Dự đoán giá nhà đất dựa trên các thuộc tính bất động sản

Dự án này tập trung vào việc dự đoán giá nhà dựa trên các thuộc tính của bất động sản, sử dụng các thuật toán học máy như **Neural Network, k-Nearest Neighbors (kNN), Decision Tree,** và **Random Forest**. Mỗi thuật toán có những ưu nhược điểm riêng, từ khả năng xử lý dữ liệu phức tạp của **Neural Network**, sự đơn giản và dễ hiểu của **kNN**, đến tính giải thích dễ dàng của **Decision Tree** và hiệu suất cao của **Random Forest**. Kết quả cho thấy **Random Forest** có độ chính xác cao nhất với $\mathbf{R^2}$ là 0.8915, tiếp theo là **Neural Network** với $\mathbf{R^2}$ là 0.8743

1. Giới thiệu

Bài toán dự đoán giá nhà, một vấn đề quan trọng trong lĩnh vực **bất động sản và học máy**. Mục tiêu chính là xây dựng các mô hình có khả năng dự đoán chính xác giá bán của một căn nhà dựa trên các **đặc điểm và thuộc tính** của nó.

Dự đoán giá nhà là một công cụ quan trọng trong việc định giá bất động sản, giúp người mua, người bán và các nhà đầu tư đưa ra quyết định sáng suốt. Các mô hình dự đoán chính xác có thể hỗ trợ các chuyên gia bất động sản, ngân hàng và các tổ chức tài chính trong việc đánh giá tài sản thế chấp và quản lý rủi ro.

Nghiên cứu này góp phần phát triển và so sánh hiệu quả của các thuật toán học máy khác nhau, từ đó cung cấp insights về ưu nhược điểm của từng phương pháp trong bối cảnh cụ thể của bài toán dự đoán giá nhà.

Trong thực tế, ứng dụng của nghiên cứu này có thể bao gồm: Hỗ trợ các trang web bất động sản trong việc cung cấp ước tính giá nhà cho người dùng. Giúp các cơ quan quản lý đánh giá xu hướng thị trường bất động sản. Hỗ trợ các nhà phát triển bất động sản trong việc định giá dự án mới.

Để giải quyết bài toán, project áp dụng và so sánh hiệu suất của bốn phương pháp **học máy** chính:

- **Neural Network**: Một mạng nơ-ron đa lớp được sử dụng để học các mối quan hệ phức tạp giữa các đặc trưng.
- **k-Nearest Neighbors (kNN):** Một thuật toán đơn giản dựa trên nguyên tắc các điểm dữ liệu gần nhau có xu hướng có giá trị tương tự.
- **Decision Tree**: Một mô hình dự đoán dựa trên cấu trúc cây, phân chia dữ liệu dựa trên các đặc trưng quan trọng nhất.
- **Random Forest**: Một tập hợp các cây quyết định, kết hợp dự đoán của nhiều cây để đưa ra kết quả cuối cùng.

Các phương pháp này được áp dụng sau quá trình tiền xử lý dữ liệu kỹ lưỡng, bao gồm xử lý dữ liệu thiếu, mã hóa đặc trưng phân loại và chuẩn hóa đặc trưng số. Hiệu suất của các mô hình được đánh giá và so sánh thông qua các chỉ số như **MSE, RMSE, MAE** và $\mathbf{R^2}$**.**

2. Các công trình liên quan

2.1. Nghiên cứu sử dụng Machine Learning để dự đoán giá nhà đất tại Mỹ

Nghiên cứu sử dụng Machine Learning để dự đoán giá nhà đất tại Mỹ đã thu hút sự chú ý đáng kể trong lĩnh vực bất động sản và khoa học dữ liệu. Các nhà nghiên cứu đã áp dụng nhiều mô hình Machine Learning khác nhau, bao gồm **Linear Regression**, **Random Forest** và **Gradient Boosting**, để phân tích và dự đoán giá nhà dựa trên một loạt các biến số. Dữ liệu đầu vào bao gồm các thuộc tính vật lý của bất động sản như diện tích, số phòng ngủ, số phòng tắm, cũng như các yếu tố vị trí như khoảng cách đến trung tâm thành phố, chất lượng trường học trong khu vực, và tỷ lệ tội phạm. Kết quả cho thấy các mô hình này có khả năng dự đoán giá nhà với độ chính xác cao, đặc biệt là **Random Forest** và **Gradient Boosting**, thường vượt trội hơn so với phương pháp hồi quy tuyến tính truyền thống.

2.2. Ứng dụng Deep Learning trong dự đoán giá bất động sản tại Trung Quốc

Tại Trung Quốc, việc ứng dụng **Deep Learning** trong dự đoán giá bất động sản đã mở ra những triển vọng mới cho ngành này. Các nhà nghiên cứu đã phát triển các mô hình **Deep Neural Networks** (**DNN**) phức tạp, có khả năng xử lý và học từ các bộ dữ liệu lớn và đa chiều. Ngoài các thuộc tính bất động sản thông thường, các mô hình này còn tích hợp dữ liệu vĩ mô như tình hình kinh tế địa phương, xu hướng dân số, và chính sách quy hoạch đô thị. Kết quả cho thấy các mô hình **DNN** có khả năng dự đoán giá bất động sản với độ chính xác cao hơn đáng kể so với các phương pháp Machine Learning cổ điển, đặc biệt trong các thị trường biến động mạnh như các thành phố lớn của Trung Quốc.

2.3. Nghiên cứu kết hợp GIS và phân tích không gian trong dự đoán giá bất động sản tại Việt Nam

Nghiên cứu kết hợp **GIS** và phân tích không gian trong dự đoán giá bất động sản tại Việt Nam đã mang lại một góc nhìn mới về tầm quan trọng của yếu tố địa lý trong việc định giá nhà đất. Phương pháp này tích hợp dữ liệu từ hệ thống thông tin địa lý (**GIS**) với các kỹ thuật phân tích không gian tiên tiến để tạo ra các mô hình dự đoán chính xác. Các nhà nghiên cứu đã sử dụng dữ liệu vị trí chi tiết, bao gồm thông tin về tiện ích xung quanh như trường học, bệnh viện, công viên, cũng như dữ liệu về mức độ phát triển hạ tầng và quy hoạch đô thị. Kết quả nghiên cứu cho thấy phương pháp này có khả năng dự đoán giá nhà đất với độ chính xác cao, đặc biệt trong các khu vực đô thị phức tạp của Việt Nam, nơi giá bất động sản có sự chênh lệch lớn giữa các khu vực lân cận.

3. Phát biểu bài toán

3.1. Bài toán

Dự đoán giá nhà là một vấn đề quan trọng trong lĩnh vực bất động sản và học máy. Mục tiêu của bài toán này là xây dựng các mô hình có khả năng dự đoán chính xác giá bán của một căn nhà dựa trên các đặc điểm và thuộc tính của nó. Bài toán này thuộc loại học có giám sát (**Supervised Learning**), cụ thể là bài toán hồi quy (**Regression**), trong đó biến mục tiêu (**SalePrice**) là một giá trị liên tục.

Input của bài toán bao gồm một tập hợp các thuộc tính của bất động sản, bao gồm cả các đặc điểm số học (ví dụ: diện tích, số phòng ngủ) và các đặc điểm phân loại (ví dụ: vị trí, loại nhà). Output là giá dự đoán của căn nhà, được biểu diễn bằng một giá trị số.

Dữ liệu được chia thành hai tập: tập huấn luyện (train_data) và tập kiểm tra (test_data). Tập huấn luyện được sử dụng để xây dựng và điều chỉnh các mô hình, trong khi tập kiểm tra được dùng để đánh giá hiệu suất của các mô hình trên dữ liệu mới.

3.2. Thuật toán

Neural Network: Mạng nơ-ron đa lớp (**MLP**) với hai lớp ẩn, có 64 và 32 nơ-ron, giúp học các mối quan hệ phi tuyến phức tạp giữa các đặc trưng của bất động sản. Mạng nơ-ron tự động điều chỉnh trọng số qua thuật toán lan truyền ngược để giảm sai số. Ưu điểm là khả năng xử lý dữ liệu đa chiều và học mối quan hệ phức tạp, nhưng đòi hỏi lượng lớn dữ liệu và kết quả khó giải thích do tính "hộp đen".

Đầu ra của mỗi neural: $y = f(\sum(w_i * x_i) + b)$

k-Nearest Neighbors (kNN): Thuật toán đơn giản tìm kiếm **k** căn nhà gần nhất dựa trên khoảng cách **Euclidean** và ước tính giá dựa trên trung bình giá của chúng. Ưu điểm là đơn giản, không đòi hỏi giả định về dữ liệu, nhưng khó xử lý dữ liệu nhiều chiều và tốn thời gian tính toán với tập dữ liệu lớn.

Khoảng cách **Euclidean**: $d = \sqrt{\sum(x^i - y^i)^2}$

Decision Tree: Mô hình dự đoán dựa trên cấu trúc cây, chia dữ liệu theo các đặc trưng quan trọng. Ưu điểm là xử lý cả dữ liệu số và phân loại, nhưng dễ bị **overfitting** và không ổn định khi dữ liệu thay đổi nhỏ.

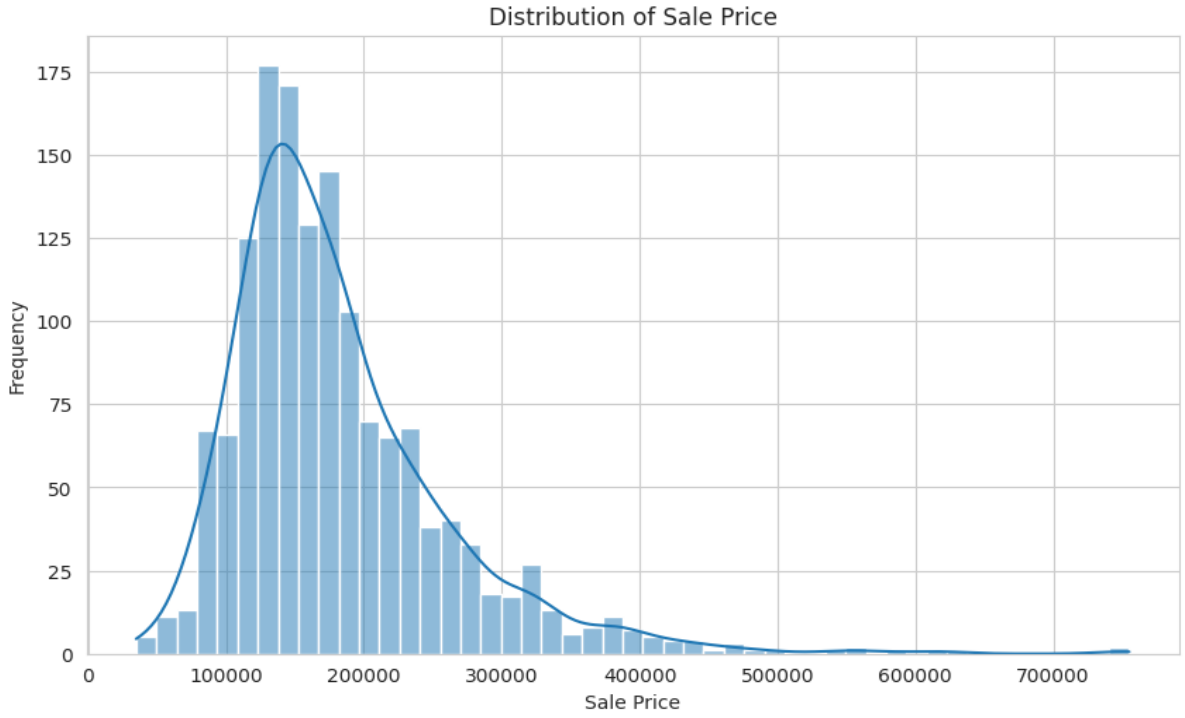
Random Forest: Tập hợp các cây quyết định, kết hợp dự đoán của nhiều cây để đưa ra kết quả cuối cùng. **Random Forest** xây dựng nhiều cây trên các mẫu con ngẫu nhiên và sử dụng trung bình dự đoán từ tất cả các cây. Ưu điểm là hiệu suất cao, ít bị **overfitting** hơn cây đơn lẻ, và xử lý dữ liệu nhiều đặc trưng, nhưng đòi hỏi nhiều tài nguyên tính toán và kết quả khó giải thích hơn.

$$y_{pred} = \frac{1}{n}\sum(y_i)$$

4. Thực nghiệm

4.1. Dữ liệu

- Số lượng mẫu: 1460, số lượng đặc trưng: 81 (bao gồm cả biến mục tiêu **SalePrice**)
- Kiểu dữ liệu: 35 cột kiểu **int64**, 3 cột kiểu **float64**, 43 cột kiểu **object** (**categorical**)



Hình 1: Dữ liệu phân bố theo SalePrice

Hình 2: Dữ liệu phân bố của SalePrice và GrLivArea

Một số đặc trưng quan trọng:

- **Id**: Mã số nhận dạng của mỗi căn nhà
- **SalePrice**: Giá bán của căn nhà (**biến mục tiêu**)
- **MSSubClass**: Loại nhà ở
- **MSZoning**: Phân loại khu vực

- **LotFrontage**: Chiều dài mặt tiền lô đất
- **LotArea**: Diện tích lô đất
- **Neighborhood**: Khu vực lân cận
- **OverallQual**: Chất lượng tổng thể của ngôi nhà
- **YearBuilt**: Năm xây dựng
- **GrLivArea**: Diện tích sinh hoạt trên mặt đất

Vấn đề của dữ liệu: Dữ liệu thiếu(Missing Data):

- **PoolQC**: 1453 giá trị thiếu
- **MiscFeature**: 1406 giá trị thiếu
- **Alley**: 1369 giá trị thiếu
- **Fence**: 1179 giá trị thiếu
- **FireplaceQu**: 690 giá trị thiếu

Tiền xử lý dữ liệu:

a) **Xử lý dữ liệu thiếu:** Để xử lý dữ liệu thiếu, ta sử dụng **SimpleImputer** với chiến lược **'mean'** để điền giá trị trung bình cho các đặc trưng số và chiến lược **'most_frequent'** để điền giá trị phổ biến nhất cho các đặc trưng phân loại. Ta mã hóa các đặc trưng phân loại bằng **OneHotEncoder** và chuẩn hóa các đặc trưng số bằng **StandardScaler** để đảm bảo các giá trị này được phân phối đồng đều, giúp cải thiện hiệu suất của các mô hình học máy.

b) **Xử lý đặc trưng:**

```
train_columns = train_data.columns.tolist()
train_columns.remove('SalePrice')  # Loại bỏ cột SalePrice khỏi danh sách cột

# Kiểm tra xem test_data có tất cả các cột cần thiết không
missing_columns = set(train_columns) - set(test_data.columns)
if missing_columns:
    print(f"Các cột sau đây không có trong test_data: {missing_columns}")
    # Thêm các cột thiếu với giá trị NaN
    for col in missing_columns:
        test_data[col] = np.nan

test_data = test_data[train_columns]

print("Số cột trong train_data (không bao gồm SalePrice):",
len(train_columns))

X = train_data.drop(['SalePrice'], axis=1)
y = train_data['SalePrice']

# Xác định các cột số và cột phân loại
numeric_features = X.select_dtypes(include=['int64', 'float64']).columns
categorical_features = X.select_dtypes(include=['object']).columns
```

```
# Tạo các transformer cho các cột số và cột phân loại
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

# Kết hợp các transformer lại với nhau thành ColumnTransformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])

# Áp dụng tiền xử lý dữ liệu
X = preprocessor.fit_transform(X)
test_data = preprocessor.transform(test_data)

print("Số đặc trưng trong X sau khi tiền xử lý:", X.shape[1])
```

Số đặc trưng ban đầu: 80 (không tính **SalePrice**)

Số đặc trưng sau khi tiền xử lý: 288 (do quá trình **one-hot encoding** tạo ra nhiều cột mới)

***Phân chia dữ liệu:**

Tập huấn luyện (training set): 70% dữ liệu / Tập kiểm tra (validation set): 30% dữ liệu

Nhận xét:

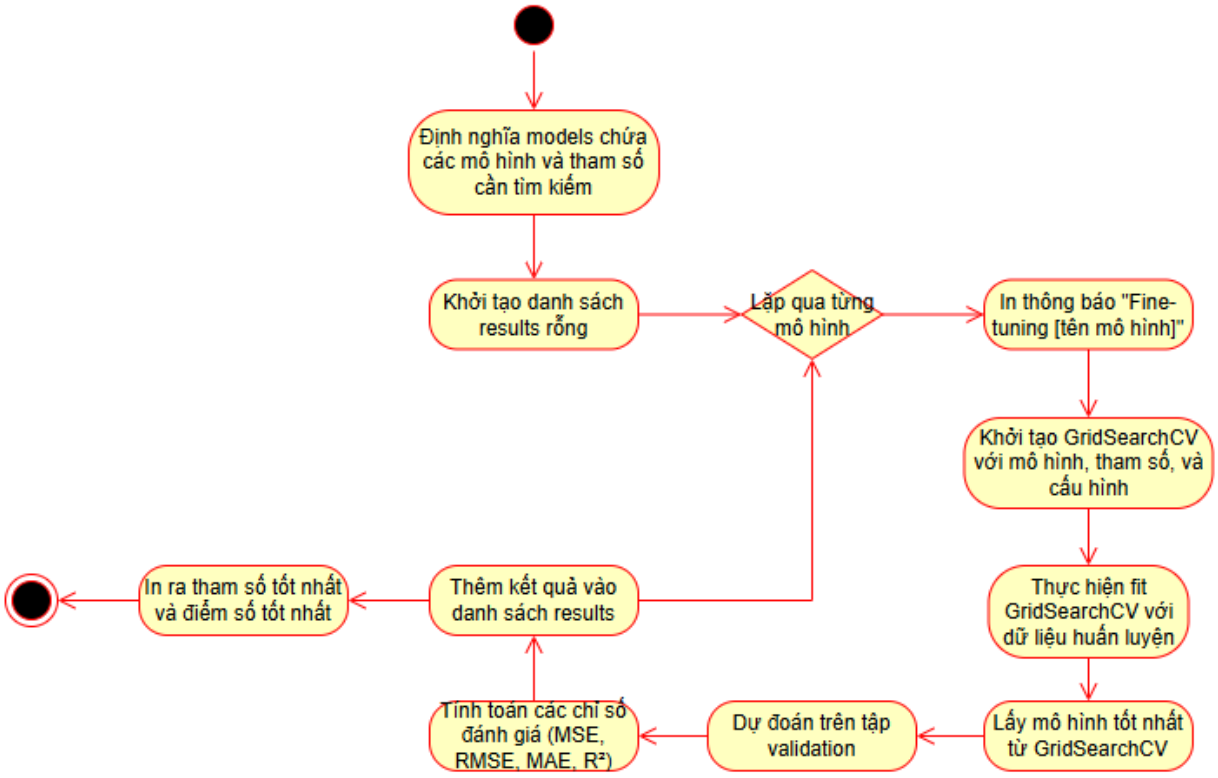
Quá trình tiền xử lý đã giải quyết vấn đề dữ liệu **thiếu** và **mã hóa đặc trưng** phân loại.

Số lượng đặc trưng tăng đáng kể sau quá trình tiền xử lý, có thể cần xem xét việc áp dụng kỹ thuật **giảm chiều dữ liệu** hoặc **lựa chọn đặc trưng** trong tương lai.

Cần theo dõi hiệu suất của mô hình trên tập kiểm tra để đảm bảo không xảy ra hiện tượng **overfitting**.

4.2. Phương pháp

Các bước thực hiện:



Hình 2: Flowchart của bài toán

Hyperparameters:

- **Neural Network (MLPRegressor):**
  - **hidden_layer_sizes:** [(32,), (64,), (32, 16), (64, 32)]
  - **activation:** ['relu', 'tanh']
  - **alpha**: [0.0001, 0.001, 0.01]
  - **max_iter:** [1000, 2000]
- **k-Nearest Neighbors (KNeighborsRegressor):**
  - **n_neighbors:** [3, 5, 7, 9, 11]
  - **weights:** ['uniform', 'distance']
  - **algorithm:** ['auto', 'ball_tree', 'kd_tree']
- **Decision Tree (DecisionTreeRegressor):**
  - • **max_depth:** [None, 10, 20, 30]
  - • **min_samples_split:** [2, 5, 10]
  - • **min_samples_leaf:** [1, 2, 4]
- **Random Forest (RandomForestRegressor):**
  - • **n_estimators**: [50, 100, 200]
  - • **max_depth:** [None, 10, 20, 30]
  - • **min_samples_split:** [2, 5, 10]
  - • **min_samples_leaf:** [1, 2, 4]

Để đánh giá hiệu suất của các mô hình, một số chỉ số phổ biến trong Machine Learning sử dụng như:

- **Mean Squared Error (MSE):** Đo lường trung bình bình phương của sai số giữa giá dự đoán và giá thực tế:

$$\text{MSE} = \frac{\sum(y\_true - y\_pred)^2}{n}$$

- **Root Mean Squared Error (RMSE):** Căn bậc hai của MSE, cho phép đánh giá sai số trong cùng đơn vị với biến mục tiêu:

$$\text{RMSE} = \sqrt{\text{MSE}}$$

- **Mean Absolute Error (MAE):** Đo lường trung bình của giá trị tuyệt đối sai số:

$$\text{MAE} = \frac{\sum |y\_true - y\_pred|}{n}$$

- **Coefficient of Determination (R^2):** Đo lường mức độ phù hợp của mô hình, cho biết phần trăm biến thiên của biến mục tiêu được giải thích bởi mô hình.

$$R^2 = 1 - \frac{\sum(y\_true - y\_pred)^2}{\sum(y\_true - y\_mean)^2}$$

Việc so sánh hiệu suất của các mô hình thông qua các chỉ số này và biểu diễn kết quả bằng bảng và biểu đồ cho phép đánh giá toàn diện khả năng dự đoán giá nhà của từng thuật toán. Điều này giúp hiểu rõ hơn về ưu và nhược điểm của mỗi phương pháp trong bối cảnh cụ thể của bài toán dự đoán giá nhà, từ đó có thể đưa ra quyết định về việc lựa chọn mô hình phù hợp nhất hoặc kết hợp các mô hình để tận dụng ưu điểm của từng phương pháp.

4.3. Kết quả

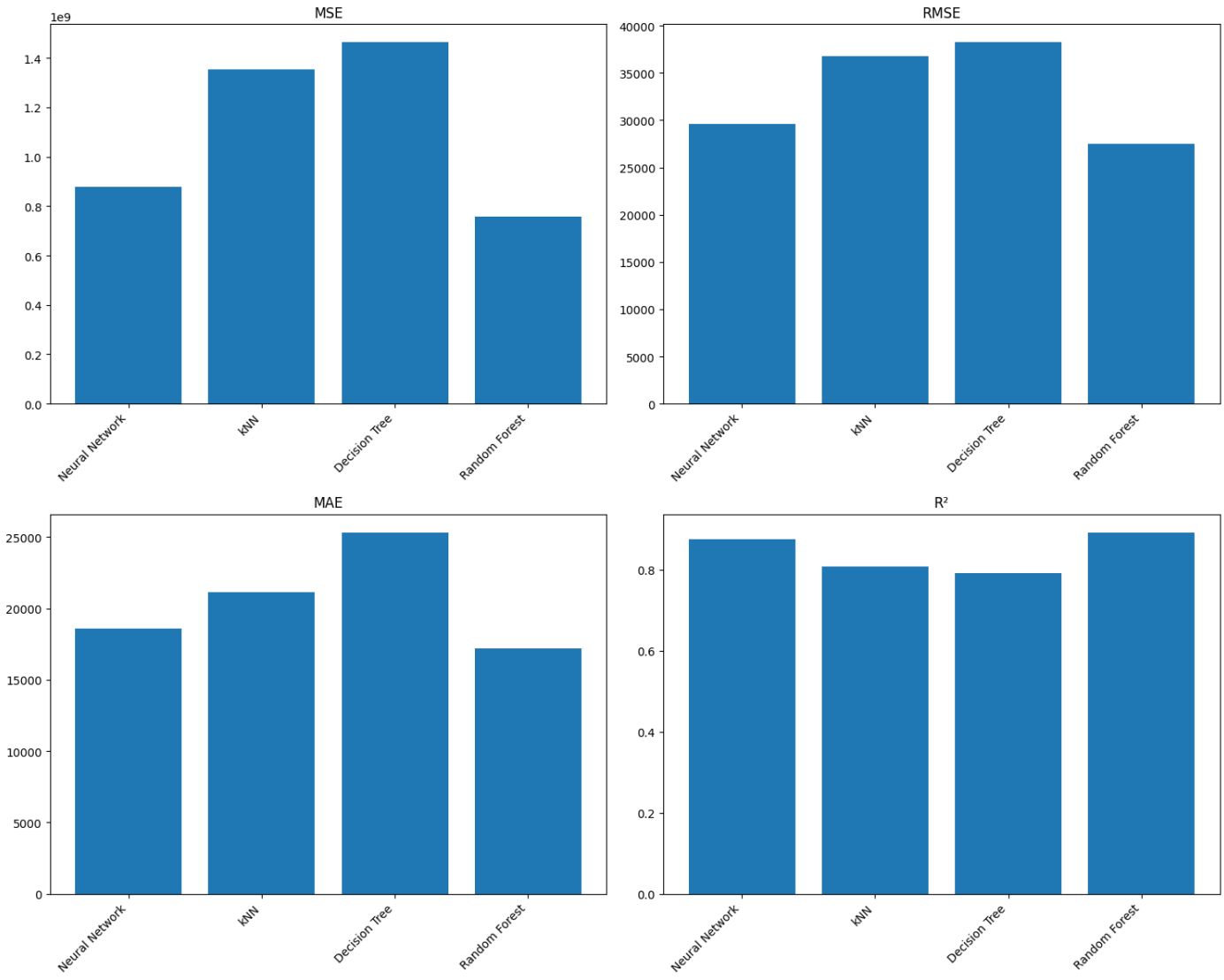
```
Best parameters for Neural Network: {'activation': 'relu', 'alpha': 0.01, 'hidden_layer_sizes': (64, 32), 'max_iter': 1000}
Best score: 1387625952.3119

Fine-tuning kNN...
Best parameters for kNN: {'algorithm': 'auto', 'n_neighbors': 11, 'weights': 'distance'}
Best score: 1484831253.6598

Fine-tuning Decision Tree...
Best parameters for Decision Tree: {'max_depth': 20, 'min_samples_leaf': 2, 'min_samples_split': 10}
Best score: 1907400683.1318

Fine-tuning Random Forest...
Best parameters for Random Forest: {'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}
Best score: 1085052285.4766
```

Model	MSE	RMSE	MAE	R^2
Neural Network	876957659.9613	29613.4709	18598.2031	0.8743
kNN	1352315628.6275	36773.8444	21162.0105	0.8062
Decision Tree	1463030984.5533	38249.5880	25324.8345	0.7903
Random Forest	757246603.1666	27518.1141	17221.9010	0.8915



Hình 3: Kết quả của 4 thuật toán sau train

Phân tích kết quả thực nghiệm và so sánh hiệu suất của các mô hình cho thấy **Random Forest** nổi bật với hiệu suất tốt nhất. Mô hình này đạt được **R^2** cao nhất (0.8915) và **MSE** thấp nhất (757246603.1666), cùng với **RMSE** và **MAE** thấp nhất, cho thấy độ chính xác cao trong dự đoán. Sự vượt trội của **Random Forest** có thể được giải thích bởi khả năng xử lý tốt các mối **quan hệ phi tuyến** và **giảm thiểu overfitting** thông qua việc **kết hợp nhiều cây quyết định**.

Neural Network đứng thứ hai về hiệu suất với **R^2** là 0.8743. Mặc dù không bằng **Random Forest**, nhưng **Neural Network** vẫn thể hiện khả năng học các mối quan hệ phức tạp trong dữ liệu. Cấu hình tối ưu của mạng bao gồm hàm kích hoạt *'relu'*, *alpha 0.01*, và hai lớp ẩn với 64 và 32 node. Điều này cho thấy mô hình có khả năng cân bằng giữa việc nắm bắt thông tin phức tạp và **tránh overfitting**.

kNN và **Decision Tree** cho kết quả kém hơn, với **kNN** đứng thứ ba (**R^2** = 0.8062) và **Decision Tree** cuối cùng (**R^2** = 0.7903). **kNN** sử dụng **11 neighbors** gần nhất và trọng số dựa trên khoảng cách, trong khi **Decision Tree** có độ sâu tối đa là 20. Hiệu suất thấp hơn của **kNN** có thể do khó khăn trong

việc xử lý dữ liệu nhiều chiều, trong khi **Decision Tree** có thể gặp vấn đề overfitting hoặc không đủ khả năng nắm bắt các mối quan hệ phức tạp trong dữ liệu.

Dựa trên kết quả này, **Random Forest** nên được xem xét như mô hình chính cho bài toán. **Tuy nhiên**, vẫn có thể cải thiện hiệu suất bằng cách thử nghiệm với số lượng cây lớn hơn. Đối với **Neural Network**, việc thử nghiệm với các cấu trúc mạng phức tạp hơn hoặc tăng số lần lặp có thể mang lại kết quả tốt hơn. **kNN** có thể được cải thiện bằng cách áp dụng các phương pháp **giảm chiều dữ liệu** trước khi huấn luyện. **Decision Tree**, mặc dù có hiệu suất thấp nhất khi sử dụng đơn lẻ, vẫn đóng góp vào hiệu suất tốt của **Random Forest**.

5. Kết luận

Random Forest đạt hiệu suất tốt nhất với $\mathbf{R^2}$ cao nhất (0.8915) và **MSE** thấp nhất (757246603.1666).

Thứ tự hiệu suất của các mô hình từ tốt nhất đến kém nhất là:

Random Forest > Neural Network > k-Nearest Neighbors > Decision Tree

Neural Network đứng thứ hai với $\mathbf{R^2}$ là 0.8743, sử dụng hàm kích hoạt *'relu'*, *alpha 0.01*, và hai lớp ẩn (64, 32 nodes).

kNN sử dụng **11 neighbors** gần nhất và trọng số dựa trên khoảng cách.

Decision Tree có độ sâu tối đa là **20.**

Bảng đánh giá 4 thuật toán		
	Ưu điểm	**Nhược điểm**
Random Forest	Xử lý tốt các mối quan hệ phi tuyến, giảm thiểu overfitting	Có thể tốn nhiều tài nguyên tính toán, khó giải thích kết quả
Neural Network	Khả năng học các mối quan hệ phức tạp trong dữ liệu	Có thể cần nhiều dữ liệu để huấn luyện hiệu quả, khó giải thích
kNN	Không cần giả định về dạng phân phối của dữ liệu	Khó xử lý dữ liệu nhiều chiều, tốn thời gian tính toán với dữ liệu lớn
Decision Tree	Có thể xử lý cả dữ liệu số và phân loại: Không cần tiền xử lý phức tạp cho dữ liệu đầu vào.	Dễ bị overfitting, không ổn định với thay đổi nhỏ trong dữ liệu

Đề xuất hướng phát triển:

- Tối ưu hóa **Random Forest**: Thử nghiệm với số lượng cây lớn hơn để cải thiện hiệu suất.
- Cải tiến **Neural Network**: Thử nghiệm với cấu trúc mạng phức tạp hơn hoặc tăng số lần lặp.
- Cải thiện **kNN**: Áp dụng các phương pháp giảm chiều dữ liệu trước khi huấn luyện.
- Thử nghiệm với các thuật toán khác: Như **Gradient Boosting**, **XGBoost** để so sánh hiệu suất.
- Tối ưu hóa **hyperparameters**: Sử dụng các phương pháp tìm kiếm **hyperparameters** nâng cao như **Bayesian Optimization**.

Tài liệu tham khảo

1. **Scikit-learn: Machine Learning in Python**, Pedregosa et al., Journal of Machine Learning Research, 12, 2011, pp. 2825-2830.
2. Géron, A., "**Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**", 2nd Edition, O'Reilly Media, Sebastopol, CA, 2019.
3. Chollet, F., "**Deep Learning with Python**", **2nd Edition**, Manning Publications, Shelter Island, NY, 2021.
4. Breiman, L., "**Random Forests**", Machine Learning, 45(1), 2001, pp. 5-32.
5. Goodfellow, I., Bengio, Y., & Courville, A., "**Deep Learning**", MIT Press, Cambridge, MA, 2016.
6. Abadi, M., et al., "**TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**", Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, Savannah, GA, Nov. 2-4, 2016.
7. Gulli, A., & Pal, S., "**Deep Learning with Keras**", Packt Publishing, Birmingham, UK, 2017.
8. Flach, P., "**Machine Learning**: The Art and Science of Algorithms that Make Sense of Data", Cambridge University Press, Cambridge, UK, 2012.
9. Bergstra, J., Bengio, Y., **Random Search for Hyper-Parameter Optimization, Journal of Machine Learning Research** (13), 2012, pp. 281-305.
10. LeCun, Y., Bengio, Y., Hinton, G., **Deep learning**, **Nature** (521), 2015, pp. 436-444.
11.

Bảng phân chia công việc	
Ngô Cự Văn	Thu thập và làm sạch dữ liệu: • Phân tích dữ liệu ban đầu • Xử lý dữ liệu thiếu và ngoại lai • Mã hóa các đặc trưng phân loại • Chuẩn hóa dữ liệu số Xây dựng và đánh giá các mô hình: Random Forest, Decision Tree Tối ưu hóa hyperparameters cho các mô hình trên So sánh hiệu suất giữa tất cả các mô hình Tổng hợp kết quả và viết báo cáo cuối cùng
Lâm Chí Dũng	Xây dựng và đánh giá các mô hình: Neural Network, k-Nearest Neighbors (kNN) Tối ưu hóa hyperparameters cho các mô hình trên Phân tích và trực quan hóa kết quả của các mô hình này Nghiên cứu và đề xuất cải tiến: • Thử nghiệm với các thuật toán khác như Gradient Boosting, XGBoost • Tìm hiểu về các phương pháp tối ưu hóa hyperparameters nâng cao Tìm hiểu các công trình liên quan

Link source code:
https://colab.research.google.com/drive/1oKE_dO98l4nyIQATVwCgOpWqXKbt7tc9?usp=drive_link