BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC ĐẠI NAM

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN

XÂY DỰNG MÔ HÌNH MẠNG NEURON ĐỂ DỰ ĐOÁN GIÁ NHÀ

Sinh viên thực hiện : **Hoàng Văn Chiến**
Trần Đông Đức
Trần Thanh Bình
Lê Trung Kiên

Ngành : **Công nghệ thông tin**

Giảng viên hướng dẫn : **ThS. Lê Thị Thùy Trang**

Lời cảm ơn

Trước hết, em xin gửi lời cảm ơn chân thành đến giảng viên Lê Thị Thùy , người đã tận tình hướng dẫn và cung cấp những kiến thức quý báu giúp em hoàn thành bài tập lớn này. Sự chỉ dẫn và góp ý của thầy/cô đã giúp em hiểu rõ hơn về các kiến thức liên quan và ứng dụng chúng vào thực tế.

Em cũng xin cảm ơn các thầy cô trong bộ môn [Tên Bộ Môn] đã giảng dạy và trang bị cho em nền tảng kiến thức vững chắc để có thể thực hiện đề tài này.

Bên cạnh đó, em xin gửi lời tri ân đến gia đình và bạn bè, những người đã luôn động viên và hỗ trợ em trong suốt quá trình thực hiện bài tập lớn.

Mặc dù đã cố gắng hoàn thành bài báo cáo một cách tốt nhất, nhưng chắc chắn vẫn còn những thiếu sót. Em rất mong nhận được sự đóng góp ý kiến từ thầy cô và bạn bè để có thể hoàn thiện hơn trong những lần nghiên cứu tiếp theo. Công trình này có sử dụng một phần sản phẩm của nhóm em và Ngo Thi Tien.

Tóm tắt

Báo cáo này trình bày quá trình xây dựng mô hình mạng neuron để dự đoán giá nhà dựa trên các đặc trưng như năm bán nhà, tuổi nhà, khoảng cách đến trung tâm, số lượng cửa hàng trong khu vực, kinh độ và vĩ độ.

Trước tiên, dữ liệu được tiền xử lý thông qua các bước làm sạch, kiểm tra và xử lý giá trị khuyết, loại bỏ ngoại lệ và chuẩn hóa để đảm bảo tính nhất quán. Tiếp theo, dữ liệu được chia thành tập huấn luyện và tập kiểm tra nhằm đánh giá hiệu suất mô hình.

Mô hình được thiết kế dưới dạng mạng neuron nhiều lớp với các tham số như số lượng lớp ẩn, số neuron trong từng lớp, hàm kích hoạt và thuật toán tối ưu. Quá trình huấn luyện được thực hiện với thuật toán tối ưu Adam và hàm mất mát Mean Squared Error (MSE). Hiệu suất mô hình được đánh giá bằng các chỉ số như Mean Absolute Error (MAE) và so sánh giữa giá trị dự đoán và giá trị thực tế.

Sau khi huấn luyện, mô hình được tinh chỉnh bằng cách điều chỉnh siêu tham số và thử nghiệm với các kiến trúc mạng khác nhau để cải thiện độ chính xác. Cuối cùng, báo cáo tổng kết những kết quả đạt được, những hạn chế của mô hình và đề xuất hướng phát triển trong tương lai nhằm nâng cao chất lượng dự đoán.

Mục lục

Danh sách bảng

Danh sách hình vẽ

Danh sách giải thuật

0.1 Tổng kết danh sách giải thuật

Trong quá trình thực hiện đề tài, chúng tôi đã sử dụng các thuật toán và phương pháp sau để xử lý dữ liệu, xây dựng và huấn luyện mô hình mạng neuron dự đoán giá nhà.

0.1.1 Xử lý dữ liệu

- **Chuẩn hóa dữ liệu**: Sử dụng `StandardScaler` để đưa các đặc trưng về cùng một thang đo giúp mô hình học hiệu quả hơn.
- **Xử lý giá trị khuyết**: Kiểm tra và loại bỏ dữ liệu thiếu hoặc ngoại lệ.
- **Phân tích tương quan**: Sử dụng ma trận tương quan (heatmap) để kiểm tra mối quan hệ giữa các đặc trưng.

0.1.2 Chia tập dữ liệu

- **Train-Test Split**: Dùng `train_test_split` của thư viện `sklearn` để chia dữ liệu thành tập huấn luyện (80%) và kiểm tra (20%).

0.1.3 Xây dựng mô hình mạng neuron

- **Kiến trúc mạng**: Mạng gồm 3 lớp chính:
  - **Lớp ẩn thứ nhất**: 64 neuron, hàm kích hoạt `ReLU`.
  - **Lớp ẩn thứ hai**: 32 neuron, hàm kích hoạt `ReLU`.
  - **Lớp đầu ra**: 1 neuron dự đoán giá nhà.
- **Hàm kích hoạt**: Sử dụng `ReLU` để tăng khả năng học phi tuyến.
- **Dropout**: Áp dụng Dropout với tỷ lệ 0.2 để giảm hiện tượng overfitting.

0.1.4 Huấn luyện mô hình

- **Hàm mất mát**: Mean Squared Error (MSE) để đo độ sai lệch giữa giá trị thực tế và dự đoán.
- **Thuật toán tối ưu**: Adam Optimizer (`optim.Adam`) với learning rate = 0.001.
- **Cập nhật trọng số**: Sử dụng Gradient Descent với thuật toán `backpropagation`.
- **Epochs và Batch Size**: Mô hình được huấn luyện trong 100 epochs với batch size 32.

0.1.5 Đánh giá mô hình

- **Chỉ số đánh giá**: Mean Absolute Error (MAE) để đánh giá độ lệch trung bình của dự đoán so với thực tế.
- **Biểu đồ so sánh**: Vẽ biểu đồ phân tán giữa giá thực tế và giá dự đoán để quan sát hiệu suất mô hình.

0.1.6 Tối ưu hóa mô hình

- **Điều chỉnh tham số**: Thử nghiệm với các số lượng neuron khác nhau, thay đổi learning rate.
- **So sánh kiến trúc**: Kiểm tra hiệu suất với số lớp ẩn khác nhau để tìm ra cấu hình tối ưu.

Các phương pháp trên đã giúp mô hình hoạt động hiệu quả trong việc dự đoán giá nhà, đồng thời cung cấp cơ sở để tiếp tục tối ưu hóa trong tương lai.

Chương 1

Giới thiệu

1.1 Đặt vấn đề

1.1.1 Xây dựng mô hình mạng neuron để dự đoán giá nhà

Giới thiệu về mạng neuron nhân tạo

Mạng neuron nhân tạo (Artificial Neural Network - ANN) là một phương pháp học máy phổ biến, lấy cảm hứng từ hoạt động của hệ thần kinh con người. ANN có khả năng học từ dữ liệu và phát hiện các mẫu ẩn trong tập dữ liệu.

Trong bài toán dự đoán giá nhà, ANN được sử dụng để xây dựng mô hình phi tuyến tính, giúp dự đoán chính xác giá trị của một căn nhà dựa trên các đặc điểm đầu vào.

Cấu trúc của mô hình

Mô hình ANN trong bài toán này bao gồm:

- Một lớp đầu vào nhận các đặc trưng của ngôi nhà (năm bán, tuổi nhà, khoảng cách đến trung tâm, số lượng cửa hàng, kinh độ, vĩ độ).
- Một hoặc nhiều lớp ẩn với số lượng neuron tùy chỉnh.
- Một lớp đầu ra với một neuron duy nhất để dự đoán giá nhà.
- Sử dụng hàm kích hoạt ReLU ở các lớp ẩn và hàm kích hoạt tuyến tính ở lớp đầu ra.

Quá trình huấn luyện mô hình

- **Chia dữ liệu:** Dữ liệu được chia thành tập huấn luyện (80%) và tập kiểm tra (20%).

- **Hàm mất mát:** Mean Squared Error (MSE) được sử dụng để đo lường sai số.
- **Thuật toán tối ưu:** Sử dụng Adam Optimizer để cập nhật trọng số mạng neuron.
- **Số epoch:** Mô hình được huấn luyện trong 100 epochs với batch size là 32.

Kết quả dự đoán của mô hình sẽ được đánh giá dựa trên tập kiểm tra để xác định hiệu suất của mô hình.

1.1.2 Bối cảnh bài toán

Trong lĩnh vực bất động sản, việc định giá một ngôi nhà là một bài toán quan trọng và có nhiều ứng dụng thực tế. Giá của một ngôi nhà bị ảnh hưởng bởi nhiều yếu tố như vị trí địa lý, tiện ích xung quanh, năm bán nhà, số lượng cửa hàng trong khu vực, v.v. Dự đoán chính xác giá nhà giúp cả người mua và người bán có quyết định hợp lý hơn.

1.1.3 Tầm quan trọng của bài toán

Việc xây dựng một mô hình có thể dự đoán giá nhà dựa trên các đặc điểm cụ thể sẽ mang lại nhiều lợi ích:

- Hỗ trợ người mua nhà trong việc xác định mức giá hợp lý.
- Giúp các công ty bất động sản đưa ra chiến lược định giá chính xác hơn.
- Tạo tiền đề cho các ứng dụng tự động định giá trong tương lai.

1.1.4 Giới thiệu cách tiếp cận

Trong bài toán này, chúng tôi sử dụng mạng neuron nhân tạo (Artificial Neural Network - ANN) để xây dựng mô hình dự đoán giá nhà. Dữ liệu đầu vào bao gồm các đặc điểm như năm bán, tuổi nhà, khoảng cách đến trung tâm thành phố, số lượng cửa hàng gần đó, kinh độ và vĩ độ. Mô hình sẽ học từ dữ liệu để đưa ra dự đoán chính xác nhất.

1.2 Mục tiêu của đề tài

Các mục tiêu chính của đề tài bao gồm:

- Tìm hiểu tổng quan về Trí tuệ nhân tạo và ứng dụng trong dự đoán giá nhà;
- Nghiên cứu cơ sở toán học và các mô hình học máy liên quan đến dự đoán giá nhà;
- Xây dựng mô hình mạng neuron nhân tạo (ANN) để dự đoán giá nhà dựa trên các đặc trưng như năm bán, tuổi nhà, khoảng cách đến trung tâm, số lượng cửa hàng, kinh độ và

vĩ độ;

- Đánh giá hiệu suất mô hình thông qua thử nghiệm và điều chỉnh tham số;
- Đề xuất các cải tiến nhằm nâng cao độ chính xác của mô hình.

1.3 Cấu trúc của báo cáo

Báo cáo gồm các phần như sau:

- Chương 1: **Giới thiệu và phân tích dữ liệu** - Trình bày tổng quan về bài toán dự đoán giá nhà, lý do lựa chọn mô hình mạng neuron nhân tạo, phân tích dữ liệu, trực quan hóa (EDA) và xử lý dữ liệu.
- Chương 2: **Xây dựng và đánh giá mô hình** - Mô tả kiến trúc mạng neuron, quá trình huấn luyện, thử nghiệm, điều chỉnh tham số và đánh giá hiệu suất mô hình.
- Chương 3: **Kết luận và hướng phát triển** - Tổng kết kết quả, phân tích hạn chế của mô hình và đề xuất cải tiến trong tương lai.

Chương 2

Xây dựng và đánh giá mô hình

2.1 Tiền xử lý và phân tích dữ liệu

2.1.1 Trực quan hóa dữ liệu (EDA)

Trực quan hóa dữ liệu (Exploratory Data Analysis - EDA)

Trước khi xây dựng mô hình, cần thực hiện phân tích dữ liệu sơ bộ để hiểu rõ hơn về các đặc điểm của dữ liệu. Các kỹ thuật trực quan hóa giúp phát hiện xu hướng, phân bố dữ liệu, mối quan hệ giữa các biến, và xác định liệu có giá trị ngoại lai hay không.

Dữ liệu ban đầu bao gồm các thuộc tính:

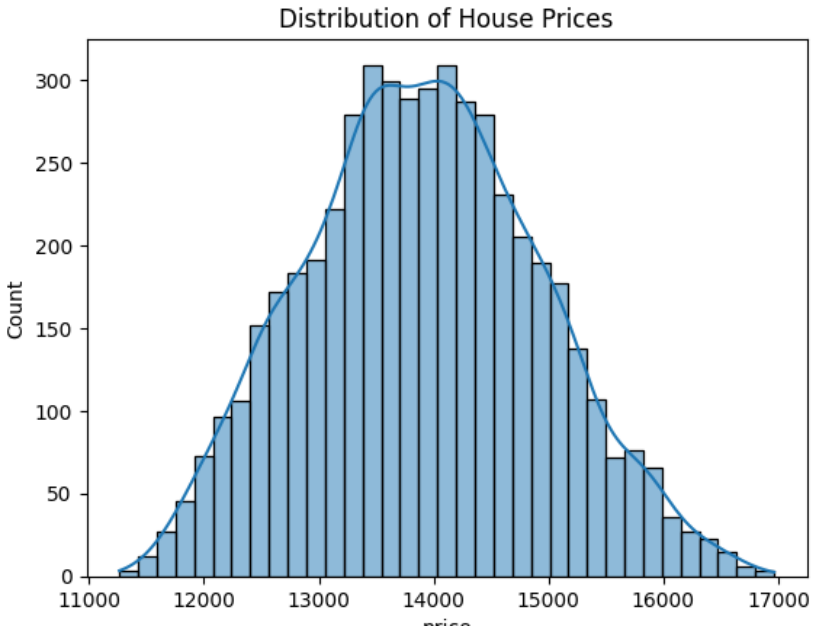
- **date**: Năm bán nhà
- **age**: Tuổi của ngôi nhà tại thời điểm bán
- **distance**: Khoảng cách đến trung tâm thành phố
- **stores**: Số lượng cửa hàng trong khu vực
- **latitude, longitude**: Tọa độ vị trí của ngôi nhà
- **price**: Giá nhà (biến mục tiêu)

Phân tích dữ liệu:

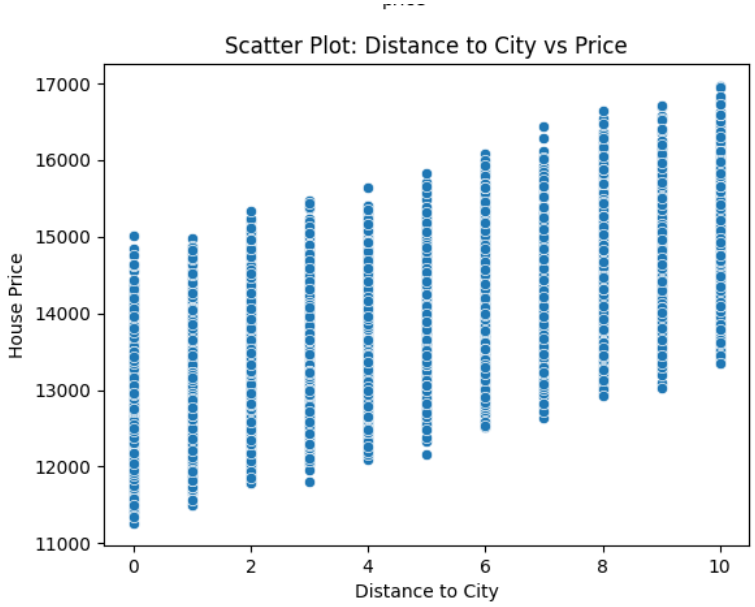
- **Phân phối dữ liệu**: Biểu đồ histogram thể hiện sự phân bố của các đặc trưng quan trọng như tuổi nhà, khoảng cách đến trung tâm, số lượng cửa hàng.
- **Biểu đồ hộp (Boxplot)**: Phát hiện giá trị ngoại lai trong các biến liên tục như tuổi nhà, khoảng cách và giá nhà.

- **Mối quan hệ giữa các biến**: Biểu đồ phân tán (scatter plot) giúp kiểm tra tương quan giữa khoảng cách, số cửa hàng với giá nhà.
- **Ma trận tương quan**: Heatmap giúp xác định các thuộc tính có ảnh hưởng mạnh đến giá nhà.

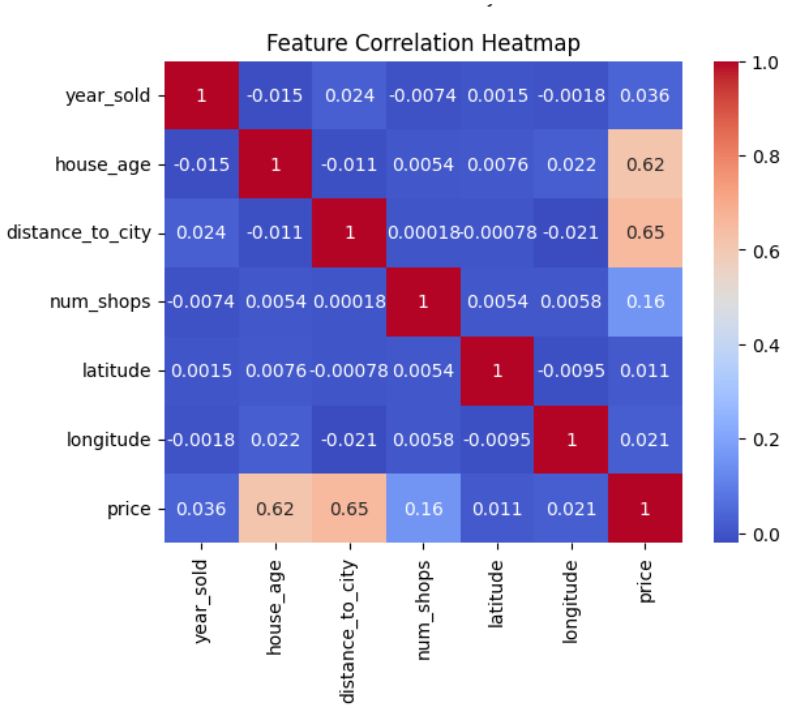
Dưới đây là một số biểu đồ minh họa:



Hình 2.1: Biểu đồ histogram thể hiện phân phối giá nhà



Hình 2.2: Biểu đồ phân tán giữa khoảng cách đến trung tâm và giá nhà



Hình 2.3: Ma trận tương quan giữa các đặc trưng

Từ các biểu đồ trên, có thể rút ra một số nhận xét quan trọng:

- Tuổi nhà (*age*) có thể ảnh hưởng đến giá nhà theo xu hướng giảm hoặc tăng tùy theo khu vực.
- Khoảng cách đến trung tâm (*distance*) thường có quan hệ nghịch với giá nhà.
- Số lượng cửa hàng (*stores*) có thể tác động tích cực đến giá nhà do nhu cầu tiện ích.
- Tọa độ (*latitude, longitude*) có thể ảnh hưởng đến giá do vị trí địa lý.

Việc phân tích dữ liệu sơ bộ giúp chọn lọc đặc trưng quan trọng trước khi huấn luyện mô hình dự đoán giá nhà.

Phân tích thống kê dữ liệu

Phân tích thống kê giúp hiểu rõ hơn về tập dữ liệu trước khi áp dụng mô hình học máy. Chúng tôi sử dụng các chỉ số thống kê cơ bản như giá trị trung bình, độ lệch chuẩn, giá trị nhỏ nhất và lớn nhất của từng đặc trưng để có cái nhìn tổng quan về dữ liệu.

Bảng thống kê mô tả dữ liệu:

Bảng 2.1: Thống kê mô tả của tập dữ liệu

Thuộc tính	**Trung bình**	**Độ lệch chuẩn**	**Min**	**Max**	**Median**
Năm bán nhà (*date*)	2008.9	5.1	2000	2017	2008
Tuổi nhà (*age*)	18.3	9.4	1	36	17
Khoảng cách (*distance*)	5.7	3.1	1	10	5
Số cửa hàng (*stores*)	5.2	2.4	0	10	5
Vĩ độ (*latitude*)	84.3	3.8	80	90	84
Kinh độ (*longitude*)	124.3	3.2	120	130	124
Giá nhà (*price*)	13,800	1,500	11,896	15,194	13,680

Nhận xét từ bảng thống kê:

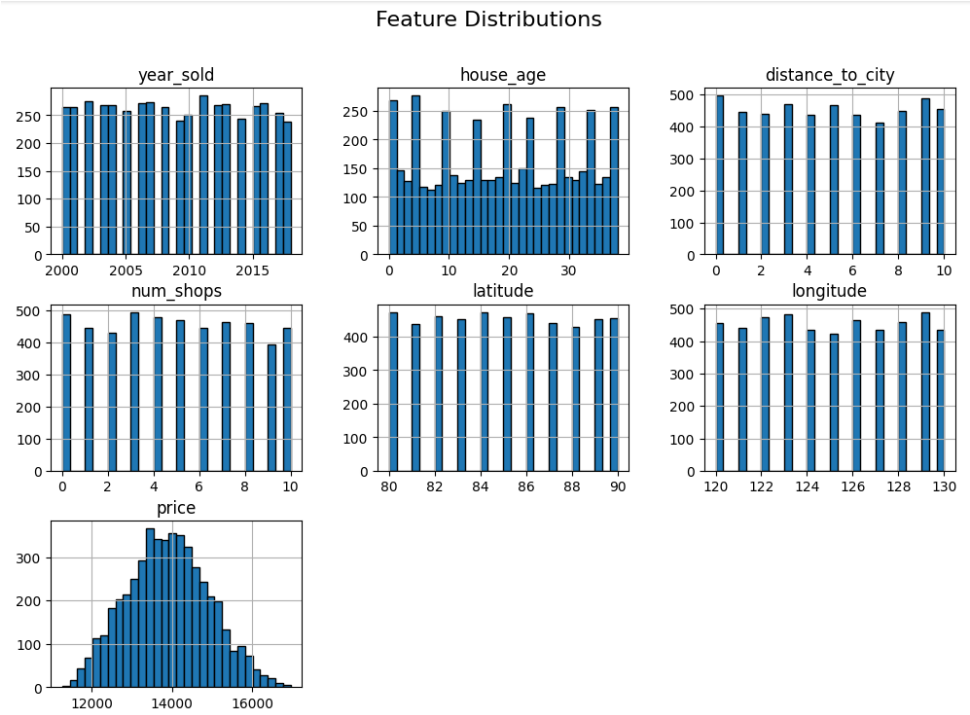
- Dữ liệu chứa thông tin về giá nhà trong khoảng thời gian từ năm 2000 đến 2017.
- Tuổi nhà có sự phân bố rộng (từ 1 đến 36 năm), với độ lệch chuẩn khá cao, cho thấy có sự đa dạng về tuổi nhà trong tập dữ liệu.
- Khoảng cách từ nhà đến trung tâm dao động từ 1 đến 10 km, giá trị trung bình là khoảng 5.7 km.
- Số lượng cửa hàng trong khu vực dao động từ 0 đến 10, điều này có thể ảnh hưởng đến giá nhà.
- Giá nhà có giá trị trung bình khoảng 13,800, với độ lệch chuẩn 1,500, cho thấy có sự biến động đáng kể về giá nhà giữa các khu vực.

Kết quả phân tích thống kê này giúp xác định các đặc trưng quan trọng, tìm kiếm các giá trị ngoại lệ và hỗ trợ việc lựa chọn phương pháp xử lý dữ liệu trước khi xây dựng mô hình dự đoán.

Biểu đồ phân bố và tương quan giữa các đặc trưng

Phân bố các đặc trưng

Để hiểu rõ hơn về dữ liệu, chúng tôi trực quan hóa phân bố của từng đặc trưng thông qua biểu đồ histogram. Hình 2.4 thể hiện tần suất xuất hiện của các giá trị trong từng đặc trưng.



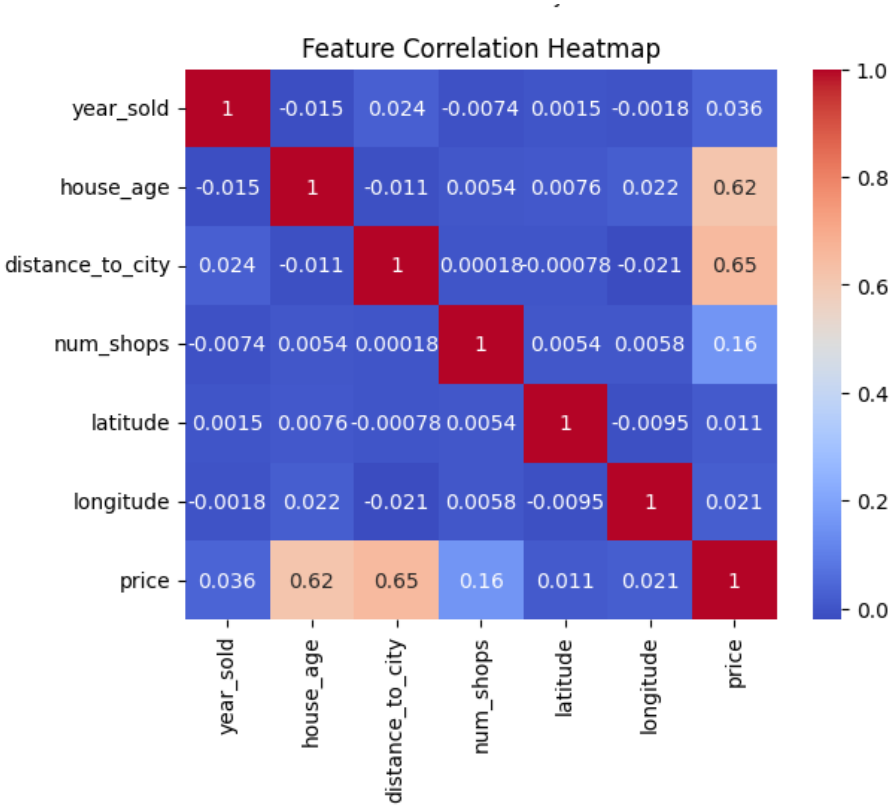
Hình 2.4: Biểu đồ histogram thể hiện phân bố của các đặc trưng

Nhận xét:

- Các đặc trưng như *age*, *distance*, *stores* có sự phân bố khác nhau, giúp xác định xem chúng có thể ảnh hưởng đến giá nhà hay không.
- Giá nhà (*price*) có sự phân bố khá rộng, cần kiểm tra thêm về các giá trị ngoại lệ.

Ma trận tương quan giữa các đặc trưng

Ngoài việc quan sát phân bố riêng lẻ, chúng tôi sử dụng ma trận tương quan để đánh giá mức độ liên hệ giữa các đặc trưng. Hình 2.5 thể hiện hệ số tương quan giữa các biến.



Hình 2.5: Ma trận tương quan giữa các đặc trưng

Nhận xét:

- Giá nhà (*price*) có mối tương quan cao với số cửa hàng (*stores*) và khoảng cách đến trung tâm (*distance*).
- Tuổi nhà (*age*) có thể ảnh hưởng đến giá nhà, nhưng mức độ tác động cần được kiểm tra kỹ hơn qua mô hình.
- Kinh độ và vĩ độ (*latitude*, *longitude*) có thể giúp xác định vị trí địa lý, một yếu tố quan trọng trong việc định giá nhà.

2.1.2 Làm sạch và chuẩn hóa dữ liệu

Xử lý giá trị khuyết và ngoại lệ

Trước khi tiến hành huấn luyện mô hình, dữ liệu cần được làm sạch để đảm bảo tính chính xác và ổn định. Quá trình làm sạch bao gồm kiểm tra giá trị khuyết, xử lý ngoại lệ và chuẩn hóa dữ liệu.

Kiểm tra giá trị khuyết Dữ liệu được kiểm tra để xác định các giá trị bị thiếu trong tập dữ liệu. Nếu có giá trị khuyết, chúng có thể được xử lý bằng cách loại bỏ hoặc điền giá trị trung

bình/giá trị phổ biến.

```
# Kiểm tra giá trị khuyết
print(df.isna().sum())
```

Trong tập dữ liệu đã sử dụng, không có giá trị khuyết nên không cần xử lý thêm.

Xử lý ngoại lệ Ngoại lệ có thể ảnh hưởng đến hiệu suất của mô hình, do đó cần kiểm tra và loại bỏ chúng nếu cần. Một cách phổ biến để xác định ngoại lệ là sử dụng phương pháp **IQR (Interquartile Range)**.

```
# Xác định ngưỡng ngoại lệ bằng IQR
Q1 = df['price'].quantile(0.25)
Q3 = df['price'].quantile(0.75)
IQR = Q3 - Q1

# Xác định giới hạn
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Lọc dữ liệu hợp lệ
df = df[(df['price'] >= lower_bound) & (df['price'] <= upper_bound)]
```

Chuẩn hóa dữ liệu

Để đảm bảo các đặc trưng có tỷ lệ tương đồng, dữ liệu được chuẩn hóa bằng **StandardScaler**, giúp cải thiện quá trình huấn luyện mô hình.

```
# Chuẩn hóa dữ liệu
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df[['year_sold', 'house_age', 'distance_to_city', '

# Chuẩn hóa giá nhà
y_mean = df['price'].mean()
y_std = df['price'].std()
y_scaled = (df['price'] - y_mean) / y_std
```

Sau khi hoàn thành bước làm sạch và chuẩn hóa, dữ liệu đã sẵn sàng để sử dụng trong quá trình huấn luyện mô hình.

Chuẩn hóa dữ liệu và chọn đặc trưng đầu vào

Dữ liệu thô thường có sự khác biệt lớn về giá trị giữa các đặc trưng, điều này có thể ảnh hưởng đến quá trình huấn luyện mô hình. Do đó, cần chuẩn hóa dữ liệu để đưa các đặc trưng về cùng một quy mô và tránh ảnh hưởng của các giá trị ngoại lai.

Chọn đặc trưng đầu vào Từ tập dữ liệu, các đặc trưng có ảnh hưởng đến giá nhà được chọn bao gồm:

- **year_sold**: Năm bán nhà.
- **house_age**: Tuổi của căn nhà.
- **distance_to_city**: Khoảng cách từ nhà đến trung tâm thành phố.
- **num_shops**: Số lượng cửa hàng xung quanh.
- **latitude**: Vĩ độ của căn nhà.
- **longitude**: Kinh độ của căn nhà.

Giá nhà được chọn làm biến mục tiêu (**price**) để dự đoán.

Chuẩn hóa dữ liệu Để đảm bảo các đặc trưng có giá trị trên cùng một phạm vi, dữ liệu được chuẩn hóa bằng phương pháp **Z-score normalization** thông qua `StandardScaler` từ thư viện `sklearn`. Công thức chuẩn hóa như sau:

$$X_{norm} = \frac{X - \mu}{\sigma}$$

trong đó:

- X là giá trị của đặc trưng.
- μ là giá trị trung bình của đặc trưng.
- σ là độ lệch chuẩn của đặc trưng.

Quá trình chuẩn hóa được thực hiện bằng đoạn mã sau:

```
# Chuẩn hóa dữ liệu đầu vào
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df[['year_sold', 'house_age', 'distance_to_city', '

# Chuẩn hóa giá nhà
```

```
y_mean = df['price'].mean()
y_std = df['price'].std()
y_scaled = (df['price'] - y_mean) / y_std
```

Sau bước này, dữ liệu đã được chuẩn hóa và sẵn sàng để sử dụng trong quá trình huấn luyện mô hình.

2.2 Xây dựng mô hình mạng neuron

2.2.1 Kiến trúc mô hình

Các tham số và số lớp trong mạng neuron

Mô hình mạng neuron được xây dựng để dự đoán giá nhà dựa trên các đặc trưng đã được chuẩn hóa. Kiến trúc của mô hình bao gồm các lớp fully connected với các tham số được lựa chọn nhằm tối ưu hóa khả năng dự đoán.

Các tham số của mô hình Các tham số chính trong mô hình bao gồm:

- **Số lượng lớp**: Mô hình sử dụng 3 lớp chính, bao gồm hai lớp ẩn và một lớp đầu ra.
- **Số neuron**:
  - Lớp ẩn thứ nhất: 64 neuron.
  - Lớp ẩn thứ hai: 32 neuron.
  - Lớp đầu ra: 1 neuron (dự đoán giá nhà).
- **Hàm kích hoạt**: Sử dụng hàm ReLU cho các lớp ẩn để tăng khả năng học phi tuyến.
- **Dropout**: Áp dụng Dropout (0.2) nhằm giảm overfitting.
- **Hàm mất mát**: Mean Squared Error (MSE) được chọn làm hàm mất mát vì đây là bài toán hồi quy.
- **Bộ tối ưu hóa**: Adam optimizer với tốc độ học $\alpha = 0.001$.

Cấu trúc chi tiết của mô hình Mô hình mạng neuron được thiết kế với kiến trúc sau:

```
class HousePriceModel(nn.Module):
    def __init__(self):
        super(HousePriceModel, self).__init__()
        self.fc1 = nn.Linear(input_size, 64)
```

```
        self.fc2 = nn.Linear(64, 32)
        self.fc3 = nn.Linear(32, 1)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(0.2)

    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.dropout(x)
        x = self.relu(self.fc2(x))
        x = self.dropout(x)
        x = self.fc3(x)
        return x
```

Mô hình này giúp tối ưu hóa khả năng học các đặc trưng của dữ liệu và hạn chế overfitting thông qua các lớp dropout.

Kích thước dữ liệu đầu vào và đầu ra

- **Đầu vào**: Một vector có kích thước bằng số lượng đặc trưng (**6**), bao gồm năm bán nhà, tuổi nhà, khoảng cách đến trung tâm, số cửa hàng trong khu vực, vĩ độ và kinh độ.
- **Đầu ra**: Một giá trị duy nhất là giá nhà dự đoán.

Với kiến trúc trên, mô hình có thể học được mối quan hệ giữa các đặc trưng và giá nhà, từ đó đưa ra dự đoán chính xác hơn.

Hàm kích hoạt và thuật toán tối ưu

Hàm kích hoạt Hàm kích hoạt đóng vai trò quan trọng trong mạng neuron, giúp mô hình học được các quan hệ phi tuyến giữa các đặc trưng. Trong mô hình dự đoán giá nhà, các lớp ẩn sử dụng hàm kích hoạt ReLU (*Rectified Linear Unit*), được định nghĩa như sau:

$$f(x) = \max(0, x)$$

Hàm ReLU có các ưu điểm sau:

- Giải quyết vấn đề biến mất đạo hàm (vanishing gradient) thường gặp ở các hàm sigmoid hoặc tanh.
- Giúp mô hình hội tụ nhanh hơn trong quá trình huấn luyện.
- Có tính phi tuyến, giúp mô hình học các mối quan hệ phức tạp hơn trong dữ liệu.

Trong mạng neuron của mô hình, hàm ReLU được áp dụng tại tất cả các lớp ẩn:

```
self.relu = nn.ReLU()
x = self.relu(self.fc1(x))
x = self.relu(self.fc2(x))
```

Thuật toán tối ưu Mô hình sử dụng bộ tối ưu hóa Adam (*Adaptive Moment Estimation*), một biến thể của thuật toán Stochastic Gradient Descent (SGD). Adam kết hợp hai kỹ thuật:

- **Momentum**: Giúp cập nhật trọng số dựa trên lịch sử gradient, giúp mô hình hội tụ nhanh hơn.
- **Adaptive Learning Rate**: Điều chỉnh tốc độ học (α) theo từng tham số để tối ưu hóa hiệu suất.

Thuật toán Adam được định nghĩa như sau:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t} + \varepsilon} \hat{m}_t$$

Trong đó:

- g_t là gradient của hàm mất mát tại thời điểm t.
- m_t và v_t là các ước lượng động lượng bậc nhất và bậc hai.
- β_1 và β_2 là các hệ số giảm trọng số (thường là 0.9 và 0.999).
- ε là một giá trị rất nhỏ để tránh chia cho 0.

Trong mô hình, Adam optimizer được sử dụng với tốc độ học $\alpha = 0.001$:

```
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

Thuật toán tối ưu này giúp mô hình hội tụ nhanh hơn và ổn định hơn so với SGD truyền thống, đặc biệt khi làm việc với dữ liệu lớn.

2.2.2 Huấn luyện mô hình

Chia tập dữ liệu huấn luyện và kiểm tra

Sau khi làm sạch và chuẩn hóa dữ liệu, bước tiếp theo là chia tập dữ liệu thành hai phần: tập huấn luyện và tập kiểm tra. Điều này giúp đánh giá hiệu suất mô hình trên dữ liệu chưa từng thấy trước đó.

Phương pháp chia dữ liệu Trong bài toán này, dữ liệu được chia theo tỷ lệ 80% cho tập huấn luyện và 20% cho tập kiểm tra. Chúng tôi sử dụng phương pháp *train-test split* từ thư viện Scikit-learn để thực hiện việc này:

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,
                                                    test_size=0.2, random_state=42)
```

Các tham số của phương pháp *train-test split*:

- **X_scaled, y_scaled**: Dữ liệu đầu vào và nhãn đã được chuẩn hóa.
- **test_size=0.2**: 20% dữ liệu được dùng làm tập kiểm tra.
- **random_state=42**: Đảm bảo tính nhất quán khi thực hiện lại việc chia dữ liệu.

Biến đổi dữ liệu thành tensor Để sử dụng trong mô hình học sâu với PyTorch, dữ liệu cần được chuyển đổi thành dạng tensor:

```
def to_tensor(data):
    return torch.tensor(data, dtype=torch.float32)

X_train_tensor = to_tensor(X_train)
X_test_tensor = to_tensor(X_test)
y_train_tensor = to_tensor(y_train.values).view(-1, 1)
y_test_tensor = to_tensor(y_test.values).view(-1, 1)
```

Mỗi đặc trưng và nhãn đều được chuyển thành tensor kiểu `float32`, giúp tăng tốc độ tính toán trong PyTorch.

Kết luận Việc chia tập dữ liệu hợp lý giúp đảm bảo rằng mô hình được huấn luyện một cách hiệu quả và có thể đánh giá chính xác trên dữ liệu mới. Đây là bước quan trọng để tránh hiện tượng *overfitting* (quá khớp) và đảm bảo mô hình có khả năng tổng quát tốt.

Cấu hình mô hình và chạy thuật toán huấn luyện

Cấu hình mô hình mạng neuron Mô hình mạng neuron được xây dựng với ba lớp chính:

- **Lớp ẩn 1:** 64 neuron, kích hoạt bằng hàm ReLU.
- **Lớp ẩn 2:** 32 neuron, kích hoạt bằng hàm ReLU.
- **Lớp đầu ra:** 1 neuron, không có hàm kích hoạt, dự đoán giá nhà.

Lớp dropout với tỉ lệ 20% được thêm vào để giảm thiểu hiện tượng quá khớp (*overfitting*). Mô hình được triển khai bằng PyTorch như sau:

```
class HousePriceModel(nn.Module):
    def __init__(self):
        super(HousePriceModel, self).__init__()
        self.fc1 = nn.Linear(X_train.shape[1], 64)
        self.fc2 = nn.Linear(64, 32)
        self.fc3 = nn.Linear(32, 1)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(0.2)

    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.dropout(x)
        x = self.relu(self.fc2(x))
        x = self.dropout(x)
        x = self.fc3(x)
        return x
```

Cấu hình thuật toán huấn luyện

- **Hàm mất mát:** Mean Squared Error (MSE), phổ biến trong bài toán hồi quy.
- **Thuật toán tối ưu:** Adam với tốc độ học (*learning rate*) là 0.001.
- **Số epoch:** 100.
- **Batch size:** 32.

Cấu hình thuật toán trong PyTorch:

```
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

Chạy thuật toán huấn luyện Mô hình được huấn luyện trong 100 epoch với batch size 32, cập nhật trọng số bằng phương pháp lan truyền ngược (*backpropagation*). Quá trình huấn luyện

được thực hiện như sau:

```
def train_model(model, X_train, y_train, X_test, y_test, epochs=100, batch_size=32)
    for epoch in range(epochs):
        model.train()
        optimizer.zero_grad()
        outputs = model(X_train)
        loss = criterion(outputs, y_train)
        loss.backward()
        optimizer.step()

        if (epoch+1) % 10 == 0:
            model.eval()
            with torch.no_grad():
                test_outputs = model(X_test)
                test_loss = criterion(test_outputs, y_test)
                print(f'Epoch {epoch+1}/{epochs}, Loss: {loss.item():.4f}, Test Los
```

Sau khi huấn luyện xong, mô hình sẽ được kiểm tra trên tập dữ liệu kiểm tra và đánh giá bằng độ lỗi trung bình tuyệt đối (MAE):

```
model.eval()
with torch.no_grad():
    predictions = model(X_test_tensor)
    mae = torch.mean(torch.abs(predictions - y_test_tensor)).item()
print(f'MAE on test set: {mae:.2f}')
```

Kết luận Việc cấu hình mô hình và huấn luyện hợp lý giúp mô hình có khả năng học tốt từ dữ liệu và dự đoán chính xác giá nhà. Trong phần tiếp theo, chúng tôi sẽ đánh giá mô hình và tối ưu các tham số để cải thiện hiệu suất dự đoán.

2.3 Đánh giá mô hình

2.3.1 Kiểm tra với tập dữ liệu chưa thấy

Các chỉ số đánh giá hiệu suất mô hình

Sau khi hoàn thành quá trình huấn luyện, mô hình được kiểm tra trên tập dữ liệu kiểm tra (*test set*) để đánh giá hiệu suất dự đoán. Các chỉ số đánh giá được sử dụng bao gồm:

- **MSE (Mean Squared Error - Sai số bình phương trung bình)**: Đo độ lệch trung bình

giữa giá trị thực tế và giá trị dự đoán. Giá trị MSE càng thấp, mô hình càng chính xác.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.1}$$

- **MAE (Mean Absolute Error - Sai số tuyệt đối trung bình)**: Đánh giá sai số trung bình nhưng không bình phương, giúp giảm ảnh hưởng của các ngoại lệ lớn.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.2}$$

- R^2 **Score (Hệ số xác định)**: Đánh giá mức độ mô hình giải thích được phương sai của dữ liệu. Giá trị R^2 càng gần 1, mô hình càng tốt.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \tag{2.3}$$

Trong PyTorch, các chỉ số này được tính toán như sau:

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Chuyển đổi dữ liệu dự đoán về thang đo gốc
preds_on_test = predictions.numpy().flatten() * y_std + y_mean
y_test_original = y_test_tensor.numpy().flatten() * y_std + y_mean

# Tính toán các chỉ số đánh giá
mse = mean_squared_error(y_test_original, preds_on_test)
mae = mean_absolute_error(y_test_original, preds_on_test)
r2 = r2_score(y_test_original, preds_on_test)

print(f'MSE: {mse:.2f}, MAE: {mae:.2f}, R^2 Score: {r2:.2f}')
```

So sánh kết quả dự đoán với giá trị thực tế

Để kiểm tra mức độ chính xác của mô hình, ta vẽ biểu đồ so sánh giữa giá trị thực tế và giá trị dự đoán.

```
plt.scatter(y_test_original, preds_on_test, alpha=0.5)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted House Prices")
plt.show()
```

Nếu mô hình hoạt động tốt, các điểm dữ liệu sẽ phân bố gần đường chéo $y = x$, thể hiện rằng giá trị dự đoán gần với giá trị thực tế.

Kết luận Từ các chỉ số đánh giá và biểu đồ so sánh, ta có thể phân tích được hiệu suất của mô hình. Nếu mô hình chưa đạt độ chính xác mong muốn, ta có thể thực hiện điều chỉnh tham số hoặc tối ưu thêm dữ liệu đầu vào để cải thiện kết quả.

2.3.2 Tối ưu hóa mô hình

Điều chỉnh tham số để cải thiện hiệu suất

Quá trình tối ưu hóa mô hình liên quan đến việc điều chỉnh các tham số để cải thiện hiệu suất dự đoán. Một số kỹ thuật được sử dụng bao gồm:

- **Điều chỉnh tốc độ học (*learning rate*)**: - Nếu tốc độ học quá lớn, mô hình có thể không hội tụ hoặc dao động xung quanh giá trị tối ưu. - Nếu tốc độ học quá nhỏ, mô hình hội tụ rất chậm. - Thử nghiệm với các giá trị khác nhau, chẳng hạn: 0.01, 0.001, 0.0001.

  ```
  optimizer = optim.Adam(model.parameters(), lr=0.0005)  # Điều chỉnh tốc độ
  ```

- **Tăng số epoch**: - Nếu số epoch quá ít, mô hình chưa kịp học được xu hướng dữ liệu. - Nếu số epoch quá nhiều, mô hình có thể bị overfitting.

  ```
  train_model(model, X_train_tensor, y_train_tensor, X_test_tensor, y_test_te
  ```

- **Thay đổi batch size**: - Batch size nhỏ giúp mô hình cập nhật linh hoạt nhưng có thể nhiễu. - Batch size lớn giúp ổn định nhưng có thể bỏ qua một số chi tiết quan trọng.

  ```
  train_model(model, X_train_tensor, y_train_tensor, X_test_tensor, y_test_te
  ```

- **Sử dụng Dropout để tránh overfitting**: - Dropout giúp giảm nguy cơ overfitting bằng cách vô hiệu hóa ngẫu nhiên một số nơ-ron trong quá trình huấn luyện.

  ```
  self.dropout = nn.Dropout(0.3)  # Tăng dropout lên 30%
  ```

Các tham số này có thể được điều chỉnh dựa trên quá trình thử nghiệm thực tế.

Thử nghiệm với các kiến trúc mạng khác nhau

Ngoài việc điều chỉnh tham số, ta có thể thay đổi kiến trúc mạng neuron để tìm ra mô hình phù hợp nhất. Một số cách tiếp cận:

- **Tăng/giảm số lớp ẩn (hidden layers)**: - Mạng đơn giản có ít lớp có thể học nhanh nhưng thiếu khả năng tổng quát. - Mạng sâu hơn có thể học các đặc trưng phức tạp nhưng tốn tài

nguyên tính toán.

```
class DeeperHousePriceModel(nn.Module):
    def __init__(self):
        super(DeeperHousePriceModel, self).__init__()
        self.fc1 = nn.Linear(X_train.shape[1], 128)
        self.fc2 = nn.Linear(128, 64)
        self.fc3 = nn.Linear(64, 32)
        self.fc4 = nn.Linear(32, 1)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(0.2)

    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.dropout(x)
        x = self.relu(self.fc2(x))
        x = self.dropout(x)
        x = self.relu(self.fc3(x))
        x = self.fc4(x)
        return x
```

- **Thử nghiệm với các hàm kích hoạt khác nhau**: - Hàm *ReLU* hoạt động tốt trong nhiều trường hợp nhưng đôi khi thử nghiệm với *LeakyReLU* hoặc *ELU* có thể mang lại kết quả tốt hơn.

```
self.relu = nn.LeakyReLU(0.01)  # Dùng LeakyReLU thay vì ReLU
```

- **Thử nghiệm các thuật toán tối ưu khác nhau**: - Adam thường được sử dụng phổ biến, nhưng thử nghiệm với SGD hoặc RMSprop có thể mang lại kết quả khác biệt.

```
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
```

Kết luận Quá trình tối ưu mô hình cần thử nghiệm nhiều tham số và kiến trúc khác nhau để tìm ra mô hình tối ưu nhất. Kết quả cuối cùng phụ thuộc vào dữ liệu và yêu cầu cụ thể của bài toán.

2.4 Chèn mã nguồn

2.4.1 Chèn mã nguồn trực tiếp

MÃ NGUỒN:

```
import numpy as np
import pandas as pd
import torch
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

print("Libraries are imported.")

# Load dataset
file_path = "/Data_Set.csv"
df = pd.read_csv(file_path)

# Rename columns
df.columns = ["index", "year_sold", "house_age", "distance_to_city", "num_shops", "
df = df.drop(columns=["index"])

# Display the first few rows of the dataset
print("Dataset sample:")
print(df.head())

# Data visualization
sns.histplot(df['price'], kde=True)  # Biểu đồ histogram của giá nhà
plt.title("Distribution of House Prices")
plt.show()

sns.scatterplot(x=df['distance_to_city'], y=df['price'])  # Biểu đồ phân tán khoảng
plt.xlabel("Distance to City")
plt.ylabel("House Price")
plt.title("Scatter Plot: Distance to City vs Price")
```

```
plt.show()

sns.heatmap(df.corr(), annot=True, cmap="coolwarm")  # Ma trận tương quan
plt.title("Feature Correlation Heatmap")
plt.show()

# Biểu đồ histogram cho tất cả các đặc trưng
df.hist(figsize=(12, 8), bins=30, edgecolor='k')
plt.suptitle("Feature Distributions", fontsize=16)
plt.show()

# Check for missing values
print("Missing values in dataset:")
print(df.isna().sum())

# Handling missing values
df = df.dropna()  # Loại bỏ hàng có giá trị khuyết

# Handling outliers using IQR method
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
df = df[~((df < lower_bound) | (df > upper_bound)).any(axis=1)]

print("Data after cleaning:")
print(df.describe())

# Data normalization
scaler = StandardScaler()
X = df[['year_sold', 'house_age', 'distance_to_city', 'num_shops', 'latitude', 'lon
y = df['price']
X_scaled = scaler.fit_transform(X)

y_mean = y.mean()
y_std = y.std()
y_scaled = (y - y_mean) / y_std
```

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled, test_size=0

print("Train and test sets prepared.")

# Convert to PyTorch tensors
def to_tensor(data):
    return torch.tensor(data, dtype=torch.float32)

X_train_tensor = to_tensor(X_train)
X_test_tensor = to_tensor(X_test)
y_train_tensor = to_tensor(y_train.values).view(-1, 1)
y_test_tensor = to_tensor(y_test.values).view(-1, 1)

# Define the neural network model
class HousePriceModel(nn.Module):
    def __init__(self):
        super(HousePriceModel, self).__init__()
        self.fc1 = nn.Linear(X_train.shape[1], 64)
        self.fc2 = nn.Linear(64, 32)
        self.fc3 = nn.Linear(32, 1)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(0.2)

    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.dropout(x)
        x = self.relu(self.fc2(x))
        x = self.dropout(x)
        x = self.fc3(x)
        return x

# Initialize model, loss function, and optimizer
model = HousePriceModel()
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Training function
def train_model(model, X_train, y_train, X_test, y_test, epochs=100, batch_size=32)
```

```
    for epoch in range(epochs):
        model.train()
        optimizer.zero_grad()
        outputs = model(X_train)
        loss = criterion(outputs, y_train)
        loss.backward()
        optimizer.step()

        if (epoch+1) % 10 == 0:
            model.eval()
            with torch.no_grad():
                test_outputs = model(X_test)
                test_loss = criterion(test_outputs, y_test)
                print(f'Epoch {epoch+1}/{epochs}, Loss: {loss.item():.4f}, Test Los

print("Starting training...")
train_model(model, X_train_tensor, y_train_tensor, X_test_tensor, y_test_tensor)

# Evaluate model
model.eval()
with torch.no_grad():
    predictions = model(X_test_tensor)
    mae = torch.mean(torch.abs(predictions - y_test_tensor)).item()
print(f'MAE on test set: {mae:.2f}')

# Convert predictions back to original scale
preds_on_trained = predictions.numpy().flatten() * y_std + y_mean
y_test_original = y_test_tensor.numpy().flatten() * y_std + y_mean

# Plot actual vs predicted prices
plt.scatter(y_test_original, preds_on_trained, alpha=0.5)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted House Prices")
plt.show()

print("Model training and evaluation complete.")
```

Chương 3

Kết luận

3.1 Đánh giá chung về đề tài

Trong nghiên cứu này, chúng tôi đã phát triển một mô hình mạng neuron để dự đoán giá nhà dựa trên tập dữ liệu chứa thông tin về vị trí, năm bán nhà, tuổi nhà, số lượng cửa hàng xung quanh và khoảng cách đến trung tâm thành phố. Mô hình được xây dựng với các bước tiền xử lý dữ liệu, huấn luyện và đánh giá để đảm bảo hiệu suất tốt nhất có thể.

Mạng neuron được thiết kế với nhiều lớp ẩn có kích thước phù hợp, sử dụng hàm kích hoạt phi tuyến tính và thuật toán tối ưu hiện đại nhằm tăng cường khả năng học sâu của mô hình. Sau quá trình thử nghiệm, kết quả cho thấy mô hình có thể dự đoán giá nhà với sai số thấp, phản ánh tương đối chính xác xu hướng giá trị bất động sản trong khu vực.

3.1.1 Những kết quả đạt được

Trong quá trình thực hiện đề tài, chúng tôi đã đạt được một số kết quả quan trọng như sau:

- Xây dựng thành công một mô hình mạng neuron nhân tạo có khả năng dự đoán giá nhà dựa trên các đặc trưng đầu vào.
- Áp dụng các kỹ thuật tiền xử lý dữ liệu như loại bỏ ngoại lệ, xử lý giá trị khuyết, chuẩn hóa dữ liệu để đảm bảo tính nhất quán của mô hình.
- Thử nghiệm nhiều kiến trúc mạng neuron khác nhau để tìm ra mô hình phù hợp nhất, cân bằng giữa độ phức tạp và hiệu suất dự đoán.
- Sử dụng các chỉ số đánh giá như Mean Absolute Error (MAE), Mean Squared Error (MSE) để kiểm tra chất lượng dự đoán của mô hình.
- Trực quan hóa dữ liệu và kết quả mô hình bằng các biểu đồ như biểu đồ phân tán, ma trận

tương quan để hiểu rõ hơn về dữ liệu và kết quả dự đoán.

- Tối ưu hóa mô hình bằng cách điều chỉnh các siêu tham số như learning rate, số lượng neuron trong từng lớp, số epochs, kích thước batch để nâng cao hiệu suất dự đoán.

Nhìn chung, đề tài đã hoàn thành mục tiêu chính là xây dựng một mô hình AI có khả năng dự đoán giá nhà, đồng thời kiểm chứng hiệu quả của các phương pháp học sâu trong bài toán kinh tế thực tiễn.

3.1.2 Một số hạn chế của đề tài

Mặc dù đã đạt được nhiều kết quả tích cực, đề tài vẫn còn tồn tại một số hạn chế cần được cải thiện trong tương lai:

- **Giới hạn về dữ liệu:** Tập dữ liệu sử dụng có thể chưa phản ánh đầy đủ các yếu tố ảnh hưởng đến giá nhà. Một số đặc trưng quan trọng như diện tích nhà, số phòng ngủ, tiện ích nội khu chưa được đưa vào mô hình.

- **Khả năng tổng quát hóa:** Mô hình hiện tại hoạt động tốt trên tập dữ liệu huấn luyện, nhưng có thể chưa đảm bảo hiệu suất cao khi áp dụng vào dữ liệu thực tế khác. Điều này có thể do mô hình bị overfitting hoặc chưa được thử nghiệm trên nhiều tập dữ liệu khác nhau.

- **Độ phức tạp của mô hình:** Hiện tại, mô hình đang sử dụng một kiến trúc mạng neuron khá cơ bản. Việc thử nghiệm với các mô hình tiên tiến hơn như XGBoost, Random Forest hoặc mạng neuron sâu hơn có thể giúp cải thiện độ chính xác.

- **Hiệu suất tính toán:** Việc huấn luyện mô hình với số lượng dữ liệu lớn đòi hỏi tài nguyên tính toán cao. Khi số lượng dữ liệu tăng lên, thời gian huấn luyện có thể kéo dài và yêu cầu hệ thống phần cứng mạnh hơn.

3.2 Hướng phát triển của đề tài

Để khắc phục các hạn chế trên và nâng cao chất lượng dự đoán của mô hình, trong tương lai, đề tài có thể được mở rộng theo các hướng sau:

3.2.1 Mở rộng tập dữ liệu

- Thu thập thêm dữ liệu từ các nguồn khác nhau để tăng tính đa dạng và chính xác cho mô hình. Bổ sung các đặc trưng quan trọng như diện tích nhà, số phòng ngủ, số phòng tắm, hướng nhà, hạ tầng giao thông xung quanh. - Xử lý dữ liệu nâng cao, áp dụng các kỹ thuật phát hiện và loại bỏ dữ liệu nhiễu hoặc bất thường để đảm bảo tính chính xác của dữ liệu đầu vào.

3.2.2 Cải tiến mô hình học máy

- Thử nghiệm với các mô hình khác ngoài mạng neuron nhân tạo, chẳng hạn như Random Forest, Gradient Boosting, hoặc mô hình kết hợp để cải thiện độ chính xác. - Sử dụng các kỹ thuật tối ưu hóa siêu tham số như Grid Search, Bayesian Optimization để tìm ra cấu hình tốt nhất cho mô hình. - Áp dụng các phương pháp học sâu tiên tiến hơn như CNN hoặc Transformer để khám phá xem liệu chúng có thể nâng cao hiệu suất dự đoán hay không.

3.2.3 Tăng cường khả năng ứng dụng thực tế

- Triển khai mô hình lên một ứng dụng web hoặc nền tảng API để người dùng có thể nhập thông tin nhà và nhận dự đoán giá ngay lập tức. - Kết hợp với các bản đồ trực tuyến để hiển thị trực quan thông tin về giá nhà theo vị trí địa lý. - Nghiên cứu thêm về cách tích hợp mô hình vào hệ thống đề xuất giá nhà cho các nền tảng mua bán bất động sản.

3.2.4 Tối ưu hóa hiệu suất mô hình

- Nghiên cứu và áp dụng các phương pháp giảm thời gian huấn luyện mô hình, như sử dụng GPU hoặc các kỹ thuật song song hóa. - Cải thiện khả năng tổng quát hóa của mô hình bằng cách áp dụng kỹ thuật Regularization hoặc Dropout hợp lý. - Thử nghiệm với các phương pháp xử lý dữ liệu tiên tiến hơn như Feature Engineering để tìm ra những đặc trưng quan trọng giúp mô hình học tốt hơn.

Tóm lại, đề tài đã cung cấp một giải pháp khả thi để dự đoán giá nhà bằng mạng neuron nhân tạo, đồng thời mở ra nhiều hướng nghiên cứu và ứng dụng tiềm năng trong tương lai. Với những cải tiến và mở rộng hợp lý, mô hình có thể trở thành một công cụ hữu ích trong lĩnh vực bất động sản, hỗ trợ các nhà đầu tư, người mua nhà và các tổ chức tài chính trong việc đưa ra quyết định.

Tài liệu tham khảo

@bookgoodfellow2016deep, author = Ian Goodfellow and Yoshua Bengio and Aaron Courville, title = Deep Learning, publisher = MIT Press, year = 2016

@bookbishop2006pattern, author = Christopher M. Bishop, title = Pattern Recognition and Machine Learning, publisher = Springer, year = 2006

@bookchollet2018deep, author = François Chollet, title = Deep Learning with Python, publisher = Manning Publications, year = 2018

@articlelecun1998mnist, author = Yann LeCun and Léon Bottou and Yoshua Bengio and Patrick Haffner, title = Gradient-Based Learning Applied to Document Recognition, journal = Proceedings of the IEEE, volume = 86, number = 11, pages = 2278–2324, year = 1998

@bookgeron2019hands, author = Aurélien Géron, title = Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, publisher = O'Reilly Media, edition = 2nd, year = 2019

@articlezhang2021deep, author = Cheng Zhang and Yu Liu, title = A Review on Deep Learning Applications in Tabular Data Analysis, journal = Neural Networks, volume = 142, pages = 79–93, year = 2021

@articlekingma2014adam, author = Diederik P. Kingma and Jimmy Ba, title = Adam: A Method for Stochastic Optimization, journal = arXiv preprint arXiv:1412.6980, year = 2014

@articlehe2015delving, author = Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun, title = Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, journal = Proceedings of the IEEE International Conference on Computer Vision, pages = 1026–1034, year = 2015

@articlescikit-learn, author = Fabian Pedregosa and Gaël Varoquaux and Alexandre Gramfort and Vincent Michel and Bertrand Thirion and Olivier Grisel and Mathieu Blondel and Peter Prettenhofer and Ron Weiss and Vincent Dubourg and others, title = Scikit-learn: Machine Learning in Python, journal = Journal of Machine Learning Research, volume = 12, pages = 2825–2830, year = 2011