|  |
| --- |
| **HỌC VIỆN KỸ THUẬT QUÂN SỰ**  **HỌ VÀ TÊN: NGUYỄN ĐỨC HẬU**  **NGÔ VĂN THƯỜNG**  **KHÓA: 15**  **HỆ ĐÀO TẠO: KỸ SƯ DÂN SỰ**  **ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**  **CHUYÊN NGÀNH: CÔNG NGHỆ DỮ LIỆU**  **ỨNG DỤNG KỸ THUẬT HỌC SÂU TRONG XÂY DỰNG DỊCH VỤ DỰ BÁO GIÁ PHÒNG CHO THUÊ**  **NĂM** **2021** |
| **HỌC VIỆN KỸ THUẬT QUÂN SỰ**  **HỌ VÀ TÊN: NGUYỄN ĐỨC HẬU**  **NGÔ VĂN THƯỜNG**  **KHÓA: 15**  **HỆ ĐÀO TẠO: KỸ SƯ DÂN SỰ**  **ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**  **NGÀNH: CÔNG NGHỆ THÔNG TIN**.  **MÃ SỐ: 52480201**  **ỨNG DỤNG KỸ THUẬT HỌC SÂU TRONG XÂY DỰNG DỊCH VỤ DỰ BÁO GIÁ PHÒNG CHO THUÊ**  ***Cán bộ hướng dẫn khoa học: Trung tá, GV, TS. Cao Văn Lợi***  **NĂM** **2021** |

|  |  |
| --- | --- |
| **HỌC VIỆN KỸ THUẬT QUÂN SỰ** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM** |
| **KHOA CÔNG NGHỆ THÔNG TIN** | **ĐỘC LẬP - TỰ DO - HẠNH PHÚC** |
| **BỘ MÔN:** |  |

**NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP**

Họ và tên: Nguyễn Đức Hậu Lớp: CNDL15 Khóa: 15

Ngô Văn Thường Lớp: CNDL15 Khóa: 15

Ngành: Công nghệ thông tin Chuyên ngành: Công nghệ dữ liệu.

1. Tên đề tài: Ứng dụng kỹ thuật học sâu trong xây dựng dịch vụ dự báo giá phòng cho thuê tại Hà Nội.

2. Các số liệu ban đầu:

- Quyết định Giao đồ án tốt nghiệp đại học – Học viện KTQS

- Tài liệu tham khảo

3. Nội dung bản thuyết minh:

- Mở đầu

- Chương 1: Tổng quan về học sâu

- Chương 2: Tổng quan về học sâu cho dự báo giá phòng

- Chương 3: Học sâu cho định giá phòng tại Hà Nội

- Chương 4: Xây dựng chương trình định giá

- Kết luận

- Tài liệu tham khảo

4. Số lượng, nội dung các bản vẽ (ghi rõ loại, kích thước và cách thực hiện các bản vẽ) và các sản phẩm cụ thể (nếu có):

Được sử dụng máy tính và máy chiếu để trình chiếu.

5. Cán bộ hướng dẫn:

- Họ và tên: Cao Văn Lợi

- Cấp bậc: Trung tá

- Học hàm, học vị: GV, TS

- Đơn vị: Bộ môn An toàn thông tin - Khoa Công nghệ thông tin

- Hướng dẫn toàn bộ

Ngày giao: ......../......../2020 Ngày hoàn thành ........./........../2021

*Hà Nội, ngày ........ tháng ……..năm* 2021

**Chủ nhiệm bộ môn Cán bộ hướng dẫn**

(Ký, ghi rõ họ tên, học hàm, học vị)

**Học viên thực hiện**

Đã hoàn thành và nộp đồ án *........* tháng *……..*năm2021

(Ký và ghi rõ họ tên)

**MỤC LỤC**

[MỞ ĐẦU 1](#_Toc73537578)

[1. Lý do chọn đề tài 1](#_Toc73537579)

[3. Mục đích nghiên cứu 2](#_Toc73537580)

[4. Đối tượng và phạm vi nghiên cứu 3](#_Toc73537581)

[5. Nhiệm vụ nghiên cứu 3](#_Toc73537582)

[6. Phương pháp nghiên cứu 3](#_Toc73537583)

[CHƯƠNG 1: CÁC KIẾN THỨC LIÊN QUAN 4](#_Toc73537584)

[1.1. Tổng quan về Khai phá dữ liệu 4](#_Toc73537585)

[1.1.1. Các bước khám phá trí thức (KDD) 5](#_Toc73537586)

[1.1.2. Các phương pháp trong Khai phá dữ liệu 10](#_Toc73537587)

[1.2. Giới thiệu về Neural Network 11](#_Toc73537588)

[1.2.1. Lịch sử phát triển 11](#_Toc73537589)

[1.2.2. Ứng dụng 12](#_Toc73537590)

[1.2.3. Kiến thức cơ sở 13](#_Toc73537591)

[1.2.4. Mô hình Neural Network 24](#_Toc73537592)

[1.3. Giới thiệu về Học sâu 25](#_Toc73537593)

[1.4. Các mô hình, kiến trúc học sâu 28](#_Toc73537594)

[1.4.1. Multilayer Perceptrons (MLP) 28](#_Toc73537595)

[1.4.2. Convolutional Neural Networks (CNN) 29](#_Toc73537596)

[1.4.3. Recurrent Neural Network (RNN) 31](#_Toc73537597)

[1.4.4. Deep Belief Network (DBN) 32](#_Toc73537598)

[1.4.5. Deep Predictive coding network (DPCN) 33](#_Toc73537599)

[1.5. Tổng kết chương 34](#_Toc73537600)

[CHƯƠNG 2: MÔ HÌNH HỌC MÁY CHO ĐỊNH GIÁ PHÒNG 35](#_Toc73537601)

[2.1. Mô hình Linear Regression 35](#_Toc73537602)

[2.2. Mô hình KNN 36](#_Toc73537603)

[2.3. Mô hình Random Forest 38](#_Toc73537604)

[CHƯƠNG 3: MÔ HÌNH HỌC SÂU CHO ĐỊNH GIÁ PHÒNG 39](#_Toc73537605)

[3.1. Mô tả dữ liệu 39](#_Toc73537608)

[3.1.1. Nguồn gốc dữ liệu 39](#_Toc73537609)

[3.1.2. Cách thức thu thập dữ liệu 40](#_Toc73537610)

[3.1.3. Mô tả dữ liệu thô 45](#_Toc73537611)

[3.2. Tiền xử lý dữ liệu 47](#_Toc73537612)

[3.3. Phân tích lựa chọn thuộc tính 50](#_Toc73537613)

[3.4. Lựa chọn xây dựng mô hình học sâu 55](#_Toc73537614)

[3.4.1. Lọc dữ liệu nhiễu 55](#_Toc73537615)

[3.4.2. Thiếu dữ liệu 55](#_Toc73537616)

[3.4.3. Chuẩn hóa dữ liệu 56](#_Toc73537617)

[3.4.4. Xây dựng mô hình học sâu 57](#_Toc73537618)

[3.5. Đánh giá chất lượng mô hình 59](#_Toc73537619)

[CHƯƠNG 4: XÂY DỰNG CHƯƠNG TRÌNH ĐỊNH GIÁ 70](#_Toc73537620)

[4.1. Phân tích yêu cầu 70](#_Toc73537621)

[4.2. Xây dựng API định giá 70](#_Toc73537622)

[4.3. Xây dựng chương trình minh họa 72](#_Toc73537623)

[KẾT LUẬN 74](#_Toc73537624)

[1. Kết luận 74](#_Toc73537625)

[2. Hướng phát triển 75](#_Toc73537626)

[TÀI LIỆU THAM KHẢO 76](#_Toc73537627)

**DANH MỤC HÌNH ẢNH**

[Hình 1.1. Quy trình khai phá dữ liệu 5](#_Toc73538971)

[Hình 1.2. Mô hình đơn giản của một Artificial Neuron 14](#_Toc73538972)

[Hình 1.3. Thông số đầu vào và ra giữa các Layer 15](#_Toc73538973)

[Hình 1.4. Mô hình mạng MLP với 2 tầng ẩn 16](#_Toc73538974)

[Hình 1.5. Một số hàm Activation 17](#_Toc73538975)

[Hình 1.6. Đồ thị hàm Sigmoid 18](#_Toc73538976)

[Hình 1.7. Đồ thị hàm Tang 19](#_Toc73538977)

[Hình 1.8. Đồ thị hàm ReLU 20](#_Toc73538978)

[Hình 1.9. Mô hình mạng neuron truyền thẳng 23](#_Toc73538979)

[Hình 1.10. Mô hình mạng neuron hồi quy 23](#_Toc73538980)

[Hình 1.11. Kiến trúc cơ bản của Neural Network 24](#_Toc73538981)

[Hình 1.12. Lịch sử phát triển của Deep Learning 26](#_Toc73538982)

[Hình 1.13. Mô hình Deep Neural Network 27](#_Toc73538983)

[Hình 1.14. Mô hình kiến trúc mạng CNN 30](#_Toc73538984)

[Hình 1.15. Mô hình kiến trúc mạng RNN 31](#_Toc73538985)

[Hình 1.16. Mô hình kiến trúc DBN 33](#_Toc73538986)

[Hình 2.1. Công thức tính khoảng cách trong KNN 37](#_Toc73538987)

[Hình 2.2. Công thức Hamming 37](#_Toc73538988)

[Hình 2.3. Cơ chế họa động của Random forest 39](#_Toc73538989)

[Hình 3.1. Thông tin khảo sát trang website 41](#_Toc73538990)

[Hình 3.2. Trang web phongtro123.com 43](#_Toc73538991)

[Hình 3.3. Giao diện tìm lọc thông tin website 43](#_Toc73538992)

[Hình 3.4. Giao diện kết quả trả về 44](#_Toc73538993)

[Hình 3.5. Giao diện trả về click các trang 44](#_Toc73538994)

[Hình 3.6. Thông tin cơ bản của phòng 45](#_Toc73538995)

[Hình 3.7. Thông tin mô tả chi tiết phòng 45](#_Toc73538996)

[Hình 3.8. Thông tin về đặc điểm bài đăng 45](#_Toc73538997)

[Hình 3.9.Ví dụ một document được lưu trong file url 46](#_Toc73538998)

[Hình 3.10.Ví dụ một document được lưu trong file data 47](#_Toc73538999)

[Hình 3.11. Một số lỗi sai của dữ liệu 48](#_Toc73539000)

[Hình 3.12. Kết quả quá trình tách trường thời gian, giá, diện tích 49](#_Toc73539001)

[Hình 3.13. Kết quả quá trình tách địa chỉ thành sang quận 49](#_Toc73539002)

[Hình 3.14. Quy luật trong cách viết địa chỉ 49](#_Toc73539003)

[Hình 3.15. Kết quả quá trình tách địa chỉ thành đường, phường 50](#_Toc73539004)

[Hình 3.16. Sử dụng tính năng đo khoảng cách 50](#_Toc73539005)

[Hình 3.17. Tách thuộc tính loại phòng và loại wc 51](#_Toc73539006)

[Hình 3.18. Đồ thị số lượng bản ghi trên từng khu vực 52](#_Toc73539007)

[Hình 3.19. Đồ thị phân phối giá phòng và tần suất 53](#_Toc73539008)

[Hình 3.20. Đồ thị phân phối diện tích và tần suất 53](#_Toc73539009)

[Hình 3.21. Biều đồ mức độ quan trọng thuộc tính sử dụng RF 55](#_Toc73539010)

[Hình 3.22. Cơ chế hoạt động k fold cross validation 62](#_Toc73539011)

[Hình 3.23. Biểu đồ giao động giữa giá thực tế và dự báo MLR 66](#_Toc73539012)

[Hình 3.24. Biểu đồ giao động giữa giá thực tế và dự báo KNN 67](#_Toc73539013)

[Hình 3.25. Biểu đồ giao động giữa giá thực tế và dự báo RFR 68](#_Toc73539014)

[Hình 3.26. Biểu đồ giao động giữa giá thực tế và dự báo MLP 69](#_Toc73539015)

[Hình 4.1. Giao diện chương trình demo API định giá 74](#_Toc73539016)

[Hình 4.2. Hiển thị kết quả định giá 75](#_Toc73539017)

# MỞ ĐẦU

## Lý do chọn đề tài

Không phải ai cũng biết định giá cho thuê phòng, nhiều chủ nhà do chưa biết đánh giá định lượng giá cho phòng thuê nên thường đưa ra giá cho thuê không phù hợp so với giá cho thuê phòng của thị trường thực tế.Tương tự như vậy, với người thuê phòng cũng khó khăn trong việc định lượng giá phòng như vậy đã hợp lý chưa.

Từ việc không đánh giá đúng giá trị phòng dẫn đến có chủ phòng thì cho thuê nhà giá thấp, có người thì đưa ra giá cho thuê quá cao so với thị trường.Trong một số trường hợp, chủ nhà cho thuê chỉ dựa trên một vài tiêu chí cá nhân để định giá phòng dẫn đến không đồng nhất trong thị trường thuê phòng.

Vì thế đề tài **“Nghiên cứu và ứng dụng học máy cho định giá phòng”** được ra đời nhằm nghiên cứu các giải pháp công nghệ sử dụng các kĩ thuật học sâu để có thể dự báo một cách tương đối nhất giá trị của phòng, căn nhà cho thuê. Điều này giúp không chỉ giúp cho việc định giá trở lên công bằng hơn giữa những người cho thuê, giữa các chủ đầu tư. Đồng thời cũng nó cũng là một kênh tham khảo với những người đi thuê phòng trước khi học tiến hành thuê.

1. **Cơ sở khoa học, tính thực tiễn đề tài**

Cơ sở khoa học:

* Cơ sở lý thuyết về các phương pháp học sâu.

Ý nghĩa thực tiễn đề tài:

Điều 473 tại Bộ Luật dân sự 2015 làm cơ sở định giá thuê nhà:

**“***1. Giá thêu do các bên thỏa thuận hoặc do bên thứ ba xác định theo yêu cầu của các bên thừ, trừ trường hợp luật có quy định khác [1].*

*2. Trường hợp không có thỏa thuận hoặc thỏa thuận không rõ ràng thì giá thuê được xác định theo giá thị trường tại đại điểm và thời điểm giao kết hợp đồng thuê***”**

Điều 129 Luật Nhà ở 2014 cũng quy định:

**“***1. Bên cho thuê và bên thuê nhà ở được thỏa thuận về thời hạn thuê, giá thuê và hình thức trả tiền thuê nhà theo định kỳ hoặc trả một lần; trường hợp Nhà nước có quy định về giá thuê nhà ở thì các bên phải thực hiện theo quy định đó [2].*

*2. Trường hợp chưa hết hạn hợp đồng thuê nhà ở mà bên cho thuê thực hiện cải tạo nhà ở và được bên thuê đồng ý thì bên cho thuê được quyền điều chỉnh giá thuê nhà ở. Giá thuê nhà ở mới do các bên thỏa thuận; trường hợp không thỏa thuận được thì bên cho thuê có quyền đơn phương chấm dứt hợp đồng thuê nhà ở và phải bồi thường cho bên thuê theo quy định của pháp luật.*

*3. Bên cho thuê và bên thuê nhà ở được Nhà nước bảo hộ các quyền và lợi ích hợp pháp trong quá trình thuê và cho thuê nhà ở*.**”**

Như vậy, để định giá một căn phòng cho thuê là thỏa thuận giữa hai bên gia chủ cho thuê và người thuê nhà, điều này làm cho giá thuê nhà rất khó có thể xác định. Việc này dẫn đến hệ lụy gia chủ cho thuê có thể cho thuê với mức giá cao và người thuê có thể phải thuê với một mức giá cao hơn giá thị trường.

Hiện nay, với sự phát triển của công nghệ đặc biệt là các công nghệ trí tuệ nhân tạo, học máy hoàn toàn có thể giải quyết được những khó khăn này bằng cách đưa ra dự báo giá nhà cho thuê một cách khách quan và chân thực nhất.

Với việc áp dụng công nghệ vào việc dự báo giá sẽ giúp cho người cho thuê có thể đưa ra một mức giá phù hợp nhất tránh tình trạng người cho thuê đưa ra mức giá qua cao dẫn đến không tìm được người thuê, còn người thuê sẽ thuê được phòng, căn nhà với mức giá phải chăng phù hợp với mình.

## Mục đích nghiên cứu

* Tìm hiểu một số phương pháp học sâu ứng dụng trong các bài toán định giá phòng.
* Xây dựng bộ dữ liệu cho định giá nhà cho thuê trên địa bàn một số quận/ huyện Hà Nội.

## Đối tượng và phạm vi nghiên cứu

Đối tượng và phạm vi nghiên cứu của đề tài là nghiên cứu, thử nghiệm các phương pháp học sâu áp dụng cho định giá nhà sử dụng bộ dữ liệu nhà cho thuê tại Hà Nội.

## Nhiệm vụ nghiên cứu

* Xây dựng bộ dữ liệu cho định giá nhà cho thuê tại Hà Nội.
* Xây dựng mô hình định giá nhà cho thuê tại một số quận, huyện Hà Nội.

## Phương pháp nghiên cứu

Phương pháp nghiên cứu tài liệu:

* Đọc, tìm hiểu các bài báo về định giá nhà cho thuê trên thế giới.
* Đọc, tìm hiều cơ bản về học sâu, các phương pháp sử dụng cho định giá, các bước thực hiện đề tài.
* Tìm hiểu các công cụ lập trình hỗ trợ cho đề tài.

Phương pháp nghiên cứu thực hành: Lập trình thử nghiệm, đánh giá kết quả.

**Phân chia công việc**

|  |  |  |
| --- | --- | --- |
|  | Nguyễn Đức Hậu | Ngô Văn Thường |
| Mở đầu | Chung | Chung |
| Phần 1 | 1.3 Giới thiệu về Học sâu.  1.5 Nguyên cứu các mô hình truyền thống cho bài toán định giá phòng | 1.1 Tổng quan về khái phá dữ liệu.  1.2 Nghiên cứu mô hình học sâu cho bài toán định giá phòng |
| Phần 2 | Xây dựng các mô hình mạng truyền thống | Xây dựng mô hình mạng học sâu |
| Phần 3 | Xây dựng UI cho bài toán  Xây dựng tool crawl Url | Xây dựng Api cho bài toán  Xây dựng tool crawl data |

# CÁC KIẾN THỨC LIÊN QUAN

## Tổng quan về Khai phá dữ liệu

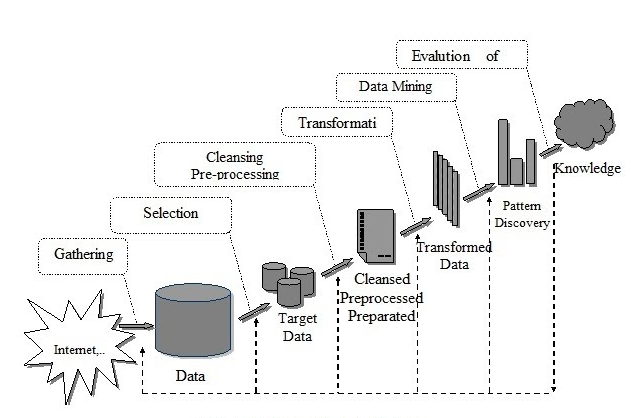
Thuật ngữ Khai phá dữ liệu (KPDL) ra đời vào cuối những năm 80 thế kỷ trước. Có nhiều định nghĩa khác nhau về KPDL, nhưng diễn đạt một cách dễ hiểu thì KPDL là quá trình tìm kiếm những thông tin (tri thức) có ích, tiềm ẩn và mang tính dự đoán trong các khối cơ sở dữ liệu (CSDL) lớn.

Mục đích việc phát hiện tri thức từ dữ liệu KPDL là cốt lõi của **quá trình khám phá tri thức** gồm có các giải thuật KPDL chuyên dùng, dưới một số quy định về hiệu quả tính toán chấp nhận được. KPDL nhằm tìm ra những mẫu mới, những thông tin tiềm ẩn mang tính dự đoán chưa được biết đến, có khả năng mang lại lợi ích cho người sử dụng và KPDL là tìm ra các mẫu được quan tâm nhất tồn tại trong CSDL, nhưng chúng lại bị che giấu bởi một số lượng lớn dữ liệu.

Khai phá dữ liệu là một bước trong bảy bước của quá trình KDD (Knowleadge Discovery in Database).

Trong bài toán dự đoán giá phòng cho thuê cũng sẽ áp dụng các bước trong khái phá tri thức này.

### Các bước khám phá trí thức (KDD)



Hình 1.1. Quy trình khai phá dữ liệu

Bước 1- **Thu thập dữ liệu** (data gathering): Là bước lấy các dữ liệu lien quan đến vấn đề từ các nguồn khác nhau (Internet,…).

Bước 2- **Trích chọn dữ liệu** (data selection): Là bước trích chọn những tập dữ liệu cần được khai phá từ các tập dữ liệu lớn (databases,data warehouses).

Bước 3- **Tiền xử lý dữ liệu** (cleansing preprocessing): Là bước làm sạch dữ liệu (xử lý dữ liệu không đầy đủ, dữ liệu nhiễu, dữ liệu không nhất quán,...), rút gọn dữ liệu (sử dụng các phương pháp thu gọn dữ liệu, histograms, lấy mẫu,...), rời rạc hoá dữ liệu (dựa vào histograms, entropy, phân khoảng,…). Sau bước này, dữ liệu sẽ nhất quán, đầy đủ, được rút gọn và được rời rạc hoá.

Bước 4- **Biến đổi dữ liệu** (data transformation): Là bước chuẩn hoá và làm mịn dữ liệu để đưa dữ liệu về dạng thuận lợi nhất nhằm phục vụ cho các kỹ thuật khai phá ở bước sau.

Bước 5- **Khai phá dữ liệu** (data mining): Đây là bước quan trọng và tốn nhiều thời gian nhất của quá trình **khám phá tri thức**, áp dụng các kỹ thuật khai phá (phần lớn là các kỹ thuật của machine learning) để khai phá, trích chọn được các mẫu (pattern) thông tin, các mối liên hệ đặc biệt trong dữ liệu.

Bước 6- **Đánh giá và biểu diễn tri thức** (knowledge representation & evaluation): Dùng các kỹ thuật hiển thị dữ liệu để trình bày các mẫu thông tin (tri thức) và mối liên hệ đặc biệt trong dữ liệu đã được khai phá ở bước trên biểu diễn theo dạng gần gũi với người sử dụng như đồ thị, cây, bảng biểu, luật,… Đồng thời, bước này cũng đánh giá những tri thức khai phá được theo những tiêu chí nhất định.

Trong giai đoạn KPDL, có thể cần sự tương tác của người dùng để điều chỉnh và rút ra các tri thức cần thiết nhất. Cáctri thức nhận được cũng có thể được lưu và sử dụng lại.

Trong đó đề tài sẽ nhấn mạnh vào 4 bước: ***Thu thập dữ liệu,*** ***Tiền xử lý dữ liệu, Biến đổi dữ liệu, Khai phá dữ liệu.*** Tiếp theo ta sẽ đi vào từng bước trong khai phá dữ liệu:

#### Thu thập dữ liệu

Bước này nhằm mục đích nghiên cứu, đưa ra các hướng giải pháp trong việc thu thập toàn bộ dữ liệu về thông tin phòng cho thuê từ các nguồn thông tin khác nhau trên Internet. Tiêu chí đặt ra trong bước này là không nhằm thu thập tối đa thông tin cần thiết tránh bỏ sót thông tin, sẽ ảnh hưởng đến quá quá trình xứ lý dữ liệu và khai phá.

#### Lựa chọn dữ liệu

Tại bước này chúng ta tìm kiếm thu thập lựa chọn những bộ dữ liệu về phòng với đầy đủ các thông tin mô tả đặc điểm của phòng, những đặc điểm mô tả này phải là những đặc điểm có ảnh hưởng tới định giá giá phòng. Những dữ liệu này phải là những bộ dữ liệu uy tín, chính xác, thể hiện được giá cả thị trường nhà đất. Nó được phân loại theo khu vực, diện tích, kiểu phòng trọ, kiểu phòng vệ sinh, có view như thế nào,… . Điều quan trọng nhất trong bước này là đánh giá, lựa chọn ra các thuộc tính có ảnh hưởng đến giá phòng và có khả năng mô hình hóa nó cho việc định giá.

#### Xử lý dữ liệu

Sau khi lựa chọn được bộ dữ liệu và khám phá nó để thấy được những đặc điểm của bộ dữ liệu chúng ta cần thực hiện các phương pháp để làm sạch dữ liệu, lọc bỏ dữ liệu nhiễu và impute missing value.

Lọc dữ liệu nhiễu (outlier) dữ liệu có những giá trị bất thường, (do giá trị đo của các trường hợp rất đặc biệt, do lỗi của công cụ đo lường,…). Lỗi này là lỗi khó phát hiện nhất, gây ra các sai sót nghiêm trọng làm sai lệch, giảm hiệu quả thực hiện các thuật toán khai phá.

Xử lý dữ liệu khuyết thiếu (impute missing value): là công việc chúng ta phải tìm một giá trị phù hợp để điền vào vị trí bị khuyết thiếu trong bộ dữ liệu. Có một số phương pháp để xử lý dữ liệu khuyết thiếu như:

* Một giá trị hằng có ý nghĩa trong miền xác định của dữ liệu, ví dụ như 0.
* Một giá trị của một đặc trưng từ một mẫu dữ liệu ngẫu nhiên trong tập dữ liệu.
* Các giá trị thống kê cơ bản như giá trị trung bình, giá trị trung vị hay giá trị mốt (mode) của cột.
* Một giá trị được ước lượng từ một mô hình dự đoán khác.

Mỗi phương pháp sẽ phù hợp với từng bộ dữ liệu và từng thuộc trường cụ thể.

#### Biến đổi dữ liệu

Biến đổi dữ liệu là việc chuyển toàn bộ tập giá trị của một thuộc tính sang một tập các giá trị thay thế sao ho mỗi giá trị cũ tương ứng với một giá trị mới. Trong bước này chúng ta lựa chọn và thực hiện các công việc tùy thuộc vào mục đích và đặc điểm của bộ dữ liêu để đưa dữ liệu về dạng chuẩn:

* Điều chỉnh lại tỉ lệ dữ liệu trên các đặc tính (Rescale)
* Chuẩn hóa dữ liệu (Standardization)
* Bình thường hóa dữ liệu (normalization)
* Số hóa dữ liệu (Digitalization)

Điều chỉnh tỷ lệ (rescale) Dữ liệu gồm nhiều đặc tính (cột), và mỗi đặc tính thì lại có các đơn vị và độ lớn nhỏ khác nhau. Điều này tác động tới tính hiệu quả của nhiều thuật toán, ví dụ thời gian thực hiện, quá trình hội tụ, hay thậm chí ảnh hưởng cả tới độ chính xác của thuật toán. Chính vì vậy, người ta thường tiến hành điều chỉnh dữ liệu để các đặc tính cùng có chung một tỉ lệ (data scaling). Và thường để các đặc tính có giá trị trong khoảng [0, 1]. Kết quả sẽ giúp cho nhiều thuật toán quan trọng trong Máy học sử dụng kĩ thuật Gradient Descent hội tụ nhanh. Việc điều chỉnh tỉ lệ thường dùng công thức sau đây (giả sử chúng ta đang làm trên một cột dữ liệu số cụ thể, gọi là F):

Trong đó: là gì trị ban đầu trong cột F

là giá trị sau khi được điều chỉnh

là gì trị nhỏ nhất trong cột F

là giá trị lớn nhất trong cột F

Phương pháp trên được gọi là MinMaxScaler sẽ đưa bộ dữ liệu của chúng ta về khoảng giá trị [0,1].

Chuẩn hóa dữ liệu (Standardize Data): Nhiều thuật toán trong Máy học giả định rằng dữ liệu đầu vào có phân phối Gauss. Chính vì vậy, khi chuẩn hóa dữ liệu về dạng chuẩn phân phối Gauss với giá trị trung bình bằng 0 và phương sai bằng 1. Nhờ việc chuẩn hóa, các thuật toán như linear regression, logistic regression được cải thiện. Công thức được sử dụng trong phương pháp này là :

Trong đó: giá trị trung bình

Độ lệch chuẩn :

Bình thường hóa dữ liệu (Normalize Data): Bình thường hóa dữ liệu là sự điều chỉnh tỉ lệ dữ liệu sao cho mỗi thể hiện (trên hàng) đều cho độ dài là 1. Kĩ thuật này rất cần thiết cho dữ liệu thưa (gồm nhiều số 0) trên mỗi cột đặc tính. Điều này đặc biệt ảnh hưởng tới các thuật toán lấy trọng số của các giá trị nhập vào như neuron networks, hay các thuật toán dùng độ đo khoảng cách (như k-Nearest Neighbors)

Số hóa dữ liệu (Digitalization): Rất nhiều thuật toán học máy hiện nay chỉ chấp nhận dữ liệu dạng số. Những những liệu dạng category chúng ta đều phải chuyển sang dạng số. Ở đây chúng ta có 3 phương pháp đễ mã hóa dữ liệu, chuyển dữ liệu category về dạng sô. Ví dụ một đặc tính có N giá trị khác nhau.

* Mã hoá một trong N (One of N encoding): sử dụng một tập gồm N các cột nguyên để chuẩn hoá, low (0, 0, 1), mid (0, 1, 0), high (1,0,0)
* Mã hoá nhị phân (Binary encoding): chuẩn hoá thành M cột nhị phân, với M l= [log2N]. low (0,0), mid (1,0), high (1,1)
* Mã hoá số nguyên (Numeric encoding): chuẩn hoá thành một cột có giá trị nguyên, mỗi giá trị của cột tương ứng với thứ tự của giá trị gốc có trong tập giá trị ban đầu: low (1), mid (2), high (3)

Bên cạnh đó, trong bước này chúng ta còn thực hiện rút gọn dữ liệu, lựa chọn các đặc trưng phù hợp để đưa vào mô hình hay còn gọi là feature selection

### Các phương pháp trong Khai phá dữ liệu

#### Phương pháp luật kết hợp

Một trong những chủ đề phổ biến của KPDL là khai phá luật kết hợp. Mục đích của khai phá luật kết hợp là xác định mối quan hệ, sự kết hợp giữa các mục dữ liệu (item) trong một CSDL lớn.

#### Phương pháp cây quyết định

Mô tả tri thức dạng đơn giản nhằm phân các đối tượng dữ liệu thành một số lớp nhất định. Các nút của cây được gán nhãn là tên các mục dữ liệu, các cạnh được gán các giá trị có thể của các mục dữ liệu, các lá mô tả các lớp khác nhau. Các đối tượng được phân lớp theo các đường đi trên cây, qua các cạnh tương ứng với các giá trị của mục dữ liệu tới lá.

#### Phương pháp K-Mean

Có nhiều phương pháp được sử dụng trong phân cụm, phương pháp k-Mean được coi là các kỹ thuật cơ bản của phân cụm. Với phương pháp này sẽ chia tập có n đối tượng thành k cụm sao cho các đối tượng trong cùng một cụm thì giống nhau, các đối tượng khác cụm thì khác nhau.

#### Các phương pháp dựa trên mẫu

Phương pháp này sử dụng khai phá chuỗi theo thời gian (Sequential temporal patterns). Xét về mặt kỹ thuật thì tương tự như KPDL bằng luật kết hợp nhưng có thêm tính thứ tự và tính thời gian. Một luật mô tả mẫu tuần tự có dạng tiêu biểu X -> Y phản ánh sự xuất hiện của biến cố X sẽ dẫn đến việc xuất hiện kế tiếp biến cố Y. Hướng tiếp cận này được ứng dụng nhiều trong lĩnh vực tài chính và thị trường chứng khoán bởi chúng có tính dự báo cao.

## Giới thiệu về Neural Network

Mạng nơ-ron (Neural Network) hay có tên gọi đầy đủ là Artificial Neural network (ANN) là một chuỗi các thuật toán cố gắng nhận ra các mối quan hệ cơ bản trong một tập hợp dữ liệu thông qua một quá trình bắt chước cách bộ não con người hoạt động. Theo nghĩa này, mạng nơ-ron đề cập đến hệ thống nơ-ron, có bản chất sinh học hoặc nhân tạo. Mạng nơron có thể thích ứng với việc thay đổi đầu vào; vì vậy mạng tạo ra kết quả tốt nhất có thể mà không cần thiết kế lại các tiêu chí đầu ra. Khái niệm về mạng thần kinh, có nguồn gốc từ [trí tuệ nhân tạo](https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp) ([**artificial intelligence**](https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp)), và ngày càng nhanh chóng trở nên phổ biến trong sự phát triển của các hệ thống giao dịch.

### Lịch sử phát triển

Cuối thế kỷ 19 đầu thế kỷ 20 sự phát triển chủ yếu chỉ là những công việc có sự tham gia của cả ba ngành Vật lý học, Tâm lý học, và thần kinh học, bởi các nhà khoa học như Hermann von Hemholtz, Ernst Mach, Ivan Pavlov. Các công trình nghiên cứu của họ chủ yếu đi sâu vào các lý thuyết tổng quát về HỌC, NHÌN, và lập luận,..và không đưa ra các mô hình toán học cụ thể mô tả hoạt động của các nơron.

Mọi chuyện thực sự bắt đầu vào những năm 1940 với công trình của Warren McCulloch và Walter Pitts. Họ chỉ ra rằng về nguyên tắc, mạng của các nơron nhân tạo có thể tính toán bất kỳ một hàm số học hay logic nào.

Tiếp theo là Donald Hebb, ông đã phát biểu rằng việc thuyết lập luận cổ điển (như Pavlov) đưa ra là hiện thực bởi do các thuộc tính của từng nơron riêng biệt, ông cũng nêu ra một phương pháp học của các nơron nhân tạo Ứng dụng thực nghiệm đầu tiên của các nơron nhân tạo có được vào cuối những năm 50 cùng với phát minh của mạng nhận thức và luật học tương ứng bởi Frank Rosenblatt. Mạng này có khả năng nhận dạng các mẫu. Điều này đã mở ra rất nhiều hy vọng cho việc nghiên cứu mạng nơron. Tuy nhiên nó có hạn chế là chỉ có thể giải quyết một số lớp hữu hạn các bài toán. Cùng thời gian đó, Bernard Widrow và Marcian Hoff đã đưa ra một thuật toán học mới và sử dụng nó để huấn luyện cho các mạng nơron tuyến tính thích nghi, mạng có cấu trúc và chức năng tương tự như mạng của Rosenblatt. Luật học Widrow – Hoff vẫn còn được sử dụng cho đến ngày nay.

Tuy nhiên cả Rosenblatt và Widrow – Hoff đều cùng vấp phải một vấn đề do Marvin Minsky và Seymour Papert phát hiện ra, đó là các mạng nhận thức chỉ có khả năng giải quyết được các bài toán tách được tuyến tính. Họ cố gắng cải tiến luật học và mạng để có thể vượt qua được hạn chế này nhưng họ đã không thành công trong việc cải tiến luậtt học để có thể huấn luyện được các mạng có cấu trúc phức tạp hơn. Do những kết quả của Minsky Papert nên việc nghiên cứu về mạng nơron gần như bị chững lại trong suốt một thập kỷ 70 do nguyên nhân là không có được các máy tính đủ mạnh để có thể thực nghiệm.

Mặc dù vậy, vẫn có một vài phát kiến quan trọng vào những năm 70. Năm 1972, Teuvo Kohonen và James anderson độc lập nhau phát triển một loại mạng mới có thể hoạt đông như một bộ nhớ. Stephen Grossberg cũng rất tích cực trong việc khảo sát các mạng tự tổ chức. Vào những năm 80, việc nghiên cứu mạng nơron phát triển rất mạnh mẽ cùng với sự ra đời của PC. Có hai khái niệm mới liên quan đến sự hồi sinh này, đó là:

• Việc sử dụng các phương pháp thống kê để giải thích hoạt động của một lớp các mạng hồi qui có thể được dùng như bộ nhớ liên hợp trong công trình của nhà vật lý học John Hopfield.

• Sự ra đời của thuật toán lan truyền ngược để luyện các mạng nhiều lớp được một vài nhà nghiên cứu độc lập tìm ra như: David Rumelhart, James Mc Celland,...đó cũng là câu trả lời cho Minsky-Papert.

### Ứng dụng

Ứng dụng Trong quá trình phát triển, mạng nơron đã được ứng dụng trong rất nhiều lĩnh vực:

• Tài chính: Định giá bất động sản, cho vay, kiểm tra tài sản cầm cố, đánh giá mức độ hợp tác, phân tích đường tín dụng, chương trình thương mại qua giấy tờ, phân tích tài chính liên doanh, dự báo tỷ giá tiền tệ.

• Ngân hàng: Bộ đọc séc và các tài liệu, tính tiền của thẻ tín dụng

• Giải trí: Hoạt hình, các hiệu ứng đặc biệt.

• Bảo hiểm: Đánh giá việc áp dụng chính sách, tối ưu hoá sản phẩm.

• Điện tử học: Dự báo mã tuần tự, sơ đồ chíp IC, điều khiển tiến trình, phân tích nguyên nhân hỏng chíp, nhận dạng tiếng nói, mô hình phi tuyến.

• Quốc phòng: Định vị - phát hiện vũ khí, dò mục tiêu, phát hiện đối tượng, nhận dạng nét mặt, các bộ cảm biến thế hệ mới, xử lý ảnh radar,..

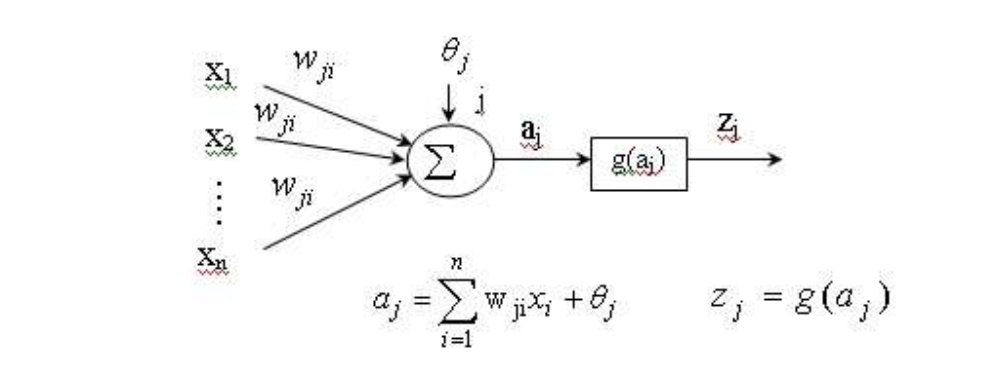
### Kiến thức cơ sở

Trong phần này sẽ trình bày một số khái niệm liên quan đến Neural Network.

Đây là nhưng kiến thức cơ bản nhất.

#### Neural

Một Neural còn được gọi là một đơn vị xử lý hay một nút (node), thực hiện một chức năng : nhận tín hiệu vào từ một nguồn bên ngoài hay từ các đơn vị phía trước và tính tín hiệu ra từ các tín hiệu vào sau đó lan truyền sang các đơn vị khác.



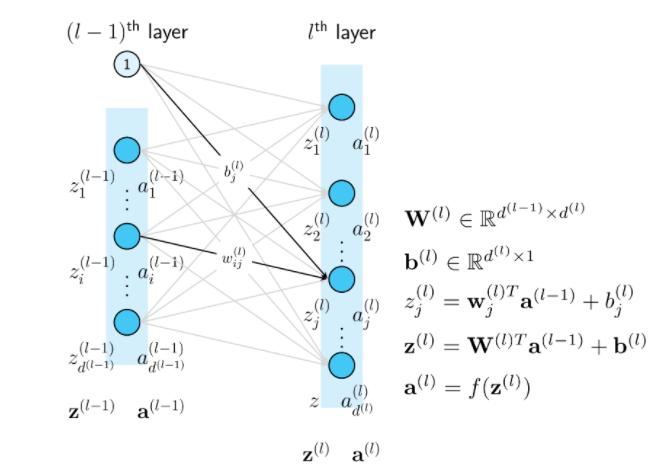
Hình 1.2. Mô hình đơn giản của một Artificial Neuron

Trong đó :

* xi : là các đầu vào
* wji : các trọng số tương ứng với các đầu vào
* θj : ngưỡng của nơron thứ j
* aj : tổng đầu vào của nơron thứ j (net input)
* zj : đầu ra của nơron thứ j
* g(.): hàm chuyển (hàm kích hoạt)

Một Neural trong mạng có thể có nhiều đầu vào (x1, x2,......,xn) nhưng chỉ có một đầu ra zj. Đầu vào của một nơron có thể là từ bên ngoài mạng, hoặc đầu ra của một nơron khác, hay là đầu ra của chính nó

Đầu vào của các hidden layer được ký hiệu bởi , đầu ra của mỗi unit thường được ký hiệu là  (thể hiện activation, tức giá trị của mỗi unit sau khi ta áp dụng activation function lên ). Đầu ra của unit thứ    trong layer thứ    được ký hiệu là . Giả sử thêm rằng số unit trong layer thứ  (không tính bias) là . Vector biểu diễn output của layer thứ    được ký hiệu là



Hình 1.3. Thông số đầu vào và ra giữa các Layer

Trong mạng neural có ba kiểu neural:

• Neural đầu vào, nhận tín hiệu từ bên ngoài

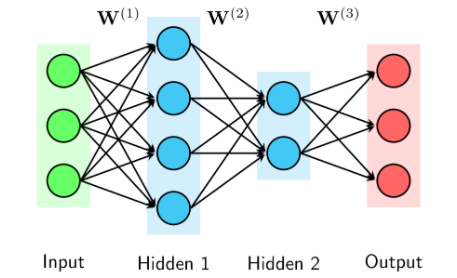
• Neural ẩn, tín hiệu vào và ra của nó nằm trong mạng

• Neural đầu ra, gửi tín hiệu ra bên ngoài

#### Layers

Layers (tầng) là tập hợp các node cùng vị trí trong mạng nơ-ron. Một mạng nơ-ron cơ bản bao gồm 3 lớp: *Input layer, Hidden layer, Output layer* . Đôi khi một Multi-layer Perceptron (MLP) có thể có nhiều *Hidden layers* ở giữa. Các *Hidden layers* theo thứ tự từ input layer đến output layer được đánh số thứ thự là *Hidden layer 1, Hidden layer 2,…*

Trong mạng nơ-ron, trừ input layer thì tất cả các node thuộc các layer khác đều full-connected với các node thuộc layer trước nó. Mỗi node thuộc hidden layer nhận vào ma trận đầu vào từ layer trước và kết hợp với trọng số để ra được kết quả.



Hình 1.4. Mô hình mạng MLP với 2 tầng ẩn

Số lượng layer trong một MLP được tính bằng số hidden layers cộng với 1. Tức là khi đếm số layers của một MLP, ta không tính input layers. Số lượng layer trong một MLP thường được ký hiệu là  . Trong hình trên,  =3.

#### Weights và Biases

Có  ma trận trọng số cho một MLP có  layers. Các ma trận này được ký hiệu là   trong đó  thể hiện các kết nối từ layer thứ  tới layer thứ  (nếu ta coi input layer là layer thứ 0). Cụ thể hơn, phần tử  thể hiện kết nối từ node thứ    của layer thứ  tới node từ    của layer thứ . Các biases của layer thứ  được ký hiệu là . Các trọng số này được ký hiệu như trên *Hình 1.3*. Khi tối ưu một MLP cho một công việc nào đó, chúng ta cần đi tìm các weights và biases này.

Tập hợp các weights và biases lần lượt được ký hiệu làvà .

#### Activation functions

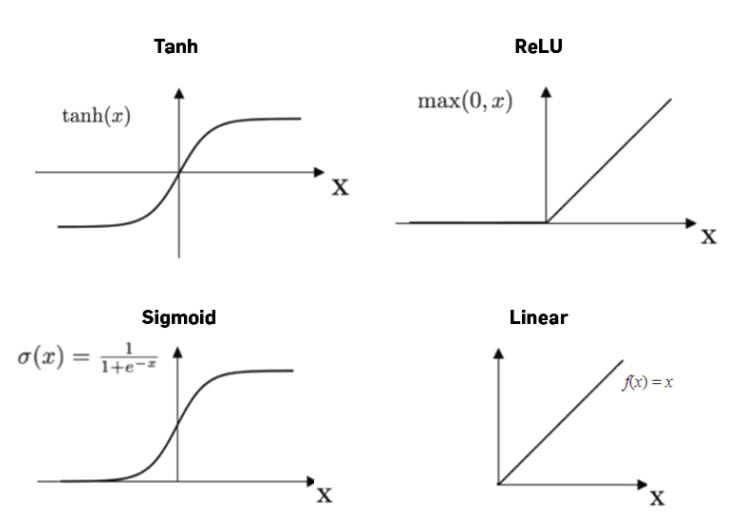
Trong mạng nơ-ron, một hàm kích hoạt chuẩn hóa đầu vào và tạo ra đầu ra, sau đó được chuyển tiếp vào lớp tiếp theo. Các chức năng kích hoạt thêm tính phi tuyến tính vào đầu ra cho phép mạng nơ-ron giải quyết các vấn đề phi tuyến tính. Nói cách khác, một mạng nơron không có chức năng kích hoạt về cơ bản chỉ là một mô hình hồi quy tuyến tính.

Mỗi output của một unit (trừ các input units) được tính dựa vào công thức:

Trong đó  là một (nonlinear) activation function. Ở dạng vector, biểu thức bên trên được viết là:

Khi activation function  được áp dụng cho một ma trận (hoặc vector), ta hiểu rằng nó được áp dụng cho từng thành phần của ma trận đó. Sau đó các thành phần này được sắp xếp lại đúng theo thứ tự để được một ma trận có kích thước bằng với ma trận input. Trong tiếng Anh, việc áp dụng lên từng phần tử như thế này được gọi là element-wise.

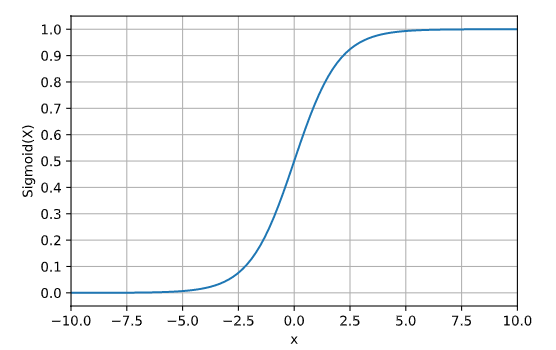
Một số hàm activation phổ biến **Linear**, **Sigmoid**, **Tanh**, and **ReLU**



Hình 1.5. Một số hàm Activation

* *Hàm Sigmoid*

Công thức :



Hình 1.6. Đồ thị hàm Sigmoid

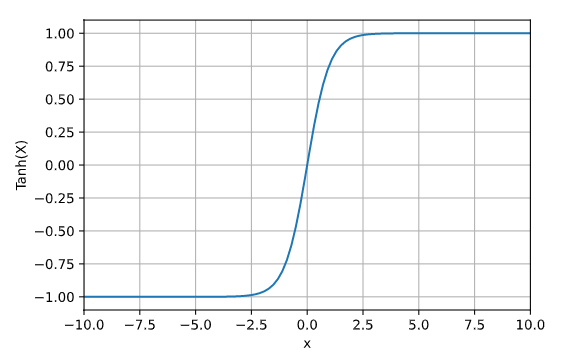
Hàm Sigmoid nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng (0;1) (xem đồ thị phía trên). Đầu vào là số thực âm rất nhỏ sẽ cho đầu ra tiệm cận với 0, ngược lại, nếu đầu vào là một số thực dương lớn sẽ cho đầu ra là một số tiệm cận với 1. Trong quá khứ hàm Sigmoid hay được dùng vì có đạo hàm rất đẹp. Tuy nhiên hiện nay hàm Sigmoid rất ít được dùng vì những nhược điểm sau:

* Hàm Sigmoid bão hào và triệt tiêu gradient**:** Một nhược điểm dễ nhận thấy là khi đầu vào có trị tuyệt đối lớn (rất âm hoặc rất dương), gradient của hàm số này sẽ rất gần với 0. Điều này đồng nghĩa với việc các hệ số tương ứng với unit đang xét sẽ gần như không được cập nhật (còn được gọi là *vanishing gradient*).
* **Hàm Sigmoid không có trung tâm là 0 gây khó khăn cho việc hội tụ.**

Để giải quyết yếu điểm này, chúng ta có thể giải quyết vấn đề này bằng cách chuẩn hoá dữ liệu về dạng có trung tâm là 0 (zero-centered) với các thuật toán batch/layer normalization.

* *Hàm Tanh*

Công thức :



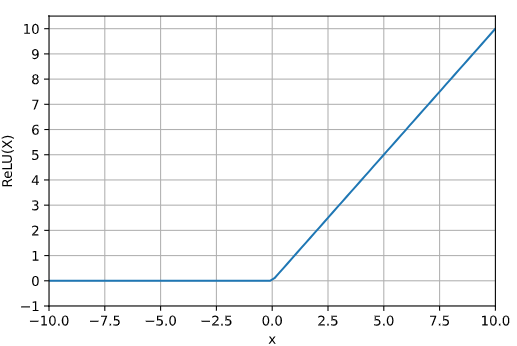
Hình 1.7. Đồ thị hàm Tang

Hàm tanh nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng (-1; 1). Cũng như Sigmoid, hàm bị bão hoà ở 2 đầu (gradient thay đổi rất ít ở 2 đầu). Tuy nhiên hàm lại đối xứng qua 0 nên khắc phục được một nhược điểm của Sigmoid.

Hàm  còn có thể được biểu diễn bằng hàm sigmoid như sau:

* *Hàm ReLU*

Công thức :



Hình 1.8. Đồ thị hàm ReLU

Hàm ReLU đang được sử dụng khá nhiều trong những năm gần đây khi huấn luyện các mạng neuron. ReLU đơn giản lọc các giá trị < 0. Đựa vào công thức ta có thể dễ dàng hiểu được cách hoạt động của nó. Một số ưu điểm và nhược điểm của ReLU so với Sigmoid và Tanh:

Ưu điểm :

* Tốc độ hội tụ nhanh hơn hẳn. ReLU có tốc độ hội tụ nhanh gấp 6 lần Tanh [5]. Điều này có thể do ReLU không bị bão hoà ở 2 đầu như Sigmoid và Tanh.
* Tính toán nhanh hơn. Tanh và Sigmoid sử dụng hàm exp và công thức phức tạp hơn ReLU rất nhiều do vậy sẽ tốn nhiều chi phí hơn để tính toán.

Nhược điểm :

* ReLU cũng có một nhược điểm: Với các node có giá trị nhỏ hơn 0, qua ReLU activation sẽ thành 0, hiện tượng đấy gọi là “Dying ReLU“. Nếu các node bị chuyển thành 0 thì sẽ không có ý nghĩa với bước linear activation ở lớp tiếp theo và các hệ số tương ứng từ node đấy cũng không được cập nhật với gradient descent.
* Khi learning rate lớn, các trọng số (weights) có thể thay đổi theo cách làm tất cả neuron dừng việc cập nhật.

Việc lựa chọn hàm activation cho mạng còn phụ thuộc vào nhiều yếu tố như : Độ sâu của mạng , output mong muốn , việc tìm hiều về các hàm activation sẽ phục vụ cho việc chọn hàm phù hợp khi ta xây dựng cho mô hình dự báo giá phòng được đề cập trong chương sau.

Một số chú ý về việc chọn activation function:

* Các hàm sigmoid và sự kết hợp của chúng thường phù hợp với những bài toán phân loại.
* Sigmoid và tanh đôi khi nên tránh sử dụng đồng thời vì có thể khiến gradient biến mất.
* ReLU là 1 activation function phổ biến và thường dùng nhất hiện nay.
* ReLU function chỉ có thể được sử dụng trong những hidden layer

Trên đây là kiến thức cơ bản, tổng hợp về mạng nơ-ron. Nó là tiền đề cho chúng ta nghiên cứu các mô hình học sâu ngay sau. Tóm tắt lại, mạng nơ-ron là nền tảng của học sâu, được sử dụng rộng rãi, với các ứng dụng cho hoạt động tài chính, lập kế hoạch doanh nghiệp, giao dịch, phân tích kinh doanh và bảo trì sản phẩm. Mạng nơ-ron cũng đã được áp dụng rộng rãi trong các ứng dụng kinh doanh như các giải pháp nghiên cứu tiếp thị và dự báo, phát hiện gian lận và [đánh giá rủi ro](https://www.investopedia.com/terms/r/risk-assessment.asp).

Dựa vào kiến thức hàm activation ta có thể đánh giá chung và rút ra một vài kết luận cho các bài toán :

* ***Vấn đề hồi quy***

Một vấn đề trong đó bạn dự đoán một đại lượng có giá trị thực.

* Cấu hình lớp đầu ra : Một nút với đơn vị kích hoạt tuyến tính.
* Chức năng mất : Lỗi bình phương trung bình (MSE).
* ***Vấn đề phân loại nhị phân***

Một vấn đề mà bạn phân loại một ví dụ thuộc một trong hai lớp. Vấn đề được đóng khung là dự đoán khả năng xảy ra một ví dụ thuộc về lớp một, ví dụ như lớp mà bạn gán giá trị nguyên 1, trong khi lớp khác được gán giá trị 0.

* Cấu hình lớp đầu ra : Một nút với đơn vị kích hoạt Sigmoid.
* Hàm mất mát : Cross-Entropy, còn được gọi là mất mát Logarit.
* ***Vấn đề phân loại nhiều lớp***

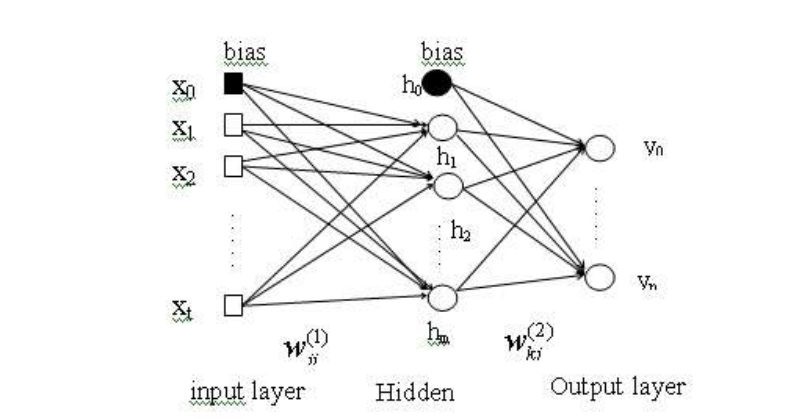
Một vấn đề trong đó bạn phân loại một ví dụ thuộc một trong nhiều hơn hai lớp.

Vấn đề được đóng khung là dự đoán khả năng xảy ra một ví dụ thuộc về mỗi lớp.

* Cấu hình lớp đầu ra: Một nút cho mỗi lớp sử dụng chức năng kích hoạt Softmax.
* Hàm mất mát : Cross-Entropy, còn được gọi là mất mát Logarit.

#### Mạng truyền thẳng

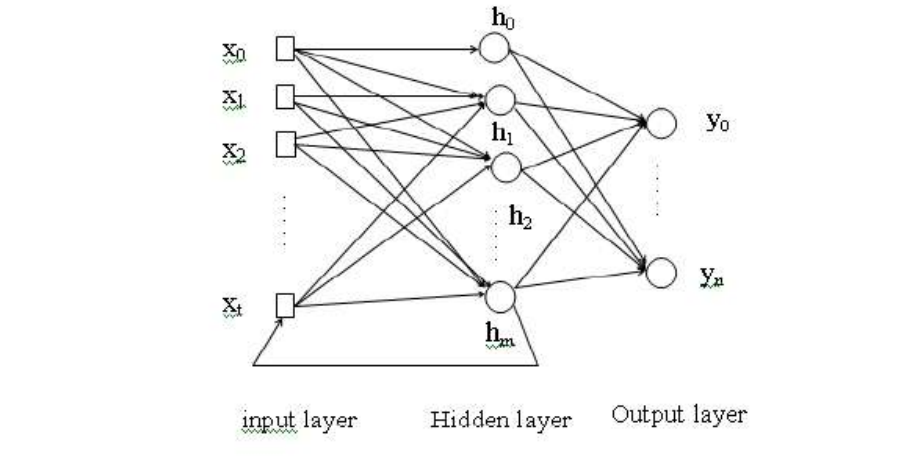
Dòng dữ liệu từ nơron đầu vào đến nơron đầu ra chỉ được truyền thẳng. Không có các liên kết từ nơron đầu ra đến các nơron đầu vào trong cùng một lớp hay của các lớp trước đó.



Hình 1.9. Mô hình mạng neuron truyền thẳng

#### Mạng hồi quy

Có chứa các liên kết ngược, nghĩa là các nơron lớp ra tới nơron lớp vào trong cùng một lớp hoặc các lớp trước đó.



Hình 1.10. Mô hình mạng neuron hồi quy

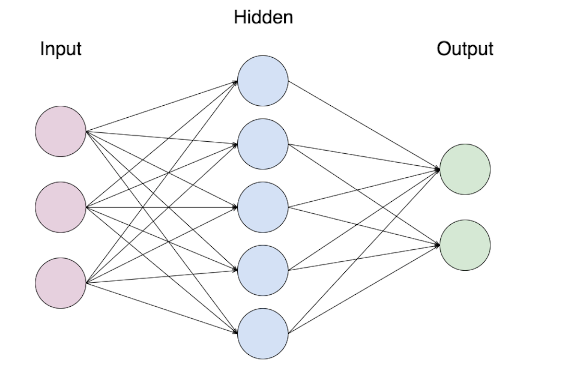
### Mô hình Neural Network

Mạng nơ-ron (Neural Network) hoạt động tương tự như mạng lưới thần kinh não bộ của con người. Một "nơ-ron" trong mạng nơ-ron là một hàm toán học thu thập và phân loại thông tin theo một kiến ​​trúc cụ thể. Mạng này khá tương đồng mạnh mẽ với các phương pháp thống kê như phân tích đường cong và phân tích hồi quy.

Một mạng nơ-ron chứa các lớp (layer) là tập hợp các nút được kết nối với nhau. Mỗi nút là một perceptron và tương tự như một [hồi quy tuyến tính bội](https://www.investopedia.com/terms/m/mlr.asp) ( [multiple linear regression](https://www.investopedia.com/terms/m/mlr.asp)). Perceptron cung cấp tín hiệu được tạo ra bởi một hồi quy nhiều tuyến tính thành một hàm kích hoạt có thể là phi tuyến.

Trong một perceptron nhiều lớp (Multi-layer Perceptron), các perceptron được sắp xếp thành các lớp liên kết với nhau. Lớp đầu vào thu thập các mẫu đầu vào. Lớp đầu ra có các phân loại hoặc tín hiệu đầu ra mà các mẫu đầu vào có thể ánh xạ. Ví dụ, các mẫu có thể bao gồm danh sách [các chỉ số kỹ thuật](https://www.investopedia.com/terms/t/technicalindicator.asp) về chứng khoán; đầu ra mong muốn có thể là “mua”, “giữ” hoặc “bán”.

Các lớp ẩn sẽ tinh chỉnh trọng số đầu vào cho đến khi biên độ lỗi của mạng nơ-ron là nhỏ nhất. Giả thuyết rằng các lớp ẩn ngoại suy các đặc điểm nổi bật trong dữ liệu đầu vào có khả năng dự đoán liên quan đến kết quả đầu ra. Điều này mô tả việc trích xuất tính năng, hoàn thành một tiện ích tương tự như các kỹ thuật thống kê, chẳng hạn như phân tích thành phần chính.



Hình 1.11. Kiến trúc cơ bản của Neural Network

* Input layer : Là tầng bao gồm các giá trị đầu vào của mạng.
* Hidden layer : Có thể gồm 1 hoặc nhiều tầng, các tầng này đóng vai trò như các logic, suy luận của mạng.
* Output layer : Thể hiện kết quả đầu ra của mạng.

## Giới thiệu về Học sâu

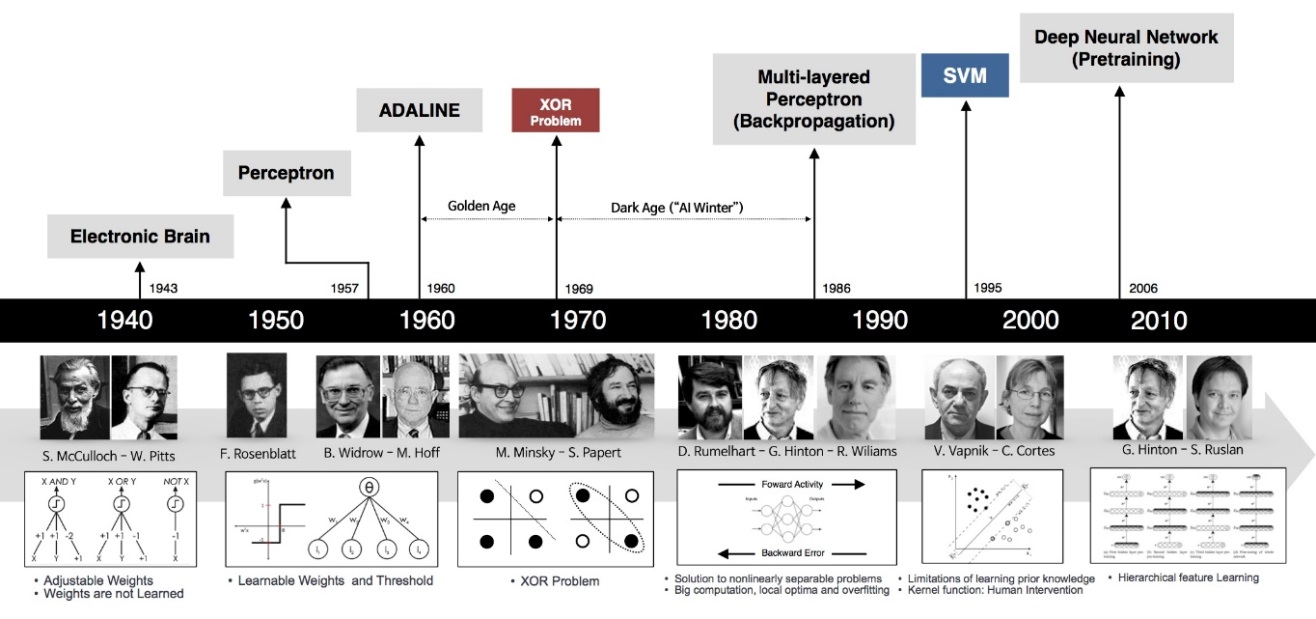
Những năm gần đây, AI - Artificial Intelligence (Trí Tuệ Nhân Tạo), và cụ thể hơn là Machine Learning (Học Máy hoặc Máy Học) nổi lên như một bằng chứng của cuộc cách mạng công nghiệp lần thứ tư (1 - động cơ hơi nước, 2 - năng lượng điện, 3 - công nghệ thông tin). Trí Tuệ Nhân Tạo đang len lỏi vào mọi lĩnh vực trong đời sống mà có thể chúng ta không nhận ra. Xe tự hành của Google và Tesla, hệ thống tự tag khuôn mặt trong ảnh của Facebook, trợ lý ảo Siri của Apple, hệ thống gợi ý sản phẩm của Amazon, hệ thống gợi ý phim của Netflix, máy chơi cờ vây AlphaGo của Google DeepMind, …, chỉ là một vài trong vô vàn những ứng dụng của AI/Machine Learning.

Học sâu ([tiếng Anh](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh): deep learning) là một chi của ngành [máy học](https://vi.wikipedia.org/wiki/M%C3%A1y_h%E1%BB%8Dc) dựa trên một tập hợp các thuật toán để cố gắng mô hình dữ liệu trừu tượng hóa ở mức cao bằng cách sử dụng nhiều lớp xử lý với cấu trúc phức tạp, hoặc bằng cách khác bao gồm nhiều biến đổi phi tuyến.

Nhiều kiến trúc học sâu khác nhau như [mạng neuron sâu](https://vi.wikipedia.org/w/index.php?title=M%E1%BA%A1ng_n%C6%A1-ron_s%C3%A2u&action=edit&redlink=1), [mã mạng neuron tích chập sâu](https://vi.wikipedia.org/w/index.php?title=Convolutional_neuron_network&action=edit&redlink=1), [mạng niềm tin sâu](https://vi.wikipedia.org/w/index.php?title=Deep_belief_network&action=edit&redlink=1) và [mạng neuron tái phát](https://vi.wikipedia.org/w/index.php?title=Recurrent_neuron_network&action=edit&redlink=1) đã được áp dụng cho các lĩnh vực như [thị giác máy tính](https://vi.wikipedia.org/wiki/Th%E1%BB%8B_gi%C3%A1c_m%C3%A1y_t%C3%ADnh), [tự động nhận dạng giọng nói](https://vi.wikipedia.org/wiki/Nh%E1%BA%ADn_d%E1%BA%A1ng_ti%E1%BA%BFng_n%C3%B3i), [xử lý ngôn ngữ tự nhiên](https://vi.wikipedia.org/wiki/X%E1%BB%AD_l%C3%BD_ng%C3%B4n_ng%E1%BB%AF_t%E1%BB%B1_nhi%C3%AAn), nhận dạng âm thanh ngôn ngữ và [tin sinh học](https://vi.wikipedia.org/wiki/Tin_sinh_h%E1%BB%8Dc), chúng đã được chứng minh là tạo ra các kết quả rất tốt đối với nhiều nhiệm vụ khác nhau.

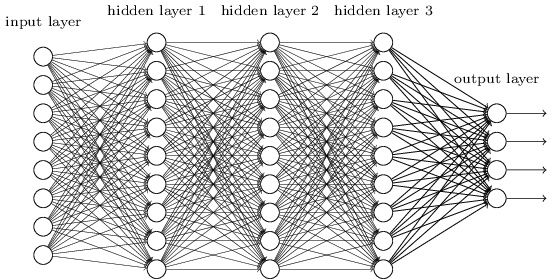
Các thuật toán học sâu tương phản với các thuật toán học nông bởi số biến đổi được tham số hóa một tín hiệu gặp phải khi nó lan truyền từ các lớp đầu vào đến lớp đầu ra, nơi một biến đổi được tham số hóa là một đơn vị xử lý có các thông số có thể huấn luyện được, chẳng hạn như trọng số và ngưỡng. Một chuỗi các biến đổi từ đầu vào đến đầu ra là một *đường gán kế thừa* (CAP- credit assignment path). CAP mô tả các kết nối quan hệ nhân quả tiềm năng giữa đầu vào và đầu ra và có thể thay đổi chiều dài. Đối với một mạng neuron nuôi tiến (feedforward), độ sâu của CAP, và do đó độ sâu của mạng đó, là số lượng các lớp ẩn cộng 1 (lớp đầu ra cũng là tham số hóa). Đối với [mạng neuron tái phát](https://vi.wikipedia.org/w/index.php?title=Recurrent_neuron_network&action=edit&redlink=1), trong đó một tín hiệu có thể truyền thông qua một lớp nhiều hơn một lần, CAPcó khả năng không bị giới hạn chiều dài. Không có sự thống nhất chung về ngưỡng của độ sâu chia học cạn với học sâu, nhưng hầu hết các nhà nghiên cứu trong lĩnh vực đồng ý rằng học sâu có nhiều lớp phi tuyến (CAP > 2) và Schmidhuber coi CAP > 10 để là học rất sâu

Lịch sử phát triển của Deep Learning:



Hình 1.12. Lịch sử phát triển của Deep Learning

Mạng neuron sâu (DNN-Deep neural Network) là một [mạng neuron nhân tạo](https://vi.wikipedia.org/wiki/M%E1%BA%A1ng_n%C6%A1-ron_nh%C3%A2n_t%E1%BA%A1o) (ANN) với nhiều đơn vị lớp ẩn giữa lớp đầu vào và đầu ra. Tương tự như các ANN, các DNN nông có thể mô hình mối quan hệ phi tuyến phức tạp. Các kiến trúc DNN, ví dụ như để phát hiện và [phân tích](https://vi.wikipedia.org/wiki/X%E1%BB%AD_l%C3%BD_ng%C3%B4n_ng%E1%BB%AF_t%E1%BB%B1_nhi%C3%AAn) đối tượng tạo ra các mô hình hỗn hợp trong đó đối tượng này được thể hiện như một thành phần được xếp lớp của các hình ảnh nguyên thủy. Các lớp phụ cho phép các thành phần của các đặc điểm từ các lớp thấp hơn, đem lại tiềm năng của mô hình hóa dữ liệu phức tạp với các đơn vị ít hơn so với một mạng lưới nông thực hiện tương tự như vậy.



Hình 1.13. Mô hình Deep Neural Network

Deep Neural Network được xây dựng với mục đích mô phỏng hoạt động não bộ phức tạp của con người và được áp dụng vào nhiều lĩnh vực khác nhau, mang lại thành công và những hiệu quả đáng kinh ngạc cho con người*.* Deep Learning thường yêu cầu lượng dữ liệu lớn và nguồn tài nguyên sử dụng nhiều hơn các phương pháp thông thường, tuy nhiên cho độ chính xác cao hơn.

Mạng neuron sâu thường được giải thích theo cách: định lý xấp xỉ tổng quát hoặc [Suy luận xác suất](https://vi.wikipedia.org/wiki/Suy_lu%E1%BA%ADn_Bayes).

Định lý xấp xỉ phổ quát : [Định lý xấp xỉ phổ quát](https://vi.wikipedia.org/w/index.php?title=Universal_approximation_theorem&action=edit&redlink=1) đề cập đến khả năng của [mạng neuron tiến tiếp](https://vi.wikipedia.org/w/index.php?title=Feedforward_neuron_networks&action=edit&redlink=1) (feedforward) với một lớp ẩn có kích thước hữu hạn đơn để xấp xỉ các [hàm liên tục](https://vi.wikipedia.org/wiki/H%C3%A0m_li%C3%AAn_t%E1%BB%A5c).

Năm 1989, là bằng chứng đầu tiên được xuất bản bởi George Cybenko cho các hàm kích hoạt [h](https://vi.wikipedia.org/w/index.php?title=Sigmoid&action=edit&redlink=1)ình sigma và được mở rộng đối với các kiến trúc nuôi tiến nhiều lớp vào năm 1991 bởi Kurt Hornik

Định lý [Suy luận xác suất](https://vi.wikipedia.org/wiki/Suy_lu%E1%BA%ADn_Bayes): Diễn giải [xác suất](https://vi.wikipedia.org/wiki/X%C3%A1c_su%E1%BA%A5t) bắt nguồn từ lĩnh vực [máy học](https://vi.wikipedia.org/wiki/M%C3%A1y_h%E1%BB%8Dc). Nó có đặc điểm suy luận, cũng như các khái niệm [tối ưu hóa](https://vi.wikipedia.org/wiki/T%E1%BB%91i_%C6%B0u_h%C3%B3a_(to%C3%A1n_h%E1%BB%8Dc)) [huấn luyện](https://vi.wikipedia.org/wiki/%C4%90%C3%A0o_t%E1%BA%A1o) và và [kiểm tra](https://vi.wikipedia.org/w/index.php?title=Test_(assessment)&action=edit&redlink=1), liên quan đến việc phù hợp và [tổng quát hóa](https://vi.wikipedia.org/wiki/Generalization) tương ứng. Cụ thể hơn, diễn giải xác suất sẽ xem xét kích hoạt một cách phi tuyến như là một [hàm phân phối tích lũy](https://vi.wikipedia.org/wiki/H%C3%A0m_ph%C3%A2n_ph%E1%BB%91i_t%C3%ADch_l%C5%A9y). Xem mạng tin sâu.Diễn giải xác suất dẫn đến sự ra đời của dropout như regularizer trong mạng neuron.

Diễn giải xác suất đã được giới thiệu và phổ biến rộng rãi bởi những tiên phong như [Geoff Hinton](https://vi.wikipedia.org/w/index.php?title=Geoff_Hinton&action=edit&redlink=1), [Yoshua Bengio](https://vi.wikipedia.org/w/index.php?title=Yoshua_Bengio&action=edit&redlink=1), [Yann Le Cun](https://vi.wikipedia.org/w/index.php?title=Yann_Le_Cun&action=edit&redlink=1), [Juergen Schmidhuber](https://vi.wikipedia.org/w/index.php?title=Juergen_Schmidhuber&action=edit&redlink=1).

## Các mô hình, kiến trúc học sâu

Có một lượng rất lớn các biến thể của kiến trúc học sâu. Hầu hết chúng là nhánh sinh ra từ một số kiến trúc cha ban đầu. Không phải là luôn luôn có thể so sánh hiệu suất của nhiều kiến trúc cùng với nhau, vì chúng không phải là tất cả đánh giá trên cùng một tập dữ liệu. Học sâu học là một lĩnh vực phát triển nhanh, và các kiến trúc, biến thể, hoặc các thuật toán mới xuất hiện mỗi vài tuần.

### Multilayer Perceptrons (MLP)

Multilayer Perceptron, hay viết tắt là MLP, là kiểu mạng nơ-ron cổ điển.

Chúng bao gồm một hoặc nhiều lớp tế bào thần kinh. Dữ liệu được đưa đến lớp đầu vào (input), có thể có một hoặc nhiều lớp ẩn cung cấp mức độ trừu tượng và các dự đoán được thực hiện trên lớp đầu ra, còn được gọi là lớp hiển thị.

MLP thích hợp cho các vấn đề dự đoán phân loại trong đó đầu vào được gán một lớp hoặc nhãn.

Chúng cũng thích hợp cho các bài toán dự đoán hồi quy trong đó một đại lượng có giá trị thực được dự đoán với một tập hợp các đầu vào. Dữ liệu thường được cung cấp ở định dạng bảng, chẳng hạn như bạn sẽ thấy trong tệp CSV hoặc bảng tính.

**Sử dụng MLP cho:**

* Bộ dữ liệu dạng bảng
* Các vấn đề dự đoán phân loại
* Các vấn đề về dự đoán hồi quy

Chúng rất linh hoạt và có thể được sử dụng chung để học cách ánh xạ từ đầu vào đến đầu ra.

Tính linh hoạt này cho phép chúng được áp dụng cho các loại dữ liệu khác. Ví dụ: các pixel của hình ảnh có thể được giảm xuống thành một hàng dài dữ liệu và được đưa vào MLP. Các từ của một tài liệu cũng có thể được rút gọn thành một hàng dài dữ liệu và được đưa vào MLP. Ngay cả những quan sát về độ trễ đối với vấn đề dự đoán chuỗi thời gian cũng có thể được giảm xuống thành một hàng dài dữ liệu và được cung cấp cho MLP.

Chẳng hạn như hình ảnh, tài liệu hoặc chuỗi thời gian, chúng ta hoàn toàn có thể thử nghiệm MLP cho vấn đề. Kết quả có thể được sử dụng làm điểm cơ sở so sánh để xác nhận rằng các mô hình khác có thể xuất hiện giá trị gia tăng phù hợp hơn.

### Convolutional Neural Networks (CNN)

Một CNN gồm có một hoặc nhiều hơn các lớp [tích chập](https://vi.wikipedia.org/wiki/T%C3%ADch_ch%E1%BA%ADp) với các lớp đầy đủ kết nối (đáp ứng phù hợp với những mạng neuron nhân tạo tiêu biểu) trên đỉnh. Nó cũng sử dụng trọng số gắn liền và các lớp thăm dò. Kiến trúc này cho phép các CNN tận dụng lợi thế của cấu trúc 2D của dữ liệu đầu vào. So với những kiến trúc sâu khác, mạng neuron tích chập đang bắt đầu thể hiện kết quả vượt trội trong các ứng dụng hình ảnh và giọng nói. Chúng cũng có thể được huấn luyện với tiêu chuẩn truyền ngược. CNN dễ dàng được đào tạo hơn các mạng neural sâu nuôi tiến thông thường khác, và có ít thông số ước tính hơn, khiến cho chúng trở thành một kiến trúc rất hấp dẫn để sử dụng.

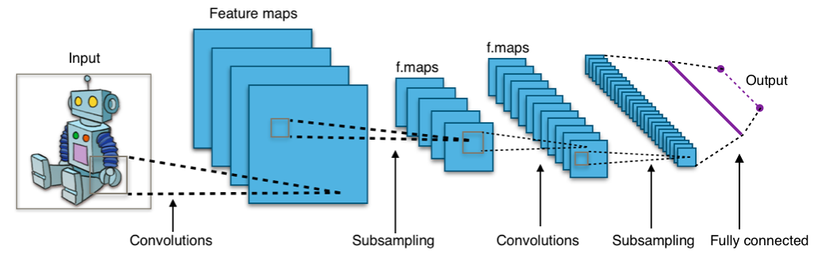
Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Trong mô hình mạng truyền ngược (feedforward neural network) thì mỗi neural đầu vào (input node) cho mỗi neural đầu ra trong các lớp tiếp theo.

Mô hình này gọi là mạng kết nối đầy đủ (fully connected layer) hay mạng toàn vẹn (affine layer). Còn trong mô hình CNNs thì ngược lại. Các layer liên kết được với nhau thông qua cơ chế convolution.

Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó.

Mỗi một lớp được sử dụng các filter khác nhau thông thường có hàng trăm hàng nghìn filter như vậy và kết hợp kết quả của chúng lại. Ngoài ra có một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu).

Trong quá trình huấn luyện mạng (traning) CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà bạn thực hiện. Ví dụ trong tác vụ phân lớp ảnh, CNNs sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự raw pixel > edges > shapes > facial > high-level features. Layer cuối cùng được dùng để phân lớp ảnh.



Hình 1.14. Mô hình kiến trúc mạng CNN

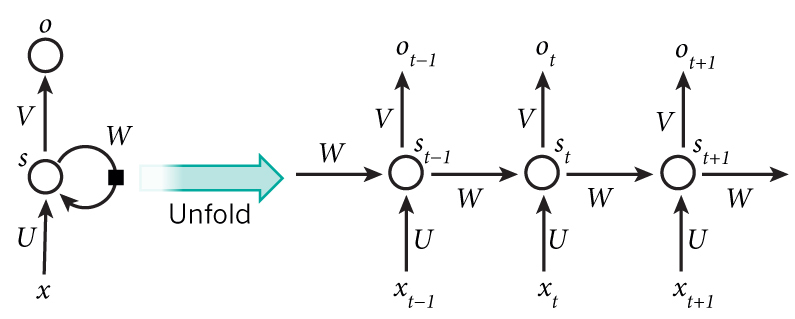
Lợi ích của việc sử dụng CNN là khả năng phát triển một hình ảnh hai chiều bên trong của họ. Điều này cho phép mô hình tìm hiểu vị trí và tỷ lệ trong các cấu trúc biến thể trong dữ liệu, điều này rất quan trọng khi làm việc với hình ảnh.

**Sử dụng CNN cho:**

* Dữ liệu hình ảnh
* Các vấn đề dự đoán phân loại
* Các vấn đề về dự đoán hồi quy

### Recurrent Neural Network (RNN)

RNN (Recurrent Neural Network) là sử dụng chuỗi các thông tin. Trong các mạng nơ-ron truyền thống tất cả các đầu vào và cả đầu ra là độc lập với nhau. Tức là chúng không liên kết thành chuỗi với nhau. Nhưng các mô hình này không phù hợp trong rất nhiều bài toán. Ví dụ, nếu muốn đoán từ tiếp theo có thể xuất hiện trong một câu thì ta cũng cần biết các từ trước đó xuất hiện lần lượt thế nào chứ nhỉ? RNN được gọi là hồi quy (Recurrent) bởi lẽ chúng thực hiện cùng một tác vụ cho tất cả các phần tử của một chuỗi với đầu ra phụ thuộc vào cả các phép tính trước đó. Nói cách khác, RNN có khả năng nhớ các thông tin được tính toán trước đó. Trên lý thuyết, RNN có thể sử dụng được thông tin của một văn bản rất dài, tuy nhiên thực tế thì nó chỉ có thể nhớ được một vài bước trước đó (ta cùng bàn cụ thể vấn đề này sau) mà thôi. Về cơ bản một mạng RNN có dạng như sau:



Hình 1.15. Mô hình kiến trúc mạng RNN

Huấn luyện mạng RNN cũng tương tự như các mạng nơ-ron truyền thống, tuy nhiên giải thuật lan truyền ngược (backpropagation) phải thay đổi một chút. Đạo hàm tại mỗi đầu ra phụ thuộc không chỉ vào các tính toán tại bước đó, mà còn phụ thuộc vào các bước trước đó nữa, vì các tham số trong mạng RNN được sử dụng chung cho tất cả các bước trong mạng. Ví dụ, để tính đạo hàm tại t = 4 ta phải lan truyền ngược cả 3 bước phía trước rồi cộng tổng đạo hàm của chúng lại với nhau. Việc tính đạo hàm kiểu này được gọi là lan truyền ngược liên hồi ([BPTT](https://en.wikipedia.org/wiki/Backpropagation_through_time) - Backpropagation Through Time). Còn giờ, chỉ cần nhớ rằng với các bước phụ thuộc càng xa thì việc học sẽ [càng khó khăn hơn](https://arxiv.org/pdf/1211.5063v2.pdf) vì sẽ xuất hiện vấn đề hao hụt/bùng nổ (vanishing/exploding) của đạo hàm. Có một vài phương pháp được đề xuất để giải quyết vấn đề này và các kiểu mạng RNN hiện nay đã được thiết kế để triệt tiêu bớt chúng như LSTM chẳng hạn.

RNN nói chung và LSTM nói riêng đã nhận được nhiều thành công nhất khi làm việc với các chuỗi từ và đoạn văn, thường được gọi là xử lý ngôn ngữ tự nhiên.

Điều này bao gồm cả chuỗi văn bản và chuỗi ngôn ngữ nói được biểu thị dưới dạng chuỗi thời gian. Chúng cũng được sử dụng như các mô hình tổng quát yêu cầu đầu ra theo trình tự, không chỉ với văn bản, mà còn trên các ứng dụng như tạo chữ viết tay.

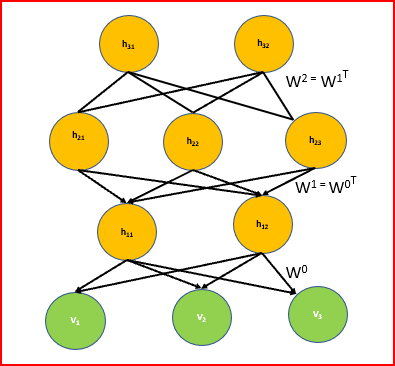
**Sử dụng RNN cho:**

* Dữ liệu văn bản
* Dữ liệu giọng nói
* Các vấn đề dự đoán phân loại
* Các vấn đề về dự đoán hồi quy
* Các mô hình tạo ra

### Deep Belief Network (DBN)

Một mạng niềm tin sâu (DBN) là một mô hình xác suất [thể sinh](https://vi.wikipedia.org/w/index.php?title=Generative_model&action=edit&redlink=1), tạo thành bởi nhiều đơn vị ẩn nhiều lớp. Nó có thể được coi là một [hàm hợp](https://vi.wikipedia.org/wiki/H%C3%A0m_h%E1%BB%A3p) các mô-đun học đơn giản tạo thành mỗi lớp.

Một DBN có thể được sử dụng để huấn luyện trước khả sinh một DNN bằng cách sử dụng các trọng số DBN học như các trọng số DNN ban đầu. Các thuật toán truyền ngược hoặc suy xét khác sau đó có thể được áp dụng để điều chỉnh những trọng số này. Điều này đặc biệt hữu ích khi dữ liệu đào tạo giới hạn là có sẵn, vì các trọng số khởi tạo nghèo nàn có thể cản trở đáng kể hiệu suất của mô hình được học. Các trọng số đào tạo trước này là một vùng không gian trọng số là gần gũi hơn với trọng số tối ưu hơn là các trọng số ban đầu được chọn ngẫu nhiên. Điều này cho phép cả mô hình hóa được cải thiện và hội tụ tinh chỉnh pha nhanh hơn.



Hình 1.16. Mô hình kiến trúc DBN

### Deep Predictive coding network (DPCN)

Có những lợi thế của một mô hình mà có thể chủ động cập nhật bản thân từ ngữ cảnh trong dữ liệu. Mạng lập trình (DPCN) là một chương trình lập trình [tiên đoán](https://vi.wikipedia.org/w/index.php?title=Predictive_modelling&action=edit&redlink=1), trong đó thông tin từ trên xuống được sử dụng để điều chỉnh theo kinh nghiệm của những cái trước đó cần thiết cho một thủ tục [suy luận](https://vi.wikipedia.org/wiki/Suy_lu%E1%BA%ADn) từ dưới lên bằng các phương tiện của một [mô hình thể sinh](https://vi.wikipedia.org/w/index.php?title=M%C3%B4_h%C3%ACnh_th%E1%BB%83_sinh&action=edit&redlink=1) kết nối cục bộ sâu. Điều này hoạt động bằng cách chiết tách các đặc điểm rời rạc các quan sát biến đổi theo thời gian bằng cách sử dụng một mô hình động học tuyến tính. Sau đó, một chiến lược thăm dò được sử dụng để học các đại diện đặc điểm bất biến. Các đơn vị này tập hợp lại để tạo thành một kiến trúc sâu và được huấn luyện bởi [học không giám sát](https://vi.wikipedia.org/wiki/H%E1%BB%8Dc_kh%C3%B4ng_c%C3%B3_gi%C3%A1m_s%C3%A1t) layer-wise [tham lam](https://vi.wikipedia.org/wiki/Gi%E1%BA%A3i_thu%E1%BA%ADt_tham_lam). Các lớp tạo thành một loại [xích Markov](https://vi.wikipedia.org/wiki/X%C3%ADch_Markov) mà các trạng thái tại bất kỳ lớp nào cũng chỉ phụ thuộc vào các lớp trước và các lớp sau (kế thừa).

Mạng lập tình dự đoán sâu (DPCN) dự đoán đại diện của lớp, bằng cách sử dụng một cách tiếp cận từ trên xuống bằng cách sử dụng thông tin ở lớp trên và các phụ thuộc thời gian từ các trạng thái trước đó.

## Tổng kết chương

Như vậy trong chương này chúng ta có được cái nhìn tổng thể về neural network và học sâu, một số mô hình học sâu. Từ đó có thể phát triển hướng nghiên cứu tìm ra phương pháp để xây dựng bài toán dự báo giá phòng cho thuê.

# MÔ HÌNH HỌC MÁY CHO ĐỊNH GIÁ PHÒNG

Hiện nay, đã có nhiều đề tài đề cập đến các phương pháp giải quyết vấn đề cho bài toán định giá hồi quy. Trong đó nổi bật phải kể đế một vài thuật toán cho bài toán hồi quy như : Linear regression, KNN, SVR, Decision Tree. Trong phần tiếp theo sẽ nghiên cứu, đánh giá các thuật toán trên từ đó ta có thể rút ra những ưu nhược điểm của nó so với phương pháp học sâu đã đề cập ở trên:

## Mô hình Linear Regression

"Hồi quy tuyến tính" là một phương pháp thống kê để hồi quy dữ liệu với biến phụ thuộc có giá trị liên tục trong khi các biến độc lập có thể có một trong hai giá trị liên tục hoặc là giá trị phân loại. Nói cách khác "Hồi quy tuyến tính" là một phương pháp để dự đoán biến phụ thuộc (Y) dựa trên giá trị của biến độc lập (X). Nó có thể được sử dụng cho các trường hợp chúng ta muốn dự đoán một số lượng liên tục.

Một ví dụ liên quan đến

Một căn nhà rộng , có phòng ngủ và cách trung tâm thành phố  km có giá là bao nhiêu. Giả sử chúng ta đã có số liệu thống kê từ 1000 căn nhà trong thành phố đó, liệu rằng khi có một căn nhà mới với các thông số về diện tích, số phòng ngủ và khoảng cách tới trung tâm, chúng ta có thể dự đoán được giá của căn nhà đó không? Nếu có thì hàm dự đoán sẽ có dạng như thế nào. Ở đây là một vector hàng chứa thông tin *input*, ***y*** là một số vô hướng (scalar) biểu diễn *output* (tức giá của căn nhà trong ví dụ này).

Ta có thể suy luận đơn giản nhất, chúng ta thấy:

* Diện tích nhà càng lớn thì giá nhà càng cao;
* Số lượng phòng ngủ càng lớn thì giá nhà càng cao;
* Càng xa trung tâm thì giá nhà càng giảm.

Một hàm số đơn giản nhất có thể mô tả mối quan hệ giữa giá phòng và 3 đại lượng đầu vào là:

Trong đó :

* là các hằng số,
* được gọi là bias.
* Mối quan hệ  bên trên là một mối quan hệ tuyến tính (linear).

***Ưu điểm***:

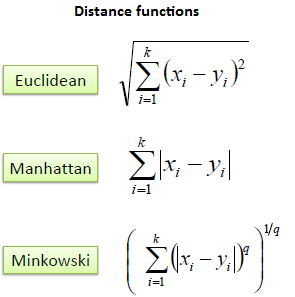
* Mô hình tương đối đơn giản có thể giải thích được mối quan hệ giữa các thuộc tính và giá trị đầu ra.

***Nhược điểm***:

* Hạn chế đầu tiên của Linear Regression là nó rất **nhạy cảm với nhiễu** (sensitive to noise). Trong ví dụ về mối quan hệ giữa diện tích và giá phòng bên trên, nếu có chỉ một cặp dữ liệu nhiễu (10 m2, 4.00.000 triệu) thì kết quả sẽ sai khác đi rất nhiều.
* Hạn chế thứ hai của Linear Regression là nó không biễu diễn được các **mô hình phức tạp.**

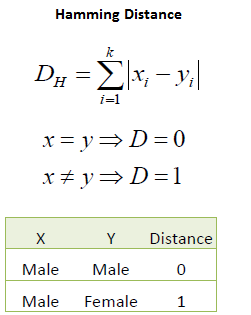
## Mô hình KNN

Một cách thực hiện đơn giản của hồi quy KNN là tính trung bình của mục tiêu số của K lân cận gần nhất. Một cách tiếp cận khác sử dụng trung bình khoảng cách nghịch đảo của K lân cận gần nhất. Hồi quy KNN sử dụng các hàm khoảng cách giống như phân loại KNN.



Hình 2.1. Công thức tính khoảng cách trong KNN

Ba biện pháp khoảng cách trên chỉ có giá trị cho các biến liên tục. Trong trường hợp các biến phân loại, bạn phải sử dụng khoảng cách Hamming, đây là thước đo số lượng phiên bản trong đó các ký hiệu tương ứng khác nhau ở hai chuỗi có độ dài bằng nhau.



Hình 2.2. Công thức Hamming

Chọn giá trị tối ưu cho K được thực hiện tốt nhất bằng cách kiểm tra dữ liệu trước tiên. Nói chung, giá trị K lớn chính xác hơn vì nó làm giảm nhiễu tổng thể; tuy nhiên, sự thỏa hiệp là các ranh giới riêng biệt trong không gian tính năng bị mờ. Xác thực chéo là một cách khác để xác định lại giá trị K tốt bằng cách sử dụng bộ dữ liệu độc lập để xác thực giá trị K của bạn. K tối ưu cho hầu hết các bộ dữ liệu là 10 trở lên. Điều đó tạo ra kết quả tốt hơn nhiều so với 1-NN.

***Ưu điểm***:

* Độ phức tạp tính toán của quá trình training là bằng 0.
* Việc dự đoán kết quả của dữ liệu mới rất đơn giản.
* Không cần giả sử gì về phân phối của các class.

***Nhược điểm***:

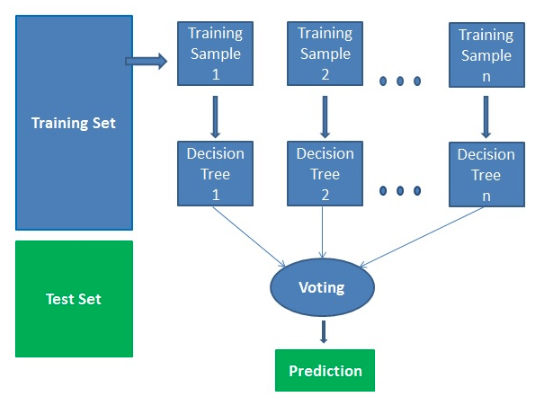
* KNN rất nhạy cảm với nhiễu khi K nhỏ.
* KNN là một thuật toán mà mọi tính toán đều nằm ở khâu test. Trong đó việc tính khoảng cách tới *từng* điểm dữ liệu trong training set sẽ tốn rất nhiều thời gian, đặc biệt là với các cơ sở dữ liệu có số chiều lớn và có nhiều điểm dữ liệu. Với K càng lớn thì độ phức tạp cũng sẽ tăng lên. Ngoài ra, việc lưu toàn bộ dữ liệu trong bộ nhớ cũng ảnh hưởng tới hiệu năng của KNN.

## Mô hình Random Forest

Random Forest — rừng ngẫu nhiên: đây là phương pháp xây dựng một tập hợp rất nhiều cây quyết định và sử dụng phương pháp voting để đưa ra quyết định về biến target cần được dự báo. Một ví dụ về Random Forest như sau: giả sử bạn muốn đi tham quan du lịch Anh và có sự cân nhắc cho việc tham quan thành phố nào như: Manchester, Liverpool hay Birmingham. Để trả lời câu hỏi này chúng ta sẽ cần tham khảo rất nhiều ý kiến từ bạn bè, blog du lịch, tour lữ hành … Mỗi một ý kiến tương ứng với một Decision Tree trả lời các câu hỏi như: thành phố này đẹp không, có được tham quan các sân vận động khi đến thăm không, số tiền bỏ ra là bao nhiêu, thời gian để tham quan thành phố là bao lâu… Sau đó bạn sẽ có một rừng các câu trả lời để quyết định xem mình sẽ đi tham quan thành phố nào. Random Forest hoạt động bằng cách đánh giá các Decision Tree sử dụng cách thức voting để đưa ra kết quả cuối cùng.

Nó hoạt động theo bốn bước:

* Chọn các mẫu ngẫu nhiên từ tập dữ liệu đã cho.
* Thiết lập cây quyết định cho từng mẫu và nhận kết quả dự đoán từ mỗi quyết định cây.
* Hãy bỏ phiếu cho mỗi kết quả dự đoán.
* Chọn kết quả được dự đoán nhiều nhất là dự đoán cuối cùng.



Hình 2.3. Cơ chế họa động của Random forest

Về mặt toán học thuật toán có thể được giải thích như sau: Random Forest là một tập hợp của hàng trăm Decision Tree, trong đó mỗi Decision Tree được tạo nên ngẫu nhiên từ việc tái chọn mẫu (chọn random 1 phần của data để xây dựng) và random các biến từ toàn bộ các biến trong trong data. Với một cơ chế như vậy, Random Forest cho ta một kết quả chính xác rất cao nhưng đánh đổi bằng việc ta không thể hiểu cơ chế hoạt động của thuật toán này do cấu trúc quá phức tạp của mô hình này, do vậy thuật toán này là một trong những phương thức Black Box, tức ta sẽ bỏ tay vào bên trong và rút ra được kết quả chứ không thể giải thích được cơ chế hoạt động của mô hình. Đó là sự đánh đổi giữa khả năng giải thích và khả năng dự báo như mình đã nêu ở bài đầu tiên.

Random Forest là một phương pháp Supervised Learning do vậy có thể xử lý được các bài toán về Classification (phân loại) và Regression (dự báo về các giá trị)

***Ưu điểm***:

* Random forests được coi là một phương pháp chính xác và mạnh mẽ vì số cây quyết định tham gia vào quá trình này. Nó không bị vấn đề overfitting. Lý do chính là nó mất trung bình của tất cả các dự đoán, trong đó hủy bỏ những thành kiến. Thuật toán có thể được sử dụng trong cả hai vấn đề phân loại và hồi quy. Random forests cũng có thể xử lý các giá trị còn thiếu. Có hai cách để xử lý các giá trị này: sử dụng các giá trị trung bình để thay thế các biến liên tục và tính toán mức trung bình gần kề của các giá trị bị thiếu.

***Nhược điểm***:

* Random forests chậm tạo dự đoán bởi vì nó có nhiều cây quyết định. Bất cứ khi nào nó đưa ra dự đoán, tất cả các cây trong rừng phải đưa ra dự đoán cho cùng một đầu vào cho trước và sau đó thực hiện bỏ phiếu trên đó. Toàn bộ quá trình này tốn thời gian.
* Mô hình khó hiểu hơn so với cây quyết định, nơi bạn có thể dễ dàng đưa ra quyết định bằng cách đi theo đường dẫn trong cây.

# MÔ HÌNH HỌC SÂU CHO ĐỊNH GIÁ PHÒNG



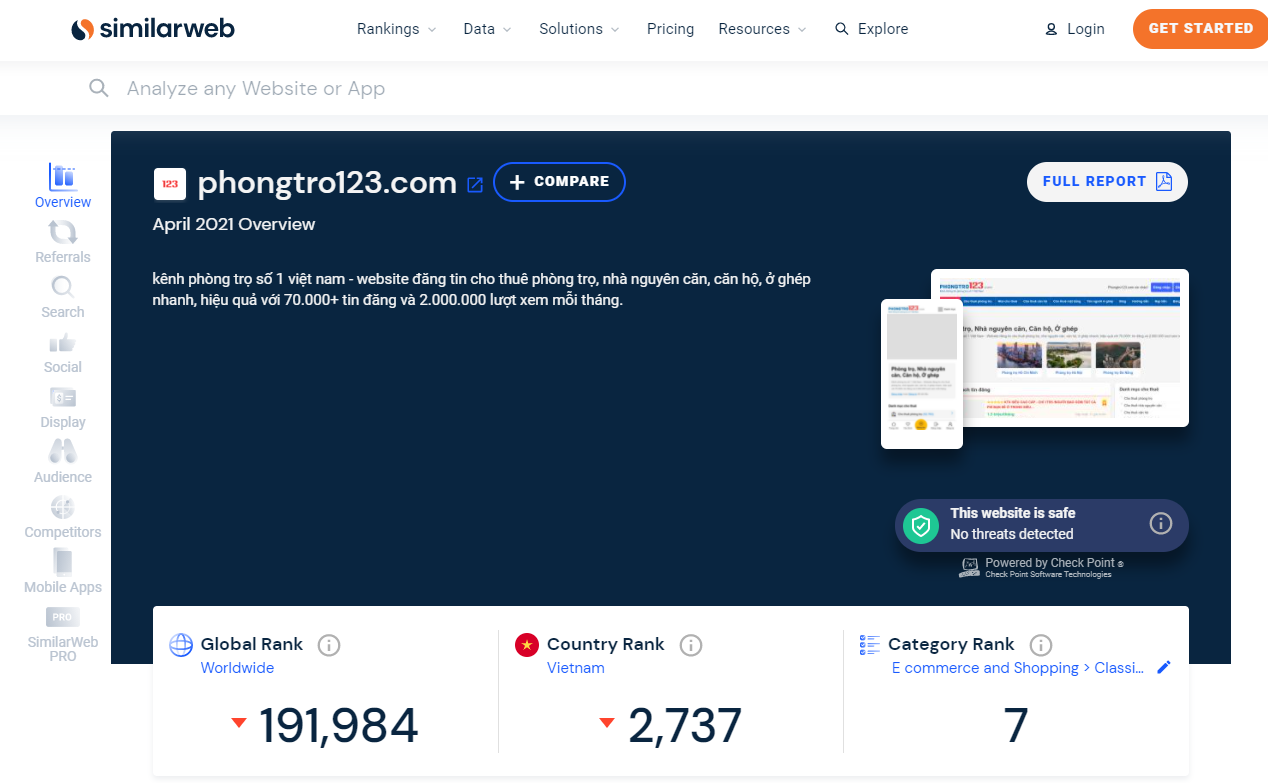
## Mô tả dữ liệu

Trong phần này sẽ trình bày quy trình xử lý từ việc chọn dữ liệu đến phân tích lựa chọn thuộc tính cho bài toán.

### Nguồn gốc dữ liệu

Nhằm mục đích phục vụ cho bài toán định giá phòng cho thuê, chúng ta cần một bộ dữ liệu hoàn chỉnh gồm giá và các thông tin đặc trưng của phòng kèm theo. Tuy nhiên hiện nay, không có một cơ quan, tổ chức nào thu thập và xây dựng dữ liệu này. Nên việc chúng ta có thể sử dụng dữ liệu có sẵn là bất khả thi.

Sau thời gian tìm kiếm, thật may mắn chúng ta có một nguồn dữ liệu thay thế có sẵn trên internet. Đó là các tin cho thuê phòng dày đặc trên các trang web thuê phòng, các trang mạng xã hội… Đây là các thông tin của các chủ phòng trọ đăng cho thuê phòng, thông tin này có độ chính xác chưa cao hoặc chưa đầy đủ, nhưng nó hoàn toàn có thể sử dụng cho việc định giá phòng. Dữ liệu sử dụng trong đề tài đến từ một trong các trang web cho thuê phòng phổ biến <https://phongtro123.com/>:



Hình 3.1. Thông tin khảo sát trang website

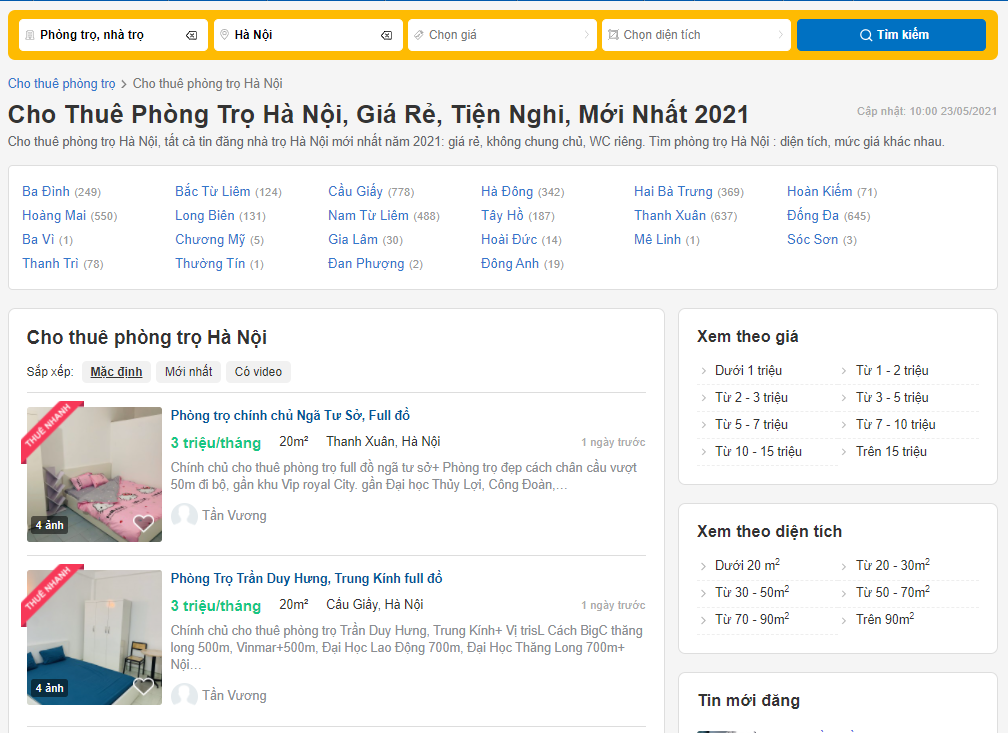
Trên đây là một vài thông kê về trang web theo khảo sát truy cập : dựa theo trang thống kê truy cập : [*https://www.similarweb.com/*](https://www.similarweb.com/)

### Cách thức thu thập dữ liệu

Trang web được lựa chọn để thu thập dữ liệu là [*https://phongtro123.com/*](https://phongtro123.com/%20%20) *,*một trang chuyên đăng tin cho thuê phòng trọ đông đảo người dùng ở Việt Nam. Những tin được chọn lọc là những tin cho thuê phòng tại những địa chỉ nằm trên địa bàn thành phố Hà Nội trong vòng 5 năm từ năm 2016 đến 5/2021. Các thông tin đặc trưng của phòng được cung cấp tối đa từ các thông tin cung cấp trên trang web. Sau đây là chi tiết cách thức thực hiện thu thập dữ liệu.

#### Khảo sát trang web

Trang web [*https://phongtro123.com*](https://phongtro123.com)là trang web chuyên giao thông tin cho thuê phòng trên phạm vị toàn quốc. Trang web có các tính năng lọc thông tin tìm kiếm như vị trí, giá phòng, diện tích, các khu vực được chia theo các quận huyện. Các kết quả trả về là một danh sách theo thứ tự được sắp xếp theo trình tự lọc tìm kiếm, chúng được chia theo từng trang với mỗi trang gồm các mục chứa link dẫn đến bài viết chi tiết. Mỗi trang có thể được truy cập bằng URL: [*https://phongtro123.com/tinh-thanh/ha-noi?page=2*](https://phongtro123.com/tinh-thanh/ha-noi?page=2%20). Mỗi mục sẽ có thông tin khái quát về phòng chúng ta có thể xem các thông tin cơ bản.



Hình 3.2. Trang web phongtro123.com

#### Quy trình các bước thu thập dữ liệu

Sau khi tiến hành khảo sát trang web ta sẽ tiến hành thu thập dữ liệu

**Bước 1**: ***Thu thập thông tin về URL***

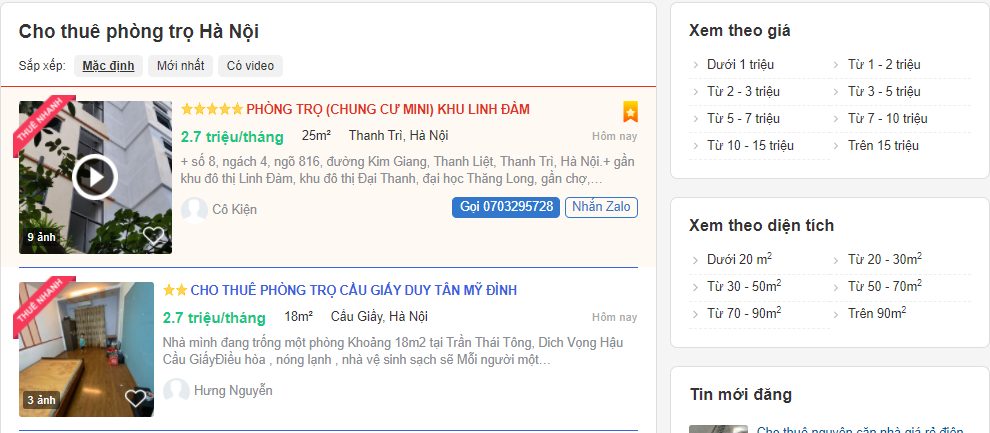
Sau khi thực hiện lọc theo các thông tin ta cần tìm kiếm :

* Khu vực : Hà Nội
* Loại bất động sản : Nhà trọ, phòng trọ



Hình 3.3. Giao diện tìm lọc thông tin website

Ta có URL tìm kếm trả về của trang web : [*https://phongtro123.com/tinh-thanh/ha-noi*](https://phongtro123.com/tinh-thanh/ha-noi)



Hình 3.4. Giao diện kết quả trả về

Click trang 2 ta có url: [*https://phongtro123.com/tinh-thanh/ha-noi?page=2*](https://phongtro123.com/tinh-thanh/ha-noi?page=2)



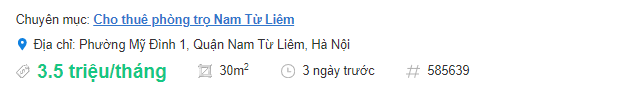
Hình 3.5. Giao diện trả về click các trang

Từ đó ta có thể kết luận: Ta có thể lấy hết thông tin về url bài đăng thông qua url tìm kiếm bằng các thay chỉ số page trên url tìm kiếm.

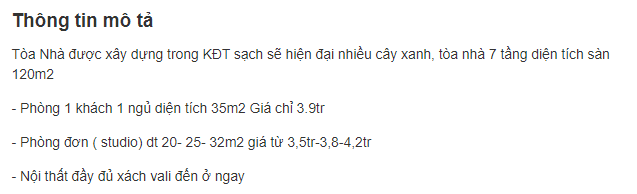
Tại mỗi trang ta thu thập các đường dẫn URL được đính kèm trong title tin đăng.

**Bước 2**: ***Thu thập thông tin cơ bản***

Sau khi thu thập được các đường dẫn đến chi tiết tin đăng ta tiến hành thu thập chi tiết các tin đăng. Nó bao gồm: Thông tin cơ bản , Thông tin mô tả , Hình ảnh, Đặc điểm đăng tin



Hình 3.6. Thông tin cơ bản của phòng



Hình 3.7. Thông tin mô tả chi tiết phòng



Hình 3.8. Thông tin về đặc điểm bài đăng

Với thông tin và hình ảnh đính kèm ta có thể có được thông tin về nội thất của căn phòng

#### Xây dựng cơ sở dữ liệu

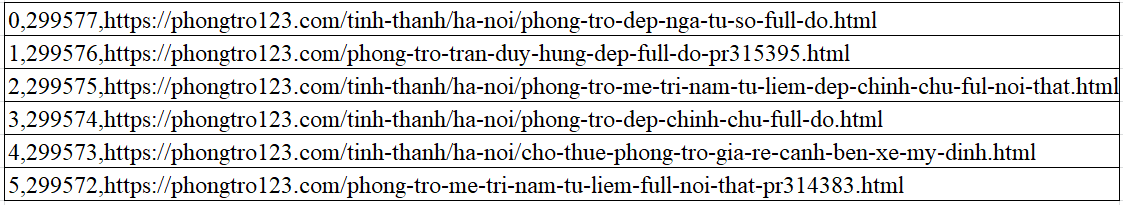
Sau quá trình khảo sát trang web, xây dựng các bước thu thập dữ liệu ta có thể năm được những thông tin nào cần lưu trữ trong cơ sở dữ liệu. Chính vì thế ta chuyển sang bước tiếp theo là bước xây dựng cơ sở dữ liệu.

Dữ liệu thu thập được không quá lớn nên được lưu trữ trong file dữ liệu dạng csv. Các cột tương ứng với các trường dữ liệu. Ta có 2 cơ sở dữ liệu trong đó có file dữ liệu url, file dữ liệu rawData .

Bảng 3.1. Các thông tin lưu trữ trong file url

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên thuộc tính | Ý nghĩa | Ví dụ |
| 1 | Id | Id của document | 299577 |
| 2 | url | Đường dẫn tin đăng | https://phongtro123.com/tinh-thanh/ha-noi/phong-tro-dep-nga-tu-so-full-do.html |

Ví dụ một vài row trong file :



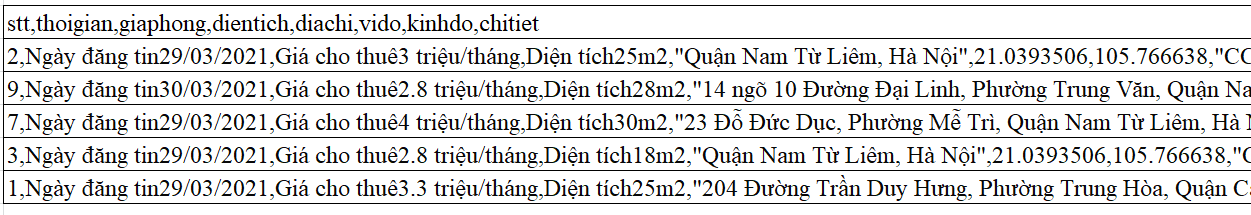
Hình 3.9.Ví dụ một document được lưu trong file url

Các thông tin được lưu trữ trong file dataraw – dữ liệu thô chưa qua xử lý là:

Bảng 3.2. Các thông tin lưu trữ trong file dataraw

|  |  |  |
| --- | --- | --- |
| STT | Tên thuộc tính | Ý nghĩa |
| 1 | stt | Stt tương ứng trong url |
| 2 | thoigian | Thời gian đăng tin |
| 3 | giaphong | Giá thuê của phòng theo tháng |
| 4 | dientich | Diện tích căn phòng |
| 5 | diachi | Tên quận/ huyện và tỉnh/ thành phố |
| 6 | vido | Vi độ của căn phòng trên map |
| 7 | kinhdo | Kinh độ của căn phòng trên map |
| 8 | chitiet | Mô tả đặc tính phòng |

Ví dụ một vài row trong file :



Hình 3.10.Ví dụ một document được lưu trong file data

### Mô tả dữ liệu thô

Dữ liệu thô là dữ liệu nguyên bản được thu thập về từ website chưa thông qua xử lý, trích rút bổ sung thông tin.

Tổng số bản ghi thu thập được là 3115 bản ghi.

Bảng 3.3. Các thông tin lưu trữ trong file url

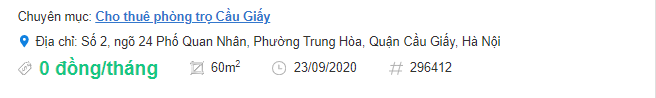
|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên thuộc tính | Ý nghĩa | Ví dụ |
|  | stt | Stt tương ứng trong url | 22 |
|  | thoigian | Thời gian đăng tin | Ngày đăng tin29/03/2021 |
|  | giaphong | Giá thuê căn phòng theo tháng | Giá cho thuê 4.5triệu/tháng |
|  | dientich | Diện tích căn phòng | Diện tích75m2 |
|  | diachi | Tên quận/ huyện và tỉnh/ thành phố | 609 Trương Định, Quận Hai Bà Trưng |
|  | vido | Vi độ của căn phòng trên map | 20.9801028 |
|  | kinhdo | Kinh độ của căn phòng trên map | 105.8447406 |
|  | chitiet | Độ rộng mặt tiền | CCMN số 2 ngõ 72 Miếu Đầm, Nam Từ Liêm, Hà Nội.- Khép kín full nội thất, có gác xép bao gồm:điều hòa, nóng lạnh, giường, tủ,.. |

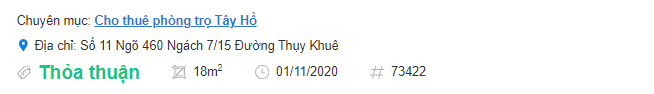
Dữ liệu trên chỉ là dữ liệu người thuê tự đăng không được kiểm chứng. Độ chính xác phụ thuộc rất nhiều vào người đăng tin. Có rất nhiều trường vị lỗi sai vô lý và thiết. Chủ yếu là địa chỉ không rõ ràng, giá phòng không đúng…

Địa chỉ phòng trọ không rõ ràng



Giá phòng không hợp lệ hoặc sai giá trị





Hình 3.11. Một số lỗi sai của dữ liệu

## Tiền xử lý dữ liệu

Dữ liệu thu thập được là dữ liệu thô, chưa được thống nhất về loại dữ liệu, kiểu dữ liệu. Chính vì thế bước tiền xử lý dữ liệu cần được thực hiện để đưa dữ liệu được thống nhất theo các trường, chia, tách, biến đổi các thuộc tính thu thập được thành các thuộc tính thống nhất, có cấu trúc và ngoài ra tìm phương án bổ sung thêm thuộc tính cho dữ liệu. Các công việc tiền xử lý dữ được thực hiện qua các bước sau:

**Bước 1**: Chuẩn hóa các trường đơn vị tiêu chuẩn.

Thuộc tính thời gian, giá phòng, diện tích không theo quy chuẩn chứa cả text. Chúng ta sẽ tách lấy dữ liệu và bỏ text khỏi trường. Thống nhất đơn vị của các trường : giá phòng float (nghìn/tháng), diện tích (m²). Riêng thuộc tính thời gian ta chỉ lấy tháng và năm.

Kết quả :



Hình 3.12. Kết quả quá trình tách trường thời gian, giá, diện tích

**Bước 2:** Tách thông tin quận huyện từ trường địa chỉ.

Thông tin về quận/ huyện trong dữ liệu thu thập về được ghép chung vào một thuộc tính đia chỉ. Ví dụ “Ngõ 28, Đường Tăng Thiết Giáp, Phường Cổ Nhuế 2, Bắc Từ Liêm, Hà Nội”. Chính vì thế chúng ta cần phải tách hai trường thuộc tính này ra trường quận độc lập.

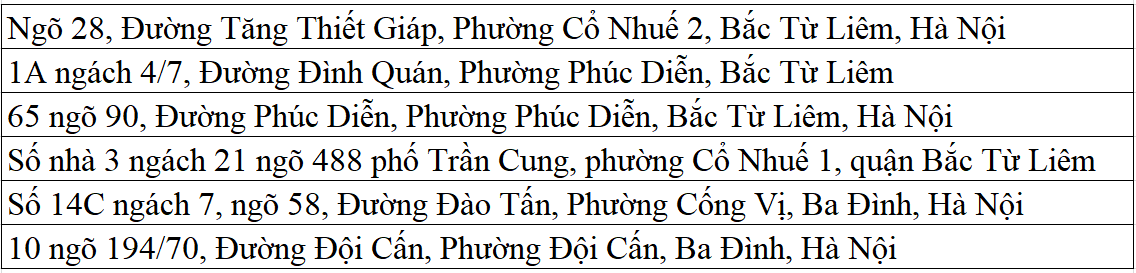
Kết quả :



Hình 3.13. Kết quả quá trình tách địa chỉ thành sang quận

**Bước 3:** Tách thông tin đường phường từ trường địa chỉ.

Quan sát thông tin trường địa chỉ ta nhận thấy rằng có rất nhiều thông tin địa chỉ của bài đăng được viết theo số nhà, tên đường, tên phường/xã, tên quận/ huyện, tên tỉnh/thành phố nhưng vẫn có nhiều các bản ghi thông tin địa chỉ không chuẩn theo thứ tự hoặc thiếu như hình dưới:



Hình 3.14. Quy luật trong cách viết địa chỉ

Phương pháp xây dưng tool tách dữ liệu kết hợp kiểm tra.

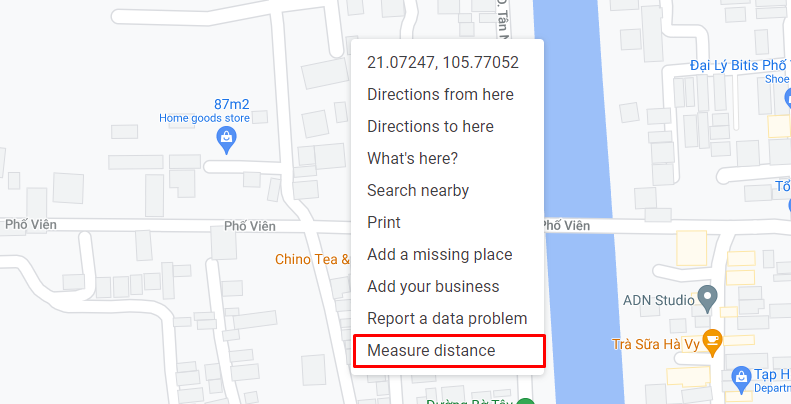
Kết quả:

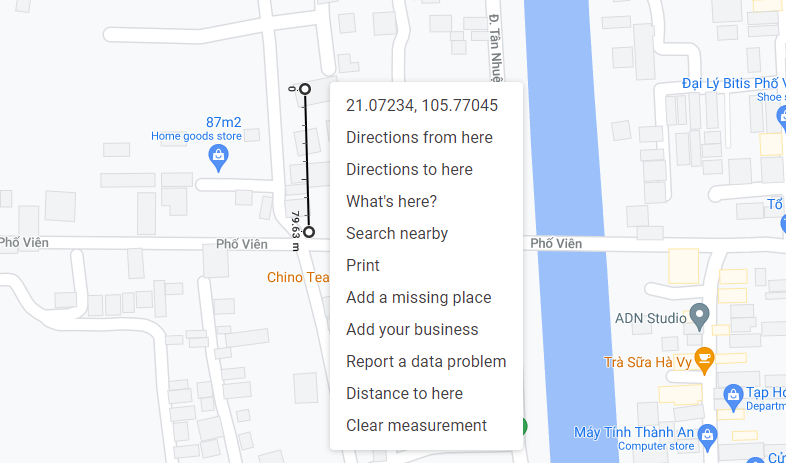


Hình 3.15. Kết quả quá trình tách địa chỉ thành đường, phường

**Bước 4**: Kiểm tra tọa độ lat, long, tiến hành lấy độ rộng mặt đường và khoảng cách đường chính.

Địa chỉ hiện tại ta không thể xác định được vị trí của tin đăng, nó cũng không phù hợp cho mô hình học máy nên ta sẽ chuyển chúng về dạng lat,long trong thực hiện ta sử dụng hai phương pháp: GoogleApi hoặc GoogleMap. Do trường độ rộng mặt đường và khoảng cách đường chính không thể sử dụng Api, nên ta sẽ sử dụng tính năng Measure distance dụng phương pháp GoogleMap :

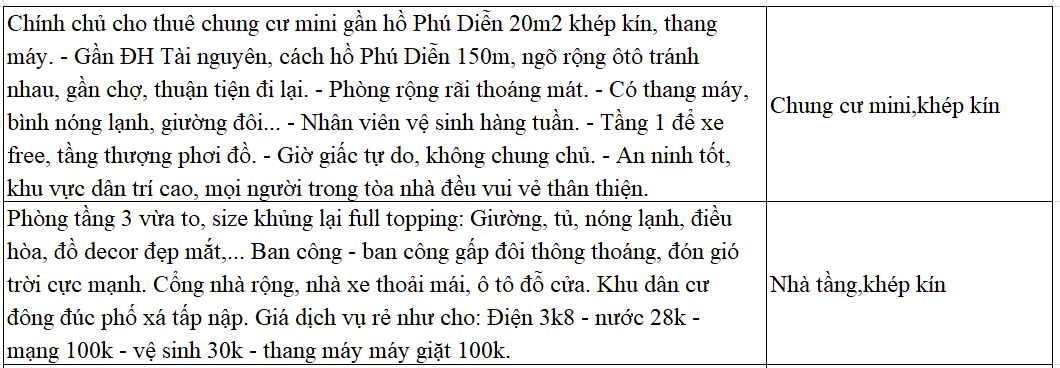




Hình 3.16. Sử dụng tính năng đo khoảng cách

**Bước 5:** Tách thêm thông tin từ thông tin chi tiết

Dựa vào thông tin phần mô tả chi tiết kết hợp với hình ảnh ta có thể tách thêm được một vài trường thông tin từ đây. Sau quá trình khảo sát, phân tích mô tả của căn phòng chúng tôi trích xuất thêm loại phòng,wc khép khí hay không, có các trang thiết bị hay không : Điều hòa , tủ lạnh , gác xếp, ban công, bếp,…

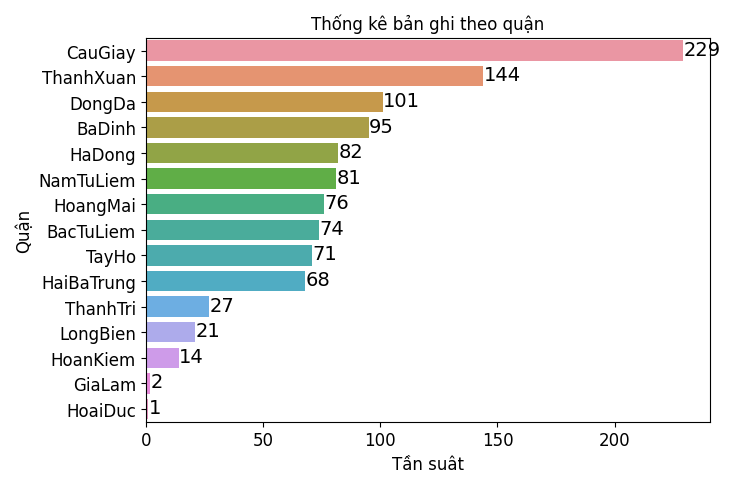


Hình 3.17. Tách thuộc tính loại phòng và loại wc

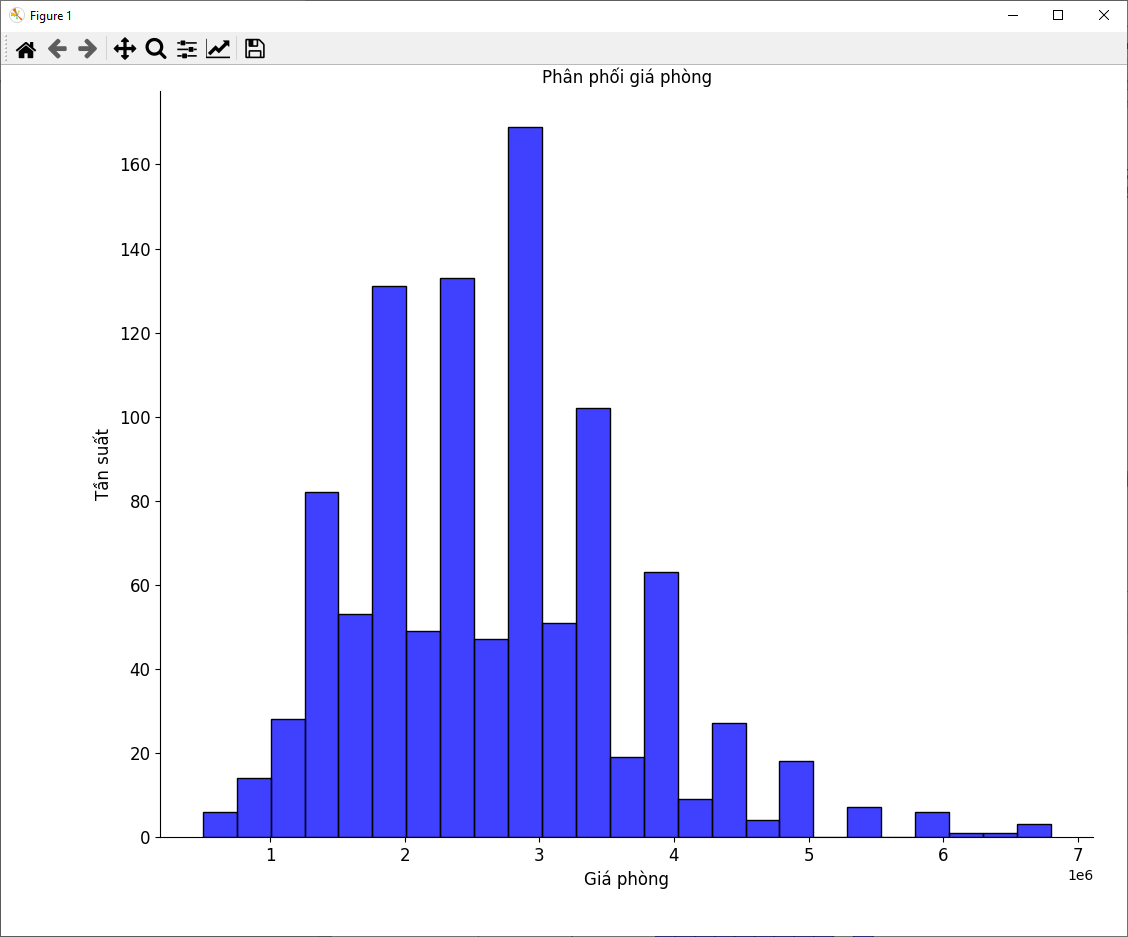
## Phân tích lựa chọn thuộc tính

Trước khi tiến hành phân tích lựa chọn thuộc tính ta cần khám phá về dữ liệu. Tổng bản ghi sau khi tiền xử lý : 1024 bản ghi . Trong đó số lượng bản ghi thu thập được cụ thể theo từng quận/huyện được thống kế trong hình:

Thống kê bản ghi theo từng huyện:

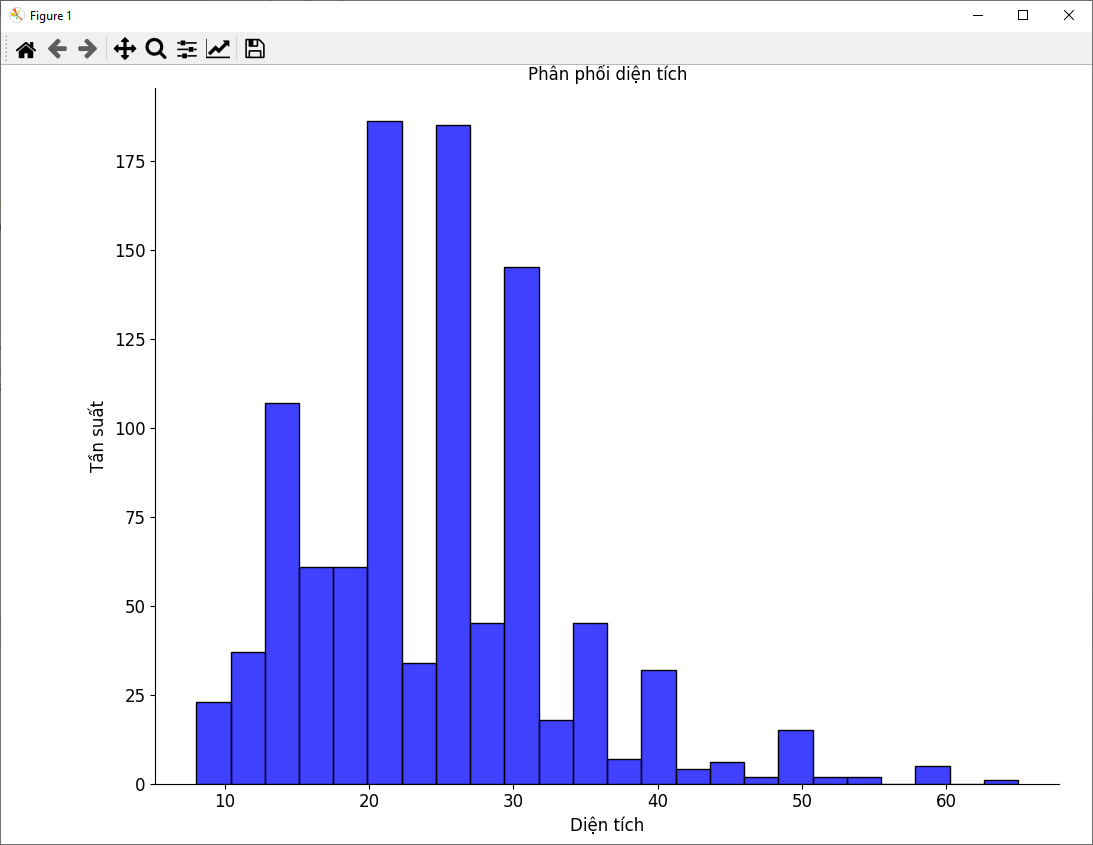


Hình 3.18. Đồ thị số lượng bản ghi trên từng khu vực



Hình 3.19. Đồ thị phân phối giá phòng và tần suất

Phân phối diện tích :



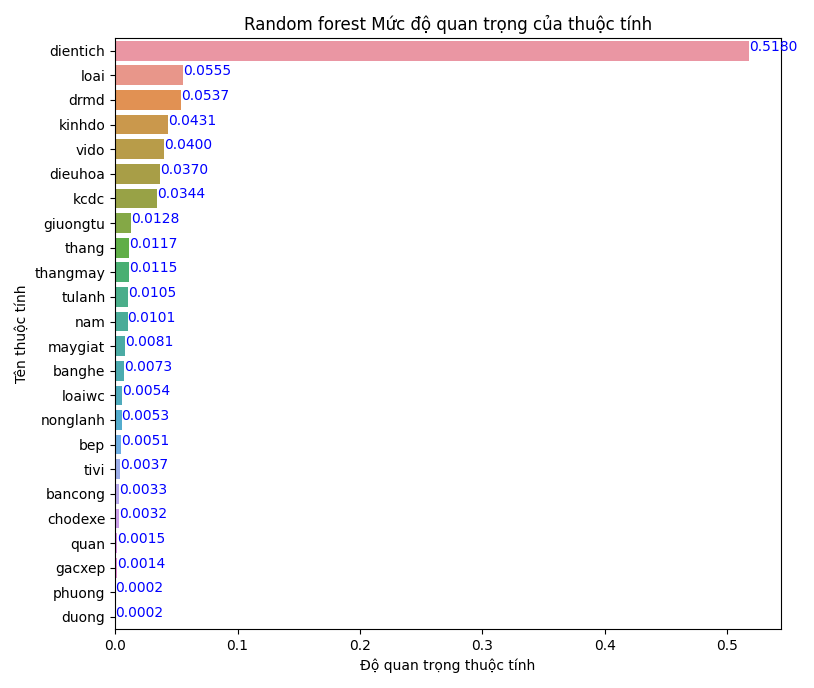
Hình 3.20. Đồ thị phân phối diện tích và tần suất

Sau bước tiền xử lý dữ liệu những thuộc tính có thể được đưa vào mô hình là:

Bảng 3.4. Danh sách các thuộc tính có thể đưa vào mô hình

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên thuộc tính | Kiểu dữ liệu | Ý nghĩa | Ví dụ |
|  | stt | Number | Stt tương ứng trong url | 1 |
|  | thoigian | Number | Thời gian đăng tin ngày tháng năm | 29/03/2021 |
|  | giaphong | Number | Giá thuê căn phòng theo tháng  Đơn vị : triệu/tháng | 2.500.000 |
|  | dientich | Number | Diện tích căn phòng Đơn vị triệu/ m². | 25 |
|  | diachi | Text | Số nhà,  Ngõ, Đường, Phường/Xã,  Quận/Huyện, Tỉnh/Thành phố | Số 12, đường Hoàng Quốc Việt, Phường Cổ Nhuế, Quận Cầu Giấy, Hà Nội |
|  | duong/pho | Category | Tên đường, phố | Hoàng Quốc Việt |
|  | phuong/xa | Category | Tên phường, xã | Cổ Nhuế 1 |
|  | quan/huyen | Category | Tên quận, huyện | Quận Cầu Giấy |
|  | vido | Number | Vi độ của căn phòng trên map | 20.9801028 |
|  | kinhdo | Number | Kinh độ của căn phòng trên map | 105.844740 |
|  | loai | Category | Loại căn phòng | Nhà tầng |
|  | drmd | Number | Độ rộng của mặt ngõ phòng  Đơn vị : m | 2.5 |
|  | kcdc | Number | Khoảng cách ngắn nhất ra đường chính  Đơn vị : m | 100 |
|  | loaiwc | Category | Loại hình nhà vệ sinh | Khép kín |
|  | giuongtu | BOOL | Có giường hoặc tủ | Có |
|  | banghe | BOOL | Có bàn hoặc ghế | Không |
|  | nonglanh | BOOL | Có nóng lạnh | Có |
|  | dieuhoa | BOOL | Có điều hòa | Có |
|  | tulanh | BOOL | Có tủ lạnh | Có |
|  | maygiat | BOOL | Có máy giặt | Có |
|  | tivi | BOOL | Có tivi | Không |
|  | bep | BOOL | Có bếp nấu | Có |
|  | gacxep | BOOL | Có gác xếp | Có |
|  | thangmay | BOOL | Có thang máy | Không |
|  | bancong | BOOL | Có ban công | Có |
|  | chodexe | BOOL | Có chỗ để xe | Có |

Dựa vào thông tin đặc điểm các thuộc tính thu được ta có thể thấy được những thuộc tính được lựa chọn để có thể đưa vào mô hình học máy phải là những thông tin rõ ràng, miêu tả chính xác đặc điểm căn nhà. Đó chính là các thuộc tính thuộc kiểu dữ liệu number hoặc category hoặc có thể biến đổi về 2 loại trên. Ta nhận thấy rằng trường phong/xa, duong/pho có thể đưa vào mô hình học máy, nhưng do lượng dữ liệu không đủ để thực hiện điều này nên nhằm tăng độ chính xác mô hình ta sẽ không đưa chúng vào thuộc tính lựa chọn. Ngoài ra ta có thể sử dụng phương pháp Random forest trong việc xác định mức độ quan trọng của thuộc tính giúp ta dễ dàng tạo ra mô hình tối ưu nhất:



Hình 3.21. Biều đồ mức độ quan trọng thuộc tính sử dụng RF

Dựa vào những thông tin trên, ta lựa chọn được thuộc tính có thể đưa vào mô hình học máy là : *thoigian,* *giaphong, dientich,quan, vido, kinhdo, drmd, kcdc, loai, loaiwc, giuongtu, banghe, nonglanh, dieuhoa, tulanh, maygiat, tivi, bep, gacxep, thangmay, bancong, chodexe.*

Đưới đây là bảng tóm tắt thông tin về dữ :

Bảng 3.3. Bảng tóm tắt thông tin một số trường dữ liệu

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| STT | Tên Thuộc tính | Giá trị  trung bình | Giá trị  nhỏ nhất | Giá trị  trung vị | Giá trị  trung vị | Giá trị  lớn nhất |
| 1 | giaphong  (triệu) | 2705778 | 500000 | 2600000 | 3300000 | 6950000 |
| 2 | dientich  ( | 24.2 | 8 | 24.5 | 30.0 | 65.0 |
| 3 | drmd  (m) | 3.2 | 1.6 | 3.2 | 4.0 | 14.5 |
| 4 | kcdc  (m) | 99.6 | 1.0 | 75.8 | 129.3 | 800.0 |

## Lựa chọn xây dựng mô hình học sâu

Định giá phòng là một bài toán hồi quy. Chính vì thế các thuật toán được lựa chọn là các thuật toán học máy phục vụ cho bài toán phổ biển và thông dụng nhất hiện nay. Nghĩ đến bài toán hồi quy thì thuật toán đầu tiên được nghĩ đến đó chính là Linear Regression.Nhưng trong đề tài chúng ta sẽ không đề cập đến nó, mà đề cập đến các mô hình Deeplearning.Về cơ bản với một bài toán định giá thì việc lựa chọn thuật toán này là không thực sự cần thiết, nhưng như đã nói ở trên, nhóm vẫn muốn tìm ra một hướng giải quyết từ đó có cái nhìn tổng quan và những đánh giáchính xác về các phương pháp tiếp cận bài toán.

### Lọc dữ liệu nhiễu

Với mỗi trường thuộc tính đã trích xuất ta vẫn có các trường giá trị tồn tại không hợp lý, điều này có thể gây ra do thông tin người dùng đăng tin không chính xác, hoặc sai trong quá trình input. Đặc điểm chung của những dữ liệu nhiễu này là quá lớn hoặc quá nhỏ so với thực tế. Giải pháp được thực hiện trong bài toán là việc loại bỏ bản ghi có trường giá trị không chính xác. Điều tương tự cũng có thể được thực hiện với những trường giá trị bị thiếu

Ví dụ trong trường hợp giá phòng nhỏ nhưng lại được diện tích quá lớn đây là sự bất hợp lý trong bản dữ liệu

### Thiếu dữ liệu

Mới một cơ sở dữ liệu tập chung vào phần mô tả thông tin thì việc thiếu dữ liệu trường là hoàn toàn xảy ra. Trong đó phổ biến là trường địa chỉ : *diachi,* các thông tin liên quan đến thiết bị phòng.

Với những trường được đánh giá có mức độ ảnh hưởng cao đến mô hình thì giải pháp tối ưu nhất là loại bỏ bản ghi đó.

Với trường dữ liệu liên quan đến trang thiết bị của phòng ta có thể sử dụng hình ảnh, kết hợp với phần mô tả để có được nhiều thông tin hơn.

### Chuẩn hóa dữ liệu

* ***Đối với trường dữ liệu Thời gian*** : Mô hình mạng không thể nhận đầu vào có kiểu dữ liệu là thời gian nên ta sẽ tách dữ liệu thời gian thành 3 trường: *ngay,thang, nam*. Cộng thêm giá phòng chỉ có chu kỳ theo tháng nên ta chỉ chọn hai trường là tháng, năm sau khi tách từ trường *thoigian.*

Do tính chất chu kỳ của tháng nên ta sẽ chuẩn hóa tháng theo sin, cos. Cụ thể với trường thang: ta sẽ tách trường này thành *thang\_sin, thang\_cos* giá trị của các trường này năm trong khoảng [0->1]. Khi đo ta có :

*thoigian = nam + thang\_sin + thang\_cos*

Chuẩn hóa dữ liệu ta thường đưa dữ liệu về dạng số nguyên :

Ta có 2 loại dữ liệu chủ yếu đối với bài toán :

* Number :
* Category:
* Bool :
* ***Đối với trường dữ liệu dạng Number*** : Ta thường sử dụng hai phương pháp chủ yếu để chuẩn hóa dữ liệu là: Normalize và Standardizes mỗi loại chuẩn hóa này sẽ tương ứng và được khuyên dùng cho một kiểu number cụ thể :
* Với những trường dữ liệu có tính chất phân phối chuẩn ta sử dụng Standardizes
* Với những trường dữ liệu có tính chất liên tục: ta sử dụng Normalize

Nó chỉ là theo kinh nghiệm thực tế, ngoài ra ta hoàn toàn có thể thử các phương sao sao cho mô hình có chất lượng cao nhất.

Dựa trên các thông tin trên ta tiến hành Standardizes với các trường dữ liệu : *dientich, vido, kinhdo, drmd, kcdc.* Normalize với trường: *nam.*

* ***Đối với trường dữ liệu dạng Category***: Ta thường có hai phương pháp chủ yếu để chuẩn hóa dữ liệu: onehot encoding và ordinal encoding.
* Onehot encoding nó rất phù hợp với các Category mà giá trị của chúng độc lập nhau không có tính cấp độ. Đơn giản là việc tác dữ liệu Category thành các cột tương ứng.
* Ordinal encoding nó phù hợp với các Category có thứ tự ví dụ như : cold, warm, hot

Dựa trên các thông tin trên ta tiến hành Ordinal encoding với các trường dữ liệu : *loai, loaiwc*. Sau đó ta tiếp tục sử dụng Normalize để chuẩn hóa dữ liệu về khoảng [0->1]

* ***Đối với trường dữ liệu dạng BOOL:*** Ta đơn gian chỉ là quy đổi nó về 0 hoặc 1: *giuongtu,banghe,nonglanh,dieuhoa,tulanh,maygiat,tivi,bep,gacxep,*

*thangmay, bancong,chodexe.*

### Xây dựng mô hình học sâu

Với kiến thức ở trên ta nhận thấy kiến trúc mạng MLP rất phù hợp cho bài toán dự báo giá báo nói riêng và bài toán hồi quy nói chung. Vì vậy, ta sẽ sử dụng kiến trúc mạng MLP cho đề tài này.

Muốn xây dựng mô hình mạng neural học sâu (cụ thể là mạng MLP) ta cần các tham số cho mạng bao gồm :

* Số neural tầng input và ouput
* Số lớp ẩn và số neural mỗi lớp
* Hàm kích hoạt và hàm output

Ta sẽ lần lượt tìm ra các thông số này cho mạng:

Sau quá trình chuẩn hóa dữ liệu ta sẽ có dữ liệu gồm: 22 thuộc tính. Riêng với trường tháng: ta sẽ chia làm 2 trường, quận sẽ chia làm 15 trường (sử dụng onehot). Từ đó bộ dữ liệu của ta bao gồm 37 thuộc tính tương đương với lớp input sẽ có ***37 Neural***.

Bài toán của chúng ta là một bài toán hồi quy với giá trị đầu ra là một số nguyên dương, nên đầu ra của mạng neural tương ứng là một. Từ đó ta có số neural output là ***1 Neural.*** Đồng nghĩa với việc hàm kích hoạt của ***output layer là tuyến tính***. Chỉ là tổng hợp đẩu ra của các lớp trước đó.

Đối với hàm kích hoạt (Activation Function) hiện nay được sử dụng phổ biến là hàm ***ReLU*** và các biến thể của nó.

Với một bài toán mới, không có một quy tắc nhất định nào cho phép ta quyết định được số tầng ẩn và lượng neural của mỗi tầng ẩn để cho ta một mô hình tốt nhất. Nhưng có một vài chú ý có thể giúp ta tìm được các thông số này bao gồm: Số lượng neural của mỗi tầng ẩn thường được chọn nằm trong kích thước input và output, chúng ta đang xây dựng một mạng neural học sâu nên số lớp ẩn thường lớn hơn hoặc bằng 2. Điều này sẽ có ích cho việc chọn tham số cho mạng. Như vậy ta sẽ khởi tạo mạng với ***2 tầng ẩn*** và số neural mỗi tầng ẩn là ***37 Neural.***

Để thẩm định độ chính xác mô hình ta có thể sử dụng validation nhưng do dữ liệu là tương đối ít. Nên để đánh giá mô hình ta sử dụng K Fold Cross Validation đánh giá dự trên tham số MAPE. Trong phần sau sẽ trình bầy về MAPE và K-Fold Cross Validaiton.

Model Optimizers: là kỹ thuật tìm kiếm được sử dụng để cập nhật trọng số trong mô hình. Chúng ta cũng có thể sử dụng các tham số mặc định của trình tối ưu hóa bằng cách chỉ định tên của trình tối ưu hóa cho đối số của trình tối ưu hóa. Một số công cụ tối ưu hóa gradient gốc phổ biến:

* **SGD**: stochastic gradient descent, with support for momentum.
* **RMSprop**: adaptive learning rate optimization method proposed by Geoff Hinton.
* **Adam**: Adaptive Moment Estimation (Adam) that also uses adaptive learning rates.

Để đào tạo mô hình ta cần 2 tham số cho mô hình bao gồm Epochs và Batch Size trong đó:

* Epochs (nb\_epoch) là số lần mà mô hình học hết từ tập train.
* Batch Size (batch\_size) là số lượng mẫu sử dụng cho mỗi lần cập nhật trong số.

Trong mô hình sẽ sử dụng Epochs: 600 và Batch Size : 32

Ngoài ra nhằm mục đích đánh giá giữa mô hình học sâu MLP với các mô hình truyền thống. Đề tài sẽ xây dựng thêm các mô hình truyền thống: Multiple Linear regression, Knn, Random Forest.

## Đánh giá chất lượng mô hình

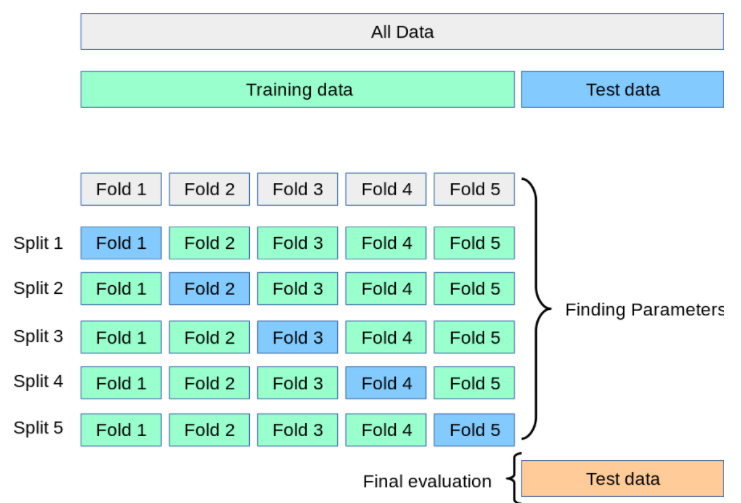
Ta đánh giá chất lượng mô hình dựa vào các độ đo là sai số tuyệt đối trung bình MAE (mean absolute error) và phần trăm sai số tuyệt đối trung bình MAPE (mean absolute percentage error). Đây là hai độ đo được tính toán trên kết quả ở cả tập training và tập test ở tất cả các mô hình được xây dựng để lựa chọn đánh giá mô hình.

Một mô hình tốt là một mô hình có kết quả các độ đo MAE và MAPE càng nhỏ càng tốt

Ngoài ra có một khái niện nữa để đánh giá chất lượng mô hình là overfitting. Overfitting là hiện tượng mô hình tìm được quá khớp với dữ liệu training. Việc quá khớp này có thể dẫn đến việc dự đoán nhầm nhiễu, và chất lượng mô hình không còn tốt trên dữ liệu test nữa. Dữ liệu test được giả sử là không được biết trước, và không được sử dụng để xây dựng các mô hình Machine Learning. Về cơ bản, overfitting xảy ra khi mô hình quá phức tạp để mô phỏng training data. Điều này đặc biệt xảy ra khi lượng dữ liệu training quá nhỏ trong khi độ phức tạp của mô hình quá cao và không bị hiện tượng overfitting tức sự chênh lệch giữa các độ đô trên tập training và tập testing càng nhỏ càng tốt. Một mô hình tốt sẽ không xảy ra hiện tượng overfitting hoặc sự chênh lệch này được giảm xuống thấp nhất có thể.

Từ viện đánh giá chất lượng mô hình chúng ta sẽ quyết định cải tiến mô hình bằng cách thay đổi, bổ sung phương pháp xử lý dữ liệu hay chuẩn hóa dữ liệu hay tiến hành tunning tham số của thuật toán để tăng độ chính xác và giảm hiện tượng overfitting của mô hình.

Do dữ liệu thu thập được là tương đối ít, điều này dẫn đến việc có thể một số điểm dữ liệu có ích cho quá trình train đã bị ném vào để làm validation, test và model không có cơ hội học điểm dữ liệu đó. Thậm chí, đôi khi do ít dữ liệu nên có một vài class chỉ có trong validation, test mà không có trong train. Vì thế ta không thể đánh giá mô hình một cách chính xác nhất. Nên ta sẽ không chia dữ liệu làm thành 3 tập train, validation, và test. Ta sẽ sử dụng k-fold cross-validation thay vì một tập validation riêng.



Hình 3.22. Cơ chế hoạt động k fold cross validation

Phần dữ liệu Training sẽ được chia ngẫu nhiên thành K phần (K là một số nguyên, hay chọn là 5 hoặc 10). Sau đó train model K lần, mỗi lần train sẽ chọn 1 phần làm dữ liệu validation và K-1 phần còn lại làm dữ liệu training. Kết quả đánh giá model cuối cùng sẽ là trung bình cộng kết quả đánh giá của K lần train. Đó chính là lý do vì sao ta đánh giá khách quan và chính xác hơn.

Phần tiếp theo ta sẽ tối ưu tham số cho mô hình mạng. Mục đích chính của phần này là làm tăng khả năng học của mạng và giảm thời gian train mô hình, tăng hiệu quả mô hình. Trong phần này ta sẽ quan tâm đến các tham số của mạng như sau:

* Số layer tầng ẩn
* Số Neural ở mỗi tầng ẩn
* Khởi tạo trọng số của mạng

Bắt đầu với số lượng tầng ẩn. Dựa vào mô hình ban đầu ta tiến hành thử nghiệm với các số lượng tầng ẩn khác nhau để đánh giá và so sánh. Phương pháp là sử dụng tham số MAPE và K fold.

Dưới đây là bảng thông kê khi thay đổi tham số lớp ẩn :

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Số lớp ẩn | Số neural/ Lớp ẩn | MAPE (K\_ Fold) |
| 1 | 2 | 37 | 0.15 +/- 0.02 |
| 2 | 3 | 37 | 0.14 +/- 0.01 |
| 3 | 4 | 37 | 0.14 +/- 0.01 |
| 4 | 5 | 37 | 0.13 +/- 0.02 |

Dựa vào bảng thống kê trên ta có thể thấy: Hệ số mape trong K\_ Fold giảm dần nếu ta tăng số tầng ẩn từ 2-> 5. Tuy nhiên, độ lệnh chuẩn của chúng có xu hướng giảm từ 2->3 và tăng từ 4->5. Từ đó ta có thể kết luận với dữ liệu trên mô hình MLP với 3 lớp ẩn sẽ cho kết quả tốt.

Tiếp theo ta sẽ thí nghiệm dữ liệu trên với số lượng nearal ở mỗi lớp ẩn khác nhau. Ta sẽ dựa vào một vài quy tắc để thực hiện việc chọn ra số neural test :

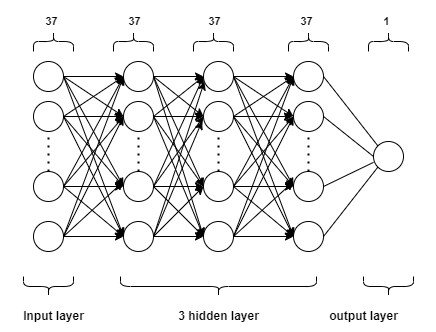
* + Số lượng tế bào thần kinh ẩn nên nằm giữa kích thước của lớp đầu vào và kích thước của lớp đầu ra.
  + Số lượng tế bào thần kinh ẩn nên bằng 2/3 kích thước của lớp đầu vào, cộng với kích thước của lớp đầu ra.

Dưới đầy là kết quả của các lần test số lượng neural mỗi lớp.

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Số lớp ẩn | Số neural/ Lớp ẩn | MAPE (K\_ Fold) |
| 1 | 3 | 37 | 0.14 +/- 0.01 |
| 2 | 3 | 30 | 0.14 +/- 0.01 |
| 3 | 3 | 25 | 0.15 +/- 0.02 |
| 4 | 3 | 20 | 0.15 +/- 0.02 |
| 5 | 3 | 5 | 0.16 +/- 0.03 |

Theo thống kê trên ta nhận thấy, số neural lớp ẩn để mô hình đạt được độ chính xác tốt nhất là 37 hoặc 30. Nhưng do số 37 là trùng với input đầu vào nên ta sẽ chọn 37 là số reural của lớp ẩn.

Khi đó ta sẽ có mô hình MLP như sau:



Cuối cùng cũng vô cùng quan trọng đó là việc khởi tạo trọng số cho mạng. Việc khởi tạo trong số cho mạng giúp cho mạng tăng hiệu quả. Tránh được các hiện tượng *biến mất* hoặc *bùng nổ* gradients.

* Khi trọng số của chúng ta nhỏ dẫn đến các gradient của chúng ta gần bằng 0, các gradient trong các lớp ngược dòng của ta sẽ biến mất vì ta đang nhân các giá trị nhỏ và ví dụ: 0,1 x 0,1 x 0,1 x 0,1 = 0,0001. Do đó, sẽ rất khó để tìm ra điểm tối ưu, vì các lớp thượng nguồn của ta học chậm.
* Điều ngược lại cũng có thể xảy ra. Khi trọng số của ta và do đó gradient> 1, các phép nhân trở nên thực sự mạnh mẽ. 10 x 10 x 10 x 10 = 1000. Gradient cũng có thể bùng nổ, gây ra tràn số lượng trong các lớp thượng nguồn của ta, khiến chúng không thể kiểm soát được (thậm chí làm chết các tế bào thần kinh trong các lớp đó).

Một vài phương pháp khởi tạo trọng số được sử dụng phổ biến:

* He\_normal là phương pháp khởi tạo khởi tạo dựa chiến thuật khởi tạo ngẫu nhiên với phân phối chuẩn mean là 0 có phương sai là 2/N. Trong đố N là số input đầu vào của mạng neural.
* Glorot\_normal là phương pháp khởi tạo khởi tạo dựa chiến thuật khởi tạo ngẫu nhiên với phân phối chuẩn mean là 0 có phương sai là 2/(N +O). Trong đó O số lượng output đầu ra.

Do mạng chúng ta sử dụng ReLU nên phương pháp khởi tạo He\_normal được sử dụng.

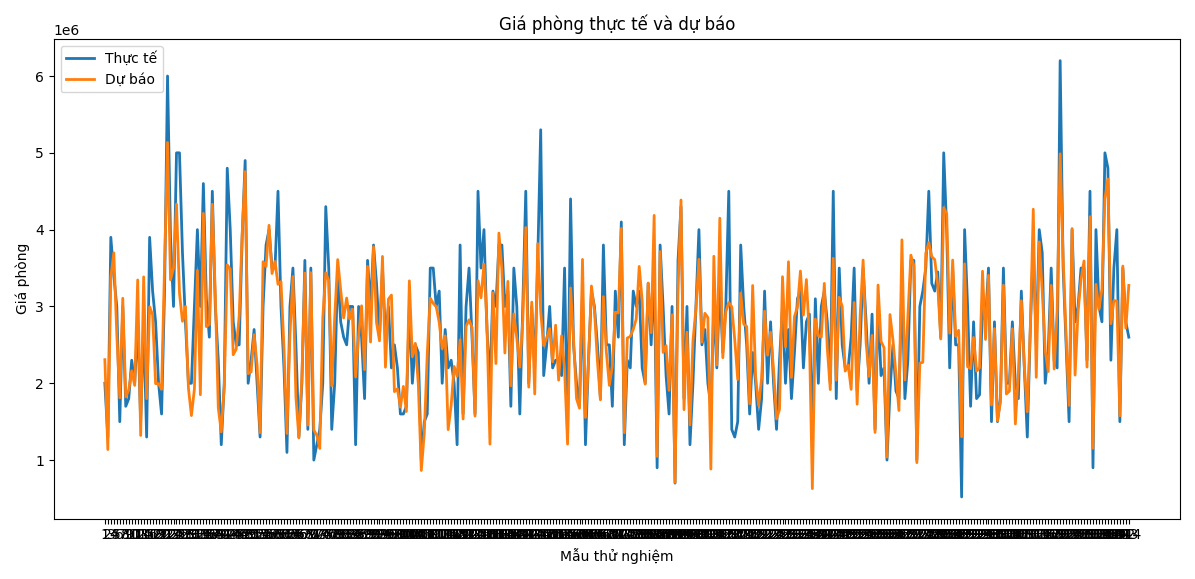
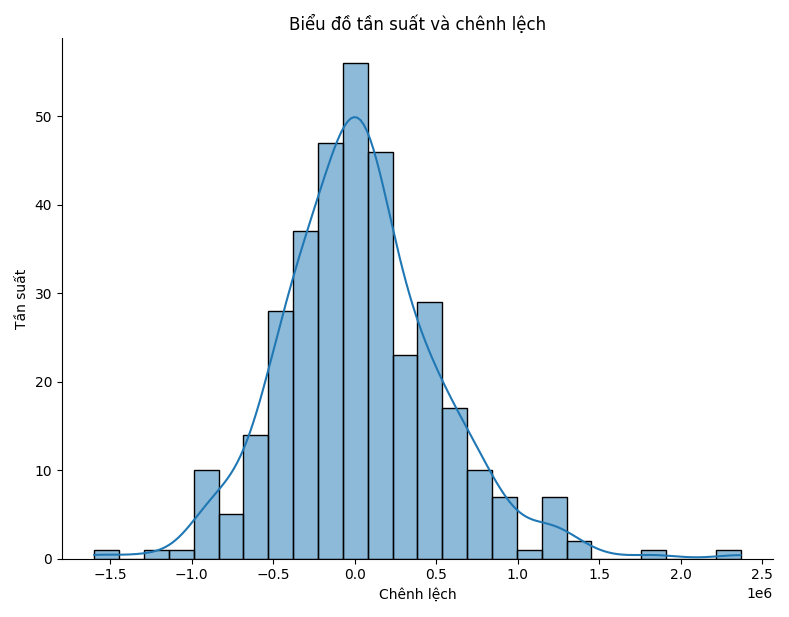
Dưới đây là bảng kết quả test khi sử dụng các phương pháp khởi tạo khác nhau sử dụng K\_Ford để đánh giá:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Số lớp ẩn | Phương pháp | MAPE (K\_ Fold) |
| 1 | 3 | glorot\_uniform | 0.15 +/- 0.01 |
| 2 | 3 | he\_normal | 0.14 +/- 0.01 |

Ta tiếp tục xây dựng và huấn luyện các mô hình: multiple linear regression, knn, random forest. Việc huấn luyện trên nhiều mô hình giúp ta đánh giá được dữ liệu, cũng như so sánh độ chính xác của mô hình học sâu với các mô hình truyền thống.

Phần train trên một bộ dữ liệu và nó sẽ là model dự báo cho Api.

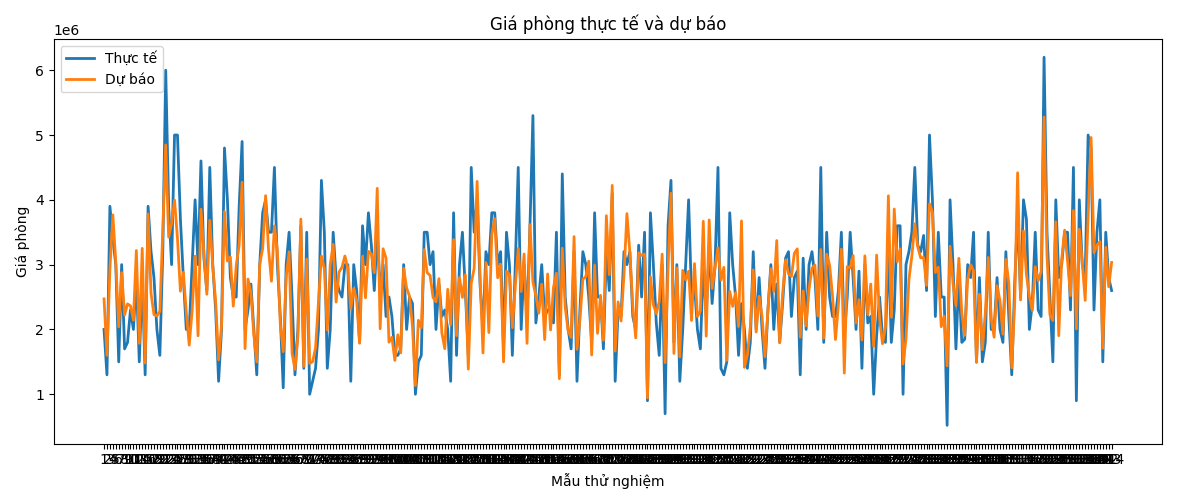
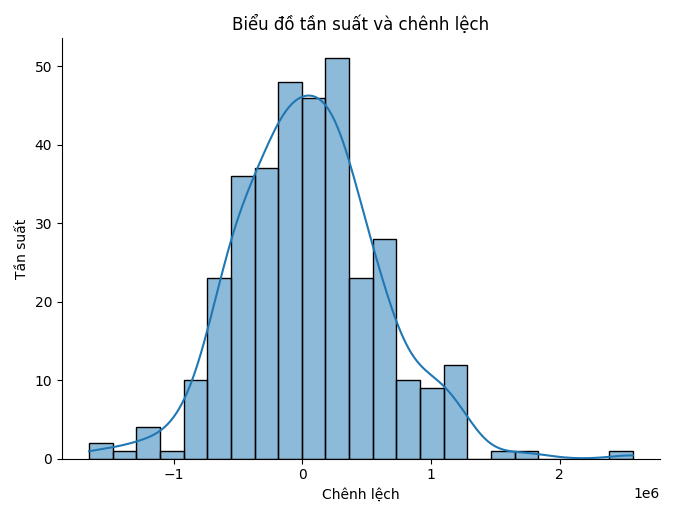
* Multiple linear regression



Hình 3.23. Biểu đồ giao động giữa giá thực tế và dự báo MLR

Ta chú ý đến 2 giá trị là MAPE : 14,6 % và có độ VarSore: 0.74

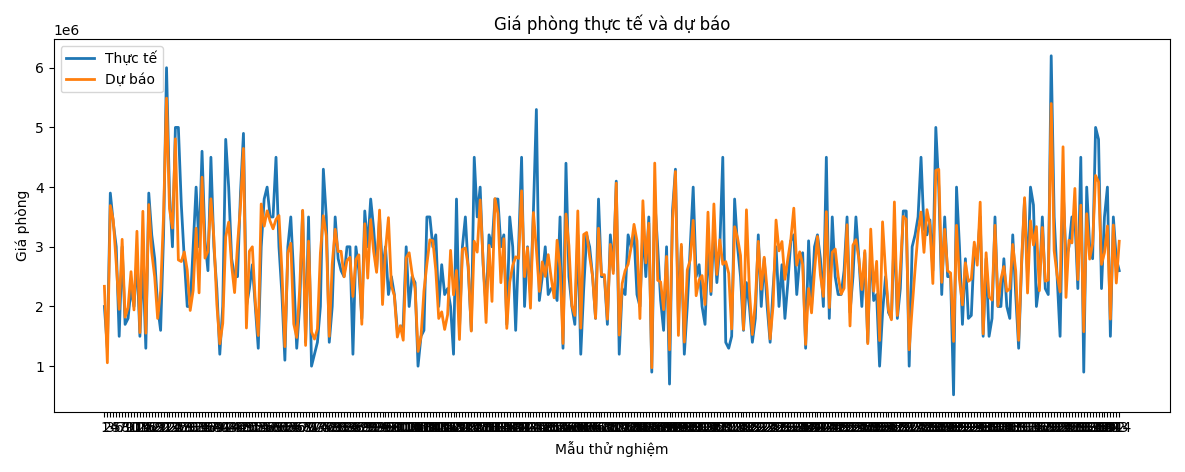
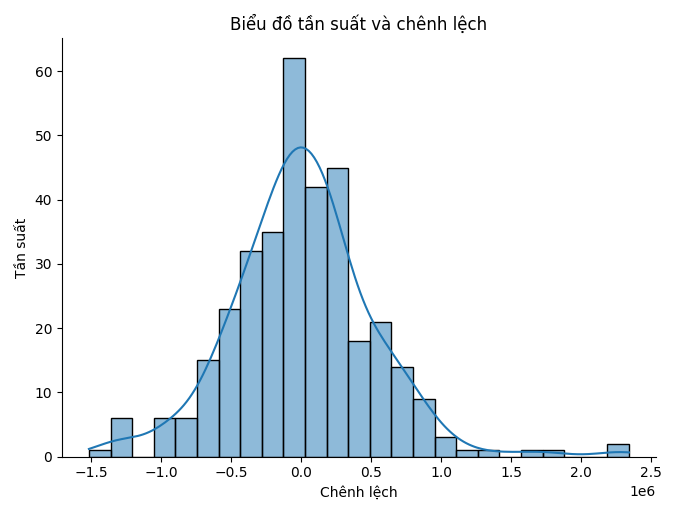
* KNN (K-Nearest Neighbors Regression)



Hình 3.24. Biểu đồ giao động giữa giá thực tế và dự báo KNN

Ta chú ý đến 2 giá trị là MAPE : 17.7% và có độ VarSore: 0.66

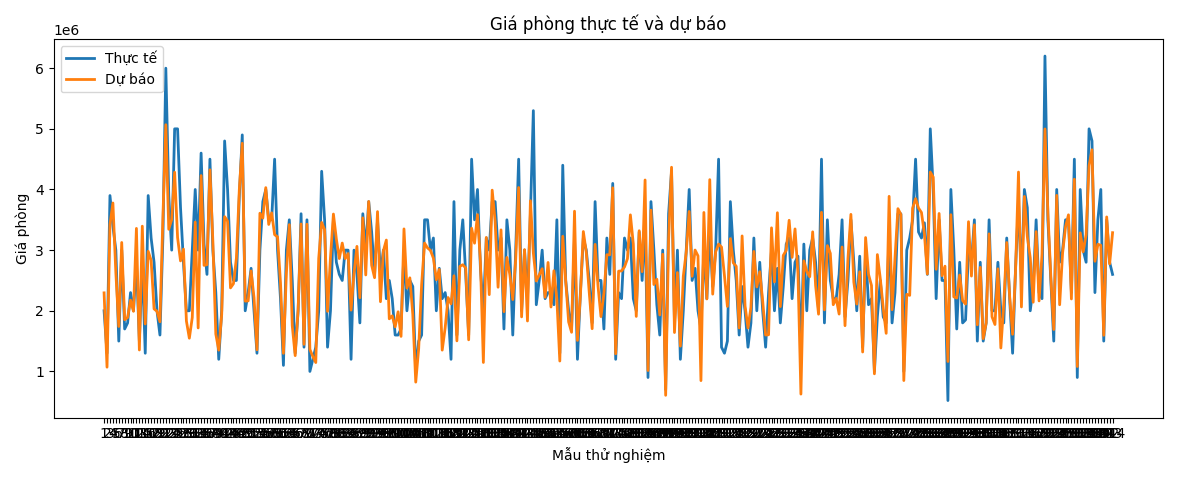
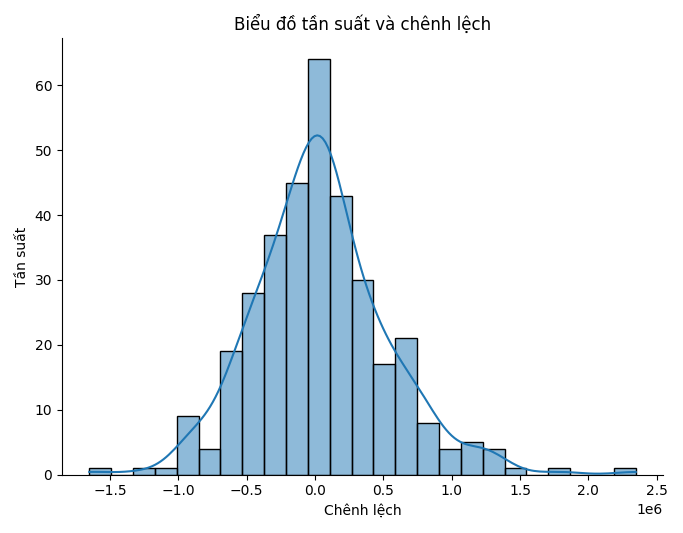
* Random Forest Regressions



Hình 3.25. Biểu đồ giao động giữa giá thực tế và dự báo RFR

Ta chú ý đến 2 giá trị là MAPE : 15.6% và có giá trị VarSore: 0.71

* Multiple Layer Perceptron



Hình 3.26. Biểu đồ giao động giữa giá thực tế và dự báo MLP

Ta chú ý đến 2 giá trị là MAPE : 14.8% và có giá trị VarSore: 0.73

Ta có bảng thống kê như sau trên sử dụng K\_Fold:

|  |  |
| --- | --- |
| Mô hình | MAPE (test) |
| Multiple linear regression | 0.15 +/- 0.01 |
| KNN | 0.18 +/- 0.02 |
| Random Forest Regressions | 0.16 +/- 0.01 |
| Multiple Layer Perceptron | 0.14 +/- 0.01 |

Dựa vào thông tin trên ta có thể đưa ra nhận xét với dữ liệu hiện tại mô hình Multiple Layer Perceptron đang cho định lượng chính xác nhất. Xếp sau lần lượt là Multiple linear regression và Random Forest Regressions. Cuối cùng là KNN. Ta có thể đưa ra kết luận với bộ dữ liệu hiện tại mô hình MLP (học sâu) vẫn sẽ cho mô hình 1 nhất.

# XÂY DỰNG CHƯƠNG TRÌNH ĐỊNH GIÁ

## Phân tích yêu cầu

Chương trình định giá phòng sẽ chỉ có một chức năng chính đó chính là định giá phòng cho thuê. Tức là người dùng sẽ đưa vào các thông số của căn phòng (diện tích, vị trí, tiện nghi,… ), chương trình sẽ sử dụng mô hình đã được huấn luyện để định giá và trả về giá trị cho thuê của căn phòng. Với một chức năng như thế, định giá phòng sẽ là một module trong một hệ thống. Chính vì thế để có thể sử dụng được module này thì việc tạo một API service để cung cấp dịch vụ định giá là một giải pháp hàng đầu được lựa chọn vì nó dễ dàng tích hợp vào các hệ thống có sẵn ví dụ như website dịch vụ hay các ứng dụng trên mobile, desktop khác với đa dạng công nghệ, nền tảng, ngôn ngữ khác nhau.

Công nghệ được lựa chọn cho việc xây dựng API service là framework Flask. Bởi Flask là một framework cho phép chúng ta xây dựng một trang web hay một service đơn giản, nhanh chóng và đặc biệt là viết bằng ngôn ngữ python. Việc này thuận tiện cho chúng ta bơi các model học máy được sử dụng ở trên đều được xây dựng bằng ngôn ngữ Python.

## Xây dựng API định giá

Để xây dựng được API định giá phòng cho thuê chúng ta cần xác định rõ đầu vào và đầu ra của API service.

Đầu vào của API định giá mà một mảng các dictionary gồm có các tham số là các thuộc tính của mô hình cũng với tham số để định danh xác định loại hình bất động sản định giá:

Bảng 4.1. Các thuộc tính đầu vào của API định giá

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** | **Chú thích** |
| 1 | nam (\*) | Int  (Required) | Năm | Nhận giá trị từ năm 2016 đến nay |
| 2 | thang (\*) | Int  (Required) | Tháng |  |
| 3 | dientich(\*) | Float  (Required) | Diện tích | Nhận giá trị từ 1 đến 200 |
| 4 | vido(\*) | Int  (Required) | Vĩ độ | Nhận giá trị từ 20.93 đến 21.09 (Tại khu vực Hà Nội) |
| 5 | kinhdo (\*) | Float  (Required) | Kinh độ | Nhận giá trị từ 105.73 đến 105.93 (Tại khu vực Hà Nội) |
| 6 | loai (\*) | String  (Required) | Loại phòng | Nhận các giá trị: Nhacap, Nhatang, Ccmn |
| 7 | drmd (\*) | Float  (Required) | Độ rộng mặt đường | Nhận giá trị từ 1 đến 100 |
| 8 | kcdc(\*) | Float (Required) | Khoảng các đến đường chính | Nhận giá trị từ 0 đến 2000 |
| 9 | loaiwc | String  (Required) | Loại WC | Nhận các giá trị: KKK, Khepkin |
| 10 | giuongtu | Bool  (Optional) | Có giường hoặc tủ |  |
| 11 | banghe | Bool  (Optional) | Có bàn hoặc ghế |  |
| 12 | nonglanh | Bool  (Optional) | Có bình nóng lạnh hay không |  |
| 13 | dieuhoa | Bool  (Optional) | Có điêu hòa hay không |  |
| 14 | tulanh | Bool  (Optional) | Có tủ lạnh hay không |  |
| 15 | maygiat | Bool  (Optional) | Có máy giặt hay không |  |
| 16 | tivi | Bool  (Optional) | Có tivi hay không |  |
| 17 | bep | Bool  (Optional) | Có bếp hay không |  |
| 18 | gacxep | Bool  (Optional) | Có gác xép hay không |  |
| 19 | thangmay | Bool  (Optional) | Có thang máy hay không |  |
| 20 | bancong | Bool  (Optional) | Có ban công hay không |  |

Đầu ra của API định giá là giá của một căn phòng cho thuê nếu API trả ra giá trị than công hoặc lỗi nếu api trả ra giá trị thất bại:

Bảng 4.2. Danh sách các tham số output của API định giá

|  |  |  |
| --- | --- | --- |
| Thuộc tính | Kiểu giá trị | Ý nghĩa |
| predict | Int | Giá trị mô hình định giá, đơn vị VNĐ |
| success | Bool | Xác định thành công hay lỗi |
| err | Obj | Trả về lỗi của api |

Đầu ra khi API bị lỗi sẽ có dạng:

|  |
| --- |
| {  "err": {  "loai": "Giá trị hợp lệ: [Nhà cấp/ Nhà tầng, Ccmn]",  "nam": "Giá trị năm từ 2016 trở lên",  "thang": "Giá trị tháng [1->12]"  },  "success": false  { |
|  |

Đầu ra khi API thành công sẽ có dạng:

|  |
| --- |
| {  "predict": 1.500.000,  "success": true  { |
|  |

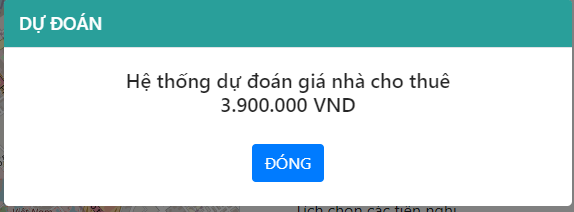
## Xây dựng chương trình minh họa

Chương trình minh họa là một form cho phép người dùng có thể nhập thông tin của căn phòng, chọn vị trí của phòng, độ rộng mặt đường và khoản cách của căn phòng đến đường chính. Từ những thông tin này chương trình sẽ biến đổi để phù hợp với API định giá và gửi request đến API để định giá phòng. Sau khi nhận kết quả từ phía API sẽ được hiển thị cho người dùng. Ở đây ngoài mô hình học sâu chúng ta sẽ có them 3 mô hình khác để có thể so sánh dự đoán của các mô hình với nhau, từ đó biết được hiệu quả của các mô hình dự báo



Hình 4.1. Giao diện chương trình demo API định giá

Kết quả nhận được:



Hình 4.2. Hiển thị kết quả định giá

Ở đây chúng ta sẽ sử dụng từ “AI đinh giá” bởi đối với người dùng phổ thông, khái niện học máy định giá sẽ rất xa vời và khó hiểu đối với người dùng. Còn khi nhắc tới “AI” hay “trí tuệ nhân tạo” người ta sẽ hiểu ngay đây là kết quả của máy tính định lượng. Bởi những khái niện này phổ biến hơn, được truyền thông và đông đảo mọi người nhắc đến nhiều hơn khi quảng cáo sản phẩm.

# KẾT LUẬN

## Kết luận

Những kết quả nghiên cứu và những đóng góp cụ thể trong quá trình làm đồ án đã đạt được như sau:

* Đã xây dựng được bộ dữ liệu giá phòng trên một vài quận, huyện trên địa bàn Hà Nội.
* Đã xây dựng được toàn bộ quy trình xây dựng mô hình định giá phòng cho thuê tại một khu vực từ thu thập dữ liệu, xử lý, xây dựng mô hình đến cung cấp dịch vụ API định giá.
* Đã xây dựng được mô hình được mô hình định giá phòng cho một số quận, huyện của Hà Nội đáp ứng được phạm vi đề tài đã đặt ra.
* Đã xây dựng được API định giá định lượng được giá phòng cho thuê. Đó là một kênh tham khảo giúp các chủ phòng và người đi thuê có thể định lượng giá phòng trọ của mình.
* Đã xây dựng được chương trình demo API định giá cho phép người dùng nhập liệu thông tin về căn phòng và định giá.

Tuy nhiên đề tài vẫn còn tồn tại một số điểm hạn chế, khó khăn như:

* Bộ dữ liệu chưa thực sự hoàn chỉnh hoàn chỉnh.
* Phương pháp khai thác dữ liệu chưa thực sự đạt được hiệu quả cao.
* Độ chính xác của các mô hình chỉ ở mức trung bình 65 -70%.
* Khó khăn trong việc xử lý bộ dữ liệu thực tế vì phụ thuộc vào tính chính xác của người dùng đăng tin.
* Đề tài mới chỉ đánh giá được một vài khu vực trên địa bàn Hà Nội.

## Hướng phát triển

Hướng phát triển của đề tài là:

* Tiếp tục xây dựng hoàn thiện bộ dữ liệu.
* Tiếp tục xây dựng và hoàn thiện phương pháp khai thác dữ liệu tăng hiệu quả của mô hình định giá.
* Tìm hiểu nghiên cứu các phương pháp khác để nâng cao chất lượng của mô hình định giá.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | "Điêu 473 tại Bô Luật dân sự 2015". |
| [2] | "Điều 129 Luật Nhà ở 2014". |
| [3] | Byeonghwa Park, Jae Kwon Bae, "Using machine learning algorithms for housing price prediction: The case of Fairfax County, Virginia housing data," *Expert Systems with Applications,* vol. 42, pp. 2928-2934, 2015. |
| [4] | Tiêu chuẩn thẩm định giá Việt Nam số 02 Giá trị thị trường làm cơ sở cho thẩm định giá (Ký hiệu: TĐGVN 02), “Ban hành kèm theo Thông tư số 158/2014/TT-BTC ngày 27 tháng 10 năm 2014 của Bộ trưởng Bộ Tài chính”. |
| [5] | Vũ Đức Độ (2020) Báo cáo "Nghiên cứu và ứng dụng học máy cho định giá nhà**"**. |