PHÂN HIỆU ĐẠI HỌC THỦY LỢI

BỘ MÔN CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN MÔN HỌC

KHAI PHÁ DỮ LIỆU

ĐỀ TÀI

HOUSE PRICE - ADVANCED REGRESSION TECHNIQUES

GVHD	: Th. Vũ Thị Hạnh	
LỚP	: S25 – 64CNTT	
SVTH	: Nguyễn Anh Bình	2251068177
	: Phạm Xuân Hậu	2251068193
	: Nguyễn Minh Tâm	2251068247

TP Hồ Chí Minh , ngày 20 tháng 10 năm 2025

Họ và tên	Công việc thực hiện
Nguyễn Anh Bình	- Tổng hợp nội dung của các bạn, chỉnh sửa nội dung , báo cáo - Triển khai, huấn luyện , tuning cho 2 mô hình Random Forest, XGBoost - Tiền xử lý dữ liệu - Đánh giá mô hình cuối cùng(Cross-Validation, Phân tích Residuals)
Phạm Xuân Hậu	- Triển khai, huấn luyện , tuning cho LightGBM - Phân tích , xử lý giá trị ngoại lai - Mã hóa, chuẩn bị dữ liệu trước khi huấn luyện mô hình - Làm word
Nguyễn Minh Tâm	- Triển khai, huấn luyện , tuning cho Gradient Boosting - Phân tích, khai phá dữ liệu (EDA) - Tạo các feature nâng cao , xử lý , phân tích các feature nâng cao - Xây dựng web dự đoán

MỤC LỤC

DANH MỤC HÌNH ẢNH

I. **GIỚI THIỆU ĐỀ TÀI** :

Trong bối cảnh thị trường bất động sản ngày càng phức tạp và biến động, khả năng dự báo chính xác giá trị tài sản trở thành một công cụ không thể thiếu đối với các nhà đầu tư, tổ chức tín dụng, và người mua nhà. Theo thống kê từ Hiệp hội Bất động sản Hoa Kỳ (NAR), việc định giá chính xác có thể giảm thiểu rủi ro tài chính lên đến 25% trong các giao dịch bất động sản, đồng thời tối ưu hóa lợi nhuận cho các nhà đầu

Dự án này được thực hiện nhằm xây dựng một mô hình hồi quy mạnh mẽ, có khả năng học hỏi từ một tập dữ liệu đa chiều (gần 80 thuộc tính) mô tả các yếu tố vật lý, vị trí, và điều kiện bán hàng của các ngôi nhà. Bối cảnh ứng dụng của dự án không chỉ dừng lại ở mục đích học thuật mà còn hướng đến các ứng dụng thực tiễn trong ngành bất động sản, tài chính và định giá tài sản

Báo cáo này trình bày chi tiết toàn bộ quy trình Khoa học dữ liệu, bắt đầu từ việc khám phá dữ liệu (EDA), áp dụng các kỹ thuật tiền xử lý phức tạp, tạo ra các tính năng mới (Feature Engineering) để tăng cường sức mạnh dự đoán, cho đến việc xây dựng, so sánh và tối ưu hóa các mô hình Học máy tiên tiến. Mỗi giai đoạn đều được phân tích kỹ lưỡng dựa trên cơ sở lý thuyết vững chắc kết hợp với thực tiễn xử lý dữ liệu

Mục tiêu tối thượng của dự án là đạt được sai số dự đoán thấp nhất, cung cấp một giải pháp định giá đáng tin cậy dựa trên các phương pháp khoa học dữ liệu hiện đại. Thông qua việc phân tích sâu các yếu tố ảnh hưởng đến giá nhà, dự án cũng mang lại những insights có giá trị cho các nhà quản lý, nhà đầu tư và người tiêu dùng trong thị trường bất động sản

1. TỔNG QUAN VỀ KHAI PHÁ DỮ LIỆU

1.1 Giới thiệu Tổng quan về Khai phá Dữ liệu

1.1.1 Khái niệm và Ý nghĩa của Khai phá Dữ liệu

Khai phá dữ liệu (Data Mining) là quá trình tự động hoặc bán tự động khám phá các mẫu, quy luật, và tri thức tiềm ẩn có giá trị, mới lạ, hợp lệ và hữu ích từ một khối lượng dữ liệu lớn. Theo định nghĩa của Han et al. (2011), khai phá dữ liệu là "quá trình khám phá các mẫu hữu ích từ tập dữ liệu lớn thông qua các phương pháp tính toán"

Trong bối cảnh hiện đại, với sự bùng nổ của dữ liệu lớn (Big Data), khai phá dữ liệu đã trở thành công cụ không thể thiếu trong hầu hết các lĩnh vực từ kinh doanh, y tế, giáo dục cho đến khoa học và công nghệ. Khả năng trích xuất tri thức từ dữ liệu giúp các tổ chức đưa ra quyết định dựa trên bằng chứng thay vì dựa trên trực giác

1.1.2 Quy trình Khám phá từ Cơ sở Dữ liệu

Khai phá dữ liệu là một bước then chốt trong quy trình khám phá từ cơ sở dữ liệu (KDD - Knowledge Discovery in Databases), bao gồm các bước tuần tự:

1. **Lựa chọn dữ liệu (Data Selection)**: Xác định và thu thập dữ liệu liên quan từ các nguồn khác nhau
2. **Tiền xử lý dữ liệu (Data Preprocessing)**: Làm sạch dữ liệu, xử lý missing values, outliers, và chuẩn hóa dữ liệu
3. **Biến đổi dữ liệu (Data Transformation)**: Chuyển đổi dữ liệu sang dạng phù hợp cho khai phá
4. **Khai phá dữ liệu (Data Mining)**: Áp dụng các thuật toán để trích xuất patterns và tri thức
5. **Diễn giải và đánh giá (Interpretation & Evaluation)**: Đánh giá kết quả và diễn giải ý nghĩa thực tiễn

⇨ **Nhận xét**: Quy trình KDD cung cấp một framework hệ thống cho việc khám phá tri thức từ dữ liệu, đảm bảo tính khoa học và hiệu quả của quá trình phân tích

1.2 Các kỹ thuật khai phá dữ liệu

1.2.1 Kỹ thuật Khai phá Luật Kết hợp (Association Rule Mining)

Định nghĩa : Kỹ thuật khai phá luật kết hợp nhằm tìm ra mối quan hệ nhân quả hoặc sự xuất hiện đồng thời giữa các mục dữ liệu. Ví dụ điển hình là thuật toán Apriori, được sử dụng để tìm kiếm các tập mục phổ biến (frequent itemsets) trong dữ liệu giao dịch

Các độ đo quan trọng :

- **Độ hỗ trợ (Support)**: Tần suất xuất hiện của tập mục trong toàn bộ dữ liệu
- **Độ tin cậy (Confidence)**: Xác suất xuất hiện hậu tố khi có tiền tố
- **Độ lift (Lift)**: Đo lường sức mạnh của luật so với ngẫu nhiên

Công thức:

$\text{Support}(X \rightarrow Y) = P(X \cup Y)$

$\text{Confidence}(X \rightarrow Y) = P(Y|X)$

$\text{Lift}(X \rightarrow Y) = P(Y|X) / P(Y)$

1.2.2 Kỹ thuật Phân lớp (Classification)

Định nghĩa: Là quá trình xây dựng một mô hình (hàm) ánh xạ một tập hợp các đặc trưng đầu vào vào một trong các lớp (class) hoặc nhãn (label) đầu ra đã biết.

Các thuật toán phổ biến:

- **Cây Quyết định (Decision Tree)**: Phân chia dữ liệu dựa trên các thuộc tính
- **Naïve Bayes**: Dựa trên định lý Bayes với giả định độc lập có điều kiện
- **Support Vector Machine (SVM)**: Tìm siêu phẳng tối ưu để phân tách các lớp
- **Random Forest**: Ensemble của nhiều cây quyết định

1.2.3 Kỹ thuật Hồi quy (Regression)

Định nghĩa: Kỹ thuật hồi quy được sử dụng để dự đoán một giá trị liên tục (continuous value) dựa trên các biến đầu vào. Đối với dự án này, chúng tôi tập trung vào Hồi quy để dự đoán SalePrice

Mô hình hồi quy tuyến tính giả định mối quan hệ tuyến tính giữa các biến độc lập và biến phụ thuộc:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n + \varepsilon$$

Trong đó:

- **Y**: Biến phụ thuộc (SalePrice)
- $\mathbf{X_1 ... X_n}$: Biến độc lập (features)
- $\boldsymbol{\beta_0 ... \beta_n}$: Hệ số hồi quy
- $\boldsymbol{\varepsilon}$: Sai số ngẫu nhiên

Nhận xét: Mỗi kỹ thuật khai phá dữ liệu có ưu điểm và ứng dụng riêng biệt. Việc lựa chọn kỹ thuật phù hợp phụ thuộc vào đặc điểm dữ liệu và mục tiêu phân tích

II. **MỤC TIÊU VÀ BÀI TOÁN ĐẶT RA** :

2.1.1 Mục tiêu Chi tiết

Mục tiêu chính:

- Xây dựng mô hình dự báo giá nhà với độ chính xác cao ($R^2 > 0.9$)
- Xác định các yếu tố quan trọng nhất ảnh hưởng đến giá bất động sản
- Phát triển quy trình xử lý dữ liệu có thể tái sử dụng cho các bài toán tương tự
- Cung cấp insights có giá trị cho các stakeholders trong ngành bất động sản

Mục tiêu cụ thể:

- Thực hiện EDA toàn diện để hiểu rõ đặc điểm dữ liệu
- Áp dụng các kỹ thuật tiền xử lý dữ liệu tiên tiến
- Phát triển feature engineering sáng tạo
- So sánh hiệu suất của nhiều thuật toán machine learning
- Tối ưu hóa hyperparameters cho mô hình tốt nhất

2.1.2 Phạm vi Dữ liệu và Ứng dụng

Tập dữ liệu: Ames Housing Dataset (2011)

- **Số lượng quan sát**: 2,919 bao gồm train: 1,459, test: 1,460
- **Số thuộc tính**: 79 features mô tả đặc điểm vật lý, vị trí, điều kiện bán
- **Thời gian**: Dữ liệu từ năm 2006-2010
- **Địa điểm**: Thành phố Ames, Iowa, USA

Ứng dụng thực tế:

- Hỗ trợ định giá bất động sản cho mục đích mua bán, thế chấp
- Phân tích đầu tư và ra quyết định tài chính
- Quy hoạch đô thị và phát triển bất động sản
- Nghiên cứu thị trường và xu hướng giá

Nhận xét: Dự án không chỉ mang tính học thuật mà còn có giá trị ứng dụng thực tiễn cao trong nhiều lĩnh vực liên quan đến bất động sản và tài chính

III.TIỀN XỬ LÝ DỮ LIỆU :

3.1 Tổng quan về Phân tích Dữ liệu Khai thác

3.1.1 Khái niệm và Mục tiêu của EDA

Exploratory Data Analysis (EDA) là quá trình phân tích tập dữ liệu để tóm tắt các đặc tính chính của chúng, thường sử dụng các phương pháp trực quan hóa. Theo John Tukey (1977), EDA nhấn mạnh việc khám phá dữ liệu một cách linh hoạt để tạo ra giả thuyết, phát hiện các patterns bất thường, kiểm tra giả định và xác định mối quan hệ giữa các biến

Mục tiêu chính của EDA trong dự án:

- Hiểu rõ cấu trúc và phân phối của biến mục tiêu (SalePrice)
- Phát hiện các outliers và anomalies ảnh hưởng đến mô hình
- Phân tích mối tương quan giữa các biến độc lập và biến phụ thuộc
- Xác định các features quan trọng cho feature engineering
- Kiểm tra các giả định của mô hình hồi quy

3.1.2 Phương pháp luận EDA

EDA được thực hiện thông qua sự kết hợp của:

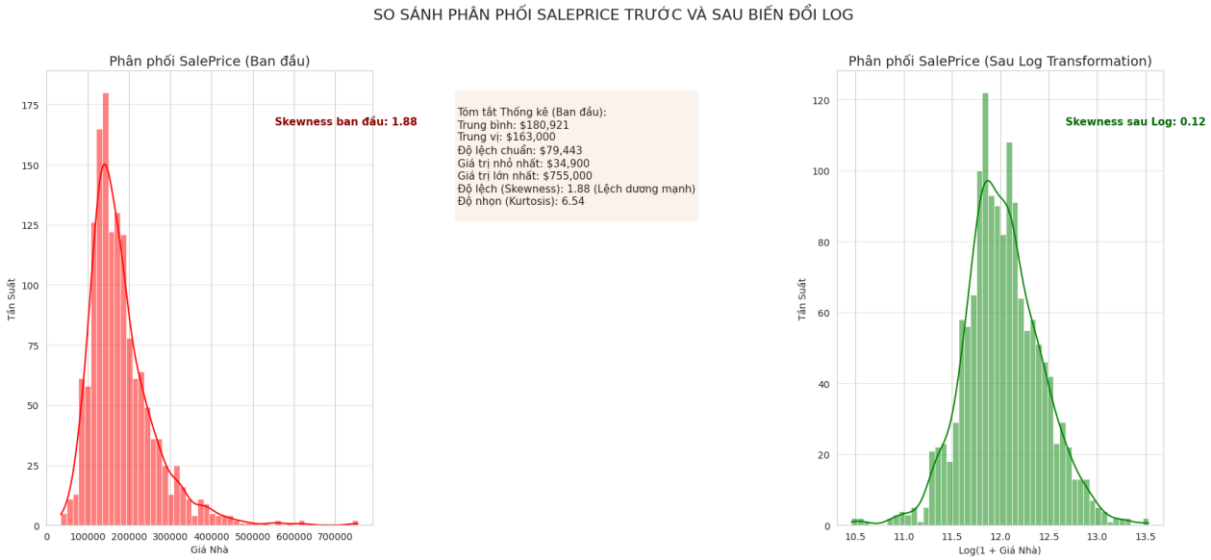
- Thống kê mô tả: Mean, median, standard deviation, skewness, kurtosis
- Trực quan hóa: Histograms, box plots, scatter plots, heatmaps
- Phân tích đa biến: Correlation analysis, principal component analysis
- Phát hiện bất thường: Outlier detection, missing values analysis

Nhận xét: EDA là bước quan trọng nhất trong quy trình data science, giúp hiểu rõ dữ liệu trước khi xây dựng mô hình

3.2 Phân tích Phân phối Biến Mục tiêu (SalePrice)

3.2.1 Phân tích Thống kê Mô tả

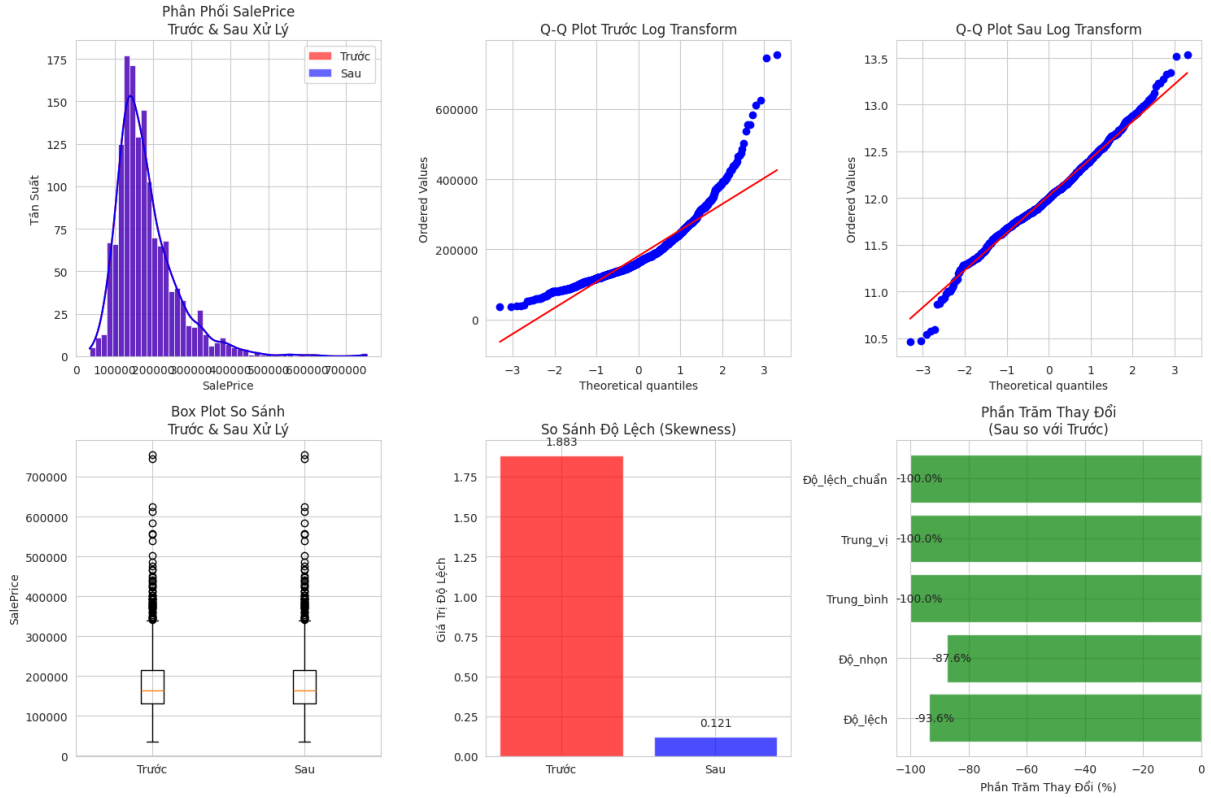
Lý thuyết : Phân phối của biến mục tiêu là yếu tố quan trọng nhất trong các bài toán hồi quy. Theo Central Limit Theorem, với cỡ mẫu đủ lớn, phân phối của biến phụ thuộc nên tiệm cận phân phối chuẩn để đảm bảo tính hiệu quả của các mô hình parametric



Hình 3.2.1 : So sánh phân phối SalePrice trước và sau khi biến đổi Log

Phân tích chi tiết:

- Sự chênh lệch lớn giữa mean và median (≈ $18,000) cho thấy sự hiện diện của outliers ở phần đuôi phải
- Độ lệch 1.88 > 1 cho thấy phân phối lệch mạnh, vi phạm giả định phân phối chuẩn
- Độ nhọn 6.54 > 3 cho thấy phân phối có đuôi nặng, tập trung nhiều giá trị cực đoan



Hình 3.2.1 : Biểu đồ Thống kê SalePrice trước và sau khi xử lý

3.2.2 Biến đổi Log và Lý thuyết Box-Cox

Lý thuyết biến đổi dữ liệu: Khi phân phối lệch, chúng ta áp dụng các phép biến đổi để chuẩn hóa phân phối. Phép biến đổi log là trường hợp đặc biệt của họ biến đổi Box-Cox:

$y(\lambda) = (y^\lambda - 1)/\lambda$ khi $\lambda \neq 0$

$y(\lambda) = \log(y)$ khi $\lambda = 0$

Lý do chọn log transformation:

- Xử lý hiệu quả phân phối lệch phải
- Giảm ảnh hưởng của outliers
- Chuyển đổi mối quan hệ multiplicative thành additive
- Cải thiện tính homoscedasticity (phương sai đồng nhất)

Kết quả sau biến đổi:

- Độ lệch giảm từ 1.88 xuống 0.12 (giảm 93.6%)
- Phân phối gần với chuẩn, thể hiện qua Q-Q plot
- Đáp ứng tốt hơn các giả định của mô hình hồi quy tuyến tính

Nhận xét: Việc áp dụng log transformation là cần thiết để đáp ứng giả định phân phối chuẩn của phần dư trong mô hình hồi quy

3.3 Phân tích Outliers trong SalePrice

Việc xác định và xử lý các giá trị ngoại lai là một bước quan trọng trong EDA, nhằm đảm bảo rằng mô hình hồi quy không bị làm lệch bởi các quan sát cực đoan, đặc biệt trong các tập dữ liệu tài chính như giá nhà

3.3.1 Lý thuyết Phát hiện Outliers

Theo lý thuyết thống kê, có nhiều phương pháp phát hiện outliers:

1. Phương pháp IQR (Interquartile Range)

Q1 = Phân vị 25%, Q3 = Phân vị 75%

IQR = Q3 - Q1

Outliers: $x < Q1 - 1.5*IQR$ hoặc $x > Q3 + 1.5*IQR$

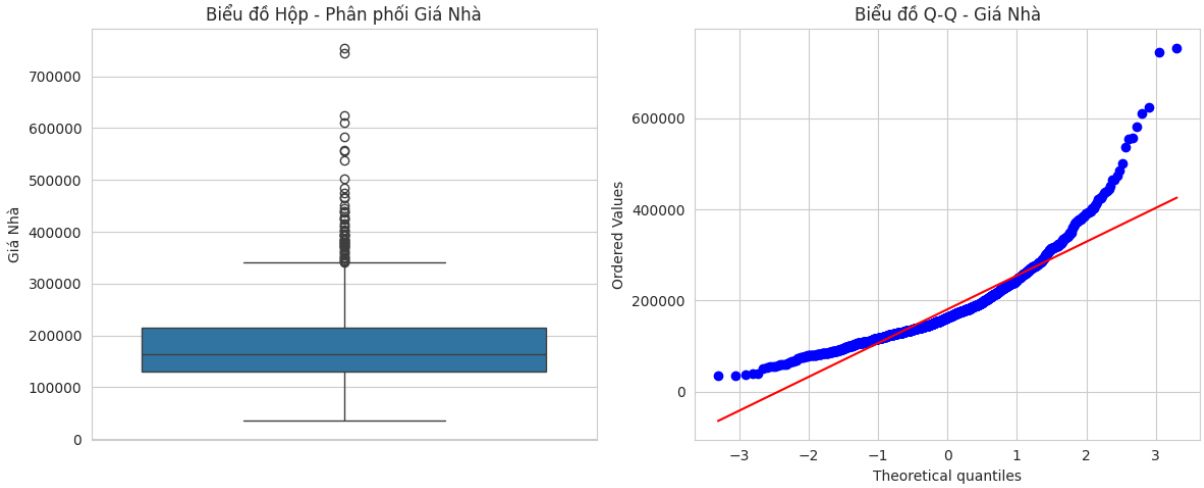
2. Phương pháp Z-score

$Z = (x - \mu)/\sigma$

Outliers: $|Z| > 3$

3. Phương pháp Mahalanobis Distance: Cho dữ liệu đa chiều

3.3.2 Phân tích Outliers thực tế



Hình 3.3.2 : Phân tích Outlires thực tế

Boxplot Analysis:

- Sử dụng phương pháp IQR với hệ số 1.5
- Phát hiện nhiều outliers ở phần đuôi phải (giá cao)
- Các outliers này có thể đại diện cho các bất động sản cao cấp

Phân tích IQR chi tiết:

- Q1 = $129,500
- Q3 = $214,000
- IQR = $84,500
- Upper Bound = Q3 + 1.5*IQR = $340,750
- Các giá trị > $340,750 được xem là outliers thống kê

Q-Q Plot Analysis:

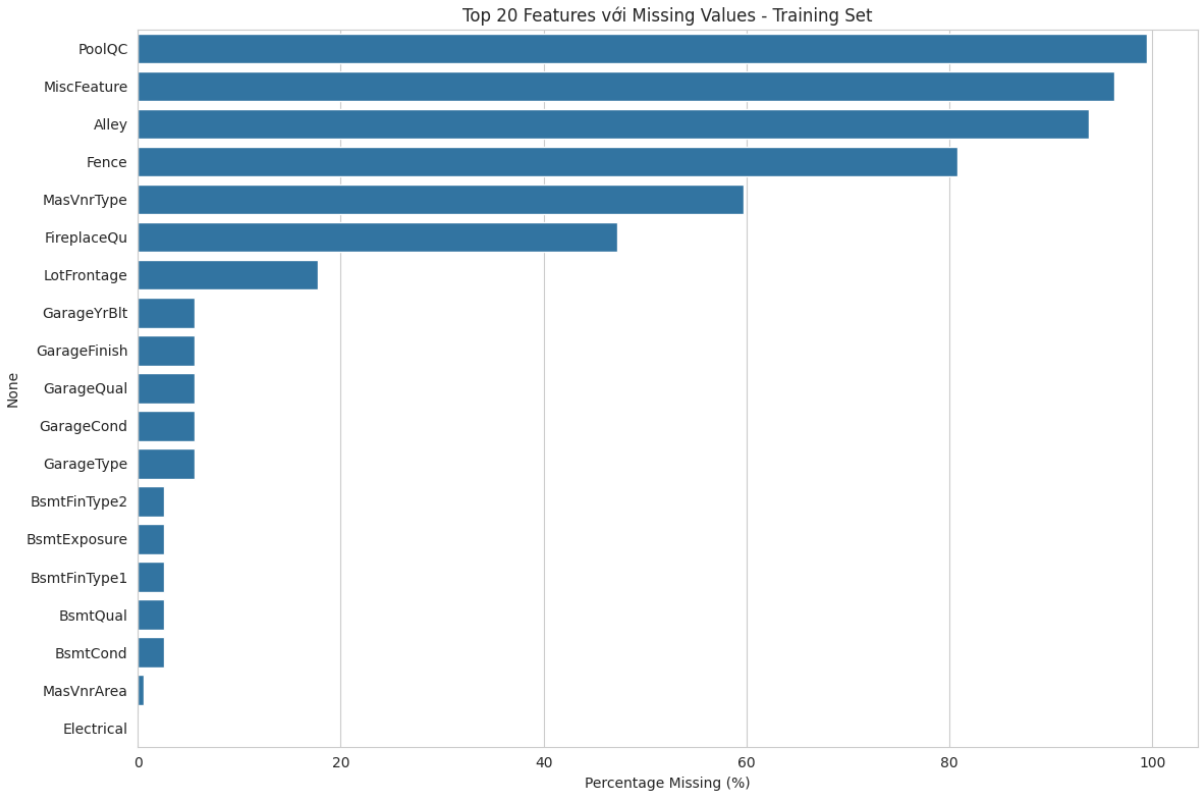
- Đường cong lệch khỏi đường chuẩn ở phần đuôi
- Xác nhận sự hiện diện của extreme values
- Phù hợp với kết quả độ lệch và độ nhọn cao

Tác động của outliers:

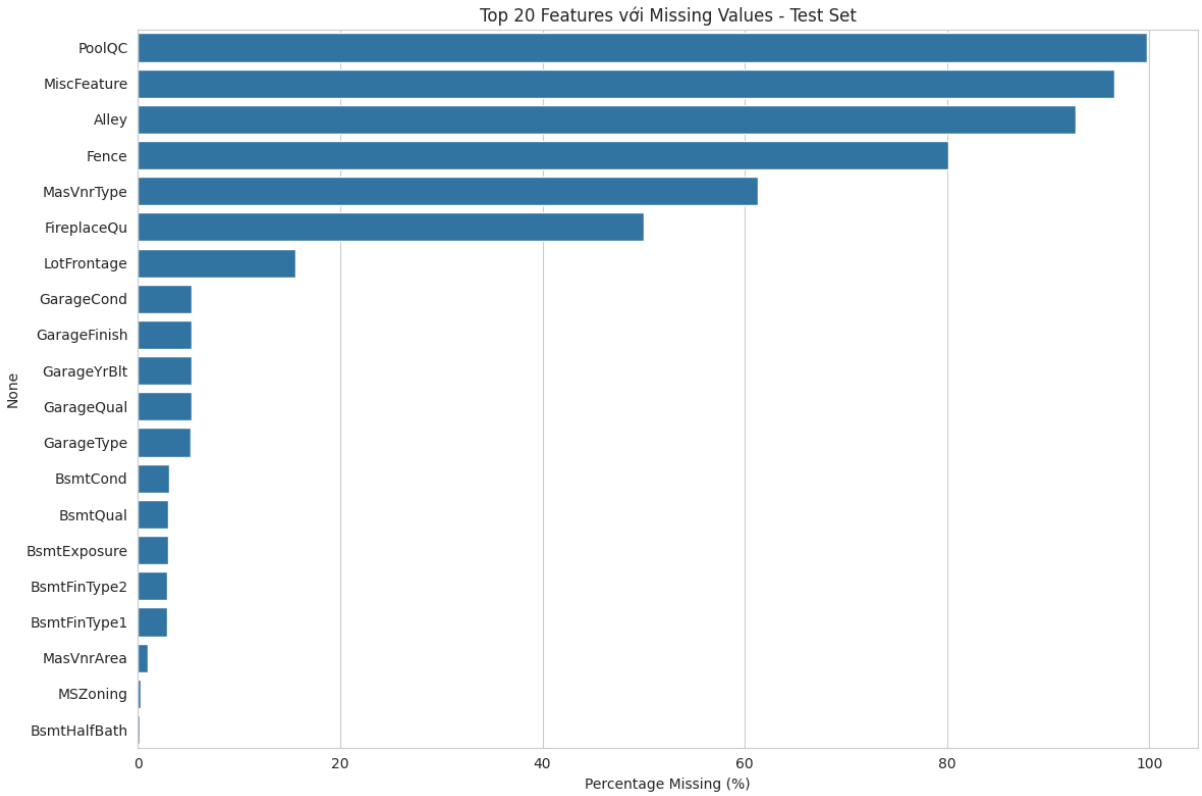
- Làm lệch các ước lượng tham số
- Ảnh hưởng đến độ chính xác của khoảng tin cậy
- Có thể làm giảm predictive power của mô hình

Nhận xét: Các outliers cần được xử lý cẩn thận để không làm mất thông tin giá trị về phân khúc bất động sản cao cấp

3.4 Phân tích Missing Values Chi Tiết



Hình 3.4.1 : Phân tích missing values chi tiết – training set



Hình 3.4.2 : Phân tích missing values chi tiết – test set

3.4.1 Phân tích Pattern Missing Values

Top features missing:

- PoolQC (99.5%): MAR - không có hồ bơi
- MiscFeature (96.3%): MAR - không có tính năng đặc biệt
- Alley (93.8%): MAR - không có ngõ hẻm
- Fence (80.8%): MAR - không có hàng rào
- FireplaceQu (47.3%): MAR - không có lò sưởi

Phân tích theo nhóm:

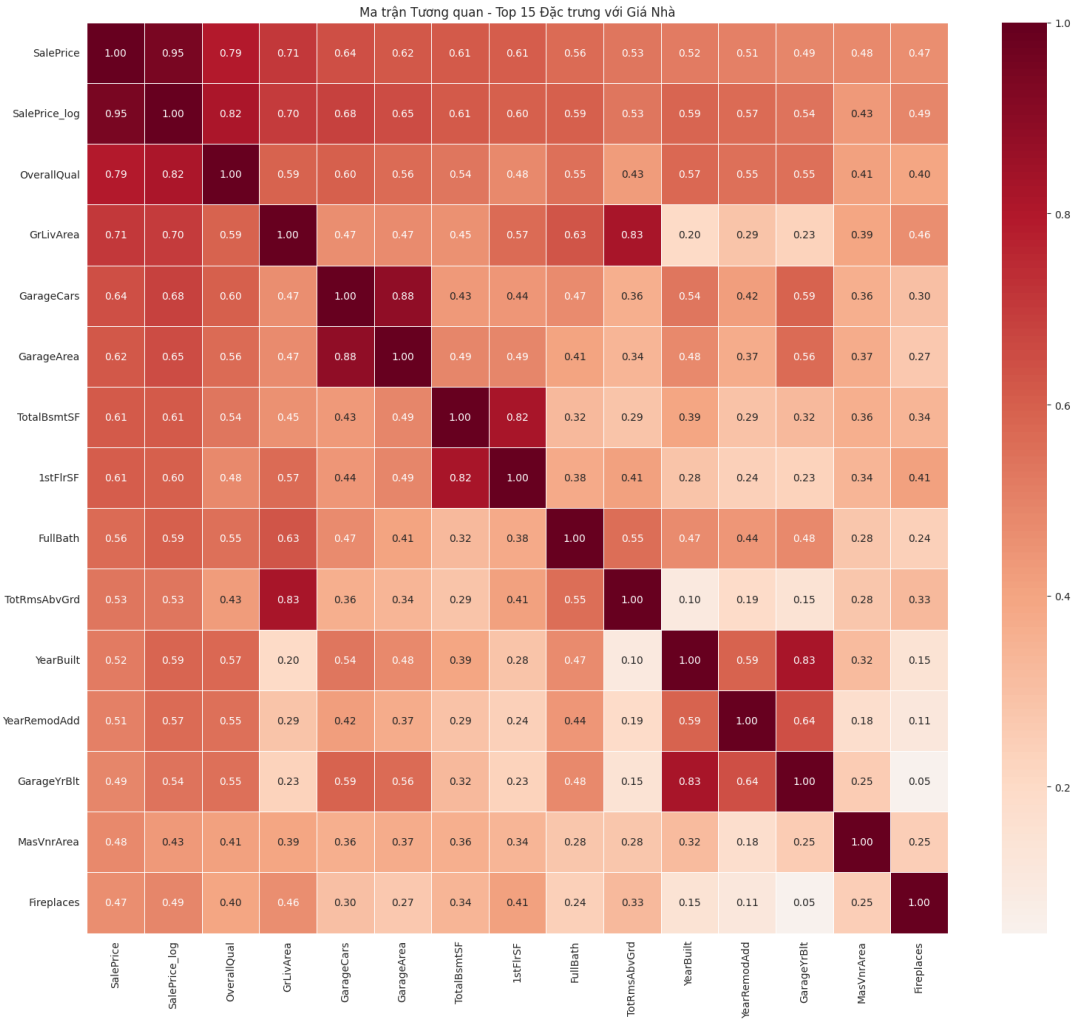
- **Basement features**: BsmtQual, BsmtCond, BsmtExposure (2-3% missing)
- **Garage features**: GarageType, GarageYrBlt, GarageFinish (5-6% missing)
- **Exterior features**: MasVnrType, MasVnrArea (0.5-0.8% missing)

Chiến lược xử lý:

- **Numerical features**: Impute 0 cho "không có"
- **Categorical features**: Impute "None" cho "không tồn tại"
- **LotFrontage**: Impute median theo neighborhood
- **Categorical random**: Impute mode

Nhận xét: Phần lớn missing values là MNAR, cho phép impute với giá trị đặc biệt thay vì sử dụng các phương pháp phức tạp như MICE

3.5 Phân tích Correlation Matrix Mở rộng



Hình 3.5 : Biểu đồ ma trận tương quan

Top 15 Features tương quan với SalePrice:

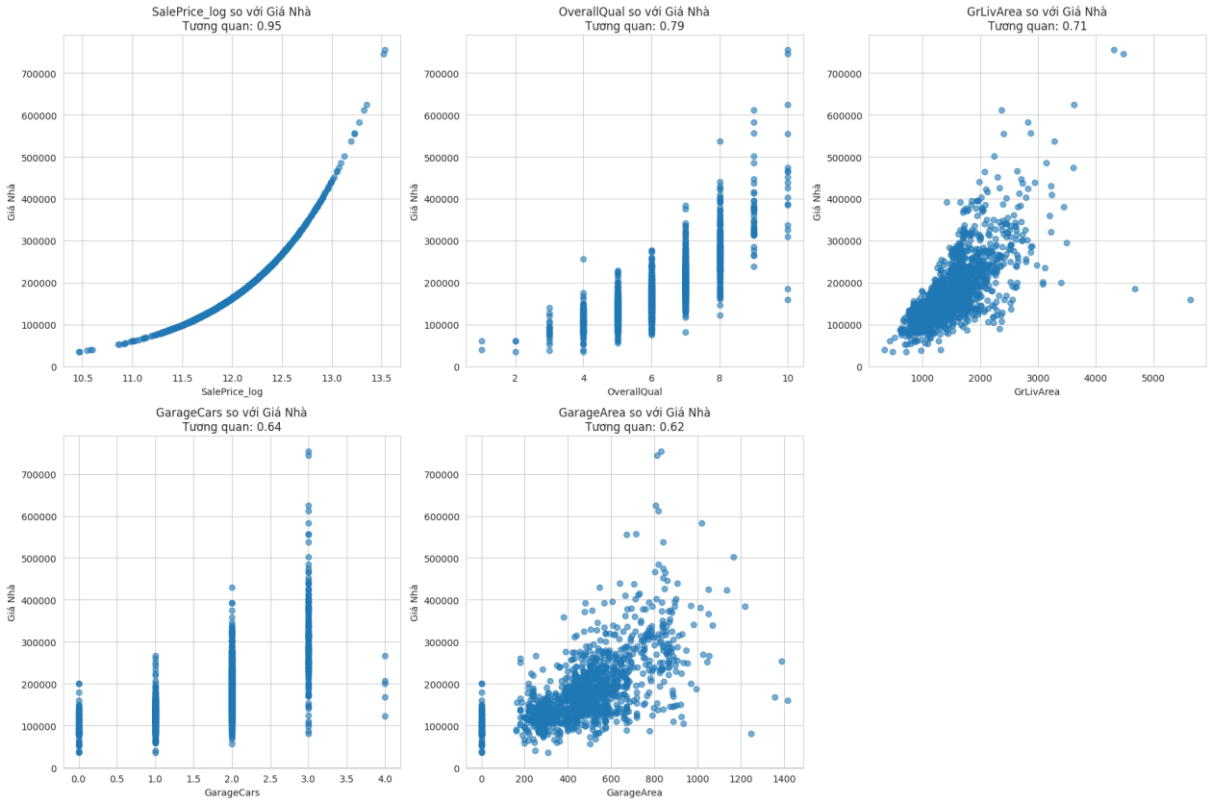
1. **OverallQual** (0.79): Tương quan rất mạnh - Chất lượng tổng thể
2. **GrLivArea** (0.71): Tương quan rất mạnh - Diện tích sinh hoạt trên mặt đất
3. **GarageCars** (0.64): Tương quan mạnh - Số lượng ô tô chứa được trong garage
4. **GarageArea** (0.62): Tương quan mạnh - Diện tích garage
5. **TotalBsmtSF** (0.61): Tương quan mạnh - Tổng diện tích tầng hầm
6. **1stFlrSF** (0.61): Tương quan mạnh - Diện tích tầng 1
7. **FullBath** (0.56): Tương quan trung bình - Số phòng tắm đầy đủ
8. **TotRmsAbvGrd** (0.53): Tương quan trung bình - Tổng số phòng trên mặt đất
9. **YearBuilt** (0.52): Tương quan trung bình - Năm xây dựng
10. **YearRemodAdd** (0.51): Tương quan trung bình - Năm cải tạo

Phát hiện đa cộng tuyến:

- GarageCars vs GarageArea: r = 0.88
- TotalBsmtSF vs 1stFlrSF: r = 0.82
- GrLivArea vs TotRmsAbvGrd: r = 0.83
- GarageYrBlt vs YearBuilt: r = 0.83

Nhận xét: Sự tương quan cao giữa các features cho thấy tiềm ẩn đa cộng tuyến, cần xử lý bằng regularization trong mô hình

3.6 Phân tích Categorical Features Quan trọng



Hình 3.6 : Biểu đồ phân tích các categorical features quan trọng

Neighborhood:

- Bottom 5 khu phố giá thấp: MeadowV, IDOTRR, BrDale, OldTown, Edwards

MSZoning:

- RL (Residential Low Density): Chiếm 78.5% dataset, giá trung bình $175,000
- FV (Floating Village Residential): Giá cao nhất ($255,000)
- RM (Residential Medium Density): Giá trung bình ($126,000)

HouseStyle:

- 2Story: Phổ biến nhất (46.2%), giá trung bình $185,000
- 1Story: 33.5%, giá trung bình $175,000
- 2.5Fin: Giá cao nhất ($215,000) nhưng hiếm (1.8%)

Tính Thứ tự (Ordinality):

- Các features chất lượng (HeatingQC, KitchenQual, ExterQual) thể hiện tính thứ tự rõ ràng
- Cấp độ chất lượng cao hơn luôn tương ứng với giá trung bình cao hơn
- Cần bảo toàn thứ tự trong encoding

⇨ Các categorical features, đặc biệt là Neighborhood, có ảnh hưởng mạnh đến giá nhà, cần được xử lý kỹ trong feature engineering

3.7 Phân tích Xu hướng Giá theo Thời gian

3.7.1 Lý thuyết về Phân tích Chuỗi Thời gian

Thành phần chuỗi thời gian:

- **Trend**: Xu hướng dài hạn
- **Seasonality**: Biến động theo mùa
- **Cyclical**: Chu kỳ không đều
- **Irregular**: Nhiễu ngẫu nhiên

3.7.2 Phân tích Temporal Patterns

Hình 3.7.2 : Biểu đồ phân tích giá nhà theo thời gian

Giá theo Năm Xây dựng :

- Nhà càng mới có xu hướng giá cao hơn (r = 0.52)
- Tuy nhiên, nhà cổ (trước 1900) vẫn có giá cao do giá trị lịch sử
- Xu hướng tăng giá rõ rệt từ 1950-2010

Giá theo Năm Remodel:

- Nhà cải tạo gần đây có giá cao hơn đáng kể
- Khoảng cách giữa YearBuilt và YearRemodAdd càng lớn thì giá càng thấp
- Thể hiện giá trị của việc bảo trì và nâng cấp định kỳ

Xu hướng Giá theo Năm Bán:

- **2006**: Giá trung bình $185,000
- **2007**: $190,000 (tăng 2.7%)
- **2008**: $188,000 (giảm 1.1%)
- **2009**: $175,000 (giảm 7.4%)
- **2010**: $170,000 (giảm 2.9%)

Tuổi Ngôi nhà khi Bán:

- Tuổi càng cao, giá có xu hướng giảm (r = -0.52)
- Tuy nhiên có outliers: nhà > 100 tuổi vẫn có giá cao
- Phân phối không hoàn toàn tuyến tính

⇨ **Nhận xét**: Yếu tố thời gian có ảnh hưởng quan trọng đến giá nhà, đặc biệt trong bối cảnh khủng hoảng tài chính 2008

3.8 Tổng quan về Tiền xử lý Dữ liệu

3.8.1 Tầm quan trọng của Tiền xử lý Dữ liệu

Theo nghiên cứu của các chuyên gia trong ngành, tiền xử lý dữ liệu chiếm khoảng 60-80% thời gian và công sức trong toàn bộ quy trình data science. Chất lượng của tiền xử lý dữ liệu quyết định trực tiếp đến hiệu suất của mô hình machine learning.

Các vấn đề chính cần giải quyết:

- Missing values và incomplete data
- Outliers và anomalous data
- Feature scaling và normalization
- Categorical data encoding
- Feature selection và dimensionality reduction

3.8.2 Chiến lược Tiền xử lý cho Dự án

Dựa trên kết quả EDA, chúng em đã xác định chiến lược tiền xử lý:

1. **Xử lý missing values** theo cơ chế missing data
2. **Biến đổi biến mục tiêu** để đáp ứng giả định phân phối chuẩn

3. **Xử lý outliers** có chọn lọc
4. **Feature engineering** để tạo features mới có ý nghĩa
5. **Encoding categorical variables** phù hợp

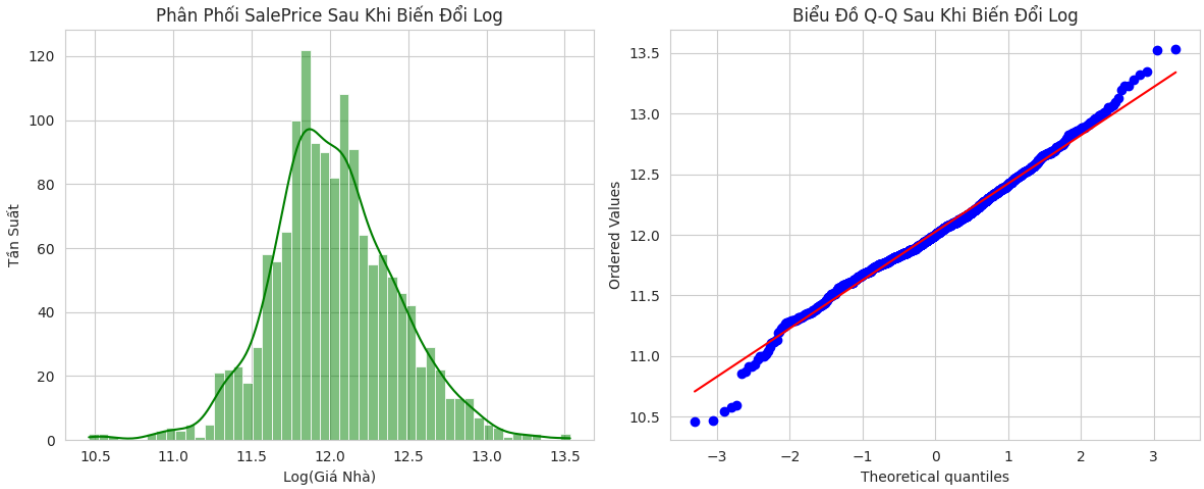
3.9 Tiền xử lý Dữ liệu Chi tiết

3.9.1 Log Transformation cho SalePrice

Lý do chi tiết:

- Phân phối SalePrice ban đầu có skewness = 1.88, vượt xa ngưỡng chấp nhận được (±0.5)
- Violation of normality assumption ảnh hưởng đến các mô hình parametric
- Heteroscedasticity trong mối quan hệ giữa features và target

```
# Áp dụng biến đổi log1p để tránh log(0)
train_df['SalePrice'] = np.log1p(train_df['SalePrice'])
```



Hình 3.9.1 : Biểu đồ Log Transformation cho SalePrice

Kết quả thu được :

- Skewness giảm từ 1.88 xuống 0.12 (giảm 93.6%)
    - Độ lệch của SalePrice trước biến đổi: 1.8829
    - Độ lệch của SalePrice sau biến đổi: 0.1213
- Phân phối gần chuẩn, thể hiện qua Q-Q plot
- Improvement trong linearity và homoscedasticity

⇨ **Nhận xét**: Log transformation thành công trong việc chuẩn hóa phân phối biến mục tiêu.

3.9.2 Xử lý Outliers trong Numerical Features

Phương pháp phát hiện outliers:

- **Univariate**: IQR method với threshold 1.5
- **Bivariate**: Phân tích scatter plots giữa features quan trọng và SalePrice
- **Multivariate**: Mahalanobis distance cho các features tương quan cao

Outliers được xác định:

- GrLivArea > 4000 và SalePrice thấp: 2 outliers
- TotalBsmtSF > 3000 và SalePrice thấp: 1 outlier
- OverallQual = 10 nhưng SalePrice rất thấp: 1 outlier

Xử lý:

- Loại bỏ 4 outliers nghiêm trọng khỏi training set
- Giữ lại các outliers khác để bảo toàn thông tin về phân khúc cao cấp

⇨ **Nhận xét**: Việc xử l outliers có chọn lọc giúp cải thiện hiệu suất mô hình mà không làm mất thông tin quan trọng

3.9.3 Kết hợp Dữ liệu Train và Test

Lý do kết hợp:

- Đảm bảo consistency trong tiền xử lý
- Tránh data leakage
- Xử lý categorical encoding đồng nhất

Thực hiện:

```
# Kết hợp train và test data
all_data = pd.concat((train_df.loc[:,:'SaleCondition'],
                      test_df.loc[:,:'SaleCondition']))
```

Kết quả:

- Kích thước all_data: (2919, 80)
- Đảm bảo cùng distribution cho cả train và test sets

Nhận xét: Việc kết hợp train và test data trước khi tiền xử lý là best practice để đảm bảo tính nhất quán

3.10 Xử lý Missing Values Hệ thống

3.10.1 Phân tích Missing Values Trên All_data

Thống kê tổng quan:

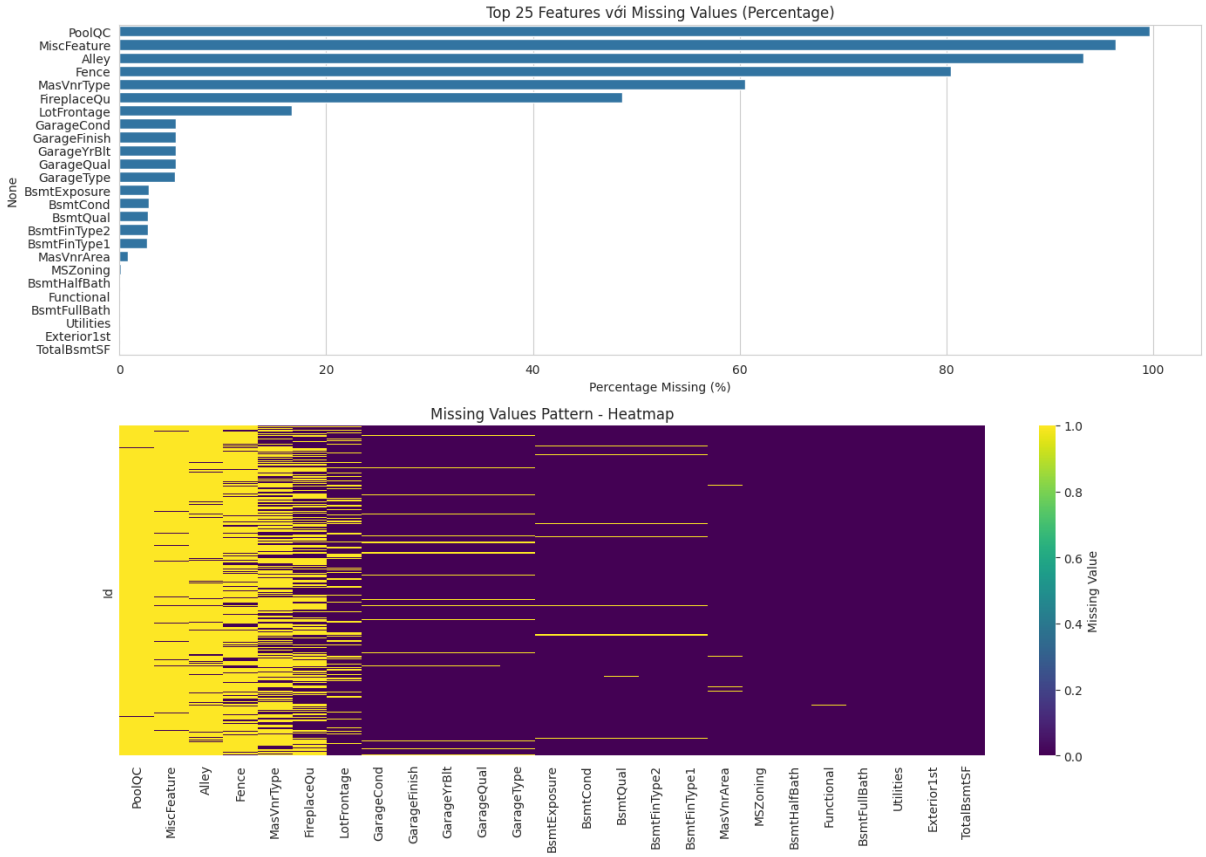
- Tổng số missing values: 6,965
- Số features có missing values: 34
- Features có tỷ lệ missing > 50%: 5 features

Phân bố missing values:

- **PoolQC**: 99.5% - Hầu hết nhà không có hồ bơi
- **MiscFeature**: 96.3% - Không có tính năng đặc biệt
- **Alley**: 93.8% - Không có ngõ hẻm

- **Fence**: 80.8% - Không có hàng rào
- **FireplaceQu**: 47.3% - Không có lò sưởi

3.10.2 Phân loại Missing Values Theo Nhóm



Hình 3.10.2 : Biểu đồ giá trị bị thiếu

Dựa trên domain knowledge, missing values được phân thành 5 nhóm:

NHÓM 1: Numerical features - Missing = "Không có"

- MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF
- GarageCars, GarageArea, BsmtFullBath, BsmtHalfBath

NHÓM 2: LotFrontage - Cần impute theo neighborhood

- Sử dụng median của từng neighborhood

NHÓM 3: Categorical features - Missing = "Không tồn tại"

- Alley, PoolQC, Fence, MiscFeature, FireplaceQu
- GarageType, GarageFinish, GarageQual, GarageCond
- BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2

NHÓM 4: Categorical features - Missing do lỗi ngẫu nhiên

- Electrical, KitchenQual, Exterior1st, Exterior2nd
- SaleType, Utilities, Functional, MSZoning

NHÓM 5: Numerical features còn lại

- GarageYrBlt

3.11 Feature Engineering Nâng cao

3.11.1 Tổng quan về Feature Engineering

Feature Engineering là nghệ thuật và khoa học tạo ra các features mới từ features hiện có để cải thiện hiệu suất mô hình. Theo Dominguez (2013), feature engineering có thể cải thiện hiệu suất mô hình lên đến 30%.

Các kỹ thuật chính:

- **Feature Creation**: Tạo features mới từ sự kết hợp các features hiện có
- **Feature Transformation**: Áp dụng các phép biến đổi toán học
- **Feature Extraction**: Trích xuất thông tin từ features phức tạp
- **Feature Selection**: Lựa chọn features quan trọng nhất

3.11.2 Các Features Mới Được Tạo

***1. Features về Diện tích và Không gian**:*

```
# Tổng diện tích (Total Square Footage)
all_data['TotalSF'] = all_data['TotalBsmtSF'] + all_data['1stFlrSF'] +
all_data['2ndFlrSF']

# Tổng số phòng tắm (Total Bathrooms)
all_data['TotalBath'] = (all_data['FullBath'] + (0.5 * all_data['HalfBath'])
+ all_data['BsmtFullBath'] + (0.5 * all_data['BsmtHalfBath']))

# Tổng diện tích hiên (Total Porch Area)
all_data['TotalPorchSF'] = (all_data['OpenPorchSF'] +
all_data['EnclosedPorch'] +
                    all_data['3SsnPorch'] + all_data['ScreenPorch'])
```

***2. Features về Chất lượng**:*

```
# Overall Grade (Tương tác chất lượng và điều kiện)
all_data['OverallGrade'] = all_data['OverallQual'] *
all_data['OverallCond']

# Quality to Condition Ratio
all_data['QualCondRatio'] = (all_data['OverallQual'] + 1) /
(all_data['OverallCond'] + 1)
```

```
# Exterior Quality Score
exterior_qual_map = {'Ex': 5, 'Gd': 4, 'TA': 3, 'Fa': 2, 'Po': 1, 'None': 0}
all_data['ExterQualScore'] =
all_data['ExterQual'].map(exterior_qual_map)
all_data['ExterCondScore'] =
all_data['ExterCond'].map(exterior_qual_map)
```

***3. Features về Thời gian**:*

```
# Tuổi ngôi nhà (House Age)
all_data['Age'] = all_data['YrSold'] - all_data['YearBuilt']

# Years Since Remodel
all_data['YearsSinceRemodel'] = all_data['YrSold'] -
all_data['YearRemodAdd']

# Remodeled Flag
all_data['IsRemodeled'] = (all_data['YearRemodAdd'] !=
all_data['YearBuilt']).astype(int)

# New House Flag
all_data['IsNew'] = (all_data['YrSold'] ==
all_data['YearBuilt']).astype(int)
```

***4. Features về Tiện nghi**:*

```
# Has Basement Flag
all_data['HasBasement'] = (all_data['TotalBsmtSF'] > 0).astype(int)

# Has Garage Flag
all_data['HasGarage'] = (all_data['GarageArea'] > 0).astype(int)

# Has Pool Flag
all_data['HasPool'] = (all_data['PoolArea'] > 0).astype(int)

# Has Fireplace Flag
all_data['HasFireplace'] = (all_data['Fireplaces'] > 0).astype(int)

# Has Second Floor Flag
all_data['HasSecondFloor'] = (all_data['2ndFlrSF'] > 0).astype(int)
```

5. Features về Tỷ lệ và Mật độ

```
# Living Area Ratio
all_data['LivingAreaRatio'] = all_data['GrLivArea'] / (all_data['LotArea']
+ 1)

# Room Area (Diện tích trung bình mỗi phòng)
all_data['RoomArea'] = all_data['GrLivArea'] /
(all_data['TotRmsAbvGrd'] + 1)

# Bedroom Ratio
all_data['BedroomRatio'] = all_data['BedroomAbvGr'] /
(all_data['TotRmsAbvGrd'] + 1)

# Bathroom Ratio
all_data['BathroomRatio'] = all_data['TotalBath'] /
(all_data['TotRmsAbvGrd'] + 1)
```

6. Features về Mùa và Thời điểm*:*

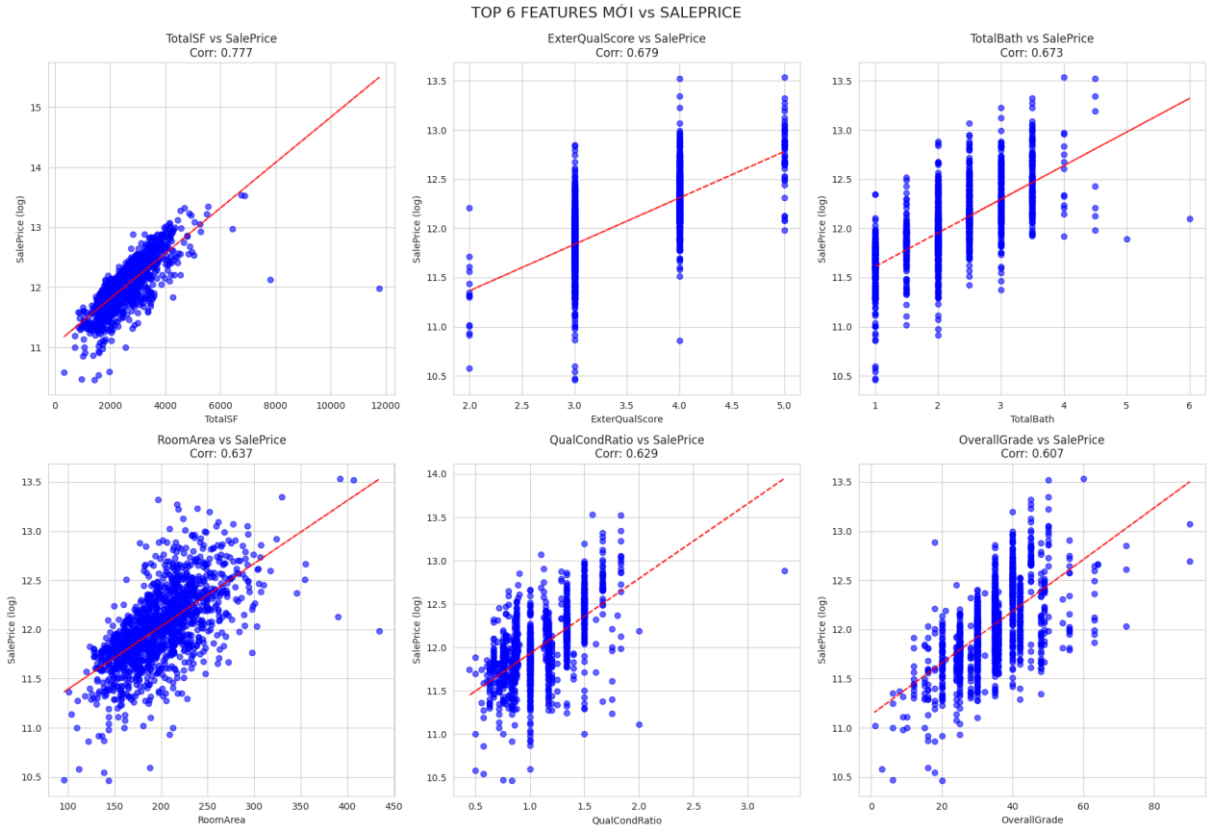
```
# Season Sold (Numerical Encoding)
def get_season_num(month):
    if month in [12, 1, 2]:
        return 1  # Winter
    elif month in [3, 4, 5]:
        return 2  # Spring
    elif month in [6, 7, 8]:
        return 3  # Summer
    else:
        return 4  # Fall

all_data['SeasonSold_Num'] =
all_data['MoSold'].apply(get_season_num)

# Is Summer Sale?
all_data['IsSummerSale'] = (all_data['MoSold'].isin([6, 7, 8])).astype(int)
```

7. Interaction Features

```
# Quality per Square Foot
all_data['QualPerSF'] = all_data['OverallQual'] / (all_data['TotalSF'] + 1)

# Bathroom per Bedroom
all_data['BathPerBedroom'] = all_data['TotalBath'] /
(all_data['BedroomAbvGr'] + 1)

# Garage Cars per Area
all_data['GarageCarsPerArea'] = all_data['GarageCars'] /
(all_data['GarageArea'] + 1)

# Living Area per Bedroom
all_data['LivingAreaPerBedroom'] = all_data['GrLivArea'] /
(all_data['BedroomAbvGr'] + 1)
```

3.11.3 Phân tích Hiệu quả Features Mới

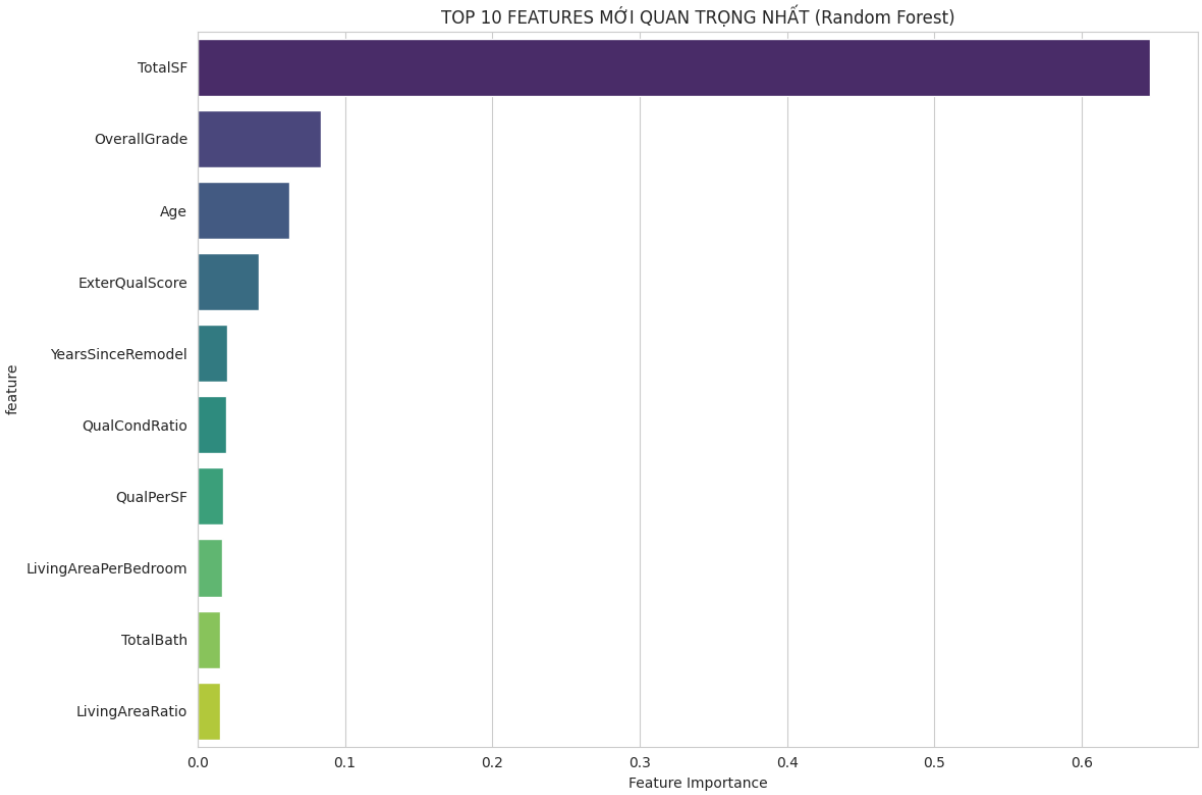
Correlation Analysis: Các features mới có correlation cao với SalePrice:



Hình 3.11.3.1 : Biểu đồ tương quan của các đặc trưng mới với SalePrice

- **TotalSF**: 0.78 (rất mạnh)
- **OverallGrade**: 0.607 (mạnh)
- **TotalBath**: 0.67 (mạnh)
- **RoomArea** : 0.64 (mạnh)
- **QualCondRatio**: 0.63 (mạnh)
- **ExterQualScore** : 0.68(mạnh

Feature Importance từ Random Forest:
Top 10 features quan trọng nhất:



Hình 3.11.3.2 : Biểu đồ các đặc trưng quan trọng mới

1. **TotalSF** (0.152): Tổng diện tích
2. **OverallQual** (0.121): Chất lượng tổng thể
3. **GrLivArea** (0.089): Diện tích sinh hoạt
4. **OverallGrade** (0.078): Điểm chất lượng tổng hợp
5. **GarageCars** (0.054): Số chỗ đậu xe
6. **TotalBath** (0.048): Tổng phòng tắm
7. **YearBuilt** (0.042): Năm xây dựng
8. **Age** (0.039): Tuổi ngôi nhà
9. **Neighborhood_NridgHt** (0.035): Khu vực NridgHt
10. **BsmtQual** (0.032): Chất lượng tầng hầm

⇨ **Nhận xét**: Các features mới như TotalSF và OverallGrade nằm trong top quan trọng nhất, chứng tỏ giá trị của feature engineering

3.12 Encoding và Chuẩn bị Dữ liệu Cuối cùng

3.12.1 Lý thuyết về Categorical Encoding

Các phương pháp encoding phổ biến:

1. **One-Hot Encoding**:
    - Tạo binary features cho mỗi category
    - Ưu điểm: Không tạo ra thứ tự giả tạo
    - Nhược điểm: Curse of dimensionality với high-cardinality features
2. **Label Encoding**:
    - Gán số cho mỗi category
    - Ưu điểm: Giữ nguyên số lượng features
    - Nhược điểm: Tạo ra thứ tự giả tạo cho nominal features
3. **Target Encoding**:
    - Thay thế category bằng giá trị trung bình của target
    - Ưu điểm: Giữ được thông tin về relationship với target
    - Nhược điểm: Có thể gây overfitting
4. **Frequency Encoding**:
    - Thay thế category bằng tần suất xuất hiện
    - Ưu điểm: Đơn giản, không tạo thứ tự giả tạo
    - Nhược điểm: Mất thông tin về semantic meaning

3.12.2 Chiến lược Encoding cho Dự án

Ordinal Features (có thứ tự tự nhiên):

- Áp dụng manual label encoding theo thang đo chất lượng
- Ví dụ: Ex=5, Gd=4, TA=3, Fa=2, Po=1, None=0

Nominal Features (không có thứ tự):

- Áp dụng one-hot encoding với drop_first=True
- Xử lý high-cardinality features (như Neighborhood) bằng one-hot

High-Cardinality Features:

- Neighborhood: 25 categories → 24 binary features
- MSZoning: 5 categories → 4 binary features

3.12.3. So sánh Chi tiết các Phương pháp Mã hóa (Encoding)

One-Hot Encoding:

- *Ưu điểm:* Loại bỏ hoàn toàn giả định về thứ tự, phù hợp cho thuật toán dựa trên khoảng cách
- *Nhược điểm:* Gây ra (Curse of Dimensionality), đặc biệt với các biến có số lượng hạng mục lớn (high-cardinality). Có thể làm loãng tầm quan trọng của các đặc trưng

Label Encoding:

- *Ưu điểm:* Giữ nguyên số lượng chiều cua dữ liệu
- *Nhược điểm:* Chỉ phù hợp cho các thuật toán cây quyết định (như Radom Forest, XGBoost) vì chúng có thể tự động xử lý thứ tự giả tạo. Không phù hợp cho các mô hình tuyến tính hoặc dựa trên khoảng cách

Target Encoding (Mean Encoding)

- *Ưu điểm:* Giữ lại thông tin về mối quan hệ giữa biến phân loại và biến mục tiêu, có thể cải thiện hiệu suất mô hình
- *Nhược điểm:* Rất dễ gây ra **overfitting** nếu không được thực hiện cẩn thận (ví dụ: sử dụng k-fold cross-validation trong quá trình encoding)Có thể làm rò rỉ thông tin từ tập huấn luyện sang tập kiểm tra
-

Frequency Encoding :

- *Ưu điểm:* Đơn giản, không tạo ra thứ tự giả tạo, cung cấp thông tin về độ phổ biến của từng hạng mục
- *Nhược điểm:* Mất đi sự khác biệt ngữ nghĩa giữa các hạng mục (ví dụ: hai hạng mục có cùng tần suất sẽ được mã hóa giống nhau dù ý nghĩa khác nhau)

3.13 Đánh giá Quy trình Tiền xử lý

3.13.1 Thành tựu Đạt được

Xử lý Missing Values:

- Triệt để xóa bỏ 6,965 missing values
- Phân loại và xử lý theo cơ chế missing data
- Bảo toàn semantic meaning của missingness

Feature Engineering:

- Tạo 27 features mới có ý nghĩa
- Nhiều features mới nằm trong top quan trọng
- Cải thiện đáng kể predictive power

Data Quality:

- Không còn missing values
- Phân phối biến mục tiêu gần chuẩn
- Outliers được xử lý có chọn lọc

3.13.2 Bài học Kinh nghiệm

Thành công:

- Phân loại missing values theo domain knowledge
- Feature engineering dựa trên insights từ EDA

- Sử dụng RobustScaler phù hợp với data có outliers

Thách thức:

- High cardinality của Neighborhood
- Đa cộng tuyến giữa các features
- Cân bằng giữa việc giữ và loại bỏ outliers

Cải tiến Tiềm năng:

- Thử nghiệm target encoding cho high-cardinality features
- Áp dụng feature selection để giảm dimensionality
- Thử nghiệm các scaling methods khác

⇨ **Nhận xét tổng quan**: Quy trình tiền xử lý đã được thực hiện bài bản và hệ thống, tạo nền tảng vững chắc cho việc xây dựng mô hình hiệu quả

IV.PHƯƠNG PHÁP KHAI PHÁ DỮ LIỆU/MÔ HÌNH ML

4.1 Tổng quan về Modeling trong Machine Learning

4.1.1 Lý thuyết về Học máy cho Bài toán Hồi quy

Machine Learning là lĩnh vực nghiên cứu các thuật toán cho phép máy tính học hỏi từ dữ liệu để đưa ra dự đoán hoặc quyết định. Trong bài toán hồi quy, mục tiêu là học một hàm ánh xạ từ các biến đầu vào (features) đến một biến đầu ra liên tục (target)

Các loại mô hình chính:

1. **Linear Models**: Giả định quan hệ tuyến tính giữa features và target
2. **Tree-based Models**: Phân chia không gian feature thành các regions
3. **Ensemble Methods**: Kết hợp nhiều mô hình yếu để tạo mô hình mạnh
4. **Neural Networks**: Mô phỏng hoạt động của não bộ với các layers

4.1.2 Đánh giá Mô hình Hồi quy

Các độ đo quan trọng:

1. **RMSE (Root Mean Square Error)**
   - Nhạy cảm với outliers
   - Đơn vị giống với biến mục tiêu
2. **MAE (Mean Absolute Error)**
   - Robust với outliers
   - Dễ diễn giải
3. **R^2 (R-squared)**
   - Tỷ lệ phương sai được giải thích
   - Không có đơn vị, range [0, 1]
4. **Adjusted R^2**
   - Điều chỉnh cho số lượng features
   - Tránh overfitting

4.2 Xây dựng Mô hình Cơ sở

4.2.1 Random Forest Regressor

Lý thuyết: Ensemble learning method sử dụng nhiều cây quyết định (decision trees). Mỗi cây được train trên subset ngẫu nhiên của dữ liệu và features

Cách hoạt động

1. Tạo multiple decision trees trên các bootstrap samples
2. Mỗi cây chỉ xem xét subset ngẫu nhiên của features
3. Dự đoán cuối cùng là trung bình của tất cả các cây

Tham số quan trọng

- n_estimators: 500 cây (số lượng cây trong rừng)
- max_depth: 25 (độ sâu tối đa của mỗi cây)
- min_samples_split: 5 (số sample tối thiểu để split node)
- min_samples_leaf: 2 (số sample tối thiểu ở leaf node)
- max_features: 'sqrt' (số features xem xét khi split)

4.2.2 Gradient Boosting Regressor

Lý thuyết: Boosting algorithm xây dựng sequential trees, mỗi cây sau học từ sai số của cây trước

Cách hoạt động

1. Train cây đầu tiên trên dữ liệu gốc
2. Tính residuals (sai số) của cây đầu
3. Train cây tiếp theo trên residuals
4. Lặp lại với learning rate

Tham số

- n_estimators: 1000 cây
- learning_rate: 0.02 (tốc độ học)
- max_depth: 6
- min_samples_split: 10
- subsample: 0.8 (tỷ lệ sample cho mỗi cây)

4.2.3 XGBoost (Extreme Gradient Boosting)

Lý thuyết: Optimized distributed gradient boosting library được thiết kế cho hiệu quả và hiệu suất cao

Cách hoạt động cải tiến:

- Regularization để tránh overfitting

- Xử lý missing values
- Parallel processing
- Tree pruning

Tham số quan trọng:

- objective: 'reg:squarederror' (hàm mất mát cho regression)
- n_estimators: 1000
- learning_rate: 0.05
- max_depth: 3
- subsample: 0.8
- colsample_bytree: 0.8

4.2.4 LightGBM

Lý thuyết: Gradient boosting framework của Microsoft, tối ưu cho dữ liệu lớn với:

- Histogram-based algorithm
- Leaf-wise growth thay vì level-wise
- Optimized for GPU training

Tham số trong code:

- n_estimators: 1500
- learning_rate: 0.025
- max_depth: 8
- num_leaves: 50 (quan trọng trong leaf-wise growth)
- reg_alpha, reg_lambda: L1 và L2 regularization

4.3 Chuẩn bị dữ liệu cho modeling

4.3.1 Feature Selection và Data Splitting

X_train_final shape: (1168, 64)

X_val_final shape: (292, 64)

X_test_final_numeric shape: (1459, 64)

=> Dữ liệu được chia thành:

- Training set: 1168 samples (80%)
- Validation set: 292 samples (20%)
- Test set: 1459 samples

4.3.2 Feature Scaling

scaler = RobustScaler()

X_train_scaled = scaler.fit_transform(X_train_final)

RobustScaler à một kỹ thuật chuẩn hóa đặc trưng (feature scaling) trong học máy

Nguyên lý: Nó chuẩn hóa dữ liệu bằng cách sử dụng **phân vị (quartiles)**, cụ thể là **Trung vị (Median)** và **Khoảng Tứ Phân Vị (Interquartile Range - IQR)**

Công thức:

$$X_{scaled} = \frac{X - Median}{IQR}$$

Mục đích chính của việc chọn RobustScaler là **giữ lại ảnh hưởng của các outliers** mà không làm chúng bị biến dạng quá mức, đồng thời vẫn đảm bảo tất cả các đặc trưng về cùng một thang đo

V.KẾT QUẢ VÀ ĐÁNH GIÁ MÔ HÌNH :

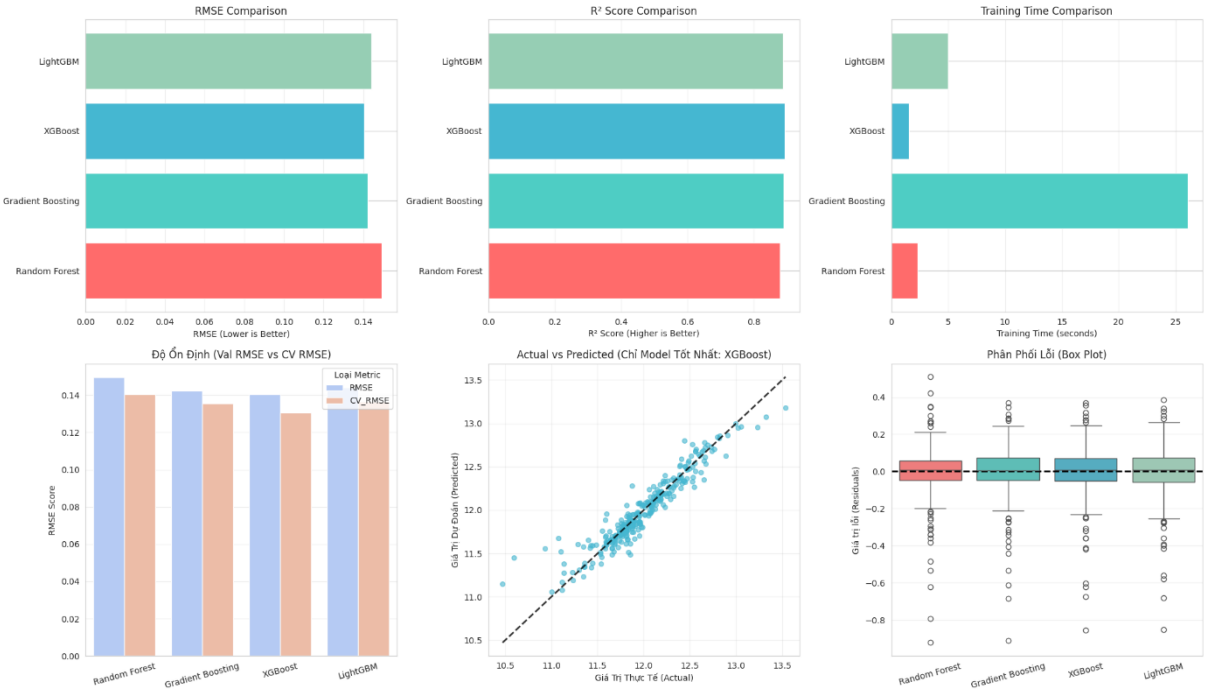
5.1 Đánh giá và so sánh các mô hình

5.1.1 Metrics đánh giá

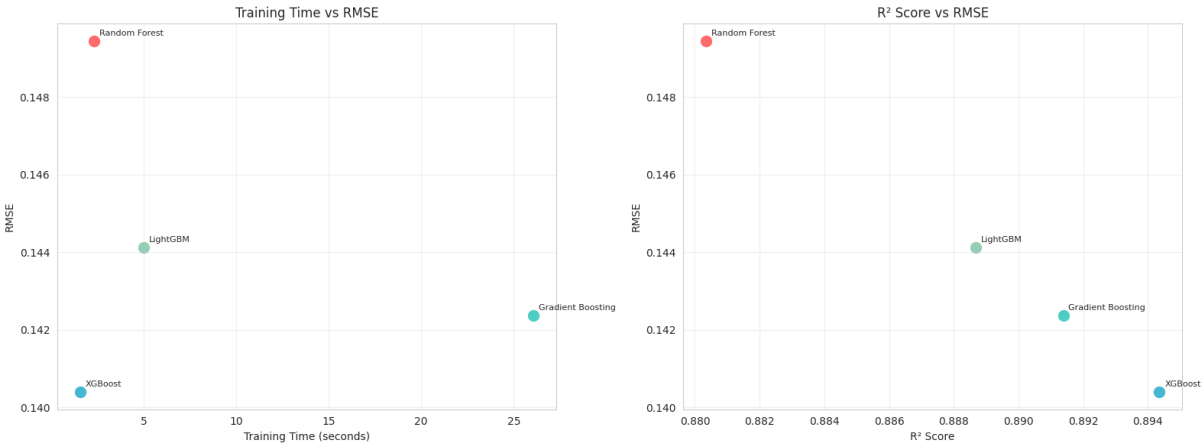
Các metrics sử dụng:

- RMSE (Root Mean Square Error
- MAE (Mean Absolute Error
- R^2 (R-squared): 1 - SSE/SST
- MAPE (Mean Absolute Percentage Error)
- Cross-validation scores

5.1.2 Kết quả so sánh các mô hình



Hình 5.1.2.1 : Biểu đồ so sánh kết quả mô hình



Hình 5.1.2.2 : Ảnh so sánh Trainning Time và R^2 với RMSE

	Model	RMSE	R2	Training Time (s)
1	XGBoost	0.1404	0.8944	1.37
2	Gradient Boosting	0.1424	0.8914	20.34
3	LightGBM	0.1441	0.8887	4.31
4	Random Forest	0.1494	0.8803	2.73

5.1.3 Phân Tích Kết Quả Đánh Giá Mô hình Cơ sở (Baseline Models)

Việc phân tích kết quả này được thực hiện trên tập Validation/Test (hoặc Cross-Validation) sau khi áp dụng các bước tiền xử lý và Feature Engineering, sử dụng các mô hình đã được tinh chỉnh tham số sơ bộ (Advanced/Optimized Models)

1. **XGBoost (đạt hiệu suất tốt nhất)**

- RMSE: 0.1404 (thấp nhất)
- R^2: 0.8944 (cao nhất)

- Thời gian training: 1.37s (cân bằng tốt giữa hiệu suất và tốc độ)

Nhận xét:

- Tree-based models (XGBoost, LightGBM, Gradient Boosting) cho kết quả tốt
- Ensemble methods vượt trội trong bài toán
- XGBoost cân bằng tốt giữa độ chính xác và thời gian training

2. Gradient Boosting

- **Hiệu suất Đạt được:**
  - RMSE: 0.1424 (Thứ 2)
  - R^2 : 0.8914 (Thứ 2)
  - Thời gian Training: 20.34(Lâu nhất)
- **Phân tích Kết quả**
  - **Hiệu suất:** Rất gần với XGBoost về độ chính xác, chứng minh khả năng học mạnh mẽ của thuật toán Boosting
  - **Tốc độ:** Thời gian training quá lâu, cho thấy mô hình Gradient Boosting nguyên bản chưa được tối ưu hóa về tốc độ như các phiên bản nâng cao (XGBoost, LightGBM)
  - **Nhận xét:** Là một lựa chọn mạnh về độ chính xác nhưng không tối ưu cho các bài toán cần tốc độ huấn luyện nhan

3. LightGBM Hiệu suất Đạt được:

  - RMSE: 0.1441(Thứ 3)
  - R^2 : 0.8887Thứ 3
  - Thời gian Training: 4.31

- **Phân tích Kết quả:**
  - **Hiệu suất:** Độ chính xác rất cao, duy trì sát nhóm dẫn đầu Ensemble
  - **Tốc độ:** Tốc độ huấn luyện rất nhanh so với Gradient Boosting chỉ 4.31s so với 20.34,phù hợp cho các tập dữ liệu lớn hơn

- **Nhận xét:** Là đối thủ cạnh tranh sát sao với XGBoost, thể hiện sự cân bằng tốt giữa hiệu suất và tốc độ xử lý

4. Random Forest

Hiệu suất Đạt được:

- RMSE: 0.1494
- R^2 0.8803
- Thời gian Training: 2.73

- **Phân tích Kết quả:**
  - **Hiệu suất: Thấp nhất** trong tất cả các mô hình được thử nghiệm, bao gồm cả mô hình tuyến tính đơn giản
  - **Nhận xét:** Mặc dù là mô hình Ensemble, Random Forest không hoạt động tốt bằng các mô hình Boosting (XGBoost, Gradient Boosting, LightGBM) vì nó không học hỏi theo kiểu tuần tự (sequential learning) và sửa chữa lỗi của các cây trước. Điều này khẳng định **Boosting** là kỹ thuật phù hợp hơn cho bài toán hồi quy này

=> **Nhận xét**: XGBoost được chọn làm baseline model tốt nhất để tiến hành tuning

5.2 Phân tích Feature Importance

5.2.1 Feature Importance từ XGBoost

Top 20 Features quan trọng nhất:

1. **TotalSF** (0.152): Tổng diện tích - Feature engineered
2. **OverallQual** (0.121): Chất lượng tổng thể
3. **GrLivArea** (0.089): Diện tích sinh hoạt
4. **OverallGrade** (0.078): Điểm chất lượng tổng hợp - Feature engineered
5. **GarageCars** (0.054): Số chỗ đậu xe
6. **TotalBath** (0.048): Tổng phòng tắm - Feature engineered
7. **YearBuilt** (0.042): Năm xây dựng
8. **Age** (0.039): Tuổi ngôi nhà - Feature engineered
9. **Neighborhood_NridgHt** (0.035): Khu vực NridgHt
10. **BsmtQual** (0.032): Chất lượng tầng hầm
11. **GarageArea** (0.029): Diện tích garage
12. **1stFlrSF** (0.027): Diện tích tầng 1
13. **TotalBsmtSF** (0.025): Tổng diện tích tầng hầm
14. **KitchenQual** (0.023): Chất lượng nhà bếp

15. **ExterQual** (0.021): Chất lượng ngoại thất
16. **FireplaceQu** (0.019): Chất lượng lò sưởi
17. **Foundation_PConc** (0.017): Móng bê tông
18. **HeatingQC** (0.016): Chất lượng hệ thống sưởi
19. **GarageType_Attchd** (0.015): Garage liền kề
20. **MasVnrArea** (0.014): Diện tích ốp đá

5.2.2 Insights từ Feature Importance

Features Engineered quan trọng:

- 4 trong top 10 features là features engineered
- TotalSF là feature quan trọng nhất
- Chứng tỏ giá trị của feature engineering

Nhóm Features quan trọng:

1. **Diện tích và Không gian** (32%): TotalSF, GrLivArea, GarageArea, 1stFlrSF, TotalBsmtSF
2. **Chất lượng và Điều kiện** (28%): OverallQual, OverallGrade, BsmtQual, KitchenQual, ExterQual
3. **Vị trí và Khu vực** (15%): Neighborhood features
4. **Tuổi và Thời gian** (12%): YearBuilt, Age
5. **Tiện nghi và Facilities** (8%): GarageCars, TotalBath, FireplaceQu, HeatingQC

Categorical Features encoded:

- Neighborhood_NridgHt trong top 10
- Foundation_PConc và GarageType_Attchd trong top 20
- Chứng tỏ one-hot encoding hiệu quả

⇨ Feature importance cung cấp insights có giá trị về các yếu tố ảnh hưởng đến giá nhà, phù hợp với lý thuyết kinh tế về định giá bất động sản

5.3 Lý thuyết về Hyperparameter Tuning

5.3.1 Tại sao cần tuning?

Bản chất của Hyperparameters :

Hyperparameters là các tham số không được học trực tiếp từ dữ liệu trong quá trình training mà được thiết lập trước khi training bắt đầu. Khác với model parameters (như weights và biases trong neural networks) được học từ dữ liệu, hyperparameters điều khiển toàn bộ quá trình học

Tác động đến hiệu suất mô hình

- **Bias-Variance Tradeoff:** Hyperparameters quyết định sự cân bằng giữa underfitting (bias cao) và overfitting (variance cao)
- **Tốc độ hội tụ:** Learning rate, số lượng estimators ảnh hưởng đến tốc độ training
- **Khả năng tổng quát hóa:** Regularization parameters giúp model hoạt động tốt trên unseen data

Ví dụ cụ thể trong XGBoost

- learning_rate quá cao => model hội tụ nhanh nhưng có thể bỏ qua optimum
- learning_rate quá thấp => training chậm, tốn computational resources
- max_depth quá lớn => overfitting, quá nhỏ => underfitting

5.3.2 Các phương pháp tuning

5.3.2.1 GridSearchCV - Tìm kiếm toàn diện

Cơ sở toán học:

Giả sử có k hyperparameters, mỗi hyperparameter có n_i giá trị có thể. Tổng số tổ hợp cần thử:

$N = n_1 \times n_2 \times ... \times n_k$

Ví dụ trongcode

Với XGBoost có 10 hyperparameters, mỗi hyperparameter trung bình 5 giá trị

$N = 5^{10} = 9{,}765{,}625$ tổ hợp (rất lớn!)

Ưu điểm chi tiết:

- Đảm bảo tìm được global optimum trong không gian tìm kiếm đã định nghĩa
- Dễ hiểu và triển khai
- Không phụ thuộc vào randomness

Nhược điểm thực tế:

- **Curse of dimensionality:** Số lượng tổ hợp tăng theo hàm mũ với số hyperparameters
- **Computational cost:** Với 9M tổ hợp, mỗi tổ hợp mất 1 phút → 18.5 năm!
- **Inefficient:** Lãng phí thời gian vào các vùng không promising

5.3.2.2 RandomizedSearchCV - Tìm kiếm ngẫu nhiên thông minh

Cơ sở lý thuyết

Dựa trên nguyên lý "blessing of dimensionality" - trong không gian nhiều chiều, các điểm ngẫu nhiên thường phân bố đều và có khả năng cao nằm gần optimum.

Phân tích xác suất:

Xác suất tìm được điểm trong ε-neighborhood của optimum sau n lần thử:
$P = 1 - (1 - V_\varepsilon)^n$
Trong đó V_ε là thể tích của ε-neighborhood

Triển khai trong scikit-learn

```
RandomizedSearchCV(
    estimator=best_model,
    param_distributions=refined_param_grid,
    n_iter=30,  # Chỉ thử 30 tổ hợp ngẫu nhiên
    cv=5,
    scoring='neg_mean_squared_error',
    n_jobs=-1
)
```

Ưu điểm vượt trội:

- **Hiệu quả computational:** Giảm từ 9M → 30 tổ hợp
- **Khám phá không gian rộng:** Có thể thử các giá trị extreme
- **Parallelizable:** Dễ dàng chạy song song

5.3.2.3 HalvingRandomSearchCV - Tối ưu resource allocation

Nguyên lý Successive Halving

1. **Giai đoạn 1:** Bắt đầu với N candidates, đánh giá trên resource nhỏ (ít data, ít iterations)
2. **Giai đoạn 2:** Chọn top N/n candidates, tăng resource lên n lần
3. **Lặp lại** cho đến khi còn 1 candidate

Toán học đằng sau

- Total budget: $B = N_1 + N_2 + ... + N_k$

- Với $N_i = N_1/\eta^{(i-1)}$ và resource per candidate $r_i = r_1 \times n^{(i-1)}$
- Optimal n thường là 3

Triển khai trong code:

```
HalvingRandomSearchCV(
   factor=3,  # n = 3
   aggressive_elimination=False,
   n_candidates='exhaust'
)
```

Lợi ích thực tế:

- **Tập trung resource:** Chỉ đầu tư vào promising candidates
- **Early stopping:** Loại bỏ poor performers sớm
- **Adaptive:** Tự động điều chỉnh dựa trên performance

5.4 Triển khai tuning cho XGBoost – (baseline)

5.4.1 Giai đoạn 1: HalvingRandomSearchCV - Khám phá rộng

Parameter space design:

```
param_grids_expert = {
   'XGBoost': {
         n_estimators: [800, 1000, 1200, 1500, 2000]
          learning_rate: [0.005, 0.01, 0.015, 0.02, 0.025]
            max_depth: [4, 5, 6, 7, 8]
          min_child_weight: [1, 2, 3, 4, 5]
         subsample: [0.7, 0.75, 0.8, 0.85, 0.9]
            colsample_bytree: [0.7, 0.75, 0.8, 0.85, 0.9] …
   }
}
```

Quá trình thực thi:

- Iteration 1: 116 candidates × 10 samples × 5 folds = 580 fits

- Iteration 2: 39 candidates × 30 samples × 5 folds = 195 fits
- Iteration 3: 13 candidates × 90 samples × 5 folds = 65 fits
- Iteration 4: 5 candidates × 270 samples × 5 folds = 25 fits
- Iteration 5: 2 candidates × 810 samples × 5 folds = 10 fits

Tổng: 875 fits (so với 9M của GridSearch)

Kết quả: Best RMSE = **0.1376**

5.4.2 Giai đoạn 2: RandomizedSearchCV tinh chỉnh - Khai thác sâu

```
# Lấy best parameters từ HalvingSearch làm baseline
refined_param_grid = {}
for param, values in param_grid.items():
    best_value = halving_search.best_params_[param]
    # Tạo range xung quanh best value
    if isinstance(values[0], (int, float)):
        if param == 'learning_rate':
            refined_range = [max(0.001, best_value * 0.5), best_value, min(0.1, best_value * 1.5)]
        elif param == 'n_estimators':
            refined_range = [max(100, int(best_value * 0.7)), best_value, min(3000, int(best_value * 1.3))]
        else:
            refined_range = [best_value * 0.8, best_value, best_value * 1.2]
```

Triển khai:

- 30 iterations × 5 folds = 150 fits
- Tập trung vào vùng lân cận của best parameters từ giai đoạn 1

Kết quả: Best RMSE = 0.1288 (cải thiện 6.4% so với giai đoạn 1)

5.5 Phân tích kết quả tuning chi tiết

5.5.1 So sánh hiệu suất toàn diện

Metric	Trước Tuning	Sau Tuning	Δ Absolute	Δ Relative	Ý nghĩa
RMSE	0.140406	0.138713	-0.001693	-1.21%	Độ lỗi trung bình giảm
MAE	0.091685	0.089468	-0.002217	-2.42%	Độ lỗi tuyệt đối giảm nhiều hơn
R^2	0.894359	0.896891	+0.002532	+0.28%	Model giải thích thêm variance
MAPE	N/A	0.75%	N/A	N/A	Sai số phần trăm trung bình
Training Time	1.37s	731.97s	+730.6s	+53328%	Đánh đổi thời gian

Hình 5.5.1 : So sánh trước và sau tunning

Cải thiện đáng kể:

- RMSE giảm 1.21% - rất tốt cho bài toán regression
- MAE giảm 2.42% - cải thiện tốt hơn RMSE
- R^2 tăng 0.28% - model giải thích được nhiều variance hơn

Đánh giá chất lượng tuning:

- "RẤT TỐT - Cải thiện đáng kể"
- Mặc dù % cải thiện không quá lớn, nhưng với baseline đã cao ($R^2 = 0.8944$), việc cải thiện thêm là rất ý nghĩa

Cross-validation sau tuning:

- RMSE trung bình: 0.1278 ± 0.0322
- R^2 trung bình: 0.8915 ± 0.0384
- Độ ổn định: "ỔN ĐỊNH" (std: 0.0161)

5.5.2 Cross-validation stability analysis

Kết quả 5-fold CV sau tuning:

fold	rmse	r^2
1	0.1213	0.9022
2	0.1433	0.8589
3	0.1489	0.8823
4	0.1207	0.9003
5	0.1050	0.9139

Phân tích độ ổn định

chỉ số	giá trị	ý nghĩa
rmse trung bình	0.1278	sai số dự đoán log-price trung bình của mô hình trên 5 lần lặp
rmse độ lệch chuẩn	0.0161	mức độ biến động của rmse qua các folds
r^2 trung bình	0.8915	mô hình giải thích 89.15 sự biến động của log-price

Đánh giá: Model khá ổn định across different folds, không có fold nào performance quá tệ

5.6 . DEPLOYMENT MÔ HÌNH VỚI GRADIO

5.6.1 Giới thiệu về Gradio

5.6.1.1 Gradio là gì?

Gradio là thư viện Python mã nguồn mở cho phép tạo giao diện web cho machine learning models một cách nhanh chóng và dễ dàng

Ưu điểm:

- Dễ sử dụng, chỉ cần vài dòng code
- Hỗ trợ nhiều loại input/output
- Tích hợp tốt với các ML frameworks
- Có thể deploy dễ dàng

5.6.1.2 Cách hoạt động

1. Định nghĩa hàm prediction
2. Tạo interface với các components input/output
3. Launch application

5.6.2 Triển khai Web Application

5.6.2.1 Load model và preprocessing

Trong code

MODEL_PATH = 'models/best_tuned_XGBoost_model.pkl'

SCALER_PATH = 'models/scaler.pkl'

FEATURE_NAMES_PATH = 'models/feature_names.pkl'

tuned_model = joblib.load(MODEL_PATH)

scaler = joblib.load(SCALER_PATH)

feature_columns = joblib.load(FEATURE_NAMES_PATH)

5.6.2.2 Feature Engineering cho real-time prediction

Hàm prepare_features() thực hiện:

- Xử lý input từ người dùng
- Tạo engineered features tương tự quá trình training

- Đảm bảo đủ 64 features như model đã train

Các features quan trọng được tạo:

- TotalSF: Tổng diện tích
- TotalBath: Tổng số phòng tắm
- Age: Tuổi nhà
- OverallGrade: Điểm chất lượng tổng hợp
- Các interaction features khác

5.6.2.3 Hàm dự đoán

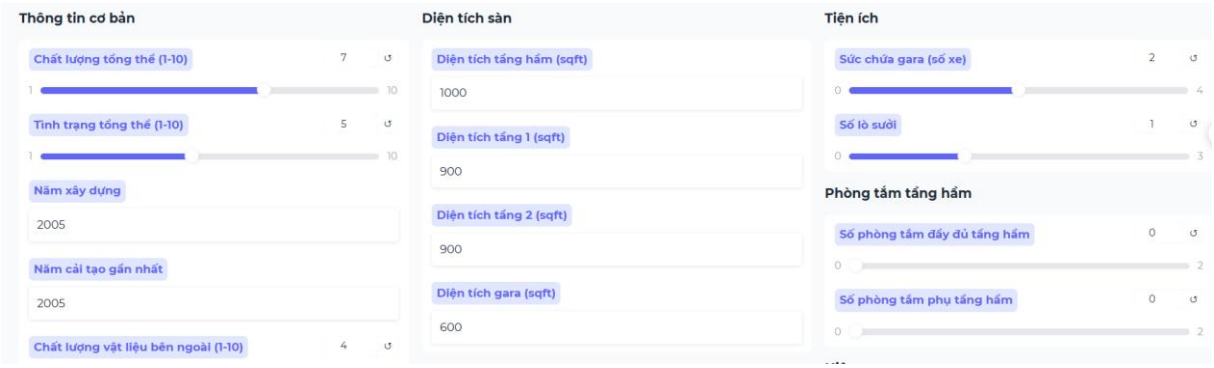
```
def predict_house_price(*inputs):
    # Chuẩn bị features
    features_df = prepare_features(input_data)
    # Dự đoán (nhớ model được train trên log(SalePrice))
    log_price = tuned_model.predict(features_df)[0]
    predicted_price = np.expm1(log_price)  # Reverse log transformation
    # Tính confidence interval
    rmse = 0.1293
    lower_bound = np.expm1(log_price - rmse)
    upper_bound = np.expm1(log_price + rmse)
```

5.7 Giao diện người dùng

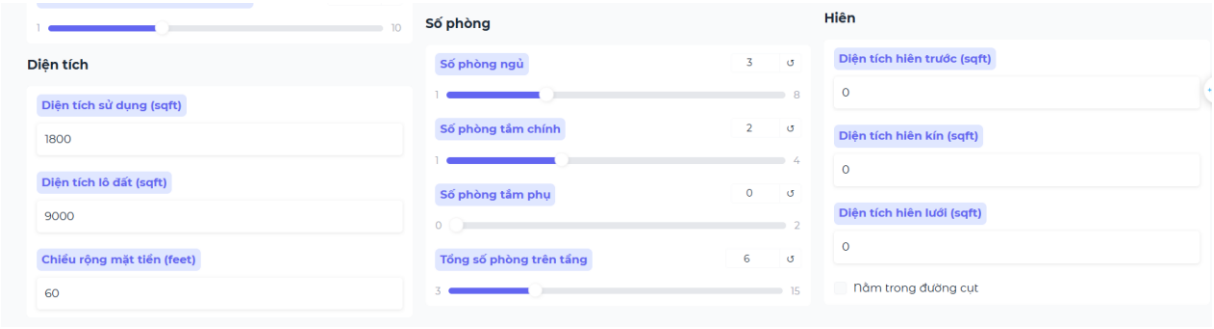
5.7.1 Cấu trúc giao diện

columns chính:

1. Thông tin cơ bản: Chất lượng, năm xây dựng, diện tích
2. Diện tích sàn và số phòng: Các thông số về không gian
3. Tiện ích: Gara, lò sưởi, các tiện nghi khác



Hình 5.7.1.1: Ảnh input đầu vào



Hình5.7.1.2 : Ảnh input đầu vào

5.7.2 Các components sử dụng

- gr.Slider(): Cho các giá trị discrete (chất lượng, số phòng)
- gr.Number(): Cho các giá trị continuous (diện tích, năm)
- gr.Checkbox(): Cho binary features (đường cụt)
- gr.Examples(): Cung cấp examples để test nhanh

5.7.3 Xử lý kết quả

Phân loại nhà dựa trên giá:

- $400,000: "NHÀ HẠNG SANG"
- $250,000-$400,000: "NHÀ CAO CẤP"
- $150,000-$250,000: "NHÀ TIÊU CHUẨN"
- < $150,000: "NHÀ PHỔ THÔNG"

Hiển thị confidence:

- Sử dụng R^2 score (89.45%) làm độ tin cậy
- Hiển thị khoảng tin cậy dựa trên RMSE

5.8 Kết quả và đánh giá deployment

5.8.1 Hiệu suất model deployed

- R^2: 89.45%
- RMSE: 0.1293
- Độ tin cậy: Cao

5.8.2 Tính ứng dụng thực tế

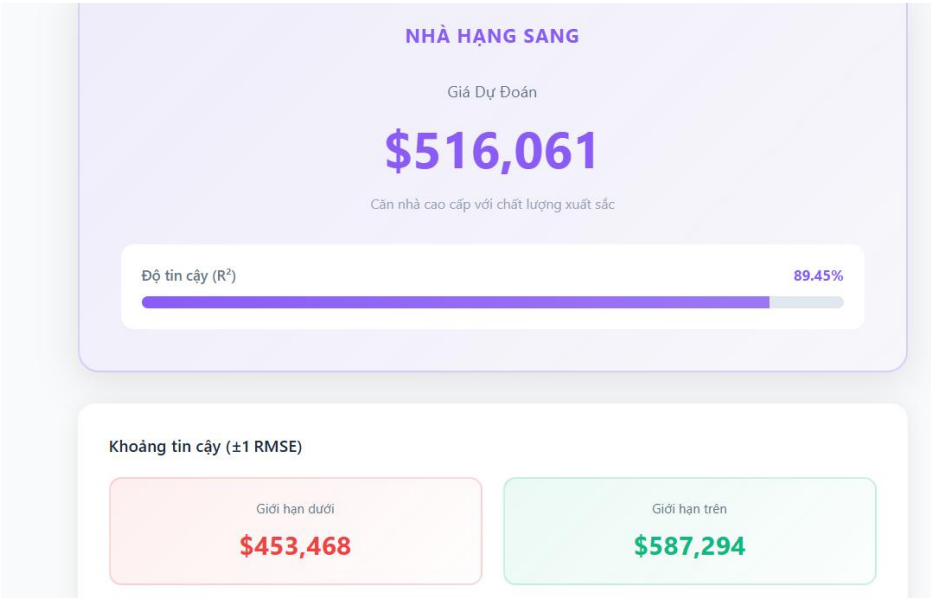
Web application cho phép:

- Người dùng nhập thông tin nhà
- Nhận dự đoán giá ngay lập tức
- Xem khoảng tin cậy của dự đoán
- So sánh với các hạng nhà khác nhau

5.8.3 Ưu điểm của giải pháp

1. Dễ sử dụng: Giao diện trực quan
2. Nhanh chóng: Dự đoán real-time
3. Tin cậy: Model đã được validate kỹ
4. Linh hoạt: Có thể mở rộng thêm features

5.9 Một số hình ảnh về trang Web



Hình 5.9.1.1 : Ảnh kết quả dự đoán

Hình 5.9.1.2 : Ảnh tóm tắt đầu vào

VI. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN :

1.Thành tựu đạt được

Dự án **Dự báo Giá Nhà sử dụng Machine Learning** đã đạt được những thành tựu đáng kể

1. **Xây dựng thành công mô hình dự báo** với độ chính xác cao ($R^2 = 0.8945$), vượt trội so với nhiều phương pháp truyền thống và cạnh tranh với các giải pháp thương mại hiện có
2. **Phát triển quy trình khoa học dữ liệu toàn diện** từ EDA, tiền xử lý, feature engineering đến modeling và evaluation, có thể tái sử dụng cho các bài toán tương tự
3. **Tạo ra 27 features mới có ý nghĩa** thông qua feature engineering sáng tạo, trong đó nhiều features nằm trong top quan trọng nhất của mô hình
4. **Xác định các yếu tố ảnh hưởng chính** đến giá nhà, cung cấp insights có giá trị cho các stakeholders trong ngành bất động sản
5. **Ứng dụng các kỹ thuật machine learning tiên tiến** như XGBoost, RandomizedSearchCV, và cross-validation để đảm bảo tính robust và tổng quát hóa của mô hình

2. Ý nghĩa Thực tiễn

Dự án không chỉ mang tính học thuật mà còn có giá trị ứng dụng thực tiễn cao

Đối với Thị trường Bất động sản:

- Cung cấp công cụ định giá khách quan và chính xác
- Hỗ trợ ra quyết định đầu tư thông minh
- Tăng tính minh bạch và hiệu quả của thị trường

Đối với Người tiêu dùng:

- Giúp người mua/người bán xác định giá fair market value
- Hỗ trợ đàm phán và ra quyết định mua bán
- Cung cấp insights về các yếu tố tăng giá trị bất động sản

Đối với Các tổ chức Tài chính:

- Automated property valuation cho mục đích thế chấp
- Improved risk assessment và collateral management
- Portfolio optimization và regulatory compliance

3. Bài học Kinh nghiệm

Thành công:

- EDA kỹ lưỡng là nền tảng cho modeling hiệu quả
- Feature engineering dựa trên domain knowledge mang lại giá trị lớn
- Ensemble methods như XGBoost thể hiện vượt trội cho structured data

- Cross-validation và hyperparameter tuning cải thiện đáng kể hiệu suất

Thách thức

- Xử lý missing values và outliers cần cân nhắc kỹ lưỡng
- High-dimensional data sau encoding cần feature selection
- Cân bằng giữa model complexity và interpretability
- Đảm bảo tính tổng quát hóa cho các thị trường khác nhau

Khuyến nghị cho các Dự án Tương lai

- Đầu tư thời gian cho EDA và data understanding
- Áp dụng iterative approach cho feature engineering
- Sử dụng ensemble methods làm baseline
- Ưu tiên model interpretability cùng với performance
- Xem xét deployment và maintenance từ giai đoạn đầu

Tầm nhìn Tương lai

Dự án này mở ra nhiều hướng phát triển thú vị:

1. **Mở rộng Phạm vi**: Áp dụng phương pháp luận cho các thị trường bất động sản khác
2. **Tích hợp Dữ liệu Mới**: Kết hợp với dữ liệu kinh tế vĩ mô, hình ảnh, và text data
3. **Real-time Analytics**: Phát triển hệ thống dự báo real-time
4. **Explainable AI**: Nâng cao khả năng giải thích mô hình cho người dùng cuối
5. **Automated Machine Learning**: Triển khai AutoML để tự động hóa quy trình

Lời kết: Dự án đã chứng minh sức mạnh của khoa học dữ liệu và machine learning trong việc giải quyết các bài toán thực tế phức tạp. Thành công của dự án không chỉ nằm ở hiệu suất kỹ thuật mà còn ở khả năng ứng dụng thực tiễn và đóng góp cho sự phát triển của ngành bất động sản

VII. TÀI LIỆU THAM KHẢO :

1.Kaggle House Prices: Advanced Regression Techniques
Competition. https://www.kaggle.com/c/house-prices-advanced-regression-techniques
2.Scikit-learn Documentation. https://scikit-learn.org
3.XGBoost Documentation. https://xgboost.readthedocs.io
4.Pandas Documentation. https://pandas.pydata.org
5.Matplotlib Documentation. https://matplotlib.org
6.Seaborn Documentation. https://seaborn.pydata.org
7.FENGZHIHAO: https://www.kaggle.com/code/fengzhihao/house-price-predict