TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM

KHOA TOÁN - TIN HỌC

MÔN HỌC: PYTHON CHO KHOA HỌC DỮ LIỆU

BÁO CÁO MÔN HỌC

BÀI TOÁN DỰ ĐOÁN GIÁ NHÀ Ở VIỆT NAM SỬ DỤNG MÔ HÌNH HỌC MÁY

Nhóm 20:

1. Nguyễn Đức Bảo **MSSV:** 23280040
2. Tô Trương Đồng **MSSV:** 23280047
3. Nguyễn Đức Hiếu **MSSV:** 23280058

Thành phố Hồ Chí Minh, tháng 12 năm 2025

Mục lục

Phân công công việc

Sinh viên	MSSV	Nhiệm vụ
Nguyễn Đức Bảo	23280040	Model, Visualize
Tô Trương Đồng	23280047	Data Preprocessing, Report
Nguyễn Đức Hiếu	23280058	Model, Pipeline

1 Tổng quan

1.1 Bài toán: Dự đoán giá nhà ở Việt Nam

Thị trường bất động sản Việt Nam phức tạp và giá phụ thuộc nhiều yếu tố. Việc dự đoán giá chính xác giúp:

- Người mua/bán đưa ra quyết định hợp lý
- Nhà đầu tư đánh giá tiềm năng
- Các nền tảng bất động sản tự động định giá tài sản

⇒ Bài toán này là hồi quy: dự đoán giá nhà (biến liên tục) dựa trên các đặc tính như diện tích, vị trí, số phòng, tình trạng pháp lý, và các yếu tố khác.

1.2 Dữ liệu

Dataset: *Vietnam Housing Dataset* gồm 30,229 bản ghi về bất động sản tại Việt Nam, với 12 thuộc tính.

Thuộc tính định lượng:

- Area (Diện tích): Tổng diện tích tính bằng m^2
- Frontage (Mặt tiền): Chiều rộng mặt tiền (mét)
- Access Road (Đường vào): Chiều rộng đường vào (mét)
- Floors (Số tầng): Tổng số tầng
- Bedrooms (Phòng ngủ): Số phòng ngủ
- Bathrooms (Phòng tắm): Số phòng tắm
- Price (Giá): Giá bán (tỷ VNĐ) — biến mục tiêu

Thuộc tính định tính:

- Address (Địa chỉ): Địa chỉ đầy đủ (dự án, đường, phường, quận, thành phố)
- House direction (Hướng nhà): Hướng mặt tiền (Đông, Tây, Nam, Bắc)
- Balcony direction (Hướng ban công): Hướng ban công
- Legal status (Tình trạng pháp lý): Giấy tờ pháp lý (sổ đỏ, đang bán, v.v.)
- Furniture state (Tình trạng nội thất): Nội thất (đầy đủ, một phần, không có)

Đặc điểm:

- Dataset có nhiều giá trị thiếu, đặc biệt ở các cột như **Frontage, Access Road, House direction, Balcony direction**.
- Địa chỉ dạng text, cần xử lý để trích xuất thông tin vị trí
- Giá trị mục tiêu (price) không thiếu, phù hợp cho bài toán hồi quy
- Dataset này phù hợp để xây dựng mô hình dự đoán giá nhà, áp dụng các kỹ thuật xử lý dữ liệu và học máy.

2 Thư viện sử dụng

2.1 `pandas`

- Đọc/ghi dữ liệu (CSV, Excel, JSON)
- Xử lý và làm sạch dữ liệu (DataFrame)
- Feature engineering (tạo cột mới, split cột)
- Được dùng trong `data_preprocessing.py, data_handler.py, ...`

2.2 `numpy`

- Tính toán số học và thao tác mảng
- Hỗ trợ các phép toán trong preprocessing và modeling
- Được dùng trong `data_preprocessing.py, evaluator.py, plots.py, ...`

2.3 `scipy`

- Thống kê và phân tích dữ liệu
- Được dùng trong `data_preprocessing.py` (import stats)

2.4 `scikit-learn (==1.6.1)`

- Machine learning: models (RandomForestRegressor, ElasticNet), preprocessing (imputer, scaler, encoder), pipeline, model selection (train_test_split, KFold, cross_val_score), metrics (RMSE, MAE, R2)
- Được dùng trong `model_trainer.py, pipeline_factory.py, evaluator.py`, các module preprocessing

2.5 `optuna`

- Tối ưu hyperparameters tự động
- Được dùng trong `model_trainer.py` để tìm tham số tối ưu cho mô hình dự đoán

2.6 `xgboost`

- Gradient boosting model (XGBRegressor)
- Được dùng trong `model_trainer.py` như một mô hình để huấn luyện

2.7 `lightgbm`

- Gradient boosting model (LGBMRegressor)
- Được dùng trong `model_trainer.py` như một mô hình để huấn luyện

2.8 `catboost`

- Gradient boosting model (CatBoostRegressor)
- Được dùng trong `model_trainer.py` như một mô hình để huấn luyện

2.9 `joblib`

- Lưu và tải mô hình (định dạng `.joblib`)
- Được dùng trong `model_trainer.py` để lưu mô hình đã được huấn luyện

2.10 `openpyxl`

- Đọc/ghi file Excel (.xlsx, .xls)
- Được pandas sử dụng gián tiếp qua `pd.read_excel()` và `df.to_excel()` trong `data_preprocessing.py`

2.11 `shap`

- Giải thích mô hình (SHAP values)
- Được dùng trong `visualizer/plots.py` để trực quan hóa feature importance

2.12 `seaborn`

- Visualization nâng cao
- Được dùng trong visualizer/plots.py để tạo các biểu đồ phân tích dữ liệu

2.13 `matplotlib`

- Vẽ biểu đồ cơ bản
- Được dùng trong visualizer/plots.py và các utils để tạo biểu đồ trực quan hóa.

3 Các Class được triển khai

3.1 Package modeltrainer

(a) Module `config.py` gồm 01 class `Config` có tác dụng quản lý cấu hình từ file .ini:

- `__init__()`: Có cơ chế tìm file `config.ini` linh hoạt theo nhiều đường dẫn. Hỗ trợ **Greatful Degradation**: nếu không tìm thấy file hoặc lỗi, tự động chuyển sang dùng cấu hình mặc định (`_set_defaults`)
- `_set_defaults()`: Đặt giá trị mặc định khi không tìm thấy file config
- `update_from_args()`: Cập nhật cấu hình từ command-line arguments
- Instance `settings` mặc định của `Config`, luôn được khởi tạo.

(b) Module `data_handler.py` gồm 01 class `DataHandler` có tác dụng tách features/target và chia train/test:

- `load_and_split_data()`: Tách target khỏi features, chia train/test với test_size cho trước

(c) Module `evaluator.py` gồm 01 class `ModelEvaluator` cho việc tính toán và lưu trữ các metric đánh giá mô hình:

- `evaluate()`: Tính RMSE, MAE, R2, ghi log kết quả, lưu và trả về dictionary metrics
- `best_model()`: Tìm mô hình tốt nhất dựa vào các metric
- `save_metrics()`: Lưu tất cả các metric ra file JSON

(d) Module `hyperparams_config.py` gồm 01 class `HyperparameterConfig` có tác dụng xác định không gian tìm kiếm hyperparameter cho các mô hình dùng Optuna:

- `get_model_params()`: Trả về dictionary hyperparameters cho mô hình, hỗ trợ 5 mô hình RandomForestRegressor, XGBRegressor, LGBMRegressor, CatBoostRegressor, ElasticNet

(e) Module `logger.py` triển khai 01 phương thức `setup_logger()` có tác dụng thiết lập logger với console và file handler

(f) Module `model_trainer.py` gồm 01 class `ModelTrainer` điều phối quy trình huấn luyện từ đầu đến cuối:

- `load_and_split()`: Wrapper gọi DataHandler.load_and_split_data()
- `optimize()`: Tối ưu hyperparameter bằng Optuna.
  - Sử dụng chiến lược **K-Fold Cross-Validation (k = 5)** để đảm bảo độ ổn định của bộ tham số.
  - Metric tối ưu: `neg_root_mean_squared_error`.
- `_fit_final_model()`: Retrain mô hình với best params trên toàn bộ training set
- `evaluate_model(): Đánh giá` mô hình trên test set
- `save_model()`: Lưu pipeline đã train bằng joblib
- `save_best_params()`: Lưu best hyperparameters ra JSON
- `save_X_train()`: Lưu X_train ra CSV

(g) Module `pipeline_factory.py` gồm 01 class `PipelineFactory` có tác dụng tạo sklearn pipeline với preprocessing tự động:

- `__init__()`: Tiếp nhận `features_config` để tiếp nhận danh sách các cột cần xử lý.
- `create_pipeline()`: Tạo pipeline tích hợp mô hình và xử lý dữ liệu:
  - Sử dụng các **Custom Transformers** (từ package `datapreprocessor`) để kiểm soát chi tiết quá trình xử lý:
    - Numeric: Median imputation → **CustomStandardScaler**
    - Categorical (OneHot): Constant imputation ('missing') → **CustomOneHotEncoding**
    - Categorical (Target): Constant imputation ('missing') → **CustomTargetEncoding (smooth = 10.0)**
  - Sử dụng `ColumnTransformer` để **định tuyến** dữ liệu vào đúng bộ xử lý
  - Kết hợp với model cuối pipeline

  - Trả về sklearn.pipeline.Pipeline

⇒ **Luồng hoạt đông tổng thể**:

- `Config` đọc cấu hình từ file .ini
- `DataHandler` tải và chia dữ liệu
- `ModelTrainer.optimize()`:
  - Dùng `HyperparameterConfig` để đề xuất hyperparameters
  - Dùng `PipelineFactory` để tạo pipeline
  - Chạy Optuna với cross-validation
  - Retrain với best params
- `ModelEvaluator` đánh giá trên test set
- Lưu model và metrics

3.2 Package datapreprocessor

- Module `data_preprocessing.py` gồm 01 class `DataPreprocessor` cho các thao tác tiền xử lý cơ bản:
  - `load_file()`: Đọc file CSV/Excel/JSON
  - `clean_name()`: Chuẩn hóa tên cột theo định dạng snake_case
  - `detect_outliers()`: Phát hiện giá trị ngoại lại bằng các phương pháp IQR/Z-score/IsolationForest
  - `create_datetime_features()`: Tạo features từ datetime
  - `split_column()`: Tách cột thành nhiều cột
  - `save_to_file()`: Lưu DataFrame ra file
- Module `encoder.py` gồm 03 class dùng cho mã hóa biến phân loại:
  - `CustomOneHotEncoder`
    - `fit()`: Học categories từ training data
    - `transform()`: Áp dụng one-hot encoding
    - `get_feature_names_out()`: Lấy tên các cột đã được fit
  - `CustomLabelEncoder`
    - `fit()`: Học label mappings thủ công (tự cài đặt)
    - `transform()`: Áp dụng label encoding
    - `get_feature_names_out()`: Tương tự như method của one-hot
  - `CustomTargetEncoder`
    - `fit()`: Học target encoding mappings thủ công (tự cài đặt), sử dụng công thức **Smoothing** với `m = 10` để xử lý vấn đề overfitting và các nhóm hiếm (rare categories) theo công thức:

$$\text{SmoothedMean} = \frac{n \times \mu_{\text{group}} + m \times \mu_{\text{global}}}{n + m}$$

      Trong đó:
      · n: Số lượng mẫu của category (count).

· μ_{group}: Giá trị trung bình biến mục tiêu của nhóm.
· μ_{global}: Giá trị trung bình biến mục tiêu toàn cục.
· m: Tham số làm trơn (smoothing parameter), mặc định $m = 10$.

* `transform()`: Áp dụng target encoding
* `get_feature_names_out()`: Tương tự như method của one-hot

- Module `imputer.py` gồm 01 class `CustomImputer` cho yêu cầu xử lý giá trị thiếu:
  - `fit()`: Học imputation parameters từ class SimpleImputer có sẵn của sklearn
  - `transform()`: Áp dụng imputation
  - `get_feature_names_out()`: Lấy tên features output
- Module `scaler.py` gồm 02 class `CustomStandardScaler` và `CustomMinMaxScaler` dùng để chuẩn hóa biến số học:
  - `CustomStandardScaler`
    * `fit()`: Học mean và std từ training data (tự cài đặt)
    * `transform()`: Áp dụng standardization
    * `get_feature_names_out()`: Lấy tên cột đã được fit
  - `CustomMinMaxScaler`
    * `fit()`: Học min và max từ training data (tự cài đặt)
    * `transform()`: Áp dụng min-max scaling
    * `get_feature_names_out()`: Lấy tên cột đã được fit

3.3 Package visualizer

- Module `plots.py`
  - `boxplot()`: Vẽ box plots
  - `countplot()`: Vẽ count plots cho categorical data và numerical data có ít giá trị
  - `piechart()`: Vẽ pie charts
  - `scatterplot()`: Vẽ scatter plot
  - `residualplot()`: Vẽ residual plot
  - `histogram()`: Vẽ histogram
  - `heatmap()`: Vẽ correlation heatmap
  - `linechart()`: Vẽ line chart
  - `KDEplot()`: Vẽ KDE plots
  - `feature_importance_plot()`: Vẽ feature importance
  - `shap_summary_plot()`: Vẽ SHAP summary plot
  - `PDP_plot()`: Vẽ Partial Dependence Plot
- Module `ultis_layout.py`
  - `get_grid_dimensions()`: Tính toán grid dimensions
  - `setup_figure()`: Thiết lập figure cho subplots
  - `setup_single_plot()`: Thiết lập figure cho single plot
- Module `ultis_style.py`

  - Cài đặt màu sắc và style
  - `apply_grid()`: Áp dụng cài đặt grid
- Module `utils_validation.py`
  - `ensure_columns_exist()`: Validate columns tồn tại
  - `ensure_numeric()`: Validate columns là numeric
  - `ensure_categorical_or_low_cardinality()`: Validate categorical và numerical data có ít giá trị ($\leq$ 20 giá trị)
  - `to_list()`: Convert input thành list

4 Luồng xử lý dữ liệu (Pipeline Workflow)

Toàn bộ pipeline cho tiền xử lý, huấn luyện và đánh giá mô hình được triển khai trong file `main.py`. Chỉ cần thực thi file này thì toàn bộ pipeline sẽ tự động diễn ra.

Quy trình thực hiện:

- Cấu hình (Configuration):
  - Tải cấu hình mặc định từ config/config.ini
  - Cập nhật cấu hình dựa trên tham số dòng lệnh (CLI arguments) nếu có
- Nhập và Tiền xử lý dữ liệu (Data Ingestion & Preprocessing):
  - Đọc dữ liệu các định dạng file *.csv, .xlsx, .xls, .json*
  - Làm sạch tên cột
  - Phát hiện và loại bỏ giá trị ngoại lai
  - Feature Engineering:
    * Tách và xử lý cột địa chỉ `address` thành `new_address` (Quận, Thành phố) để tăng tính tương quan.
    * Tạo thêm cột `condo_complex` từ cột `address` $\Rightarrow$ Tăng khả năng biểu diễn thông tin.
  - Lưu dataset đã được làm sạch vào data/cleaned_dataset.csv
- Lựa chọn đặc trưng cho mô hình máy học (Feature Selection)
- Huấn luyện và tối ưu hóa siêu tham số
  - Khởi tạo `ModelTrainer` và chia tập dữ liệu (Train/Test split)
  - Xác định danh sách mô hình cần chạy (do người dùng chỉ định hoặc danh sách mặc định: Elastic Net, RandomForest, XGBoost, CatBoost, Lightgbm)
  - Chạy vòng lặp tối ưu hóa tham số (Hyperparameter Tuning) sử dụng Optuna cho từng mô hình
- Đánh giá, lưu trữ và triển khai (Evaluation & Deployment)
  - Đánh giá hiệu năng mô hình trên tập Test.
  - Lưu mô hình tốt nhất dưới định dạng *.joblib* kèm timestamp vào thư mục models
  - So sánh các mô hình và chọn ra **Mô hình tối ưu nhất** dựa trên chỉ số do người dùng nhập vào (RMSE, MAE hoặc R2) với tùy chọn **hướng tối ưu hóa**

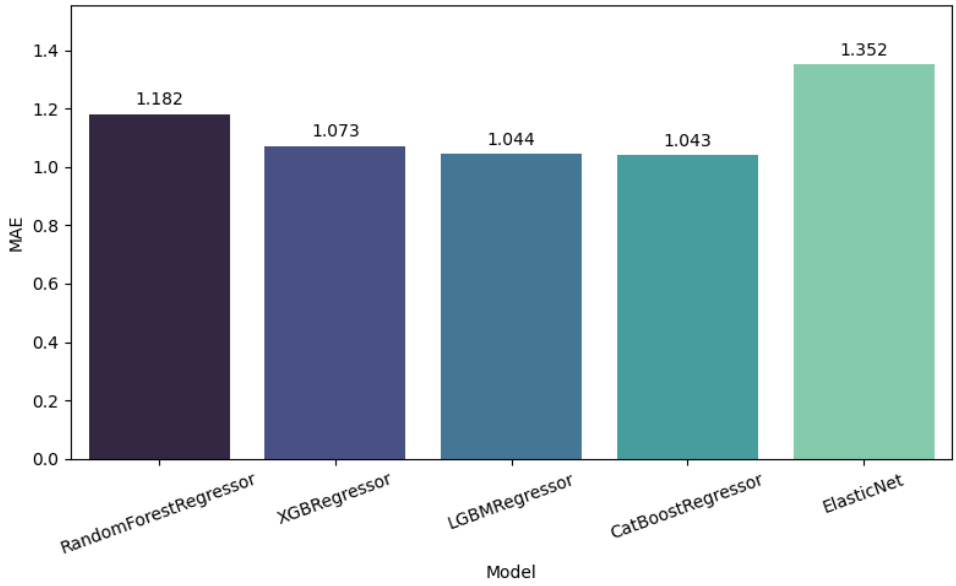
5 Phân tích kết quả thí nghiệm

Lưu ý: Các chỉ số trong báo cáo được lấy từ 1 lần chạy model của nhóm, đối với những lần chạy khác, có thể các chỉ số sẽ khác nhau.

Vấn đề nhóm đang giải quyết là bài toán hồi quy, do đó để đánh giá hiệu suất của mô hình ML cho bài toán hồi quy ta sử dụng các chỉ số sau:

5.1 Chỉ số Mean Absolute Error (MAE)

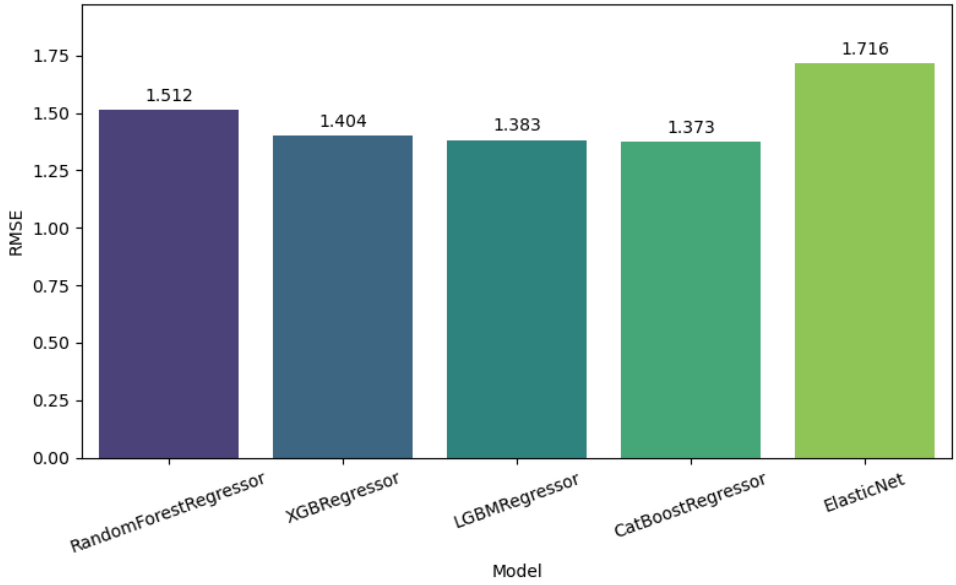
- Ý nghĩa chỉ số: Đo sai số trung bình giữa giá trị dự đoán và giá trị thực tế
- Chỉ số MAE trong bài toán này giải thích rằng trung bình mô hình sai khoảng X tỷ đồng. Chỉ số này càng thấp càng tốt
- Model CatBoostRegressor có MAE thấp nhất (1.043) cho thấy trung bình mô hình dự đoán sai khoảng 1.043 tỷ đồng (trên thực tế sai số này khá lớn)



Hình 1: Mean Absolute Error của 05 model

5.2 Chỉ số Root Mean Squared Error (RMSE)

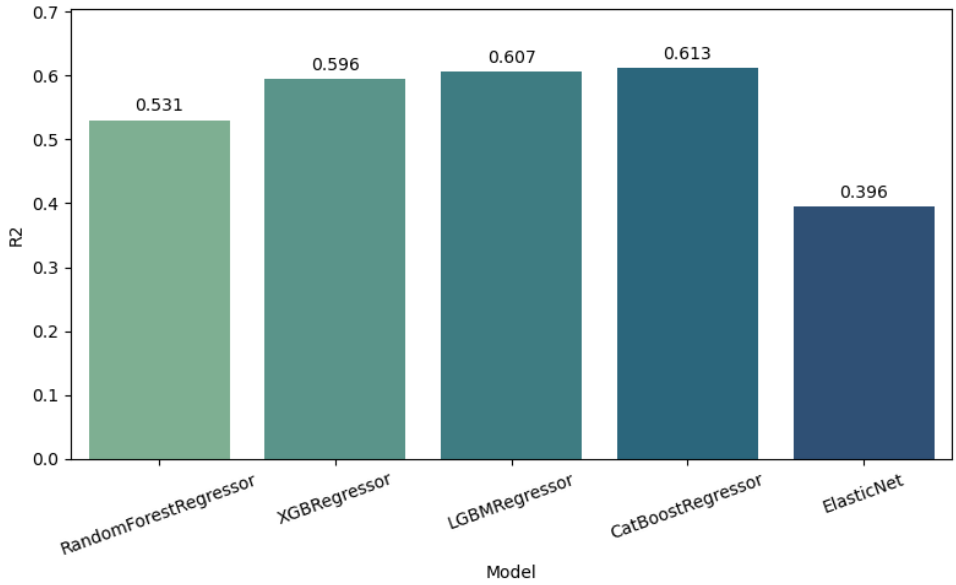
- Ý nghĩa: Độ lớn của sai số
- Chỉ số RMSE trong bài toán hồi quy giải thích sự sai lệch giữa giá nhà dự đoán và giá nhà thực tế. Vì công thức có bình phương sai số nên chỉ số phản ánh rõ các sai số lớn. Chỉ số này càng thấp càng tốt
- Model CatBoostRegressor có RMSE thấp nhất (1.373) cho thấy độ lớn sai số là 1.373 tỷ đồng (cũng khá lớn so với thực tế)



Hình 2: Root Mean Squared Error của 05 model

5.3 Chỉ số R-squared score (R^2)

- Ý nghĩa: Đo tỷ lệ phương sai của biến mục tiêu được giải thích bởi mô hình
- Chỉ số R^2 trong bài toán này cho biết rằng mô hình có khả năng giải thích được $X\%$ sự biến thiên của giá nhà. Chỉ số này càng lớn càng tốt. Tuy nhiên cũng nên xem xét các chỉ số khác theo kèm để tránh overfitting nếu R^2 quá cao
- Model CatBoostRegressor có giá trị $R^2 = 0.613$, điều này có nghĩa là mô hình CatBoostRegressor giải thích được 61.3% sự biến thiên trong giá nhà ở Việt Nam



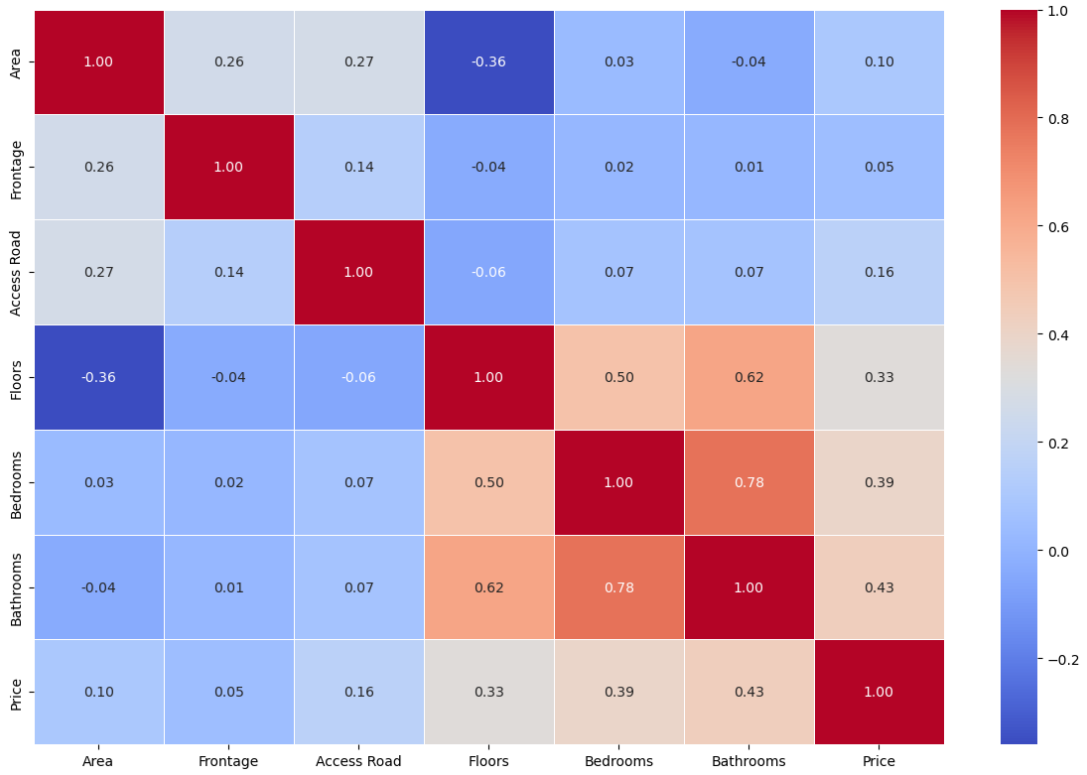
Hình 3: R^2 score của 05 model

5.4 Giải thích sự chênh lệch metric giữa các mô hình

Ta thấy rằng có tree-based model đều cho các chỉ số MAE và RMSE gần tương đương nhau còn linear model như Elastic Net lại cho ra kết quả cao hơn, điều này cũng dẫn tới chỉ số R^2 cũng

khác nhau giữa tree-based và linear model.

Lí do gây ra điều này bởi vì tính đa cộng tuyến (multicollinearity) và tính chất phi tuyến giữa các feature và target (*price*).



Hình 4: Correlation Heatmap

Mặc dù nhóm đã sử dụng model Elastic Net có tác dụng chọn lọc biến, phù hợp cho xử lý đa cộng tuyến nhưng vì mối tương quan tuyến tính giữa biến target và các feature không cao (theo heatmap) nên một linear model như Elastic Net không phù hợp cho dữ liệu giá nhà này.

Tính chất phi tuyến này thể hiện rõ hơn khi thay đổi từ linear model thành tree-based model, các chỉ số MAE, RMSE và R^2 đề cải thiện theo hướng tốt hơn (MAE và RMSE thấp hơn, R^2 tăng lên). Do đó các model tree-based sẽ thích hợp hơn cho dữ liệu giá nhà mà nhóm đã chọn vì có thể phát hiện và học được những cấu trúc phức tạp.