BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT HƯNG YÊN

BÀI TẬP LỚN

HỌC MÁY CƠ BẢN

PHÁT HIỆN SINH VIÊN CÓ NGUY CƠ BỎ HỌC SỬ DỤNG CÁC MÔ HÌNH LOGISTIC REGRESSION, SUPPORT VECTOR MACHINE, DECISION TREE, RANDOM FOREST, GRADIENT BOOSTING

NGÀNH: KHOA HỌC MÁY TÍNH

SINH VIÊN: **NGUYỄN ĐỖ KHẢI HOÀN**

LỚP: **12423TN**

NGƯỜI HƯỚNG DẪN: **PGS.TS. NGUYỄN VĂN HẬU**

HƯNG YÊN – 2025

NHẬN XÉT

Nhận xét của giảng viên hướng dẫn:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

GIÁO VIÊN HƯỚNG DẪN

Nguyễn Văn Hậu

LỜI CAM ĐOAN

Em xin cam đoan bài tập lớn "Phát hiện sinh viên có nguy cơ nghỉ học sử dụng các mô hình Logistic Regression, Support Vector Machine, Decision Tree, Random Forest, Gradient Boosting" là kết quả thực hiện của bản thân em dưới sự hướng dẫn của thầy Nguyễn Văn Hậu.

Những phần sử dụng tài liệu tham khảo trong bài tập lớn đã được nêu rõ trong phần tài liệu tham khảo. Các kết quả trình bày trong bài tập lớn hoàn toàn là kết quả do bản thân em thực hiện.

Nếu vi phạm lời cam đoan này, em xin chịu hoàn toàn trách nhiệm trước khoa và nhà trường.

Hưng Yên, ngày tháng 12 năm 2025

Sinh viên

Nguyễn Đỗ Khải Hoàn

LỜI CÁM ƠN

Để có thể hoàn thành bài tập lớn này, lời đầu tiên em xin phép gửi lời cảm ơn tới bộ môn Học máy cơ bản, Khoa Công nghệ thông tin – Trường Đại học Sư phạm Kỹ thuật Hưng Yên đã tạo điều kiện thuận lợi cho em thực hiện bài tập lớn môn học này.

Đặc biệt em xin chân thành cảm ơn thầy Nguyễn Văn Hậu đã rất tận tình hướng dẫn, chỉ bảo em trong suốt thời gian thực hiện bài tập lớn vừa qua.

Em cũng xin chân thành cảm ơn tất cả các Thầy, các Cô trong Trường đã tận tình giảng dạy, trang bị cho em những kiến thức cần thiết, quý báu để giúp em thực hiện được bài tập lớn này.

Mặc dù em đã có cố gắng, nhưng với trình độ còn hạn chế, trong quá trình thực hiện đề tài không tránh khỏi những thiếu sót. Em hi vọng sẽ nhận được những ý kiến nhận xét, góp ý của các Thầy giáo, Cô giáo về những kết quả triển khai trong bài tập lớn.

Em xin trân trọng cảm ơn!

MỤC LỤC

DANH SÁCH HÌNH ẢNH

DANH SÁCH BẢNG

CHƯƠNG 1 TỔNG QUAN BÀI TOÁN

1.1 Bài toán

Ngày nay, cùng với sự phát triển nhanh chóng của xã hội, áp lực học tập và các vấn đề kinh tế – xã hội ngày càng gia tăng, kéo theo đó là tình trạng sinh viên bỏ học xuất hiện nhiều hơn, đặc biệt ở các quốc gia đang phát triển. Nguyên nhân dẫn đến bỏ học không chỉ xuất phát từ năng lực học tập mà còn liên quan đến các đặc điểm về nhân khẩu học (độ tuổi, giới tính, hoàn cảnh gia đình), điều kiện kinh tế – xã hội (khả năng đóng học phí, học bổng, việc làm của phụ huynh, biến động kinh tế), cũng như môi trường và định hướng học tập ngay từ giai đoạn nhập học. Trong nhiều trường hợp, các dấu hiệu bỏ học diễn ra âm thầm và chỉ trở nên rõ ràng khi người học đã nghỉ học kéo dài hoặc quyết định dừng học, khiến nhà trường khó can thiệp kịp thời.

Mặc dù nhà trường và các cơ quan quản lý giáo dục đã đưa ra nhiều chính sách hỗ trợ như tư vấn học tập, miễn giảm học phí, cấp học bổng hay chương trình phụ đạo, nhưng họ vẫn gặp khó khăn trong việc theo dõi và phát hiện sớm những trường hợp có nguy cơ. Thực tế, số lượng sinh viên lớn khiến giáo viên, cố vấn học tập không thể đánh giá thủ công và liên tục cho từng cá nhân. Điều này dẫn đến tình trạng quá tải trong công tác quản lý, đồng thời làm giảm hiệu quả của các biện pháp hỗ trợ vì thường đến muộn so với thời điểm cần can thiệp.

Để giải quyết vấn đề này, tôi có ý tưởng ứng dụng máy học (Machine Learning) để phân loại và dự đoán sinh viên có nguy cơ bỏ học, dựa trên những thông tin có thể thu thập ngay từ đầu quá trình học tập, chẳng hạn như thông tin học vấn trước khi nhập học**,** nhân khẩu học, yếu tố kinh tế – xã hội, và kết quả học tập ở giai đoạn đầu. Mô hình học máy sẽ học từ dữ liệu lịch sử để nhận diện các mẫu rủi ro, từ đó đưa ra cảnh báo sớm cho từng sinh viên. Nhờ vậy, nhà trường có thể ưu tiên nguồn lực hỗ trợ đúng đối tượng, giảm bớt khối lượng công việc theo dõi thủ công cho giáo viên/cố vấn, đồng thời tăng khả năng can thiệp kịp thời trước khi sinh viên thực sự bỏ học.

1.2 Dữ liệu bài toán

Dữ liệu bài toán được lấy từ website:

[Student Dropout Prediction]

	Marital status	Application mode	Application order	Course	Daytime/evening attendance\t	Previous qualification	Previous qualification (grade)	Nacionality	Mother's qualification	Father's qualification	...	Curricular units 2nd sem (credited)	Curricular units 2nd sem (enrolled)	Curricular units 2nd sem (evaluations)	Curricular units 2nd sem (approved)	Curricular units 2nd sem (grade)	Curricular units 2nd sem (without evaluations)	Unemployment rate	Inflation rate	GDP	Target
0	1	17	5	171	1	1	122.0	1	19	12	...	0	0	0	0	0.000000	0	10.8	1.4	1.74	Dropout
1	1	15	1	9254	1	1	160.0	1	1	3	...	0	6	6	6	13.666667	0	13.9	-0.3	0.79	Graduate
2	1	1	5	9070	1	1	122.0	1	37	37	...	0	6	0	0	0.000000	0	10.8	1.4	1.74	Dropout
3	1	17	2	9773	1	1	122.0	1	38	37	...	0	6	10	5	12.400000	0	9.4	-0.8	-3.12	Graduate
4	2	39	1	8014	0	1	100.0	1	37	38	...	0	6	6	6	13.000000	0	13.9	-0.3	0.79	Graduate

5 rows × 37 columns

Hình 1.1: Tổng quan bộ dữ liệu

```
print(df.shape[0])    # số dòng
print(df.shape[1])    # số cột

print(df.info())      # thông tin toàn bộ DataFrame
```

```
(4424, 37)
4424
37
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4424 entries, 0 to 4423
Data columns (total 37 columns):
 #   Column                                          Non-Null Count  Dtype
---  ------                                          --------------  -----
 0   Marital status                                  4424 non-null   int64
 1   Application mode                                4424 non-null   int64
 2   Application order                               4424 non-null   int64
 3   Course                                          4424 non-null   int64
 4   Daytime/evening attendance                      4424 non-null   int64
 5   Previous qualification                          4424 non-null   int64
 6   Previous qualification (grade)                  4424 non-null   float64
 7   Nacionality                                     4424 non-null   int64
 8   Mother's qualification                          4424 non-null   int64
 9   Father's qualification                          4424 non-null   int64
 10  Mother's occupation                             4424 non-null   int64
 11  Father's occupation                             4424 non-null   int64
 12  Admission grade                                 4424 non-null   float64
 13  Displaced                                       4424 non-null   int64
 14  Educational special needs                       4424 non-null   int64
 15  Debtor                                          4424 non-null   int64
 16  Tuition fees up to date                         4424 non-null   int64
 17  Gender                                          4424 non-null   int64
 18  Scholarship holder                              4424 non-null   int64
 19  Age at enrollment                               4424 non-null   int64
 20  International                                   4424 non-null   int64
 21  Curricular units 1st sem (credited)             4424 non-null   int64
 22  Curricular units 1st sem (enrolled)             4424 non-null   int64
 23  Curricular units 1st sem (evaluations)          4424 non-null   int64
 24  Curricular units 1st sem (approved)             4424 non-null   int64
 25  Curricular units 1st sem (grade)                4424 non-null   float64
 26  Curricular units 1st sem (without evaluations)  4424 non-null   int64
 27  Curricular units 2nd sem (credited)             4424 non-null   int64
 28  Curricular units 2nd sem (enrolled)             4424 non-null   int64
 29  Curricular units 2nd sem (evaluations)          4424 non-null   int64
 30  Curricular units 2nd sem (approved)             4424 non-null   int64
 31  Curricular units 2nd sem (grade)                4424 non-null   float64
 32  Curricular units 2nd sem (without evaluations)  4424 non-null   int64
 33  Unemployment rate                               4424 non-null   float64
 34  Inflation rate                                  4424 non-null   float64
 35  GDP                                             4424 non-null   float64
 36  Target                                          4424 non-null   object
dtypes: float64(7), int64(29), object(1)
memory usage: 1.2+ MB
```

Hình 1.2: Thông tin của bộ dữ liệu

- Dữ liệu bài toán gồm các cột:
    - Marital status: Tình trạng hôn nhân của sinh viên
    - Nacionality: Quốc tịch của sinh viên
    - Gender: Giới tính của sinh viên
    - Age at enrollment: Tuổi của sinh viên tại thời điểm nhập học
    - Mother's qualification: Trình độ học vấn của mẹ
    - Father's qualification: Trình độ học vấn của cha
    - Mother's occupation: Nghề nghiệp của mẹ

- Father's occupation: Nghề nghiệp của cha
- Application mode: Phương thức nộp hồ sơ
- Application order: Thứ tự nộp hồ sơ
- Course: Ngành sinh viên viên đăng ký
- Previous qualification: Bằng cấp hoặc trình độ học vấn trước khi vào đại học
- Previous qualification (grade): điểm của trình độ học vấn trước khi sinh viên nhập học đại học
- Daytime/evening attendance**:** Sinh viên học ban ngày hay buổi tối
- International: Có phải sinh viên quốc tế hay không
- Displaced: Sinh viên có phải người di dời / tị nạn hay không
- Educational special needs: Sinh viên có nhu cầu giáo dục đặc biệt hay không
- Debtor: Sinh viên có nợ học phí hay không
- Tuition fees up to date: Sinh viên có đóng học phí đúng hạn hay không
- Scholarship holder: Sinh viên có nhận học bổng hay không
- Unemployment rate: Tỷ lệ thất nghiệp (thời điểm nhập học)
- Inflation rate: Tỷ lệ lạm phát
- GDP: Tổng sản phẩm quốc nội
- Curricular units 1st sem (credited): Số học phần được công nhận
- Curricular units 1st sem (enrolled): Số học phần đăng ký
- Curricular units 1st sem (evaluations): Số học phần được đánh giá
- Curricular units 1st sem (approved): Số học phần đạt
- Curricular units 1st sem (grade): Điểm trung bình học kỳ 1
- Curricular units 1st sem (without evaluations): Số học phần không tham gia đánh giá
- Curricular units 2nd sem (credited): Số học phần được công nhận
- Curricular units 2nd sem (enrolled): Số học phần đăng ký
- Curricular units 2nd sem (evaluations): Số học phần được đánh giá
- Curricular units 2nd sem (approved): Số học phần đạt
- Curricular units 2nd sem (grade): Điểm trung bình học kỳ 2
- Curricular units 2nd sem (without evaluations): Số học phần không tham gia đánh giá
- Target:
  - Dropout: Bỏ học

- Enrolled: Đang theo học
- Graduate: Đã tốt nghiệp

- Dữ liệu bài toán là 1 file csv gồm 4424 rows x 37 columns
- Sau khi mô tả dữ liệu, ta có:

```
# Sử dụng hàm describe() để thống kê dữ liệu
df.describe()
```

	Marital status	Application mode	Application order	Course	Daytime/evening attendance\t	Previous qualification	Previous qualification (grade)	Nacionality	Mother's qualification	Father's qualification	...	Curricular units 1st sem (without evaluations)	Curricular units 2nd sem (credited)	Curricular units 2nd sem (enrolled)	Curricular units 2nd sem (evaluations)	Curricular units 2nd sem (approved)	Curricular units 2nd sem (grade)	Curricular units 2nd sem (without evaluations)	Unemployment rate	Inflation rate	GDP
count	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	...	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000
mean	1.178571	18.669078	1.727848	8856.642631	0.890823	4.577758	132.613314	1.873192	19.561935	22.275316	...	0.137658	0.541817	6.232143	8.063291	4.435805	10.230206	0.150316	11.566139	1.228029	0.001969
std	0.605747	17.484682	1.313793	2063.566416	0.311897	10.216592	13.188332	6.914514	15.603186	15.343108	...	0.690880	1.918546	2.195951	3.947951	3.014764	5.210808	0.753774	2.663850	1.382711	2.269935
min	1.000000	1.000000	0.000000	33.000000	0.000000	1.000000	95.000000	1.000000	1.000000	1.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	7.600000	-0.800000	-4.060000
25%	1.000000	1.000000	1.000000	9085.000000	1.000000	1.000000	125.000000	1.000000	2.000000	3.000000	...	0.000000	0.000000	5.000000	6.000000	2.000000	10.750000	0.000000	9.400000	0.300000	-1.700000
50%	1.000000	17.000000	1.000000	9238.000000	1.000000	1.000000	133.100000	1.000000	19.000000	19.000000	...	0.000000	0.000000	6.000000	8.000000	5.000000	12.200000	0.000000	11.100000	1.400000	0.320000
75%	1.000000	39.000000	2.000000	9556.000000	1.000000	1.000000	140.000000	1.000000	37.000000	37.000000	...	0.000000	0.000000	7.000000	10.000000	6.000000	13.333333	0.000000	13.900000	2.600000	1.790000
max	6.000000	57.000000	9.000000	9991.000000	1.000000	43.000000	190.000000	109.000000	44.000000	44.000000	...	12.000000	19.000000	23.000000	33.000000	20.000000	18.571429	12.000000	16.200000	3.700000	3.510000

8 rows × 36 columns

Hình 1.3: Thống kê dữ liệu

+ Tình trạng hôn nhân tương ứng với 6 tình trạng:
  - 1: độc thân (single)
  - 2: đã kết hôn (married)
  - 3: góa vợ/góa chồng (widower)
  - 4: đã ly hôn (divorced)
  - 5: sống chung không đăng ký kết hôn (facto union)
  - 6: ly thân hợp pháp (legally separated)
+ Phương thức nộp hồ sơ:
  - 1: giai đoạn 1- đoàn đại biểu chung (1st phase - general contingent)
  - 2: nghị định số 612/93 (Ordinance No. 612/93)
  - 5: giai đoạn 1- đoàn đại biểu đặc biệt (Đảo Azores) (1st phase - special contingent (Azores Island))
  - 10: nghị định số 854-B/99 (Ordinance No. 854-B/99)
  - 15: sinh viên quốc tế cử nhân (Cử nhân) (International student (bachelor))
  - 16: giai đoạn 1- đoàn đại biểu đặc biệt (Đảo Madeira) (1st phase - special contingent (Madeira Island))
  - 17: giai đoạn 2- đoàn đại biểu chung (2nd phase - general contingent)

- 18: giai đoạn 3- đoàn đại biểu chung (3nd phase - general contingent)
- 26: nghị định số 533-A/99, mục số b2 (Chương trình khác) (Ordinance No. 533-A/99, item b2) (Different Plan))
- 27: nghị định số 533-A/99, mục số b3 (Cơ sở giáo dục khác) (Ordinance No. 533-A/99, item b3 (Other Institution))
- 39: trên 23 tuổi (Over 23 years old)
- 42: chuyển trường (Transfer)
- 43: thay đổi ngành học (Change of course)
- 44: Người có bằng tốt nghiệp chuyên ngành công nghệ(Technological specialization diploma holders)
- 51: Thay đổi cơ sở giáo dục/ngành học (Change of institution/course)
- 53: Người có bằng tốt nghiệp ngắn hạn (Short cycle diploma holders)
- 57: Thay đổi cơ sở giáo dục/ngành học (Quốc tế) (Change of institution/course (International))

+ Thứ tự nộp hồ sơ: 0 (lựa chọn đầu tiên) - 9 (lựa chọn cuối cùng)
+ Khóa học:
  - 33: Công nghệ sản xuất nhiên liệu sinh học
  - 171: Thiết kế hoạt hình và đa phương tiện
  - 8014: Dịch vụ xã hội (học buổi tối)
  - 9003: Nông học
  - 9070: Thiết kế truyền thông
  - 9085: Điều dưỡng thú y
  - 9119: Kỹ thuật tin học
  - 9130: Chăn nuôi ngựa
  - 9147: Quản lý
  - 9238: Dịch vụ xã hội
  - 9254: Du lịch
  - 9500: Điều dưỡng
  - 9556: Vệ sinh răng miệng
  - 9670: Quản lý quảng cáo và tiếp thị
  - 9773: Báo chí và truyền thông
  - 9853: Giáo dục cơ bản

- 9991: Quản lý (học buổi tối)

+ Học ban ngày hay buổi tối: 1: ban ngày, 0: buổi tối

+ Bằng cấp hoặc trình độ học vấn trước khi vào đại học:
  - 1: Giáo dục trung học
  - 2: Giáo dục đại học - bằng cử nhân
  - 3: Giáo dục đại học - bằng chuyên ngành
  - 4: Giáo dục đại học - bằng thạc sĩ
  - 5: Giáo dục đại học - bằng tiến sĩ
  - 6: Trình độ học vấn cao hơn
  - 9: Lớp 12 - chưa hoàn thành
  - 10: Lớp 11 - chưa hoàn thành
  - 12: Khác - Lớp 11
  - 14: Lớp 10
  - 15: Lớp 10 - chưa hoàn thành
  - 19: Giáo dục cơ bản bậc 3 (lớp 9/10/11) hoặc tương đương
  - 38: Giáo dục cơ bản bậc 2 (lớp 6/7/8) hoặc tương đương
  - 39: Khóa học chuyên ngành công nghệ
  - 40: Giáo dục đại học - bằng chuyên ngành (bậc 1)
  - 42: Khóa học kỹ thuật cao cấp chuyên nghiệp
  - 43: Giáo dục đại học - bằng thạc sĩ (bậc 2)
+ Trình độ học vấn trước khi nhập học (điểm): từ 0-200
+ Quốc tịch:
  - 1: Bồ Đào Nha
  - 2: Đức
  - 6: Tây Ban Nha
  - 11: Ý
  - 13: Hà Lan
  - 14: Anh
  - 17: Litva
  - 21: Angola
  - 22: Cape Verde
  - 24: Guinea
  - 25: Mozambique
  - 26: Santosme
  - 32: Nhĩ Kỳ
  - 41: Brazil
  - 62: Romania
  - 100: Moldova (Cộng hòa)
  - 101: Mexico
  - 103: Ukraina
  - 105: Nga
  - 108: Cuba
  - 109: Colombia
+ Trình độ cha và mẹ sinh viên:
  - 1: Giáo dục Trung học - Lớp 12 hoặc tương đương
  - 2: Giáo dục Đại học - Bằng Cử nhân

- 3: Giáo dục Đại học - Bằng Cao đẳng
- 4: Giáo dục Đại học - Bằng Thạc sĩ
- 5: Giáo dục Đại học - Bằng Tiến sĩ
- 6: Trình độ Giáo dục Đại học
- 9: Lớp 12 - Chưa hoàn thành
- 10: Lớp 11 - Chưa hoàn thành
- 11: Lớp 7 (Cũ)
- 12: Khác - Lớp 11
- 14: Lớp 10
- 18: Khóa học Thương mại Tổng quát
- 19: Giáo dục Cơ bản Chu kỳ 3 (Lớp 9/10/11) hoặc tương đương
- 22: Khóa học kỹ thuật chuyên nghiệp
- 26: Năm thứ 7 của bậc học
- 27: Chu kỳ 2 của chương trình trung học phổ thông
- 29: Năm thứ 9 của bậc học - Chưa hoàn thành
- 30: Năm thứ 8 của bậc học
- 34: Không rõ
- 35: Không biết đọc hoặc viết
- 36: Biết đọc nhưng chưa học hết năm thứ 4
- 37: Giáo dục cơ bản chu kỳ 1 (năm thứ 4/5) hoặc tương đương
- 38: Giáo dục cơ bản chu kỳ 2 (năm thứ 6/7/8) hoặc tương đương
- 39: Khóa học chuyên ngành công nghệ
- 40: Giáo dục đại học - bằng cấp (chu kỳ 1)
- 41: Khóa học nghiên cứu cao cấp chuyên ngành
- 42: Khóa học kỹ thuật chuyên nghiệp cao cấp
- 43: Giáo dục đại học - Thạc sĩ (chu kỳ 2)
- 44: Giáo dục đại học - Tiến sĩ (chu kỳ 3)

+ Nghề nghiệp cha/mẹ sinh viên:

- 0: Sinh viên
- 1: Đại diện Cơ quan Lập pháp và Hành pháp, Giám đốc, Giám đốc và Quản lý điều hành
- 2: Chuyên gia trong lĩnh vực Hoạt động Trí tuệ và Khoa học
- 3: Kỹ thuật viên và Chuyên gia trình độ trung cấp
- 4: Nhân viên hành chính
- 5: Nhân viên Dịch vụ Cá nhân, An ninh và An toàn, và Nhân viên Bán hàng
- 6: Nông dân và Lao động lành nghề trong Nông nghiệp, Ngư nghiệp và Lâm nghiệp
- 7: Lao động lành nghề trong Công nghiệp, Xây dựng và Thủ công mỹ nghệ
- 8: Nhân viên Lắp đặt và Vận hành Máy móc, và Công nhân Lắp ráp
- 9: Lao động phổ thông
- 10: Chuyên gia trong Lực lượng Vũ trang
- 90: Trường hợp khác
- 99: (trống)
- 122: Chuyên gia Y tế
- 123: Giáo viên

- 125: Chuyên gia Công nghệ Thông tin và Truyền thông (CNTT)
- 131: Kỹ thuật viên và Chuyên gia Khoa học và Kỹ thuật trình độ trung cấp
- 132: Kỹ thuật viên và Chuyên gia Y tế trình độ trung cấp
- 134: Kỹ thuật viên trình độ trung cấp từ các dịch vụ pháp lý, xã hội, thể thao, văn hóa và các dịch vụ tương tự
- 141: Nhân viên Văn phòng, Thư ký và Nhân viên Xử lý Dữ liệu
- 143: Dữ liệu, Kế toán
- 144: Nhân viên thống kê, dịch vụ tài chính và các hoạt động liên quan đến đăng ký
- 151: Nhân viên hỗ trợ hành chính khác
- 152: Nhân viên dịch vụ cá nhân
- 153: Nhân viên bán hàng
- 171: Nhân viên chăm sóc cá nhân và những người tương tự
- 173: Công nhân xây dựng lành nghề và những người tương tự, trừ thợ điện
- 175: Công nhân lành nghề trong ngành in ấn, sản xuất dụng cụ chính xác, thợ kim hoàn, thợ thủ công và những người tương tự
- 191: Công nhân trong ngành chế biến thực phẩm, chế biến gỗ, may mặc và các ngành công nghiệp và thủ công khác
- 192: Nhân viên vệ sinh
- 193: Công nhân không lành nghề trong nông nghiệp, chăn nuôi, thủy sản và lâm nghiệp
- 194: Công nhân không lành nghề trong ngành khai thác, xây dựng, sản xuất và vận tải

+ Điểm nhập học: từ 0-200 điểm
+ Nhập cư:
  - 1: có
  - 0: không
+ Nhu cầu giáo dục đặc biệt:
  - 1: có
  - 0: không
+ Nợ:
  - 1: có
  - 0: không
+ Đóng học phí đúng hạn:
  - 1: có
  - 0: không
+ Giới tính:
  - 1: nam
  - 2: nữ
+ Học bổng
  - 1: có
  - 0: không
+ Sinh viên quốc tế:
  - 1: có
  - 0: không

1.3 Tiền xử lý dữ liệu

- Đồng nhất dữ liệu cho cột Father's qualification/Mother's qualification

```
qual_mapping = {1: "highschool",
                38:"pre highschool",
                37:"pre highschool",
                19:"highschool",
                11:"pre highschool",
                3:"graduate",
                2:"graduate",
                34:"unknown",
                4:"masters",
                27:"pre highschool",
                12:"unknown",
                39:"graduate",
                42:"masters",
                5:"masters",
                40:"graduate",
                6:"unknown",
                36:"less than 4",
                44:"masters",
                41:"graduate",
                29:"pre highschool",
                30:"pre highschool",
                9:"highschool",
                10:"highschool",
                35:"less than 4",
                14:"highschool",
                43:"masters",
                26:"pre highschool",
                25:"pre highschool",
                18:"graduate",
                22:"masters",
                31:"graduate",
                20:"highschool"}
```

Hình 1.4: Đồng nhất dữ liệu cho cột Father’s qualification/Mother’s qualification

- Đồng nhất dữ liệu cho cột Father's occupation/Mother's occupation

```
occ_mapping = {0: "student",
               1: "managerial",
               2: "professional",
               3: "technical",
               4: "professional",
               5: "service",
               6: "agriculture",
               7: "craftsmen",
               8: "factory",
               9: "elementary",
               10: "armed forces",
               90: "unknown",
               99: "unknown",
               101: "armed forces",
               102: "armed forces",
               103: "armed forces",
               112: "managerial",
               114: "managerial",
               121: "professional",
               122: "professional",
               123: "professional",
               124: "professional",
               131: "technical",
               132: "technical",
               134: "technical",
               135: "technical",
               141: "clerical",
               143: "technical",
               144: "clerical",
               151: "service",
               152: "service",
               153: "service",
               154: "service",
               161: "agriculture",
               163: "agriculture",
               171: "craftsmen",
               172: "craftsmen",
               174: "craftsmen",
               175: "craftsmen",
               181: "factory",
               182: "factory",
               183: "factory",
               192: "elementary",
               193: "elementary",
               194: "elementary",
               195: "elementary"}
```

Hình 1.5: Đồng nhất dữ liệu cho cột Father’s occupation/Mother’s occupation

- Đồng nhất dữ liệu cho cột Previous qualification

```
prev_qual_mapping = {1:'secondary_school',
                     2: 'graduate',
                     3: 'graduate',
                     4: 'masters',
                     5: 'doctorate',
                     6: 'unknown',
                     9: 'highschool',
                     10: 'highschool',
                     12: 'unknown',
                     14: 'highschool',
                     15: 'pre-highschool',
                     19: 'highschool',
                     38: 'pre-highschool',
                     39: 'graduate',
                     40: 'graduate',
                     42: 'masters',
                     43: 'masters'}
```

Hình 1.6: Đồng nhất dữ liệu cho cột Previous qualification

- Lấy các đặc trưng để huấn luyện

```
features_to_include = ['Marital status', 'Application mode', 'Application order', 'Course',
       'Attendance_mode', 'Previous qualification (grade)', 'Admission grade', 'Displaced',
       'Educational special needs', 'Debtor', 'Tuition fees up to date', 'Gender', 'Scholarship holder',
       'Age at enrollment', 'International', 'father_occ', 'mother_occ', 'father_qual', 'mother_qual', 'previous_qual',
        'Target']
```

Hình 1.7: Các đặc trưng dùng để huấn luyện

- Mã hóa dữ liệu: Sử dụng OneHotEncoder cho dữ liệu dạng chữ và sử dụng StandardScaler cho dữ liệu dạng số

```
select_cat_features = ColumnTransformer([('select_cat', 'passthrough', cat_features)])
cat_transformers = Pipeline([('selector', select_cat_features),
                             ('onehot', OneHotEncoder(handle_unknown='ignore')),
                             ])


select_num_features = ColumnTransformer([('select_num', 'passthrough', num_features)])
num_transformers = Pipeline([('selector', select_num_features),
                             ('scaler', StandardScaler()),
                             ])

preprocess_pipe = FeatureUnion([('cat', cat_transformers),
                                ('num', num_transformers),
                                ])
```

Hình 1.8: Mã hóa dữ liệu

1.4 Thống kê và trực quan hóa dữ liệu

a) Thống kê số sinh viên bỏ học, đang theo học, tốt nghiệp

```
def plot_counts(df,column, figsize = (4, 4)):
    plt.figure(figsize=figsize)
    sns.countplot(data=df, x = column, palette='viridis')
    plt.xlabel(column)
    plt.show()
```

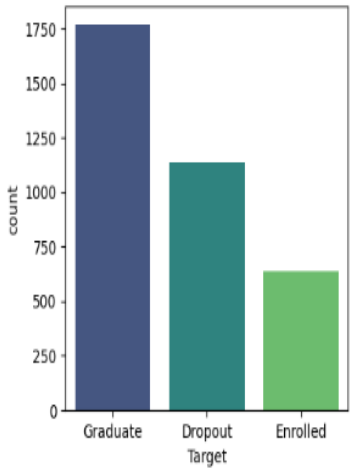
Hình 1.9: Hàm vẽ biểu đồ cột đếm tần suất

```
plot_counts(train,'Target')
```

```
/tmp/ipython-input-4160683961.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data=df, x = column, palette='viridis')
```



Hình 1.10: Thống kê số lượng sinh viên theo từng nhãn

```
dropout_rate = (train['Target'] == 'Dropout').sum() / train.shape[0]
enrolled_rate = (train['Target'] == 'Enrolled').sum() / train.shape[0]
graduation_rate = (train['Target'] == 'Graduate').sum() / train.shape[0]
```

```
print(f"Dropout: {np.round(dropout_rate, 2) * 100}%")
print(f"Enrolled: {np.round(enrolled_rate, 2) * 100}%")
print(f"Graduation: {np.round(graduation_rate, 2) * 100}%")
```

```
Dropout: 32.0%
Enrolled: 18.0%
Graduation: 50.0%
```

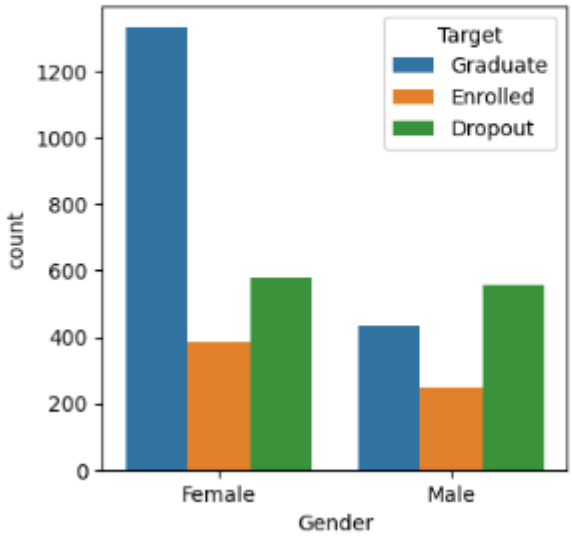
Hình 1.11: Phần trăm số sinh viên theo từng nhãn

- Nhận xét: có 32% sinh viên bỏ học, 18% vẫn tiếp tục theo học và 50% sinh viên tốt nghiệp

b) Đặc điểm nhân khẩu học

- Giới tính

```
plt.figure(figsize=(4,4))
sns.countplot(data=train, x='Gender', hue='Target')
plt.xticks(ticks=[0, 1], labels=['Female', 'Male'])
plt.show()
```



Hình 1.12: Thống kê giới tính sinh viên theo từng nhãn

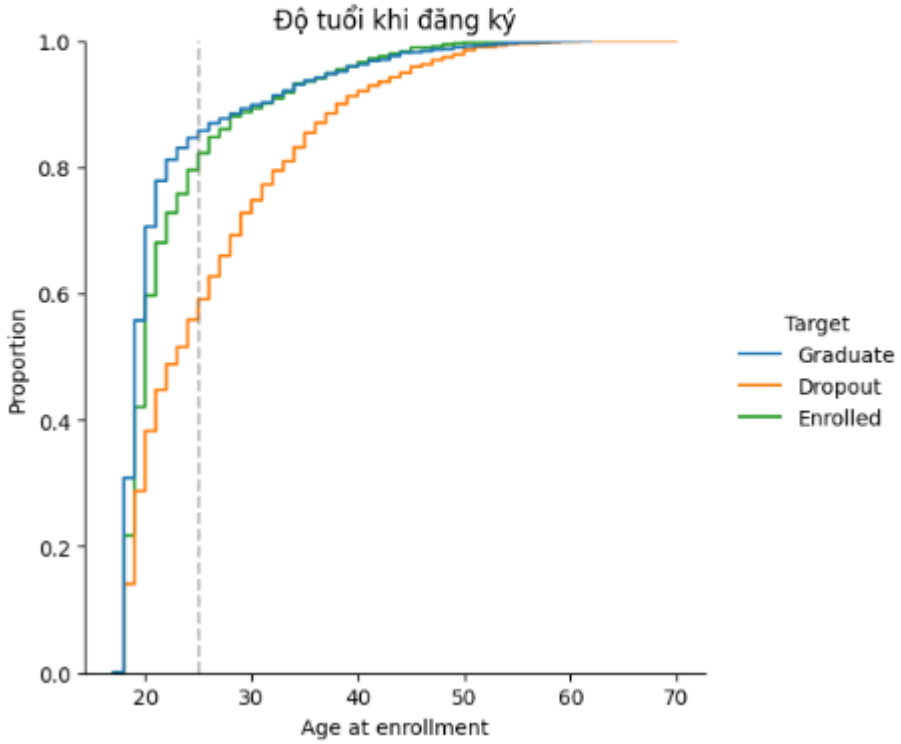
Picture 1. 1: Code for statistic and visualize for b

- Nhận xét: Sinh viên nữ có khả năng tốt nghiệp cao hơn sinh viên nam

- Độ tuổi khi đăng ký nhập học

```
plt.figure(figsize=(4,4))
sns.displot(data=train, x='Age at enrollment', hue='Target', kind='ecdf')
plt.axvline(x=25, linestyle='--', color="grey", alpha=0.5)
plt.title("Độ tuổi khi đăng ký")
plt.show()
```

```
<Figure size 400x400 with 0 Axes>
```

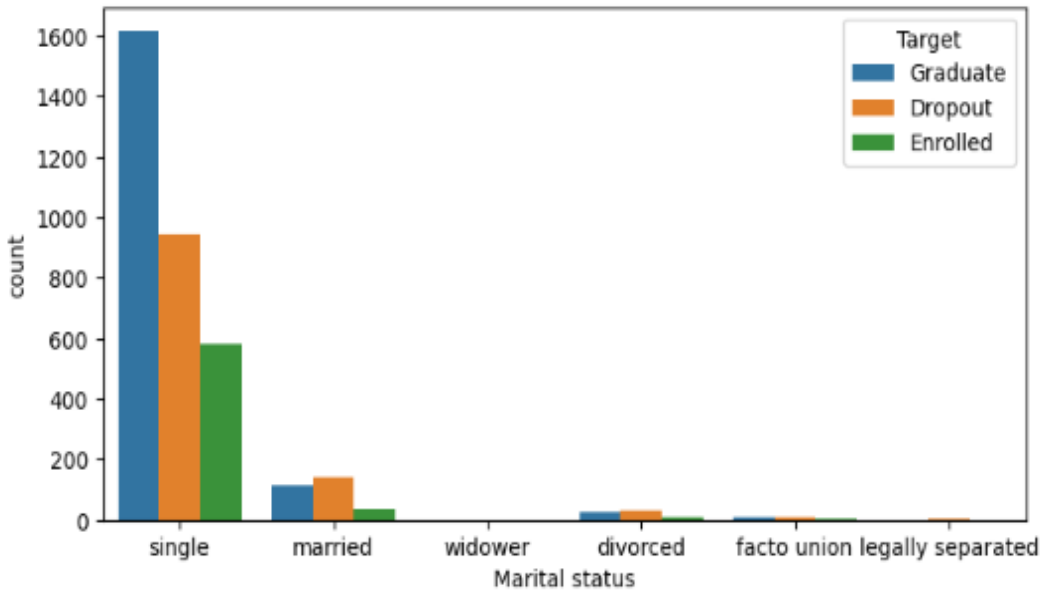


Hình 1.13: Thống kê độ tuổi sinh viên theo từng nhãn

- Nhận xét: Khoảng 40% số sinh viên bỏ học có độ tuổi từ 25 trở lên tại thời điểm nhập học. Hơn 80% sinh viên tốt nghiệp có độ tuổi dưới 25 tuổi

- Tình trạng hôn nhân

```
plt.figure(figsize=(8,4))
sns.countplot(data=train, x='Marital status', hue='Target')
plt.xticks(ticks=range(6), labels=['single', 'married', 'widower', 'divorced', 'facto union', 'legally s
plt.show()
```



Hình 1.14: Thống kê tình trạng hôn nhân của sinh viên theo từng nhãn

```
train[['Marital status', 'Target']].groupby(by = ['Marital status', 'Target']).value_counts()
```

Marital status	Target	count
1	Dropout	948
	Enrolled	581
	Graduate	1615
2	Dropout	144
	Enrolled	38
	Graduate	114
3	Dropout	1
	Enrolled	1
	Graduate	1
4	Dropout	33
	Enrolled	12
	Graduate	28
5	Dropout	8
	Enrolled	2
	Graduate	9
6	Dropout	3
	Enrolled	1

Hình 1.15: Số lượng sinh viên tương ứng với tình trạng hôn nhân theo từng nhãn

- Nhận xét: Tỷ lệ bỏ học cao hơn khi ở nhóm không độc thân

c) Nhu cầu giáo dục đặc biệt

```
counts = train['Educational special needs'].value_counts()
print(f"Có {counts[1]} sinh viên có nhu cầu trên {counts.sum()} sinh viên")
```

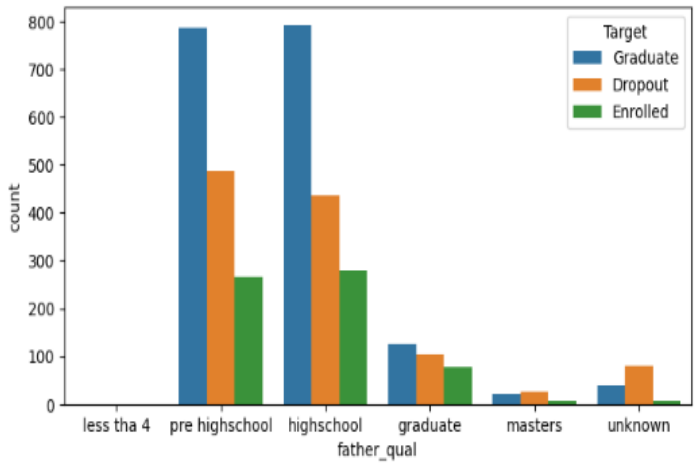
```
Có 41 sinh viên có nhu cầu trên 3539 sinh viên
```

Hình 1.16: Số sinh viên có nhu cầu giáo dục đặc biệt

d) Đặc điểm kinh tế- xã hội

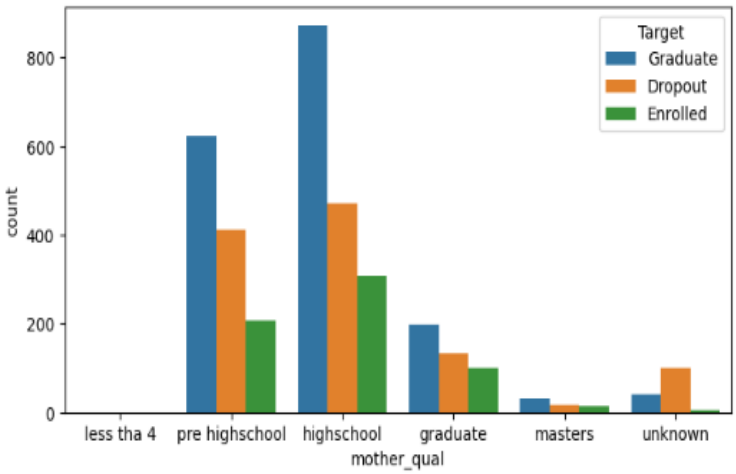
- Trình độ của cha mẹ sinh viên

```
plt.figure(figsize=(8,4))
sns.countplot(data=train, x='father_qual', order=["less tha 4", "pre highschool", "highschool", "graduate", "masters", "unknown"], hue='Target')
plt.show()
```



Hình 1.17: Thống kê trình độ học vấn của cha sinh viên theo từng nhãn

```
plt.figure(figsize=(8,4))
sns.countplot(data=train, x='mother_qual', order=["less tha 4", "pre highschool", "highschool", "graduate", "masters", "unknown"], hue='Target')
plt.show()
```

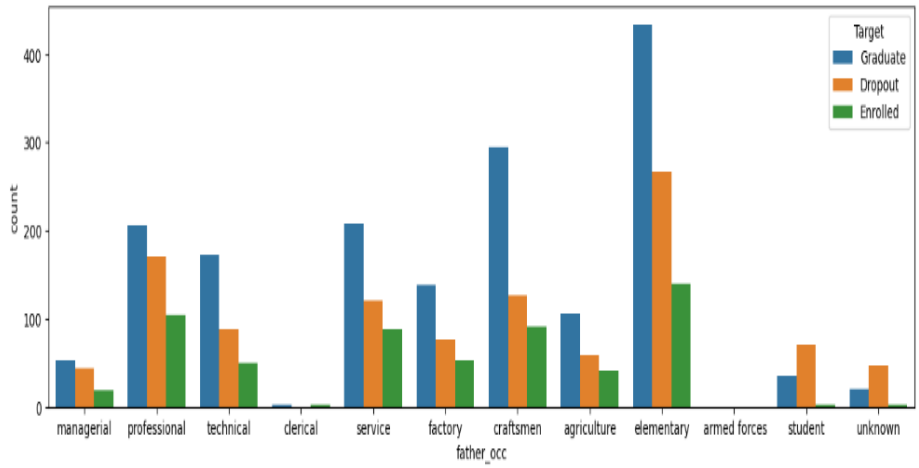


Hình 1.18: Thống kê trình độ học vấn của mẹ sinh viên theo từng nhãn

- Nhận xét: Sinh viên có cha mẹ đã tốt nghiệp trung học phổ thông có khả năng tốt nghiệp cao hơn so với những sinh viên chưa tốt nghiệp. Sinh viên có cha mẹ có bằng cử nhân hoặc cao hơn có khả năng bỏ học hoặc tốt nghiệp muộn hơn

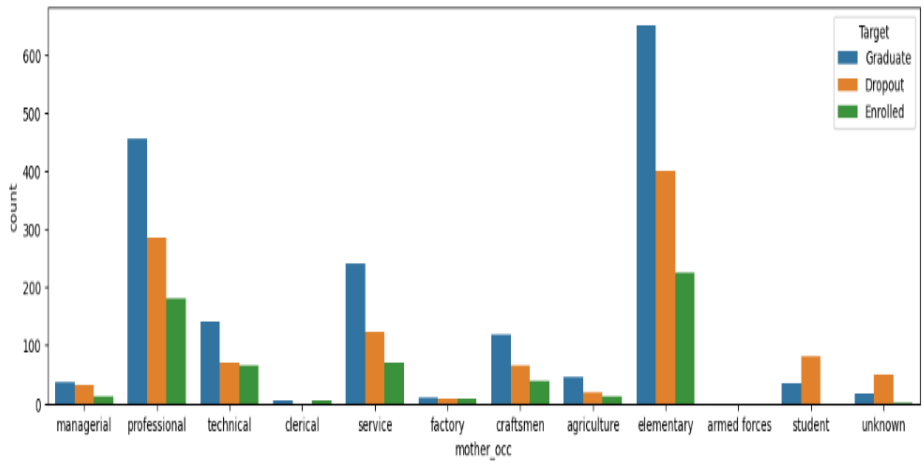
  - Nghề nghiệp của cha mẹ sinh viên

```
plt.figure(figsize=(16,4))
sns.countplot(data=train, x='father_occ', order=["managerial", "professional", "technical", "clerical", "service", "factory", "craftsmen", "agriculture", "elementary"," armed forces", "student", "unknown"], hue='Target')
plt.show()
```



Hình 1.19: Thống kê nghề nghiệp của cha sinh viên theo từng nhãn

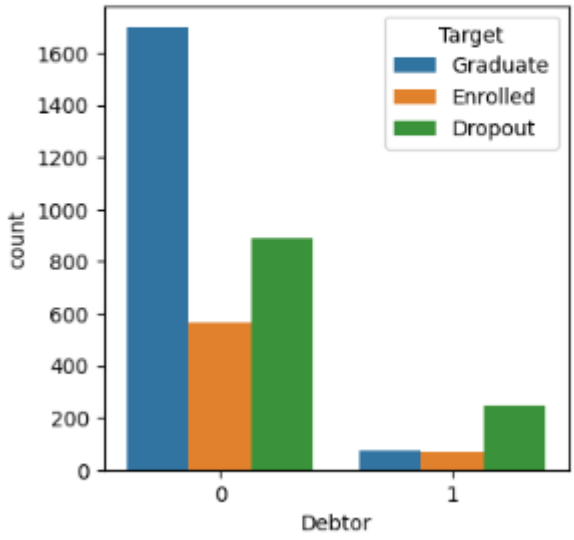
```
plt.figure(figsize=(16,4))
sns.countplot(data=train, x='mother_occ', order=["managerial", "professional", "technical", "clerical", "service", "factory", "craftsmen", "agriculture", "elementary"," armed forces", "student", "unknown"], hue='Target')
plt.show()
```



Hình 1.20: Thống kê nghề nghiệp của mẹ sinh viên theo từng nhãn

- Nhận xét: Tỷ lệ bỏ học cao hơn ở những sinh viên có cha mẹ không đi làm
  - Nợ nần

```
plt.figure(figsize=(4,4))
sns.countplot(data=train, x='Debtor', hue='Target')
plt.show()
```



Hình 1.21: Thống kê tình trạng nợ của sinh viên theo từng nhãn

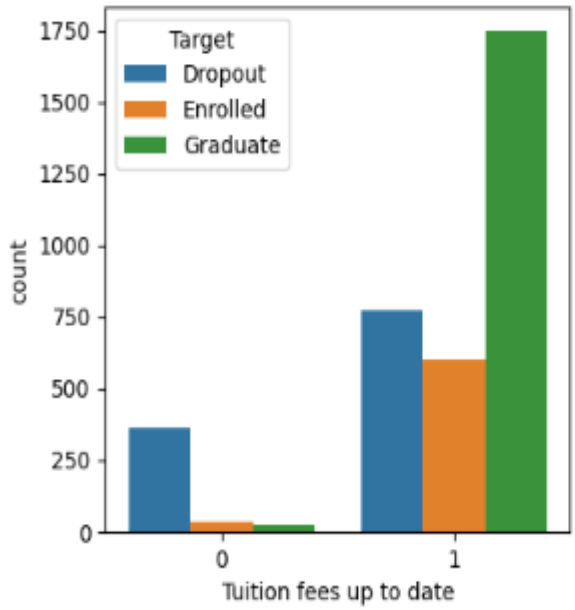
```
total_debtors = train.Debtor.sum()
total_students = train.shape[0]
debtors_dropped = train[train["Debtor"] == 1]["Target"].value_counts()["Dropout"]
print(f"{total_debtors*100/total_students:.2f}% sinh viên là người mắc nợ. {debtors_dropped*100/total_debtors:.2f}% trong số đó bỏ học")
```

```
10.99% sinh viên là người mắc nợ. 64.01% trong số đó bỏ học
```

Hình 1.22: Phần trăm sinh viên mắc nợ

- Tình trạng đóng học phí

```
plt.figure(figsize=(4,4))
sns.countplot(data=train, x='Tuition fees up to date', hue='Target')
plt.show()
```



Hình 1.23: Thống kê tình trạng đóng học phí của sinh viên theo từng nhãn

```
count_table = train.groupby(['Tuition fees up to date', 'Target']).size().reset_index(name='count')

total_per_status = count_table.groupby('Tuition fees up to date')['count'].transform('sum')
count_table['percentage'] = round(count_table['count'] / total_per_status * 100, 2)

pivot_percent = count_table.pivot(index='Tuition fees up to date', columns='Target', values='percentage')
pivot_percent
```

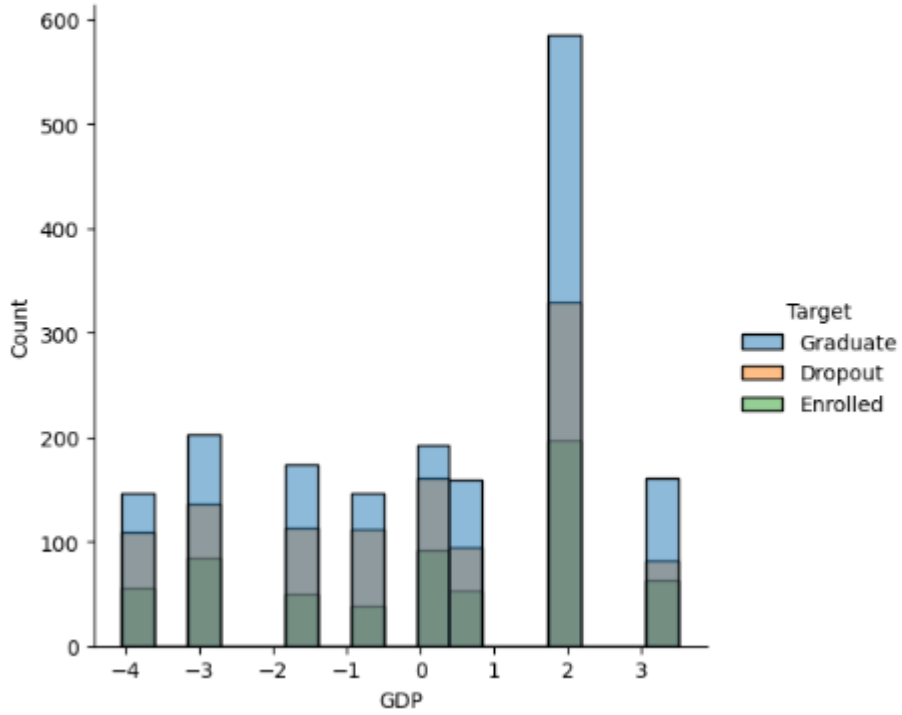
Target	Dropout	Enrolled	Graduate
Tuition fees up to date			
0	86.87	8.11	5.01
1	24.78	19.26	55.96

Hình 1.24: Phần trăm sinh viên đóng học phí theo từng nhãn

- Nhận xét: Lý do bỏ học cũng có thể do không đủ tiền đóng học phí
  - GDP

```
plt.figure(figsize=(4,4))
sns.displot(data=train, x='GDP', hue='Target', kind='hist', alpha=0.5)
plt.show()
```

```
<Figure size 400x400 with 0 Axes>
```

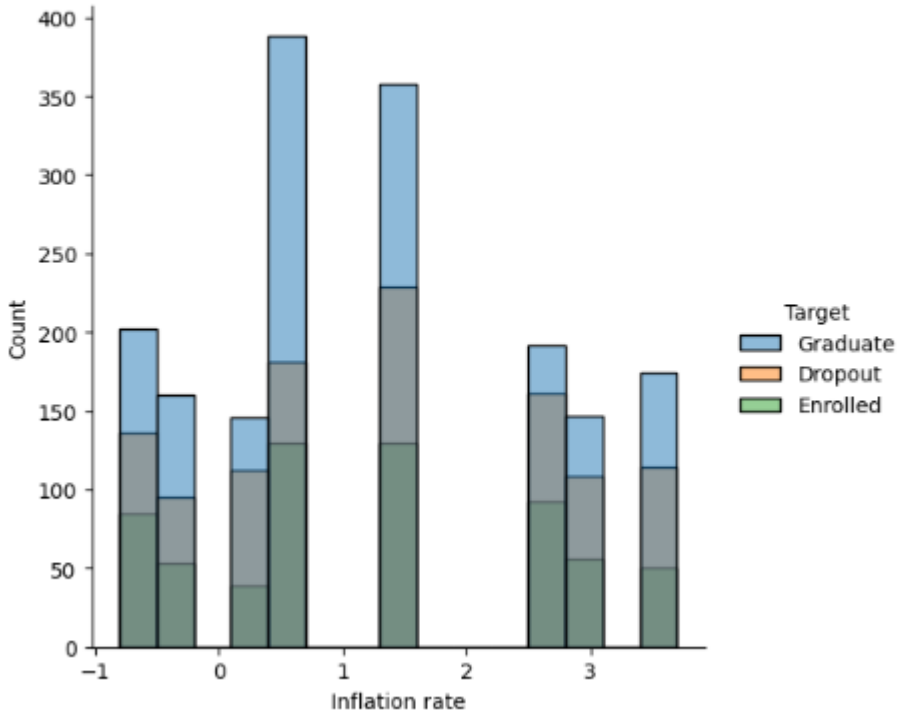


Hình 1.25: Thống kê GDP của sinh viên theo từng nhãn

- Tỷ lệ lạm phát

```
plt.figure(figsize=(4,4))
sns.displot(data=train, x='Inflation rate', hue='Target', kind='hist', alpha=0.5)
plt.show()
```

```
<Figure size 400x400 with 0 Axes>
```

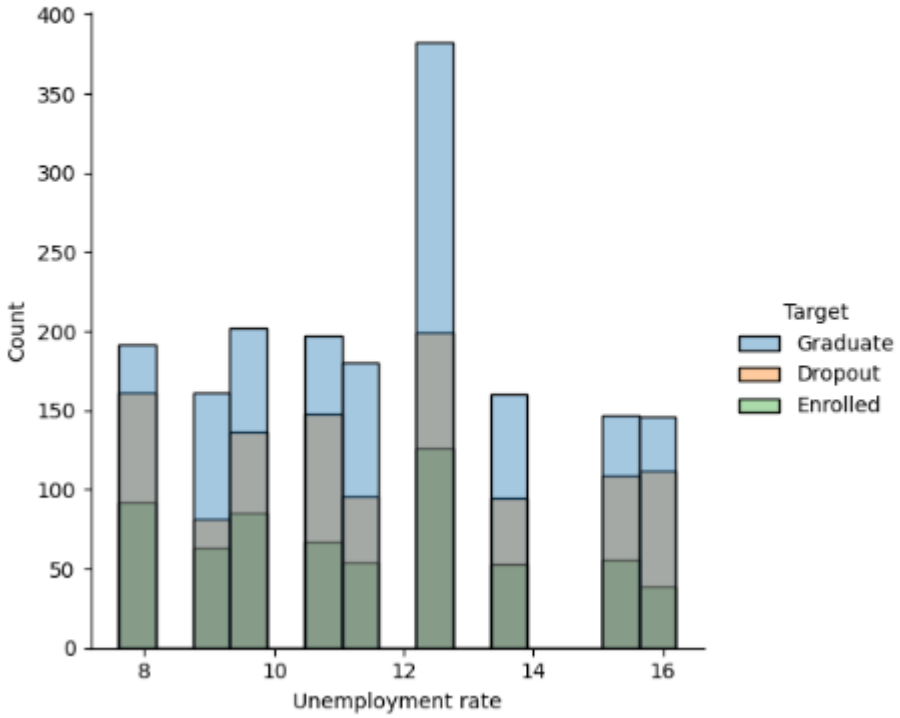


Hình 1.26: Thống kê tỷ lệ lạm phát của sinh viên theo từng nhãn

- Tỷ lệ thất nghiệp

```
plt.figure(figsize=(4,4))
sns.displot(data=train, x='Unemployment rate', hue='Target', kind='hist', alpha=0.4)
plt.show()
```

```
<Figure size 400x400 with 0 Axes>
```

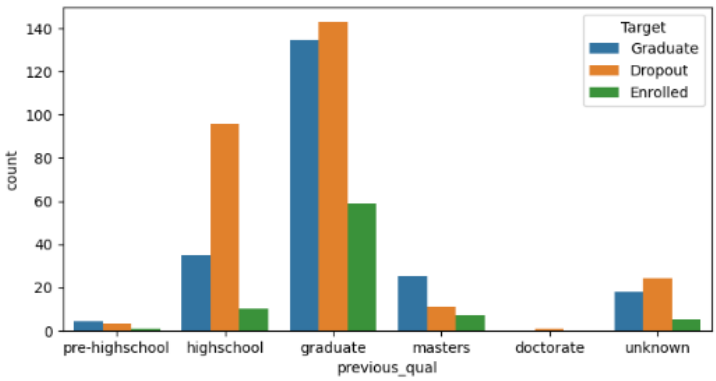


Hình 1.27: Thống kê tỷ lệ thất nghiệp của sinh viên theo từng nhãn

e) Kết quả học tập của sinh viên

- Trình độ học vấn của sinh viên

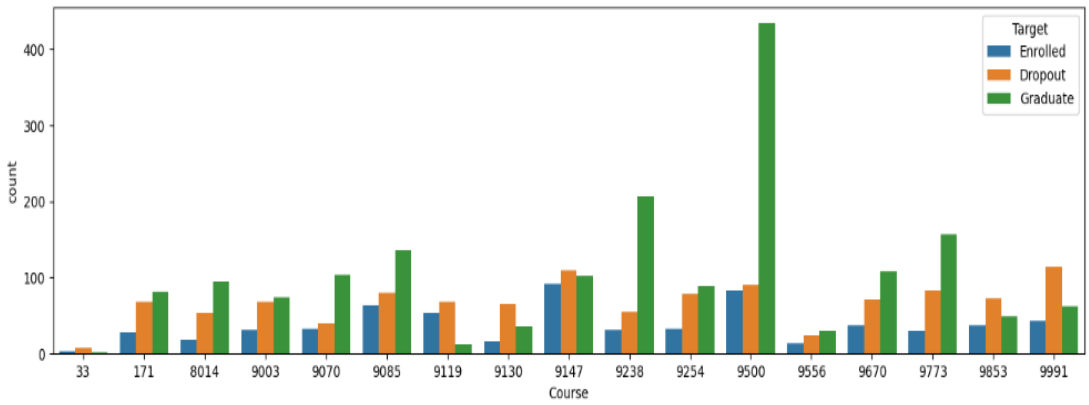
```
plt.figure(figsize=(8,4))
sns.countplot(data=train, x='previous_qual', order=["pre-highschool", "highschool", "graduate", "masters","doctorate", "unknown"], hue='Target')
plt.show()
```



Hình 1.28: Thống kê trình độ học vấn của sinh viên theo từng nhãn

- Khóa sinh viên viên đăng kí

```
plt.figure(figsize=(16,4))
sns.countplot(data=train, x='Course', hue='Target')
plt.show()
```



Hình 1.29: Thống kê khóa học sinh viên đăng ký theo từng nhãn

```
count_table = train.groupby(['Course', 'Target']).size().reset_index(name='count')

total_per_status = count_table.groupby('Course')['count'].transform('sum')
count_table['percentage'] = round(count_table['count'] / total_per_status * 100, 2)

pivot_percent = count_table.pivot(index='Course', columns='Target', values='percentage')
pivot_percent
```

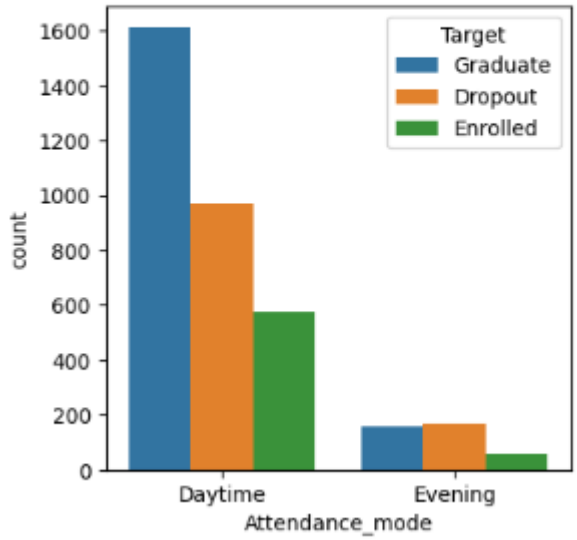
Course	Dropout	Enrolled	Graduate
33	70.00	20.00	10.00
171	38.29	15.43	46.29
8014	32.12	10.91	56.97
9003	39.18	18.13	42.69
9070	22.41	18.39	59.20
9085	28.52	22.74	48.74
9119	51.13	39.85	9.02
9130	55.65	13.91	30.43
9147	36.09	30.13	33.77
9238	18.90	10.31	70.79
9254	39.39	16.16	44.44
9500	14.83	13.67	71.50
9556	36.36	19.70	43.94
9670	32.86	16.90	50.23
9773	30.71	10.86	58.43
9853	45.57	23.42	31.01
9991	52.07	19.35	28.57

Hình 1.30: Phần trăm khóa học sinh viên đăng kí theo từng nhãn

- Nhận xét : 5 khóa học có tỉ lệ bỏ học cao là Công nghệ sản xuất nhiên liệu sinh học (70%), Kỹ thuật tin học (51,13%), Chăn nuôi ngựa (55,65%), Giáo dục cơ bản (45,57%), Quản lý (tham dự buổi tối) (52,07%)

  - Thời gian học(Ban ngày/buổi tối)

```
plt.figure(figsize=(4,4))
sns.countplot(data=train, x='Attendance_mode', hue='Target')
plt.show()
```

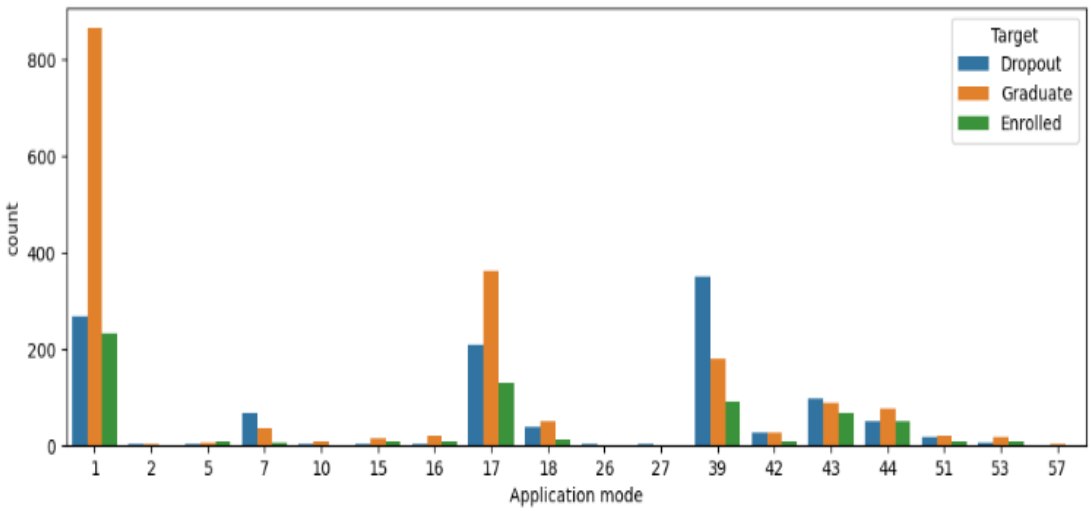


Hình 1.31: Thống kê thời gian học của sinh viên theo từng nhãn

- Nhận xét: Học buổi tối có khả năng bỏ học cao hơn

  - Phương thức nộp hồ sơ

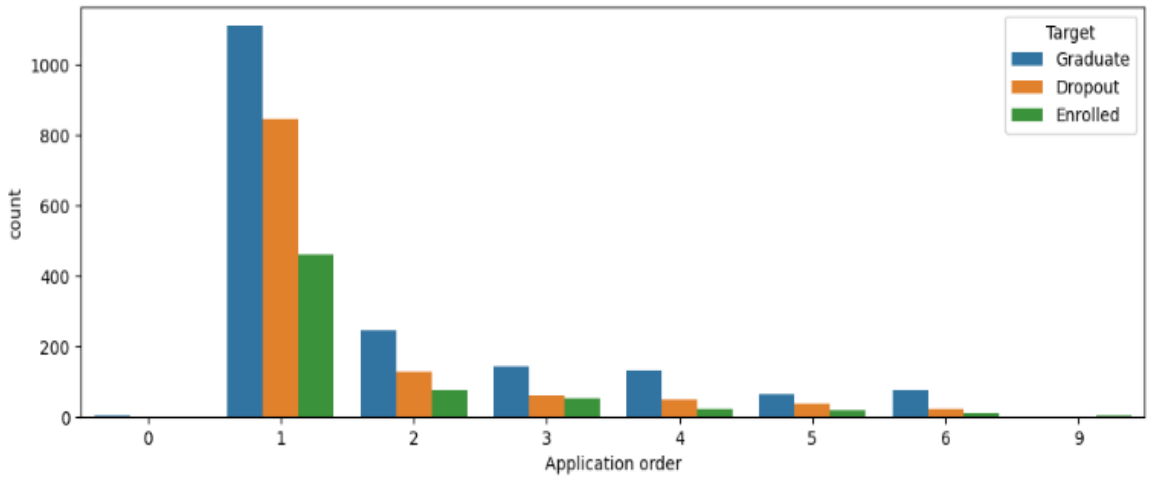
```
plt.figure(figsize=(12,4))
sns.countplot(data=train, x='Application mode', hue='Target')
plt.show()
```



Hình 1.32: Thống kê phương thức nộp hồ sơ của sinh viên theo từng nhãn

- Thứ tự nộp hồ sơ

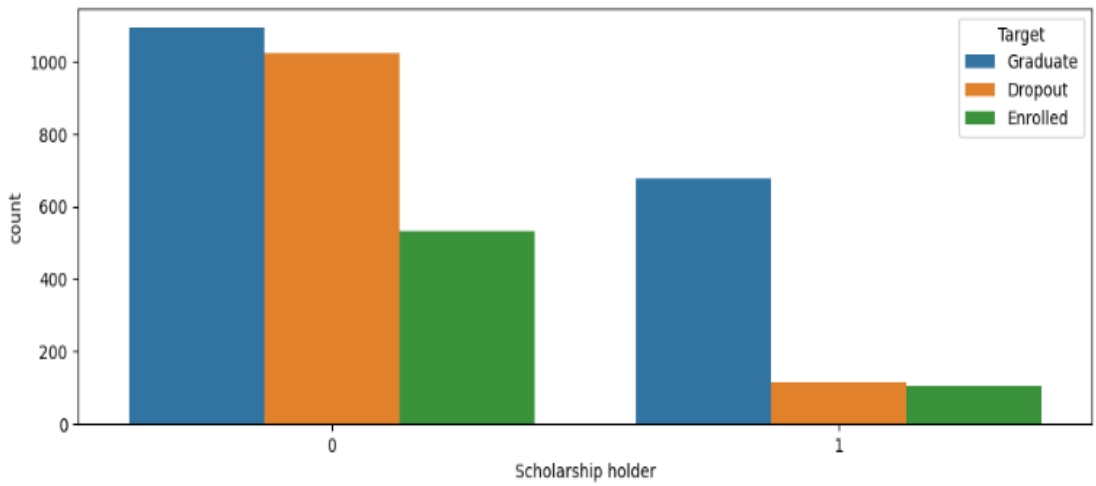
```
plt.figure(figsize=(12,4))
sns.countplot(data=train, x='Application order', hue='Target')
plt.show()
```



Hình 1.33: Thống kê thứ tự nộp hồ sơ của sinh viên theo từng nhãn

- Học bổng

```
plt.figure(figsize=(12,4))
sns.countplot(data=train, x='Scholarship holder', hue='Target')
plt.show()
```

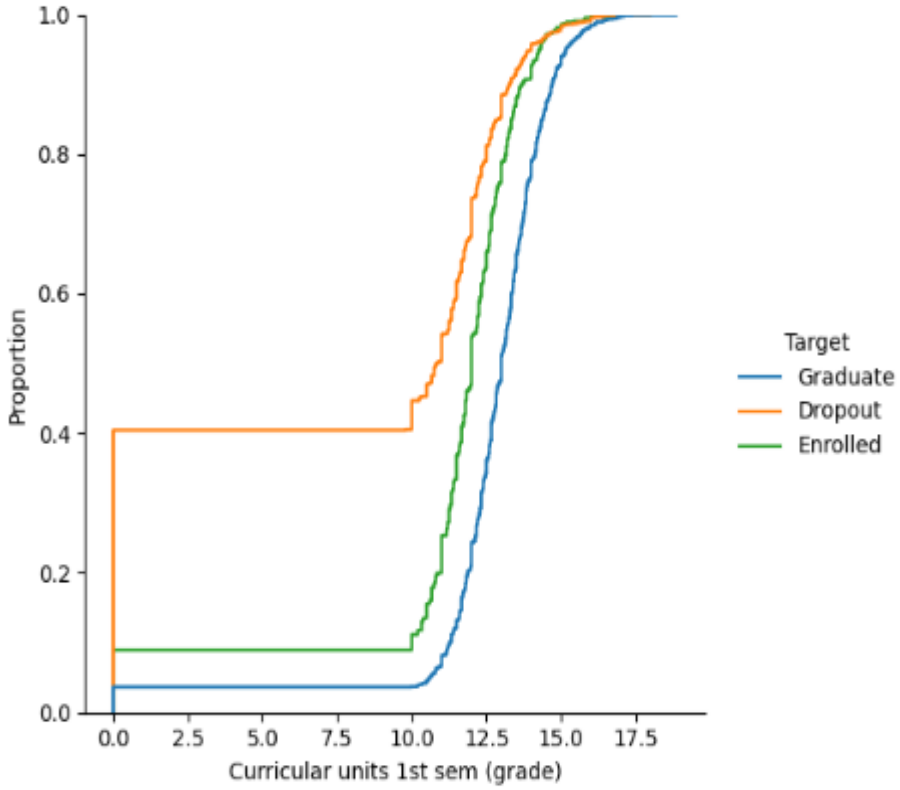


Hình 1.34: Thống kê học bổng của sinh viên theo từng nhãn

- Điểm số 2 kỳ đầu

```
plt.figure(figsize=(4,4))
sns.displot(data=train, x='Curricular units 1st sem (grade)', hue='Target', kind='ecdf')
plt.show()
```

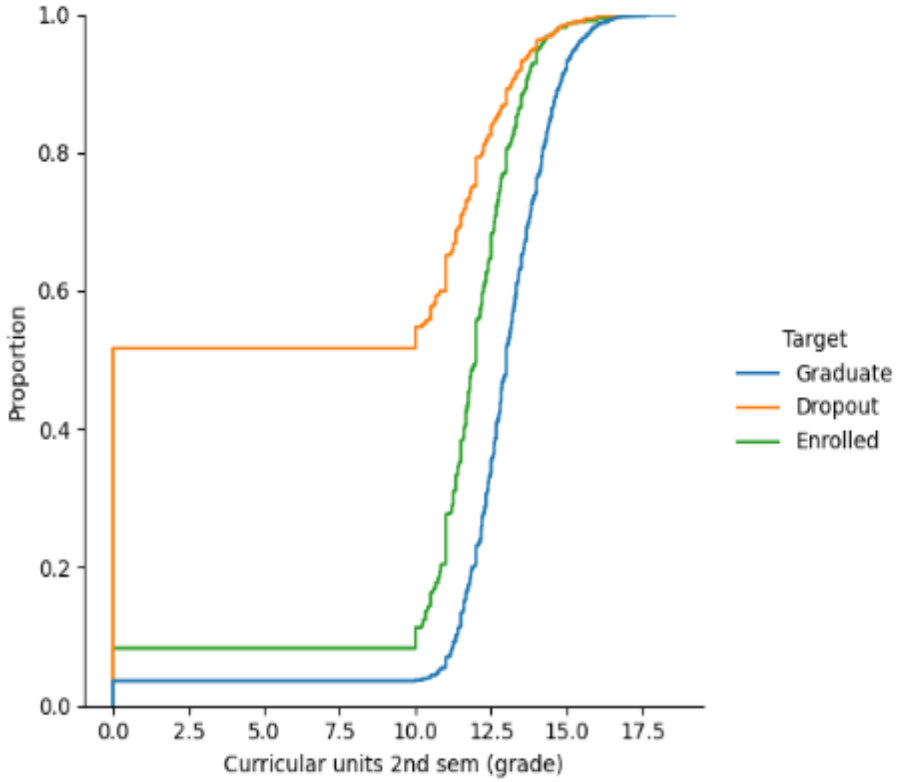
```
<Figure size 400x400 with 0 Axes>
```



Hình 1.35: Thống kê điểm học kì 1 của sinh viên theo từng nhãn

```
plt.figure(figsize=(4,4))
sns.displot(data=train, x='Curricular units 2nd sem (grade)', hue='Target', kind='ecdf')
plt.show()
```

```
<Figure size 400x400 with 0 Axes>
```

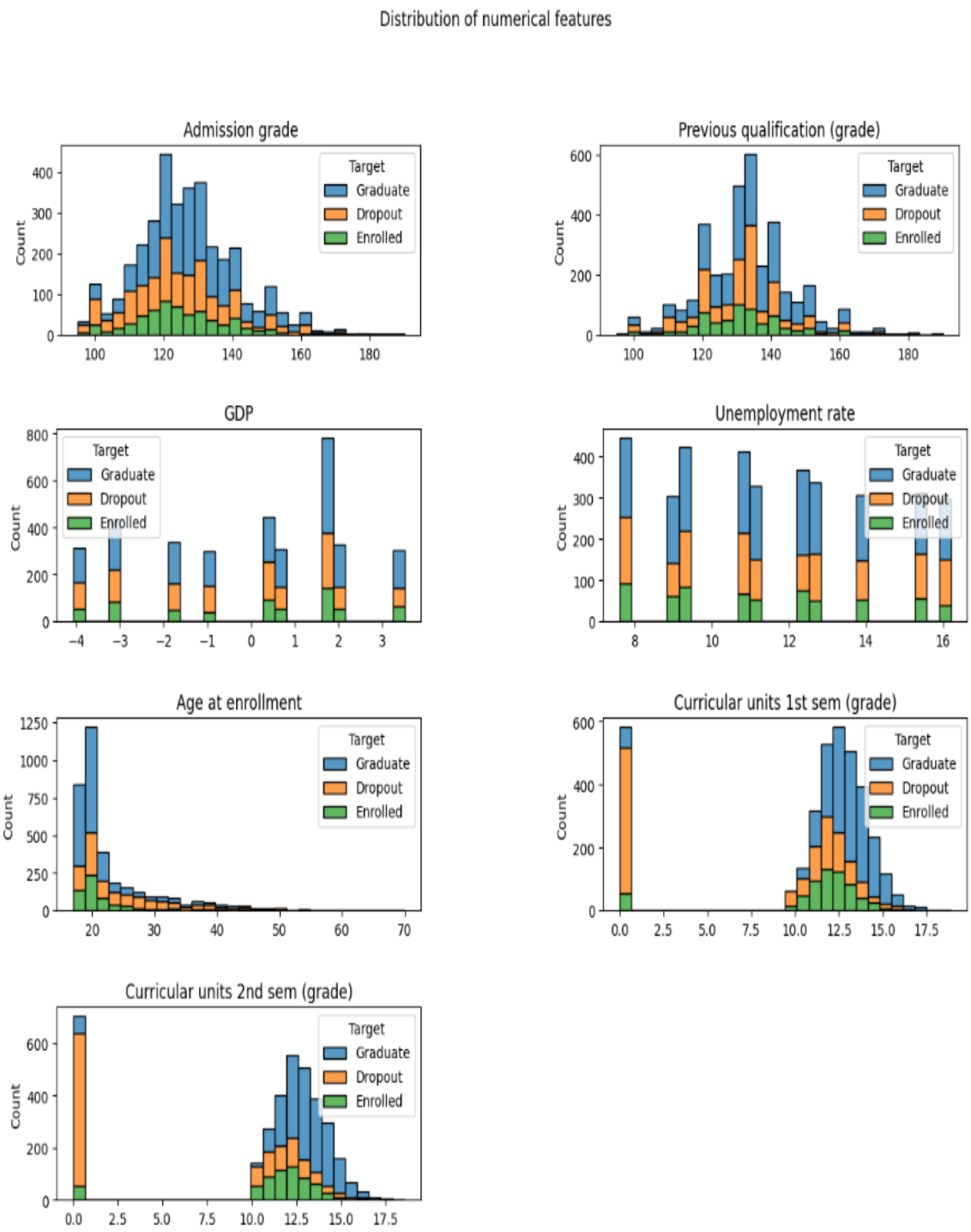


Hình 1.36: Thống kê điểm kì 2 của sinh viên theo từng nhãn

- Nhận xét: Điểm 0 ở 2 kỳ là lý do bỏ học cao

f) Tổng quan dữ liệu

```
fig = plt.figure(figsize=(14, 12))
for i, feature in enumerate(num_features):
    ax = plt.subplot(4, 2, i+1)
    sns.histplot(data=train, x=feature, hue='Target', multiple = 'stack', bins=28,  ax=ax)
    plt.xlabel('')
    plt.title(feature)
plt.subplots_adjust(wspace=0.5, hspace=0.5)
plt.suptitle("Distribution of numerical features")
plt.show()
```



Hình 1.37: Trực quan hóa các cột dữ liệu

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Pandas

Pandas là một thư viện Python cung cấp các cấu trúc dữ liệu nhanh, mạnh mẽ, linh hoạt và dễ hiểu. Tên thư viện được lấy từ "panel data" (dữ liệu bảng). Pandas được thiết kế để làm việc dễ dàng và trực quan với dữ liệu có cấu trúc (dạng bảng, đa chiều, có thể không đồng nhất) và dữ liệu chuỗi thời gian¹. Thư viện này thường được sử dụng để thao tác, phân tích và làm sạch dữ liệu. Pandas cung cấp nhiều cấu trúc dữ liệu và các phép toán hỗ trợ thao tác dữ liệu số và dữ liệu thời gian². Nó là một công cụ toàn diện để xử lý dữ liệu trong Python, phù hợp với nhiều nhiệm vụ phân tích và quản lý dữ liệu. Thư viện Pandas trong Python là một thư viện mã nguồn mở cung cấp hỗ trợ mạnh mẽ cho việc thao tác dữ liệu. Nó cũng là một bộ công cụ phân tích và xử lý dữ liệu mạnh mẽ của ngôn ngữ lập trình Python. Thư viện này được sử dụng rộng rãi trong cả nghiên cứu và phát triển các ứng dụng khoa học dữ liệu. Thư viện này sử dụng một cấu trúc dữ liệu độc đáo được gọi là Dataframe. Pandas cung cấp rất nhiều hàm để thao tác và làm việc trên cấu trúc dữ liệu này. Chính sự linh hoạt và hiệu quả đã làm cho Pandas được sử dụng rộng rãi.

Tại sao nên chọn pandas?

- Pandas phù hợp với nhiều loại dữ liệu khác nhau

- Dữ liệu dạng bảng với các cột được nhập không đồng đều, chẳng hạn như trong bảng SQL hoặc bảng tính Excel

- Dữ liệu chuỗi thời gian có thứ tự và không có thứ tự (không nhất thiết phải có tần số cố định)

- Dữ liệu ma trận tùy ý (được nhập đồng đều hoặc không đồng đều) với nhãn hàng và cột

- Bất kỳ dạng tập dữ liệu quan sát/thống kê nào khác. Dữ liệu thực tế không cần phải được gắn nhãn trong cấu trúc dữ liệu pandas

- Pandas được xây dựng trên NumPy. Hai cấu trúc dữ liệu chính của pandas, Series (1 chiều) và DataFrame (2 chiều), xử lý các trường hợp sử dụng điển hình nhất trong tài chính, thống kê, khoa học xã hội và nhiều lĩnh vực kỹ thuật

Ưu điểm của pandas:

- Dễ dàng xử lý dữ liệu bị mất mát, được biểu diễn dưới dạng NaN, trong dữ liệu dấu phẩy động cũng như dữ liệu dấu phẩy cố định theo ý muốn của người dùng: bỏ qua hoặc gán giá trị 0
- Có thể thay đổi kích thước: có thể chèn và xóa cột Khởi DataFrame và các đối tượng đa chiều
- Tự động và rõ ràng căn chỉnh dữ liệu: các đối tượng có thể được căn chỉnh rõ ràng với một tập hợp nhãn hoặc người dùng có thể bỏ qua các nhãn và để Series, DataFrame, v.v. tự động căn chỉnh dữ liệu cho bạn trong các phép tính
- Linh hoạt trong việc định hình lại và xoay trục tập dữ liệu
- Cắt lát dựa trên nhãn thông minh, lập chỉ mục nâng cao và chọn tập con của các tập dữ liệu lớn
- Hiệu suất tốt

Cài đặt thư viện Pandas:

- Sử dụng pip và gõ lệnh: `pip install pandas`
- Hoặc với Anaconda, sử dụng lệnh: `conda install pandas`
- Khai báo thư viện Pandas: `import pandas as pd`

2.1.1 Series

Series([data, index, dtype, name, copy, . . .])

Một Series là một mảng đa chiều tương tự như mảng Numpy hoặc một cột của bảng, nhưng nó bao gồm thêm một nhãn để đánh dấu bảng. Series có thể được khởi tạo thông qua NumPy, các kiểu Dict hoặc các kiểu dữ liệu vô hướng thông thường. Series có nhiều thuộc tính như index, array, value, dtype, v.v. Bạn có thể thực hiện chuyển đổi Series sang một dtype cụ thể, tạo bản sao của bảng, trả về giá trị boolean của một phần tử, chuyển đổi Series từ DatetimeIndex sang PeriodIndex, v.v.

2.1.2 DataFrame

DataFrame([data, index, columns, dtype, copy])

DataFrame là một cấu trúc dữ liệu hai chiều có nhãn với các cột và hàng giống như bảng tính hoặc bảng. Giống như Series, DataFrame có thể chứa bất kỳ loại dữ liệu nào. Một điều quan trọng cần nhấn mạnh là tất cả các cột trong dataframe đều là các Series của

Pandas. Vì vậy, DataFrame là sự kết hợp của nhiều Series hoạt động như các cột! DataFrame được sử dụng rộng rãi và là một trong những cấu trúc dữ liệu quan trọng nhất.

2.2 Matplotlib

Matplotlib là một thư viện đồ thị dành cho ngôn ngữ lập trình Python và mở rộng thư viện số học NumPy. Nó cung cấp API hướng đối tượng để nhúng đồ thị vào các ứng dụng sử dụng các bộ công cụ GUI đa năng như Tkinter, wxPython, Qt hoặc GTK. Matplotlib cho phép bạn tạo và hiển thị đồ thị, hình ảnh và các đồ họa khác, rất hữu ích cho việc trực quan hóa dữ liệu trong Python. Một phần quan trọng của Matplotlib là Pyplot, một mô-đun cung cấp các hàm đơn giản để thêm các thành phần đồ thị như đường thẳng, hình ảnh, văn bản, v.v. vào khung vẽ (trục). Matplotlib là một trong những thư viện Python phổ biến nhất được sử dụng để trực quan hóa dữ liệu. Nó là một thư viện đa nền tảng để tạo đồ thị 2D từ dữ liệu trong mảng. Matplotlib được viết bằng Python và sử dụng NumPy, phần mở rộng toán học của Python. Nó cung cấp API hướng đối tượng để nhúng đồ thị vào các ứng dụng và sử dụng các bộ công cụ GUI của Python như PyQt và WxPython hoặc Tkinter. Nó cũng có thể được sử dụng trong các trình thông dịch Python và IPython, Jupyter Notebooks và máy chủ web.

Matplotlib có một giao diện tên là Pylab, được thiết kế để giống với MATLAB - một ngôn ngữ lập trình độc quyền được phát triển bởi MathWorks. Matplotlib, cùng với NumPy, có thể được coi là phiên bản mã nguồn mở tương đương với MATLAB.

Matplotlib ban đầu được viết bởi John D. Hunter vào năm 2003. Phiên bản ổn định hiện tại là 2.2.0, được phát hành vào tháng 1 năm 2018.

2.2.1 Matplotlib được dùng để làm gì?

Để thực hiện các suy luận thống kê cần thiết, việc trực quan hóa dữ liệu là rất quan trọng và Matplotlib là một giải pháp như vậy dành cho người dùng Python. Đây là một thư viện đồ thị rất mạnh mẽ, hữu ích cho những người làm việc với Python và NumPy. Mô-đun được sử dụng nhiều nhất của Matplotlib là Pyplot, cung cấp giao diện tương tự như MATLAB nhưng thay vào đó, nó sử dụng Python và là mã nguồn mở.

Để cài đặt Matplotlib, nếu bạn có Anaconda, chỉ cần gõ `conda install matplotlib` hoặc sử dụng công cụ pip `pip install matplotlib`.

2.2.2 General Concept

Một hình vẽ trong Matplotlib có thể được phân loại thành nhiều phần như sau:
Hình vẽ: Nó giống như một cửa sổ chứa mọi thứ bạn sẽ vẽ trên đó.
Axes: Các thành phần chính của một hình vẽ là các trục (các khung nhỏ hơn để vẽ). Một hình vẽ có thể chứa một hoặc nhiều trục. Nói cách khác, hình vẽ chỉ là vùng chứa, các trục là nơi các hình vẽ thực tế được vẽ.
Axis: Chúng là các đối tượng giống như đường số và chịu trách nhiệm tạo ra các ranh giới của đồ thị.
Đối tượng vẽ (Artist): Mọi thứ bạn có thể thấy trên hình vẽ đều là đối tượng vẽ, chẳng hạn như đối tượng Văn bản, đối tượng Đường 2D, đối tượng Tập hợp. Hầu hết các Đối tượng vẽ đều được gắn với Trục.

2.2.3 Ưu điểm:

Matplotlib là một thư viện tương tự như GNUplot. Ưu điểm chính so với GNUplot là Matplotlib là một module của Python. Do sự phổ biến ngày càng tăng của Python, Matplotlib cũng nhận được sự chú ý tương tự.
Một lý do khác khiến Matplotlib được ưa chuộng là nó được coi là một sự thay thế hoàn hảo cho MATLAB, nếu được sử dụng kết hợp với Numpy và Scipy. Trong khi MATLAB đắt tiền và là mã nguồn đóng, Matplotlib là phần mềm miễn phí và mã nguồn mở. Nó cũng là một ngôn ngữ hướng đối tượng. Hơn nữa, nó có thể được sử dụng với các bộ công cụ GUI đa năng như wxPython, Qt và GTK+. Ngoài ra còn có một hàm "pylab", được thiết kế để giống với MATLAB. Điều này có thể giúp những người quen thuộc với MATLAB dễ dàng chuyển sang Matplotlib.

Matplotlib có thể được sử dụng để tạo ra các hình ảnh chất lượng cao cho nhiều định dạng in ấn và môi trường tương tác trên nhiều nền tảng.

Một tính năng khác của Matplotlib là đường cong học tập của nó, có nghĩa là người dùng thường tiến bộ nhanh chóng sau khi bắt đầu. Trang web chính thức cho biết: "matplotlib cố gắng làm cho những việc khó khăn, phức tạp trở nên dễ dàng nhất có thể. Bạn có thể tạo hình ảnh, biểu đồ tần số, biểu đồ phổ, biểu đồ cột, biểu đồ lỗi, biểu đồ phân tán, v.v., chỉ với một vài dòng mã."

Matplotlib có một số giao diện để tương tác với thư viện matplotlib: API hướng đối tượng, Giao diện kịch bản (pyplot), Giao diện MATLAB (pylab). Cả pyplot và pylab đều là các giao diện nhẹ, nhưng Pyplot cung cấp giao diện thủ tục cho các thư viện vẽ đồ thị hướng đối tượng trong matplotlib. Các lệnh vẽ đồ thị của nó được thiết kế tương tự như của Matlab cả về tên gọi và ý nghĩa đối số. Thiết kế này đã làm cho việc sử dụng pyplot dễ dàng và dễ hiểu hơn, vì vậy trong các bài viết về Matplotlib, tôi sẽ sử dụng giao diện pyplot thay vì hai giao diện còn lại. Nếu chúng ta muốn can thiệp sâu hơn, với nhiều tùy chỉnh hơn, API hướng đối tượng sẽ là lựa chọn đúng đắn.

Để cài đặt Matplotlib nếu bạn có Anaconda, chỉ cần gõ `conda install matplotlib` hoặc sử dụng công cụ `pip install matplotlib`.

`pip install matplotlib`

2.3 Numpy

NumPy (Numeric Python) là một thư viện Python mã nguồn mở được sử dụng trong hầu hết các lĩnh vực khoa học và kỹ thuật. Nó là tiêu chuẩn thực tế để làm việc với dữ liệu số trong Python và là tiêu chuẩn cốt lõi của hệ sinh thái Python và PyData. NumPy hỗ trợ mạnh mẽ việc tính toán với ma trận, vectơ và các hàm đại số tuyến tính cơ bản. Điều này làm cho nó trở thành một công cụ quan trọng trong việc triển khai các thuật toán Học máy. Nó cho phép làm việc hiệu quả với ma trận và mảng, đặc biệt là dữ liệu ma trận và mảng lớn, với tốc độ xử lý nhanh hơn nhiều lần so với chỉ sử dụng "Python cốt lõi".

2.3.1 Cài đặt

- pip install numpy

2.3.2 Các phép toán với Numpy

- Khai báo thư viện: import numpy as npInitialize array:
  + Khởi tạo một mảng một chiều với kiểu dữ liệu của các phần tử là số nguyên:

    arr = np.array([1,3,4,5,6], dtype = int)
  + Khởi tạo một mảng một chiều với kiểu dữ liệu của các phần tử là kiểu mặc định:

    arr = np.array([1,3,4,5,6])
  + Khởi tạo một mảng hai chiều:

    arr1 = np.array([(4,5,6), (1,2,3)])
  + Khởi tạo một mảng ba chiều:

```
arr2 = np.array(([(2,4,0,6), (4,7,5,6)],
                 [(0,3,2,1), (9,4,5,6)],
                 [(5,8,6,4), (1,4,6,8)]))
```

- Khởi tạo mảng bằng hàm tích hợp sẵn:
  + np.zeros
  + np.ones((2,3,4), dtype = int): 3-dimensional array with elements is 1 and size is 2x3x4.
  + np.arange(1,7,2): Array has elements from 1-6 and step is 2.
  + np.full((2,3),5): A 2-dimensional array with elements is 5 and size is 2x3.
  + np.eye(4, dtype=int): A unit matrix with dimensions of 4x4.
  + np.random.random((2,3)): A random matrix with size is 2x3.

2.3.3 Thao tác mảng

- dtype: Kiểu dữ liệu của phần tử trong mảng
- shape: Kích thước của mảng
- size: Số lượng phần tử trong mảng
- ndim: Kích thước của mảng

- Truy cập các phần tử trong mảng: Các phần tử trong mảng được đánh số từ 0
  + arr[i]: Truy cập vào phần tử thứ i của mảng một chiều.
  + arr1[i,j]: Truy cập vào hàng thứ i và cột thứ j của mảng hai chiều.
  + arr2[n,i,j]: Truy cập vào mảng 3 chiều có n chiều, i hàng và j cột.

+ arr[a:b]: Truy cập các phần tử từ a đến b-1 trong mảng một chiều.
+ arr1[:,:i]: Truy cập các phần tử từ cột 0 đến i-1.

- Chức năng thống kê:
  + arr.max() hoặc np.max(arr): Lấy giá trị lớn nhất trong mảng.
  + arr.min() hoặc np.min(arr): Lấy giá trị nhỏ nhất trong mảng.
  + arr.sum() hoặc np.sum(arr): Tính tổng phần tử trong mảng.
  + arr.mean() hoặc np.mean(arr): Tính giá trị trung bình của các phần tử trong mảng
  + np.median(arr): Lấy giá trị trung vị của mảng.

2.4 Scikit-learn

Scikit-learn là một thư viện học máy mã nguồn mở hỗ trợ học có giám sát và không giám sát. Nó cũng cung cấp nhiều công cụ để huấn luyện mô hình, tiền xử lý dữ liệu, lựa chọn mô hình, đánh giá mô hình và nhiều tiện ích khác.

- Các công cụ đơn giản và hiệu quả để phân tích dữ liệu dự đoán
- Dễ tiếp cận với mọi người và có thể tái sử dụng trong nhiều ngữ cảnh
- Được xây dựng trên NumPy, SciPy và matplotlib
- Mã nguồn mở, có thể sử dụng thương mại - Giấy phép BSD

2.4.1 Cài đặt

- pip install -U scikit-learn

2.4.2 Thao tác với scikit-learn

- Nhiệm vụ :
  + Phân loại:
    - Xác định đối tượng thuộc loại nào.
    - Ứng dụng: Phát hiện thư rác, nhận dạng hình ảnh.
    - Thuật toán: Gradient boosting, nearest neighbors, random forest, logistic regression và nhiều thuật toán khác.

  Hồi quy:
    - Dự đoán thuộc tính có giá trị liên tục gắn liền với một đối tượng.

- Ứng dụng: Phản ứng thuốc, giá cổ phiếu.
- Thuật toán: Gradient boosting, nearest neighbors, random forest, ridge và nhiều thuật toán khác.

+ Phân cụm:
  - Tự động nhóm các đối tượng tương tự thành các tập hợp.
  - Ứng dụng: Phân khúc khách hàng, nhóm kết quả thí nghiệm.
  - Thuật toán: k-Means, HDBSCAN, hierarchical clustering và nhiều thuật toán khác

- Chức năng tích hợp hỗ trợ xử lý dữ liệu và mô hình:

+ Giảm chiều dữ liệu:
  - Giảm số lượng biến ngẫu nhiên cần xem xét.
  - Ứng dụng: Trực quan hóa, tăng hiệu quả.
  - Thuật toán: PCA, feature selection, non-negative matrix factorization và nhiều thuật toán khác

+ Lựa chọn mô hình:
  - So sánh, xác thực và lựa chọn các tham số và mô hình.
  - Ứng dụng: Cải thiện độ chính xác thông qua việc điều chỉnh tham số.
  - Thuật toán: Grid search, cross validation, metrics, and more...

+ Tiền xử lý:
  - Trích xuất đặc trưng và chuẩn hóa.
  - Ứng dụng: Chuyển đổi dữ liệu đầu vào như văn bản để sử dụng với các thuật toán máy học.
  - Thuật toán: Preprocessing, feature extraction, và các thuật toán khác
  -

CHƯƠNG 3: GIẢI PHÁP

3.1. Lý thuyết

3.1.1 Logistic Regression

- Hồi quy logistic là một mô hình thống kê được sử dụng cho các bài toán phân loại, dự đoán xác suất một trường hợp thuộc về một trong số các lớp. Đây là một thuật toán học máy có giám sát.
- Các loại hồi quy logistic:
  + Nhị phân: Trong hồi quy Logistic nhị phân, biến phụ thuộc chỉ có thể có hai loại, chẳng hạn như 0 hoặc 1, Đạt hoặc Không đạt, v.v..
  + Đa biến: Trong hồi quy Logistic đa biến, có thể có 3 hoặc nhiều hơn các loại không được sắp xếp khác nhau của biến phụ thuộc, chẳng hạn như "mèo", "chó" hoặc "cừu".
  + Thứ tự: Trong hồi quy Logistic thứ tự, có thể có 3 hoặc nhiều hơn các loại biến phụ thuộc được sắp xếp theo thứ tự, chẳng hạn như "thấp", "trung bình" hoặc "cao".
- Khai báo mô hình:
  + Bước 1: khai báo thư viện

    ```
    from sklearn.linear_model import LogisticRegression
    ```

  + Bước 2: gán biến

    ```
    logistic_model = LogisticRegression()
    ```

- Tham số:
  + penalty{'l1', 'l2', 'elasticnet', None}, default='l2'
    - None: không thêm hình phạt nào
    - 'l2': thêm hình phạt L2 và đây là lựa chọn mặc định
    - 'l1': thêm hình phạt L1
    - 'elasticnet': cả hình phạt L1 và L2 đều được thêm vào
  + max_inter: Tổng số lần dữ liệu được khớp với mô hình.

3.1.2 Support Vector Machine

- Support Vector Machine (SVM) là một thuật toán học máy mạnh mẽ được sử dụng rộng rãi cho cả phân loại tuyến tính và phi tuyến tính, cũng như các nhiệm vụ hồi quy và phát hiện ngoại lệ. SVM có khả năng thích ứng cao, làm cho chúng phù hợp với nhiều ứng dụng khác nhau như phân loại văn bản, phân loại hình ảnh, phát hiện thư rác, nhận dạng chữ viết tay, phân tích biểu hiện gen, phát hiện khuôn mặt và phát hiện bất thường.
- Các thuật toán SVM đặc biệt hiệu quả vì chúng tập trung vào việc tìm siêu mặt phẳng phân tách tối đa giữa các lớp khác nhau trong đặc trưng mục tiêu, giúp chúng mạnh mẽ cho cả phân loại nhị phân và đa lớp.
- Khai báo mô hình:
  + Bước 1: khai báo thư viện

```
from sklearn.svm import SVC
```

  + Bước 2: gán biến

```
svm_model = SVC()
```

- Tham số:
  + max_inter: Tổng số lần dữ liệu được khớp với mô hình.

3.1.3. Decision Tree

- Cây quyết định giúp chúng ta đưa ra quyết định bằng cách vạch ra các lựa chọn khác nhau và kết quả có thể xảy ra của chúng. Nó được sử dụng trong máy học cho các tác vụ như phân loại và dự đoán.
- Cây quyết định giúp chúng ta đưa ra quyết định bằng cách thể hiện các lựa chọn khác nhau và mối quan hệ giữa chúng. Nó có cấu trúc dạng cây, bắt đầu từ một câu hỏi chính được gọi là nút gốc, đại diện cho toàn bộ tập dữ liệu. Từ đó, cây phân nhánh thành các khả năng khác nhau dựa trên các đặc điểm trong dữ liệu.
  - Nút gốc: Điểm khởi đầu đại diện cho toàn bộ tập dữ liệu.
  - Các nhánh: Các đường nối các nút thể hiện luồng thông tin từ quyết định này sang quyết định khác.
  - Các nút nội bộ: Là những điểm mà tại đó các quyết định được đưa ra dựa trên các đặc điểm dữ liệu.
  - Các nút lá**:** Là điểm cuối của cây, nơi đưa ra quyết định hoặc dự đoán cuối cùng.

- Khai báo mô hình:

+ Bước 1: khai báo thư viện

```
from sklearn.tree import DecisionTreeClassifier
```

+ Bước 2: gán biến

```
tree_model = DecisionTreeClassifier
()
```

- Tham số:
  + max_depth: Độ sâu tối đa của cây
  + min_samples_split: Số mẫu tối thiểu ở 1 nút để nút đó phân chia tiếp
  + min_samples_leaf: Số mẫu tối thiểu phải có ở mỗi lá sau khi tách

3.1.4. Random Forest

- Random Forest là một bộ phân loại học theo phương thức ensemble learning, tức là nó xây dựng nhiều mô hình đơn lẻ (các cây quyết định) và kết hợp chúng lại để đưa ra dự đoán chính xác hơn.
  - Cây quyết định (Decision Trees): Mỗi cây quyết định trong Random Forest là một cây riêng biệt được xây dựng bằng cách sử dụng một tập con ngẫu nhiên của dữ liệu và các đặc trưng.
  - Quy trình Ensemble: Sau khi các cây quyết định được xây dựng, kết quả của mỗi cây sẽ được lấy (dựa trên đa số phiếu bầu cho phân loại hoặc giá trị trung bình cho hồi quy) để đưa ra dự đoán cuối cùng.
- Cách thức hoạt động của Random Forest
  - Tạo ra các cây quyết định:
  - Tạo ra một tập hợp các mẫu ngẫu nhiên từ tập huấn luyện bằng cách sử dụng Bootstrap sampling (hay còn gọi là sampling với thay thế).
  - Tại mỗi nút trong cây, thay vì thử tất cả các đặc trưng để chia, thuật toán chỉ chọn một tập con ngẫu nhiên của các đặc trưng để tạo sự đa dạng giữa các cây.
- Dự đoán từ mỗi cây:
  - Phân loại: Mỗi cây đưa ra một dự đoán phân loại. Random Forest sẽ chọn kết quả dự đoán mà cây quyết định chiếm số phiếu bầu cao nhất (phương pháp "majority vote").

- Hồi quy: Kết quả dự đoán là giá trị trung bình của tất cả các cây trong rừng.

- Khai báo mô hình:
  + Bước 1: khai báo thư viện

```
from sklearn.ensemble import RandomForestClassifier
```

  + Bước 2: gán biến

```
rf_model = RandomForestClassifier()
```

- Tham số:
  + n_estimators: Số lượng cây trong mô hình.
  + max_depth: Độ sâu tối đa của cây
  + min_samples_split: Số mẫu tối thiểu ở 1 nút để nút đó phân chia tiếp
  + min_samples_leaf: Số mẫu tối thiểu phải có ở mỗi lá sau khi tách

3.1.5. Gradient Boosting

- Gradient Boosting Regressor là một mô hình học máy mạnh mẽ thuộc họ các thuật toán boosting, được sử dụng phổ biến trong các bài toán hồi quy. Gradient Boosting là một phương pháp học máy dựa trên nguyên lý kết hợp nhiều mô hình yếu (thường là cây quyết định) để tạo ra một mô hình mạnh mẽ hơn.
- Gradient Boosting xây dựng mô hình theo cách tuần tự, mỗi mô hình mới được huấn luyện để sửa chữa những sai số (residuals) của mô hình trước đó.
- Mỗi cây quyết định trong quá trình huấn luyện sẽ học cách dự đoán sự sai lệch giữa giá trị thực tế và dự đoán của mô hình trước đó. Sau khi học được từ sai số, mô hình tiếp theo sẽ kết hợp với mô hình hiện tại để cải thiện dự đoán.
- Mỗi mô hình được thêm vào là một mô hình yếu, có thể không chính xác lắm, nhưng khi kết hợp với các mô hình trước đó, nó tạo ra một mô hình mạnh mẽ hơn..
- Khai báo mô hình:
  + Bước 1: khai báo thư viện

```
from sklearn.ensemble import GradientBoostingClassifier
```

  + Bước 2: gán biến

```
gb_model = GradientBoostingClassifier()
```

- Tham số:
  + n_estimators: Số lượng cây trong mô hình.

3.1.6 Đánh giá

3.1.6.1 Accuracy

Hàm accuracy tính toán độ chính xác, hoặc là tỷ lệ (mặc định) hoặc là số lượng (normalize=False) các dự đoán chính xác.

Trong phân loại đa nhãn, hàm này trả về độ chính xác của tập con. Nếu toàn bộ tập hợp các nhãn được dự đoán cho một mẫu hoàn toàn khớp với tập hợp nhãn thực, thì độ chính xác của tập con là 1.0; ngược lại là 0.0.

Nếu y^i là giá trị được dự đoán của mẫu thứ i và yi là giá trị thực tương ứng, thì tỷ lệ các dự đoán chính xác trên nsamples được định nghĩa là

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

Hình 3.1: Công thức tính Accuracy

3.1.6.2 Precision

Độ chính xác là tỷ lệ tp / (tp + fp), trong đó tp là số lượng dương tính thật và fp là số lượng dương tính giả. Về mặt trực quan, độ chính xác thể hiện khả năng của bộ phân loại không gán nhãn là dương tính cho một mẫu thực tế là âm tính.

Giá trị tốt nhất là 1 và giá trị tệ nhất là 0.

$$Precision = \frac{TP}{TP + FP}$$

Hình 3.2: Công thức tính Precision

3.1.6.3 Recall

Độ nhạy (recall) là tỷ lệ tp / (tp + fn), trong đó tp là số lượng dương tính thật và fn là số lượng âm tính giả. Về mặt trực quan, độ nhạy thể hiện khả năng của bộ phân loại trong việc tìm ra tất cả các mẫu dương tính.

Giá trị tốt nhất là 1 và giá trị tệ nhất là 0.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Hình 3.3: Công thức tính Recall

3.1.6.4. F1-score

F1 kết hợp độ chính xác và độ thu hồi bằng cách sử dụng trung bình điều hòa:

$$F_1 \text{ Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Hình 3.4: Công thức tính F1-score

Do đó nó thể hiện một cách đối xứng cả độ chính xác và độ thu hồi trong cùng một chỉ số.
Giá trị tốt nhất là 1 và giá trị tệ nhất là 0.

3.2. Code

- Chia tập dữ liệu thành 2 tập: X (dùng cho giai đoạn huấn luyện), y (dùng để kiểm tra mô hình sau khi huấn luyện):

```
y = df['Target']
X = df.drop('Target', axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    shuffle=True,
                                                    stratify=y,
                                                    random_state=42)
```

Hình 3.5: Chia tập dữ liệu thành tập train và test

+ Kiểm tra tính khớp kích thước giữa dữ liệu và nhãn trước khi train/test mô hình:

```
assert X_train.shape[0] == y_train.shape[0] and X_test.shape[0] == y_test.shape[0] and X_train.shape[1] == X_test.shape[1]
```

Hình 3.6: Khớp dữ liệu ở hai tập train và test

+ X_train chưa những thông tin:

X_train

	Marital status	Application mode	Application order	Course	Daytime/evening attendance\t	Previous qualification	Previous qualification (grade)	Nacionality	Mother's qualification	Father's qualification	...	Curricular units 1st sem (without evaluations)	Curricular units 2nd sem (credited)	Curricular units 2nd sem (enrolled)	Curricular units 2nd sem (evaluations)	Curricular units 2nd sem (approved)	Curricular units 2nd sem (grade)	Curricular units 2nd sem (without evaluations)	Unemployment rate	Inflation rate	GDP
2283	1	43	1	9238	1	39	120.0	1	2	1	...	0	6	10	13	10	11.800000	0	13.9	-0.3	0.79
3874	1	39	1	9130	1	1	133.1	1	1	1	...	1	0	5	5	0	0.000000	5	12.7	3.7	-1.70
2281	1	44	1	9003	1	39	140.0	1	1	38	...	0	0	6	15	4	10.500000	0	15.5	2.8	-4.06
817	1	16	1	9070	1	1	125.0	1	1	1	...	0	0	6	6	6	12.833333	0	12.4	0.5	1.79
404	1	17	1	9500	1	1	142.0	1	37	37	...	0	0	8	8	7	14.731429	0	10.8	1.4	1.74
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2753	1	43	1	9991	0	1	120.0	1	38	38	...	0	5	10	10	10	12.800000	0	10.8	1.4	1.74
33	1	18	1	8014	0	1	138.0	1	19	1	...	0	0	6	8	6	14.375000	0	12.4	0.5	1.79
375	2	51	1	8014	0	19	133.1	1	37	37	...	0	5	11	14	10	12.800000	0	11.1	0.6	2.02
967	1	39	1	9003	1	12	133.1	1	37	19	...	0	0	6	6	0	0.000000	0	13.9	-0.3	0.79
1497	1	17	2	9670	1	1	110.0	1	1	1	...	0	0	6	8	5	13.000000	0	15.5	2.8	-4.06

3539 rows × 36 columns

Hình 3.7: Thông tin tập X_train

+ Lưu lại tập train và tập test:

```
train = X_train
train['Target'] = y_train

test = X_test
test['Target'] = y_test

train.to_csv("/content/train.csv", index=False)
test.to_csv("/content/test.csv", index=False)
```

Hình 3.8: Lưu lại tập train và tập test

+ Đọc 5 bản ghi đầu tập train:

```
train = pd.read_csv("/content/train.csv")
train.head(5)
```

	Marital status	Application mode	Application order	Course	Daytime/evening attendance\t	Previous qualification	Previous qualification (grade)	Nacionality	Mother's qualification	Father's qualification	...	Curricular units 2nd sem (credited)	Curricular units 2nd sem (enrolled)	Curricular units 2nd sem (evaluations)	Curricular units 2nd sem (approved)	Curricular units 2nd sem (grade)	Curricular units 2nd sem (without evaluations)	Unemployment rate	Inflation rate	GDP	Target
0	1	43	1	9238	1	39	120.0	1	2	1	...	6	10	13	10	11.800000	0	13.9	-0.3	0.79	Graduate
1	1	39	1	9130	1	1	133.1	1	1	1	...	0	5	5	0	0.000000	5	12.7	3.7	-1.70	Dropout
2	1	44	1	9003	1	39	140.0	1	1	38	...	0	6	15	4	10.500000	0	15.5	2.8	-4.06	Enrolled
3	1	16	1	9070	1	1	125.0	1	1	1	...	0	6	6	6	12.833333	0	12.4	0.5	1.79	Graduate
4	1	17	1	9500	1	1	142.0	1	37	37	...	0	8	8	7	14.731429	0	10.8	1.4	1.74	Graduate

5 rows × 37 columns

Hình 3.9: Đọc 5 bản ghi đầu tập train

+ Chuyển các cột sau khi từ số sang chữ và đồng nhất dữ liệu

- Cột father_qual và mother_qual

```
qual_mapping = {1: "highschool",
                38:"pre highschool",
                37:"pre highschool",
                19:"highschool",
                11:"pre highschool",
                3:"graduate",
                2:"graduate",
                34:"unknown",
                4:"masters",
                27:"pre highschool",
                12:"unknown",
                39:"graduate",
                42:"masters",
                5:"masters",
                40:"graduate",
                6:"unknown",
                36:"less than 4",
                44:"masters",
                41:"graduate",
                29:"pre highschool",
                30:"pre highschool",
                9:"highschool",
                10:"highschool",
                35:"less than 4",
                14:"highschool",
                43:"masters",
                26:"pre highschool",
                25:"pre highschool",
                18:"graduate",
                22:"masters",
                31:"graduate",
                20:"highschool"}
```

```
train["father_qual"] = train["Father\'s qualification"].map(qual_mapping)
train["mother_qual"] = train["Mother\'s qualification"].map(qual_mapping)

test["father_qual"] = test["Father\'s qualification"].map(qual_mapping)
test["mother_qual"] = test["Mother\'s qualification"].map(qual_mapping)
```

Hình 3.10: Đồng nhất cột father_qual/mother_ qual và lưu lại cho tập train và test

- Cột father_occ và mother_occ

```
occ_mapping = {0: "student",
               1: "managerial",
               2: "professional",
               3: "technical",
               4: "professional",
               5: "service",
               6: "agriculture",
               7: "craftsmen",
               8: "factory",
               9: "elementary",
               10: "armed forces",
               90: "unknown",
               99: "unknown",
               101: "armed forces",
               102: "armed forces",
               103: "armed forces",
               112: "managerial",
               114: "managerial",
               121: "professional",
               122: "professional",
               123: "professional",
               124: "professional",
               131: "technical",
               132: "technical",
               134: "technical",
               135: "technical",
               141: "clerical",
               143: "technical",
               144: "clerical",
               151: "service",
               152: "service",
               153: "service",
               154: "service",
               161: "agriculture",
               163: "agriculture",
               171: "craftsmen",
               172: "craftsmen",
               174: "craftsmen",
               175: "craftsmen",
               181: "factory",
               182: "factory",
               183: "factory",
               192: "elementary",
               193: "elementary",
               194: "elementary",
               195: "elementary"}
```

```
train["father_occ"] = train["Father\'s occupation"].map(occ_mapping)
train["mother_occ"] = train["Mother\'s occupation"].map(occ_mapping)

test["father_occ"] = test["Father\'s occupation"].map(occ_mapping)
test["mother_occ"] = test["Mother\'s occupation"].map(occ_mapping)
```

Hình 3.11: Đồng nhất cột father_occ/mother_occ và lưu lại cho tập train và test

- Cột prev_qual

```
prev_qual_mapping = {1:'secondary_school',
                     2: 'graduate',
                     3: 'graduate',
                     4: 'masters',
                     5: 'doctorate',
                     6: 'unknown',
                     9: 'highschool',
                     10: 'highschool',
                     12: 'unknown',
                     14: 'highschool',
                     15: 'pre-highschool',
                     19: 'highschool',
                     38: 'pre-highschool',
                     39: 'graduate',
                     40: 'graduate',
                     42: 'masters',
                     43: 'masters'}
```

```
train["previous_qual"] = train["Previous qualification"].map(prev_qual_mapping)

test["previous_qual"] = test["Previous qualification"].map(prev_qual_mapping)
```

Hình 3.12: Đồng nhất cột prev_qual và lưu lại cho tập train

- Đổi tên cột Daytime/evening đổi tên thành Attendance_mode và chuyển dữ liệu số thành chữ:

```
train.rename(columns={"Daytime/evening attendance\t":"Attendance_mode"}, inplace=True)
test.rename(columns={"Daytime/evening attendance\t":"Attendance_mode"}, inplace=True)

mode = {0: "Evening",
        1: "Daytime"}

train["Attendance_mode"] = train["Attendance_mode"].map(mode)
test["Attendance_mode"] = test["Attendance_mode"].map(mode)
```

Hình 3.13: Xử lý dữ liệu cột Daytime/evening và lưu lại cho tập train và test

+ Lưu lại dữ liệu đây là dữ liệu sẽ dung để huấn luyện mô hình

```
train.to_csv("/content/train_final.csv", index=False)
test.to_csv("/content/test_final.csv", index=False)
```

Hình 3.14: Lưu lại tập train và test dùng để huấn luyện mô hình

- Khai báo các thư viện trước khi huấn luyện

```
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler, OrdinalEncoder

from sklearn.model_selection import StratifiedShuffleSplit, cross_validate, GridSearchCV

from sklearn.dummy import DummyClassifier
from sklearn.linear_model import LogisticRegression, RidgeClassifier
from sklearn.svm import LinearSVC
from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, HistGradientBoostingClassifier

from sklearn.metrics import classification_report, confusion_matrix, f1_score,recall_score,accuracy_score

from sklearn import set_config

from imblearn.pipeline import Pipeline as imb_pipeline
from imblearn.over_sampling import SMOTENC
```

Hình 3.15: Khai báo thư viện

- Đọc dữ liệu tập train và test

```
train = pd.read_csv("/content/train_final.csv")
test =  pd.read_csv("/content/test_final.csv")
```

Hình 3.16: Đọc dữ liệu

+ Dữ liệu trong tập train:

```
train.columns
```

```
Index(['Marital status', 'Application mode', 'Application order', 'Course',
       'Attendance_mode', 'Previous qualification',
       'Previous qualification (grade)', 'Nacionality',
       'Mother's qualification', 'Father's qualification',
       'Mother's occupation', 'Father's occupation', 'Admission grade',
       'Displaced', 'Educational special needs', 'Debtor',
       'Tuition fees up to date', 'Gender', 'Scholarship holder',
       'Age at enrollment', 'International',
       'Curricular units 1st sem (credited)',
       'Curricular units 1st sem (enrolled)',
       'Curricular units 1st sem (evaluations)',
       'Curricular units 1st sem (approved)',
       'Curricular units 1st sem (grade)',
       'Curricular units 1st sem (without evaluations)',
       'Curricular units 2nd sem (credited)',
       'Curricular units 2nd sem (enrolled)',
       'Curricular units 2nd sem (evaluations)',
       'Curricular units 2nd sem (approved)',
       'Curricular units 2nd sem (grade)',
       'Curricular units 2nd sem (without evaluations)', 'Unemployment rate',
       'Inflation rate', 'GDP', 'Target', 'father_qual', 'mother_qual',
       'father_occ', 'mother_occ', 'previous_qual'],
      dtype='object')
```

Hình 3.17: Dữ liệu trong tập train

+ Các cột giữ lại để huấn luyện:

```
features_to_include = ['Marital status', 'Application mode', 'Application order', 'Course',
       'Attendance_mode', 'Previous qualification (grade)', 'Admission grade', 'Displaced',
       'Educational special needs', 'Debtor', 'Tuition fees up to date', 'Gender', 'Scholarship holder',
       'Age at enrollment', 'International', 'father_occ', 'mother_occ', 'father_qual', 'mother_qual', 'previous_qual',
        'Target']
```

Hình 3.18: Các cột dùng để train

+ Lấy các dữ liệu ở các cột giữ lại làm dữ liệu ở tập train và tập test:

```
train_df = train[features_to_include]
test_df = test[features_to_include]
```

Hình 3.18: Lấy các cột được giữ lại làm dữ liệu cho 2 tập train và test

+ Chia tập train thành X_train, X test và tập test thành y_train và y_test:

```
y_train = train_df["Target"]
X_train = train_df.drop("Target", axis=1)

y_test = test_df["Target"]
X_test = test_df.drop("Target", axis=1)
```

Hình 3.19: Chia tập dữ liệu

+ Lấy các cột có dữ liệu số và dữ liệu dạng chữ:

```
cat_features = ['Course', 'previous_qual', 'Debtor', 'Gender', 'Scholarship holder', 'father_occ', 'mother_occ', 'father_qual', 'mother_qual', 'Marital status', 'Application mode', 'Attendance_mode', 'International', 'Displaced']
num_features = ['Previous qualification (grade)', 'Admission grade', 'Age at enrollment']
ordinal_features = ['Application order']
```

Hình 3.20: Lấy các cột dạng chữ và dạng số trong tập train

+ Chuẩn hóa các cột dữ liệu với OneHotEncoder cho dữ liệu dạng số, Ordinal Encoder và StandardScaler đối với dữ liệu số:

```
select_cat_features = ColumnTransformer([('select_cat', 'passthrough', cat_features)])
cat_transformers = Pipeline([('selector', select_cat_features),
                             ('onehot', OneHotEncoder(handle_unknown='ignore')),
                             ])

select_ord_features = ColumnTransformer([('select_cat', 'passthrough', ordinal_features)])
ordinal_transformers = Pipeline([('selector', select_ord_features),
                             ('ordinal_enc', OrdinalEncoder(handle_unknown='use_encoded_value', unknown_value=-1)),
                             ])

select_num_features = ColumnTransformer([('select_num', 'passthrough', num_features)])
num_transformers = Pipeline([('selector', select_num_features),
                             ('scaler', StandardScaler()),
                             ])

preprocess_pipe = FeatureUnion([('cat', cat_transformers),
                                ('ord', ordinal_transformers),
                                ('num', num_transformers),
                             ])
```

Hình 3.21: Chuẩn hóa dữ liệu

+ Tạo một tham số để kiểm chứng chéo và tìm siêu tham số tối ưu:

```
cv = StratifiedShuffleSplit(n_splits = 5, test_size=0.2, random_state=32)
```

Hình 3.22: Tạo tham số để kiểm chứng chéo và tìm siêu tham số tối ưu

- Huấn luyện các mô hình:
  + Logistic Regression:
    - Khởi tạo mô hình:

```
logistic_clf = Pipeline([
    ("preprocess", preprocess_pipe),
    ("model", LogisticRegression(
        penalty="elasticnet",
        solver="saga",
        l1_ratio=0,
        max_iter=1000
    ))
])
```

Hình 3.23: Khởi tạo mô hình Logistic Regression

    - Cross-validation mô hình:

```
cv_results = cross_validate(
    logistic_clf,
    X_train, y_train,
    cv=cv,
    scoring="f1_micro",
    return_train_score=True
)

print("Train F1:", cv_results["train_score"].mean())
print("Validation F1:", cv_results["test_score"].mean())

fitted_model = logistic_clf.fit(X_train, y_train)
```

```
Train F1: 0.6580007064641469
Validation F1: 0.6324858757062147
```

Hình 3.24: Kiểm chứng chéo cho mô hình Logistic Regression

    - Tìm kiếm các siêu tham số tối ưu cho mô hình

```
param_grid = {"model__C": [0.01, 0.1, 1, 10, 100],
              "model__l1_ratio": np.linspace(0, 1, 11)}

search = GridSearchCV(
    estimator=logistic_clf,
    param_grid=param_grid,
    cv=cv,
    scoring= "f1_micro",
    return_train_score=True
)

search.fit(X_train, y_train)

print("Best parameters: ", search.best_params_)
best_model = search.best_estimator_
```

```
Best parameters:  {'model__C': 1, 'model__l1_ratio': np.float64(1.0)}
```

Hình 3.25: Tìm các siêu tham số tối ưu cho mô hình Logistic Regression

- Kết quả dự đoán trên tập test

```
print("F1 on test set:", f1_score(y_test, y_pred, average="micro"))

y_pred = best_model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
F1 on test set: 0.5954802259887005
              precision    recall  f1-score   support

     Dropout       0.56      0.58      0.57       284
    Enrolled       0.32      0.11      0.17       159
    Graduate       0.64      0.78      0.70       442

    accuracy                           0.60       885
   macro avg       0.51      0.49      0.48       885
weighted avg       0.56      0.60      0.56       885
```

Hình 3.26: Kết quả dự đoán trên tập test của mô hình Logistic Regression

+ Support Vector Machine:

- Khởi tạo mô hình:

```
svm_clf = Pipeline(steps=[
    ("preprocess", preprocess_pipe),
    ("model", LinearSVC(random_state=32))
])
```

Hình 3.27: Khởi tạo mô hình Support Vector Machine

- Cross-validation mô hình:

```
cv_results = cross_validate(
    svm_clf,
    X_train,
    y_train,
    cv=cv,
    scoring="f1_micro",
    return_train_score=True
)
print("Train F1:", cv_results["train_score"].mean())
print("Validation F1:", cv_results["test_score"].mean())

fitted_model = svm_clf.fit(X_train, y_train)
```

```
Train F1: 0.6553867891204521
Validation F1: 0.6375706214689265
```

Hình 3.28: Kiểm chứng chéo cho mô hình Support Vector Machine

- Tìm kiếm các siêu tham số tối ưu cho mô hình

```
param_grid = {"model__penalty": ["l1", "l2"],
              "model__C": [0.001, 0.01, 0.1, 1, 10, 100]}

search = GridSearchCV(
    estimator=svm_clf,
    param_grid=param_grid,
    cv=cv,
    scoring= "f1_micro",
    return_train_score=True
)

search.fit(X_train, y_train)

print("Best parameters: ", search.best_params_)
best_model = search.best_estimator_

  Best parameters:  {'model__C': 0.1, 'model__penalty': 'l2'}
```

Hình 3.29: Tìm các siêu tham số tối ưu cho mô hình Support Vector Machine

- Kết quả dự đoán trên tập test

```
print("F1 on test set:", f1_score(y_test, y_pred, average="micro"))

y_pred = best_model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
F1 on test set: 0.5988700564971752
              precision    recall  f1-score   support

     Dropout       0.56      0.60      0.58       284
    Enrolled       0.23      0.04      0.07       159
    Graduate       0.64      0.80      0.71       442

    accuracy                           0.60       885
   macro avg       0.48      0.48      0.45       885
weighted avg       0.54      0.60      0.55       885
```

Hình 3.30: Kết quả dự đoán trên tập test của mô hình Support Vector Machine

+ Decision Tree:

- Khởi tạo mô hình:

```
tree_clf = Pipeline(steps=[
    ("preprocess", preprocess_pipe),
    ("model", DecisionTreeClassifier(random_state=32))
])
```

Hình 3.31: Khởi tạo mô hình Decision Tree

- Cross-validation mô hình:

```
cv_results = cross_validate(
    tree_clf,
    X_train,
    y_train,
    cv=cv,
    scoring="f1_micro",
    return_train_score=True
)
print("Train F1:", cv_results["train_score"].mean())
print("Validation F1:", cv_results["test_score"].mean())

fitted_model = tree_clf.fit(X_train, y_train)
```

```
Train F1: 1.0
Validation F1: 0.501412429378531
```

Hình 3.32: Kiểm chứng chéo cho mô hình Decision Tree

- Tìm kiếm các siêu tham số tối ưu cho mô hình

```
param_grid = {"model__max_depth": [2, 3, 4, 5, 10],
              "model__min_samples_split":[2, 4, 6, 8, 10],
              "model__min_samples_leaf": [1, 2, 3, 6, 7, 8]}

search = GridSearchCV(
    estimator=tree_clf,
    param_grid=param_grid,
    cv=cv,
    scoring= "f1_micro",
    return_train_score=True
)

search.fit(X_train, y_train)

print("Best parameters: ", search.best_params_)
best_model = search.best_estimator_
```

```
Best parameters:  {'model__max_depth': 5, 'model__min_samples_leaf': 6, 'model__min_samples_split': 2}
```

Hình 3.33: Tìm các siêu tham số tối ưu cho mô hình Decision Tree

- Kết quả dự đoán trên tập test

```
F1 on test set: 0.5988700564971752
              precision    recall  f1-score   support

     Dropout       0.50      0.70      0.58       284
    Enrolled       0.00      0.00      0.00       159
    Graduate       0.65      0.72      0.69       442

    accuracy                           0.58       885
   macro avg       0.38      0.47      0.42       885
weighted avg       0.49      0.58      0.53       885
```

Hình 3.34: Kết quả dự đoán trên tập test của mô hình Decision Tree

+ Random Forest:

  - Khởi tạo mô hình:

```
rf_clf = Pipeline(steps=[
    ("preprocess", preprocess_pipe),
    ("model", RandomForestClassifier(random_state=32))
])
```

Hình 3.35: Khởi tạo mô hình Random Forest

  - Cross-validation mô hình:

```
cv_results = cross_validate(
    rf_clf,
    X_train,
    y_train,
    cv=cv,
    scoring="f1_micro",
    return_train_score=True
)
print("Train F1:", cv_results["train_score"].mean())
print("Validation F1:", cv_results["test_score"].mean())

fitted_model = rf_clf.fit(X_train, y_train)
```

```
Train F1: 1.0
Validation F1: 0.6104519774011299
```

Hình 3.36: Kiểm chứng chéo cho mô hình Random Forest

  - Tìm kiếm các siêu tham số tối ưu cho mô hình

```
param_grid = {"model__n_estimators": [125, 150, 175],
              "model__max_depth": [3, 4, 5, 10, 15],
              "model__min_samples_split":[2, 4, 6, 8, 10],
              "model__min_samples_leaf": [1, 2, 3, 6, 7, 8]}

search = GridSearchCV(
    estimator=rf_clf,
    param_grid=param_grid,
    cv=cv,
    scoring= "f1_micro",
    return_train_score=True
)

search.fit(X_train, y_train)

print("Best parameters: ", search.best_params_)
best_model = search.best_estimator_
```

```
Best parameters:  {'model__max_depth': 15, 'model__min_samples_leaf': 1, 'model__min_samples_split': 4, 'model__n_estimators': 175}
```

Hình 3.37: Tìm các siêu tham số tối ưu cho mô hình Random Forest

  - Kết quả dự đoán trên tập test

```
print("F1 on test set:", f1_score(y_test, y_pred, average="micro"

y_pred = best_model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
F1 on test set: 0.5954802259887005
              precision    recall  f1-score   support

     Dropout       0.55      0.58      0.57       284
    Enrolled       0.28      0.03      0.06       159
    Graduate       0.63      0.81      0.71       442

    accuracy                           0.60       885
   macro avg       0.49      0.47      0.44       885
weighted avg       0.54      0.60      0.54       885
```

Hình 3.38: Kết quả dự đoán trên tập test của mô hình Random Forest

+ Gradient Boosting:

  - Khởi tạo mô hình:

```
gboost_clf = Pipeline(steps=[
    ("preprocess", preprocess_pipe),
    ("model", GradientBoostingClassifier(random_state=32))
])
```

Hình 3.39: Khởi tạo mô hình Gradient Boosting

  - Cross-validation mô hình:

```
cv_results = cross_validate(
    gboost_clf,
    X_train,
    y_train,
    cv=cv,
    scoring="f1_micro",
    return_train_score=True
)
print("Train F1:", cv_results["train_score"].mean())
print("Validation F1:", cv_results["test_score"].mean())

fitted_model = gboost_clf.fit(X_train, y_train)
```

```
Train F1: 0.7138820204874602
Validation F1: 0.6282485875706214
```

Hình 3.40: Kiểm chứng chéo cho mô hình Gradient Boosting

  - Tìm kiếm các siêu tham số tối ưu cho mô hình

```
param_grid = {"model__n_estimators": [50, 100, 150, 200]}

search = GridSearchCV(
    estimator=gboost_clf,
    param_grid=param_grid,
    cv=cv,
    scoring= "f1_micro",
    return_train_score=True
)

search.fit(X_train, y_train)

print("Best parameters: ", search.best_params_)
best_model = search.best_estimator_
```

```
Best parameters:  {'model__n_estimators': 200}
```

Hình 3.41: Tìm các siêu tham số tối ưu cho mô hình Gradient Boosting

- Kết quả dự đoán trên tập test

```
print("F1 on test set:", f1_score(y_test, y_pred, average="micro"))

y_pred = best_model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
F1 on test set: 0.5954802259887005
              precision    recall  f1-score   support

     Dropout       0.55      0.61      0.58       284
    Enrolled       0.33      0.09      0.15       159
    Graduate       0.65      0.77      0.71       442

    accuracy                           0.60       885
   macro avg       0.51      0.49      0.48       885
weighted avg       0.56      0.60      0.57       885
```

Hình 3.42: Kết quả dự đoán trên tập test của mô hình Gradient Boosting

- So sánh kết quả giữa 5 mô hình:

Bảng 3.1: Kết quả của 5 mô hình

Model	Accuracy	Precision	Recall	F1-score
Logistic Regression	0.595	0.56	0.58	0.57
SVM	0.603	0.56	0.6	0.58
Decision Tree	0.598	0.5	0.7	0.58
Random Forest	0.595	0.55	0.58	0.57
Gadient boosting	0.595	0.55	0.61	0.58

+ Để chọn mô hình tốt nhất giúp đưa ra cảnh báo kịp thời ta nên chọn mô hình có chỉ số Recall cao nhất ở đây là Decision Tree tuy mô hình có thể đưa ra cảnh báo không đúng khi một sinh viên không bỏ học nhưng vẫn nhận cảnh báo là bỏ

học nhưng ta thể bắt được hết những sinh viên có nguy cơ bỏ học thật sự. Giúp cho chúng ta không bỏ sót sinh viên có nguy cơ bỏ học nào.

+ Nếu chúng ta không muốn nhận những cảnh báo giả thì nên chọn mô hình có chỉ số Precision cao như Logistic Regression và Support Vector Machine nhưng sẽ bỏ sót các sinh viên có nguy cơ bỏ hoc.

+ Từ kết quả dự đoán trên tập test lớp Enrolled (tiếp tục học) có tỉ lệ khá thấp mô hình hầu như không phát hiện ra những sinh viên này. Vậy nên để mô hình học tốt hơn ý tưởng gộp 2 lớp Graduate và Enrolled lại làm một => Bài toán trở về bài toán phân loại nhị phân

- Đưa bài toán về bài toán phân loại nhị phân
  + Chuyển nhãn Drop out (Bỏ học) thành 0 và hai lớp Graduat (Tốt nghiệp) và Enrolled (tiếp tục học) thành 1

```
y_train = np.where(y_train == 'Dropout', 1, 0)
y_test = np.where(y_test == 'Dropout', 1, 0)
```

Hình 3.43: Chuyển đổi nhãn cho bài toán phân loại nhị phân

+ Lấy các các cột có dữ liệu số và dữ liệu chữ:

```
cat_features = ['Course', 'previous qual', 'Debtor', 'Gender', 'Scholarship holder', 'father occ', 'mother occ', 'father qual', 'mother qual', 'Marital status', 'Application mode',
num_features = ['Previous qualification (grade)', 'Admission grade', 'Age at enrollment', 'Application order']
```

Hình 3.44: Lấy các cột dữ liệu dạng số và dạng chữ

+ Chuẩn hóa dữ liệu với OneHotEncoder cho dữ liệu chữ và StandardScaler cho dữ liệu số

```
select_cat_features = ColumnTransformer([('select_cat', 'passthrough', cat_features)])
cat_transformers = Pipeline([('selector', select_cat_features),
                             ('onehot', OneHotEncoder(handle_unknown='ignore')),
                             ])


select_num_features = ColumnTransformer([('select_num', 'passthrough', num_features)])
num_transformers = Pipeline([('selector', select_num_features),
                             ('scaler', StandardScaler()),
                             ])

preprocess_pipe = FeatureUnion([('cat', cat_transformers),
                                ('num', num_transformers),
                                ])
```

Hình 3.45: Chuẩn hóa dữ liệu với bài toán phân loại nhị phân

+ Tạo một tham số để kiểm chứng chéo và tìm siêu tham số tối ưu:

```
cv = StratifiedShuffleSplit(n_splits = 5, test_size=0.2, random_state=32)
```

Hình 3.46: Tạo tham số cho kiểm chứng chéo và tìm siêu tham số tối ưu

- Huấn luyện các mô hình:
  + LogisticRegression:
    - Khởi tạo mô hình:

```
logistic_clf = Pipeline([
    ("preprocess", preprocess_pipe),
    ("model", LogisticRegression(
        penalty="elasticnet",
        solver="saga",
        l1_ratio=0,
        max_iter=1000
    ))
])
```

Hình 3.47: Khởi tạo mô hình Logistic Regression

    - Cross-validation mô hình:

```
cv_results = cross_validate(
    logistic_clf,
    X_train, y_train,
    cv=cv,
    scoring="recall",
    return_train_score=True
)

print("Train F1:", cv_results["train_score"].mean())
print("Validation F1:", cv_results["test_score"].mean())

fitted_model = logistic_clf.fit(X_train, y_train)
```

```
Train F1: 0.4953846153846154
Validation F1: 0.4775330396475771
```

Hình 3.48: Kiểm chứng chéo cho mô hình Logistic Regression

- Tìm kiếm các siêu tham số tối ưu cho mô hình

```
param_grid = {"model__C": [0.01, 0.1, 1, 10, 100],
              "model__l1_ratio": np.linspace(0, 1, 11)}

search = GridSearchCV(
    estimator=logistic_clf,
    param_grid=param_grid,
    cv=cv,
    scoring="recall",
    return_train_score=True
)

search.fit(X_train, y_train)

print("Best parameters: ", search.best_params_)
best_model = search.best_estimator_
```

```
Best parameters:  {'model__C': 100, 'model__l1_ratio': np.float64(0.0)}
```

Hình 3.49: Tìm các siêu tham số cho mô hình Logistic Regression

- Kết quả dự đoán trên tập test

```
y_pred = best_model.predict(X_test)
print("Recall:", recall_score(y_test, y_pred))
print("Accuracy score on test set: ", accuracy_score(y_test,
print(classification_report(y_test, y_pred))
```

```
Recall: 0.46830985915492956
Accuracy score on test set:  0.7446327683615819
              precision    recall  f1-score   support

           0       0.78      0.88      0.82       601
           1       0.64      0.47      0.54       284

    accuracy                           0.74       885
   macro avg       0.71      0.67      0.68       885
weighted avg       0.73      0.74      0.73       885
```

Hình 3.50: Kết quả dự đoán trên tập test của mô hình Logistic Regression

+ Support Vector Machine:

  - Khởi tạo mô hình:

```
svm_clf = Pipeline(steps=[
    ("preprocess", preprocess_pipe),
    ("model", LinearSVC(random_state=32))
])
```

Hình 3.51: Khởi tạo mô hình Support Vector Machine

  - Cross-validation mô hình:

```
cv_results = cross_validate(
    svm_clf,
    X_train, y_train,
    cv=cv,
    scoring="recall",
    return_train_score=True
)

print("Train F1:", cv_results["train_score"].mean())
print("Validation F1:", cv_results["test_score"].mean())
```

```
Train F1: 0.49098901098901104
Validation F1: 0.4713656387665198
```

Hình 3.52: Kiểm chứng chéo cho mô hình Support Vector Machine

  - Tìm kiếm các siêu tham số tối ưu cho mô hình

```
param_grid = {"model__penalty": ["l1", "l2"],
              "model__C": [0.1, 1, 10, 100, 150, 200]

search = GridSearchCV(
    estimator=svm_clf,
    param_grid=param_grid,
    cv=cv,
    scoring="recall",
    return_train_score=True
)

search.fit(X_train, y_train)

print("Best parameters: ", search.best_params_)
best_model = search.best_estimator_
```

```
Best parameters:  {'model__C': 10, 'model__penalty': 'l1'}
```

Hình 3.53: Tìm các siêu tham số tối ưu cho mô hình Support Vector Machine

- Kết quả dự đoán trên tập test

```
y_pred = best_model.predict(X_test)
print("Recall:", recall_score(y_test, y_pred))
print("Accuracy score on test set: ", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Recall: 0.4612676056338028
Accuracy score on test set:  0.7491525423728813
              precision    recall  f1-score   support

           0       0.78      0.89      0.83       601
           1       0.66      0.46      0.54       284

    accuracy                           0.75       885
   macro avg       0.72      0.67      0.68       885
weighted avg       0.74      0.75      0.74       885
```

Hình 3.54: Tìm các siêu tham số tối ưu cho mô hình Support Vector Machine

+ Decision Tree:

- Khởi tạo mô hình:

```
tree_clf = Pipeline(steps=[
    ("preprocess", preprocess_pipe),
    ("model", DecisionTreeClassifier(random_state=32))
])
```

Hình 3.55: Khởi tạo mô hình Decision Tree

- Cross-validation mô hình:

```
cv_results = cross_validate(
    tree_clf,
    X_train, y_train,
    cv=cv,
    scoring="recall",
    return_train_score=True
)

print("Train F1:", cv_results["train_score"].mean())
print("Validation F1:", cv_results["test_score"].mean())
```

```
Train F1: 1.0
Validation F1: 0.4969162995594714
```

Hình 3.56: Kiểm chứng chéo cho mô hình Decision Tree

- Tìm kiếm các siêu tham số tối ưu cho mô hình

```
param_grid = {"model__ccp_alpha": [0.005, 0.01, 0.05, 0.1, 0],
              "model__min_samples_split":[2, 4, 6, 8, 10],
              "model__min_samples_leaf": [1, 2, 3, 4, 5]}


search = GridSearchCV(
    estimator=tree_clf,
    param_grid=param_grid,
    cv=cv,
    scoring="recall",
    return_train_score=True
)

search.fit(X_train, y_train)

print("Best parameters: ", search.best_params_)
best_model = search.best_estimator_
```

```
Best parameters:  {'model__ccp_alpha': 0.01, 'model__min_samples_leaf': 1, 'model__min_samples_split': 2}
```

Hình 3.57: Tìm các siêu tham số tối ưu cho mô hình Decision Tree

- Kết quả dự đoán trên tập test

```
y_pred = best_model.predict(X_test)
print("Recall:", recall_score(y_test, y_pred))
print("Accuracy score on test set: ", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Recall: 0.5880281690140845
Accuracy score on test set:  0.6813559322033899
              precision    recall  f1-score   support

           0       0.79      0.73      0.76       601
           1       0.50      0.59      0.54       284

    accuracy                           0.68       885
   macro avg       0.65      0.66      0.65       885
weighted avg       0.70      0.68      0.69       885
```

Hình 3.58: Kết quả dự đoán trên tập test của mô hình Decision Tree

+ Random Forest:

  - Khởi tạo mô hình:

```
rf_clf = Pipeline(steps=[
    ("preprocess", preprocess_pipe),
    ("model", RandomForestClassifier(random_state=32))
])
```

Hình 3.59: Khởi tạo mô hình Random Forest

  - Cross-validation mô hình:

```
cv_results = cross_validate(
rf_clf,
    X_train, y_train,
    cv=cv,
    scoring="recall",
    return_train_score=True
)

print("Train F1:", cv_results["train_score"].mean())
print("Validation F1:", cv_results["test_score"].mean())
```

```
Train F1: 1.0
Validation F1: 0.4537444933920705
```

Hình 3.60: Kiểm chứng chéo cho mô hình Random Forest

  - Tìm kiếm các siêu tham số tối ưu cho mô hình

```
param_grid = {"model__n_estimators": [50, 100, 150, 200],
              "model__max_depth": [2, 10, 15, 20, 25],
              "model__min_samples_split":[2, 3, 4, 5],
              "model__min_samples_leaf": [1, 2, 3, 4, 5]}


search = GridSearchCV(
    estimator=rf_clf,
    param_grid=param_grid,
    cv=cv,
    scoring="recall",
    return_train_score=True
)

search.fit(X_train, y_train)

print("Best parameters: ", search.best_params_)
best_model = search.best_estimator_
```

```
Best parameters:  {'model__max_depth': 25, 'model__min_samples_leaf': 1, 'model__min_samples_split': 5, 'model__n_estimators': 50}
```

Hình 3.61: Tìm các siêu tham số tối ưu cho mô hình Random Forest

- Kết quả dự đoán trên tập test

```
y_pred = best_model.predict(X_test)
print("Recall:", recall_score(y_test, y_pred))
print("Accuracy score on test set: ", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Recall: 0.43661971830985913
Accuracy score on test set:  0.7242937853107345
              precision    recall  f1-score   support

           0       0.76      0.86      0.81       601
           1       0.60      0.44      0.50       284

    accuracy                           0.72       885
   macro avg       0.68      0.65      0.66       885
weighted avg       0.71      0.72      0.71       885
```

Hình 3.62: Kết quả dự đoán trên tập test của mô hình Random Forest

+ Gradient Boosting:

- Khởi tạo mô hình:

```
gboost_clf = Pipeline(steps=[
    ("preprocess", preprocess_pipe),
    ("model", GradientBoostingClassifier(random_state=32))
])
```

Hình 3.63: Khởi tạo mô hình Gradient Boosting

- Cross-validation mô hình:

```
cv_rresults = cross_validate(
gboost_clf,
    X_train, y_train,
    cv=cv,
    scoring="recall",
    return_train_score=True
)

print("Train F1:", cv_results["train_score"].mean())
print("Validation F1:", cv_results["test_score"].mean())
```

```
Train F1: 1.0
Validation F1: 0.4537444933920705
```

Hình 3.64: Kiểm chứng chéo cho mô hình Gradient Boosting

- Tìm kiếm các siêu tham số tối ưu cho mô hình

```
param_grid = {"model__n_estimators": [100, 200, 300, 400, 450, 500, 550]}

search = GridSearchCV(
    estimator=gboost_clf,
    param_grid=param_grid,
    cv=cv,
    scoring="recall",
    return_train_score=True
)

search.fit(X_train, y_train)

print("Best parameters: ", search.best_params_)
best_model = search.best_estimator_
```

```
Best parameters:  {'model__n_estimators': 550}
```

Hình 3.65: Tìm các siêu tham số cho mô hình Random Forest

- Kết quả dự đoán trên tập test

```
y_pred = best_model.predict(X_test)
print("Recall:", recall_score(y_test, y_pred))
print("Accuracy score on test set: ", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Recall: 0.4647887323943662
Accuracy score on test set:  0.7265536723163842
              precision    recall  f1-score   support

           0       0.77      0.85      0.81       601
           1       0.59      0.46      0.52       284

    accuracy                           0.73       885
   macro avg       0.68      0.66      0.67       885
weighted avg       0.71      0.73      0.72       885
```

Hình 3.66: Kết quả dự đoán trên tập test của mô hình Gradient Boosting

- Nhận xét:
  + Mô hình đang phát hiện rất kém lớp bỏ học và lớp còn lại rất cao do khi gộp lớp Graduate và Enrolled với nhau số mẫu dữ liệu ở lớp 0 đang nhiều hơn lớp 1 rất nhiều khiến mất cân bằng dữ liệu.

  + Cần dùng các phương pháp cân bằng dữ liệu và huấn luyện lại

- Dùng SMOTE để cân bằng dữ liệu: kỹ thuật cân bằng dữ liệu bằng cách tạo thêm mẫu giả cho lớp thiểu số thay vì chỉ copy y nguyên các mẫu cũ

```
cat_mask = [True if x in cat_features else False for x in X_train.columns]
smote = SMOTENC(categorical_features=cat_mask, random_state=32)
```

Hình 3.67: Xử lý mất cân bằng dữ liệu bằng SMOTE

- Huấn luyện các mô hình:
  + Logistic Regression:
    - Khởi tạo mô hình:

```
logistic_clf = imb_pipeline([
    ("sampler", smote),
    ("preprocess", preprocess_pipe),
    ("model", LogisticRegression(
        penalty="elasticnet",
        solver="saga",
        l1_ratio=0,
        max_iter=1000
    ))
])
```

Hình 3.68: Khởi tạo mô hình Logistic Regression

- Cross-validation mô hình:

```
cv = 5
scoring = "recall"

cv_results = cross_validate(
    logistic_clf,
    X_train, y_train,
    cv=cv,
    scoring=scoring,
    return_train_score=True
)

print("Train F1:", cv_results["train_score"].mean())
print("Validation F1:", cv_results["test_score"].mean())

fitted_model = logistic_clf.fit(X_train, y_train)
```

```
Train F1: 0.7128432403679928
Validation F1: 0.678124275446325
```

Hình 3.69: Kiểm chứng chéo cho mô hình Logistic Regression

- Tìm kiếm các siêu tham số tối ưu cho mô hình

```
param_grid = {"model__C": [0.01, 0.1, 1, 10, 100],
              "model__l1_ratio": np.linspace(0, 1, 11)}

search = GridSearchCV(
    estimator=logistic_clf,
    param_grid=param_grid,
    cv=cv,
    scoring="recall",
    return_train_score=True
)

search.fit(X_train, y_train)

print("Best parameters: ", search.best_params_)
best_model = search.best_estimator_
```

```
Best parameters:  {'model__C': 0.01, 'model__l1_ratio': np.float64(0.30000000000000004)}
```

Hình 3.70: Tìm các siêu tham số tối ưu cho mô hình Logistic Regression

- Kết quả dự đoán trên tập test

```
y_pred = best_model.predict(X_test)
print("Recall:", recall_score(y_test, y_pred))
print("Accuracy score on test set: ", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Recall: 0.75
Accuracy score on test set:  0.6858757062146893
              precision    recall  f1-score   support

           0       0.85      0.66      0.74       601
           1       0.51      0.75      0.61       284

    accuracy                           0.69       885
   macro avg       0.68      0.70      0.67       885
weighted avg       0.74      0.69      0.70       885
```

Hình 3.71: Kết quả dự đoán trên tập test của mô hình Logistic Regression

+ Support Vector Machine:

- Khởi tạo mô hình:

```
svm_clf = imb_pipeline(steps=[
    ("sampler",smote),
    ("preprocess", preprocess_pipe),
    ("model",LinearSVC(random_state=32))
])
```

Hình 3.72: Khởi tạo mô hình Support Vector Machine

- Cross-validation mô hình:

```
cv_results = cross_validate(
svm_clf,
    X_train,
    y_train,
    cv=cv,
    scoring="recall",
    return_train_score=True
)

print("Train F1:", cv_results["train_score"].mean())
print("Validation F1:", cv_results["test_score"].mean())
```

```
Train F1: 0.7161406690119563
Validation F1: 0.6781204111600588
```

Hình 3.73: Kiểm chứng chéo cho mô hình Support Vector Machine

- Tìm kiếm các siêu tham số tối ưu cho mô hình

```
param_grid = {"model__penalty": ["l1", "l2"],
                "model__C": [0.001, 0.005, 0.01, 0.1, 1, 10, 100]}

search = GridSearchCV(
    estimator=svm_clf,
    param_grid=param_grid,
    cv=5,
    scoring="recall",
    return_train_score=True
)

search.fit(X_train, y_train)

print("Best parameters: ", search.best_params_)
best_model = search.best_estimator_
```

```
Best parameters:  {'model__C': 0.01, 'model__penalty': 'l1'}
```

Hình 3.74: Tìm các siêu tham số tối ưu cho mô hình Support Vector Machine

- Kết quả dự đoán trên tập test

```
y_pred = best_model.predict(X_test)
print("Recall:", recall_score(y_test, y_pred))
print("Accuracy score on test set: ", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Recall: 0.7570422535211268
Accuracy score on test set:  0.6847457627118644
              precision    recall  f1-score   support

           0       0.85      0.65      0.74       601
           1       0.51      0.76      0.61       284

    accuracy                           0.68       885
   macro avg       0.68      0.70      0.67       885
weighted avg       0.74      0.68      0.70       885
```

Hình 3.75: Kết quả dự đoán trên tập test của mô hình Support Vector Machine

+ Decision Tree:

  - Khởi tạo mô hình:

```
tree_clf = imb_pipeline([
    ("sampler", smote),
    ("preprocess", preprocess_pipe),
    ("model", DecisionTreeClassifier(random_state=32))
])
```

Hình 3.76: Khởi tạo mô hình Decision Tree

  - Cross-validation mô hình:

```
cv = 5
scoring = "recall"

cv_results = cross_validate(
    tree_clf,
    X_train, y_train,
    cv=cv,
    scoring=scoring,
    return_train_score=True
)

print("Train F1:", cv_results["train_score"].mean())
print("Validation F1:", cv_results["test_score"].mean())

fitted_model = tree_clf.fit(X_train, y_train)
```

```
Train F1: 1.0
Validation F1: 0.5259989179998454
```

Hình 3.77: Kiểm chứng chéo cho mô hình Decision Tree

- Tìm kiếm các siêu tham số tối ưu cho mô hình

```
param_grid = {
    "model__max_depth": [2, 3, 4, 5, 6, 7, 8],
    "model__min_samples_split": [2, 4, 6, 8, 10],
    "model__min_samples_leaf": [1, 2, 3, 4, 5, 6]
}

search = GridSearchCV(
    estimator=tree_clf,
    param_grid=param_grid,
    cv=cv,
    scoring=scoring,
    return_train_score=True
)

search.fit(X_train, y_train)

print("Best parameters: ", search.best_params_)
best_model = search.best_estimator_
```

```
Best parameters:  {'model__max_depth': 3, 'model__min_samples_leaf': 1, 'model__min_samples_split': 2}
```

Hình 3.78: Tìm các siêu tham số tối ưu cho mô hình Decision Tree

- Kết quả dự đoán trên tập test

```
y_pred = best_model.predict(X_test)
print("Recall:", recall_score(y_test, y_pred))
print("Accuracy score on test set: ", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Recall: 0.8838028169014085
Accuracy score on test set:  0.5581920903954802
              precision    recall  f1-score   support

           0       0.88      0.40      0.55       601
           1       0.41      0.88      0.56       284

    accuracy                           0.56       885
   macro avg       0.65      0.64      0.56       885
weighted avg       0.73      0.56      0.56       885
```

Hình 3.79: Kết quả dự đoán trên tập test của mô hình Decision Tree

+ Random Forest:

  - Khởi tạo mô hình:

```
rf_clf = imb_pipeline([
    ("sampler", smote),
    ("preprocess", preprocess_pipe),
    ("model", RandomForestClassifier(random_state=32))
])
```

Hình 3.80: Khởi tạo mô hình Random Forest

  - Cross-validation mô hình:

```
cv = 5
scoring = "recall"

cv_results = cross_validate(
    rf_clf,
    X_train, y_train,
    cv=cv,
    scoring=scoring,
    return_train_score=True
)

print("Train F1:", cv_results["train_score"].mean())
print("Validation F1:", cv_results["test_score"].mean())

fitted_model = rf_clf.fit(X_train, y_train)
```

Hình 3.81: Kiểm chứng chéo cho mô hình Random Forest

- Tìm kiếm các siêu tham số tối ưu cho mô hình

```
param_grid = {"model__n_estimators": [50, 100, 150, 200],
              "model__max_depth": [2, 10, 15, 20, 25],
              "model__min_samples_split":[2, 3, 4, 5],
              "model__min_samples_leaf": [1, 2, 3, 4, 5]}

search = GridSearchCV(
    estimator=rf_clf,
    param_grid=param_grid,
    cv=cv,
    scoring=scoring,
    return_train_score=True
)

search.fit(X_train, y_train)

print("Best parameters: ", search.best_params_)
best_model = search.best_estimator_
```

```
Best parameters:  {'model__max_depth': 2, 'model__min_samples_leaf': 1, 'model__min_samples_split': 2, 'model__n_estimators': 150}
```

Hình 3.82: Tìm các siêu tham số tối ưu cho mô hình Random Forest

- Kết quả dự đoán trên tập test

```
y_pred = best_model.predict(X_test)
print("Recall:", recall_score(y_test, y_pred))
print("Accuracy score on test set: ", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Recall: 0.7394366197183099
Accuracy score on test set:  0.6937853107344633
              precision    recall  f1-score   support

           0       0.85      0.67      0.75       601
           1       0.52      0.74      0.61       284

    accuracy                           0.69       885
   macro avg       0.68      0.71      0.68       885
weighted avg       0.74      0.69      0.70       885
```

Hình 3.83: Kết quả dự đoán trên tập test của mô hình Random Forest

+ Gradient Boosting:

- Khởi tạo mô hình:

```
gboost_clf = imb_pipeline([
    ("sampler", smote),
    ("preprocess", preprocess_pipe),
    ("model", RandomForestClassifier(random_state=32))
])
```

Hình 3.84: Khởi tạo mô hình Gradient Boosting

- Cross-validation mô hình:

```
cv = 5
scoring = "recall"

cv_results = cross_validate(
    gboost_clf,
    X_train, y_train,
    cv=cv,
    scoring=scoring,
    return_train_score=True
)

print("Train F1:", cv_results["train_score"].mean())
print("Validation F1:", cv_results["test_score"].mean())

fitted_model = gboost_clf.fit(X_train, y_train)
```

```
Train F1: 1.0
Validation F1: 0.593693484813355
```

Hình 3.85: Kiểm chứng chéo cho mô hình Gradient Boosting

- Tìm kiếm các siêu tham số tối ưu cho mô hình

```
param_grid = {"model__n_estimators": [100, 200, 300, 400, 450, 500, 550]}


search = GridSearchCV(
    estimator=gboost_clf,
    param_grid=param_grid,
    cv=cv,
    scoring=scoring,
    return_train_score=True
)

search.fit(X_train, y_train)

print("Best parameters: ", search.best_params_)
best_model = search.best_estimator_
```

```
Best parameters:  {'model__n_estimators': 200}
```

Hình 3.86: Tìm các siêu tham số cho mô hình Gradient Boosting

- Kết quả dự đoán trên tập test

```
y_pred = best_model.predict(X_test)
print("Recall:", recall_score(y_test, y_pred))
print("Accuracy score on test set: ", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Recall: 0.5915492957746479
Accuracy score on test set:  0.7141242937853107
              precision    recall  f1-score   support

           0       0.80      0.77      0.79       601
           1       0.55      0.59      0.57       284

    accuracy                           0.71       885
   macro avg       0.68      0.68      0.68       885
weighted avg       0.72      0.71      0.72       885
```

Hình 3.87: Kết quả dự đoán trên tập test của mô hình Gradient Boosting

- So sánh kết quả 5 mô hình

Bảng 3.2: Kết quả 5 mô hình với bài toán phân loại nhị phân

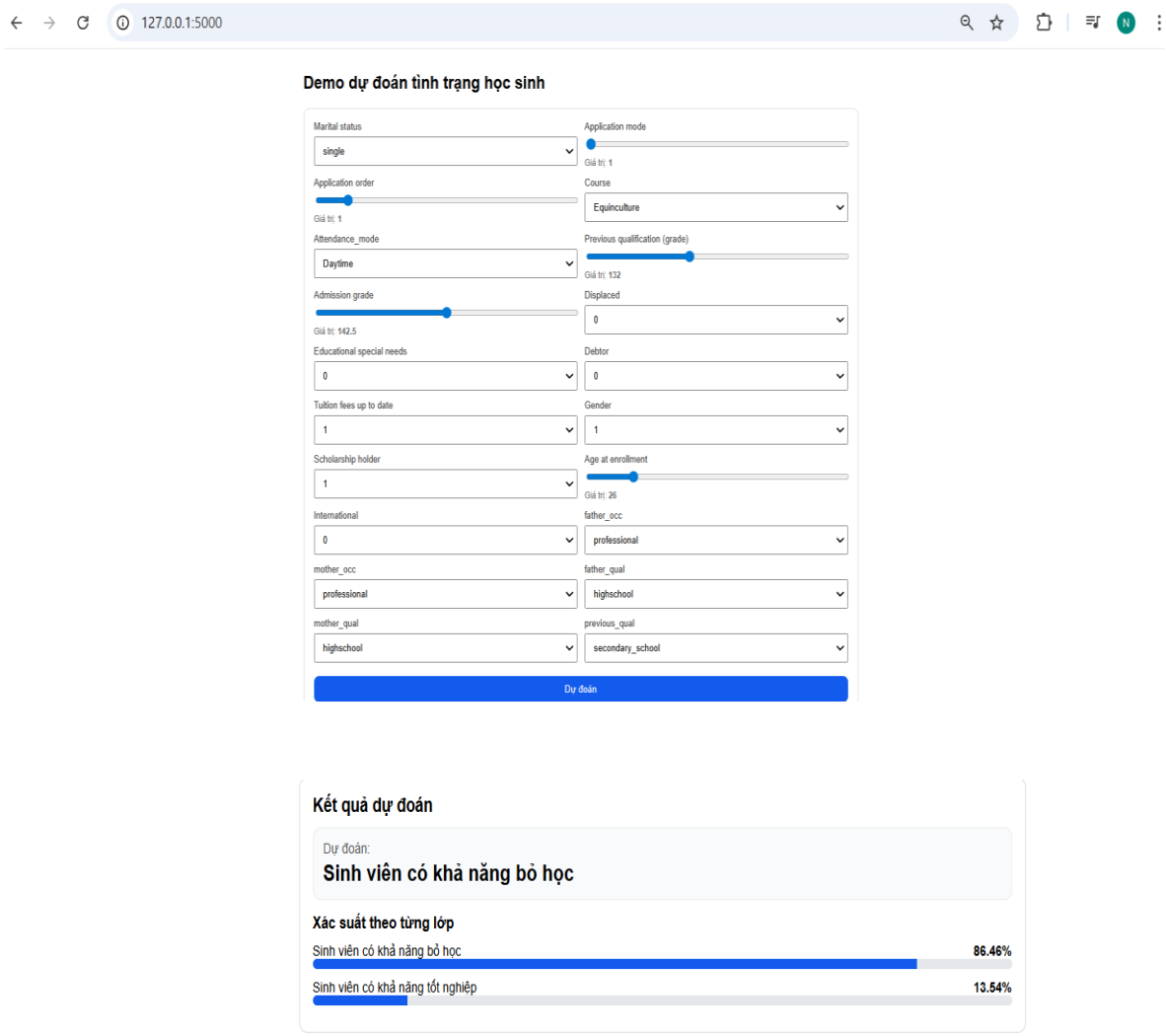
Model	Accuracy	Precision	Recall	F1-score
Logistic Regression	0.685	0.51	0.75	0.61
SVM	0.684	0.51	0.76	0.61
Decision Tree	0.558	0.41	0.88	0.56
Random Forest	0.693	0.52	0.74	0.61
Gadient boosting	0.714	0.55	0.59	0.57

+ Mô hình có khả năng phát hiện sinh viên nghỉ học cao nhất vẫn là Decision Tree khi chuyển về bài toán phân loại nhị phân khả năng phát hiện cải thiện lên là 0,88 dựa trên chỉ số Recall đây là một kết khá cao để đưa ra áp dụng cảnh báo.

+ Nếu muốn chọn mô hình có độ chính xác và độ tin cậy tốt nên chon Gradient Boosting nhưng khả năng bỏ sót những sinh viên có nguy cơ nghỉ học là không cao bằng các mô hình còn lại.

+ Các mô hình như Logistic Regression và Support Vector Machine là tương đối ổn định dựa trên các chỉ số khả năng phát hiện là khá cao và độ chính xác khá tốt.

- Sản phẩm



Hình 3.88: Giao diện dự đoán sinh viên nghỉ học dựa trên mô hình Decision Tree

TỔNG KẾT

Báo cáo này tập trung giải quyết bài toán phát hiện sinh viên có nguy cơ bỏ học nhằm hỗ trợ nhà trường cảnh báo sớm và can thiệp kịp thời. Dữ liệu sử dụng là bộ Student Dropout Prediction với nhãn mục tiêu gồm: Dropout – Enrolled – Graduate, tổng cộng 4424 bản ghi và 37 thuộc tính, phản ánh cả đặc điểm nhân khẩu học, kinh tế – xã hội và kết quả học tập. Về xử lý dữ liệu và phân tích đã thực hiện đồng nhất/mã hóa lại một số nhóm thuộc tính (trình độ/ nghề nghiệp cha mẹ, trình độ học vấn trước đó của sinh viên, …) và áp dụng OneHotEncoder cho dữ liệu dạng chữ, kết hợp StandardScaler cho dữ liệu dạng số để đảm bảo dữ liệu phù hợp cho huấn luyện mô hình. Qua thống kê, có thể thấy tập dữ liệu có 32% sinh viên bỏ học, 18% đang theo học và 50% tốt nghiệp. Một số nhận xét từ trực quan hóa dữ liệu cho thấy: nhóm tuổi ≥ 25 có tỷ lệ bỏ học cao hơn, nhóm không độc thân có xu hướng bỏ học nhiều hơn, và học buổi tối có khả năng bỏ học cao hơn. Trong việc xây dựng mô hình và thử nghiệm 5 thuật toán phân loại gồm: Logistic Regression, Support Vector Machine, Decision Tree, Random Forest và Gradient Boosting, đánh giá bằng các chỉ số Accuracy /Precision/Recall/F1-score, đồng thời dùng cross-validation và GridSearchCV để tinh chỉnh siêu tham số. Kết quả so sánh cho thấy nếu mục tiêu là cảnh báo sớm (ưu tiên không bỏ sót sinh viên có nguy cơ bỏ học), thì nên chọn mô hình có Recall cao, trong đó Decision Tree nổi bật; ngược lại nếu muốn giảm cảnh báo nhầm thì Logistic Regression/Support Vector Machine có kết quả Precision tốt hơn nhưng có thể bỏ sót một phần sinh viên bỏ học. Ngoài ra, do lớp Enrolled được dự đoán kém và hầu như không phát hiện nên đã đưa bài toán về phân loại nhị phân bằng cách gộp hai lớp Graduate và Enrolled thành một lớp, tuy nhiên phát sinh mất cân bằng dữ liệu nên cần kỹ thuật cân bằng như SMOTE để cải thiện khả năng học của mô hình. Sau khi cân bằng dữ liệu, Decision Tree vẫn là mô hình phát hiện sinh viên có nguy cơ nghỉ học tốt nhất, với Recall cải thiện lên khoảng 0,88, phù hợp cho bài toán cảnh báo sớm trong thực tế; trong khi đó Gradient Boosting có thể là lựa chọn cân bằng hơn nếu ưu tiên độ chính xác/độ tin cậy tổng thể.

TÀI LIỆU THAM KHẢO

[1] Dữ liệu: Student Dropout Prediction

[2] Logistic Regression in Machine Learning. "https://www.geeksforgeeks.org/machine-learning/understanding-logistic-regression/"

[3] Introduction to Support Vector Machines (SVM). "https://www.geeksforgeeks.org/machine-learning/introduction-to-support-vector-machines-svm/"

[4] Decision Tree. "https://www.geeksforgeeks.org/machine-learning/decision-tree/"

[5] Random Forest Algorithm in Machine Learning. "https://www.geeksforgeeks.org/machine-learning/random-forest-algorithm-in-machine-learning/"

[6] Gradient Boosting in ML. "https://www.geeksforgeeks.org/machine-learning/ml-gradient-boosting/"

[7] Confusion matrix. "https://en.wikipedia.org/wiki/Confusion_matrix"