|  |  |  |
| --- | --- | --- |
|  | TRƯỜNG ĐẠI HỌC BÁCH KHOA  KHOA CÔNG NGHỆ THÔNG TIN |  |

TIỂU LUẬN

KHOA HỌC DỮ LIỆU

**DỰ ĐOÁN LƯƠNG CỦA DEVELOPER Ở CHÂU Á**

Giảng viên chấm: ……………………………………………….

|  |  |  |
| --- | --- | --- |
| HỌ VÀ TÊN SINH VIÊN | LỚP HỌC PHẦN | ĐIỂM BẢO VỆ |
| Nguyễn Kim Huy | 18N15 |  |
| Trần Ngọc Huy | 18N15 |  |
| Nguyễn Nhật Tùng | 18N15 |  |

ĐÀ NẴNG, 08/2021

**TÓM TẮT**

Lấy cảm hứng từ vấn đề sinh viên ngành công nghệ thông tin ra trường không biết lương của mình có thể đạt bao nhiêu và việc những người có ý định chuyển sang ngành công nghệ thông tin muốn biết mức lương của developer là bao nhiêu dựa vào các loại tham số như vị trí, số năm kinh nghiệm cần để có mức lương đó, ... Vì vậy nhóm đã ứng dụng kiến thức về khoa học dữ liệu và học máy để xây dựng mô hình dự đoán mức lương của developer dựa vào vị trí làm việc (loại developer), quốc gia làm việc trong khu vực Châu Á, trình độ giáo dục và số năm kinh nghiệm của developer. Kết quả là mô hình hồi quy tuyến tính hoạt động tốt nhất với sai số tuyệt đối trung bình là 15,068.969 (USD); mô hình mạng nơ-ron nhân tạo nhiều lớp hoạt động kém nhất với sai số tuyệt đối trung bình là 25,867.952 (USD). Mô hình cây quyết định dự đoán tốt trên bộ dữ liệu train nhưng rất tệ trên bộ dữ liệu test. Giải pháp của nhóm có thể phát triển thêm bằng cách tăng thêm tham số, thu thập thêm dữ liệu, cân bằng các mẫu dữ liệu giữa các quốc gia nhằm giảm sai số của các mô hình học máy, giúp dự đoán mức lương chính xác hơn.

**BẢNG PHÂN CÔNG NHIỆM VỤ**

|  |  |  |
| --- | --- | --- |
| Sinh viên thực hiện | Các nhiệm vụ | Tự đánh giá |
| Nguyễn Kim Huy | Thu thập dữ liệu.  Viết báo cáo phần thu thập dữ liệu và mô tả (trang 6-12). | Hoàn thành |
| Trần Ngọc Huy | Trích xuất đặc trưng.  Viết báo cáo phần trích xuất đặc trưng (trang 13-18). | Hoàn thành |
| Nguyễn Nhật Tùng | Mô hình hóa dữ liệu.  Viết báo cáo phần mô hình hóa dữ liệu (trang 19-26). | Hoàn thành |

**MỤC LỤC**

[1. Giới thiệu 5](#_Toc79524531)

[2. Thu thập và mô tả dữ liệu 5](#_Toc79524532)

[2.1 Thu thập dữ liệu 5](#_Toc79524533)

[a) Nguồn dữ liệu 5](#_Toc79524534)

[b) Công cụ 5](#_Toc79524535)

[c) Dữ liệu đầu vào 5](#_Toc79524536)

[d) Dữ liệu đầu ra 5](#_Toc79524537)

[e) Ví dụ minh họa 6](#_Toc79524538)

[2.2 Mô tả dữ liệu 8](#_Toc79524539)

[3. Trích xuất đặc trưng 13](#_Toc79524540)

[3.1 Xử lí dữ liệu ban đầu 13](#_Toc79524541)

[3.2 Làm sạch dữ liệu 13](#_Toc79524542)

[3.3 Xử lí trường “DevType” 14](#_Toc79524543)

[3.4 Xử lí trường “Country” 15](#_Toc79524544)

[3.5 Xử lí trường “EdLevel” 16](#_Toc79524545)

[3.6 Xử lí trường “YearsCode” 17](#_Toc79524546)

[3.7 Tạo bộ dữ liệu đã được xử lí 18](#_Toc79524547)

[4. Mô hình hóa dữ liệu 19](#_Toc79524548)

[4.1 Chia bộ dữ liệu train/test 19](#_Toc79524549)

[4.2 Các mô hình sử dụng 19](#_Toc79524550)

[4.3 Biểu đồ hiệu suất train, validate của từng mô hình 24](#_Toc79524551)

[4.4 Kết quả huấn luyện 25](#_Toc79524552)

[5. Kết luận 26](#_Toc79524553)

[6. Tài liệu tham khảo 26](#_Toc79524554)

# Giới thiệu

Lấy cảm hứng từ vấn đề sinh viên ngành công nghệ thông tin ra trường không biết lương của mình có thể đạt bao nhiêu và việc những người có ý định chuyển sang ngành công nghệ thông tin muốn biết mức lương của developer là bao nhiêu dựa vào các loại tham số như vị trí, số năm kinh nghiệm cần để có mức lương đó, ....

Để giải quyết vấn đề này nhóm đã ứng dụng kiến thức về khoa học dữ liệu và học máy để xây dựng mô hình dự đoán mức lương của developer dựa vào vị trí làm việc (loại developer), quốc gia làm việc trong khu vực Châu Á, trình độ giáo dục và số năm kinh nghiệm của developer.

Quy trình giải quyết vấn đề gồm có 3 bước. Bắt đầu bằng việc thu thập dữ liệu điều tra developer trên toàn thế giới năm 2020. Sau đó là khám phá dữ liệu, trích xuất đặc trưng, giảm chiều và chuẩn hóa. Cuối cùng là thiết lập và huấn luyện các mô hình khác nhau để dự đoán mức lương.

# Thu thập và mô tả dữ liệu

## Thu thập dữ liệu

### Nguồn dữ liệu

* Dữ liệu được thu thập trên trang: <http://stackoverflow.itoa.vn>.

### Công cụ

* Môi trường: Node.js.
* Thư viện: Puppeteer.js.
* Trình soạn thảo văn bản (editor): Visual Studio Code.

### Dữ liệu đầu vào

* Địa chỉ URL trang web chứa nguồn dữ liệu: <http://stackoverflow.itoa.vn>.

### Dữ liệu đầu ra

* Tập tin chứa toàn bộ dữ liệu thu thập được trên trang web, có định dạng file comma-separated values (CSV).

### Ví dụ minh họa

#### Mô tả quá trình cấu trúc JSON từ một URL đầu vào

URL

HTML

JSON

puppeteer.js

HTML DOM

Hình 1. Sơ đồ khối quy trình cấu trúc ra JSON từ một URL.

* Sử dụng thư viện puppeteer.js để lấy HTML từ một URL.
* Thông qua puppeteer.js, chạy javascript can thiệp vào HTML DOM để lấy dữ liệu.

#### Mô tả cách thức tổng hợp dữ liệu

* Dùng puppeteer.js duyệt qua tất cả các URL. Theo bước (i) lấy kết quả và ghi JSON có được vào CSV.

URL

HTML

JSON

CSV

HTML

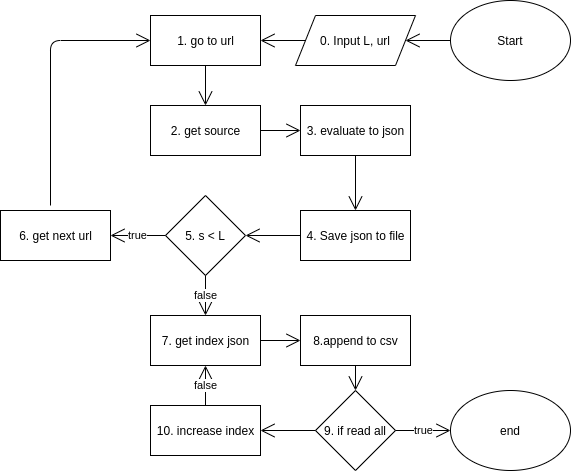
HTML

JSON

JSON

Hình 1. Sơ đồ khối quy trình tổng hợp dữ liệu.

#### Tổng quan thuật toán



Hình 2. Sơ đồ khối tổng hợp 2 quá tình trên.

* Mô tả thuật toán:

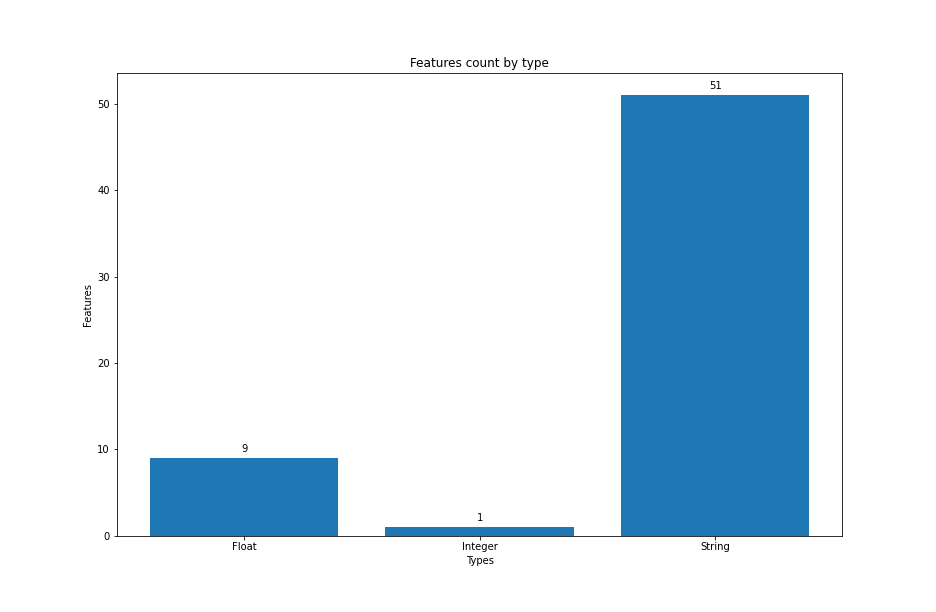
1. Mở kết nối đến trang (ban đầu là <http://stackoverflow.itoa.vn>).
2. Nhận source HTML từ phản hồi.
3. Nhận dạng, đánh giá HTML để đưa các thông tin cần thiết vào một JSON.
4. Lưu JSON vừa nhận được ra file.
5. Kiểm tra số lượng file (số mẫu) còn thiếu hay không?.
6. Nếu còn thiếu mẫu, tìm URL trang tiếp theo và quay lại bước 1.
7. Nếu đã đủ, đọc dữ liệu từ file JSON đang trỏ (con trỏ bắt đầu từ 0).
8. Điền dữ liệu từ file JSON thành một row của file CSV.
9. Kiểm tra file JSON đang trỏ có phải là file cuối hay không.
10. Nếu không, tăng con trỏ và quay lại bước 7.
11. Nếu đúng, kết thúc.

## Mô tả dữ liệu

* Bộ dữ liệu thu thập được có số đặc trưng là 61 đặc trưng, số mẫu là 64,000 mẫu. Kiểu dữ liệu của mỗi đặc trưng được cho theo bảng dưới. Biểu đồ cột biểu diễn số các đặc trưng theo kiểu dữ liệu được cho theo sơ đồ ở hình 4.

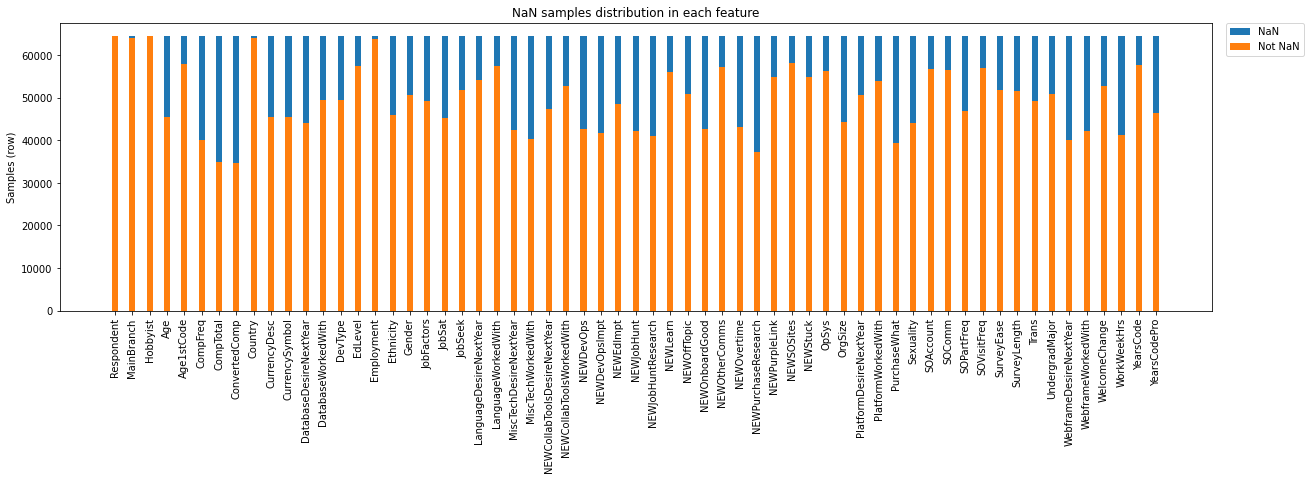
|  |  |
| --- | --- |
| Kiểu dữ liệu | Tên đặc trưng |
| Float | "Age", "CompTotal", "ConvertedComp", "NEWJobHunt", "NEWJobHuntResearch", "NEWOnboardGood", "PurchaseWhat", "SOPartFreq", "WorkWeekHrs". |
| Integer | "Respondent". |
| String | "MainBranch", "Hobbyist", "Age1stCode", "CompFreq", "Country", "CurrencyDesc", "CurrencySymbol", "DatabaseDesireNextYear", "DatabaseWorkedWith", "DevType", "EdLevel", "Employment", "Ethnicity", "Gender", "JobFactors", "JobSat", "JobSeek", "LanguageDesireNextYear", "LanguageWorkedWith", "MiscTechDesireNextYear", "MiscTechWorkedWith", "NEWCollabToolsDesireNextYear", "NEWCollabToolsWorkedWith", "NEWDevOps", "NEWDevOpsImpt", "NEWEdImpt", "NEWLearn", "NEWOffTopic", "NEWOtherComms", "NEWOvertime", "NEWPurchaseResearch", "NEWPurpleLink", "NEWSOSites", "NEWStuck", "OpSys", "OrgSize", "PlatformDesireNextYear", "PlatformWorkedWith", "Sexuality", "SOAccount", "SOComm", "SOVisitFreq", "SurveyEase", "SurveyLength", "Trans", "UndergradMajor", "WebframeDesireNextYear", "WebframeWorkedWith", "WelcomeChange", "YearsCode", "YearsCodePro". |

Bảng 1. Bảng mô tả kiểu dữ liệu của các đặc trưng.



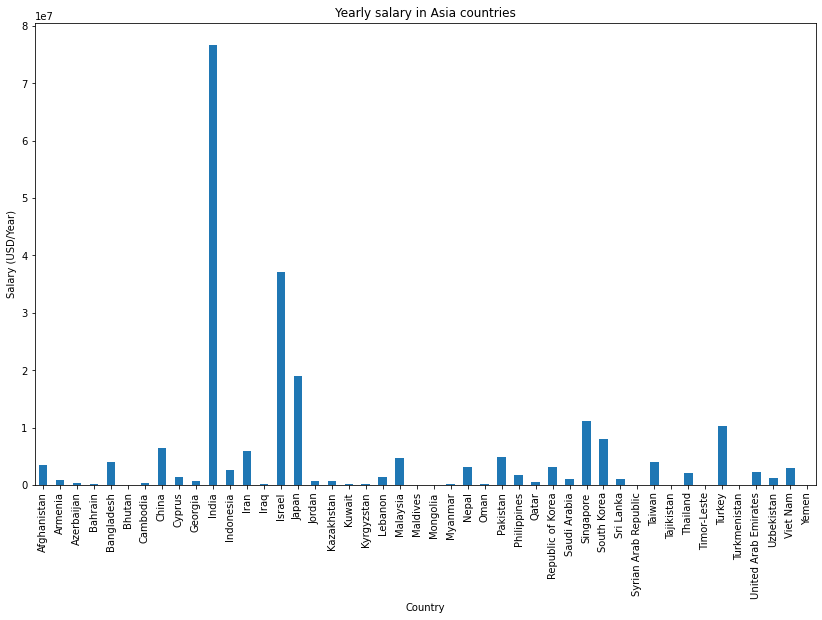
Hình 3. Biểu đồ cột số đặc trưng theo kiểu dữ liệu.

* Số mẫu dữ liệu trống của mỗi đặc trưng được mô tả bằng biểu đồ cột dưới đây.

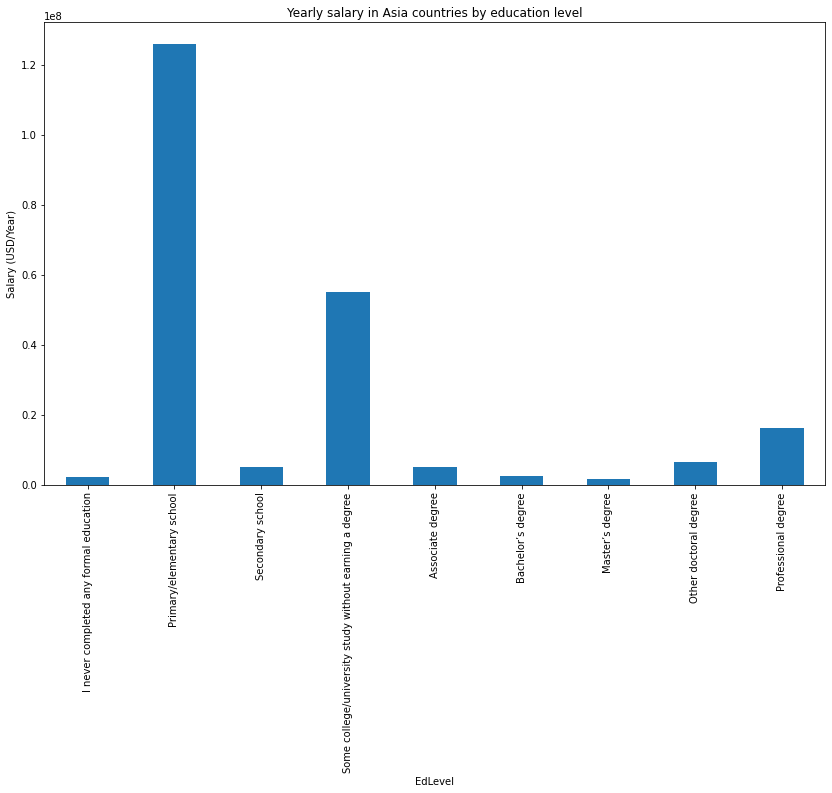


Hình 4. Biểu đồ cột số mẫu dữ liệu trống trên tổng số mẫu của các đặc trưng.

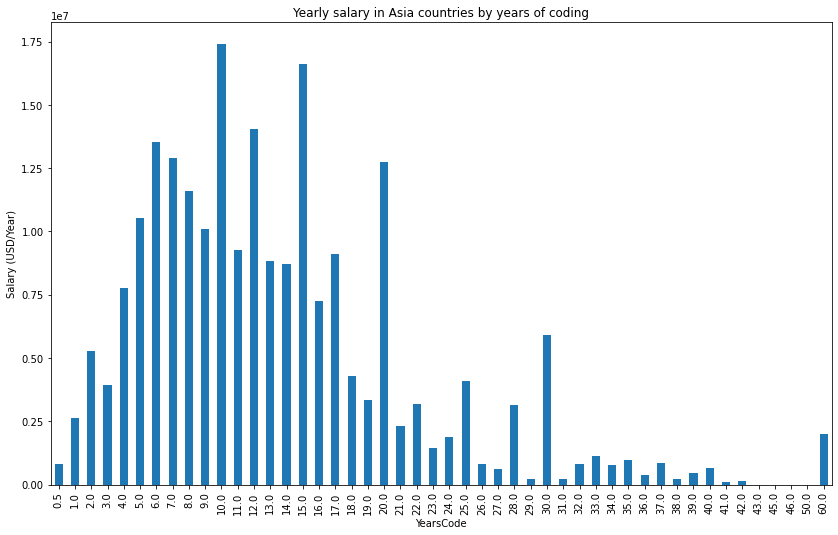
* Các thông kê mô tả trực quan về một vài đặc trưng được mô tả bằng cách biểu đồ dưới đây.



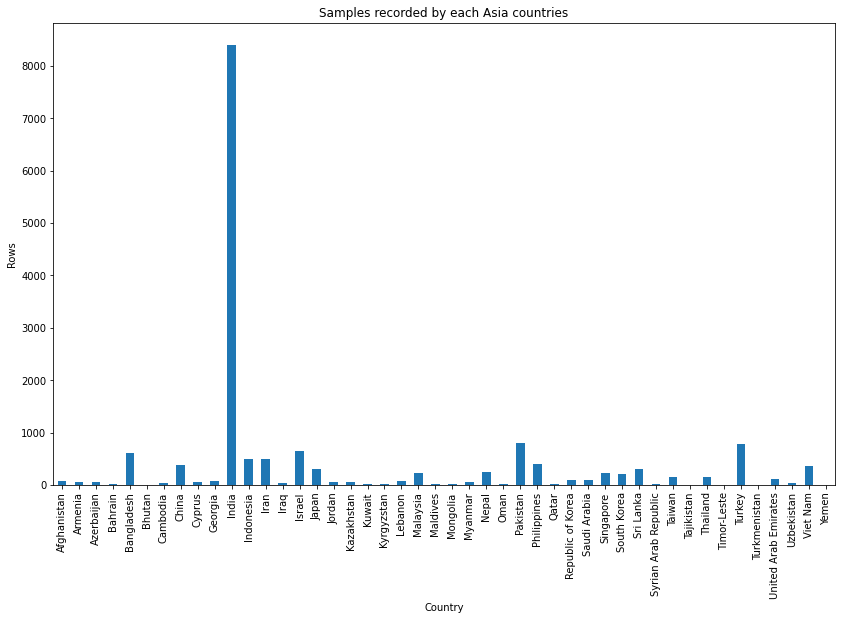
Hình 5. Biểu đồ cột thể hiện mức lương hằng năm giữa các quốc gia ở Châu Á.



Hình 6. Biểu đồ cột thể hiện quan hệ giữa mức lương và mức học vấn ở Châu Á.



Hình 7. Biểu đồ cột thể hiện quan hệ giữa mức lương và số năm kinh nghiệm ở Châu Á.



Hình 8. Biểu đồ cột thể hiện số lượng mẫu theo các quốc gia ở Châu Á.

# Trích xuất đặc trưng

## Xử lí dữ liệu ban đầu

Với dữ liệu đầu vào gồm 64461 hàng × 61 cột, nhóm tiến hành giảm chiều dữ liệu bằng cách loại bỏ những hàng có quốc gia không nằm ở châu Á. Đồng thời, dataset chỉ giữ lại những trường (field) liên quan, đó là 5 trường: “ConvertedComp”, “Country”, “DevType”, “YearsCode”, “EdLevel”. Kết quả thu được là bộ dữ liệu con gồm 16278 hàng × 5 cột:

Graphical user interface, text, application

Description automatically generated

Hình 9. Kết quả xử lý dữ liệu ban đầu.

## Làm sạch dữ liệu

Nhóm làm sạch dữ liệu bằng cách xoá các quan sát có bất kì phần tử dữ liệu nào bị bỏ trống (xoá các phần tử NaN). Kết quả thu được bộ dữ liệu bị giảm số hàng còn 5645 hàng:

Graphical user interface, text, application, email

Description automatically generated

Hình 10. Kết quả xử lý dữ liệu đã được làm sạch.

## Xử lí trường “DevType”

Trong bộ dữ liệu, trường “DevType” có kiểu dữ liệu là xâu (string) tuy nhiên giá trị của nó lại có dạng danh sách các xâu cách nhau bởi dấu chấm phẩy (‘;’). Ví dụ:

Graphical user interface, text, application

Description automatically generated

Hình 11. Trường “DevType” trước khi được xử lí.

Nhằm mã hóa trường “DevType”, Nhóm sử dụng mô hình Bag of Words Model. Ý tưởng của thuật toán là biểu diễn cho mỗi mẫu dữ liệu văn bản dưới dạng một vector số, trong đó mỗi chiều là một từ cụ thể trong kho dữ liệu và giá trị có thể là tần số của nó xuất hiện trong đoạn văn bản (giá trị có thể là 0 hoặc 1).

Table

Description automatically generated

Hình 4. Kết quả mã hóa trường “DevType”.

Sau đó, chuyển kết quả thu được thành cấu trúc dữ liệu dataframe, đồng thời giảm chiều (số hàng) bằng cách chỉ lấy những trường bắt đầu bằng “Developer”. Kết quả: Table

Description automatically generated

Hình 5. Kết quả bước xử lý trường “DevType”.

Cuối cùng, lọc dataset ban đầu dựa vào chỉ số hàng (index) của dataframe đã được xử lí. Mục đích của việc này là để index của các dataframe của các trường khác sẽ trùng khớp với nhau. Kết quả thu được:

Graphical user interface, text, application

Description automatically generated

Hình 6. Kết quả lọc dataset ban đầu dựa vào index của dataframe vừa xử lý.

## Xử lí trường “Country”

Nhóm mã hóa trường “Country” có kiểu dữ liệu xâu (string) trong bộ dữ liệu bằng cách sử dụng kĩ thuật mã hoá One-hot encoding. Cụ thể, One-hot encoding là quá trình biến đổi từng giá trị thành các đặc trưng nhị phân chỉ chứa giá trị 1 hoặc 0. Mỗi mẫu trong đặc trưng phân loại sẽ được biến đổi thành một vecto có kích thước m chỉ với một trong các giá trị là 1 (biểu thị nó là active) . Dùng hàm fit() và transform() của lớp OneHotEncoder của thư viện sklearn.preprocessing. Sau đó, chuyển kết quả thu được thành cấu trúc dữ liệu dataframe:

Table

Description automatically generated

Hình 7. Trường “Country” trước khi được xử lí.

Table

Description automatically generated

Hình 8. Kết quả bước xử lý trường “Country”.

## Xử lí trường “EdLevel”

Theo quan sát, nhóm thấy đây là dữ liệu phân loại theo thứ tự. Nhóm dùng map() của pandas để ánh xạ các đặc trưng này thành dạng số theo trình độ học vấn của người được phỏng vấn. Cách thức ánh xạ cụ thể được cho theo bảng bên dưới. Kết quả thu được được chuyển thành cấu trúc dữ liệu dataframe. Kết quả thu được:

Text, table

Description automatically generated

Hình 9. Trường “EdLevel” trước khi được xử lí.

Graphical user interface, application

Description automatically generated

Hình 10. Kết quả bước xử lý trường “EdLevel”.

|  |  |
| --- | --- |
| Giá trị ban đầu | Giá trị mã hóa |
| I never completed any formal education | 0 |
| Primary/elementary school | 1 |
| Secondary school (e.g. American high school, German Realschule or Gymnasium, etc.) | 2 |
| Some college/university study without earning a degree | 3 |
| Associate degree (A.A., A.S., etc.) | 4 |
| Bachelor’s degree (B.A., B.S., B.Eng., etc.) | 5 |
| Master’s degree (M.A., M.S., M.Eng., MBA, etc.) | 6 |
| Other doctoral degree (Ph.D., Ed.D., etc.) | 7 |
| Professional degree (JD, MD, etc.) | 8 |

Bảng 2. Cách thức ánh xạ nhằm mã hóa trường “EdLevel”.

## Xử lí trường “YearsCode”

Các phần tử dữ liệu có giá trị là “More than 50 years” và “Less than 1 year” sẽ lần lượt được chuyển thành 60 và 0.5. Chuyển kết quả thu được thành cấu trúc dữ liệu dataframe:

Table

Description automatically generated

Hình 12. Trường “YearsCode” trước khi được xử lí.

Table

Description automatically generated

Hình 13. Kết quả bước xử lý trường “YearsCode”.

## Tạo bộ dữ liệu đã được xử lí

Nối các trường đã được xử lí ở trên với trường “ConvertedComp” ban đầu để tạo thành bộ dữ liệu hoàn chỉnh để đùng cho quá trình train. Kết quả dataset cuối cùng thu được:

Table

Description automatically generatedHình 14. Bộ dữ liệu hoàn chỉnh đã xử lý.

# Mô hình hóa dữ liệu

## Chia bộ dữ liệu train/test

Bộ dữ liệu sau khi xử lý (sau đây gọi bộ dữ liệu ban đầu) được chia thành 2 bộ dữ liệu train và bộ dữ liệu test theo tỷ lệ tương ứng là 8:2. Chi tiết kích thước các bộ dự liệu được tính theo bảng 1.

|  |  |  |  |
| --- | --- | --- | --- |
| Tỷ lệ train/test = 8/2 | | | |
|  | Bộ dữ liệu ban đầu | Bộ dữ liệu train | Bộ dữ liệu test |
| Kích thước (bản ghi) | 5358 | 4286 | 1072 |

Bảng 1. Kích thước các bộ dữ liệu.

## Các mô hình sử dụng

1. Cây quyết định (Decision tree)
2. Cơ sở lý thuyết:

Thuật toán tạo ra mô hình học có cấu trúc dạng cây, mỗi nút trong cây nếu là nút nhánh sẽ chứa giá trị ngưỡng và chỉ số thuộc tính cần so sánh; nếu là nút lá sẽ chứa nhãn hoặc giá trị dự đoán được.

Thuật toán xây dựng cây quyết định có các bước cơ bản như sau [1]:

1. Khởi tạo tập thuộc tính S bằng tập thuộc tính train; khởi tạo cây rỗng T.
2. Lựa chọn thuộc tính tốt nhất nhất (attrib) trong tập S bằng cách đánh giá tập thuộc tính S theo một tiêu chí nhất định và lựa chọn ra thuộc tính có giá trị đánh giá “tốt nhất”. Hai tiêu chí thường dùng để lựa chọn thuộc tính cho nút là Entropy và Information Gain (nếu dự đoán dữ liệu rời rạc) hoặc Gini index (nếu dự đoán dữ liệu liên tục).
3. Tạo nút lá hoặc nút nhánh mới trong cây tùy vào giá trị đánh giá của attrib.
4. Chia nhỏ tập S thành tập thuộc tính S’.
5. Với mỗi nhánh mới của nút, quay lại bước 2 với S=S’ và cây T=nút N.

Để dự đoán, thuật toán thực hiện duyệt cây từ nút gốc. Ở mỗi nút, thuật toán chọn nhánh dựa vào kết quả phép so sánh giữa giá trị ngưỡng lưu trong nút đó với giá trị của thuộc tính ở bộ dữ liệu input tương ứng chỉ số thuộc tính lưu trong nút. Thuật toán trả về giá trị nút lá đầu tiên tìm được.

1. Các tham số dùng để xây dựng mô hình được cài đặt theo hàm API của thư viện sklearn như bảng sau.

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên tham số | Giá trị | Mô tả |
| 1 | criterion | 'mse' | Hàm số đánh giá chất lượng của nút nhánh. |
| 2 | splitter | 'best' | Phương pháp phân nhánh một nút. |
| 3 | max\_depth | None | Độ sâu tối đa của cây. |
| 4 | min\_samples\_split | 2 | Số mẫu tối thiểu để tách một nút nhánh. |
| 5 | min\_samples\_leaf | 1 | Số mẫu tối thiểu để trở thành nút lá. |
| 6 | min\_weight\_fraction\_leaf | 0.0 | Phân số có trọng số tối thiểu của tổng các trọng số để giữ lại ở nút lá. |
| 7 | max\_features | None | Số thuộc tính xét đến khi tách nhánh. |
| 8 | random\_state | None | Điều khiển mức độ ngẫu nhiên của mô hình học. |
| 9 | max\_leaf\_nodes | None | Số nút lá tối đa. |
| 10 | min\_impurity\_decrease | 0.0 | Ngưỡng pha tạp để trở thành nút nhánh. |
| 11 | ccp\_alpha | 0.0 | Tham số độ phức tạp dùng cho thuật toán tỉa cây Minimal Cost-Complexity Pruning. |

Bảng 2. Chi tiết tham số hàm API tạo mô hình cây quyết định.

1. Hồi quy tuyến tính (Linear regression)
2. Cơ sở lý thuyết:

Thuật toán giả định mối quan hệ tuyến tính giữa tập thuộc tính đầu vào (*X ∈ Rn*) và thuộc tính đầu ra (*y ∈ R*) theo công thức:

Khi thực hiện, thuật toán lặp lại việc cập nhật các trọng số (*W ∈ Rn*) và bias (*b ∈ R*) dựa vào giá trị hàm lỗi sao cho hàm chi phí đạt giá trị nhỏ nhất. Thông thường, các trọng số và bias được đặt bằng 0 hoặc được đặt giá trị ngẫu nhiên; hàm lỗi thường được sử dụng như Linear Least Squares, Sigmoid Function, …; hàm cost thường là giá trị trung bình của hàm lỗi cho mỗi mẫu dữ liệu.

1. Các tham số dùng để xây dựng mô hình được cài đặt theo hàm API của thư viện sklearn như bảng 3. Hàm lỗi được sử dụng là hàm Ordinary Least Squares có công thức như sau:

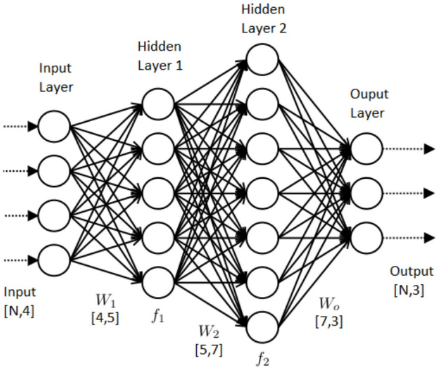
[2]

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên tham số | Giá trị | Mô tả |
| 1 | fit\_intercept | True | Chỉ định có nên tính intercept hay không. |
| 2 | normalize | False | Chỉ định có nên chuẩn hóa dữ liệu trước hay không. |
| 3 | copy\_X | True | Chỉ định có nên copy bộ dữ liệu train hay không. |
| 4 | n\_jobs | None | Số tác vụ thực hiện tính toán. |
| 5 | positive | False | Chỉ định các trọng số đều dương hay không. |

Bảng 3. Chi tiết tham số hàm API tạo mô hình hồi quy tuyến tính.

1. Mạng nơ-ron nhân tạo nhiều lớp (Multi-layer Perceptron Regression, MLP)
2. Cơ sở lý thuyết:

Mô hình được sử dụng để “học” một hàm số phi tuyến tính , với *m* là số chiều của input và *o* là số chiều của output. Khác với mô hình hồi quy tuyến tính, MLP có một hoặc nhiều hơn các lớp phi tuyến tính (lớp ẩn, hidden layer). Mỗi nơ-ron thứ *i* ở một lớp được nối với mọi nơ-ron thứ *j* ở lớp tiếp theo và được gán các trọng số *wij*; mỗi lớp *l* trong mô hình có *d* nơ-ron thì có *d* bias, .



Hình 1. Ví dụ về một mạng MLP với số chiều cụ thể.

Mỗi nơ-ron thứ *j* ở lớp ẩn thứ *l* biến đổi outputcủa các nơ-ron ở lớp *(l-1)* trước đó theo công thức:

[3]

Output này sau đó được áp dụng một hàm kích hoạt phi tuyến tính để tính toán ra giá trị output cuối cùng của nơ-ron thứ *i* ở lớp thứ *l*:

Các nơ-ron ở lớp output có thể không có hàm kích hoạt mà trả về giá trị output .

Về cơ bản, với một tập *N* bộ dữ liệu input , thuật toán thực hiện 4 bước:

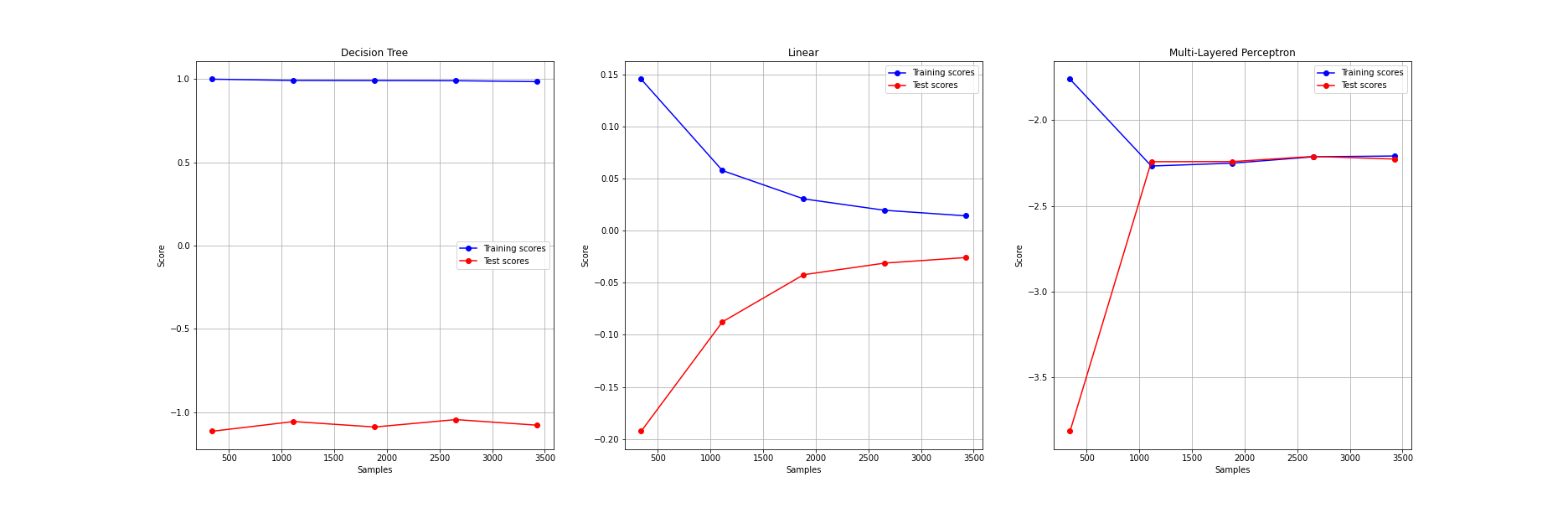
1. Bước feedforward: Với một bộ dữ liệu input , tính giá trị đầu ra của network, trong quá trình tính toán, lưu lại các activation *a(l)* tại mỗi layer*.*
2. Tính đạo hàm riêng giá trị hàm lỗi cho mỗi nơ-ron thứ *j* ở output layer .
3. Cập nhật trọng số và bias ở output layer.
4. Làm tương tự bước 2 và bước 3 cho các layer *l = L-1, L-2,…1*, với L = Lhidden+1, Lhidden là số layer ẩn của mô hình.
5. Các tham số dùng để xây dựng mô hình được cài đặt theo hàm API của thư viện sklearn như bảng sau.

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên tham số | Giá trị | Mô tả |
| 1 | hidden\_layer\_sizes | (64,64,64) | Chỉ định kích thước các lớp perceptron ẩn. |
| 2 | activation | 'relu' | Chỉ định hàm kích hoạt được sử dụng. |
| 3 | solver | 'adam' | Chỉ định hàm tối ưu trọng số. |
| 4 | alpha | 0.0001 | Tham số thuật toán L2 penalty. |
| 5 | batch\_size | 'auto' | Chỉ định số batch hoặc mặc định là min(200, số mẫu input). |
| 6 | learning\_rate | 'constant' | Chỉ định tính chất của tốc độ học. |
| 7 | learning\_rate\_init | 0.001 | Tốc độ học ban đầu. |
| 8 | max\_iter | 8572 | Số vòng lặp tối đa của thuật toán hồi quy. |
| 9 | shuffle | True | Chỉ định có tráo đổi thứ tự các mẫu dữ liệu train hay không. |
| 10 | random\_state | None | Điều khiển tính ngẫu nhiên khi khởi tạo trọng số và bias. |
| 11 | verbose | False | Chỉ định in tiến độ quá trình huấn luyện hay không. |
| 12 | warm\_start | False | Chỉ định sử dụng lại kết quả của lần huấn luyện trước hay không. |
| 13 | early\_stopping | False | Chỉ định sử dụng tham số ‘tol’ để kết thúc hồi quy sớm hay không. |
| 14 | beta\_1 | 0.9 | Tốc độ decay của vector mô-men thứ nhất khi dùng hàm tối ưu trọng số ‘adam’. |
| 15 | beta\_2 | 0.999 | Tốc độ decay của vector mô-men thứ hai khi dùng hàm tối ưu trọng số ‘adam’. |
| 16 | epsilon | 1e-08 | Giá trị ổn định số học của thuật toán ‘adam’. |

Bảng 4. Tham số mô hình MLP.

## Biểu đồ hiệu suất train, validate của từng mô hình

Biểu đồ hiệu suất trên bộ dữ liệu train và validate, được tạo bằng cách chia bộ dữ liệu train ban đầu thành 5 phần, mỗi phần có kích thước lần lượt bằng 0.1, 0.33, 0.55, 0.78, 1.0 lần bộ dữ liệu train. Sau đó thuật toán 5-fold cross-validation [4] được áp dụng cho mỗi phần để vẽ biểu đồ hiệu suất ở hình 1. Chi tiết kích thước bộ dữ liệu train và validate của quá trình được tính theo bảng 3.

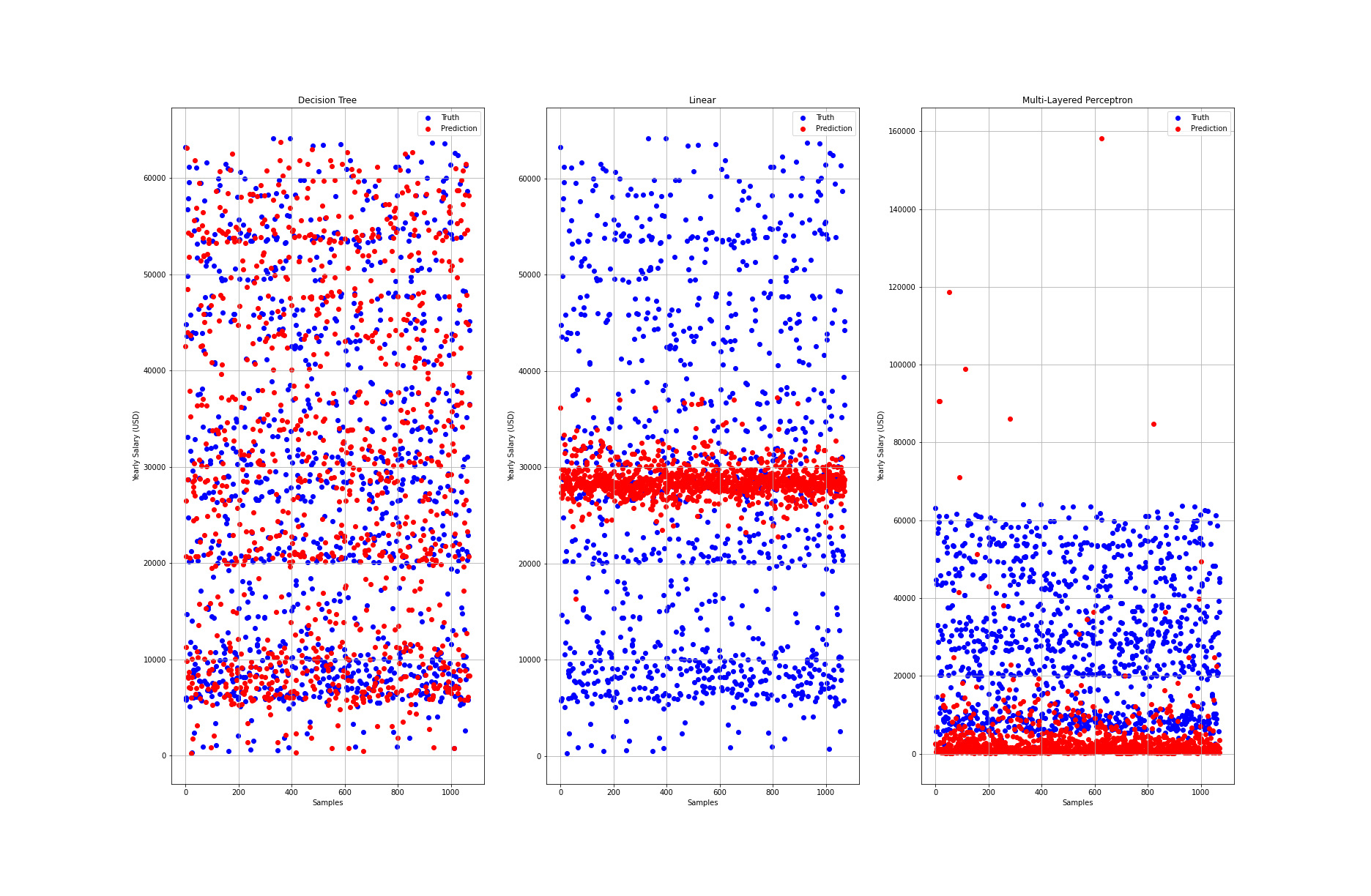
Hình 2. Biểu đồ train, validate của các mô hình.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Kích thước bộ dữ liệu train = 4286* | | | | |
| STT phần | Tỷ lệ kích thước | Kích thước bộ dữ liệu | Kích thước bộ dữ liệu train | Kích thước bộ dữ liệu validate |
| 1 | 0.1 | 428 | 342 | 86 |
| 2 | 0.33 | 1414 | 1114 | 283 |
| 3 | 0.55 | 2357 | 1885 | 472 |
| 4 | 0.78 | 3343 | 2656 | 669 |
| 5 | 1.0 | 4286 | 3428 | 858 |

Bảng 5. Chi tiết kích thước bộ dữ liệu train, validate.

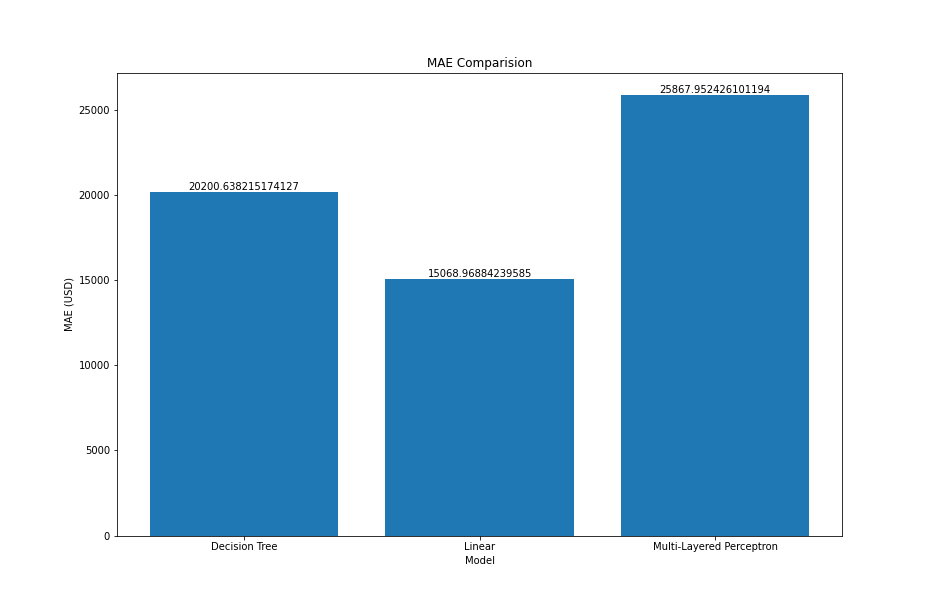
## Kết quả huấn luyện

Kết quả dự đoán từ 1072 mẫu được lấy từ bộ dữ liệu test được vẽ trên biểu đồ scatter ở hình 3.



Hình 3. Biểu đồ so sánh kết quả dự đoán

Sai số tuyệt đối trung bình (mean absolute error, MAE) của mỗi mô hình dựa trên 1072 mẫu trên được vẽ trên biểu đồ cột ở hình 3.



Hình 4. Biểu đồ sai số giữa của cácmô hình.

# Kết luận

* Mô hình hồi quy tuyến tính dự đoán tốt nhất với sai số tuyệt đối trung bình là 15,068.969 (USD); mô hình mạng nơ-ron nhân tạo nhiều lớp dự đoán kém nhất với sai số tuyệt đối trung bình là 25,867.952 (USD). Mô hình cây quyết định dự đoán tốt trên bộ dữ liệu train nhưng rất tệ trên bộ dữ liệu test.
* Giải pháp của nhóm có thể phát triển thêm bằng cách tăng thêm tham số, thu thập thêm dữ liệu, cân bằng các mẫu dữ liệu giữa các quốc gia nhằm giảm sai số của các mô hình học máy, giúp dự đoán mức lương chính xác hơn.

# Tài liệu tham khảo

1. Chauhan, N. S. (2020). *Decision Tree Algorithm, Explained*. Kdnuggets. <https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html>.
2. Wikipedia contributors. (2021, July 18). Ordinary least squares. In *Wikipedia, The Free Encyclopedia*. Retrieved 14:24, August 4, 2021, from <https://en.wikipedia.org/w/index.php?title=Ordinary_least_squares&oldid=1034182640>.
3. Tiep, V. H. (2017, Feb 24). *Bài 14: Multi-layer Perceptron và Backpropagation*. Machine Learning cơ bản. <https://machinelearningcoban.com/2017/02/24/mlp/>.
4. Brownlee, J. (2018, May 23). *A Gentle Introduction to k-fold Cross-Validation*. Machine Learning Mastery. <https://machinelearningmastery.com/k-fold-cross-validation/>.