

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH



ĐỒ ÁN MÔN HỌC
DỰ ĐOÁN ĐIỂM TRUNG BÌNH HỌC KỲ SINH VIÊN
DỰA TRÊN DỮ LIỆU HỌC TẬP QUÁ KHỨ

**Leveraging Past Academic Data
for Future Performance Forecasting**

KHAI THÁC DỮ LIỆU VÀ ỨNG DỤNG – CS313.N22

GV hướng dẫn: ThS. Nguyễn Thị Anh Thư

Nhóm thực hiện:

1. Trương Thành Thắng – 20521907
2. Ngô Văn Tấn Lưu – 20521591
3. Lê Trương Ngọc Hải – 20520481
4. Ngô Ngọc Sương – 20521852
5. Trần Văn Lực – 20521587

TP. HỒ CHÍ MINH, 2023

MỤC LỤC

| | |
|--|----|
| Chương 1. TỔNG QUAN..... | 5 |
| 1.1. Giới thiệu..... | 5 |
| 1.2. Phát biểu bài toán | 5 |
| 1.3. Thách thức của bài toán | 6 |
| 1.4. Đối tượng và phạm vi | 6 |
| 1.5. Mục tiêu | 6 |
| 1.6. Bố cục báo cáo | 7 |
| Chương 2. CÁC NGHIÊN CỨU LIÊN QUAN..... | 8 |
| 2.1. Prediction of Student Performance Using Linear Regression [1] | 8 |
| 2.2. Predicting Academic Performance of Students Using a Hybrid Data Mining Approach [2] | 8 |
| 2.3. Predicting Students Academic Performance Using Education Data Mining [3] | 9 |
| Chương 3. BỘ DỮ LIỆU GIÁO DỤC ĐẠI HỌC..... | 10 |
| 3.1. Tiền xử lí dữ liệu..... | 10 |
| 3.1.1. Làm sạch file 01.sinhvien.xlsx..... | 10 |
| 3.1.1.1. Xử lí dữ liệu không hợp lệ | 10 |
| 3.1.1.2. Thống nhất thuộc tính noisinh..... | 10 |
| 3.1.1.3. Xử lí khoảng trống không mong muốn | 12 |
| 3.1.2. Sử dụng dữ liệu về điểm cộng khu vực để thay cho giá trị của thuộc tính noisinh..... | 14 |
| 3.1.3. Lưu trữ dữ liệu dưới dạng .json | 16 |
| 3.1.4. Chia tập huấn luyện và tập kiểm thử | 17 |

| | | |
|-----------|--|----|
| 3.1.5. | Bộ dữ liệu sau khi rút trích..... | 19 |
| 3.1.6. | Dữ liệu đầu vào của mô hình | 20 |
| Chương 4. | CÁC PHƯƠNG PHÁP ĐƯỢC DÙNG TRONG THỰC NGHIỆM | 21 |
| 4.1. | Linear Regression | 21 |
| 4.2. | Polynomial regression | 22 |
| 4.3. | Random Forest..... | 24 |
| 4.4. | XGBoost | 27 |
| 4.5. | Neural Networks | 28 |
| Chương 5. | THỰC NGHIỆM..... | 29 |
| 5.1. | Thang đo..... | 29 |
| 5.1.1. | Mean Square Error..... | 29 |
| 5.1.2. | Thời gian huấn luyện | 30 |
| 5.2. | Cài đặt chi tiết..... | 30 |
| 5.3. | Kết quả và phân tích..... | 31 |
| 5.3.1. | Bảng kết quả với bài toán dự đoán kết quả 1 kì tiếp theo..... | 31 |
| 5.3.2. | Bảng kết quả với bài toán dự đoán kết quả 2 kì tiếp theo..... | 33 |
| 5.4. | Thực nghiệm khác..... | 35 |
| 5.4.1. | Thực nghiệm với dữ liệu 2013 – 2019 không có thông tin sinh viên | 35 |
| 5.4.2. | Thực nghiệm với dữ liệu 2013 – 2022 không có thông tin sinh viên | 38 |
| Chương 6. | KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN | 40 |
| 6.1. | Kết luận..... | 40 |
| 6.2. | Hướng phát triển..... | 41 |

| | |
|---------------------------|----|
| TÀI LIỆU THAM KHẢO..... | 42 |
| PHÂN CÔNG CÔNG VIỆC | 43 |

Chương 1. TỔNG QUAN

1.1. Giới thiệu

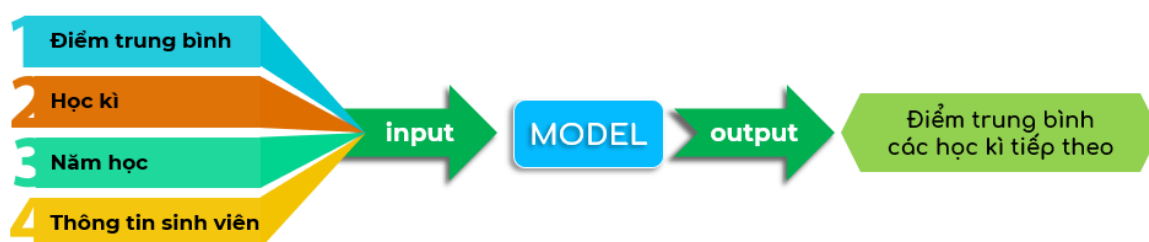
Trong môi trường giáo dục hiện nay, việc đánh giá năng lực và tiềm năng của sinh viên đóng vai trò quan trọng trong quá trình giáo dục và tuyển dụng.

- *Đối với nhà trường:* Việc dự đoán được kết quả học tập của sinh viên trong các kì tiếp theo có thể giúp nhà trường đánh giá hiệu quả của chương trình đào tạo, giúp nhà trường xác định được vấn đề tiềm ẩn nếu lượng lớn sinh viên có kết quả học tập đi xuống đột ngột, sớm đưa ra giải pháp cải thiện chất lượng giảng dạy.
- *Đối với doanh nghiệp:* Giúp nhà tuyển dụng có thêm thông tin để đánh giá năng lực và tiềm năng của sinh viên một cách chính xác hơn.
- *Đối với cố vấn học tập, sinh viên:* Sớm phát hiện và phân tích được vấn đề trong học kì tới ở một số sinh viên có kết quả dự đoán thấp để kịp thời chú ý và đưa ra kế hoạch học tập hợp lý.

Với nhận thức về vấn đề này, chúng tôi đã lựa chọn bài toán dự đoán điểm trung bình các học kỳ tiếp theo của sinh viên để tìm hiểu và thực hiện đề tài cuối kỳ. Mục tiêu của chúng tôi là khảo sát và phân tích để tìm ra các cách khai thác dữ liệu tốt hơn về độ hiệu quả và mô hình đạt được kết quả cao trên bài toán đã đặt ra. Để đạt được mục tiêu này, chúng tôi tiếp cận vấn đề bằng cách áp dụng các phương pháp tiền xử lí, phân tích dữ liệu và thực hiện các thí nghiệm toàn diện để đánh giá hiệu quả cách tiếp cận của chúng tôi.

1.2. Phát biểu bài toán

Đầu vào của bài toán là thông tin của sinh viên (ví dụ như nơi sinh, khoa, điểm trung bình, học kì, năm học, ...) ở các học kỳ trước, đầu ra của bài toán là điểm trung bình các học kỳ tiếp theo của sinh viên.



Hình 1-1 Mô hình bài toán.

1.3. Thách thức của bài toán

Bài toán “Dự đoán điểm trung bình học kỳ sinh viên dựa trên dữ liệu học tập quá khứ” đặt ra một số thách thức cần được giải quyết để đạt được kết quả có độ chính xác cao. Dưới đây là những thách thức của bài toán này:

- Điểm trung bình học kỳ của sinh viên có thể bị ảnh hưởng bởi sự thay đổi trong động lực, mục tiêu học tập và sự phát triển cá nhân của sinh viên theo thời gian.
- Môi trường học tập (giảng viên, bạn bè, ...) có thể ảnh hưởng đến hiệu suất học tập của sinh viên.
- Yêu cầu về tính khả thi (độ chính xác cao) và áp dụng thực tế (tốc độ thực thi nhanh).

1.4. Đối tượng và phạm vi

Đối tượng nghiên cứu: Sinh viên từ sau khóa 8 của trường Đại học Công nghệ Thông tin – Đại học Quốc gia TP.HCM.

Phạm vi thực hiện: Trường Đại học Công nghệ Thông tin – Đại học Quốc gia TP.HCM.

Thời gian thực hiện: Từ ngày 04/04/2023 đến ngày 28/05/2023.

1.5. Mục tiêu

Mục tiêu của đề tài là xây dựng một mô hình dự đoán điểm các học kỳ tiếp theo của sinh viên dựa trên dữ liệu lịch sử học tập của sinh viên với độ chính xác cao

để cung cấp phản hồi chính xác, trong khi vẫn đảm bảo tốc độ thực thi đủ nhanh để đáp ứng yêu cầu người dùng. Trong đề tài này, chúng tôi ưu tiên độ chính xác lên hàng đầu vì yêu cầu độ chính xác và tin cậy cao để đưa ra quyết định quan trọng liên quan đến học tập và nghề nghiệp của sinh viên.

Cụ thể, nghiên cứu sẽ tiến hành thu thập và phân tích dữ liệu học tập của các sinh viên trong khoảng thời gian giữa các kỳ học, bao gồm các thông tin như: điểm trung bình các học kỳ, tín chỉ của môn học, trình độ anh văn. Sau đó, sẽ sử dụng các kỹ thuật và phương pháp học máy để xây dựng một mô hình dự đoán điểm cho các học kỳ tiếp theo dựa trên dữ liệu lịch sử học tập của sinh viên.

Kết quả của đề tài sẽ là một mô hình dự đoán điểm số của sinh viên, có độ chính xác cao và tốc độ thực thi nhanh. Có thể áp dụng được cho các sinh viên khác nhau trong các trường học khác nhau. Ngoài ra, kết quả của nghiên cứu sẽ giúp các giáo viên và nhà quản lý giáo dục có thể đưa ra các giải pháp hỗ trợ phù hợp để giúp sinh viên đạt kết quả học tập tốt hơn.

1.6. Bố cục báo cáo

Báo cáo của chúng tôi bao gồm 6 chương:

Chương 1: Giới thiệu tổng quan về đề tài thực hiện.

Chương 2: Tìm hiểu các nghiên cứu liên quan.

Chương 3. Trình bày nội dung Tiền xử lí dữ liệu

Chương 4: Trình bày các phương pháp được dùng trong thực nghiệm.

Chương 5. Cài đặt thực nghiệm và đánh giá.

Chương 6. Tổng kết các nội dung đã thực hiện được và định hướng phát triển bài toán.

Cuối cùng, báo cáo kết thúc với phần tài liệu tham khảo và bảng phân công công việc.

Chương 2. CÁC NGHIÊN CỨU LIÊN QUAN

2.1. Prediction of Student Performance Using Linear Regression [1]

Boddeti Sravani và các đồng sự trình bày về việc ứng dụng Học máy có tác động to lớn như thế nào trong việc dạy và học nhằm cải thiện hơn nữa môi trường học tập ở giáo dục đại học. Do sự quan tâm của sinh viên đối với các khóa học trực tuyến và kỹ thuật số đã tăng lên nhanh chóng, các trang web như Course Era, Udemy, v.v. đã trở nên rất có ảnh hưởng. Họ triển khai các ứng dụng mới của máy học trong dạy và học dựa trên nền tảng của học sinh, điểm số học tập trong quá khứ của học sinh và xem xét các thuộc tính khác. Khi quy mô lớp học lớn, sẽ khó hỗ trợ từng học viên trong mỗi khóa học mở, điều này có thể làm tăng tỷ lệ bỏ học khi kết thúc khóa học. Trong bài báo này, các tác giả sử dụng mô hình hồi quy tuyến tính, một thuật toán học máy, để dự đoán kết quả học tập của học sinh. Dữ liệu được sử dụng trong bài này bao gồm 100 sinh viên. Trong nghiên cứu này các tác giả giải quyết với 100 trường hợp và 10 thuộc tính: giới tính, tuổi, học thức của bố mẹ, chuẩn bị kì thi, loại gia đình, nghề nghiệp của bố/mẹ, ngày nghỉ, tình trạng gia đình, điểm thi, thời gian du lịch., Các tác giả sử dụng mô hình hồi quy tuyến tính để dự đoán điểm thi với độ chính xác là 95% trên tập test.

2.2. Predicting Academic Performance of Students Using a Hybrid Data Mining Approach [2]

Phân tích và dự đoán kết quả học tập của học sinh là một yếu tố cần thiết trong môi trường giáo dục. Các tác giả cho rằng giáo dục khai thác dữ liệu có thể được thực hiện bằng cách sử dụng một số kỹ thuật khai thác dữ liệu cụ thể là neural network, decision tree, support vector, Naïve bayes và K-nearest neighbor. Tuy nhiên, tất cả các thuật toán này thuộc một trong hai phương pháp tiếp cận chính cụ thể là phân loại và phân cụm. Trong khi có các thuật toán khác nhau

giúp dự đoán chính xác kết quả học tập của sinh viên sử dụng phương pháp phân loại, một số thuật toán khác hoàn thành mục tiêu tương tự bằng cách phân cụm. Do đó, các tác giả thực hiện nghiên cứu này và đề xuất một cách tiếp cận lai kết hợp cả thuật toán phân cụm và thuật toán phân loại để đạt được kết quả tốt hơn độ chính xác dự đoán khi so sánh với độ chính xác của các thuật toán hiện có. Cụ thể, các tác giả sử dụng bốn phương pháp SVM, Naive Bayes, Decision Tree, Neural Network để dự đoán các đặc trưng cần sử dụng. Sau đó, dữ liệu với các đặc trưng được chọn được đưa vào mô hình K-mean để dự đoán kết quả học tập cuối cùng. Dữ liệu được thu thập từ nhiều nguồn và chứa các đặc trưng ban đầu như đặc trưng học thuật, đặc trưng về hành vi ứng xử,... Kết quả cuối cùng đạt khoảng 0.7547 accuracy trên tập test.

2.3. Predicting Students Academic Performance Using Education Data Mining [3]

Thời gian qua đã chứng kiến sự phát triển nhanh chóng trong việc sử dụng Data Mining như một công cụ mà các tổ chức học thuật sử dụng để trích xuất thông tin hữu ích trong bộ dữ liệu kết quả của sinh viên nhằm cải thiện quá trình học tập. Mục tiêu chính của bài nghiên cứu này là dự đoán kết quả đại học của sinh viên trong trên cơ sở thành tích của họ trong bài kiểm tra, bài tập, tỷ lệ tốt nghiệp và điểm chuyên cần. Các tác giả đã thực hiện nghiên cứu dựa trên các thuộc tính khác nhau. Trong nghiên cứu của họ, các quy tắc quan trọng được tạo ra để đo lường mối tương quan giữa các thuộc tính khác nhau sẽ giúp cải thiện kết quả học tập của học sinh. Thử nghiệm được thực hiện bằng cách sử dụng Weka Association Rule Mining và bộ dữ liệu thời gian thực có sẵn trong trường đại học. Cụ thể, bộ dữ liệu chứa các yếu tố của học sinh như tuổi, giới tính, loại trường trung học,...

Chương 3. BỘ DỮ LIỆU GIÁO DỤC ĐẠI HỌC

3.1. Tiền xử lí dữ liệu

3.1.1. Làm sạch file 01.sinhvien.xlsx

3.1.1.1. Xử lí dữ liệu không hợp lệ

Trong bài toán dự đoán “DỰ ĐOÁN ĐIỂM TRUNG BÌNH HỌC KỲ SINH VIÊN”, chúng tôi sử dụng dữ liệu từ hai nguồn dữ liệu chính, đó là **01.sinhvien.xlsx** và ***sinhvien_dtb_hocky.xlsx***. Trong đó, dữ liệu của ***sinhvien_dtb_hocky.xlsx*** đã hoàn toàn hợp lệ, do đó nguồn dữ liệu cần làm sạch hiện tại là **01.sinhvien.xlsx**.

Tuy nhiên, trong file này có đến 69 thuộc tính, trong đó có đến 56 cột dư thừa (là những cột không đặt tên) và 7 thuộc tính không có đóng góp đáng kể vào bài toán. Sau khi chắt lọc thì nhóm quyết định sử dụng 5 thuộc tính sau đây:

- ***mssv***: Mã số sinh viên, đây là khóa chính để kết hợp hai nguồn dữ liệu lại với nhau và phục vụ cho mục đích thống kê.
- ***gioitinh***: Giới tính (Nam: 1, Nữ: 0).
- ***noisinh***: Nơi sinh của sinh viên (giá trị chuỗi của tên tỉnh/thành phố).
- ***khhoa***: Khoa mà sinh viên đang theo học tại trường (giá trị chuỗi của tên khoa viết tắt. VD: CNTT – Công nghệ thông tin).
- ***hedt***: Hệ đào tạo mà sinh viên đang theo học (giá trị chuỗi của tên hệ đào tạo viết tắt. VD: CLC – Chất lượng cao).

3.1.1.2. Thống nhất thuộc tính noisinh

Vấn đề:

Qua quá trình thống kê, thuộc tính nơi sinh chứa các giá trị không rõ ràng và không thống nhất ở các trường hợp sau:

- ***Giá trị ở mức quận, xã thay vì tỉnh/thành phố***. VD: quận Thủ Đức,...
- ***Giá trị mang tên cũ của tỉnh thành trước khi tách/sáp nhập***. VD: Hải Hưng, Long Hồ,...

- *Giá trị là tên của các quốc gia khác.* VD: Campuchia, Australia,...
- *Giá trị là một chuỗi dài bao gồm cả tên xã, huyện.* VD: Nghĩa Sơn - Nghĩa Hưng - Nam Định.
- *Cùng một tên tỉnh/thành phố nhưng lại biểu diễn khác nhau.* VD: TP Hồ Chí Minh, Thành phố Hồ Chí Minh, Hồ Chí Minh...

Giải pháp:

Sau khi giải quyết các trường hợp không thống nhất dữ liệu bằng việc thay thế các giá trị. Ví dụ, thay giá trị tên xã, quận thành tên tỉnh/thành phố chứa nó (quận Thủ Đức -> Hồ Chí Minh).

Tên các quốc gia khác vẫn được giữ lại trong bộ dữ liệu.

Tuy nhiên để thống nhất các giá trị biểu diễn cho cùng một thành phố thì chúng tôi sẽ sử dụng khoảng cách Levenshtein. Khoảng cách Levenshtein giữa chuỗi S1 và chuỗi S2 là số bước ít nhất biến chuỗi S1 thành chuỗi S2 thông qua 3 phép biến đổi là: xoá 1 ký tự, thêm 1 ký tự, thay ký tự này bằng ký tự khác.

```
from Levenshtein import distance

# noisinh
noisinh = df_sv['noisinh']
# nltk.edit_distance(s1, s2)
tinh_thanh = ["Hà Nội", "Hồ Chí Minh", "Hải Phòng", "Đà Nẵng", "Cần Thơ", "An Giang", "Bà Rịa - Vũng Tàu", "Bắc Giang", "Bắc Kạn",
              "Bạc Liêu", "Bắc Ninh", "Bến Tre", "Bình Định", "Bình Dương", "Bình Phước", "Bình Thuận", "Cà Mau", "Cao Bằng", "Đắk Lắk",
              "Đắk Nông", "Điện Biên", "Đồng Nai", "Đồng Tháp", "Gia Lai", "Hà Giang", "Hà Nam", "Hà Tĩnh", "Hải Dương", "Hậu Giang", "Hòa",
              "Hưng Yên", "Khánh Hòa", "Kiên Giang", "Kon Tum", "Lai Châu", "Lâm Đồng", "Lạng Sơn", "Lào Cai", "Long An", "Nam Định", "Ngh",
              "Ninh Bình", "Ninh Thuận", "Phú Thọ", "Quảng Bình", "Quảng Nam", "Quảng Ngãi", "Quảng Ninh", "Quảng Trị", "Sóc Trăng", "Sơn",
              "Thái Bình", "Thái Nguyên", "Thanh Hóa", "Thừa Thiên Huế", "Tiền Giang", "Trà Vinh", "Tuyên Quang", "Vĩnh Long", "Vĩnh Phúc",
              "Cộng hoà Séc", "Campuchia", "Australia", "Liên Bang Nga"]

df_raw
noisinh_new = []
for row in df_sv['noisinh']:
    distances = np.array([distance(row, s) for s in tinh_thanh])
    min_index = np.argmin(distances)
    closest_string = tinh_thanh[min_index]
    noisinh_new.append(closest_string)
noisinh_arr = np.array(noisinh_new)
noisinh_arr.reshape(-1,1)

array([[ 'Hồ Chí Minh'],
       [ 'Đồng Tháp'],
       [ 'Hà Nam'],
       ...,
       [ 'Lâm Đồng'],
       [ 'Bến Tre'],
       [ 'Hồ Chí Minh']], dtype='<U17')
```

Hình 3-1 Demo thuật toán tính khoảng cách Levenshtein

Ở hình 3-1, chúng tôi sử dụng hàm **distance** thuộc thư viện **Levenshtein** để tính khoảng cách giữa các giá trị trong thuộc tính **noisinh** và danh sách 67 tên địa điểm (63 tỉnh thành và 4 tên quốc gia khác). Output của bài toán sẽ trả về mảng tương ứng sau khi thống nhất tên tỉnh thành, quốc gia trong bộ dữ liệu.

3.1.1.3. Xử lý khoảng trắng không mong muốn

Ở hai thuộc tính còn lại là **khoa** và **hedt**, dữ liệu đều có khoảng trắng ở phía trước. VD: ' CQUI', ' CLC', ' CNTT', ' HTTT',...

Do đó, chúng tôi phải xóa khoảng trắng dư thừa đó. Cách xóa được minh họa ở Hình 3-2.

```
[ ] df_sv['khoa'] = df_sv['khoa'].replace(" ",'', regex=True)
    df_sv['hedt'] = df_sv['hedt'].replace(" ",'', regex=True)
```

*Hình 3-2 Xóa khoảng trắng trong các giá trị của hai thuộc tính **khoa** và **hedt**.*

Và cuối cùng, không quên lưu dữ liệu lại và kiểm tra một lần nữa.

```
[ ] # lưu dữ liệu đã xử lý
    df_sv.to_csv('./data_raw/01.sinhvien.csv', index=False)
```

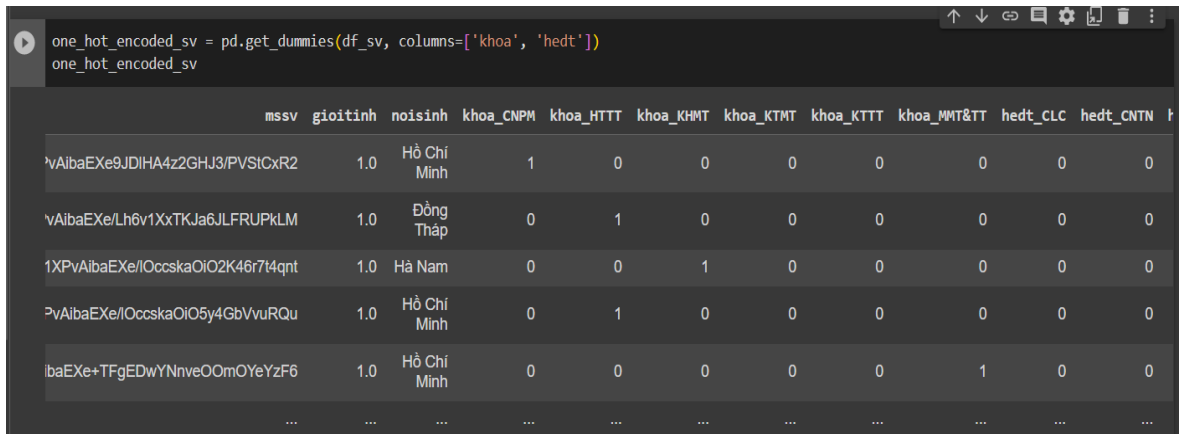
```
[6] df_sv = pd.read_csv('./data_raw/01.sinhvien.csv')
    df_sv
```

| | mssv | gioitinh | noisinh | khoa | hedt |
|-----|--|----------|-------------|--------|------|
| 0 | BE375BAAXPvAibaEXe9JDIHA4z2GHJ3/PVStCxR2 | 1.0 | Hồ Chí Minh | CNPM | CQUI |
| 1 | 2420ED57XPvAibaEXe/Lh6v1XxTKJa6JLFRUPKLM | 1.0 | Đồng Tháp | HTTT | CTTT |
| 2 | 83B76C01XPvAibaEXe/IOccskaOiO2K46r7t4qnt | 1.0 | Hà Nam | KHMT | CQUI |
| 3 | 91F785ABXPvAibaEXe/IOccskaOiO5y4GbVvuRQu | 1.0 | Hồ Chí Minh | HTTT | CTTT |
| 4 | 007C275DXPvAibaEXe+TFgEDwYNnveOOmOYeYzF6 | 1.0 | Hồ Chí Minh | MMT&TT | CQUI |
| ... | ... | ... | ... | ... | ... |

Hình 3-3 Dữ liệu sau khi làm sạch sơ bộ.

Quan sát thấy ở Hình 3-3, chúng tôi không thể huấn luyện mô hình nếu cứ để giá trị ở dạng chuỗi như trên được. Do đó, chúng tôi sẽ phải mã hóa các giá trị

này sang giá trị số bằng OneHotVector. Có thể đơn giản hóa bằng việc sử dụng hàm `get_dummies` của thư viện `pandas` đã hỗ trợ sẵn.



```
one_hot_encoded_sv = pd.get_dummies(df_sv, columns=['khoa', 'hedt'])
one_hot_encoded_sv
```

| | mssv | gioitinh | noisinh | khoa_CNPM | khoa_HTTT | khoa_KHMT | khoa_KTMT | khoa_KTTT | khoa_MMT&TT | hedt_CLC | hedt_CNTN |
|---|-----------------------------------|----------|-------------|-----------|-----------|-----------|-----------|-----------|-------------|----------|-----------|
| 0 | vAibaEXe9JDIHA4z2GHJ3/PVStCxR2 | 1.0 | Hồ Chí Minh | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | vAibaEXe/Lh6v1XxTKJa6JLFRUPkLM | 1.0 | Đồng Tháp | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1XPvAibaEXe/IOccskaOiO2K46r7t4qnt | 1.0 | Hà Nam | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 3 | PvAibaEXe/IOccskaOiO5y4GbVvuRQu | 1.0 | Hồ Chí Minh | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | baEXe+TFgEDwYNnveOOmOYeYzF6 | 1.0 | Hồ Chí Minh | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

Hình 3-4 Dữ liệu sau khi mã hóa

Nhưng chúng tôi phải giữ lại vector của từng giá trị của từng thuộc tính để sau này mã hóa dữ liệu đầu vào.

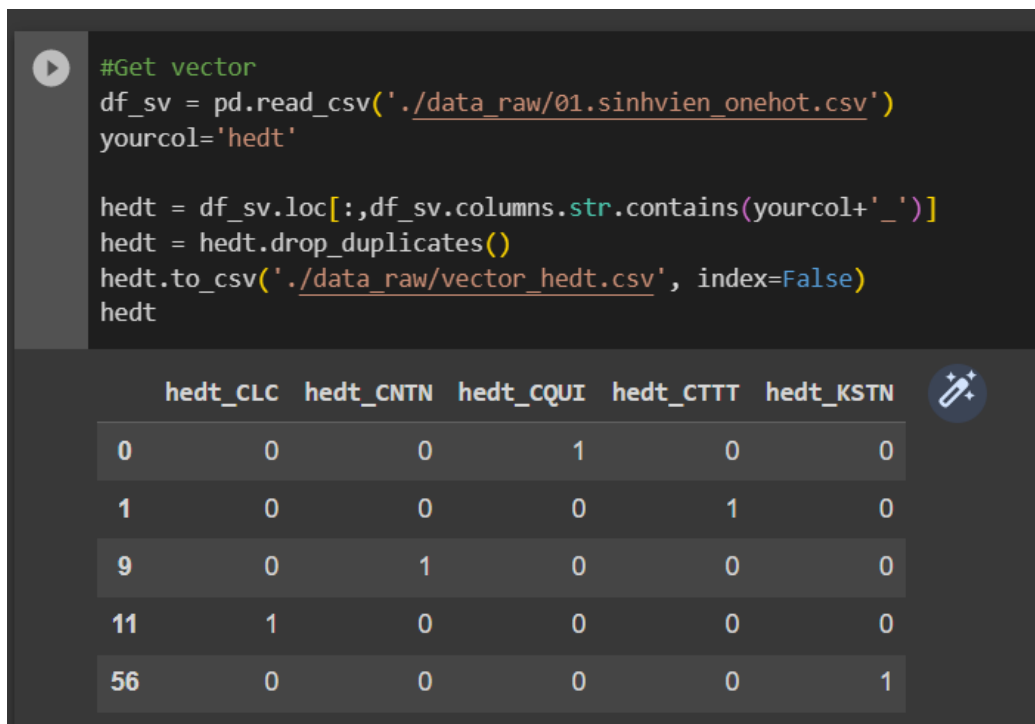


```
#Get vector
df_sv = pd.read_csv('./data_raw/01.sinhvien_onehot.csv')
yourcol='khoa'

khoa = df_sv.loc[:,df_sv.columns.str.contains(yourcol+'_')]
khoa = khoa.drop_duplicates()
khoa.to_csv('./data_raw/vector_khoa.csv', index=False)
khoa
```

| | khoa_CNPM | khoa_HTTT | khoa_KHMT | khoa_KTMT | khoa_KTTT | khoa_MMT&TT |
|----|-----------|-----------|-----------|-----------|-----------|-------------|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 1 |
| 11 | 0 | 0 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 1 | 0 |

Hình 3-5 Danh sách vector của các giá trị trong thuộc tính khoa



```
#Get vector
df_sv = pd.read_csv('./data_raw/01.sinhvien_onehot.csv')
yourcol='hedt'

hedt = df_sv.loc[:,df_sv.columns.str.contains(yourcol+'_')]
hedt = hedt.drop_duplicates()
hedt.to_csv('./data_raw/vector_hedt.csv', index=False)
hedt
```

| | hedt_CLC | hedt_CNTN | hedt_CQUI | hedt_CTTT | hedt_KSTN |
|----|----------|-----------|-----------|-----------|-----------|
| 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 9 | 0 | 1 | 0 | 0 | 0 |
| 11 | 1 | 0 | 0 | 0 | 0 |
| 56 | 0 | 0 | 0 | 0 | 1 |

Hình 3-6 . Danh sách vector của các giá trị trong thuộc tính hedt

3.1.2. Sử dụng dữ liệu về điểm cộng khu vực để thay cho giá trị của thuộc tính noisinh

Chúng tôi không thể mã hóa OneHotVector cho thuộc tính noisinh được vì miền giá trị của nó quá lớn, cụ thể là có đến 63 tỉnh thành cộng thêm 4 quốc gia khác. Nếu vẫn cố chấp sử dụng OneHotVector thì sẽ làm cho dữ liệu huấn luyện xuất hiện thêm rất nhiều cột, lên đến 67 cột, làm cho dữ liệu không khách quan.

Do đó, chúng tôi phải tìm cách khác để biến những giá trị chuỗi này thành số mà không phải tạo thêm hay làm phức tạp thêm nguồn dữ liệu hiện có. Vì vậy, chúng tôi đã tiến hành thu thập các thông tin về điểm cộng khu vực có chứa dữ liệu điểm cộng tương ứng với mỗi tỉnh thành trong kì thi đại học.

Tuy nhiên, có hai vấn đề chính:

- Một tỉnh thành có thể có nhiều khu vực có quy chế cộng điểm khác nhau.
- Điểm cộng khu vực được quy định rõ ở cấp độ xã, quận của từng nơi.

Điều này dẫn đến việc hai sinh viên trong cùng một thành phố, nhưng lại có chế độ cộng điểm khác nhau.

GIẢI QUYẾT VẤN ĐỀ: Chúng tôi tiến hành thống kê và tính điểm cộng khu vực trung bình của từng tỉnh/thành phố (trừ các quốc gia khác sẽ có điểm cộng là 0) để đảm bảo tính công bằng và khách quan của dữ liệu. Sơ bộ về dữ liệu được minh họa ở Hình 3-7.

| 1 | noisinh | diemcong |
|----|----------------|-----------------|
| 2 | Campuchia | 0 |
| 3 | Australia | 0 |
| 4 | Cộng hoà Séc | 0 |
| 5 | Liên Bang Nga | 0 |
| 6 | Hà Nội | 0.256975698 |
| 7 | Hồ Chí Minh | 0.092592593 |
| 8 | Hải Phòng | 0.522727273 |
| 9 | Đà Nẵng | 0.111111111 |
| 10 | Hà Giang | 0.75 |
| 11 | Cao Bằng | 0.75 |
| 12 | Lai Châu | 0.75 |
| 13 | Lào Cai | 0.75 |
| 14 | Tuyên Quang | 0.75 |
| 15 | Lạng Sơn | 0.75 |

Hình 3-7 Một số điểm dữ liệu của nguồn dữ liệu điểm cộng khu vực.

Sau cùng, chúng tôi tiến hành tích hợp dữ liệu này vào file chứa dữ liệu sinh viên đã mã hóa từ trước.

```
df = pd.merge(df_sv, df_diemcong, on='noisinh', how='left')
df['noisinh'] = df['diemcong']
df = df.drop('diemcong', axis=1)
df = df.rename(columns={'noisinh': 'diemcong'})
df
```

| | mssv | gioitinh | diemcong | khoa_CNPM | khoa_HTTT | khoa_KHMT | khoa_KTMT | khoa_KTTT | khoa |
|------|--|----------|----------|-----------|-----------|-----------|-----------|-----------|------|
| 0 | BE375BAAXPvAibaEXe9JDIHA4z2GHJ3/PVStCxR2 | 1.0 | 0.092593 | 1 | 0 | 0 | 0 | 0 | |
| 1 | 2420ED57XPvAibaEXe/Lh6v1XxTKJa6JLFRUPkLM | 1.0 | 0.613636 | 0 | 1 | 0 | 0 | 0 | |
| 2 | 83B76C01XPvAibaEXe/OccskaOiO2K46r7t4qnt | 1.0 | 0.437500 | 0 | 0 | 1 | 0 | 0 | |
| 3 | 91F785ABXPvAibaEXe/OccskaOiO5y4GbVvuRQu | 1.0 | 0.092593 | 0 | 1 | 0 | 0 | 0 | |
| 4 | 007C275DXPvAibaEXe+TFgEDwYNnveOOmOYeYzF6 | 1.0 | 0.092593 | 0 | 0 | 0 | 0 | 0 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 8294 | 7D7633B0XPvAibaEXe95lhrkYSGfcCQVGm4nFGyt | 1.0 | 0.670000 | 0 | 0 | 0 | 0 | 0 | |
| 8295 | AB431338XPvAibaEXe8xMCTZ03/BexWzDCyWVsT3 | 1.0 | 0.616557 | 0 | 0 | 1 | 0 | 0 | |
| 8296 | 75AD7B4AXPvAibaEXe8xMCTZ03/BewoGrWst0ZXm | 1.0 | 0.750000 | 0 | 0 | 1 | 0 | 0 | |
| 8297 | CB263C18XPvAibaEXe8xMCTZ03/Be8yk40QWdPIR | 1.0 | 0.656977 | 0 | 0 | 0 | 0 | 1 | |
| 8298 | 7CD9B404XPvAibaEXe8M4ItS8XJC9c/g97NvpP52 | 1.0 | 0.092593 | 0 | 1 | 0 | 0 | 0 | |

Hình 3-8 Dữ liệu hoàn chỉnh sau khi thay thuộc tính **noisinh** bằng thuộc tính **diemcong**.

3.1.3. Lưu trữ dữ liệu dưới dạng .json

Dữ liệu được tổ chức theo kiểu dữ liệu là dict vào file json có:

- **key** là mã số sinh viên
- **value** được biểu diễn dưới dạng mảng hai chiều như sau:
 - Dòng đầu tiên chứa thông tin sinh viên bao gồm: gioitinh, diemcong, khoa_CNPM, khoa_HTTT, khoa_KHMT, khoa_KTMT, khoa_KTTT, khoa_MMT&TT, hedt_CLC, hedt_CNTN, hedt_CQUI, hedt_CTTT, hedt_KSTN.
 - Những dòng tiếp theo là kết quả học tập của sinh viên theo từng kì. Mỗi dòng tương ứng với một kì được chuẩn hóa bằng minmax_scaler.

```
[ ] def minmax_scaler(value, min, max):
    | | return (value - min)/(max - min)
```

Hình 3-9 Định nghĩa phương thức minmax_scaler

Ví dụ với sinh viên sau:

```
{"BBAD4F2BXPvAIBAEXE879+AOG1GH8vGAQYZ3wJAR":  
  [[1.0, 0.092592593, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0],  
  [0.745, 0.6875, 0.0],  
  [0.786, 0.5625, 0.047619047619047616],  
  [0.8619999999999999, 0.78125, 0.09523809523809523],  
  [0.85, 0.46875, 0.14285714285714285],  
  [0.868, 0.625, 0.19047619047619047],  
  [0.8480000000000001, 0.5, 0.23809523809523808],  
  [0.907, 0.46875, 0.2857142857142857],  
  [0.9099999999999999, 0.3125, 0.3333333333333333]], ...}
```

3.1.4. Chia tập huấn luyện và tập kiểm thử

Nhóm sẽ sử dụng dữ liệu từ các sinh viên khóa 2018, 2019 làm dữ liệu test. Điều này có những lí do sau:

- Dữ liệu điểm sau khi merge với file 01.sinhvien.xlsx thì các sinh viên khóa 2020, 2021 tuy có điểm học tập nhưng lại không có thông tin, lý lịch.
- Bài toán tập trung vào việc dự đoán điểm của các sinh viên đến sau, do đó buộc phải sử dụng các sinh viên ở khóa trước đó để làm dữ liệu huấn luyện.

▼ Chọn ra danh sách mssv cho tập test

```
[15] sv_namhoc = pd.DataFrame(df_raw.groupby(['mssv']).min().namhoc)
      mssv_test = sv_namhoc.loc[sv_namhoc['namhoc'] >= 2018].drop(['na
      mssv_test = mssv_test.loc[mssv_test['namhoc'] <= 2019]
      mssv_test.to_csv('./data/mssv_test.csv')
      mssv_test.head()
```

| mssv | namhoc |
|--|--------|
| 0001EB57XPvAibaEXe/twT+sf632fUXnsgPGeB4G | 2019 |
| 000AD0D8XPvAibaEXe+RQyZpP6sq6qqIPZXybx3Q | 2018 |
| 0013D6E5XPvAibaEXe85hkLQQAjy3XgK9pA18A31 | 2018 |
| 0018C59CXPvAibaEXe8C3lhlh2dNniH+SYgLosUA | 2019 |
| 00256CA6XPvAibaEXe9ZBnO1fewQMfc0mu/9tLFg | 2019 |

Hình 3-10 Chọn danh sách mssv có khóa bắt đầu từ 2018 và 2019 làm tập test

▼ Chọn ra danh sách mssv cho tập train

```
sv_namhoc = pd.DataFrame(df_raw.groupby(['mssv']).min().namhoc)
mssv_train = sv_namhoc.loc[sv_namhoc['namhoc'] < 2018].drop(['namh
mssv_train.to_csv('./data/mssv_train.csv')
mssv_train.head()
```

| mssv | namhoc |
|--|--------|
| 00046394XPvAibaEXe+fmxcqgvribEcT4YmJhSFD | 2013 |
| 001BB05EXPvAibaEXe/YKAIYnC3m92BzM+VhZyc0 | 2013 |
| 001E045BXPvAibaEXe+n07P56kWx2N6EoOCUJBA4 | 2017 |
| 004E12B9XPvAibaEXe81r7KrCH0uc0uxBtoBw4BG | 2017 |
| 005116E7XPvAibaEXe8VdV75E6kcrk07Zn1wf/T8 | 2014 |

Hình 3-11 Những sinh viên còn lại sẽ thuộc tập train. Tuy nhiên buộc phải lược bỏ sinh viên có khóa từ 2020 và 2021 vì thiếu dữ liệu lý lịch.

Sau khi chia dữ liệu theo khóa, chúng tôi thu được kích thước của tập train và tập test ở Hình 3-12.

```
▼ Kiểm tra kích thước của 2 tập

0s mssv_test = pd.read_csv('./data/mssv_test.csv')
    test = mssv_test.iloc[:, 0].values

    mssv_train = pd.read_csv('./data/mssv_train.csv')
    train = mssv_train.iloc[:, 0].values

    print('train:', len(train), 'sinh viên')
    print('test:', len(test), 'sinh viên')
    print('tập train lớn hơn tập test', len(train)/len(test), 'lần')

train: 5108 sinh viên
test: 3121 sinh viên
tập train lớn hơn tập test 1.6366549182954182 lần
```

Hình 3-12 Kích thước và tỉ lệ giữa hai tập train và test

3.1.5. Bộ dữ liệu sau khi rút trích

Bộ dữ liệu giáo dục được cung cấp gồm các sinh viên nhập học từ năm 2013 đến năm 2022. Tuy nhiên, dữ liệu của các sinh viên nhập học năm 2020, 2021 và 2022 chỉ có điểm học tập mà không có thông tin và lý lịch cá nhân. Vì vậy, chúng tôi quyết định sử dụng bộ dữ liệu chỉ gồm những sinh viên nhập học từ năm 2013 đến năm 2019.

Trong bộ dữ liệu giáo dục mà chúng tôi đã rút trích gồm có : 8229 sinh viên.

Trong đó:

- Sinh viên nhập học năm 2013 có 941 sinh viên.
- Sinh viên nhập học năm 2014 có 1009 sinh viên.
- Sinh viên nhập học năm 2015 có 1030 sinh viên.
- Sinh viên nhập học năm 2016 có 1039 sinh viên.
- Sinh viên nhập học năm 2017 có 1089 sinh viên.
- Sinh viên nhập học năm 2018 có 1410 sinh viên.

- Sinh viên nhập học năm 2019 có 1711 sinh viên.

Với bộ dữ liệu giáo dục đã rút trích, chúng tôi sử dụng sinh viên từ nhập học từ năm 2013 đến năm 2017 là tập dữ liệu train và sinh viên nhập học năm 2018, năm 2019 làm tập dữ liệu test.

Trong đó:

- Tập train có 5108 sinh viên.
- Tập test có 3121 sinh viên.

3.1.6. Dữ liệu đầu vào của mô hình

Dữ liệu được tổ chức theo kiểu dữ liệu là mảng một chiều.

Trong đó:

- Phần tử thứ 0 là gioitinh (1.0, 0.0)
- Phần tử thứ 1 là diemcong.
- Phần tử thứ 2 đến 7 là khoa (khoa_CNPM, khoa_HTTT, khoa_KHMT, khoa_KTMT, khoa_KTTT, khoa_MMT&TT)
- Phần tử thứ 8 đến 12 là hedt (hedt_CLC, hedt_CNTN, hedt_CQUI, hedt_CTTT, hedt_KSTN)
- 3 phần tử tiếp theo là kết quả học tập của sinh viên trong học kì thứ 1. 3 phần tử đó lần lượt là sotchk, dtbkh, hocky. Tương tự, 3 phần tử tiếp theo là kết quả học tập của sinh viên trong học kỳ thứ 2. Cứ tiếp tục như vậy cho các phần tử còn lại.

Ví dụ với sinh viên sau input = 1:

{"BBAD4F2BXPVAiBAEXE879+AOG1GH8VGAQYZ3WJAR":

[1.0, 0.092592593, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0.745, 0.6875, 0.0]

Trong đó:

- 1.0: gioitinh (Nam)
- 0.092592593: diemcong
- 0, 0, 1, 0, 0, 0: khoa (KHMT)
- 0, 1, 0, 0, 0: hedt (CNTN)

- 0.745: sothk
- 0.06875: dtbkh
- 0.0: hocky

Chương 4. CÁC PHƯƠNG PHÁP ĐƯỢC DÙNG TRONG THỰC NGHIỆM

4.1. Linear Regression

Linear Regression [4] - 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 tôi muốn dự đoán một số lượng liên tục. Ví dụ, dự đoán giao thông ở một cửa hàng bán lẻ, dự đoán thời gian người dùng dừng lại một trang nào đó hoặc số trang đã truy cập vào một website nào đó v.v... do đó thuật toán Linear Regression được dùng trong thực nghiệm cho bài toán này để dự đoán điểm trung bình học kỳ sinh viên dựa trên dữ liệu học tập quá khứ.

Linear Regression là một trong những phương pháp đơn giản và phổ biến nhất trong machine learning và thường được sử dụng để dự đoán giá trị của biến phụ thuộc dựa trên các biến độc lập.

Ưu điểm:

- *Đơn giản và dễ hiểu:* Linear regression là một thuật toán đơn giản và dễ hiểu. Công thức toán học của nó rất đơn giản và có thể giải thích một cách trực quan.
- *Tính linh hoạt:* Linear regression có thể được áp dụng cho cả các bài toán dự đoán đơn biến và đa biến. Nó cũng có thể mở rộng để xử lý các trường hợp phức tạp hơn thông qua việc sử dụng các biến độc lập tương tác và biến độc lập bậc cao (polynomial regression).

- *Tính tường minh và giải thích được:* Kết quả của linear regression có thể được giải thích một cách rõ ràng. Các hệ số hồi quy có thể biểu thị độ ảnh hưởng của từng biến độc lập đến biến phụ thuộc.
- *Tốc độ huấn luyện nhanh:* Với một tập dữ liệu nhỏ hoặc có số lượng biến độc lập ít, quá trình huấn luyện linear regression có thể diễn ra nhanh chóng.

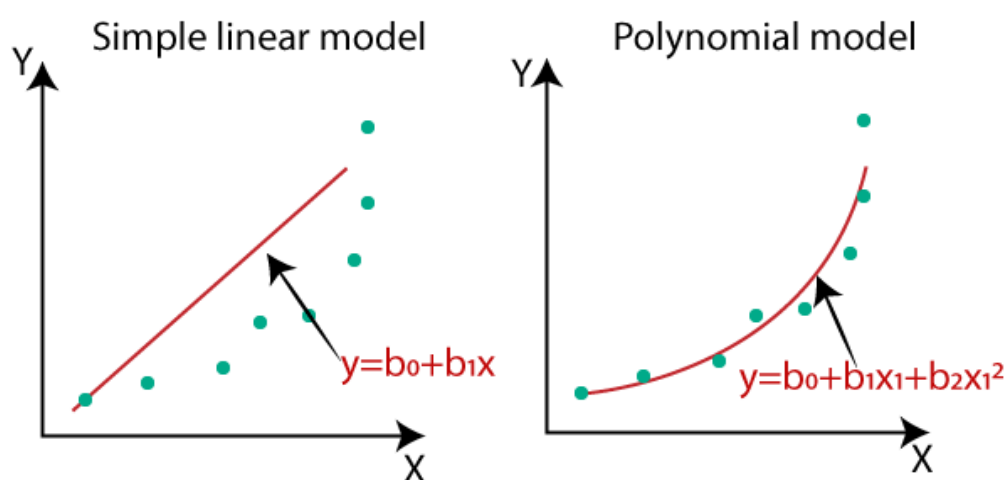
Nhược điểm:

- *Giả định về mối quan hệ tuyến tính:* Linear regression giả định rằng mối quan hệ giữa biến phụ thuộc và biến độc lập là tuyến tính. Điều này có nghĩa là nếu mối quan hệ thực tế không phải là tuyến tính, linear regression có thể cho ra kết quả không chính xác hoặc không tốt.
- *Nhạy cảm với nhiễu:* Linear regression có thể bị ảnh hưởng bởi các điểm dữ liệu nhiễu. Các điểm dữ liệu nhiễu có thể ảnh hưởng đáng kể đến đường thẳng (hoặc siêu phẳng) tìm được, làm cho mô hình dự đoán kém chính xác.
- *Không thể mô hình hóa mối quan hệ phi tuyến:* Linear regression không thể mô hình hóa mối quan hệ phi tuyến giữa biến phụ thuộc và biến độc lập một cách chính xác. Để xử lý mối quan hệ phi tuyến, cần phải sử dụng các phương pháp khác như polynomial regression hoặc các phương pháp hồi quy phi tuyến khác.
- *Dễ bị ảnh hưởng bởi đa cộng tuyến:* Khi có sự tương quan cao giữa các biến độc lập, linear regression có thể bị ảnh hưởng bởi hiện tượng đa cộng tuyến, gây khó khăn trong việc xác định độ ảnh hưởng riêng lẻ của từng biến độc lập.

4.2. Polynomial regression

Polynomial regression Là một dạng hồi quy tuyến tính (linear regression) trong đó mối quan hệ giữa biến độc lập (independent variable) x và biến phụ thuộc (dependent variable) y được mô hình hóa dưới dạng đa thức

(polynomial) bậc n . Hồi quy đa thức (polynomial regression) phù hợp với mối quan hệ phi tuyến tính (nonlinear). Mô hình hồi quy đa thức (polynomial regression) thường fit với phương pháp bình phương nhỏ nhất (least squares).



Hình 4-1 Ví dụ một phương trình của hồi quy đa thức.

Ưu điểm:

- **Đa dạng hóa mô hình:** Polynomial regression cho phép mô hình hóa các mối quan hệ phi tuyến, không chỉ giới hạn trong mô hình tuyến tính. Điều này rất hữu ích khi dữ liệu có mối quan hệ phức tạp và không tuyến tính.
- **Độ linh hoạt:** Phương pháp này cho phép chúng tôi điều chỉnh mức độ của đa thức, ví dụ như bậc của đa thức, để tìm mô hình phù hợp với dữ liệu. Điều này cho phép tăng hoặc giảm độ phức tạp của mô hình tùy thuộc vào yêu cầu và đặc điểm của dữ liệu.
- **Giải quyết vấn đề overfitting:** Polynomial regression có khả năng kháng lại hiện tượng overfitting (quá khớp) bằng cách giới hạn bậc của đa thức. Bằng cách chọn bậc thích hợp, mô hình có thể giảm sự phụ thuộc quá mức vào dữ liệu huấn luyện và tăng tính tổng quát hóa cho dữ liệu mới.

Nhược điểm:

- *Độ phức tạp cao*: Khi bậc của đa thức tăng lên, mô hình polynomial regression trở nên phức tạp hơn. Điều này có thể gây ra vấn đề khi huấn luyện và tính toán, đặc biệt là khi số lượng biến đầu vào lớn hoặc dữ liệu lớn.
- *Dễ gây overfitting*: Mặc dù polynomial regression có khả năng kháng lại overfitting, nếu không chọn bậc đa thức thích hợp, mô hình vẫn có thể dễ dàng quá khớp dữ liệu huấn luyện. Việc chọn bậc đa thức đúng là một quá trình thử và sai và yêu cầu kiến thức chuyên môn hoặc kỹ năng kinh nghiệm.
- *Suy giảm thông tin*: Đôi khi việc sử dụng polynomial regression có thể dẫn đến mất mát thông tin quan trọng. Khi đa thức có bậc cao, mô hình có thể gây mất mát chi tiết và sự khác biệt nhỏ trong dữ liệu, dẫn đến việc mô hình không thể diễn giải một cách rõ ràng và không hiệu quả.
- *Giả định về độc lập tuyến tính*: Polynomial regression dựa trên giả định rằng biến đầu vào và biến đầu ra có mối quan hệ tuyến tính khi sử dụng đa thức để mô hình hóa. Tuy nhiên, trong thực tế, mối quan hệ giữa các biến có thể không tuân theo một đa thức đơn giản, dẫn đến việc mô hình không phù hợp.

4.3. Random Forest

Ensemble Learning, tức là kết hợp nhiều mô hình học máy đơn lẻ để tạo ra một dự đoán cuối cùng. Nó được sử dụng cho cả bài toán phân loại và dự đoán.

Thuật toán Random Forest [5] gồm nhiều cây quyết định, mỗi cây quyết định đều có những yếu tố ngẫu nhiên:

1. Lấy ngẫu nhiên dữ liệu để xây dựng cây quyết định.
2. Lấy ngẫu nhiên các thuộc tính để xây dựng cây quyết định.

Do mỗi cây quyết định trong thuật toán Random Forest không dùng tất cả dữ liệu training, cũng như không dùng tất cả các thuộc tính của dữ liệu để xây dựng cây nên mỗi cây có thể sẽ dự đoán không tốt, khi đó mỗi mô hình cây quyết định không bị overfitting mà có thể bị underfitting, hay nói cách khác là mô hình có high bias. Tuy nhiên, kết quả cuối cùng của thuật toán Random Forest lại tổng hợp từ nhiều cây quyết định, thế nên thông tin từ các cây sẽ bổ sung thông tin cho nhau, dẫn đến mô hình có low bias và low variance, hay mô hình có kết quả dự đoán tốt.

Ý tưởng tổng hợp các cây quyết định của thuật toán Random Forest giống với ý tưởng của [The Wisdom of Crowds](#) được đề xuất bởi James Surowiecki vào năm 2004. The Wisdom of Crowds nói rằng thông thường tổng hợp thông tin từ 1 nhóm sẽ tốt hơn từ một cá nhân. Ở thuật toán Random Forest mình cũng tổng hợp thông tin từ 1 nhóm các cây quyết định và kết quả cho ra tốt hơn thuật toán Decision Tree với 1 cây quyết định.

Ví dụ: Mọi người muốn mua 1 sản phẩm trên tiki chẳng hạn, khi đọc review sản phẩm, nếu chỉ đọc 1 review thì có thể là ý kiến chủ quan của người đấy, hoặc sản phẩm người ấy mua không may bị lỗi gì; thông thường để có cái nhìn tốt về sản phẩm, mình hay đọc tất cả review rồi cho ra quyết định cuối cùng[1].

Các bước thực hiện thuật toán random forest regression gồm:

1. Chia tập dữ liệu thành tập huấn luyện và tập kiểm tra.
2. Xác định số lượng cây quyết định cần xây dựng.
3. Đối với mỗi cây, chọn một mẫu ngẫu nhiên và một tập con của các thuộc tính để xây dựng cây.
4. Xây dựng các cây quyết định dựa trên tập con thuộc tính này.
5. Đối với một bộ dữ liệu mới, sử dụng mỗi cây để dự đoán đầu ra và lấy trung bình giá trị của các dự đoán này làm giá trị đầu ra cuối cùng.

Ưu điểm:

- *Hiệu suất dự đoán cao*: Random Forest thường có hiệu suất dự đoán tốt hơn so với các thuật toán học máy đơn lẻ, như cây quyết định. Điều này đặc biệt đúng khi tập dữ liệu lớn và phức tạp.
- *Khả năng xử lý đa chiều và không gian biến động cao*: Random Forest có khả năng xử lý các bài toán có nhiều biến độc lập và không gian biến động cao. Nó có thể xử lý cả dữ liệu số và dữ liệu hạng mục.
- *Hạn chế ảnh hưởng của nhiễu và quá khớp*: Random Forest có khả năng ổn định hơn so với các mô hình học máy đơn lẻ khi đối mặt với nhiễu và có khả năng giảm quá khớp (overfitting).
- *Dễ dàng ước lượng độ quan trọng của biến*: Random Forest cung cấp thông tin về độ quan trọng của các biến độc lập trong quá trình dự đoán. Điều này giúp chúng tôi hiểu rõ hơn về tầm quan trọng của từng biến đối với kết quả dự đoán.

Nhược điểm:

- *Khó diễn giải*: Khi Random Forest sử dụng một số lượng lớn cây quyết định, việc diễn giải kết quả trở nên khó khăn do sự phức tạp và tương tác giữa các cây.
- *Tốn thời gian và tài nguyên tính toán*: Random Forest có thể tốn nhiều thời gian và tài nguyên tính toán khi huấn luyện và dự đoán, đặc biệt là khi số lượng cây và kích thước tập dữ liệu lớn.
- *Cần điều chỉnh siêu tham số*: Random Forest có một số siêu tham số cần được điều chỉnh, như số lượng cây, độ sâu cây, số lượng biến đặc trưng được chọn ngẫu nhiên cho mỗi cây. Việc điều chỉnh siêu tham số phù hợp là quan trọng để đạt hiệu suất tốt nhất từ thuật toán.

4.4. XGBoost

XGBoost (Extreme Gradient Boosting) [6] là một thuật toán học máy tập trung vào việc xây dựng một tập hợp các cây quyết định (decision trees) theo cách tối ưu hóa một hàm mất mát (loss function). Nó sử dụng kỹ thuật gradient boosting để khai thác lợi ích của các cây quyết định tập hợp và tối ưu hóa mô hình dự đoán. Là một giải thuật được base trên gradient boosting, tuy nhiên kèm theo đó là những cải tiến to lớn về mặt tối ưu thuật toán, về sự kết hợp hoàn hảo giữa sức mạnh phần mềm và phần cứng, giúp đạt được những kết quả vượt trội cả về thời gian training cũng như bộ nhớ sử dụng. XGBoost có khả năng xử lý cả bài toán phân loại và dự đoán.

Mã nguồn mở với ~350 contributors và ~3,600 commits trên Github, XGBoost cho thấy những khả năng ứng dụng đáng kinh ngạc của mình như:

- XGBoost có thể được sử dụng để giải quyết được tất cả các vấn đề từ hồi quy (regression), phân loại (classification), ranking và giải quyết các vấn đề do người dùng tự định nghĩa.
- XGBoost hỗ trợ trên Windows, Linux và OS X.
- Hỗ trợ tất cả các ngôn ngữ lập trình chính bao gồm C ++, Python, R, Java, Scala và Julia.
- Hỗ trợ các cụm AWS, Azure và Yarn và hoạt động tốt với Flink, Spark và các hệ sinh thái khác.

Ưu điểm:

- *Hiệu suất dự đoán cao:* XGBoost thường có hiệu suất dự đoán tốt hơn so với nhiều thuật toán học máy khác. Nó có khả năng tạo ra mô hình mạnh mẽ và ổn định, đặc biệt là khi áp dụng cho dữ liệu có kích thước lớn.
- *Xử lý tốt các biến độc lập và tương tác:* XGBoost có khả năng xử lý các biến độc lập có loại dữ liệu đa dạng, bao gồm cả dữ liệu số và dữ liệu hạng mục. Nó cũng xử lý tốt sự tương tác giữa các biến độc lập, giúp mô hình dự đoán chính xác hơn.

- *Hạn chế quá khớp*: XGBoost có khả năng giảm quá khớp (overfitting) thông qua việc sử dụng các kỹ thuật chặn (regularization) và cắt tỉa (pruning) cây quyết định.
- *Xử lý nhanh và tối ưu*: XGBoost được tối ưu hóa để đạt hiệu suất tính toán cao. Nó sử dụng các kỹ thuật như gradient boosting, việc song song hóa và sử dụng cây quyết định dựa trên cây histogram để tăng tốc độ huấn luyện và dự đoán.

Nhược điểm:

- *Đòi hỏi hiểu biết về siêu tham số*: XGBoost có nhiều siêu tham số cần được điều chỉnh để đạt hiệu suất tốt nhất. Điều này đòi hỏi người dùng có hiểu biết về thuật toán và kỹ năng điều chỉnh siêu tham số phù hợp.
- *Đòi hỏi tài nguyên tính toán*: XGBoost có thể yêu cầu nhiều tài nguyên tính toán, đặc biệt là khi xử lý dữ liệu lớn và sử dụng nhiều cây quyết định.

4.5. Neural Networks

Mạng nơ-ron là một phương thức trong lĩnh vực trí tuệ nhân tạo, được sử dụng để dạy máy tính xử lý dữ liệu theo cách được lấy cảm hứng từ bộ não con người. Đây là một loại quy trình máy học, được gọi là deep learning, sử dụng các nút hoặc nơ-ron liên kết với nhau trong một cấu trúc phân lớp tương tự như bộ não con người. Phương thức này tạo ra một hệ thống thích ứng được máy tính sử dụng để học hỏi từ sai lầm của chúng và liên tục cải thiện. Vì vậy, mạng nơ-ron nhân tạo nhằm tới giải quyết các vấn đề phức tạp, chẳng hạn như tóm tắt tài liệu hoặc nhận diện khuôn mặt, với độ chính xác cao hơn.

Mục đích chính của một mạng neural là học cách thực hiện một nhiệm vụ cụ thể thông qua việc điều chỉnh các trọng số kết nối giữa các neuron. Để làm được điều này, mạng neural cần được huấn luyện bằng cách cung cấp cho nó một tập dữ liệu đầu vào và đầu ra tương ứng với nhiệm vụ cần giải quyết. Sau đó, mạng

neural sẽ tự điều chỉnh các trọng số của nó để giảm sai số giữa đầu ra thực tế và đầu ra dự đoán của mạng.

Tuy nhiên việc xây dựng một mạng nơ-ron tốt và mạnh yêu cầu rất nhiều thời gian và tài nguyên để huấn luyện mô hình. Do đó, chúng tôi xây dựng một mạng nơ-ron có kiến trúc đơn giản như sau:

- *Input Layer*: Có số lượng node là $13 + 3 * \text{input_dim}$ với input_dim là số kỳ mà chúng tôi muốn huấn luyện (từ 1 đến 6).
- *Hidden Layer 1*: Có 64 node.
- *Hidden Layer 2*: Có 32 node.
- *Hidden Layer 3*: Có 12 node.
- *Hidden Layer 4*: Có 2 node.
- *Output Layer*: Có số lượng node là output_dim với output_dim là số lượng kỳ mà chúng tôi muốn dự đoán (1 hoặc 2).

Chương 5. THỰC NGHIỆM

5.1. Thang đo

5.1.1. Mean Square Error

Thang đo MSE (Mean Squared Error) là một trong những độ đo phổ biến được sử dụng để đánh giá hiệu suất của mô hình dự đoán. Nó được tính bằng cách tính trung bình tổng bình phương sai số giữa các giá trị dự đoán và giá trị thực tế.

$$MSE = \frac{1}{n} \sum_1^1 (y_i - \tilde{y}_i)^2$$

Trong đó:

MSE là Mean Squared Error.

n là số lượng mẫu trong tập dữ liệu.

y_i là giá trị thực tế.

\hat{y}_i là giá trị dự đoán tương ứng với y_i .

5.1.2. Thời gian huấn luyện

Độ đo thời gian huấn luyện (Training Time) là một độ đo được sử dụng để đánh giá thời gian mà mô hình mất để được huấn luyện trên tập dữ liệu. Độ đo này thường được sử dụng để so sánh hiệu quả và tốc độ huấn luyện của các mô hình khác nhau.

5.2. Cài đặt chi tiết

Với tổ chức dữ liệu đã trình bày ở phần 3.2.4, nhóm phải thiết kế một phương thức để load dữ liệu với số lượng kì đầu vào và số lượng kì đầu ra tùy ý.

Cụ thể hơn, nhóm đã định nghĩa phương thức load dữ liệu từ tập huấn luyện và kiểm thử.

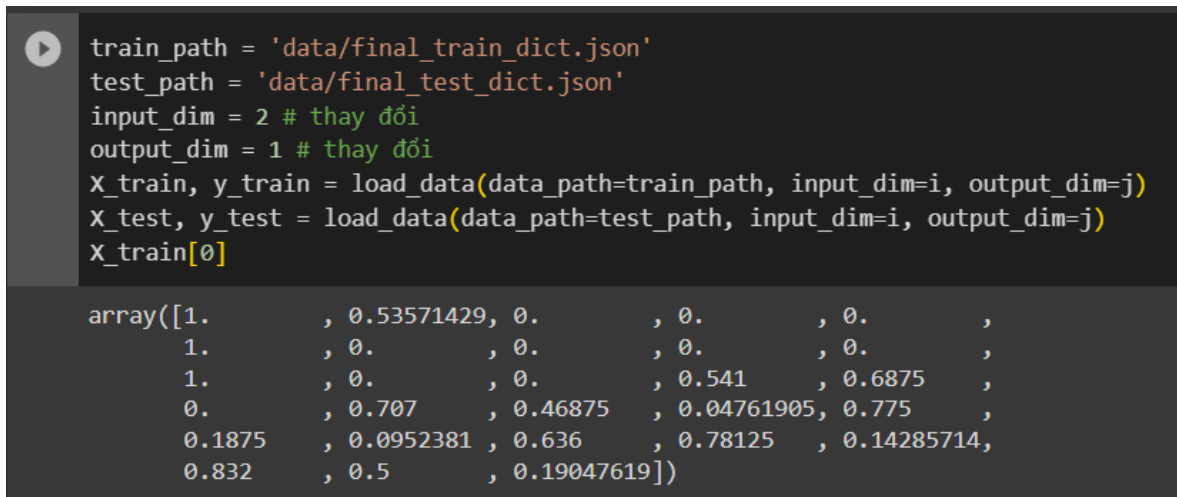
```
[11] train_path = 'data/final_train_dict.json'
    test_path = 'data/final_test_dict.json'
    for i in range(1, 7):
        for j in range(1, 3):
            X_train, y_train = load_data(data_path=train_path, input_dim=i, output_dim=j)
            X_test, y_test = load_data(data_path=test_path, input_dim=i, output_dim=j)
            print(f'Input{i}, Output{j}: ', X_train.shape, y_train.shape, X_test.shape, y_test.shape)
```

Input1, Output1: (40789, 16) (40789, 1) (19579, 16) (19579, 1)
Input1, Output2: (35808, 16) (35808, 2) (16509, 16) (16509, 2)
Input2, Output1: (35808, 19) (35808, 1) (16509, 19) (16509, 1)
Input2, Output2: (30990, 19) (30990, 2) (13520, 19) (13520, 2)
Input3, Output1: (30990, 22) (30990, 1) (13520, 22) (13520, 1)
Input3, Output2: (26282, 22) (26282, 2) (10588, 22) (10588, 2)
Input4, Output1: (26282, 25) (26282, 1) (10588, 25) (10588, 1)
Input4, Output2: (21678, 25) (21678, 2) (7711, 25) (7711, 2)
Input5, Output1: (21678, 28) (21678, 1) (7711, 28) (7711, 1)
Input5, Output2: (17184, 28) (17184, 2) (4873, 28) (4873, 2)
Input6, Output1: (17184, 31) (17184, 1) (4873, 31) (4873, 1)
Input6, Output2: (12794, 31) (12794, 2) (2120, 31) (2120, 2)

Hình 5-1 Demo phương thức load_data và kích thước của tập huấn luyện, tập kiểm thử tương ứng với cặp input, output khác nhau.

Bên cạnh đó, khi một sinh viên được chọn để load, sinh viên đó phải đảm bảo có kết quả học tập đủ nhiều (có số học kì lớn hơn số học kì đầu vào) để tránh khuyết dữ liệu. Một điểm dữ liệu sẽ chứa thông tin sinh viên và điểm các kì

trong một mảng một chiều, ví dụ được minh họa ở Hình 5-2. Cụ thể hơn, giả sử nhóm thực hiện bài toán dự đoán kết quả 1 kì tiếp theo dựa trên kết quả 2 kì trong quá khứ, điểm dữ liệu sinh viên sẽ chứa 13 giá trị lý lịch sinh viên và $3 * 2 = 6$ giá trị thể hiện kết quả của 2 kì trong quá khứ.



```
train_path = 'data/final_train_dict.json'
test_path = 'data/final_test_dict.json'
input_dim = 2 # thay đổi
output_dim = 1 # thay đổi
x_train, y_train = load_data(data_path=train_path, input_dim=i, output_dim=j)
x_test, y_test = load_data(data_path=test_path, input_dim=i, output_dim=j)
x_train[0]
```

```
array([1.          , 0.53571429, 0.          , 0.          , 0.          ,
        1.          , 0.          , 0.          , 0.          , 0.          ,
        1.          , 0.          , 0.          , 0.541         , 0.6875        ,
        0.          , 0.707         , 0.46875       , 0.04761905, 0.775         ,
        0.1875        , 0.0952381 , 0.636          , 0.78125        , 0.14285714,
        0.832         , 0.5          , 0.19047619])
```

Hình 5-2 Mô tả một điểm dữ liệu với bài toán nhận số học kì đầu vào là 2 và dự đoán 1 kì tiếp theo.

Môi trường mà nhóm sử dụng để chạy thực nghiệm là Google Colaboratory. Dữ liệu được lưu trữ thông qua Google Drive.

5.3. Kết quả và phân tích

5.3.1. Bảng kết quả với bài toán dự đoán kết quả 1 kì tiếp theo

Bảng 5-1 trình bày kết quả thực nghiệm với `output_dim = 1`, với số lượng input nhỏ, sự liên kết của các điểm dữ liệu ít phức tạp hơn, các phương pháp hồi quy hoạt động rất tốt bởi tính đơn giản và dễ thích nghi với dữ liệu này. Cụ thể hơn là Polynomial Regression đạt kết quả tốt nhất khi MSE của nó chỉ ở mức 0.0262 với `input_dim = 1` và 0.0274 với `input_dim = 2`, thấp nhất trong tất cả các phương pháp. Tuy nhiên khi tăng số lượng đầu vào thêm 1-2 kì, thì mạng nơ-ron lại chiếm ưu thế bởi cách nó tổ chức dữ liệu trong mạng một cách hệ thống

và liên kết chặt chẽ bằng các node trong mạng. Với kết quả MSE thu được của mạng nơ-ron lần lượt là 0.0294 và 0.0332 với input_dim = 3 và input_dim = 4.

Tuy nhiên để có thể đạt kết quả cao hơn, mạng nơ-ron cần được thiết kế và xây dựng cẩn thận và chuẩn chỉnh để tận dụng tối đa tiềm năng của nó. Do trong nghiên cứu của chúng tôi, mạng nơ-ron được sử dụng được xây dựng khá đơn giản, phục vụ cho mục đích khảo sát và đánh giá, do đó kết quả sẽ chưa tối ưu.

Cũng vì vậy, XGBoost – Extreme Gradient Boosting, trở nên vượt trội với phần còn lại khi bản thân nó có tốc độ huấn luyện nhanh, có khả năng scale để tính toán song song trên nhiều server, có thể tăng tốc bằng cách sử dụng GPU, nhờ vậy mà Big Data không phải là vấn đề của mô hình này. Vì thế, XGBoost thường được sử dụng và đã giành được nhiều chiến thắng trong các cuộc thi tại Kaggle. Không khó hiểu khi với lượng dữ liệu phức tạp chứa 5-6 học kì, XGBoost đạt hiệu quả cao nhất với MSE thấp nhất lần lượt là 0.0392 và 0.0536.

Bảng 5-1 Bảng kết quả thực nghiệm với output=1.

| Input | Method | Time | MSE |
|-----------|---------|---------------|---------------|
| input = 1 | dnn | 75.35653234 | 0.0276480502 |
| | polyreg | 0.8578972816 | 0.02627445206 |
| | lnreg | 0.04198122025 | 0.02789631419 |
| | rdfr | 0.0272166729 | 0.02789631419 |
| | xgboot | 0.3230397701 | 0.02656356502 |
| input = 2 | polyreg | 0.7874436378 | 0.0274186164 |
| | lnreg | 0.03136968613 | 0.02840822907 |
| | rdfr | 0.03008246422 | 0.02840822907 |
| | xgboot | 0.2988798618 | 0.02768494152 |
| | dnn | 83.12758231 | 0.02912136179 |
| input = 3 | polyreg | 0.9277992249 | 0.02956810619 |
| | lnreg | 0.02875137329 | 0.03063565008 |
| | rdfr | 0.03019833565 | 0.03063565008 |
| | xgboot | 0.3222634792 | 0.02954545852 |
| | dnn | 83.42066646 | 0.02949351604 |

| | | | |
|------------------|---------|---------------|---------------|
| input = 4 | polyreg | 1.096228361 | 0.03443242585 |
| | lnreg | 0.0531976223 | 0.03542667731 |
| | rdfr | 0.04061555862 | 0.03542667731 |
| | xgboot | 0.2922878265 | 0.03350719389 |
| | dnn | 48.37045813 | 0.03325439861 |
| input = 5 | polyreg | 1.713519573 | 0.04161933108 |
| | lnreg | 0.04180121422 | 0.04209887518 |
| | xgboot | 0.2865927219 | 0.03927958847 |
| | rdfr | 0.04024910927 | 0.04209887518 |
| | dnn | 39.14965701 | 0.04057915564 |
| input = 6 | rdfr | 0.03751587868 | 0.05804753505 |
| | xgboot | 0.2403688431 | 0.05365739737 |
| | lnreg | 0.02847957611 | 0.05804753505 |
| | polyreg | 1.21903801 | 0.05783530263 |
| | dnn | 41.97346783 | 0.06636771563 |

5.3.2. Bảng kết quả với bài toán dự đoán kết quả 2 kì tiếp theo

Bảng 5-2 trình bày kết quả thực nghiệm của các phương pháp cũng giống với Bảng 5-1, tuy nhiên thay vì dự đoán kết quả 1 kì tiếp theo, bài toán mở rộng thành 2 kì tiếp theo. Với số lượng input là 1, 2, 3, 4, Polynomial Regression là một thuật toán rất đáng để sử dụng khi số lượng chiều dữ liệu đầu vào không quá lớn. Nguồn dữ liệu đã xử lý tất cả thành phần ngoại lai (outliers), đây cũng là tác nhân ảnh hưởng tiêu cực nhất đến thuật toán này, do đó Polynomial Regression hoạt động tốt nhất so với phần còn lại. Tuy nhiên XGBoost và mạng nơ-ron vẫn chiếm ưu thế khi kết quả của hai phương pháp này vượt trội hơn phần còn lại với MSE lần lượt là 0.0370 và 0.0629 với input_dim = 5, 6.

Bảng 5-2 Bảng kết quả thực nghiệm với output = 2

| Input | Method | Time | MSE |
|------------------|--------|------------|---------------|
| input = 1 | dnn | 66.7945106 | 0.02345147943 |

| | | | |
|-----------|---------|---------------|---------------|
| | polyreg | 0.5685825348 | 0.02304176052 |
| | lnreg | 0.02359962463 | 0.02473288652 |
| | rdfr | 0.02988123894 | 0.02473288652 |
| | xgboot | 0.4516978264 | 0.02409120692 |
| input = 2 | lnreg | 0.03178977966 | 0.02442733843 |
| | polyreg | 0.644649744 | 0.02331517693 |
| | rdfr | 0.02905058861 | 0.02442733843 |
| | xgboot | 2.800322771 | 0.02443338535 |
| | dnn | 82.92106986 | 0.02358295584 |
| input = 3 | polyreg | 0.8011445999 | 0.02531252661 |
| | rdfr | 0.02487683296 | 0.02638237877 |
| | lnreg | 0.02520132065 | 0.02638237877 |
| | xgboot | 0.4529006481 | 0.0263924964 |
| | dnn | 82.94106889 | 0.02565399257 |
| input = 4 | polyreg | 1.205425501 | 0.02947913734 |
| | rdfr | 0.03472304344 | 0.0301592477 |
| | lnreg | 0.03803277016 | 0.0301592477 |
| | xgboot | 0.440690279 | 0.03006404427 |
| | dnn | 41.96437621 | 0.02956151634 |
| input = 5 | polyreg | 0.9414579868 | 0.03759147726 |
| | rdfr | 0.03643751144 | 0.03785950605 |
| | lnreg | 0.03905200958 | 0.03785950605 |
| | xgboot | 0.3770024776 | 0.03707500677 |
| | dnn | 42.24112916 | 0.03738038475 |
| input = 6 | dnn | 23.96595621 | 0.06293351266 |
| | polyreg | 0.9436926842 | 0.06544177186 |
| | lnreg | 0.0183198452 | 0.06566841791 |
| | rdfr | 0.02105164528 | 0.06566841791 |
| | xgboot | 0.3216190338 | 0.06371545172 |

Nhìn chung, chúng tôi có thể thấy rằng nếu tiếp cận đến các nguồn dữ liệu phức tạp thì XGBoost sẽ là một phương án rất có tiềm năng, bên cạnh đó, kiến thức về xây dựng mạng nơ-ron sẽ giúp cho mình có 1 kiến trúc đầy đủ và khác biệt

tương ứng với nguồn dữ liệu đầu vào. Đối với dữ liệu ít phức tạp hơn, ít chiều hơn, Polynomial Regression sẽ là một lựa chọn để giải quyết các bài toán supervised learning tương tự.

5.4. Thực nghiệm khác

5.4.1. Thực nghiệm với dữ liệu 2013 – 2019 không có thông tin sinh viên

Nhằm đánh giá và so sánh kết quả đầu ra của bài toán giữa trước và sau khi làm giàu thuộc tính huấn luyện, do đó, bảng 5-3 và bảng 5-4 mô tả kết quả thực nghiệm với dữ liệu đầu vào là không có thông tin sinh viên. Số lượng sinh viên vẫn giống như dữ liệu trong phần 5.3.

Đầu tiên, kết quả chúng tôi muốn xét đến là cho bài toán dự đoán điểm 1 kỳ tiếp theo. So sánh kết quả từ bảng 5-1 và bảng 5-3, MSE cao nhất mà bảng 5-1 ghi nhận được là 0.0664 (DNN: input = 6; output = 1), trong khi đó ở bảng 5-3 là 0.1175 (DNN: input = 6, output = 1). Điều này cho thấy các thuộc tính liên quan đến thông tin sinh viên quan trọng như thế nào trong việc dự đoán điểm trong tương lai, bên cạnh đó cũng cho thấy mức độ ảnh hưởng của giới tính, điểm cộng khu vực, khoa và hệ đào tạo tương ứng.

Bảng 5-3 Bảng kết quả thực nghiệm 13-19 output = 1

| Input | Method | Time | MSE |
|-----------|---------|----------------|---------------|
| input = 1 | dnn | 82.90018988 | 0.02902324323 |
| | polyreg | 0.05218029022 | 0.02772361788 |
| | lnreg | 0.00785446167 | 0.02837602891 |
| | rdfr | 0.007646083832 | 0.02837602891 |
| | xgboot | 0.1716291904 | 0.02786201711 |
| input = 2 | dnn | 61.43445945 | 0.02983998759 |
| | polyreg | 0.04399490356 | 0.02961865577 |
| | lnreg | 0.009330749512 | 0.03073352228 |
| | rdfr | 0.009022712708 | 0.03073352228 |

| | | | |
|-----------|---------|----------------|---------------|
| | xgboot | 1.875729561 | 0.02954653223 |
| input = 3 | dnn | 83.00171375 | 0.03529037109 |
| | polyreg | 0.0922973156 | 0.03451705595 |
| | lnreg | 0.009786367416 | 0.0357380057 |
| | rdfr | 0.00999712944 | 0.0357380057 |
| | xgboot | 0.1454958916 | 0.03366551491 |
| input = 4 | dnn | 44.7181437 | 0.04191756392 |
| | polyreg | 0.5282728672 | 0.04153535274 |
| | lnreg | 0.02218174934 | 0.042509404 |
| | rdfr | 0.01008892059 | 0.042509404 |
| | xgboot | 0.1395857334 | 0.03972705754 |
| input = 5 | dnn | 42.03529453 | 0.06632508245 |
| | polyreg | 0.1929111481 | 0.05803913556 |
| | lnreg | 0.01580095291 | 0.05888471227 |
| | rdfr | 0.009741306305 | 0.05888471227 |
| | xgboot | 0.129178524 | 0.05517211258 |
| input = 6 | dnn | 31.64044785 | 0.1175181795 |
| | polyreg | 0.2430820465 | 0.1123510193 |
| | lnreg | 0.01543545723 | 0.1115279692 |
| | rdfr | 0.008469820023 | 0.1115279692 |
| | xgboot | 0.1190736294 | 0.1028779632 |

Hơn nữa, tầm quan trọng của các thuộc tính thông tin sinh viên cũng được thể hiện ở bảng 5-4. Đối tượng mà sẽ so sánh với bảng 5-2 sẽ là bảng 5-2 với số điểm đầu ra dự đoán cho 2 học kỳ tiếp theo. Kết quả ghi nhận cho thấy, sự chênh lệch MSE (giữa 2 giá trị lớn nhất ở 2 bảng) giữa hai bảng lên đến 0.02. Về phần phân tích hiệu suất mô hình và biểu hiện của từng phương pháp đã được phân tích kỹ càng ở phần 5.3.

Bảng 5-4 Bảng kết quả thực nghiệm 13-19 với output = 2

| Input | Method | Time | MSE |
|-----------|---------|---------------|---------------|
| input = 1 | dnn | 65.97937536 | 0.02490523018 |
| | polyreg | 0.01732325554 | 0.02421043751 |

| | | | |
|-----------|---------|----------------|---------------|
| | lnreg | 0.009996175766 | 0.02444288487 |
| | rdfr | 0.009560346603 | 0.02444288487 |
| | xgboot | 0.3030605316 | 0.02463762979 |
| input = 2 | dnn | 83.35472202 | 0.02600176969 |
| | polyreg | 0.03654742241 | 0.02575596613 |
| | lnreg | 0.01055812836 | 0.02680873733 |
| | rdfr | 0.009907722473 | 0.02680873733 |
| | xgboot | 0.2027626038 | 0.02689286655 |
| input = 3 | dnn | 83.47943664 | 0.03046734124 |
| | polyreg | 0.08745861053 | 0.02953453864 |
| | lnreg | 0.01578807831 | 0.03048300579 |
| | rdfr | 0.009871959686 | 0.03048300579 |
| | xgboot | 0.2290689945 | 0.03014079702 |
| input = 4 | dnn | 41.97390914 | 0.0516152729 |
| | polyreg | 0.1080229282 | 0.03793625344 |
| | lnreg | 0.01771235466 | 0.03853621233 |
| | rdfr | 0.01003170013 | 0.03853621233 |
| | xgboot | 0.2240025997 | 0.03741446326 |
| input = 5 | dnn | 28.33940864 | 0.06889972932 |
| | polyreg | 0.1240859032 | 0.06717253439 |
| | lnreg | 0.01787424088 | 0.06679153366 |
| | rdfr | 0.009052276611 | 0.06679153366 |
| | xgboot | 0.1905703545 | 0.06455351223 |
| input = 6 | dnn | 19.52987695 | 0.08390302869 |
| | polyreg | 0.1282787323 | 0.08671217001 |
| | lnreg | 0.008570194244 | 0.08404860988 |
| | rdfr | 0.006853818893 | 0.08404860988 |
| | xgboot | 0.1721532345 | 0.08161436764 |

5.4.2. Thực nghiệm với dữ liệu 2013 – 2022 không có thông tin sinh viên

Sau khi thực nghiệm với miền giá trị là các sinh viên có khóa học từ 2013 đến 2019, chúng tôi đã mở rộng thực nghiệm lên 2022 (không sử dụng thông tin sinh viên vì sinh viên khóa sau 2019 bị thiếu đi lý lịch). Kết quả được trình bày ở bảng 5-5 và bảng 5-6.

Đối với bài toán dự đoán điểm 1 kỳ cho ra kết quả khá khả quan, tuy nhiên với input = 4 thì độ MSE lại quá cao ở tất cả phương pháp. Khác với kết quả trước đó, XGBoost vượt trội với input = 1, 2, 3, 4 nhưng khi số kỳ đầu vào lên đến 5 thì Polynomial Regression lại cao hơn, tuy nhiên sự chênh lệch là không lớn. Điều này có thể lý giải là bởi vì dữ liệu đã lược bỏ đi thông tin lý lịch, làm cho dữ liệu ít phức tạp đi, và nhóm cũng đã chưa tận dụng được đủ sức mạnh của XGBoost (khi có rất nhiều biến thể và cấu hình phù hợp với bài toán của mình).

Bảng 5-5 Bảng kết quả thực nghiệm 13-22 output = 1

| Input | Method | Time | MSE |
|-----------|---------|---------------|---------------|
| input = 1 | dnn | 105.609668 | 0.02964757844 |
| | polyreg | 0.03253555298 | 0.0297327796 |
| | rdfr | 0.01540184021 | 0.02952068549 |
| | xgboot | 0.1635363102 | 0.02884932885 |
| | lnreg | 0.01119017601 | 0.02952068549 |
| input = 2 | dnn | 93.20850801 | 0.05282228669 |
| | polyreg | 0.2215650082 | 0.04680715701 |
| | rdfr | 0.01388072968 | 0.04623924864 |
| | xgboot | 0.1870088577 | 0.04355648231 |
| | lnreg | 0.01315975189 | 0.04623924864 |
| input = 3 | dnn | 83.03839731 | 0.04839245849 |
| | polyreg | 0.1633205414 | 0.0473878935 |
| | rdfr | 0.02011632919 | 0.04722202926 |
| | xgboot | 0.183826685 | 0.04506497664 |
| | lnreg | 0.01387643814 | 0.04722202926 |

| | | | |
|------------------|---------|---------------|---------------|
| input = 4 | dnn | 60.65939569 | 0.150197395 |
| | polyreg | 0.2278838158 | 0.1538905623 |
| | rdfr | 0.01381897926 | 0.1501520343 |
| | xgboot | 0.1884832382 | 0.1395022207 |
| | lnreg | 0.01634550095 | 0.1501520343 |
| input = 5 | dnn | 45.4853189 | 0.09477514202 |
| | polyreg | 0.2546386719 | 0.08565389808 |
| | rdfr | 0.01274824142 | 0.08909920411 |
| | xgboot | 0.1700124741 | 0.08784895279 |
| | lnreg | 0.01280546188 | 0.08909920411 |

Nhưng khi đầu ra là 2 học kỳ tiếp theo thì Polynomial Regression hoàn toàn mất dạng, để lại sân chơi cho DNN và XGBoost với khả năng xử lý dữ liệu đa dạng và bài toán có độ phức tạp cao ở đầu ra lẫn đầu vào. Việc dự đoán điểm 2 kỳ một cách chính xác vẫn là 1 bài toán có nhiều thách thức, và dựa trên các kết quả thu được thì thêm dữ liệu về thông tin sinh viên (có thể phản ánh 1 phần năng lực con người) sẽ là một giải pháp ổn để tham khảo. Kết quả khi dữ liệu được mở rộng từ 2013-2019 đến 2013-2022 có phần cải thiện, cho thấy việc sở hữu một dữ liệu đủ lớn cũng có thể tạo ra sự chênh lệch nhất định. Tuy nhiên đánh đổi lại với điều này là thời gian và tài nguyên để huấn luyện một mô hình đủ tốt đáp ứng những yêu cầu cơ bản mà chúng ta quan tâm.

Bảng 5-6 Bảng kết quả thực nghiệm 13-22 với output = 2

| Input | Method | Time | MSE |
|------------------|---------------|---------------|---------------|
| input = 1 | dnn | 143.4214907 | 0.03026793931 |
| | polyreg | 0.0890212059 | 0.03025991709 |
| | rdfr | 0.01219344139 | 0.02949957665 |
| | xgboot | 0.2535748482 | 0.02853484525 |
| | lnreg | 0.01280760765 | 0.02949957665 |
| input = 2 | dnn | 83.06317854 | 0.039436194 |
| | polyreg | 0.07671260834 | 0.0317909111 |
| | rdfr | 0.01402139664 | 0.03089528701 |

| | | | |
|------------------|---------|---------------|---------------|
| | xgboot | 0.2877106667 | 0.02939674672 |
| | lnreg | 0.01464605331 | 0.03089528701 |
| input = 3 | dnn | 61.39514732 | 0.08035175597 |
| | polyreg | 0.09117460251 | 0.08846428356 |
| | rdfr | 0.01322507858 | 0.08540927429 |
| | xgboot | 0.2915875912 | 0.08212048282 |
| | lnreg | 0.01288604736 | 0.08540927429 |
| input = 4 | dnn | 83.0094161 | 0.05621754648 |
| | polyreg | 0.1396570206 | 0.05412479452 |
| | rdfr | 0.01284909248 | 0.05488238916 |
| | xgboot | 0.2602415085 | 0.05400622598 |
| | lnreg | 0.01450657845 | 0.05488238916 |

Chương 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1. Kết luận

Trong đề án này, chúng tôi đã thực hiện nhiều thử nghiệm với các mô hình dự đoán điểm trung bình các học kỳ tiếp theo của sinh viên. Chúng tôi đã thực nghiệm các mô hình như mạng neural, linear regression, random forest và xgboost toàn diện trên gần hết các tổ hợp có thể xảy ra giữa số kỳ input và số kỳ cần output. Kết quả đạt được cho thấy một số mô hình có khả năng dự đoán tốt hơn so với các mô hình khác, đánh giá dựa trên các độ đo như MSE và thời gian huấn luyện.

Tuy nhiên, còn một số hạn chế và vấn đề cần được xem xét. Một trong những hạn chế lớn nhất cần được giải quyết là việc biến đổi dữ liệu đầu vào thành mảng một chiều đã vô tình làm cho dữ liệu mất đi tính chất thời gian và liên tục, điều này ảnh hưởng không ít tới kết quả đầu ra của bài toán. Do đó, nên tìm cách tổ chức dữ liệu theo kiểu chuỗi các sự kiện nhằm tối ưu chất lượng mô hình.

6.2. Hướng phát triển

Dựa trên kết quả và kinh nghiệm từ đề án này, chúng tôi đề xuất một số hướng phát triển tiềm năng cho tương lai:

- Nâng cao mô hình hiện tại: Cải thiện mô hình Neural bằng cách thử nghiệm các kiến trúc khác nhau, tăng số lượng layer, tăng kích thước của các đơn vị.
- Mở rộng dữ liệu và đặt thử nghiệm: Mở rộng bộ dữ liệu Giáo dục đại học trên nhiều khoá hơn hoặc trên trường đại học khác.
- Xem xét các phương pháp tiền xử lý dữ liệu: Áp dụng các phương pháp tiền xử lý dữ liệu như trích xuất đặc trưng để cải thiện kết quả của mô hình.
- Tìm hiểu và áp dụng các phương pháp tiên tiến khác: Khám phá và thử nghiệm các phương pháp hybrid models để so sánh hiệu suất và khả năng ứng dụng.
- Đánh giá và so sánh với các phương pháp khác: So sánh mô hình hiện tại với các mạng RNN để đánh giá hiệu suất và ưu điểm của mô hình.
- Ứng dụng thực tế: Áp dụng mô hình vào làm hệ thống dự báo trước điểm số để sinh viên và nhà trường có giải pháp kịp thời.

TÀI LIỆU THAM KHẢO

- [1] B. Sravani and M. M. Bala, "Prediction of Student Performance Using Linear Regression," *IEEE*, 2020.
- [2] B. K. F. & S. S. Babu, "Predicting Academic Performance of Students Using a Hybrid Data Mining Approach," *Springer Nature*, 2019.
- [3] P. V. S. G. S. e. a. Nayak, "Predicting students' academic performance by mining the educational data through machine learning-based classification model," *Springer*, 2023.
- [4] NguyenDuong, "Linear Regression - Hồi quy tuyến tính trong Machine Learning," [Online]. Available: <https://viblo.asia/p/linear-regression-hoi-quy-tuyen-tinh-trong-machine-learning-4P856akRIY3>. [Accessed 27 5 2023].
- [5] T. Nguyễn, "Random Forest algorithm," [Online]. Available: https://machinelearningcoban.com/tabml_book/ch_model/random_forest.html. [Accessed 27 5 2023].
- [6] B. T. Tung, "Gradient Boosting - Tất tần tật về thuật toán mạnh mẽ nhất trong Machine Learning," [Online]. Available: <https://viblo.asia/p/gradient-boosting-tat-tan-tat-ve-thuat-toan-manh-me-nhat-trong-machine-learning-YWOZrN7vZQ0>. [Accessed 27 5 2023].

PHÂN CÔNG CÔNG VIỆC

| Sinh viên | Công việc | Mức độ hoàn thành |
|--------------------------------|---|-------------------|
| Trương Thành Thắng 2051907 | Phân công, theo dõi. Kiểm tra, tổng hợp và hoàn thiện báo cáo. | 100% |
| Ngô Văn Tấn Lưu 20521591 | Tìm hiểu các nghiên cứu liên quan. Viết demo bằng streamlit | 100% |
| Lê Trương Ngọc Hải 20520481 | Tiền xử lí dữ liệu Tìm hiểu Neural Network Thực nghiệm | 100% |
| Ngô Ngọc Sương 20521852 | Tìm hiểu dữ liệu Viết báo cáo | 100% |
| Trần Văn Lực 20521587 | Tiền xử lí dữ liệu Tìm hiểu các phương pháp học sâu Thực nghiệm | 100% |