TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CUỐI KÌ MÔN NHẬP MÔN HỌC MÁY

Người hướng dẫn: **GV.LÊ ANH CƯỜNG**

Người thực hiện: **NGUYỄN QUANG LỢI – 52100909**

Lớp **: 21050301**

Khoá **: 25**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CUỐI KÌ MÔN NHẬP MÔN HỌC MÁY

Người hướng dẫn: **GV.LÊ ANH CƯỜNG**
Người thực hiện: **NGUYỄN QUANG LỢI – 52100909**
Lớp **: 21050301**
Khoá **: 25**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành nhất đến Thầy Lê Anh Cường vì sự hướng dẫn và sự hỗ trợ không ngừng nghỉ của Thầy trong suốt thời gian của khóa học. Sự kiên nhẫn và tận tâm của Thầy đã là nguồn động viên lớn lao giúp em vượt qua những thách thức trong quá trình học tập.

Những bài học từ Thầy không chỉ giúp em hiểu rõ hơn về chuyên ngành mà còn mở ra cánh cửa cho sự phát triển cá nhân. Sự tận tâm và kiến thức sâu rộng của Thầy đã truyền cảm hứng cho em để em có thể làm tốt không những trong môn học này mà còn các môn học khác.

Em rất biết ơn vì những kiến thức mà Thầy truyền đạt sẽ là nền tảng quý báu cho sự nghiệp sau này của em. Một lần nữa, em xin chân thành cảm ơn Thầy vì mọi điều.

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của Thầy Lê Anh Cường. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày tháng năm

Tác giả

(ký tên và ghi rõ họ tên)

Nguyễn Quang Lợi

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

MỤC LỤC

CHƯƠNG 1 – BÀI CÁ NHÂN

1.1 Tìm hiểu, so sánh các phương pháp Optimizer trong huấn luyện mô hình học máy

Trong huấn luyện mô hình học máy, các phương pháp optimizer được sử dụng để tối ưu hóa hàm mất mát bằng cách điều chỉnh các tham số của mô hình. Dưới đây là một số phương pháp optimizer phổ biến và so sánh giữa chúng:

1. Stochastic Gradient Descent (SGD):

- Ưu điểm của Stochastic Gradient Descent (SGD):

Tính đơn giản: SGD là phương pháp đơn giản và dễ triển khai. Nó chỉ đơn giản là cập nhật gradient dựa trên từng điểm dữ liệu một cách ngẫu nhiên.

Hiệu quả với dữ liệu lớn: Đối với dữ liệu lớn, SGD có thể hoạt động tốt hơn so với phương pháp Gradient Descent thông thường bởi vì nó chỉ sử dụng một điểm dữ liệu hoặc một số điểm dữ liệu nhỏ để cập nhật gradient.

Hội tụ nhanh hơn: Đôi khi SGD có thể hội tụ nhanh hơn so với các phương pháp tối ưu hóa batch, đặc biệt đối với bộ dữ liệu lớn và không đồng nhất.

Ít dễ rơi vào cực tiểu cục bộ: Vì nó chỉ xem xét một mẫu nhỏ hoặc một điểm dữ liệu một cách ngẫu nhiên, nó có khả năng ít bị rơi vào cực tiểu cục bộ hơn so với Gradient Descent thông thường.

- Nhược điểm của Stochastic Gradient Descent (SGD):

Khả năng hội tụ không ổn định: Do việc cập nhật gradient ngẫu nhiên, SGD có thể không ổn định và không hội tụ đến điểm cực tiểu mong muốn.

Cần tuning learning rate cẩn thận: Learning rate trong SGD cần phải được tuning cẩn thận. Một learning rate lớn có thể làm mô hình không hội tụ hoặc "nhảy qua" cực tiểu, trong khi một learning rate nhỏ có thể làm chậm quá trình học.

Không hiệu quả với dữ liệu có tính đồng nhất cao: Đối với dữ liệu đồng nhất (homogeneous), SGD có thể không hiệu quả và có thể tốn nhiều thời gian hơn để hội tụ.

Yếu tố ngẫu nhiên: Việc chọn ngẫu nhiên các điểm dữ liệu có thể làm cho quá trình huấn luyện không ổn định và khó tái tạo kết quả.

2. Adagrad:

- Ưu điểm của Adagrad:

Điều chỉnh tự động learning rate: Adagrad tự điều chỉnh learning rate dựa trên mức độ thay đổi của từng tham số. Các tham số thường được cập nhật với learning rate lớn hơn nếu chúng ít thay đổi và ngược lại, giúp mô hình học tốt hơn.

Hiệu quả với dữ liệu thưa: Adagrad hoạt động tốt với dữ liệu thưa (sparse data) vì nó cập nhật các tham số có độ quan trọng cao nhất theo tỉ lệ nhất định.

Không yêu cầu tuning learning rate thủ công: Adagrad giúp giảm bớt việc tuning learning rate thủ công, điều này có ích đặc biệt khi làm việc với dữ liệu có độ quan trọng khác nhau.

- Nhược điểm của Adagrad:

Learning rate giảm quá nhanh: Trong quá trình huấn luyện, việc tích lũy bình phương độ dốc có thể làm cho learning rate giảm quá nhanh và dần dần trở nên rất nhỏ. Điều này có thể dẫn đến việc dừng quá trình học sớm hơn khi chưa đạt được điểm tối ưu.

Khó khăn khi áp dụng cho mô hình Deep Learning: Trong mô hình Deep Learning, Adagrad có thể gặp vấn đề với việc learning rate giảm quá nhanh, làm cho quá trình học dừng sớm trước khi đạt được điểm tối ưu.

Không giải quyết được vấn đề đào tạo trên cùng một bộ dữ liệu nhiều lần: Vì learning rate giảm theo thời gian, việc huấn luyện trên cùng một bộ dữ liệu nhiều lần có thể làm giảm khả năng học của mô hình.

Yêu cầu quản lý việc chia nguyên: Việc chia nguyên trong mẫu bình phương độ dốc có thể dẫn đến vấn đề khi cần giảm learning rate quá nhanh.

3. RMSprop:

- Ưu điểm của RMSprop:

Hiệu quả với dữ liệu thưa: RMSprop thường hoạt động tốt với dữ liệu thưa do việc tự điều chỉnh learning rate dựa trên trung bình trượt của bình phương độ dốc. Nó giúp ổn định quá trình học trong trường hợp này.

Chống lại vấn đề learning rate giảm quá nhanh: RMSprop giúp giải quyết vấn đề về learning rate giảm quá nhanh của Adagrad bằng cách sử dụng trung bình trượt của bình phương độ dốc. Điều này giúp làm giảm ảnh hưởng của việc chia nguyên trong mẫu bình phương độ dốc.

Hiệu quả với mô hình Deep Learning: RMSprop thường hoạt động tốt hơn trong việc huấn luyện các mô hình sâu (Deep Learning) hơn so với Adagrad bởi vì nó giúp ngăn chặn việc learning rate giảm quá nhanh.

- Nhược điểm của RMSprop:

Cần tuning hyperparameters: RMSprop cần tuning hyperparameters như learning rate để đảm bảo hiệu suất tối ưu. Các giá trị mặc định có thể không phù hợp cho tất cả các loại mô hình hoặc tập dữ liệu.

Khó khăn khi áp dụng cho dữ liệu đồng nhất: RMSprop có thể không hoạt động tốt với dữ liệu đồng nhất (homogeneous) vì nó vẫn có thể gặp vấn đề với việc giảm learning rate quá nhanh.

Chưa giải quyết được hoàn toàn vấn đề bias: RMSprop vẫn có thể gặp phải vấn đề bias vì việc tính toán gradient chỉ dựa trên một số lượng nhất định các lần cập nhật trước đó.

4. Adam (Adaptive Moment Estimation):

- Ưu điểm của Adam:

Hiệu quả với các tập dữ liệu lớn: Adam thường hoạt động tốt trên các tập dữ liệu lớn và không đồng nhất, giúp tối ưu hóa nhanh hơn và đạt đến điểm tối ưu một cách hiệu quả.

Tự điều chỉnh learning rate: Adam tự điều chỉnh learning rate cho mỗi tham số dựa trên gradient của nó, giúp tránh được vấn đề của việc chia nguyên trong RMSprop và giúp mô hình học ổn định hơn.

Hiệu quả với mô hình Deep Learning: Adam thường được sử dụng phổ biến trong mô hình Deep Learning và có thể cải thiện tốc độ hội tụ so với các phương pháp tối ưu hóa khác.

Kết hợp Momentum và RMSprop: Adam kết hợp ưu điểm của Momentum (giúp vượt qua điểm tối ưu cục bộ) và RMSprop (điều chỉnh learning rate) để cải thiện việc huấn luyện mô hình.

- Nhược điểm của Adam:

Yêu cầu tuning hyperparameters: Adam yêu cầu tuning hyperparameters như learning rate, beta1 (momentum decay), beta2 (exponential decay rates for the gradients), để đạt được hiệu suất tốt nhất. Các giá trị mặc định có thể không phù hợp cho mọi loại mô hình và dữ liệu.

Nhạy cảm với noise: Adam có thể nhạy cảm với noise trong dữ liệu và dễ bị ảnh hưởng bởi các giá trị gradient không chính xác hoặc nhiễu.

Khả năng overfitting: Trong một số trường hợp, Adam có thể dẫn đến overfitting nếu không được tuning hyperparameters một cách chính xác.

5. AdamW (Adam with Weight Decay):

- Ưu điểm của AdamW:

Giảm overfitting: AdamW giúp giảm overfitting bằng cách thêm trọng lượng giảm vào quá trình tối ưu hóa. Điều này giúp kiểm soát trọng lượng của các tham số trong mô hình.

Ổn định việc học: Trong AdamW, việc thêm trọng lượng giảm cải thiện tính ổn định của quá trình học. Nó giúp ngăn chặn việc trọng lượng tăng lên quá nhanh và giúp mô hình học hiệu quả hơn.

Hiệu quả với mô hình Deep Learning: AdamW thường hoạt động tốt với mô hình Deep Learning, giúp cải thiện khả năng học và kiểm soát overfitting.

Giải quyết vấn đề của Adam với trọng lượng 0: AdamW giải quyết vấn đề mà Adam gặp phải khi trọng lượng ban đầu của các tham số là 0. Trong Adam, trọng lượng 0 có thể không được cập nhật trong quá trình huấn luyện, trong khi AdamW giúp khắc phục điều này.

- Nhược điểm của AdamW:

Cần tuning hyperparameters: Cũng như Adam, AdamW cần tuning hyperparameters như learning rate, beta1, beta2, weight decay để đạt được hiệu suất tối ưu. Việc chọn các giá trị phù hợp có thể không dễ dàng.

Tính khái quát không cao: Trong một số trường hợp cụ thể, AdamW có thể không hoạt động tốt với một số loại mô hình hoặc dữ liệu cụ thể, và cần sự thử nghiệm để xem xét tác động thực sự.

Cần kiến thức vững về đối tượng tối ưu hóa: Sử dụng AdamW đòi hỏi người sử dụng phải hiểu rõ về đối tượng tối ưu hóa để chọn lựa các hyperparameters và kiểm soát quá trình tối ưu hóa một cách hiệu quả.

6. Nadam (Nesterov-accelerated Adaptive Moment Estimation):

- Ưu điểm của Nadam:

Kết hợp giữa Adam và Nesterov Accelerated Gradient (NAG): Nadam kết hợp ưu điểm của Adam (tối ưu hóa nhanh chóng, không đòi hỏi tuning hyperparameters quá nhiều) và NAG (chống overshooting, ổn định hóa quá trình học) để cải thiện quá trình tối ưu hóa.

Hiệu quả với các mô hình Deep Learning: Nadam thường hoạt động tốt với các mô hình Deep Learning và có thể cải thiện tốc độ hội tụ so với các phương pháp tối ưu hóa khác.

Giảm tác động của momen trong Adam: Nadam giảm tác động của moment trong Adam, giúp tránh được việc overshooting và ổn định hóa quá trình học.

Tự điều chỉnh learning rate: Tương tự như Adam, Nadam có khả năng tự điều chỉnh learning rate, giúp quá trình tối ưu hóa được mượt mà và ổn định hơn.

- Nhược điểm của Nadam:

Cần tuning hyperparameters: Như các phương pháp tối ưu hóa khác, Nadam cũng cần tuning hyperparameters như learning rate, beta1, beta2 để đạt được hiệu suất tối ưu.

Khả năng overfitting: Trong một số trường hợp cụ thể, Nadam có thể dẫn đến overfitting nếu không được tuning hyperparameters một cách chính xác.

Yêu cầu kiến thức vững về tối ưu hóa: Sử dụng Nadam đòi hỏi người sử dụng phải có kiến thức vững về tối ưu hóa để lựa chọn hyperparameters và kiểm soát quá trình tối ưu hóa một cách hiệu quả.

7. so sánh các phương pháp optimizer

- Ưu điểm:

+ SGD:

Dễ cài đặt và hiểu.

Hiệu quả với dữ liệu lớn và gần với các điểm tối ưu cục bộ.

+ Adagrad:

Hiệu quả với dữ liệu thưa.

Tự điều chỉnh learning rate dựa trên lịch sử của gradient.

+ RMSprop:

Hiệu quả với dữ liệu không đồng nhất.

Giảm tác động của việc giảm learning rate quá nhanh trong Adagrad.

+ Adam:

Kết hợp ưu điểm của Momentum và RMSprop.

Tự điều chỉnh learning rate, hiệu quả với mô hình Deep Learning.

+ AdamW:

Giảm overfitting thông qua việc thêm trọng lượng giảm.

Hiệu quả với mô hình Deep Learning và khả năng kiểm soát trọng lượng tham số.

+ Nadam:

Kết hợp Adam và Nesterov Accelerated Gradient.

Giảm overshooting và cải thiện ổn định quá trình học.

- Nhược điểm:

+ SGD:

Dễ rơi vào bẫy địa phương.

Khó tìm được learning rate tối ưu, đặc biệt với các dữ liệu có biến động lớn.

+ Adagrad:

Dễ bị giảm learning rate quá nhanh và không thể phục hồi.

Không hiệu quả với các mô hình Deep Learning có thể cần sự điều chỉnh.

+ RMSprop:

Yêu cầu tuning hyperparameters.

Không hoạt động tốt với dữ liệu đồng nhất.

+ Adam:

Cần tuning hyperparameters.

Có thể gây overfitting nếu không được tuning chính xác.

+ AdamW:

Cần tuning hyperparameters.

Tính tổng quát không cao trong một số trường hợp cụ thể.

+ Nadam:

Yêu cầu tuning hyperparameters.

Có thể dẫn đến overfitting nếu không được tuning chính xác.

1.2 Tìm hiểu về Continual Learning và Test Production khi xây dựng một giải pháp học máy để giải quyết một bài toán nào đó.

- Continual Learning:

Continual Learning (CL) là một lĩnh vực quan trọng trong Machine Learning, nó tập trung vào việc phát triển các mô hình có khả năng liên tục học từ dữ liệu mới mà

không quên đi kiến thức cũ. Trong học máy truyền thống, mô hình thường được huấn luyện trên một tập dữ liệu cố định và sau đó không thể áp dụng cho dữ liệu mới một cách linh hoạt. Tuy nhiên, trong thực tế, dữ liệu thay đổi liên tục và mô hình cần cập nhật để phản ánh thông tin mới nhất. Điều này đặc biệt quan trọng trong các lĩnh vực như hệ thống nhận dạng, robot tự học, hoặc các ứng dụng IoT.

+ Chiến lược Continual Learning:

Regularization và Rehearsal:

- Regularization: Sử dụng các kỹ thuật như Elastic Weight Consolidation (EWC), Synaptic Intelligence (SI) để giữ lại thông tin quan trọng từ dữ liệu cũ khi học từ dữ liệu mới. Nó giúp mô hình không quên đi kiến thức đã học.
- Rehearsal: Là việc sử dụng dữ liệu cũ trong quá trình huấn luyện để đảm bảo rằng mô hình vẫn biết về dữ liệu cũ khi học dữ liệu mới.

Knowledge Distillation:

Truyền đạt kiến thức từ mô hình cũ sang mô hình mới, giúp mô hình mới học từ thông tin quan trọng đã được mô hình cũ học từ trước.

Expansion và Compression:

- Expansion: Tăng kích thước của mô hình để chứa thông tin mới mà không ảnh hưởng đến kiến thức cũ.
- Compression: Nén kiến thức cũ vào mô hình mới để không làm giảm hiệu suất mà vẫn giữ được thông tin quan trọng.

Context-dependent Gate:

Sử dụng các cổng (gates) để quyết định xem dữ liệu nào sẽ được sử dụng để cập nhật mô hình và dữ liệu nào sẽ bị quên đi.

Meta-learning:

Sử dụng meta-learning để học cách học, giúp mô hình tự điều chỉnh để học từ dữ liệu mới một cách hiệu quả.

+ Thách thức trong Continual Learning:

Công việc quên đi và học mới: Mô hình cần phải biết cách quên đi thông tin không cần thiết và học thông tin mới một cách hiệu quả.

Nguy cơ Overfitting và Catastrophic Forgetting: Đối mặt với nguy cơ quên đi thông tin quan trọng từ dữ liệu cũ khi học từ dữ liệu mới, dẫn đến việc mô hình không thể tổng hợp được kiến thức toàn diện.

Tính ổn định và hiệu suất: Đôi khi CL có thể dẫn đến tính ổn định thấp trong quá trình học và giảm hiệu suất của mô hình.

- Test Production:

Test Production là quá trình xây dựng và duy trì hệ thống kiểm thử liên tục để đánh giá hiệu suất của mô hình học máy trong môi trường thực tế. Đây là một phần quan trọng của quy trình phát triển và triển khai mô hình, giúp đảm bảo rằng mô hình hoạt động đúng đắn và hiệu quả khi triển khai vào môi trường sản xuất.

+ Các Bước Quan Trọng trong Test Production:

- Kiểm thử mô hình: Xây dựng các bộ kiểm thử để đảm bảo mô hình hoạt động đúng đắn trên nhiều tình huống khác nhau. Các loại kiểm thử có thể bao gồm kiểm thử đơn vị, kiểm thử tích hợp và kiểm thử hệ thống.
- Tự động hóa Kiểm thử: Xây dựng quy trình tự động hoá để thực hiện các bộ kiểm thử trên mô hình một cách liên tục và tự động. Điều này giúp đảm bảo rằng mọi thay đổi hoặc cập nhật đều được kiểm tra một cách kỹ lưỡng.
- Giám sát và Logging: Xây dựng các hệ thống giám sát liên tục để theo dõi hiệu suất của mô hình trong môi trường thực tế. Các log và thông báo cần phải được gửi đi khi có bất kỳ sự cố nào xảy ra.
- Continuous Integration và Deployment (CI/CD): Sử dụng CI/CD để tự động hóa việc triển khai mô hình và kiểm thử liên tục. Điều này giúp đảm bảo rằng mọi cập nhật hoặc thay đổi đều được kiểm thử một cách tự động và đồng nhất.
- Monitoring Performance: Theo dõi hiệu suất của mô hình trong môi trường thực tế và thực hiện các bước cần thiết để cải thiện hiệu suất khi cần thiết.

- Re-training và Fine-tuning: Nếu mô hình không hoạt động tốt, cần thiết lập quy trình để tái huấn luyện mô hình với dữ liệu mới hoặc fine-tuning mô hình để cải thiện hiệu suất.

  + Thách thức trong Test Production:

- Tự động hóa và Đồng nhất hóa: Xây dựng quy trình tự động hoá kiểm thử có thể gặp khó khăn trong việc đồng bộ hóa các môi trường khác nhau.
- Quản lý Logs và Thông báo: Xử lý lượng log lớn và thông báo có thể trở nên phức tạp, đặc biệt khi có nhiều sự kiện xảy ra cùng lúc.
- Hiệu suất và Tính ổn định: Đảm bảo rằng mô hình không chỉ hoạt động tốt trên dữ liệu huấn luyện mà còn trên dữ liệu thực tế với độ ổn định cao.
- Test Production là một phần quan trọng trong quy trình phát triển mô hình học máy, giúp đảm bảo rằng mô hình có hiệu suất tốt khi triển khai vào môi trường thực tế.

CHƯƠNG 2 – BÀI NHÓM

Bài toán: Dự đoán doanh số bán hàng của cửa hàng bán lẻ

Phân tích thống kê trên dữ liệu, vẽ các đồ thị để hiểu bài toán, hiểu dữ liệu. Tìm hiểu các đặc trưng và đánh gía vai trò của các đặc trưng đối với mục tiêu bài toán

Số Feature/Attribute: Số lượng đặc trưng sẽ bao gồm cả các đặc trưng numerical và categorical.

Numerical: Số lượng khách hàng, diện tích cửa hàng, số lượng sản phẩm trưng bày.

Categorical: Loại cửa hàng (ví dụ: thời trang, thực phẩm, điện tử), quận/phường cửa hàng, các chương trình khuyến mãi đang diễn ra.

2.1 Phân tích thống kê trên dữ liệu, vẽ các đồ thị để hiểu bài toán, hiểu dữ liệu. Tìm hiểu các đặc trưng và đánh gía vai trò của các đặc trưng đối với mục tiêu bài toán;

2.1.1 Phân tích thống kê trên dữ liệu:

Mô tả dữ liệu:

Sử dụng thống kê mô tả để hiểu về giá trị trung bình, phương sai, giá trị tối đa, giá trị tối thiểu của các đặc trưng numerical.

Đếm số lượng cửa hàng trong từng loại cửa hàng để hiểu phân phối của đặc trưng categorical.

Mối quan hệ giữa các đặc trưng và doanh số bán hàng:

Vẽ biểu đồ phân tán giữa số lượng khách hàng và doanh số bán hàng để xem liệu có mối quan hệ tuyến tính hay không.

Sử dụng biểu đồ hộp để so sánh doanh số bán hàng giữa các loại cửa hàng.

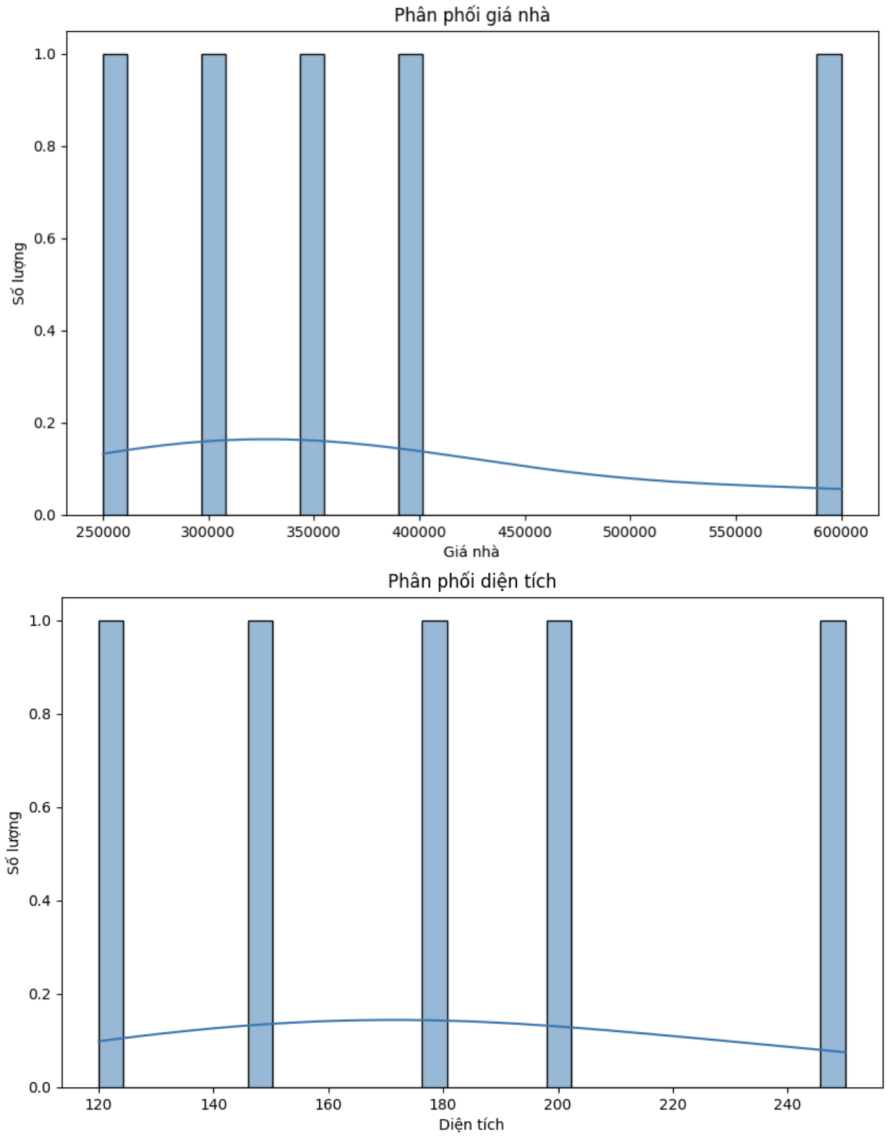
Phân phối doanh số bán hàng:

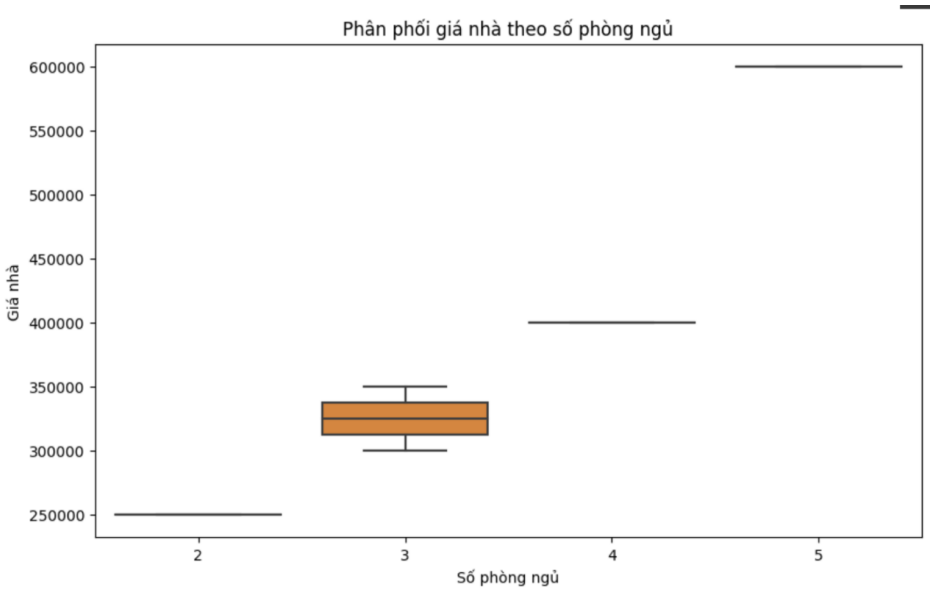
Vẽ biểu đồ histogram để thấy phân phối của doanh số bán hàng.

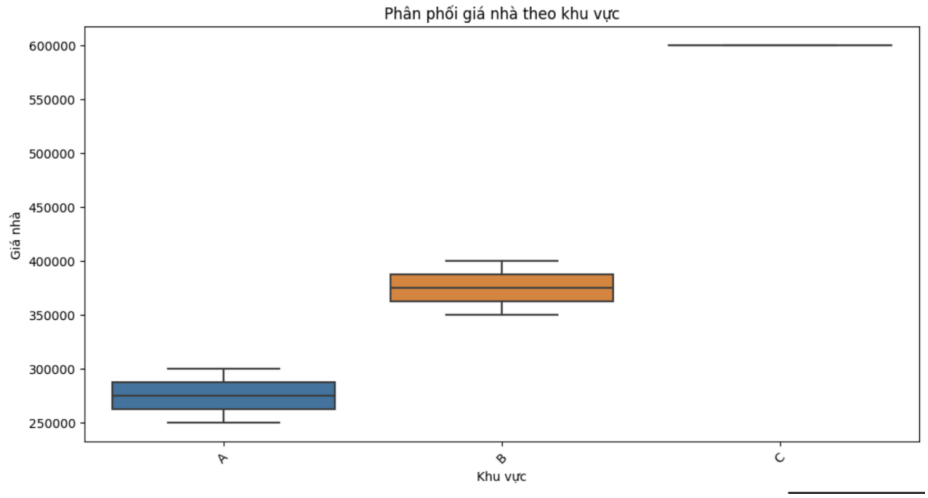
Phân tích tương quan:

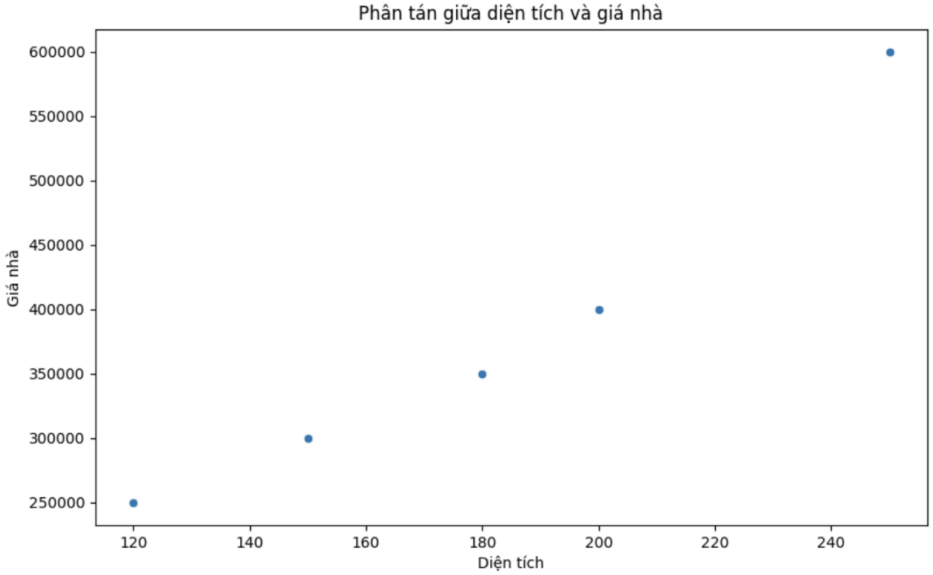
Sử dụng heatmap để kiểm tra tương quan giữa các đặc trưng numerical.

2.1.2 Vẽ các đồ thị để hiểu bài toán, hiểu dữ liệu




Phân phối giá nhà theo số phòng ngủ
Giá nhà
600000
550000
500000
450000
400000
350000
300000
250000
2
3
4
5
Số phòng ngủ


Phân phối giá nhà theo khu vực
Giá nhà
600000
550000
500000
450000
400000
350000
300000
250000
A
B
C
Khu vực



2.1.3 Tìm hiểu các đặc trưng và đánh giá vai trò của các đặc trưng đối với mục tiêu bài toán

- Mô tả thống kê:

Mô tả: Đối với đặc trưng hạng mục, bạn có thể sử dụng các thống kê mô tả như phần trăm, trung bình, độ lệch chuẩn và miền giá trị để hiểu phân phối của từng giá trị trong đặc trưng.

2.1.3.1 Hiểu Về Phân Phối:

Vai Trò: Hàm describe() cung cấp thông tin về trung bình, độ lệch chuẩn, giá trị tối thiểu, giá trị tối đa và các phân vị chính của các biến trong dữ liệu.

Ý Nghĩa: Giúp hiểu rõ hơn về hình dạng của phân phối của mỗi biến, xác định mức độ biến động, và phát hiện các giá trị ngoại lệ.

2.1.3.2 Xác Định Sự Biến Động:

Vai Trò: So sánh giá trị trung bình với độ lệch chuẩn để đánh giá mức độ biến động của dữ liệu.

Ý Nghĩa: Cho biết mức độ biến động của dữ liệu và sự đồng đều hay không đồng đều giữa các quan sát.

2.1.3.3 Phân Tích Mức Tính và Phân Vị:

Vai Trò: Mô tả các giá trị cơ bản như trung bình (mean), trung vị (median), phân vị 25%, 50%, và 75%.

Ý Nghĩa: Cung cấp cái nhìn về vị trí trung tâm và biến động của dữ liệu, giúp đánh giá sự phân phối và mức độ biến động.

2.1.3.4 Nhận Diện Giá Trị Ngoại Lệ:

Vai Trò: Hiển thị giá trị tối thiểu và tối đa, giúp xác định sự xuất hiện của giá trị ngoại lệ.

Ý Nghĩa: Cho biết về mức độ biến động cực đoan và có thể cần phải kiểm tra các giá trị ngoại lệ có ảnh hưởng đến kết quả phân tích hay không.

2.1.3.5 Đánh Giá Chất Lượng Dữ Liệu:

Vai Trò: Kiểm tra sự đầy đủ và tính đúng đắn của dữ liệu.

Ý Nghĩa: Nếu có giá trị thiếu, đặc trưng không biểu diễn đúng, hoặc các vấn đề khác, có thể cần xử lý trước khi tiến hành phân tích chi tiết hơn.

Sử dụng df.describe() để xem các thống kê cơ bản về các đặc trưng số:

	id	price	bedrooms	bathrooms	sqft_living	\
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	

	sqft_lot	floors	waterfront	view	condition	\
count	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000	
mean	1.510697e+04	1.494309	0.007542	0.234303	3.409430	
std	4.142051e+04	0.539989	0.086517	0.766318	0.650743	
min	5.200000e+02	1.000000	0.000000	0.000000	1.000000	
25%	5.040000e+03	1.000000	0.000000	0.000000	3.000000	
50%	7.618000e+03	1.500000	0.000000	0.000000	3.000000	
75%	1.068800e+04	2.000000	0.000000	0.000000	4.000000	
max	1.651359e+06	3.500000	1.000000	4.000000	5.000000	

	grade	sqft_above	sqft_basement	yr_built	yr_renovated	\
count	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	
mean	7.656873	1788.390691	291.509045	1971.005136	84.402258	
std	1.175459	828.090978	442.575043	29.373411	401.679240	
min	1.000000	290.000000	0.000000	1900.000000	0.000000	
25%	7.000000	1190.000000	0.000000	1951.000000	0.000000	
50%	7.000000	1560.000000	0.000000	1975.000000	0.000000	
75%	8.000000	2210.000000	560.000000	1997.000000	0.000000	
max	13.000000	9410.000000	4820.000000	2015.000000	2015.000000	

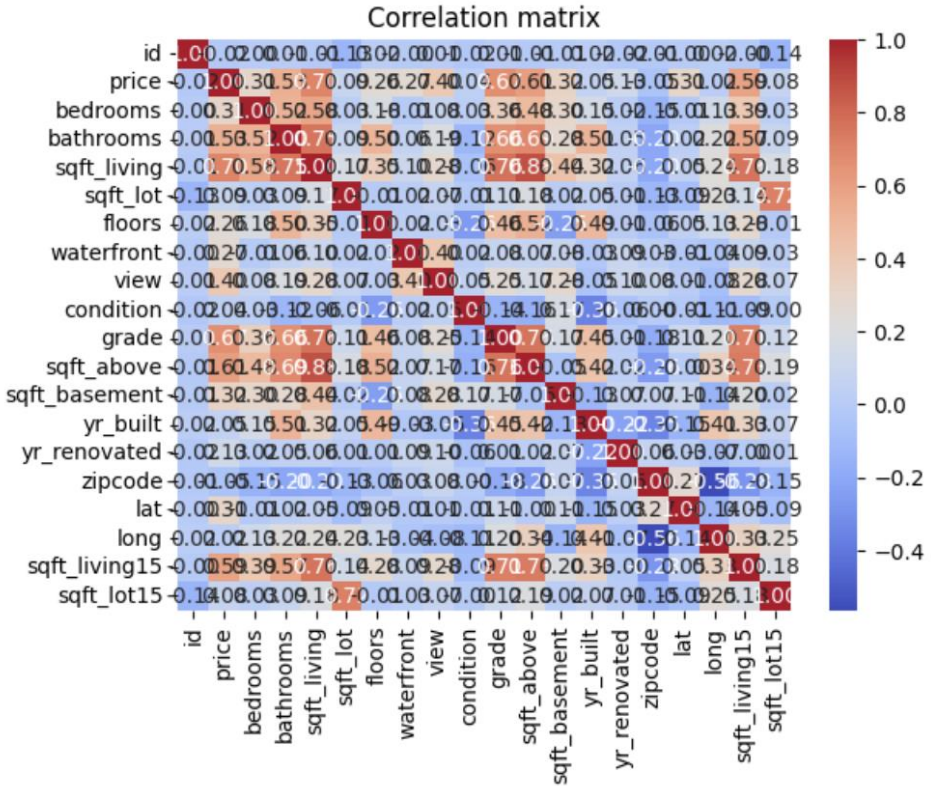
	zipcode	lat	long	sqft_living15	sqft_lot15
count	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000
mean	98077.939805	47.560053	-122.213896	1986.552492	12768.455652
std	53.505026	0.138564	0.140828	685.391304	27304.179631
min	98001.000000	47.155900	-122.519000	399.000000	651.000000
25%	98033.000000	47.471000	-122.328000	1490.000000	5100.000000
50%	98065.000000	47.571800	-122.230000	1840.000000	7620.000000
75%	98118.000000	47.678000	-122.125000	2360.000000	10083.000000
max	98199.000000	47.777600	-121.315000	6210.000000	871200.000000

2.1.3.6 Phân tích tương quan:

Vai trò: Xác định tương quan giữa các cặp đặc trưng và tương quan với giá nhà.

Mô tả: Tìm hiểu xem liệu có sự tương quan giữa các đặc trưng hay không, và làm thế nào chúng ảnh hưởng đồng thời đến giá nhà. Một ma trận tương quan có thể cung cấp thông tin này.

Sử dụng heatmap để vẽ ma trận tương quan giữa các đặc trưng số và mục tiêu:



2.1.3.7 Biểu đồ phân phối:

- Vai Trò của Biểu Đồ Phân Phối:

Hiển Thị Phân Phối: Biểu đồ phân phối giúp chúng ta hiểu rõ hơn về cách giá trị của một biến được phân phối trong dữ liệu.

Phát Hiện Mẫu Phân Phối: Cho phép nhận biết các mô hình phân phối, như phân phối chuẩn, đối với mục tiêu và các đặc trưng.

- Mô Tả Biểu Đồ Phân Phối Mục Tiêu (Target):

Mục Tiêu: Xem phân phối của giá trị mục tiêu (ví dụ: giá nhà).

Mô Tả:

+ Nếu phân phối gần với phân phối chuẩn, có thể thuận tiện cho việc sử dụng các mô hình hồi quy tuyến tính.

+ Nếu có độ lệch, có thể cần xử lý (như log-transform) để đảm bảo phân phối đối xứng và đúng chuẩn.

- Mô Tả Biểu Đồ Phân Phối Đặc Trưng Số (Numerical Features):

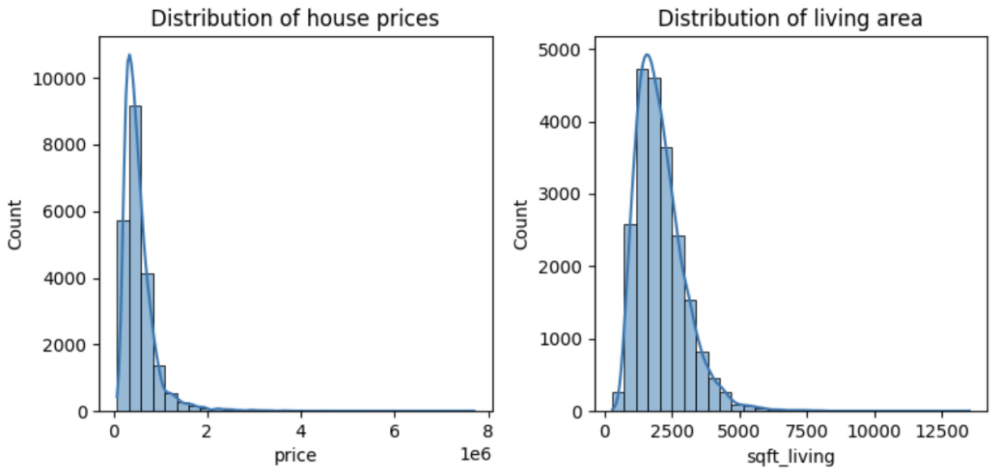
Mục Tiêu: Hiểu về phân phối của các đặc trưng số như diện tích, số phòng ngủ, v.v.

Mô Tả:

+ Nếu có đặc điểm đặc trưng số chính, như diện tích, có thể xác định xem liệu dữ liệu có phân phối đồng đều hay tập trung ở một vùng cụ thể.

+ Các biểu đồ có thể giúp phát hiện mẫu như sự phân cụm hay sự tập trung ở một khu vực cụ thể.

Sử dụng histogram để xem phân phối của mục tiêu và các đặc trưng số:



2.1.3.8 Biểu đồ phân tán:

- Mô Tả:

Hiển Thị Mối Quan Hệ Tuyến Tính: Scatter plot thường được sử dụng để xem xét mối quan hệ tuyến tính giữa một đặc trưng số và mục tiêu. Nó giúp xác định xem có xu hướng tăng hoặc giảm theo đặc trưng không.

Phát Hiện Mẫu: Có thể phát hiện mẫu hoặc cụm dữ liệu, đặc biệt là khi có sự tập trung ở các vùng cụ thể.

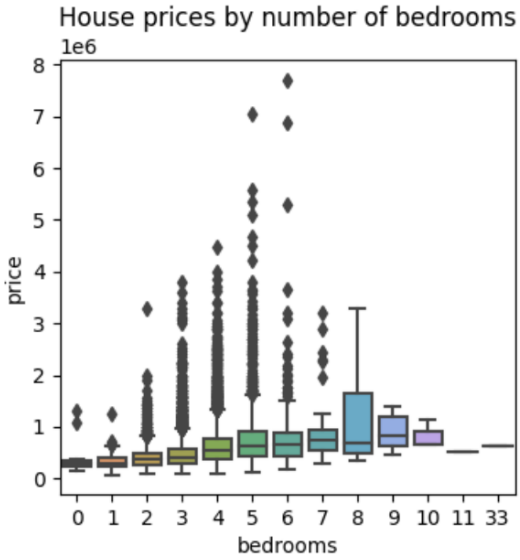
- Vai Trò và Ý Nghĩa:

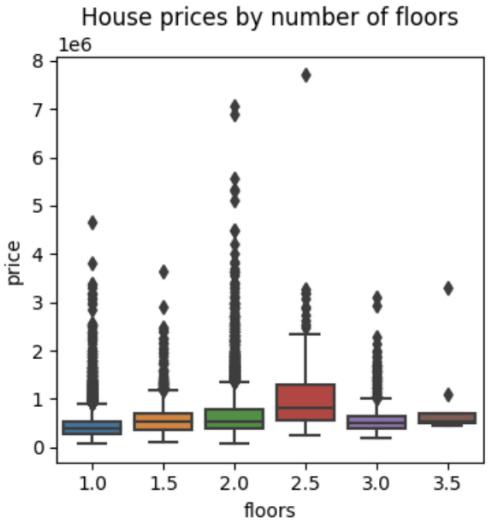
Đánh Giá Mối Quan Hệ: Scatter plot giúp đánh giá mối quan hệ giữa đặc trưng và mục tiêu. Nếu có sự tăng hoặc giảm, có thể áp dụng mô hình hồi quy tuyến tính.

Phát Hiện Outliers: Giúp nhận diện giá trị ngoại lệ, những điểm dữ liệu có giá trị đặc biệt xa khỏi xu hướng chung.

Kiểm Tra Đồng Biến Hay Nghịch Biến: Nếu scatter plot có hình dạng đường chéo, có thể ám chỉ mối quan hệ đồng biến (cùng chiều) hoặc nghịch biến (ngược chiều).

Sử dụng scatter plot để thấy mối quan hệ giữa các đặc trưng số và mục tiêu:





2.1.3.9 Phân tích ảnh hưởng:

- Mô Tả:

Correlation Matrix: Là một bảng chứa các hệ số tương quan giữa các biến trong dữ liệu. Trong trường hợp này, chúng ta sử dụng correlation matrix để đo lường mức độ tương quan giữa mỗi đặc trưng và mục tiêu (ví dụ: giá nhà).

Giá Trị Tương Quan (Correlation): Các giá trị tương quan thường nằm trong khoảng từ -1 đến 1. Giá trị càng gần -1 hoặc 1 thì mối tương quan càng mạnh, trong khi giá trị gần 0 thì không có mối tương quan.

- Vai Trò và Ý Nghĩa:

Đánh Giá Mức Độ Tương Quan: Giúp đánh giá mức độ tương quan giữa mỗi đặc trưng và mục tiêu. Nếu giá trị tương quan cao, đặc trưng có thể có ảnh hưởng lớn đến mục tiêu.

Lựa Chọn Đặc Trưng: Các đặc trưng có giá trị tương quan cao có thể được ưu tiên khi xây dựng mô hình học máy.

Phát Hiện Đặc Trưng Tương Quan Ngược: Giúp phát hiện các đặc trưng có tương quan âm với mục tiêu, tức là khi một tăng thì mục tiêu giảm và ngược lại.

Sử dụng correlation matrix để xác định mức độ tương quan giữa mỗi đặc trưng và mục tiêu:

2.1.3.10 Phân tích đặc trưng quan trọng:

- Mô Tả:

Đặc Trưng Quan Trọng (Feature Importance): Trong mô hình Random Forest, Đặc Trưng Quan Trọng đo lường mức độ ảnh hưởng của từng đặc trưng đối với khả năng dự đoán của mô hình.

Gini Importance: Được đo bằng cách xem xét sự giảm Gini impurity khi sử dụng mỗi đặc trưng để phân chia dữ liệu. Đặc trưng có ảnh hưởng cao hơn sẽ có Gini Importance cao hơn.

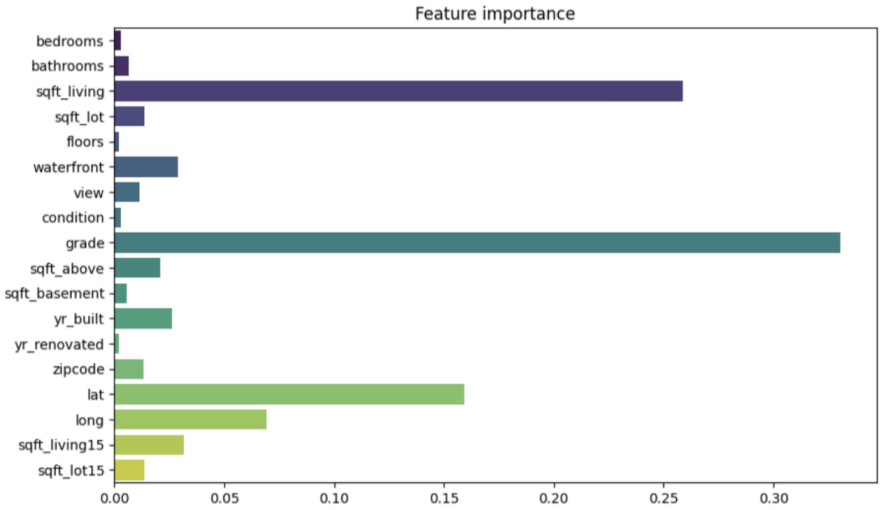
- Vai Trò và Ý Nghĩa:

Đánh Giá Ảnh Hưởng Đặc Trưng: Feature Importance giúp đánh giá mức độ ảnh hưởng của từng đặc trưng đối với mô hình. Các đặc trưng có Gini Importance cao đóng góp nhiều vào quá trình ra quyết định của mô hình.

Lựa Chọn Đặc Trưng: Có thể sử dụng thông tin về đặc trưng quan trọng để chọn ra các đặc trưng quan trọng nhất cho mô hình, giúp giảm chiều của dữ liệu và tăng hiệu suất.

Hiểu Rõ Mô Hình: Đánh giá Đặc Trưng Quan Trọng giúp hiểu rõ hơn về cơ cấu và quyết định của mô hình Random Forest.

Sử dụng mô hình học máy như Random Forest để xác định đặc trưng quan trọng:



2.1.3.11 Mô hình hóa:

- Mô Tả:

Mô Hình Hóa Độ Quan Trọng: Mô hình hóa độ quan trọng của các đặc trưng là quá trình sử dụng các thuật toán máy học để đánh giá mức độ ảnh hưởng của mỗi đặc trưng đối với dự đoán của mô hình.

Visualization: Các biểu đồ hoặc biểu đồ cột thường được sử dụng để hiển thị mức độ quan trọng của từng đặc trưng.

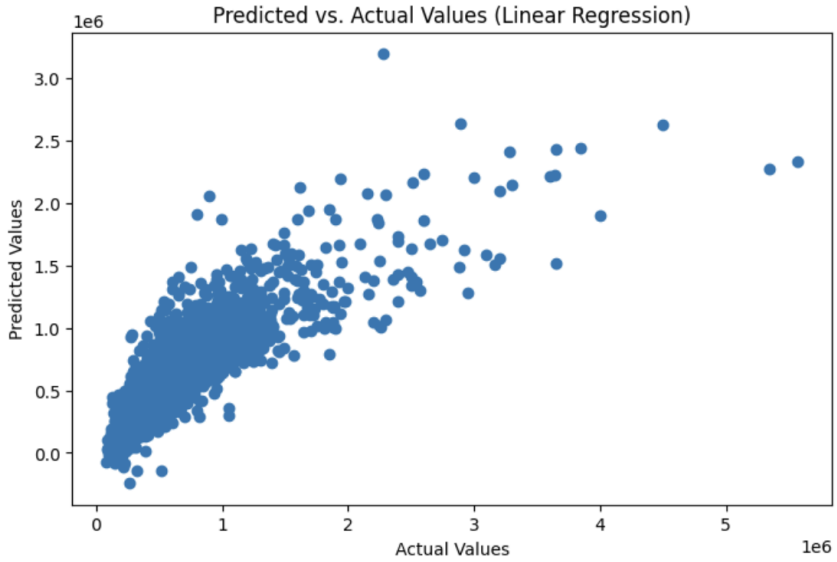
- Vai Trò và Ý Nghĩa:

Đánh Giá Đặc Trưng Quan Trọng: Mô hình hóa độ quan trọng giúp đánh giá mức độ quan trọng của mỗi đặc trưng trong quá trình dự đoán của mô hình.

Tối Ưu Hóa Mô Hình: Thông tin về độ quan trọng của đặc trưng có thể được sử dụng để tối ưu hóa mô hình bằng cách lựa chọn các đặc trưng quan trọng nhất hoặc điều chỉnh trọng số của chúng.

Hiểu Rõ Hơn về Mô Hình: Việc hiểu rõ về tầm quan trọng của từng đặc trưng giúp dự đoán cách mà mô hình sử dụng thông tin để ra quyết định.

Kiểm tra mức độ quan trọng của các đặc trưng trong mô hình:



2.2 Ứng Dụng Các Mô Hình Học Máy Cơ Bản và Ensemble Learning đồng thời áp dụng các kỹ thuật tránh overfitting

2.2.1 Chuẩn Bị Dữ Liệu:

One-Hot Encoding: Nếu có đặc trưng hạng mục, sử dụng one-hot encoding để chuyển đổi chúng thành dạng số.

Chia Tập Dữ Liệu: Chia tập dữ liệu thành tập huấn luyện và tập kiểm tra.

2.2.2 Mô Hình Hóa:

Sử dụng mô hình học máy cơ bản và các mô hình thuộc Ensemble Learning.

2.2.3 Đánh Giá Mô Hình:

Sử dụng các chỉ số như Mean Squared Error (MSE), R-squared để đánh giá hiệu suất của mô hình trên tập kiểm tra.

2.2.4 Hiện thực mô hình:

Sử dụng mô hình Linear Regression, Ridge Regression, Random Forest và Gradient Boosting

2.2.4.1 Tạo DataFrame và One-Hot Encoding:

Tạo DataFrame từ dữ liệu mẫu và sử dụng one-hot encoding để chuyển đặc trưng hạng mục thành các cột nhị phân.

```
# Tạo dữ liệu mẫu với nhiều đặc trưng hơn
data_complex = {
    'Giá nhà': [300000, 400000, 250000, 600000, 350000, 500000],
    'Diện tích': [150, 200, 120, 250, 180, 300],
    'Số phòng ngủ': [3, 4, 2, 5, 3, 4],
    'Khu vực': ['A', 'B', 'A', 'C', 'B', 'C'],
    'Loại nhà': ['Nhà phố', 'Biệt thự', 'Nhà phố', 'Biệt thự', 'Nhà phố', 'Biệt thự'],
    'Năm xây dựng': [2000, 2010, 1990, 2015, 2005, 2018],
    'Đánh giá': [4.5, 5.0, 3.5, 4.8, 4.2, 4.9]
}

# Tạo DataFrame từ dữ liệu
df_complex = pd.DataFrame(data_complex)

# One-hot encoding cho các đặc trưng hạng mục
df_encoded_complex = pd.get_dummies(df_complex, columns=['Khu vực', 'Loại nhà'])
```

2.2.4.2 Tách Đặc Trưng và Biến Mục Tiêu, Chia Tập Dữ Liệu:

Tách đặc trưng và biến mục tiêu, sau đó chia tập dữ liệu thành tập huấn luyện và tập kiểm tra.

```
# Tách đặc trưng và biến mục tiêu
X_complex = df_encoded_complex.drop('Giá nhà', axis=1)
y_complex = df_encoded_complex['Giá nhà']

# Chia tập dữ liệu thành tập huấn luyện và tập kiểm tra
X_train_complex, X_test_complex, y_train_complex, y_test_complex = train_test_split(
    X_complex, y_complex, test_size=0.2, random_state=42
)
```

2.2.4.3 Sử Dụng Các Mô Hình Học Máy:

Khởi tạo các mô hình học máy: Linear Regression, Ridge Regression, Random Forest và Gradient Boosting.

```
# Sử dụng các mô hình học máy
linear_model_complex = LinearRegression()
ridge_model_complex = Ridge(alpha=1.0)
rf_model_complex = RandomForestRegressor()
gb_model_complex = GradientBoostingRegressor()
```

Để tránh tình trạng overfitting cho các mô hình ta có thể điều chỉnh các giá trị của tham số

```
# Sử dụng các mô hình học máy
linear_model_complex = LinearRegression()
ridge_model_complex = Ridge(alpha=0.5)  # Thay đổi giá trị alpha để kiểm soát overfitting
rf_model_complex = RandomForestRegressor(n_estimators=100, max_depth=10) # Điều chỉnh n_estimators và max_depth để tránh overfitting
gb_model_complex = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=3) # Điều chỉnh các tham số để tránh overfitting
```

2.2.4.4 Huấn Luyện Các Mô Hình:

Huấn luyện các mô hình trên tập huấn luyện.

```
# Huấn luyện các mô hình
linear_model_complex.fit(X_train_complex, y_train_complex)
ridge_model_complex.fit(X_train_complex, y_train_complex)
rf_model_complex.fit(X_train_complex, y_train_complex)
gb_model_complex.fit(X_train_complex, y_train_complex)
```

2.2.4.5 Dự Đoán và Đánh Giá Hiệu Suất:

Dự đoán trên tập kiểm tra và đánh giá hiệu suất bằng MSE và R-squared.

```
# Dự đoán trên tập kiểm tra
linear_predictions_complex = linear_model_complex.predict(X_test_complex)
ridge_predictions_complex = ridge_model_complex.predict(X_test_complex)
rf_predictions_complex = rf_model_complex.predict(X_test_complex)
gb_predictions_complex = gb_model_complex.predict(X_test_complex)
```

2.2.4.6 In Kết Quả Đánh Giá:

In kết quả đánh giá hiệu suất của các mô hình.

```
# In kết quả
print(f'Linear Regression - Mean Squared Error: {linear_mse_complex}, R-squared: {linear_r2_complex}')
print(f'Ridge Regression - Mean Squared Error: {ridge_mse_complex}, R-squared: {ridge_r2_complex}')
print(f'Random Forest - Mean Squared Error: {rf_mse_complex}, R-squared: {rf_r2_complex}')
print(f'Gradient Boosting - Mean Squared Error: {gb_mse_complex}, R-squared: {gb_r2_complex}')
```

2.3 Sử dụng Feed Forward Neural Network và Reccurent Neural Network (hoặc mô thuộc loại này) để giải quyết bài toán đồng thời áp dụng các kỹ thuật tránh overfitting

2.3.1 Sử dụng Feed Forward Neural Network để dự đoán giá nhà

Phân tích chi tiết về code sử dụng Recurrent Neural Network (RNN) để giải quyết bài toán dự đoán giá nhà

2.3.1.1 Chuẩn bị Dữ Liệu:

- Mô Tả:

Dữ liệu từ data_complex được chuyển thành DataFrame df_complex.

Dữ liệu được phân chia thành các tập huấn luyện và kiểm tra bằng train_test_split.

Dữ liệu đầu vào (X_complex) và đầu ra (y_complex) được xác định.

Dữ liệu được chuẩn hóa bằng cách sử dụng StandardScaler.

```
df_complex = pd.DataFrame(data_complex)

# Phân chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_complex = df_complex.drop('Giá nhà', axis=1)
y_complex = df_complex['Giá nhà']
X_train_complex, X_test_complex, y_train_complex, y_test_complex = train_test_split(
    X_complex, y_complex, test_size=0.2, random_state=42
)

# Chuẩn hóa dữ liệu
scaler_complex = StandardScaler()
X_train_scaled_complex = scaler_complex.fit_transform(X_train_complex)
X_test_scaled_complex = scaler_complex.transform(X_test_complex)
```

2.3.1.2 Xây Dựng Mô Hình:

Mô hình FFNN được xây dựng bằng cách sử dụng lớp Sequential từ Keras.

Các lớp Dense được thêm vào mô hình với số lượng nơ-ron và hàm kích hoạt phù hợp.

```
# Xây dựng mô hình FFNN
model_complex = Sequential()
model_complex.add(Dense(64, input_dim=X_train_scaled_complex.shape[1], activation='relu'))
model_complex.add(Dense(32, activation='relu'))
model_complex.add(Dense(1, activation='linear'))
```

Để tránh tình trạng overfitting cho mô hình ta có thể sử dụng Regularization bằng cách thêm L2 Regularization vào các lớp Dense của mô hình để kiểm soát overefitting thông qua tham số ‘kernel_regularizer’

```
# Xây dựng mô hình FFNN
model_complex = Sequential()
model_complex.add(Dense(64, input_dim=X_train_scaled_complex.shape[1], activation='relu', kernel_regularizer=regularizers.l2(0.01))) #Thêm L2 regularizatic
model_complex.add(Dense(32, activation='relu', kernel_regularizer=regularizers.l2(0.01))) #Thêm L2 regularization vào các lớp Dense của mô hình để kiểm soá
model_complex.add(Dense(1, activation='linear'))
```

Để sử dụng Regularization thì ta cần inport trước

```
from tensorflow.keras import regularizers
```

2.3.1.3 Compile Mô Hình:

Mô hình được biên dịch với optimizer là Adam, hàm mất mát là mean squared error, và metric là mean absolute error.

```
# Compile mô hình
model_complex.compile(optimizer='adam', loss='mean_squared_error', metrics=['mae'])
```

2.3.1.4 Huấn Luyện Mô Hình:

Mô hình được huấn luyện trên tập huấn luyện với 100 epochs và batch size là 32.

```
# Huấn luyện mô hình
model_complex.fit(X_train_scaled_complex, y_train_complex, epochs=100, batch_size=32, verbose=1, validation_split=0.2)
```

2.3.1 5 Đánh Giá và Dự Đoán:

Mô hình được đánh giá trên tập kiểm tra và kết quả được in ra màn hình.

Dự đoán được thực hiện trên dữ liệu mới và giá trị dự đoán được in ra màn hình.

```
# Đánh giá mô hình trên tập kiểm tra
loss_complex, mae_complex = model_complex.evaluate(X_test_scaled_complex, y_test_complex)
print(f'Mean Absolute Error on Test Set: {mae_complex}')

# Dự đoán trên dữ liệu mới
new_data_complex = pd.DataFrame({
    'Diện tích': [180],
    'Số phòng ngủ': [3],
    'Khu vực_A': [0],
    'Khu vực_B': [1],
    'Khu vực_C': [0],
    'Loại nhà_Nhà phố': [1],
    'Loại nhà_Biệt thự': [0],
    'Năm xây dựng': [2008],
    'Đánh giá': [4.7]
})
new_data_scaled_complex = scaler_complex.transform(new_data_complex)
prediction_complex = model_complex.predict(new_data_scaled_complex)
print(f'Predicted Value for New Data: {prediction_complex[0][0]}')
```

2.3.1.6 Kết quả dự đoán

```
Epoch 1/100
1/1 [==============================] - 2s 2s/step - loss: 144999874560.0000 - mae: 366666.4688 - val_loss: 359999635456.0000 - val_mae: 599999.6875
Epoch 2/100
1/1 [==============================] - 0s 56ms/step - loss: 144999809024.0000 - mae: 366666.3750 - val_loss: 359999537152.0000 - val_mae: 599999.6250
Epoch 3/100
1/1 [==============================] - 0s 70ms/step - loss: 144999727104.0000 - mae: 366666.2500 - val_loss: 359999471616.0000 - val_mae: 599999.5625
Epoch 4/100
1/1 [==============================] - 0s 42ms/step - loss: 144999661568.0000 - mae: 366666.1562 - val_loss: 359999406080.0000 - val_mae: 599999.5000
Epoch 5/100
1/1 [==============================] - 0s 42ms/step - loss: 144999596032.0000 - mae: 366666.0938 - val_loss: 359999340544.0000 - val_mae: 599999.4375
Epoch 6/100
1/1 [==============================] - 0s 41ms/step - loss: 144999530496.0000 - mae: 366665.9688 - val_loss: 359999242240.0000 - val_mae: 599999.3750
Epoch 7/100
1/1 [==============================] - 0s 44ms/step - loss: 144999464960.0000 - mae: 366665.8750 - val_loss: 359999176704.0000 - val_mae: 599999.3125
Epoch 8/100
1/1 [==============================] - 0s 54ms/step - loss: 144999383040.0000 - mae: 366665.7500 - val_loss: 359999111168.0000 - val_mae: 599999.2500
Epoch 9/100
1/1 [==============================] - 0s 58ms/step - loss: 144999317504.0000 - mae: 366665.6562 - val_loss: 359999012864.0000 - val_mae: 599999.1875
Epoch 10/100
1/1 [==============================] - 0s 41ms/step - loss: 144999251968.0000 - mae: 366665.5938 - val_loss: 359998947328.0000 - val_mae: 599999.1250
Epoch 11/100
1/1 [==============================] - 0s 62ms/step - loss: 144999170048.0000 - mae: 366665.4688 - val_loss: 359998881792.0000 - val_mae: 599999.0625
Epoch 12/100
1/1 [==============================] - 0s 66ms/step - loss: 144999104512.0000 - mae: 366665.3438 - val_loss: 359998816256.0000 - val_mae: 599999.0000
Epoch 13/100
1/1 [==============================] - 0s 45ms/step - loss: 144999006208.0000 - mae: 366665.2500 - val_loss: 359998717952.0000 - val_mae: 599998.9375
Epoch 14/100
1/1 [==============================] - 0s 43ms/step - loss: 144998940672.0000 - mae: 366665.1250 - val_loss: 359998586880.0000 - val_mae: 599998.8125
Epoch 15/100
1/1 [==============================] - 0s 45ms/step - loss: 144998875136.0000 - mae: 366665.0312 - val_loss: 359998488576.0000 - val_mae: 599998.7500
Epoch 16/100
1/1 [==============================] - 0s 61ms/step - loss: 144998809600.0000 - mae: 366664.9062 - val_loss: 359998423040.0000 - val_mae: 599998.6875
Epoch 17/100
1/1 [==============================] - 0s 59ms/step - loss: 144998727680.0000 - mae: 366664.7812 - val_loss: 359998357504.0000 - val_mae: 599998.6250
Epoch 18/100
1/1 [==============================] - 0s 48ms/step - loss: 144998645760.0000 - mae: 366664.7188 - val_loss: 359998259200.0000 - val_mae: 599998.5625
Epoch 19/100
1/1 [==============================] - 0s 62ms/step - loss: 144998563840.0000 - mae: 366664.5938 - val_loss: 359998193664.0000 - val_mae: 599998.5000
Epoch 20/100
1/1 [==============================] - 0s 46ms/step - loss: 144998481920.0000 - mae: 366664.4688 - val_loss: 359998128128.0000 - val_mae: 599998.4375
Epoch 21/100
1/1 [==============================] - 0s 58ms/step - loss: 144998416384.0000 - mae: 366664.3438 - val_loss: 359998062592.0000 - val_mae: 599998.3750
Epoch 22/100
1/1 [==============================] - 0s 60ms/step - loss: 144998334464.0000 - mae: 366664.2500 - val_loss: 359997964288.0000 - val_mae: 599998.3125
Epoch 23/100
1/1 [==============================] - 0s 66ms/step - loss: 144998252544.0000 - mae: 366664.1250 - val_loss: 359997833216.0000 - val_mae: 599998.1875
Epoch 24/100
1/1 [==============================] - 0s 60ms/step - loss: 144998154240.0000 - mae: 366664.0000 - val_loss: 359997734912.0000 - val_mae: 599998.1250
```

2.3.2 Sử dụng Reccurent Neural Network để dự đoán giá nhà

Phân tích chi tiết về code sử dụng Recurrent Neural Network (RNN) để giải quyết bài toán dự đoán giá nhà

2.3.2.1 Import thư viện:

pandas: Xử lý và phân tích dữ liệu.

numpy: Xử lý mảng và ma trận.

train_test_split: Chia tập dữ liệu thành tập huấn luyện và tập kiểm tra.

StandardScaler: Chuẩn hóa dữ liệu.

Sequential, LSTM, Dense: Các lớp của Keras để xây dựng mô hình neural network.

```
1 import pandas as pd
2 import numpy as np
3 from sklearn.model_selection import train_test_split
4 from sklearn.preprocessing import StandardScaler
5 from tensorflow.keras.models import Sequential
6 from tensorflow.keras.layers import LSTM, Dense
```

2.3.2.2 Tạo dữ liệu

Tạo DataFrame từ dữ liệu

```
 8 # Tạo dữ liệu
 9 data_complex = {
10     'Giá nhà': [300000, 400000, 250000, 600000, 350000, 500000],
11     'Diện tích': [150, 200, 120, 250, 180, 300],
12     'Số phòng ngủ': [3, 4, 2, 5, 3, 4],
13     'Khu vực_A': [1, 0, 1, 0, 0, 0],
14     'Khu vực_B': [0, 1, 0, 0, 1, 0],
15     'Khu vực_C': [0, 0, 0, 1, 0, 1],
16     'Loại nhà_Nhà phố': [1, 0, 1, 0, 1, 0],
17     'Loại nhà_Biệt thự': [0, 1, 0, 1, 0, 1],
18     'Năm xây dựng': [2000, 2010, 1990, 2015, 2005, 2018],
19     'Đánh giá': [4.5, 5.0, 3.5, 4.8, 4.2, 4.9]
20 }
21
22 # Tạo DataFrame từ dữ liệu
23 df_complex = pd.DataFrame(data_complex)
```

2.3.2.3 Tách đặc trưng và biến mục tiêu:

X_complex: DataFrame chứa các đặc trưng.

y_complex: Seri chứa biến mục tiêu.

```
25 # Tách đặc trưng và biến mục tiêu
26 X_complex = df_complex.drop('Giá nhà', axis=1)
27 y_complex = df_complex['Giá nhà']
```

2.3.2.4 Chuẩn hóa dữ liệu:

Sử dụng StandardScaler để chuẩn hóa đặc trưng.

```
# Chuẩn hóa dữ liệu
scaler_complex = StandardScaler()
X_scaled_complex = scaler_complex.fit_transform(X_complex)
```

2.3.2.5 Chuyển đổi dữ liệu thành chuỗi thời gian:

Chuyển đổi dữ liệu thành dạng chuỗi thời gian phù hợp cho mô hình RNN.

```
# Chuyển đổi dữ liệu thành chuỗi thời gian (time series)
X_time_series_complex = np.reshape(X_scaled_complex, (X_scaled_complex.shape[0], 1, X_scaled_complex.shape[1]))
```

2.3.2.6 Phân chia tập dữ liệu:

Chia tập dữ liệu thành tập huấn luyện và tập kiểm tra.

```
# Phân chia tập dữ liệu
X_train_complex, X_test_complex, y_train_complex, y_test_complex = train_test_split(
    X_time_series_complex, y_complex, test_size=0.2, random_state=42
)
```

2.3.2.7 Xây dựng mô hình RNN:

Sử dụng mô hình Sequential và thêm một lớp LSTM và một lớp Dense.

```
# Xây dựng mô hình RNN
model_rnn = Sequential()
model_rnn.add(LSTM(50, input_shape=(X_time_series_complex.shape[1], X_time_series_complex.shape[2])))
model_rnn.add(Dense(1))
```

Để tránh tình trạng overfitting ta cần sử dụng Regularization

```
# Xây dựng mô hình RNN
model_rnn = Sequential()
model_rnn.add(LSTM(50, input_shape=(X_time_series.shape[1], X_time_series.shape[2]), kernel_regularizer=regularizers.l2(0.01))) #Thêm L2 regularization v
model_rnn.add(Dense(1))
```

Trước khi sử dụng ta cần import trước

```
from tensorflow.keras import regularizers
```

Đồng thời ta sẽ kết hợp thêm Early-stopping

```
# Huấn luyện mô hình
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
model_rnn.fit(X_train, y_train, epochs=50, batch_size=2, verbose=1, validation_split=0.2, callbacks=[early_stopping]) # Sử dụng early-stopping để tránh ovs
```

Trước khi sử dụng ta cần import trước

```
from tensorflow.keras.callbacks import EarlyStopping
```

2.3.2.8 Compile và huấn luyện mô hình:

Sử dụng adam làm tối ưu hóa và hàm mất mát là mean_squared_error.

```
# Compile mô hình
model_rnn.compile(optimizer='adam', loss='mean_squared_error')

# Huấn luyện mô hình
model_rnn.fit(X_train_complex, y_train_complex, epochs=50, batch_size=2, verbose=1)
```

2.3.2.9 Đánh giá mô hình trên tập kiểm tra:

Đánh giá hiệu suất của mô hình trên tập kiểm tra.

```
# Đánh giá mô hình trên tập kiểm tra
mse_rnn = model_rnn.evaluate(X_test_complex, y_test_complex)
print(f'Mean Squared Error on Test Set (RNN): {mse_rnn}')
```

2.3.2.10 Dự đoán giá nhà trên dữ liệu mới:

Dự đoán giá nhà trên dữ liệu mới bằng cách chuyển đổi và chuẩn hóa dữ liệu, sau đó sử dụng mô hình đã huấn luyện.

```
# Dự đoán giá nhà trên dữ liệu mới
new_data_complex = {'Diện tích': [210], 'Số phòng ngủ': [4], 'Khu vực_A': [0], 'Khu vực_B': [1],
                    'Khu vực_C': [0], 'Loại nhà_Nhà phố': [0], 'Loại nhà_Biệt thự': [1],
                    'Năm xây dựng': [2022], 'Đánh giá': [4.7]}
new_data_scaled_complex = scaler_complex.transform(pd.DataFrame(new_data_complex))
new_data_time_series_complex = np.reshape(new_data_scaled_complex, (1, 1, new_data_scaled_complex.shape[1]))
prediction_rnn = model_rnn.predict(new_data_time_series_complex)
print(f'Predicted Price for New Data (RNN): {prediction_rnn[0][0]}')
```

2.3.2.11 Kết quả thực nghiệm

```
Epoch 24/50
2/2 [==============================] - 0s 8ms/step - loss: 198749470720.0000
Epoch 25/50
2/2 [==============================] - 0s 9ms/step - loss: 198749437952.0000
Epoch 26/50
2/2 [==============================] - 0s 9ms/step - loss: 198749372416.0000
Epoch 27/50
2/2 [==============================] - 0s 9ms/step - loss: 198749372416.0000
Epoch 28/50
2/2 [==============================] - 0s 9ms/step - loss: 198749323264.0000
Epoch 29/50
2/2 [==============================] - 0s 9ms/step - loss: 198749290496.0000
Epoch 30/50
2/2 [==============================] - 0s 10ms/step - loss: 198749257728.0000
Epoch 31/50
2/2 [==============================] - 0s 8ms/step - loss: 198749224960.0000
Epoch 32/50
2/2 [==============================] - 0s 8ms/step - loss: 198749192192.0000
Epoch 33/50
2/2 [==============================] - 0s 8ms/step - loss: 198749143040.0000
Epoch 34/50
2/2 [==============================] - 0s 9ms/step - loss: 198749093888.0000
Epoch 35/50
2/2 [==============================] - 0s 15ms/step - loss: 198749044736.0000
Epoch 36/50
2/2 [==============================] - 0s 11ms/step - loss: 198748995584.0000
Epoch 37/50
2/2 [==============================] - 0s 16ms/step - loss: 198748962816.0000
Epoch 38/50
2/2 [==============================] - 0s 9ms/step - loss: 198748897280.0000
Epoch 39/50
2/2 [==============================] - 0s 8ms/step - loss: 198748848128.0000
Epoch 40/50
2/2 [==============================] - 0s 8ms/step - loss: 198748798976.0000
Epoch 41/50
2/2 [==============================] - 0s 9ms/step - loss: 198748749824.0000
Epoch 42/50
2/2 [==============================] - 0s 10ms/step - loss: 198748667904.0000
Epoch 43/50
2/2 [==============================] - 0s 8ms/step - loss: 198748635136.0000
Epoch 44/50
2/2 [==============================] - 0s 8ms/step - loss: 198748569600.0000
Epoch 45/50
2/2 [==============================] - 0s 8ms/step - loss: 198748520448.0000
Epoch 46/50
2/2 [==============================] - 0s 8ms/step - loss: 198748471296.0000
Epoch 47/50
2/2 [==============================] - 0s 9ms/step - loss: 198748389376.0000
Epoch 48/50
2/2 [==============================] - 0s 9ms/step - loss: 198748307456.0000
```

2.4 Sau khi huấn luyện xong mô hình thì muốn cải thiện độ chính xác, ta sẽ làm gì để giải quyết nó? Phân tích các trường hợp sai, đề ra giải pháp và thực hiện nó, sau đó đánh giá xem có cải tiến so với trước không.

Cải thiện độ chính xác của mô hình sau khi đã huấn luyện có thể được thực hiện thông qua một số bước quan trọng sau:

- Phân tích các trường hợp sai (Errors Analysis):

False Positives và False Negatives: Xác định các trường hợp mà mô hình dự đoán sai, bao gồm cả các dự đoán positive sai (False Positives) và negative sai (False Negatives).

Lấy mẫu dữ liệu: Xem xét một số trường hợp sai để hiểu vì sao mô hình dự đoán sai, kiểm tra xem liệu chúng có mô hình các biểu hiện đặc trưng nào không.

- Hiểu rõ hơn về dữ liệu (Data Understanding):

Khám phá dữ liệu (Exploratory Data Analysis - EDA): Điều tra sâu hơn về dữ liệu để hiểu rõ hơn về tính chất, phân phối và mối quan hệ giữa các đặc trưng và biến mục tiêu.

- Thử nghiệm các kiến trúc mô hình khác nhau:

Mô hình hóa khác: Thử các mô hình khác nhau hoặc thay đổi cấu trúc của mô hình hiện tại để xem liệu có mô hình nào tốt hơn không.

- Tinh chỉnh siêu tham số (Hyperparameter Tuning):

Grid Search hoặc Random Search: Sử dụng các kỹ thuật tinh chỉnh siêu tham số để điều chỉnh các tham số của mô hình và tìm ra các giá trị tối ưu.

- Thêm dữ liệu mới hoặc thực hiện feature engineering:

Thêm dữ liệu mới: Nếu có thể, việc thêm dữ liệu mới có thể cải thiện hiệu suất của mô hình.

Feature Engineering: Tạo ra các đặc trưng mới hoặc chuyển đổi các đặc trưng hiện có để mô hình có thể học được thông tin hữu ích hơn.

- Regularization và Dropout (đối với Deep Learning):

Regularization: Áp dụng L1, L2 regularization hoặc Dropout để kiểm soát overfitting trong mô hình Deep Learning.

- Kiểm tra và đánh giá lại mô hình sau mỗi thay đổi:

Cross-validation: Sử dụng kỹ thuật cross-validation để đánh giá mô hình trên nhiều tập dữ liệu khác nhau và đảm bảo tính tổng quát hóa.

Đánh giá các metric khác nhau: Không chỉ sử dụng accuracy, mà còn xem xét các metric khác như precision, recall, F1-score tùy thuộc vào bối cảnh vấn đề.

CHƯƠNG 3 – GITHUB

- Link github: https://github.com/qloi/52100909_Final_.git