các trục tọa độ của không gian cũ, PCA xây dựng một không gian mới ít chiều hơn, nhưng lại có khả năng biểu diễn dữ liệu tốt tương đương không gian cũ, tức là đảm bảo độ biến thiên của dữ liệu trên mỗi chiều mới.
- Các trục tọa độ trong không gian mới là tổ hợp tuyến tính của không gian cũ. Về mặt ngữ nghĩa, PCA xây dựng các biến mới dựa trên các biến đã quan sát được, đồng thời vẫn biểu diễn tốt dữ liệu ban đầu.
- Trong không gian mới, các liên kết tiềm ẩn của dữ liệu có thể được khám phá. Trong không gian cũ, các liên kết này có thể khó phát hiện hơn hoặc không rõ ràng.

Vậy ta muốn PCA làm điều gì?

PCA phân tích cấu trúc ma trận hiệp phương sai $\mathbf{\Sigma}$ của dữ liệu và tìm hệ trục trực giao $\{e_1, \ldots, e_p\}$ sao cho:

1. PC_1 giữ *phương sai* lớn nhất; PC_2 giữ phần còn lại lớn thứ hai và *trực giao* PC_1; tiếp tục như vậy;
2. Khi chỉ giữ k PC đầu ($k \ll p$), ta *giảm chiều* nhưng vẫn *giữ phần lớn biến thiên* của dữ liệu;
3. Các trục mới (PCs) là *tổ hợp tuyến tính* của các biến gốc và *không tương quan* với nhau.

Cơ sở toán học

Thiết lập và bài toán cực trị:
Gọi $X \in \mathbb{R}^{n\times p}$ là ma trận dữ liệu đã *quy tâm theo cột* ($\sum_{i=1}^{n} X_{i\cdot} = 0$). Ma trận hiệp phương sai

$$\Sigma = \frac{1}{n-1} X^\top X \quad (\Sigma \succeq 0,\ \text{đối xứng}).$$

PCA tìm hướng w (độ dài 1) tối đa hoá *phương sai chiếu*:

$$\max_{\|w\|=1} \operatorname{Var}(Xw) = \max_{\|w\|=1} w^\top \Sigma\, w.$$

Dùng nhân tử Lagrange cho ràng buộc $w^\top w = 1$:

$$\mathcal{L}(w,\lambda) = w^\top \Sigma w - \lambda(w^\top w - 1) \quad \Rightarrow \quad \Sigma w = \lambda w.$$

Do đó, nghiệm w là **véc-tơ riêng** và λ là **trị riêng** của Σ. Sắp $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$, ta nhận được $\mathrm{PC}_i = e_i$ (véc-tơ riêng thứ i), với phương sai PC_i giữ lại đúng bằng λ_i. Tỷ lệ phương sai tích luỹ khi giữ k PC:

$$\text{ExplainedVariance}(k) = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{p} \lambda_i}.$$

Góc nhìn SVD và ý nghĩa tối ưu của PCA.

PCA còn có thể được hiểu thông qua phân rã giá trị kỳ dị (SVD — Singular Value Decomposition). Giả sử dữ liệu đã được quy tâm có dạng:

$$X = U\, S\, V^\top,$$

trong đó:

- U và V là các ma trận trực giao (giống như ma trận "quay" dữ liệu),
- S là ma trận chéo chứa các **giá trị kỳ dị** $s_1 \geq s_2 \geq \cdots \geq s_p$ (liên hệ chặt với lượng thông tin hay "độ lan" của dữ liệu trên từng hướng).

Các trục chính của PCA **chính là các cột của** V, và phương sai mà mỗi trục giữ lại được:

$$\lambda_i = \frac{s_i^2}{n-1}.$$

Điều này có nghĩa là: thay vì phải tính trị riêng từ ma trận hiệp phương sai $\Sigma = \frac{1}{n-1} X^\top X$, ta có thể dùng SVD để tìm trực tiếp các hướng lan rộng nhất (ổn định hơn khi p lớn).

Hơn nữa, PCA không chỉ "quay" trục cho đẹp — nó còn *tối ưu theo nghĩa nén thông tin*. Nếu ta chỉ giữ k giá trị kỳ dị lớn nhất $(s_1, \ldots, s_k)$ và tạo lại dữ liệu xấp xỉ:

$$X_k = U_k S_k V_k^\top,$$

thì X_k là **bản nén hạng-k tốt nhất** của X theo chuẩn lỗi bình phương (Frobenius). Phần lỗi còn lại khi bỏ các thành phần nhỏ hơn chính là tổng lượng phương sai bị mất:

$$\|X - X_k\|_F^2 = (n-1) \sum_{i=k+1}^{p} \lambda_i.$$

Nói cách khác, *các trị riêng bị bỏ đi biểu diễn phần thông tin mà ta hy sinh khi giảm chiều.*

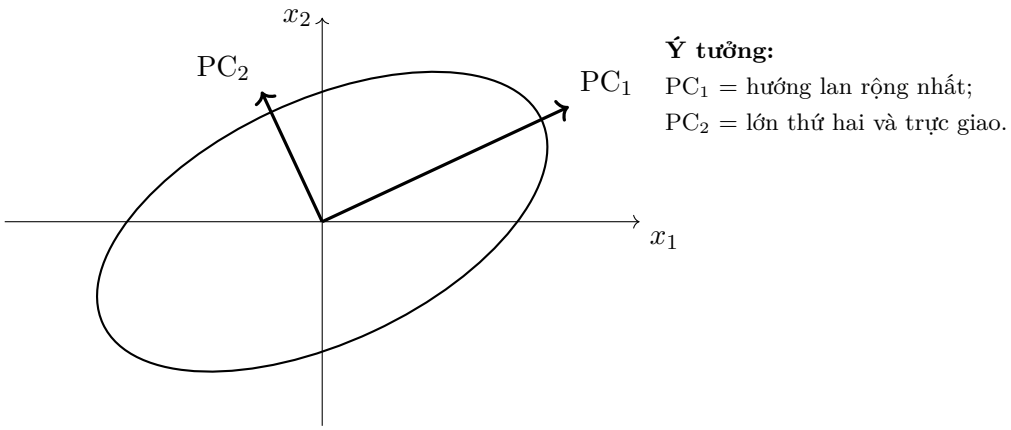
PCA giống như việc nén hình ảnh — giữ lại vài "kênh" mang nhiều thông tin nhất. Nếu ta giữ càng nhiều PC, hình (dữ liệu) càng rõ; giữ ít PC thì mờ hơn nhưng nhẹ và nhanh hơn.

Chuẩn hoá khi nào?

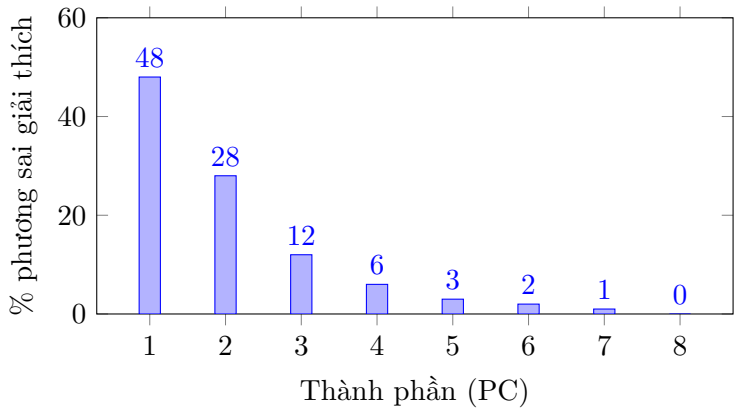
Nếu các đặc trưng khác thang đo/đơn vị (m^2, số phòng, tuổi nhà, ...), **nên chuẩn hoá** (z-score) trước PCA. Khi đó, PCA tương đương làm trên *ma trận tương quan*, tránh việc biến có phương sai tự nhiên lớn "áp đảo" PCs.

Minh hoạ

Dữ liệu là "đám mây điểm". PCA dịch gốc tới trọng tâm và *quay* hệ trục tới hướng lan rộng nhất (PC_1), rồi hướng lớn thứ hai trực giao (PC_2), v.v. (Hình 2). Đồ thị scree (Hình 3) giúp chọn k tại "điểm gãy".



Hình 2: Trực quan hình học: quay trục tới các hướng tối đa phương sai (PC_1, PC_2).



Hình 3: Đồ thị scree minh hoạ: điểm gãy sau PC_2–PC_3. Chọn $k = 2$ hoặc 3 tuỳ ngưỡng tích luỹ (vd. $\geq 95\%$).

PCA giảm chiều hiệu quả, hạ đa cộng tuyến, tăng ổn định và tốc độ huấn luyện cho bài toán giá nhà. Ta dùng PCA như *bước trung gian*—không phải mục đích cuối—và chọn k bằng tiêu chí phương sai tích luỹ/scree.

Ví dụ về PCA Giả sử chúng ta có dữ liệu về một số đặc trưng của các ngôi nhà, bao gồm:

1. Diện tích (S).

2. Số phòng ngủ (R).
3. Số phòng tắm (B).
4. Diện tích sân vườn (G).
5. Khoảng cách đến trung tâm thành phố (D).
6. Tuổi của ngôi nhà (A).

Dữ liệu này có thể được biểu diễn bằng một ma trận dữ liệu với m ngôi nhà và $n = 6$ đặc trưng:

$$\begin{bmatrix} S_1 & R_1 & B_1 & G_1 & D_1 & A_1 \\ S_2 & R_2 & B_2 & G_2 & D_2 & A_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ S_m & R_m & B_m & G_m & D_m & A_m \end{bmatrix}$$

Mục tiêu:

- Chúng ta muốn dự đoán giá trị của một ngôi nhà dựa trên các đặc trưng này.
- Tuy nhiên, có những đặc trưng tương quan chặt chẽ với nhau (như diện tích ngôi nhà, số phòng ngủ, số phòng tắm), và một số đặc trưng có thể ít quan trọng hơn (như diện tích sân vườn hoặc khoảng cách đến trung tâm thành phố).
- Vì vậy, PCA có thể giúp chúng ta giảm số chiều dữ liệu từ 6 xuống một số chiều nhỏ hơn mà vẫn giữ được phần lớn thông tin quan trọng.

Chọn k thành phần chính

- Giả sử $k = 2$, chúng ta chọn hai vector riêng tương ứng với hai giá trị riêng lớn nhất, tạo ra một không gian mới với 2 chiều. Các vector riêng này có thể biểu diễn sự kết hợp của các đặc trưng ban đầu.
- **Ví dụ**:

  - Vector riêng thứ nhất có thể đại diện cho "tổng kích thước" của ngôi nhà, là sự kết hợp của diện tích, số phòng ngủ và số phòng tắm.
  - Vector riêng thứ hai có thể đại diện cho "vị trí địa lý" của ngôi nhà, là sự kết hợp của khoảng cách đến trung tâm thành phố và diện tích sân vườn.

Dữ liệu sau khi giảm chiều có thể được biểu diễn bởi ma trận Z:

- Giả sử sau khi thực hiện PCA, ta nhận thấy rằng các đặc trưng "diện tích", "số phòng ngủ" và "số phòng tắm" tương quan chặt chẽ và có thể được thay thế bằng một thành phần chính duy nhất.
- Tương tự, "diện tích sân vườn" và "khoảng cách đến trung tâm thành phố" có thể được gộp lại thành một thành phần chính thứ hai.
- Bây giờ, thay vì sử dụng 6 đặc trưng ban đầu, chúng ta chỉ cần sử dụng 2 đặc trưng:

$$Z = \begin{bmatrix} z_1^{(1)} & z_2^{(1)} \\ z_1^{(2)} & z_2^{(2)} \\ \vdots & \vdots \\ z_1^{(m)} & z_2^{(m)} \end{bmatrix}$$

Trong đó z_1 và z_2 đại diện cho hai thành phần chính (principal components) quan trọng nhất.

Các thành phần chính

- Các thành phần chính p được dùng để tạo tổng độ biến thiên của hệ thống, nhưng thường thì phần lớn độ biến thiên này có thể được tính toán bởi một phần nhỏ số k các thành phần chính, mà vẫn giữ gần như toàn bộ thông tin trong p thành phần ban đầu.
- Do đó, k thành phần chính có thể được sử dụng thay thế cho p thành phần ban đầu, giúp giảm độ phức tạp mà vẫn đảm bảo tính chính xác.

Khái quát: Các số liệu được xem như tạo thành một "đám mây". PCA giúp phản ánh tốt nhất hình ảnh của "đám mây", tức là "bóng" của nó chiếu lên các trục tọa độ. Nói cách khác, PCA dịch chuyển gốc tọa độ đến trọng tâm của "đám mây"và xác định các trục tọa độ mới chính là các thành phần chính.

Ý tưởng và cách thức triển khai thuật toán

1. **Chuẩn hóa dữ liệu**: Trước khi áp dụng PCA, dữ liệu cần được chuẩn hóa để mỗi đặc trưng có cùng độ lớn. Điều này rất quan trọng vì các đặc trưng khác nhau có thể có **đơn vị đo khác nhau** (ví dụ: chiều dài và trọng lượng), và nếu không chuẩn hóa, các đặc trưng có giá trị lớn hơn sẽ chiếm ưu thế trong kết quả PCA.
2. **Tính ma trận hiệp phương sai**: để hiểu rõ mối quan hệ giữa các đặc trưng. Ma trận hiệp phương sai cho biết mức độ mà hai đặc trưng thay đổi cùng nhau. Nếu hai đặc trưng có tương quan cao, điều này ngụ ý rằng có thể loại bỏ một trong số chúng mà không mất nhiều thông tin.
3. **Tính giá trị riêng và vectơ riêng của ma trận hiệp phương sai**: Từ ma trận hiệp phương sai, chúng ta tiến hành tính giá trị riêng (eigenvalues) và vectơ riêng (eigenvectors). Vectơ riêng xác định hướng của các thành phần chính, còn giá trị riêng cho biết mức độ biến thiên mà mỗi thành phần chính nắm giữ. Các thành phần chính này là các tổ hợp tuyến tính của các đặc trưng ban đầu, được sắp xếp theo thứ tự từ quan trọng nhất đến ít quan trọng nhất
4. **Chọn số lượng thành phần chính**: Giữ lại những thành phần chính có tổng giá trị riêng chiếm 90-95% tổng biến thiên.
5. **Chiếu dữ liệu lên không gian mới**: Dữ liệu được chiếu lên không gian mới với số chiều giảm nhưng vẫn giữ lại phần lớn thông tin.

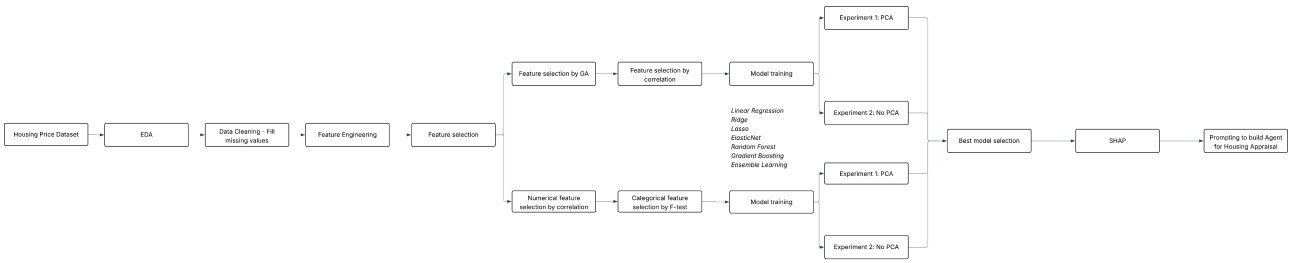
4.2 Huấn luyện mô hình và thực nghiệm PCA

4.2.1 Cấu trúc huấn luyện theo pipeline

Dựa trên quy trình trong Hình 4, nhóm thiết kế bốn hướng pipeline độc lập nhằm đánh giá hiệu quả của từng phương pháp chọn đặc trưng và kỹ thuật giảm chiều:

1. **GA_base_scaled**: đặc trưng được chọn bằng Genetic Algorithm (GA) trên dữ liệu đã chuẩn hoá (Min–Max Scaler).
2. **GA_base_raw**: đặc trưng được chọn bằng GA nhưng huấn luyện trên dữ liệu gốc chưa chuẩn hoá.
3. **Stats_base_scaled**: đặc trưng được chọn bằng thống kê (tương quan + F-test) và chuẩn hoá.
4. **Stats_base_raw**: đặc trưng được chọn bằng thống kê nhưng giữ nguyên phân phối gốc.

Mỗi pipeline được huấn luyện trên nhiều mô hình hồi quy khác nhau: *Linear Regression, Ridge, Lasso, ElasticNet, Random Forest, Gradient Boosting, Ensemble Learning*. Hai nhánh thử nghiệm song song được thực hiện cho mỗi pipeline: **Experiment 1: PCA** và **Experiment 2: No PCA**. Sau khi đánh giá, mô hình có hiệu suất cao nhất được chọn để giải thích bằng SHAP và xây dựng *Housing Appraisal Agent*.



Hình 4: Pipeline tổng quát từ lựa chọn đặc trưng đến xây dựng Agent định giá nhà

4.2.2 Quy trình huấn luyện

Mỗi mô hình trong từng pipeline đều tuân theo quy trình huấn luyện thống nhất:

1. **Chia dữ liệu** thành tập huấn luyện và kiểm thử với tỷ lệ 75%–25%.
2. **Huấn luyện mô hình** trên tập train.
3. **Dự đoán** giá nhà trên cả hai tập (train/test).
4. **Đánh giá** bằng hai chỉ số chính:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}, \qquad R^2 = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

5. **Lưu kết quả** và sắp xếp theo giá trị R^2 giảm dần để xác định mô hình tốt nhất.

```python
for name, model in models.items():
    regressor = model.fit(X_train, y_train)
    y_train_pred = regressor.predict(X_train)
    y_test_pred = regressor.predict(X_test)

    rmse = np.sqrt(mean_squared_error(y_test, y_test_pred))
    r2 = r2_score(y_test, y_test_pred)

    results.append({
        "Pipeline": pipeline_name,
        "Model": name,
        "RMSE": rmse,
        "R2": r2
    })
```

Code Listing 3: Unified model training and evaluation process

4.2.3 Kết quả và so sánh các pipeline

Bảng 1 trình bày kết quả huấn luyện của tất cả các pipeline và mô hình, được sắp xếp theo R^2 giảm dần. Các mô hình **Stats_base_raw** đạt hiệu suất vượt trội hơn hẳn so với nhóm GA, đặc biệt với mô hình *Ensemble Learning*.

Bảng 1: Kết quả so sánh giữa các pipeline và mô hình hồi quy

Pipeline	**Model**	**RMSE**	R^2
Stats_base_raw	Ensemble	0.1353	0.9018
Stats_base_raw	Gradient Boosting	0.1360	0.9008
Stats_base_scaled	Linear Regression	0.1392	0.8962
Stats_base_scaled	Ridge	0.1444	0.8883
Stats_base_raw	Random Forest	0.1447	0.8879
Stats_base_scaled	ElasticNet	0.1448	0.8877
GA_base_raw	Ensemble	0.1451	0.8872
Stats_base_scaled	Lasso	0.1455	0.8866
GA_base_raw	Gradient Boosting	0.1481	0.8825
GA_base_scaled	Ridge	0.1483	0.8822
GA_base_scaled	ElasticNet	0.1499	0.8796
GA_base_scaled	Lasso	0.1530	0.8746
GA_base_raw	Random Forest	0.1563	0.8691
GA_base_scaled	Linear Regression	0.1888	0.8089
GA_pca	Gradient Boosting	0.2483	0.6695
Stats_pca	Gradient Boosting	0.2503	0.6642
Stats_pca	Random Forest	0.2590	0.6405
GA_pca	Ridge	0.4130	0.0859
Stats_pca	Lasso	0.4300	0.0094

4.2.4 Phân tích kết quả

1. Hiệu quả của pipeline `Stats_base_raw`. Pipeline này cho kết quả cao nhất với mô hình **Ensemble**, đạt $R^2 = 0.9018$ và $RMSE = 0.1353$. Việc giữ nguyên thang đo gốc (raw) cho phép mô hình ensemble (kết hợp Gradient Boosting và Random Forest) tận dụng được mối quan hệ phi tuyến giữa các đặc trưng mà không bị bóp méo do chuẩn hoá. Điều này chứng tỏ rằng các đặc trưng được chọn bằng tương quan và F-test đã đủ tốt, không cần đến GA hoặc PCA.

2. So sánh giữa dữ liệu chuẩn hoá (scaled) và dữ liệu gốc (raw). Đối với các mô hình tuyến tính (Linear, Ridge, Lasso), dữ liệu *scaled* cho kết quả ổn định hơn (ví dụ `Stats_base_scaled` đạt $R^2 = 0.896$), do thang đo đồng nhất giúp thuật toán hội tụ nhanh. Ngược lại, các mô hình phi tuyến (Ensemble, Random Forest, Gradient Boosting) lại hoạt động tốt hơn trên dữ liệu gốc, vì chúng không phụ thuộc vào khoảng giá trị tuyệt đối mà chỉ dựa vào các phân chia nút cây.

3. So sánh giữa GA và Statistical Selection. Các pipeline sử dụng **Genetic Algorithm** (GA) có R^2 dao động từ 0.808 đến 0.887, thấp hơn nhóm **Statistical**. Điều này có thể do GA bị giới hạn bởi kích thước quần thể hoặc tiêu chí chọn lọc chưa tối ưu, trong khi phương pháp thống kê (correlation + F-test) đã loại bỏ hiệu quả các biến nhiễu.

4. Ảnh hưởng của PCA. Cả hai nhóm (`GA_pca`, `Stats_pca`) đều cho kết quả $R^2 < 0.67$, thấp hơn đáng kể so với pipeline không PCA. Nguyên nhân là PCA chỉ giữ thông tin về phương sai tổng thể mà làm mất mối quan hệ trực tiếp giữa đặc trưng và giá nhà. Do vậy, PCA không phù hợp trong bài toán này vì interpretability (giải thích đặc trưng) là mục tiêu quan trọng.

5. Tổng kết.

- **Best Model:** Stats_base_raw – Ensemble Learning.
- **Best Performance:** $R^2 = 0.9018$, $RMSE = 0.1353$.
- **Key Insight:** Statistical feature selection (Correlation + F-test) trên dữ liệu gốc cho kết quả tốt nhất, vừa đảm bảo độ chính xác, vừa duy trì khả năng giải thích mô hình.

6. Hướng phát triển. Mô hình tốt nhất được sử dụng trong bước tiếp theo của pipeline để tính toán giá trị SHAP, giúp giải thích ảnh hưởng của từng đặc trưng đến giá dự đoán. Các kết quả này sẽ được dùng để xây dựng **Agent định giá nhà (Housing Appraisal Agent)** — một hệ thống có thể trả lời truy vấn của người dùng về ước lượng giá trị bất động sản một cách minh bạch và có cơ sở dữ liệu.

5 Giải thích mô hình bằng SHAP (XAI)

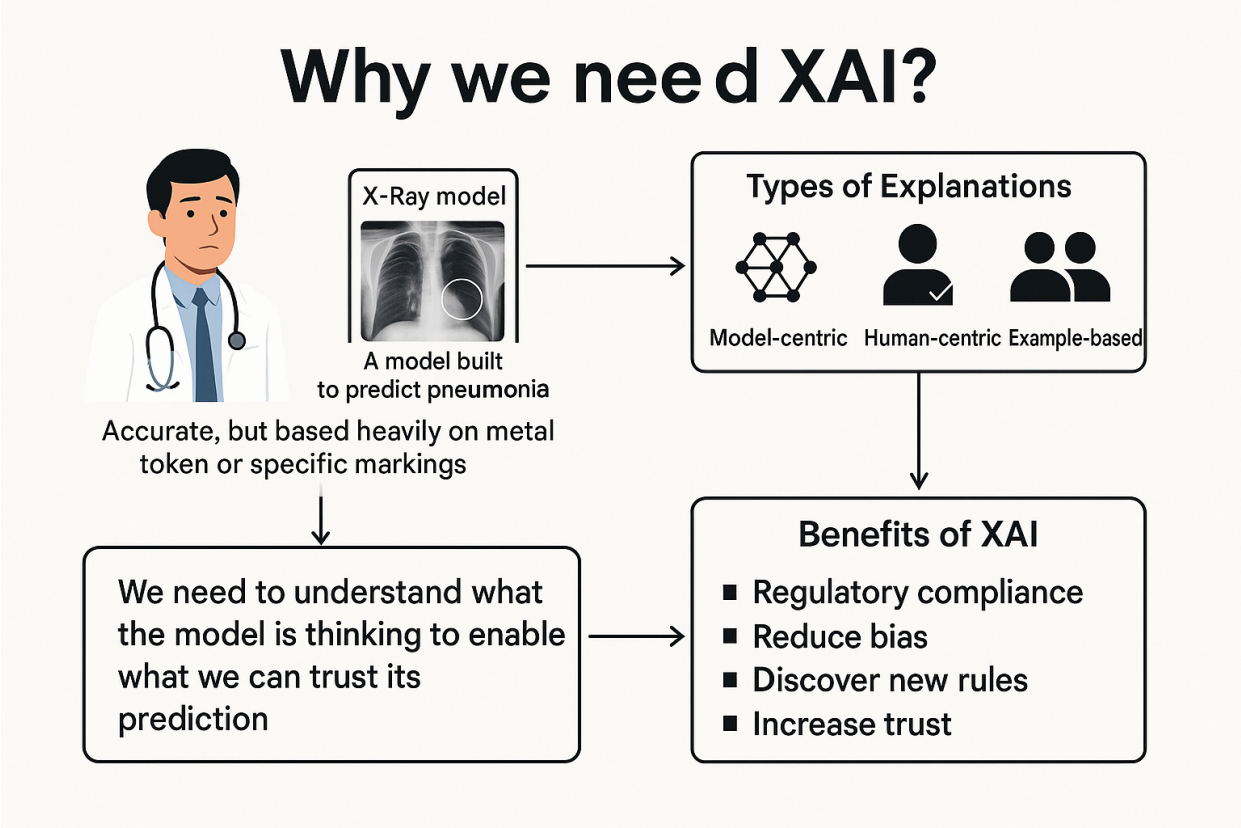
Trong học máy (Machine learning), ta thường quan tâm đến các chỉ số như **accuracy, F1-score, RMSE, AUC**,... và xem mô hình nào đạt điểm cao nhất thì "tốt nhất".

Nhưng nếu chỉ dừng ở đó, ta mới chỉ thấy đầu ra, chứ chưa hiểu cách suy nghĩ bên trong của mô hình.

Ví dụ thực tế: Mô hình chẩn đoán viêm phổi

Một mô hình Deep Learning được huấn luyện để phát hiện viêm phổi từ ảnh X-quang ngực đạt tới 95% accuracy. Tuy nhiên, khi dùng công cụ XAI để trực quan hóa vùng mà mô hình "chú ý", người ta phát hiện:

- Mô hình không hề nhìn vào vùng phổi.
- Nó dựa vào mảnh kim loại nhỏ hoặc ký hiệu của máy X-ray di động. Nguyên nhân: trong tập dữ liệu huấn luyện, phần lớn bệnh nhân nặng được chụp bằng máy X-ray di động có gắn token kim loại đó. Do đó mô hình học sai quy luật: "Nếu có token kim loại $\rightarrow$ bệnh nặng".



Hình 5: Vì sao ta cần mô hình giải thích trong mô hình chẩn đoán bệnh sử dụng ảnh X-ray

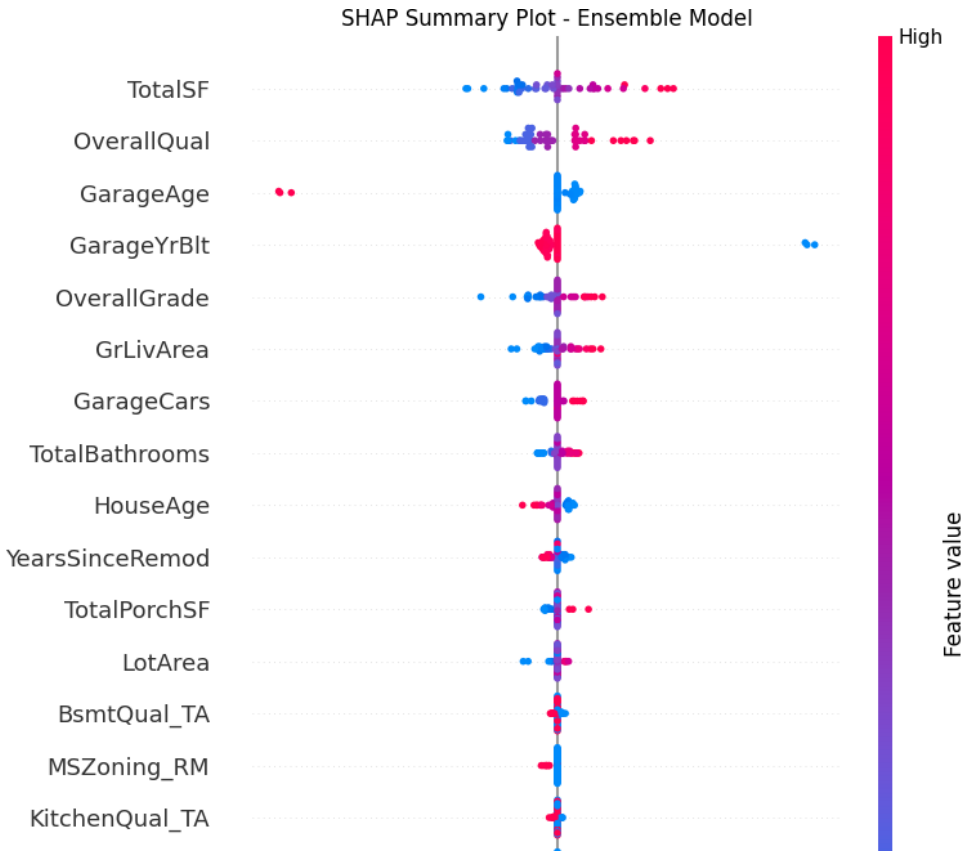
Kết luận: Mô hình đúng về mặt thống kê, nhưng sai về mặt y học. Đây là lý do ta cần **XAI (Explainable AI)** để nhìn vào bên trong "hộp đen" và hiểu vì sao mô hình đưa ra quyết định như vậy.

Trong bài toán dự đoán giá nhà, việc tìm hiểu biến nào đóng góp vào giá trị dự đoán là rất quan trọng bởi điều này giúp mô hình trở nên *minh bạch hơn*, đồng thời cung cấp góc nhìn chuyên sâu về yếu tố ảnh hưởng đến giá trị bất động sản. Đặc biệt đối với bộ dữ liệu mà nhóm đã áp dụng nhiều kỹ thuật **feature engineering**, việc kiểm tra mức độ đóng góp của các biến sẽ cho thấy liệu các đặc trưng mới có thực sự mang lại giá trị cho mô hình dự đoán hay không.

Chính vì vậy, nhóm đã thực hiện phân tích SHAP (SHapley Additive exPlanations) trên một tập mẫu đại diện, nhằm cung cấp **giải thích ở cả cấp độ toàn cục (global)** và **cục bộ (local)** cho mô hình dự đoán giá nhà. Nhóm lựa chọn 4 loại biểu đồ SHAP để minh hoạ cho các góc nhìn khác nhau. Dưới đây là mô tả và kết quả biểu đồ đầu tiên:

- **1. Summary Plot:**
  - Dùng để trình bày **mức độ quan trọng tổng thể của các biến** trong mô hình, đồng thời thể hiện **ảnh hưởng của giá trị cao/thấp của từng biến đến dự đoán**.
  - Mỗi điểm đại diện cho một mẫu dữ liệu. Màu biểu thị giá trị đặc trưng (cao → thường làm tăng giá dự đoán, thấp → có thể làm giảm hoặc ít ảnh hưởng).

    Kết quả thu được:



Hình 6: Biểu đồ SHAP summary cho 50 mẫu (chỉ in ra 15 mẫu để quan sát)

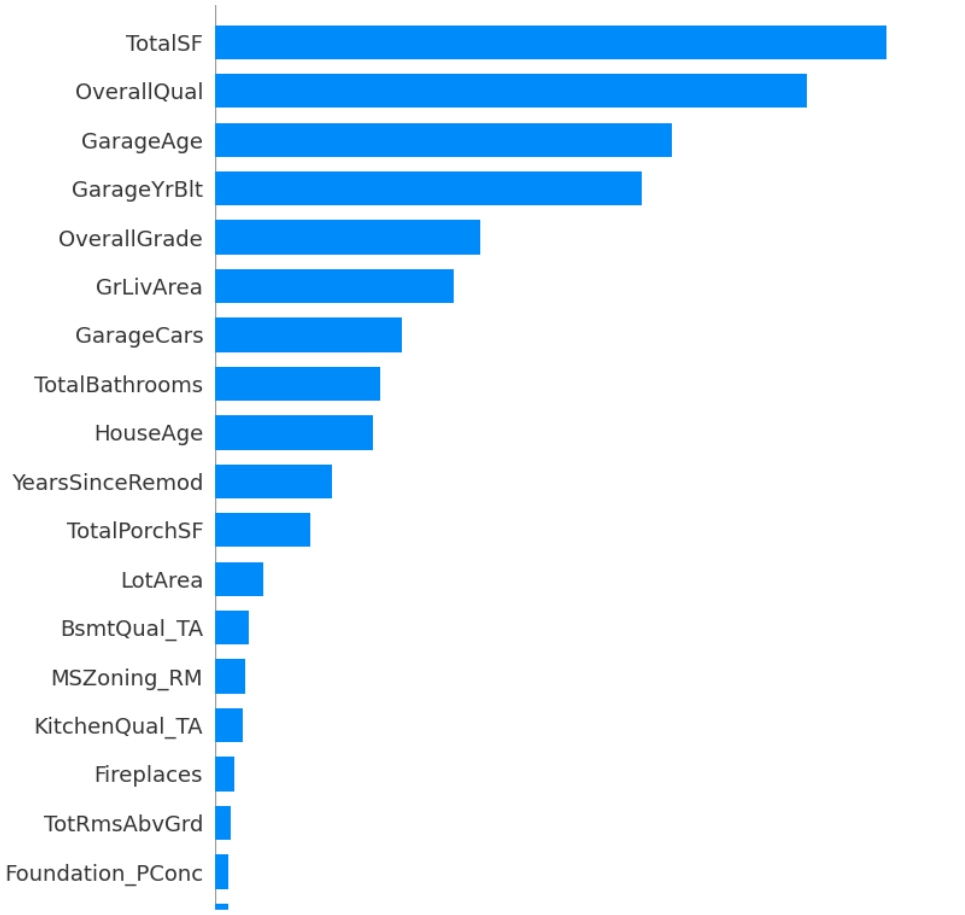
Biểu đồ Summary Plot cho thấy một số đặc trưng có ảnh hưởng mạnh nhất đến dự đoán giá nhà bao gồm:

* **TotalSF (Tổng diện tích)** — là biến ảnh hưởng mạnh nhất; diện tích càng lớn thì giá càng cao.
* **OverallQual & OverallGrade** — chất lượng tổng thể và đánh giá tổng thể của căn nhà tác động tích cực rõ rệt đến giá bán.
* **GrLivArea (Diện tích sàn)** — diện tích sử dụng lớn gắn liền với giá trị nhà cao.
* **GarageCars & GarageAge** — số lượng chỗ để xe có tác động làm tăng giá; nhà có garage mới hơn cũng có xu hướng có giá trị cao hơn.
* **TotalBathrooms, HouseAge, YearsSinceRemod** — các biến về tuổi nhà và thời điểm cải tạo thể hiện tầm quan trọng trung bình.
* Các biến như **LotArea, KitchenQual_TA, BsmtQual_TA** cũng tác động nhưng ít hơn.

Nhìn chung, các đặc trưng liên quan đến **quy mô nhà, chất lượng xây dựng và garage** đóng vai trò quan trọng nhất trong mô hình.

- **2. Feature Importance (mean absolute SHAP):**
  - Biểu đồ này hiển thị **giá trị SHAP trung bình theo trị tuyệt đối** của từng biến, đại diện cho mức độ đóng góp tổng quát của mỗi đặc trưng vào mô hình.
  - Khác với Summary Plot (hiển thị cả phân bổ giá trị cao–thấp), biểu đồ này tập trung vào **độ quan trọng tổng thể**, giúp xác định các biến then chốt ảnh hưởng đến giá nhà một cách trực quan.

Kết quả thu được:



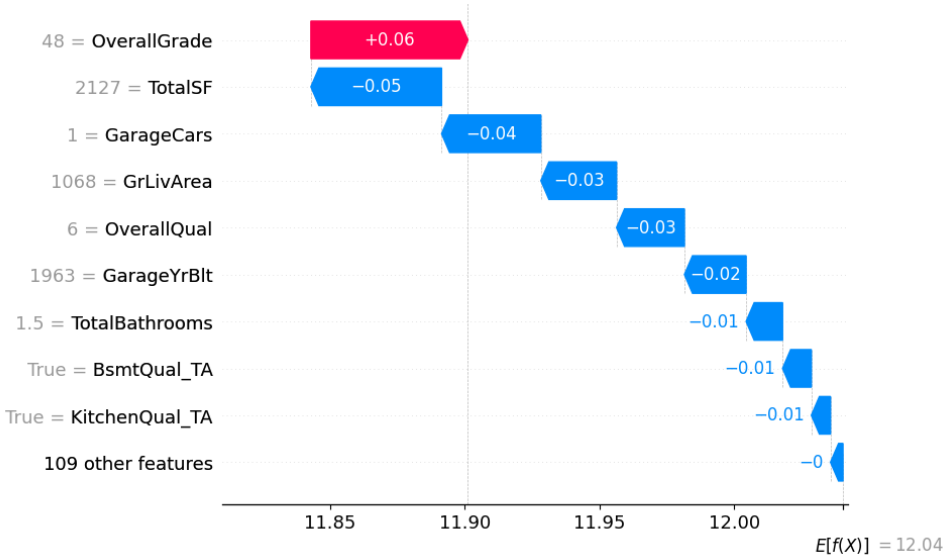
Hình 7: Biểu đồ SHAP dạng cột ngang cho 50 mẫu (chỉ in ra 15 mẫu để quan sát)

  - ∗ **TotalSF** tiếp tục là biến quan trọng nhất, khẳng định diện tích tổng thể là yếu tố quyết định giá trị bất động sản.
  - ∗ **OverallQual** và **GarageAge/GarageYrBlt** xếp sau, cho thấy chất lượng xây dựng và tình trạng/tuổi garage đóng vai trò quan trọng.
  - ∗ **GrLivArea** và **GarageCars** cũng đóng góp đáng kể, cho thấy số lượng chỗ đỗ xe và diện tích sàn đều có ý nghĩa trong dự đoán giá nhà.
  - ∗ Các biến khác như **TotalBathrooms**, **HouseAge**, **YearsSinceRemod**, **TotalPorchSF**, **LotArea** có mức ảnh hưởng trung bình.
  - ∗ Một số biến từ feature engineering như **BsmtQual_TA**, **KitchenQual_TA**, **MSZoning_RM** có ảnh hưởng nhưng thấp hơn, cho thấy chúng bổ trợ mô hình nhưng không phải yếu tố quyết định chính.

- **3. Waterfall Plot (Local SHAP Explanation)**
  - Waterfall plot thể hiện **đóng góp của từng đặc trưng đối với một mẫu cụ thể**, cho thấy cách mô hình đi từ giá trị dự đoán trung bình $E[f(X)]$ đến giá trị dự đoán cuối cùng cho căn nhà đang xét.
  - Biểu đồ minh họa hướng tác động:
    - ∗ Màu đỏ: các đặc trưng làm **tăng** giá trị dự đoán so với trung bình.

∗ Màu xanh: các đặc trưng làm **giảm** giá trị dự đoán so với

Kết quả thu được:



Hình 8: Biểu đồ SHAP dạng thác nước cho 50 mẫu (chỉ in ra 15 mẫu để quan sát)

Ví dụ cụ thể cho một căn nhà (mẫu đầu tiên) trong tập dữ liệu cho thấy:

∗ **OverallGrade** đóng góp mạnh nhất theo hướng **tăng giá** với giá trị SHAP khoảng +0.06, thể hiện căn nhà này được đánh giá tổng thể rất tốt.

∗ Ngược lại, các biến như **TotalSF**, **GarageCars**, **GrLivArea**, và **OverallQual** có giá trị tương đối thấp so với mặt bằng chung, dẫn đến tác động **giảm giá dự đoán** (giá trị SHAP từ −0.05 đến −0.03).

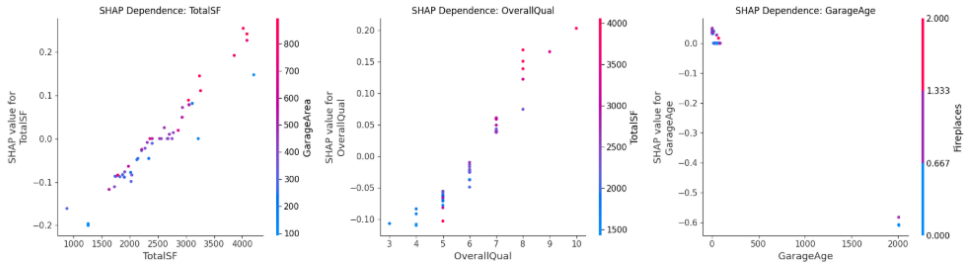
∗ Một số đặc trưng khác như **GarageYrBlt**, **TotalBathrooms**, **BsmtQual_TA**, và **KitchenQual_TA** cũng góp phần nhỏ theo hướng giảm giá trị dự đoán.

∗ Phần còn lại **109 đặc trưng khác** có ảnh hưởng rất nhỏ, gần như không thay đổi dự đoán cuối cùng.

Dựa trên biểu đồ, mô hình dự đoán giá trị căn nhà này thấp hơn mức trung bình chủ yếu bởi diện tích sử dụng và garage ít hơn chuẩn chung, mặc dù chất lượng tổng thể tốt. Điều này minh chứng khả năng **giải thích từng dự đoán riêng lẻ (local interpretability)** của SHAP, giúp người dùng hiểu rõ lý do đằng sau kết quả mô hình.

- **4. Dependence Plot (Top 3 Feature):**
  - Dependence Plot cho phép quan sát mối quan hệ giữa **giá trị đặc trưng** và **đóng góp SHAP tương ứng**, đồng thời biểu diễn **tương tác với một đặc trưng khác** thông qua màu sắc.
  - Mục tiêu: hiểu cách sự thay đổi của từng biến làm tăng/giảm giá dự đoán, và xem liệu có tương tác phi tuyến giữa các biến hay không.

Kết quả thu được:



Hình 9: Biểu đồ SHAP phụ thuộc cho 50 mẫu (chỉ in ra 15 mẫu để quan sát)

(a) TotalSF vs SHAP value (màu theo GarageArea)

- SHAP value tăng rõ rệt khi diện tích tổng thể (**TotalSF**) tăng, thể hiện xu hướng tuyến tính: **nhà càng lớn, giá dự đoán càng cao**.
- Điểm có màu đỏ (GarageArea lớn) có xu hướng nằm cao hơn, cho thấy **GarageArea có tương tác bổ trợ với TotalSF**: nhà lớn và garage lớn càng làm tăng giá trị dự đoán.
- Khi TotalSF thấp (<2000 sqft), SHAP value chủ yếu âm → những căn nhà nhỏ làm giảm giá so với trung bình.

(b) OverallQual vs SHAP value (màu theo TotalSF)

- Chất lượng tổng thể của căn nhà (**OverallQual**) càng cao thì SHAP value càng tăng mạnh, thể hiện đây là yếu tố then chốt nâng giá nhà.
- Các điểm màu đậm (TotalSF lớn) tập trung ở vùng SHAP cao → **chất lượng cao + diện tích lớn** tạo hiệu ứng cộng hưởng làm tăng giá nhà đáng kể.
- Mô hình nhận diện rõ xu hướng phi tuyến: từ mức chất lượng 7 trở lên, mức tăng giá dự đoán mạnh hơn đáng kể.

(c) GarageAge vs SHAP value (màu theo Fireplaces)

- Tuổi garage (**GarageAge**) càng lớn (garage cũ), SHAP value có xu hướng âm, cho thấy garage cũ làm giá trị dự đoán giảm.
- Một số điểm có GarageAge thấp nhưng SHAP âm nhẹ → khả năng do các yếu tố khác không thuận lợi (e.g., diện tích nhỏ hoặc chất lượng trung bình).
- Số lượng fireplace không tạo tương tác mạnh với GarageAge (màu phân bố đều), nghĩa là **Fireplaces không ảnh hưởng đáng kể đến mối quan hệ này**.

Nhìn chung, các Dependence Plot cho thấy mô hình học được các mối quan hệ hợp lý theo tri thức thực tế bất động sản: diện tích lớn, chất lượng tốt và garage mới => **tăng giá trị**; ngược lại diện tích nhỏ, chất lượng thấp, garage cũ => **giảm giá trị**.

6 Ứng dụng Prompting để xây dựng Trợ lý AI định giá bất động sản

Sau khi lựa chọn được mô hình dự đoán tốt nhất (trong dự án này là **Stats_base_raw – Ensemble** với $R^2 \approx 0.90$), bài toán không dừng lại ở việc "dự đoán được giá nhà". Trong thực tế, người dùng (ngân hàng, môi giới, nhà đầu tư, chủ nhà) thường cần thêm:

- giải thích tại sao mức giá này là hợp lý / không hợp lý;

- so sánh với các bất động sản tương đương (comparable sales);
- gợi ý cải tạo để nâng giá;
- phân tích góc nhìn đầu tư (ROI, cho thuê, rủi ro);
- báo cáo thẩm định ở dạng chuẩn hoá, có thể gửi cho khách hàng.

Do đó nhóm bổ sung một tầng ứng dụng sử dụng LLM, gọi là **Agent for Housing Appraisal**. Tầng này nhận đầu vào là **dự đoán từ mô hình ML + bản mô tả cấu trúc của căn nhà** và sinh ra **báo cáo phân tích bất động sản** ở dạng ngôn ngữ tự nhiên. Ý tưởng cốt lõi:

$$\text{ML prediction} \rightarrow \text{Chuẩn hoá đặc trưng} \rightarrow \text{Prompt cho LLM} \rightarrow \text{Báo cáo thẩm định}$$

```
You are a real estate valuation assistant.
The target property (Index 50) is located in Gilbert.
It has 3 bedrooms, 2 garage(s), 1470 square feet of living area, built in 1997, with an overall quality rating of 6.

Actual sale price in dataset: $177,000.
Predicted price from ML model: $175,451.
The prediction differs by -0.9% from the actual sale price.

The dataset's mean sale price is $180,921, ranging from $34,900 to $755,000.

Here are the top 5 comparable properties:
- The first comparable: Located in Gilbert, built in 1997, 1511 sqft, 3 bedrooms, quality 6/10, 2 garage(s), selling for $185,000 (similarity score: 0.992).
- The second comparable: Located in Gilbert, built in 1994, 1481 sqft, 3 bedrooms, quality 6/10, 2 garage(s), selling for $174,000 (similarity score: 0.989).
- The third comparable: Located in Gilbert, built in 1995, 1498 sqft, 3 bedrooms, quality 6/10, 2 garage(s), selling for $187,500 (similarity score: 0.988).
- The fourth comparable: Located in Gilbert, built in 1993, 1470 sqft, 3 bedrooms, quality 6/10, 2 garage(s), selling for $185,000 (similarity score: 0.988).
- The fifth comparable: Located in Gilbert, built in 1993, 1501 sqft, 3 bedrooms, quality 6/10, 2 garage(s), selling for $165,600 (similarity score: 0.982).

Based on these comparables and model estimates, analyze whether the predicted value is reasonable. Explain which features likely contributed most to the price difference.
```

Hình 10: Kết quả dự đoán giá nhà của Agent

6.1 Chuẩn hoá dữ liệu đầu vào cho LLM

Các bản ghi trong dataset thường chứa rất nhiều cột kỹ thuật (ví dụ `GrLivArea`, `OverallQual`, `TotalBsmtSF`, ...) và có thể bị thiếu giá trị.

Để LLM hiểu được, nhóm xây dựng hàm `extract_property_features(...)` nhằm:

1. Gom nhóm thuộc tính theo ngữ nghĩa (thông tin cơ bản, kích thước, chất lượng, tiện ích, chỉ số tài chính);
2. Xử lý thiếu dữ liệu an toàn bằng `safe_get`;
3. Định dạng lại tiền tệ, diện tích, năm xây dựng;
4. Gắn thêm kết quả từ mô hình ML: giá dự đoán, độ tự tin, các đặc trưng quan trọng nhất.

```python
import pandas as pd
from typing import Dict, Any

def extract_property_features(house_data: pd.Series,
                              predictions: Dict = None) -> Dict[str, Any]:
    def safe_get(key, default=0):
        return house_data.get(key, default) if pd.notna(house_data.get(key, default)) else
    default

    features = {
        'basic_info': {
            'sale_price': f"${safe_get('SalePrice'):,.0f}" if 'SalePrice' in house_data
    else "Not available",
```

```
            'predicted_price': f"${predictions.get('predicted_price', 0):,.0f}" if predictions else "Not available",
            'year_built': int(safe_get('YearBuilt')),
            'house_age': int(safe_get('HouseAge', 0)),
            'neighborhood': str(safe_get('Neighborhood', 'Unknown')),
            'house_style': str(safe_get('HouseStyle', 'Unknown')),
            'building_type': str(safe_get('BldgType', 'Unknown'))
        },
        'size_metrics': {
            'total_sqft': f"{safe_get('TotalSF'):,.0f} sq ft",
            'living_area': f"{safe_get('GrLivArea'):,.0f} sq ft",
            'lot_area': f"{safe_get('LotArea'):,.0f} sq ft",
            'bedrooms': int(safe_get('BedroomAbvGr')),
            'total_bathrooms': f"{safe_get('TotalBathrooms', 0):.1f}",
            'garage_cars': int(safe_get('GarageCars'))
        },
        'quality_ratings': {
            'overall_quality': f"{safe_get('OverallQual')}/10",
            'overall_condition': f"{safe_get('OverallCond')}/10",
            'kitchen_quality': str(safe_get('KitchenQual', 'Unknown')),
            'exterior_quality': str(safe_get('ExterQual', 'Unknown'))
        },
        'features_amenities': {
            'central_air': 'Yes' if safe_get('CentralAir') == 'Y' else 'No',
            'fireplaces': int(safe_get('Fireplaces')),
            'basement_sqft': f"{safe_get('TotalBsmtSF'):,.0f} sq ft",
            'garage_area': f"{safe_get('GarageArea'):,.0f} sq ft",
            'total_porch': f"{safe_get('TotalPorchSF', 0):,.0f} sq ft",
            'has_pool': 'Yes' if safe_get('PoolArea') > 0 else 'No'
        },
        'financial_metrics': {
            'price_per_sqft': f"${safe_get('SalePrice', 0) / max(safe_get('TotalSF', 1), 1):.0f}/sq ft",
            'lot_value_ratio': f"{safe_get('SalePrice', 0) / max(safe_get('LotArea', 1), 1):.2f}/sq ft lot"
        }
    }

    if predictions:
        features['ml_analysis'] = {
            'model_confidence': f"{predictions.get('confidence', 0):.1f}%",
            'value_assessment': predictions.get('value_assessment', 'Unknown'),
            'key_drivers': predictions.get('key_drivers', [])
        }

    return features
```

Code Listing 4: Extracting property features for LLM input

Nhờ hàm này, LLM luôn nhận được JSON gọn, nhất quán và có thể hiển thị thẳng cho người dùng.

6.2 Sinh prompt phân tích bất động sản

Nhóm định nghĩa hai kiểu prompt chính:

- **Comprehensive analysis prompt**: cho chatbot/LLM trả về phân tích nhiều chiều (giá thị trường, điểm mạnh, ROI, rủi ro, gợi ý nâng cấp).
- **Formal appraisal report prompt**: cho báo cáo thẩm định có cấu trúc cố định, dễ in và gửi mail.

```python
import json
from typing import Dict, Any

def generate_comprehensive_analysis_prompt(property_features: Dict[str, Any],
                                           market_context: str = "") -> str:
    prompt = f"""
You are an expert real estate analyst with 20+ years of experience in property valuation,
    market analysis, and investment advisory. Analyze the following property
    comprehensively:

 PROPERTY OVERVIEW:
{json.dumps(property_features, indent=2)}

 MARKET CONTEXT:
{market_context}

Please provide a comprehensive analysis covering:

1. Market value assessment
2. Key value drivers
3. Property analysis
4. Investment perspective
5. Improvement recommendations
6. Neighborhood & location
7. Buyer recommendations
8. Potential concerns

Provide specific, actionable insights with professional real estate terminology.
"""
    return prompt
```

Code Listing 5: Generating a comprehensive real-estate analysis prompt

```python
def create_property_report_prompt(property_features: Dict[str, Any],
                                  comparable_properties: list | None = None) -> str:
    prompt = f"""
Create a professional property appraisal report for the following property:

SUBJECT PROPERTY:
{json.dumps(property_features, indent=2)}
"""
    if comparable_properties:
        prompt += f"""
COMPARABLE SALES:
{json.dumps(comparable_properties, indent=2)}
"""

    prompt += """
Please structure your response as a formal appraisal report with these sections:
1. EXECUTIVE SUMMARY
2. PROPERTY DESCRIPTION
3. NEIGHBORHOOD ANALYSIS
4. VALUATION APPROACH
5. RISK ASSESSMENT
6. CONCLUSIONS AND RECOMMENDATIONS
"""
    return prompt
```

Code Listing 6: Generating a formal property appraisal report prompt

6.3 Tìm bất động sản tương đồng (Comparable Sales)

Để việc định giá có cơ sở hơn, agent cần chỉ ra những căn *tương tự* đã giao dịch trong dữ liệu. Hàm `find_comparable_properties_advanced(...)` thực hiện việc này bằng cách chấm điểm tương đồng có trọng số cho từng căn: diện tích sống chiếm 30%, khu phố 20%, chất lượng 20%, tuổi nhà 15%, số phòng ngủ và garage chiếm phần còn lại. Các căn có điểm cao nhất sẽ được đưa vào prompt để LLM so sánh.

```python
def find_comparable_properties_advanced(target_house, house_df, n_comps):
    weights = {
        'sqft': 0.3,
        'neighborhood': 0.2,
        'quality': 0.2,
        'age': 0.15,
        'bedrooms': 0.1,
        'garage': 0.05
    }

    scores = []
    for idx, row in house_df.iterrows():
        if idx == target_house.name:
            continue

        score = 0

        # Square footage similarity
        if pd.notna(row['GrLivArea']) and pd.notna(target_house['GrLivArea']):
            sqft_diff = abs(row['GrLivArea'] - target_house['GrLivArea']) / target_house['GrLivArea']
            score += weights['sqft'] * (1 - min(sqft_diff, 1))

        # Neighborhood match
        if row['Neighborhood'] == target_house['Neighborhood']:
            score += weights['neighborhood']

        # Quality similarity
        if pd.notna(row['OverallQual']) and pd.notna(target_house['OverallQual']):
            qual_diff = abs(row['OverallQual'] - target_house['OverallQual']) / 10
            score += weights['quality'] * (1 - qual_diff)

        # Age similarity
        if pd.notna(row['YearBuilt']) and pd.notna(target_house['YearBuilt']):
            age_diff = abs(row['YearBuilt'] - target_house['YearBuilt']) / 50
            score += weights['age'] * (1 - min(age_diff, 1))

        # Bedroom similarity
        if pd.notna(row['BedroomAbvGr']) and pd.notna(target_house['BedroomAbvGr']):
            bed_diff = abs(row['BedroomAbvGr'] - target_house['BedroomAbvGr']) / 5
            score += weights['bedrooms'] * (1 - min(bed_diff, 1))

        # Garage similarity
        if pd.notna(row['GarageCars']) and pd.notna(target_house['GarageCars']):
            garage_diff = abs(row['GarageCars'] - target_house['GarageCars']) / 3
            score += weights['garage'] * (1 - min(garage_diff, 1))

        scores.append((idx, score))

```

```
    scores.sort(key=lambda x: x[1], reverse=True)
    top_comps = scores[:n_comps]

    comparable_list = []
    for idx, similarity_score in top_comps:
        comp = house_df.loc[idx]
        comparable_list.append({
            'similarity_score': f"{similarity_score:.3f}",
            'price': f"${comp['SalePrice']:,.0f}",
            'sqft': f"{comp['GrLivArea']:,.0f}",
            'price_per_sqft': f"${comp['SalePrice']/comp['GrLivArea']:.0f}",
            'bedrooms': int(comp['BedroomAbvGr']),
            'bathrooms': f"{comp['FullBath'] + comp['HalfBath'] * 0.5:.1f}",
            'quality': f"{comp['OverallQual']}/10",
            'year_built': int(comp['YearBuilt']),
            'neighborhood': comp['Neighborhood']
        })

    return comparable_list
```

Code Listing 7: Finding advanced comparable properties

6.4 Gói thành lớp `PromptM5` và ví dụ chạy

Để tiện tái sử dụng (ví dụ gọi từ API hoặc từ một notebook khác), toàn bộ logic được gói lại thành lớp `PromptM5`. Lớp này:

- lưu dataset đã được engineering;
- tự tính trung bình/giá min/giá max của thị trường;
- cho phép chọn 1 dòng bất kỳ theo index, dự đoán lại bằng mô hình ML;
- tìm comparable;
- sinh prompt cuối cùng để gửi cho LLM.

```
class PromptM5:
    def __init__(self, df_engineered: pd.DataFrame):
        self.df = df_engineered
        self.mean_price = self.df['SalePrice'].mean()
        self.min_price = self.df['SalePrice'].min()
        self.max_price = self.df['SalePrice'].max()

    # ... (method find_comparable_properties_advanced same as above) ...

    def get_price_prompt(self, target_idx: int, model=None, df_ml=None,
                         n_comps: int = 5, context: bool = True):
        target_house = self.df.loc[target_idx]
        comps = self.find_comparable_properties_advanced(target_house, n_comps)

        predicted_price = None
        if model is not None and df_ml is not None:
            features = df_ml.loc[target_idx].values.reshape(1, -1)
            log_pred = model.predict(features)[0]
            predicted_price = np.expm1(log_pred)

        actual_price = target_house["SalePrice"]
```

```
        prompt = f"You are a real estate valuation assistant.\n"
        prompt += f"The target property (Index {target_idx}) is located in {target_house['Neighborhood']}.\n"
        prompt += (
            f"It has {int(target_house['BedroomAbvGr'])} bedrooms, "
            f"{int(target_house['GarageCars'])} garage(s), "
            f"{int(target_house['GrLivArea'])} square feet of living area, "
            f"built in {int(target_house['YearBuilt'])}, "
            f"with an overall quality rating of {target_house['OverallQual']}.\n"
        )

        prompt += f"\nActual sale price in dataset: ${actual_price:,.0f}.\n"
        if predicted_price:
            prompt += f"Predicted price from ML model: ${predicted_price:,.0f}.\n"
            diff_pct = (predicted_price - actual_price) / actual_price * 100
            prompt += f"The prediction differs by {diff_pct:+.1f}% from the actual sale price.\n"

        if context:
            prompt += (
                f"\nThe dataset's mean sale price is ${self.mean_price:,.0f}, "
                f"ranging from ${self.min_price:,.0f} to ${self.max_price:,.0f}.\n"
            )

        prompt += f"\nHere are the top {n_comps} comparable properties:\n"
        # ... loop to append comparables ...
        prompt += (
            "\nBased on these comparables and model estimates, "
            "analyze whether the predicted value is reasonable. "
            "Explain which features likely contributed most to the price difference."
        )
        return prompt
```

Code Listing 8: Wrapper class for valuation prompting

```
prompt_engine = PromptM5(df_engineered)

sample_idx = 50
prompt_text = prompt_engine.get_price_prompt(
    target_idx=sample_idx,
    model=ensemble,        # best model from previous section
    df_ml=X_train_stats,   # ML feature matrix
    n_comps=5
)

print(prompt_text)
```

Code Listing 9: Example: generating an appraisal prompt for one property

6.5 Kết luận

Phần mở rộng này biến mô hình *dự đoán giá nhà* thành một **tác nhân thẩm định giá** thực thụ:

1. mô hình ML cung cấp *giá trị ước tính*;

2. hàm trích xuất biến đổi dữ liệu thô thành JSON dễ hiểu;
3. module tìm comparable bổ sung bằng chứng định giá;
4. prompt mẫu giúp LLM sinh ra báo cáo thẩm định đầy đủ, có thể gửi cho khách hàng hoặc dùng làm ghi chú nội bộ.

Điểm mạnh của cách làm này là **tách được phần dự đoán (ML) khỏi phần trình bày (LLM)**, nhờ vậy khi thay mô hình (ví dụ chuyển từ Ensemble sang Gradient Boosting) ta *không cần* thay prompt, còn khi thay format báo cáo ta *không cần* train lại mô hình.