

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



BÁO CÁO GIỮA KÌ

NHẬP MÔN HỌC MÁY

Người hướng dẫn: **TS TRẦN LƯƠNG QUỐC ĐẠI**

Người thực hiện: **NGUYỄN HOÀNG ÂN – 52200183**

NGUYỄN DUY HÀ VỸ - 52200180

NGUYỄN NHẬT TRƯỜNG - 52200192

Lớp : 22050301

Khoá : 26

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

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



BÁO CÁO GIỮA KỲ

NHẬP MÔN HỌC MÁY

Người hướng dẫn: **TS TRẦN LƯƠNG QUỐC ĐẠI**

Người thực hiện: **NGUYỄN HOÀNG ÂN – 52200183**

NGUYỄN DUY HÀ VỸ - 52200180

NGUYỄN NHẬT TRƯỜNG - 52200192

Lớp : 22050301

Khoá : 26

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

LỜI CẢM ƠN

Đầu tiên chúng em xin chân thành cảm ơn thầy Trần Lương Quốc Đại là thầy dạy trực tiếp bộ môn Nhập môn Học Máy. Trong suốt thời gian học tập, thầy là người đã tận tình chỉ dạy, hướng dẫn cũng như giúp đỡ để chúng em có thể hoàn thành bài báo cáo giữa kỳ.

Chúng em cũng xin chân thành cảm ơn toàn thể giáo viên khoa Công nghệ thông tin của Trường Đại học Tôn Đức Thắng đã ủng hộ, đồng hành để hoàn thiện bài báo cáo này đồng thời giúp chúng em có thêm nhiều kiến thức để chuẩn bị cho hành trang sắp tới.

Bài báo cáo này là minh chứng rõ nhất cho những cố gắng, nỗ lực học hỏi của chúng em, tuy nhiên cũng không tránh khỏi thiếu sót. Chúng em mong rằng sẽ nhận được sự đóng góp, chỉ bảo từ các thầy cô để có thể sửa đổi, bổ sung và hoàn thành tốt hơn nữa.

Chúng em rất mong sẽ nhận được sự hài lòng, giúp đỡ của các thầy cô. Một lần nữa chúng em xin chân thành cảm ơn!

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là tiểu luận của riêng chúng tôi và được sự hướng dẫn khoa học của TS Trần Lương Quốc Đại. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

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

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

TP. Hồ Chí Minh, ngày 13 tháng 11 năm 2024

Tác giả

(ký tên và ghi rõ họ tên)



Nguyễn Hoàng Ân



Nguyễn Duy Hà Vỹ



Nguyễn Nhật Trường

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

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

Phần đánh giá của GV chấm bài

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

TÓM TẮT

❖ Báo cáo gồm 4 chương:

Chương 1: Giới thiệu các phương pháp học máy

- Khái quát lý thuyết.

Chương 2: Xây dựng các mô hình học máy

- Sử dụng các phương pháp học máy đã học (có thể sử dụng thêm các phương pháp học máy khác) để giải bài toán này theo cả phương pháp classification và regression.

Chương 3: Overfitting

- Tìm hiểu và trình bày về vấn đề overfitting nói chung, và giải pháp giải quyết overfitting cho các phương pháp học ở chương 2.

Chương 4: Trích chọn đặc trưng sử dụng phân tích tương quan

- Tìm hiểu và trình bày Feature selection using correlation analysis. Áp dụng vào bài toán trên đối với các phương pháp ở chương 2.

DANH MỤC HÌNH ẢNH

CHƯƠNG 1 GIỚI THIỆU CÁC PHƯƠNG PHÁP HỌC MÁY

CHƯƠNG 2 XÂY DỰNG MÔ HÌNH HỌC MÁY

Hình 2. 1	6
Hình 2. 2	7
Hình 2. 3	9
Hình 2. 4	9
Hình 2. 5	9
Hình 2. 6	10
Hình 2. 7	11
Hình 2. 8	11
Hình 2. 9	12
Hình 2. 10	13
Hình 2. 11	13
Hình 2. 12	14
Hình 2. 13	15
Hình 2. 14	15
Hình 2. 15	16
Hình 2. 16	17
Hình 2. 17	17
Hình 2. 18	18
Hình 2. 19	19
Hình 2. 20	19
Hình 2. 21	20
Hình 2. 22	20

Hình 2. 23	21
Hình 2. 24	22
Hình 2. 25	23
Hình 2. 26	23
Hình 2. 27	24
Hình 2. 28	24
Hình 2. 29	25
Hình 2. 30	26
Hình 2. 31	27
Hình 2. 32	28
Hình 2. 33	29
Hình 2. 34	30
Hình 2. 35	31
Hình 2. 36	31
Hình 2. 37	32
Hình 2. 38	32
Hình 2. 39	34
Hình 2. 40	34
Hình 2. 41	36
Hình 2. 42	36
Hình 2. 43	37
Hình 2. 44	39
CHƯƠNG 3 OVERFITTING	
Hình 3. 1	41
Hình 3. 2	44
Hình 3. 3	44
Hình 3. 4	45

Hình 3. 5	45
Hình 3. 6	47
Hình 3. 7	48
Hình 3. 8	49
Hình 3. 9	50
CHƯƠNG 4 FEATURE SELECTION USING CORRELATION ANALYSIS	
Hình 4. 1	54
Hình 4. 2	54
Hình 4. 3	55
Hình 4. 4	55
Hình 4. 5	55
Hình 4. 6	56
Hình 4. 7	57
Hình 4. 8	58
Hình 4. 9	58
Hình 4. 10	60
Hình 4. 11	61
Hình 4. 12	62

MỤC LỤC

LỜI CẢM ƠN	i
PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN	iii
TÓM TẮT	iv
DANH MỤC HÌNH ẢNH	v
MỤC LỤC.....	1
CHƯƠNG 1 – GIỚI THIỆU CÁC PHƯƠNG PHÁP HỌC MÁY.....	3
1. Giới thiệu về các phương pháp học máy.....	3
2. Các bước cơ bản trong một quy trình học máy.....	4
CHƯƠNG 2 – XÂY DỰNG CÁC MÔ HÌNH HỌC MÁY	5
1. Đề bài:	5
2. Thống kê đặc điểm của dữ liệu bằng data visualization	5
1.1 Data visualization là gì?.....	6
1.2 Biểu đồ trực quan hóa các dữ liệu	6
1.3 Phân tích chi tiết các dữ liệu	8
3. Tiền xử lý dữ liệu.....	19
4. Phân chia dữ liệu thành tập train và tập evaluation	26
5. Thực hiện phân loại và hồi quy.....	26
5.1 Thực hiện phân loại.....	26
5.1.1 LogisticRegression.....	26
5.1.2 DecisionTreeClassifier.....	28
5.1.3 NaiveBayesClassifier:.....	30
5.2 Thực hiện hồi quy	31
5.2.1 LinearRegression	31
5.2.2 SVR:.....	33
5.2.3 K - Nearest Neighbor:.....	34
6. Đánh giá kết quả và so sánh các phương pháp	36

6.1 So sánh các phương pháp phân loại với nhau:.....	36
6.2 So sánh các phương pháp hồi quy với nhau:	38
CHƯƠNG 3 – OVERFITTING	41
1. Vấn đề Overfitting:	41
2. Hiện tượng overfitting xảy ra khi :	41
3. Cách nhận dạng khi mô hình bị overfitting:	42
4. Phương pháp giải quyết overfitting:	42
5. Thực hiện code bổ sung giải pháp overfitting.....	43
5.1 Classification Overfitting	43
5.2 Regression Overfitting	47
CHƯƠNG 4 – FEATURE SELECTION USING CORRELATION ANALYSIS	52
1. Giới thiệu về Feature Selection (Trích chọn đặc trưng)	52
1.1 Các phương pháp chính	52
1.2 Feature selection using correlation analysis	52
2. Thực nghiệm	53
2.1 Tính ma trận tương quan và vẽ biểu đồ heatmap.....	53
2.2 Tính toán hệ số tương quan giữa target và các đặc trưng còn lại	55
2.3 Chọn đặc trưng.....	55
2.4 Huấn luyện lại mô hình.....	56
2.5 Áp dụng vào mô hình Classification.....	56
2.6 Áp dụng vào mô hình Regression.....	59
3. Tổng kết	63

CHƯƠNG 1 – GIỚI THIỆU CÁC PHƯƠNG PHÁP HỌC MÁY

1. Giới thiệu về các phương pháp học máy

Học máy (Machine Learning) là một nhánh của trí tuệ nhân tạo (AI) tập trung vào việc phát triển các thuật toán và mô hình cho phép máy tính học hỏi và đưa ra dự đoán hoặc quyết định dựa trên dữ liệu mà không cần lập trình tường minh cho từng tác vụ. Thay vì viết mã để xử lý mọi trường hợp, trong học máy, chúng ta cung cấp dữ liệu và để thuật toán tự động tìm ra quy luật hoặc mẫu trong dữ liệu đó.

Học máy được chia thành ba loại chính, bao gồm:

- Học có giám sát (Supervised Learning): Mô hình học từ một tập dữ liệu có nhãn (label), tức là dữ liệu đã biết kết quả đầu ra. Mục tiêu là tìm ra mối quan hệ giữa đầu vào và đầu ra để dự đoán cho dữ liệu mới. Ví dụ: Dự đoán giá nhà dựa trên kích thước và vị trí, nhận diện hình ảnh là mèo hay chó.

- Hồi quy tuyến tính (Linear Regression): Dự đoán giá trị liên tục (như giá nhà).
- Cây quyết định (Decision Tree): Chia dữ liệu thành các nhánh để dự đoán đầu ra.
- Random Forest: Một tập hợp các cây quyết định, cải thiện độ chính xác.
- Hồi quy logistic (Logistic Regression): Dự đoán biến phân loại, ví dụ như phân loại email là spam hoặc không.
- Nearest Neighbors (KNN): Phân loại hoặc dự đoán dựa trên các điểm gần nhất.

- Học không giám sát (Unsupervised Learning): Mô hình học từ dữ liệu không có nhãn, tìm cách phát hiện các mẫu ẩn hoặc cấu trúc bên trong dữ liệu. Ví dụ: Phân nhóm khách hàng thành các nhóm để tiếp thị, hoặc giảm số chiều của dữ liệu để trực quan hóa tốt hơn.

- K-Means Clustering: Phân cụm dữ liệu thành các nhóm dựa trên sự tương đồng.
- Principal Component Analysis (PCA): Giảm số chiều của dữ liệu trong khi giữ lại nhiều thông tin nhất có thể.
- Hierarchical Clustering: Xây dựng cây phân cấp của các nhóm trong dữ liệu.
- Association Rules: Phân tích các mẫu liên kết giữa các mặt hàng trong giỏ hàng (như trong phân tích giỏ hàng bán lẻ).

- Học tăng cường (Reinforcement Learning): Mô hình học thông qua quá trình thử và sai, nhận thưởng hoặc phạt dựa trên hành động, từ đó tối ưu hóa một chiến lược để đạt được mục tiêu. Ví dụ: Học cách chơi một trò chơi, robot học cách điều hướng một môi trường.

- Q-Learning: Thuật toán học một bảng Q để tối ưu hóa chiến lược dựa trên phần thưởng.
- Deep Q-Networks (DQN): Kết hợp Q-Learning và mạng nơ-ron sâu để xử lý các không gian trạng thái lớn.
- Policy Gradient Methods: Thuật toán tối ưu chính sách trực tiếp, hữu ích cho các bài toán phức tạp hơn.

2. Các bước cơ bản trong một quy trình học máy

- Thu thập và xử lý dữ liệu: Dữ liệu là nền tảng cho học máy, vì vậy việc làm sạch, chuyển đổi và xử lý dữ liệu rất quan trọng.

- Chọn mô hình: Tùy thuộc vào bài toán và dữ liệu, chọn mô hình thích hợp.

- Huấn luyện mô hình: Cung cấp dữ liệu huấn luyện để mô hình học hỏi.

- Đánh giá mô hình: Kiểm tra hiệu suất của mô hình trên dữ liệu kiểm tra hoặc qua các phương pháp đánh giá khác.

- Triển khai mô hình: Đưa mô hình vào sử dụng trong môi trường thực tế để đưa ra dự đoán hoặc hỗ trợ quyết định.

CHƯƠNG 2 – XÂY DỰNG CÁC MÔ HÌNH HỌC MÁY

1. Đề bài:

Sử dụng các phương pháp học máy đã học (có thể sử dụng thêm các phương pháp học máy khác) để giải bài toán này theo cả phương pháp classification và regression. Thực hiện các bước sau:

- Thống kê đặc điểm của dữ liệu bằng data visualization
- Tiền xử lý dữ liệu: type conversation và data normalization
- Phân chia dữ liệu thành tập train và tập evaluation
- Thực hiện phân loại (và hồi quy)
- Đánh giá kết quả và so sánh các phương pháp: so sánh các phương pháp phân loại với nhau; so sánh các phương pháp hồi quy với nhau. Sử dụng biểu đồ để hiển thị sự so sánh cho trực quan.

So sánh 2 phương pháp classification và regression:

Đặc điểm	Classification	Regression
Đầu ra	Nhãn hoặc lớp phân loại	Giá trị số liên tục
Ứng dụng	Phân loại email spam, nhận diện hình ảnh	Dự đoán giá nhà, dự báo thời tiết
Thuật toán	Logistic Regression, SVM, KNN, Naive Bayes, Decision Tree	Linear Regression, Polynomial Regression, Decision Tree Regressor

2. Thống kê đặc điểm của dữ liệu bằng data visualization

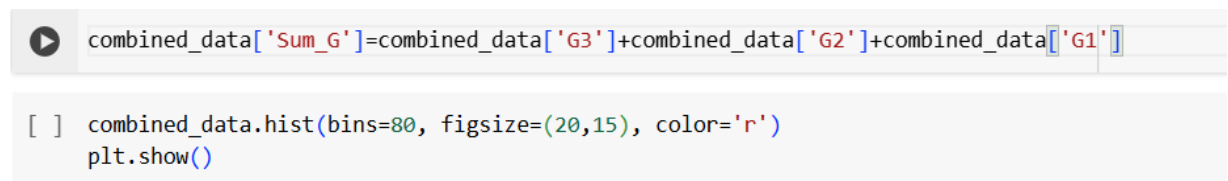
Bộ dữ liệu được sử dụng trong báo cáo này là bộ Student+performance được lấy từ UCI datasets. Bộ dữ liệu gồm có 33 đặc trưng như sau:

school; sex; age; address; famsize; Pstatus; Medu; Fedu; Mjob; Fjob; reason; guardian; traveltime; studytime; failures; schoolsup; famsup; paid; activities; nursery; higher; internet; romantic; famrel; freetime; goout; Dalc; Walc; health; absences; G1; G2; G3

1.1 Data visualization là gì?

Data Visualization (Trực quan hóa dữ liệu) là quá trình chuyển đổi dữ liệu thành các dạng biểu diễn trực quan như biểu đồ, đồ thị, bản đồ, và hình ảnh, nhằm mục đích giúp người xem dễ dàng hiểu, phân tích và khám phá các mẫu, xu hướng và thông tin ẩn chứa trong dữ liệu.

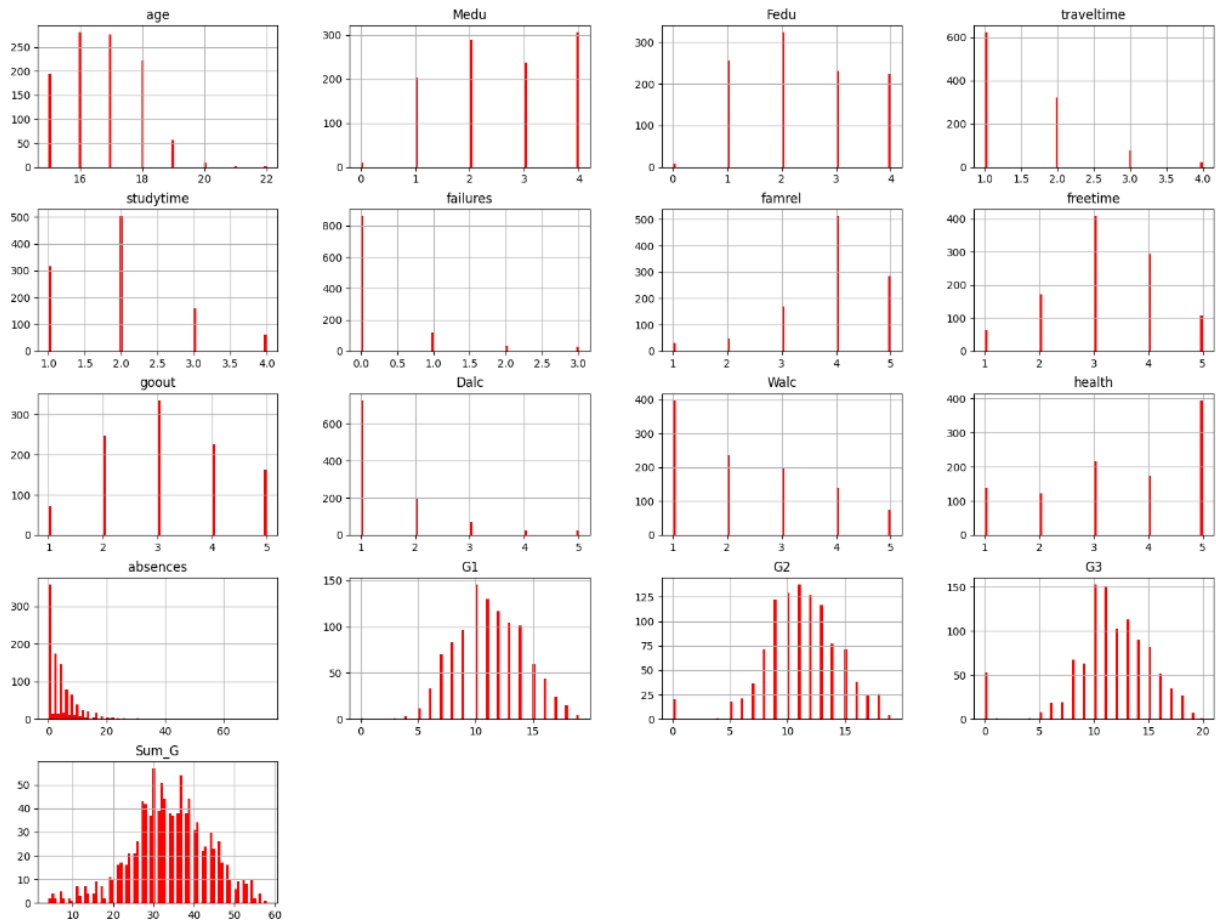
1.2 Biểu đồ trực quan hóa các dữ liệu



```
combined_data['Sum_G']=combined_data['G3']+combined_data['G2']+combined_data['G1']

[ ] combined_data.hist(bins=80, figsize=(20,15), color='r')
plt.show()
```

Hình 2. 1



Hình 2. 2

Nhận xét tổng quan:

- Biểu đồ age: Với tổng 1,044 học sinh, tuổi trung bình duy trì khoảng 16.7, cho thấy hai nhóm học sinh có độ tuổi tương đối đồng đều. Phần lớn học sinh ở trong khoảng tuổi từ 16 đến 18, cho thấy đây là nhóm tuổi chính trong dữ liệu này. Điều này phản ánh nhóm tuổi phổ biến trong các lớp học cấp ba.

- Biểu đồ Medu và Fedu: Trình độ học vấn của mẹ và bố, với giá trị từ 0 (không học vấn) đến 4 (giáo dục đại học). Trung bình, mức học vấn của bố mẹ thường ở mức 2-3, thể hiện sự học vấn tương đối cao. Mức 4 ở biểu đồ Medu cao hơn so với của Fedu.

- Biểu đồ traveltime và studytime: Thời gian di chuyển đến trường và thời gian học tập hàng tuần. Thời gian di chuyển thường là 1 (dưới 15 phút) và thời gian học trung bình là 2-3 giờ mỗi tuần.

- Biểu đồ failures: Số lượng học sinh không có lần rớt môn nào chiếm phần lớn. Có một số ít học sinh có từ 1 đến 3 lần rớt môn, nhưng số lượng này nhỏ hơn nhiều so với nhóm không rớt môn.

- Biểu đồ famel: Mối quan hệ trong gia đình từ 1 đến 5, với trung bình là 3.35, thể hiện mức quan hệ gia đình tương đối tốt.

- Biểu đồ freetime, goout, Dalc, Walc: Thời gian rảnh rỗi, thời gian ra ngoài, và mức độ tiêu thụ rượu trong ngày thường và cuối tuần. Thời gian ra ngoài và mức tiêu thụ rượu cuối tuần cao hơn trong ngày thường, cho thấy học sinh giải trí nhiều hơn vào cuối tuần.

- Biểu đồ health: Tình trạng sức khỏe từ 1 (rất xấu) đến 5 (rất tốt), với trung bình là 3.36, cho thấy phần lớn học sinh có sức khỏe tương đối ổn định.

- Biểu đồ absences: Số buổi vắng mặt với sự phân tán lớn, trung bình là khoảng 2 buổi nhưng có học sinh vắng đến 14 buổi.

- Biểu đồ G1, G2, G3: Điểm số ở các kỳ thi của môn Toán và môn Bồ Đào Nha. Trung bình điểm số dao động từ 11-12, cho thấy thành tích học tập ở mức trung bình khá.

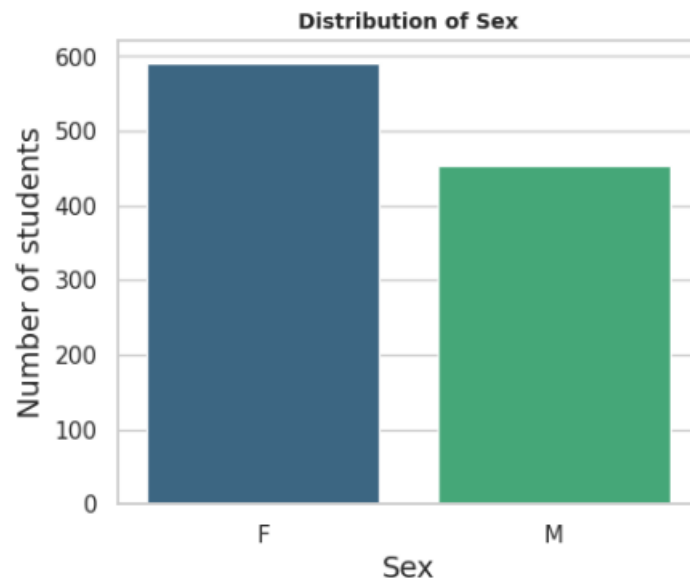
- Biểu đồ Sum_G: Tổng điểm số giao động từ 30-40 khá cao, cho thấy rằng học sinh có thành tích học tập ở mức trung bình khá nhiều hơn so với các mức còn lại.

1.3 Phân tích chi tiết các dữ liệu

- Data visualization của cột sex:

```
[ ] #Sex
sns.set(style="whitegrid")
plt.figure(figsize=(5, 4))
sns.countplot(x=combined_data['sex'],palette="viridis" )
plt.title('Distribution of Sex', fontsize=10, fontweight='bold')
plt.xlabel('Sex', fontsize=14)
plt.ylabel('Number of students', fontsize=14)
```

Hình 2. 3

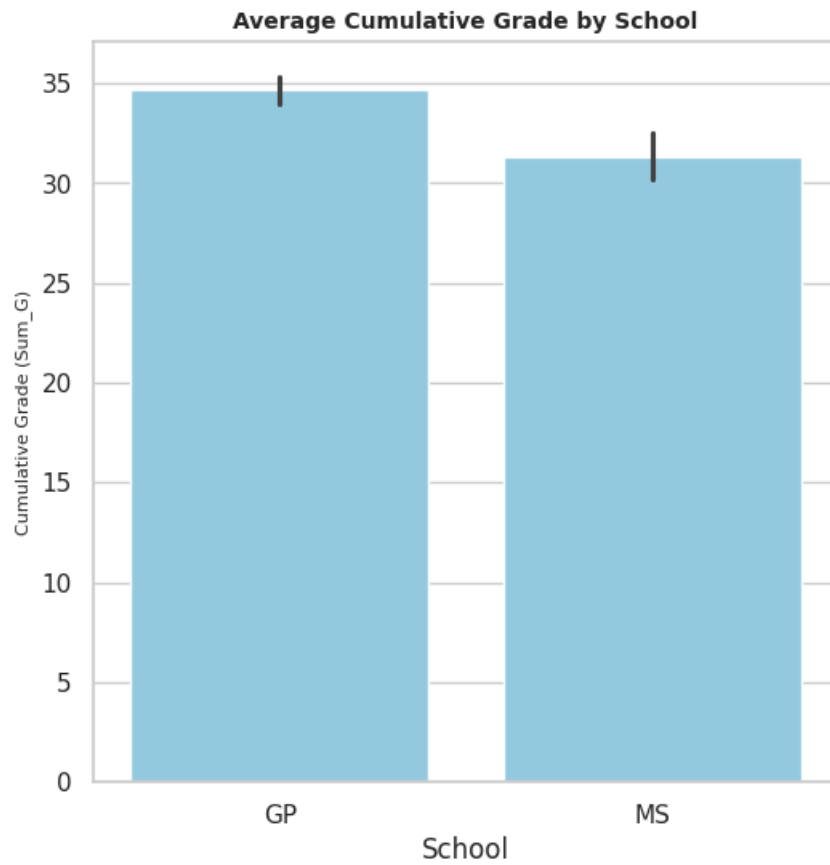


Hình 2. 4

- Data visualization của cột school so với Sum_G:

```
#students of which school are scoring
sns.set(style="whitegrid")
plt.figure(figsize=(6, 6))
sns.barplot(x='school',y='Sum_G',data=combined_data,color='skyblue')
plt.title('Average Cumulative Grade by School', fontsize=10, fontweight='bold')
plt.xlabel('School', fontsize=12)
plt.ylabel('Cumulative Grade (Sum_G)', fontsize=8)
# Hiển thị biểu đồ
plt.show()
```

Hình 2. 5

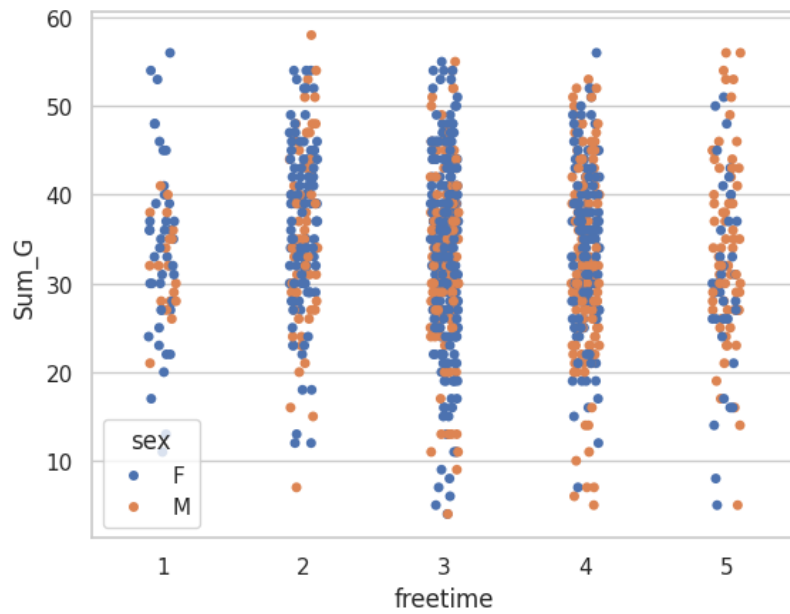


Hình 2. 6

- Data visualization của cột freetime so với Sum_G:

```
sns.stripplot(x='freetime',y='Sum_G',data=combined_data,hue='sex',jitter=True)
```

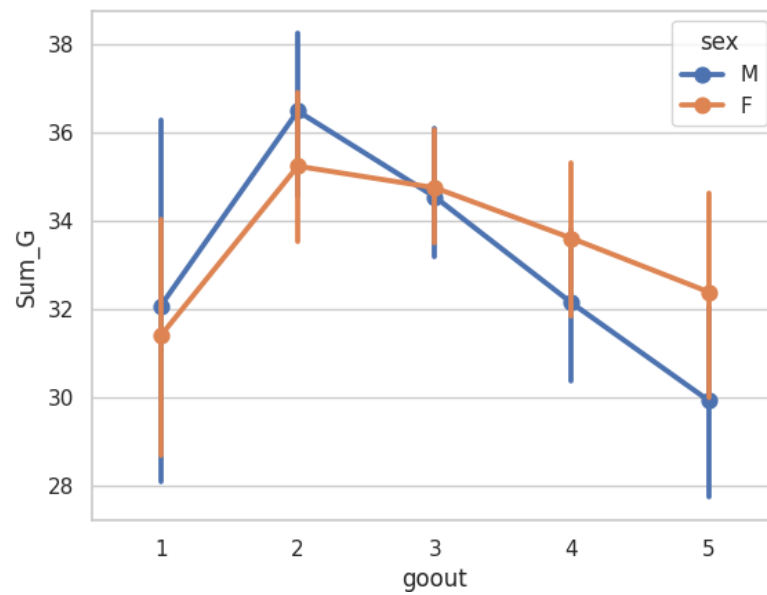
<Axes: xlabel='freetime', ylabel='Sum_G'>



Hình 2. 7

- Data visualization của cột goout so với Sum_G dựa vào tỉ lệ giới tính:

```
sns.pointplot(x="goout", y="Sum_G", hue="sex", data=combined_data)
plt.show()
```

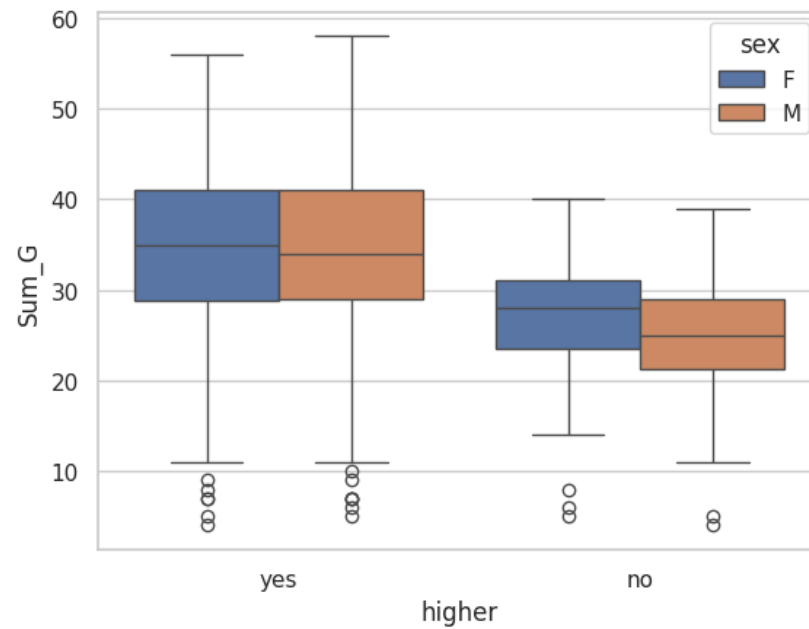


Hình 2. 8

- Data visualization của cột higher so với Sum_G:

```
sns.boxplot(x='higher',y='Sum_G',hue='sex',data=combined_data)
```

<Axes: xlabel='higher', ylabel='Sum_G'>



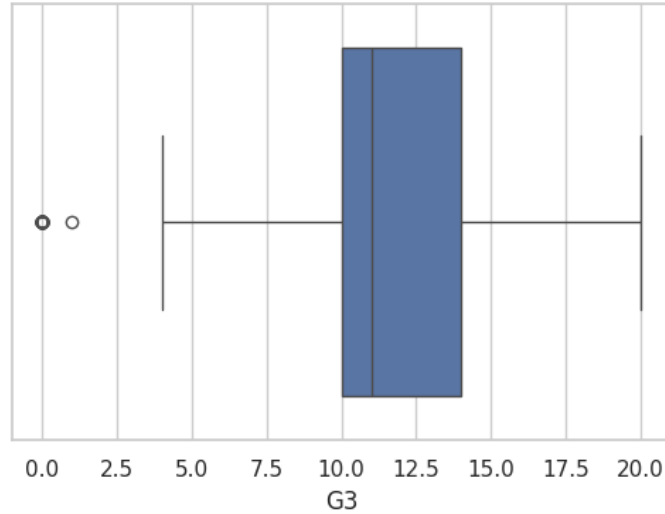
Hình 2. 9

Học sinh chọn việc học cao hơn là những học sinh còn lại.

- Data visualization của cột G3: cho thấy dữ liệu phân bố 10-13

```
plt.figure(figsize=(6, 4))
sns.boxplot(x=combined_data.G3)
```

<Axes: xlabel='G3'>



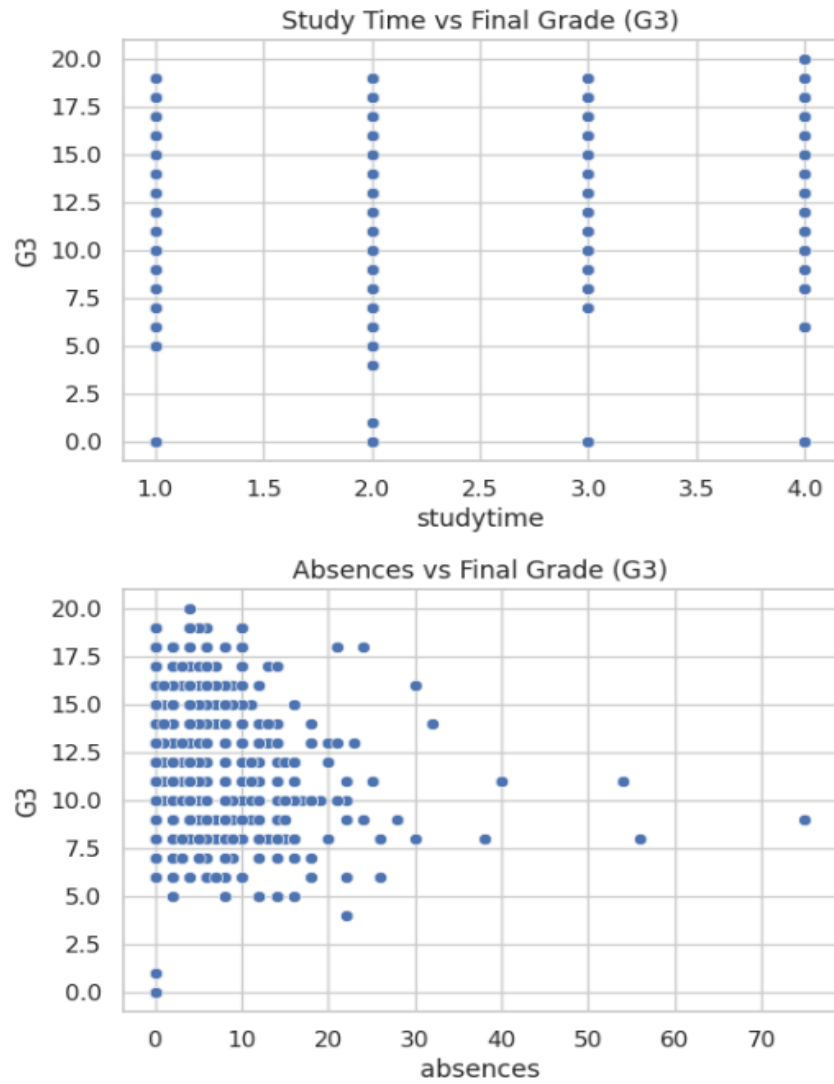
Hình 2. 10

- Data visualization của cột studytime so với G3 và absences so với G3:

```
# Scatter plot between 'studytime' and final grade 'G3'
plt.figure(figsize=(6, 4))
sns.scatterplot(data=combined_data, x='studytime', y='G3')
plt.title('Study Time vs Final Grade (G3)')
plt.show()

# Scatter plot between 'absences' and final grade 'G3'
plt.figure(figsize=(6, 4))
sns.scatterplot(data=combined_data, x='absences', y='G3')
plt.title('Absences vs Final Grade (G3)')
plt.show()
```

Hình 2. 11



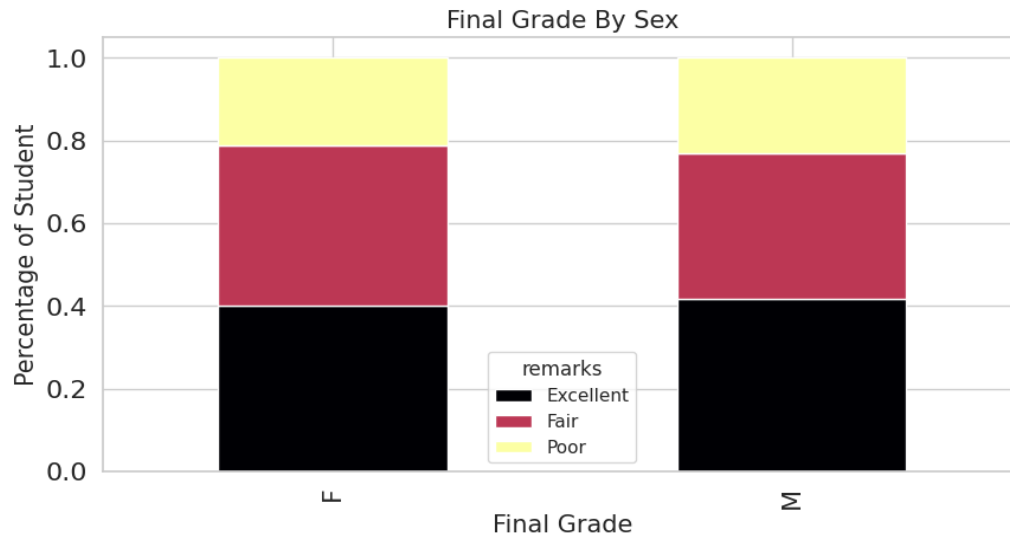
Hình 2. 12

- Data visualization của cột studytime so với G3 và absences so với G3:

```

sen = pd.crosstab(combined_data.remarks, combined_data.sex)
operc = sen.apply(feature)
operc = operc.T
operc.plot.bar(colormap = "inferno", fontsize = 16, figsize = (10,5), stacked = True)
plt.title('Final Grade By Sex', fontsize = 15)
plt.ylabel('Percentage of Student', fontsize = 15)
plt.xlabel('Final Grade', fontsize = 15)
Text(0.5, 0, 'Final Grade')

```



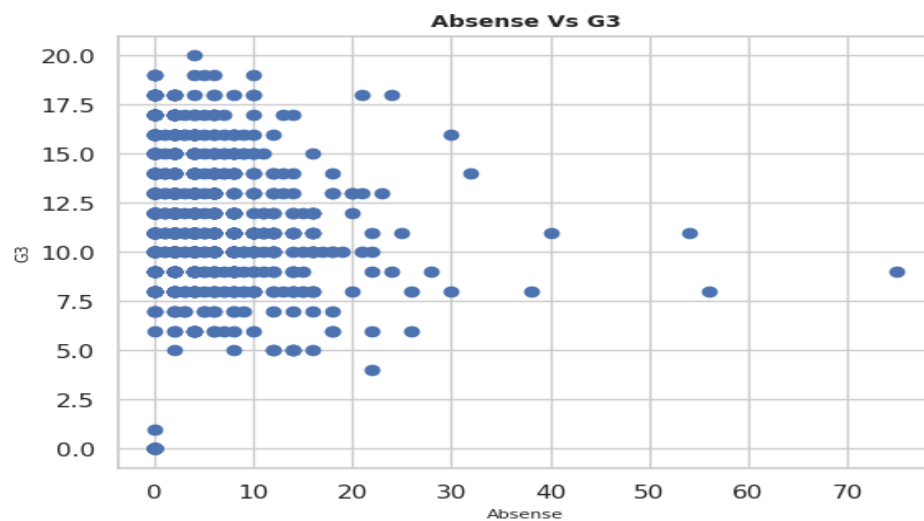
Hình 2. 13

- Data visualization của cột absences so với G3:

```

plt.scatter(x='absences',y='G3',data=combined_data)
plt.title("Absense Vs G3", fontsize=10, fontweight='bold')
plt.xlabel("Absense", fontsize=8)
plt.ylabel("G3", fontsize=8)
plt.show()

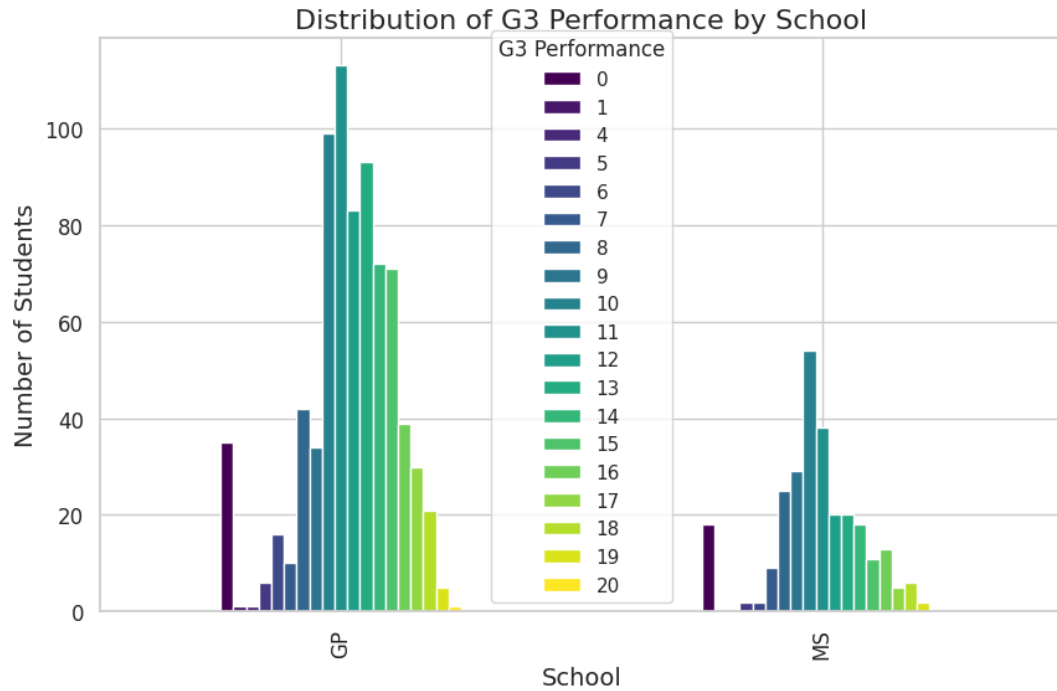
```



Hình 2. 14

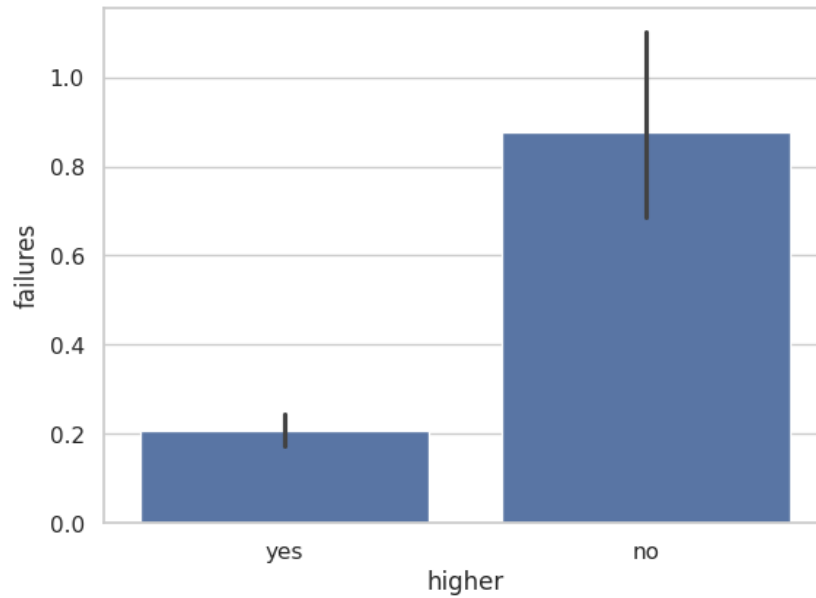
- Data visualization của cột school so với G3:

```
outschool.plot(kind='bar', colormap='viridis', fontsize=12, figsize=(10, 6))
plt.title('Distribution of G3 Performance by School', fontsize=16)
plt.xlabel('School', fontsize=14)
plt.ylabel('Number of Students', fontsize=14)
plt.legend(title='G3 Performance')
plt.show()
```



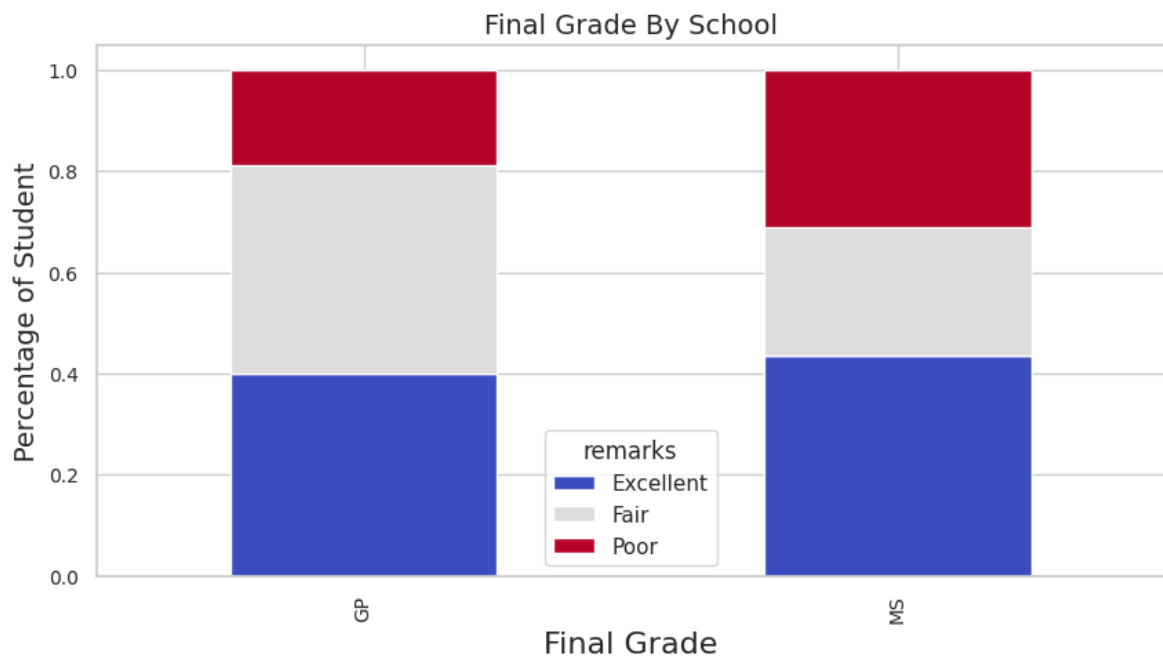
Hình 2. 15

- Data visualization của cột higher so với failures:



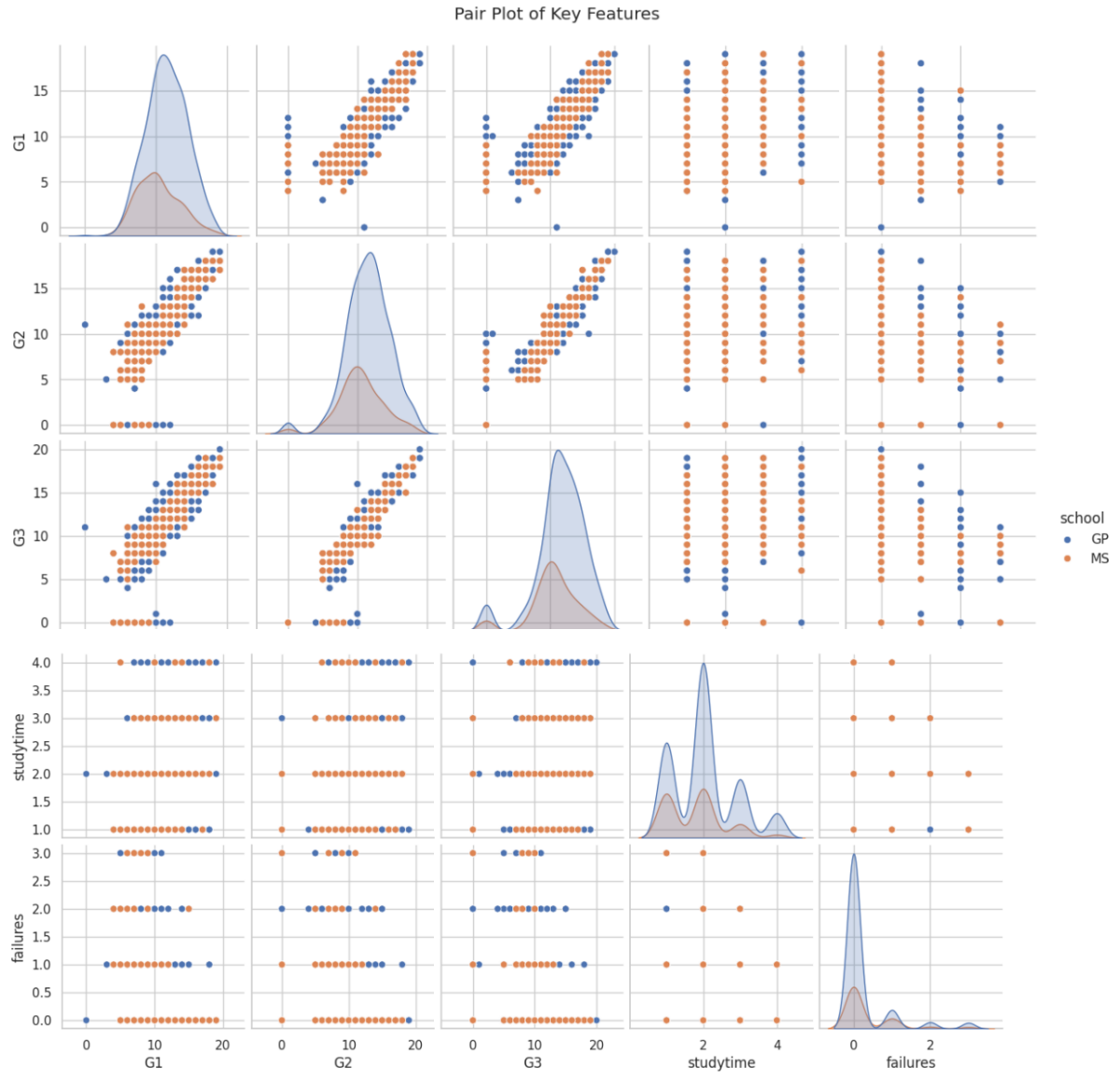
Hình 2. 16

- Data visualization của cột remarks so với G3:



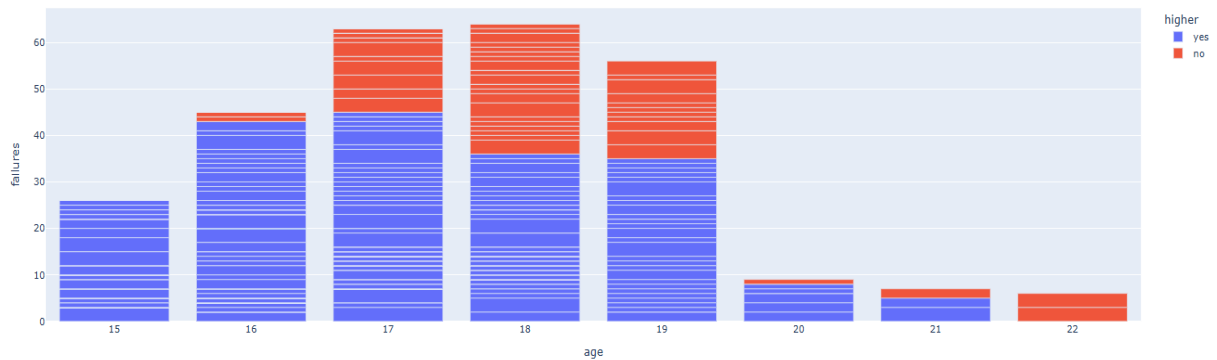
Hình 2. 17

- Data visualization của studytime, failures so với G1, G2, G3 dựa vào school:



Hình 2. 18

- Tổng quan



Hình 2. 19

3. Tiền xử lý dữ liệu

Tiền xử lý dữ liệu là một bước quan trọng trong quy trình xử lý dữ liệu, đặc biệt là đối với các ứng dụng xử lý ngôn ngữ tự nhiên (NLP) và học máy (ML). Trong trường hợp của **type conversation** và **data normalization**, mỗi khía cạnh đóng góp vào việc cải thiện chất lượng dữ liệu và hiệu suất của mô hình. Đối với bộ dữ liệu student + performance được xử lý như sau:

- Đọc dữ liệu:

```
data_math = pd.read_csv('student-mat.csv', delimiter=';')
data_por = pd.read_csv('student-por.csv', delimiter=';')
# Add a column to specify the course
data_math['subject'] = 'math'
data_por['subject'] = 'portuguese'
# Combine the two datasets
combined_data = pd.concat([data_math, data_por], ignore_index=True)
data=combined_data
# Print the first few rows of the combined data
print(combined_data.head())
df = pd.DataFrame(data)
df.to_csv(r"products.csv", index=False)
```

Hình 2. 20

Giải thích:

- Đọc dữ liệu từ hai tệp CSV.
- Thêm cột để phân biệt môn học.
- Ghép hai bảng thành một, giúp dễ dàng thao tác và phân tích trên dữ liệu tổng hợp của cả hai môn học trong cùng một bảng.
- `df.to_csv(r"products.csv", index=False)` lưu DataFrame df vào tệp CSV có tên 'products.csv'

Kết quả cho ra như sau:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	\
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	
3	GP	F	15	U	GT3	T	4	2	health	services	...	
4	GP	F	16	U	GT3	T	3	3	other	other	...	

	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3	subject
0	3	4	1	1	3	6	5	6	6	math
1	3	3	1	1	3	4	5	5	6	math
2	3	2	2	3	3	10	7	8	10	math
3	2	2	1	1	5	2	15	14	15	math
4	3	2	1	2	5	4	6	10	10	math

Hình 2. 21

- Chuẩn hóa dữ liệu:

```
# Step 2: Type Conversion - Convert categorical data to numerical
data = pd.get_dummies(data, drop_first=True)

# Step 3: Data Normalization
# Apply Min-Max Scaling for numerical features only (excluding the target column G3)
scaler = MinMaxScaler()
numerical_features = data.drop(columns=['G3']) # Chỉ chọn các cột không phải là G3
scaled_features = scaler.fit_transform(numerical_features)
data_normalized = pd.DataFrame(scaled_features, columns=numerical_features.columns)
data_normalized['G3'] = data['G3'] # Thêm lại cột G3 vào dataframe

data_normalized.drop_duplicates(inplace=True)
data_normalized.to_csv('preprocessed_student_por.csv', index=False)
```

Hình 2. 22

Giải thích:

- Chuyển đổi dữ liệu dạng phân loại thành dữ liệu dạng số
- Chuẩn hóa dữ liệu (Data Normalization)
- Tạo DataFrame mới cho dữ liệu đã chuẩn hóa và thêm lại cột mục tiêu G3
- Xóa các hàng trùng lặp và lưu dữ liệu vào file CSV

preprocessed_student_por.csv ×

age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	Walc	health	absences	G1
0.4285714285714284	1.0	1.0	0.3333333333333333	0.3333333333333333	0.0	0.75	0.5	0.75	0.0	0.0	0.5	0.08	0.2631578947368421
0.2857142857142856	0.25	0.25	0.0	0.3333333333333333	0.0	1.0	0.5	0.5	0.0	0.0	0.5	0.05333333333333334	0.2631578947368421
0.0	0.25	0.25	0.0	0.3333333333333333	1.0	0.75	0.5	0.25	0.25	0.5	0.5	0.13333333333333333	0.3684210526315789
0.0	1.0	0.5	0.0	0.6666666666666667	0.0	0.5	0.25	0.25	0.0	0.0	1.0	0.02666666666666667	0.7894736842105263
0.1428571428571428	0.75	0.75	0.0	0.3333333333333333	0.0	0.75	0.5	0.25	0.0	0.25	1.0	0.05333333333333334	0.3157894736842105
0.1428571428571428	1.0	0.75	0.0	0.3333333333333333	0.0	1.0	0.75	0.25	0.0	0.25	1.0	0.13333333333333333	0.7894736842105263
0.1428571428571428	0.5	0.5	0.0	0.3333333333333333	0.0	0.75	0.75	0.75	0.0	0.0	0.5	0.0	0.631578947368421
0.2857142857142856	1.0	1.0	0.3333333333333333	0.3333333333333333	0.0	0.75	0.0	0.75	0.0	0.0	0.0	0.08	0.3157894736842105
0.0	0.75	0.5	0.0	0.3333333333333333	0.0	0.75	0.25	0.25	0.0	0.0	0.0	0.0	0.8421052631578947
0.0	0.75	1.0	0.0	0.3333333333333333	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.7368421052631579

G2	school_MS	sex_M	address_U	famsize_LE3	Pstatus_T	Mjob_health	Mjob_other	Mjob_services	Mjob_teacher	Fjob_health	Fjob_other	Fjob_services
0.3157894736842105	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.2631578947368421	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
0.42105263157894735	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
0.7368421052631579	0.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
0.5263157894736842	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
0.7894736842105263	0.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0
0.631578947368421	0.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
0.2631578947368421	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
0.9473684210526315	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0
0.7894736842105263	0.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0

Hình 2. 23

- Tìm cột có các giá trị thiếu:

```
[4] combined_data.columns[combined_data.isna().any()]
```

```
Index([], dtype='object')
```

Kết quả: Danh sách tên các cột có giá trị thiếu trong combined_data

- Thống kê mô tả:

```
[5] combined_data.describe()
```

Giải thích: Lệnh `combined_data.describe()` cung cấp các thống kê mô tả cho các cột số (numeric) trong DataFrame `combined_data`, bao gồm:

- **count:** Số lượng giá trị không thiếu trong mỗi cột.
- **mean:** Giá trị trung bình của mỗi cột.
- **std:** Độ lệch chuẩn, thể hiện mức độ phân tán của dữ liệu.
- **min:** Giá trị nhỏ nhất.
- **25%, 50% (median), 75%:** Các phần trăm vị trí của dữ liệu (quartiles), cho biết sự phân bố dữ liệu.
- **max:** Giá trị lớn nhất.

Kết quả:



	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout
count	1044.000000	1044.000000	1044.000000	1044.000000	1044.000000	1044.000000	1044.000000	1044.000000	1044.000000
mean	16.726054	2.603448	2.387931	1.522989	1.970307	0.264368	3.935824	3.201149	3.156130
std	1.239975	1.124907	1.099938	0.731727	0.834353	0.656142	0.933401	1.031507	1.152575
min	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.000000	1.000000
25%	16.000000	2.000000	1.000000	1.000000	1.000000	0.000000	4.000000	3.000000	2.000000
50%	17.000000	3.000000	2.000000	1.000000	2.000000	0.000000	4.000000	3.000000	3.000000
75%	18.000000	4.000000	3.000000	2.000000	2.000000	0.000000	5.000000	4.000000	4.000000
max	22.000000	4.000000	4.000000	4.000000	4.000000	3.000000	5.000000	5.000000	5.000000

Dalc	Walc	health	absences	G1	G2	G3
1044.000000	1044.000000	1044.000000	1044.000000	1044.000000	1044.000000	1044.000000
1.494253	2.284483	3.543103	4.434866	11.213602	11.246169	11.341954
0.911714	1.285105	1.424703	6.210017	2.983394	3.285071	3.864796
1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000
1.000000	1.000000	3.000000	0.000000	9.000000	9.000000	10.000000
1.000000	2.000000	4.000000	2.000000	11.000000	11.000000	11.000000
2.000000	3.000000	5.000000	6.000000	13.000000	13.000000	14.000000
5.000000	5.000000	5.000000	75.000000	19.000000	19.000000	20.000000

Hình 2. 24

- Tạo remark để đánh giá loại học sinh

```
[8] combined_data['remarks'] = 'na'
     combined_data.iloc[0, 34] = 6
     combined_data.head()
```

↻

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	goout	Dalc	Walc	health	absences	G1	G2	G3	subject	remarks
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	1	1	3	6	5	6	6	math	6
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	3	1	1	3	4	5	5	6	math	na
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	2	2	3	3	10	7	8	10	math	na
3	GP	F	15	U	GT3	T	4	2	health	services	...	2	1	1	5	2	15	14	15	math	na
4	GP	F	16	U	GT3	T	3	3	other	other	...	2	1	2	5	4	6	10	10	math	na

5 rows × 35 columns

Hình 2. 25

```
[9] for i in range(len(combined_data.G3)):
     grade = combined_data.iloc[i, 32]
     if grade < 10:
         combined_data.iloc[i, 34] = 'Poor'
     elif grade > 11 and grade <= 15:
         combined_data.iloc[i, 34] = 'Fair'
     else:
         combined_data.iloc[i, 34] = 'Excellent'
```

```
[10] combined_data.head()
```

↻

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	goout	Dalc	Walc	health	absences	G1	G2	G3	subject	remarks
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	1	1	3	6	5	6	6	math	Poor
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	3	1	1	3	4	5	5	6	math	Poor
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	2	2	3	3	10	7	8	10	math	Excellent
3	GP	F	15	U	GT3	T	4	2	health	services	...	2	1	1	5	2	15	14	15	math	Fair
4	GP	F	16	U	GT3	T	3	3	other	other	...	2	1	2	5	4	6	10	10	math	Excellent

Hình 2. 26

- Dùng để phân loại học sinh theo mức điểm cuối kỳ, chia thành các mức như 'Poor', 'Fair', và 'Excellent'. Điều này giúp dễ dàng nhóm và phân tích hiệu suất học tập của học sinh theo các mức độ khác nhau, thay vì chỉ dựa trên điểm số cụ thể.
- Kiểm tra các cột chỉ số bị trùng lặp và kiểm tra bỏ trống trong tập dữ liệu.


```
[ ] print(combined_data.index.duplicated().sum())
    print(combined_data.index.isnull().sum())
```

```
⇒ 0
   0
```

Hình 2. 27

Kết quả:

- Cho biết có bao nhiêu chỉ số bị trùng lặp và vị trí dữ liệu bị bỏ trống trong DataFrame combined_data. Trả về là 0 tức là không bị trùng lặp và không bị thiếu.
- Kiểm tra các cột hiện đang có trong dữ liệu

```
[13] combined_data.columns
```

```
⇒ Index(['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'Fedu',
        'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime',
        'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery',
        'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc',
        'Walc', 'health', 'absences', 'G1', 'G2', 'G3', 'subject', 'remarks'],
        dtype='object')
```

Hình 2. 28

- Thông tin dữ liệu

```
[14] combined_data.info()
```

Giải thích:

- Số lượng hàng và cột: Hiển thị tổng số hàng và cột của DataFrame.
- Tên và kiểu dữ liệu của từng cột: Cho biết tên cột và kiểu dữ liệu (như int64, float64, object, v.v.).
- Số lượng giá trị không bị thiếu: Hiển thị số lượng giá trị khác NaN trong từng cột, giúp nhận diện các cột có dữ liệu thiếu.
- Bộ nhớ sử dụng: Hiển thị dung lượng bộ nhớ được DataFrame này sử dụng.

- Lệnh này giúp hiểu rõ hơn về cấu trúc dữ liệu, kiểu dữ liệu của từng cột, và nhận biết các cột cần xử lý nếu có giá trị thiếu.

Kết quả:

```

⇒ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 1044 entries, 0 to 1043
Data columns (total 35 columns):
#   Column      Non-Null Count  Dtype
---  -
0   school      1044 non-null   object
1   sex         1044 non-null   object
2   age         1044 non-null   int64
3   address     1044 non-null   object
4   famsize     1044 non-null   object
5   Pstatus     1044 non-null   object
6   Medu        1044 non-null   int64
7   Fedu        1044 non-null   int64
8   Mjob        1044 non-null   object
9   Fjob        1044 non-null   object
10  reason      1044 non-null   object
11  guardian    1044 non-null   object
12  traveltime  1044 non-null   int64
13  studytime   1044 non-null   int64
14  failures    1044 non-null   int64
15  schoolsup    1044 non-null   object
16  famsup      1044 non-null   object
17  paid        1044 non-null   object

18  activities  1044 non-null   object
19  nursery     1044 non-null   object
20  higher      1044 non-null   object
21  internet    1044 non-null   object
22  romantic    1044 non-null   object
23  famrel      1044 non-null   int64
24  freetime    1044 non-null   int64
25  goout       1044 non-null   int64
26  Dalc        1044 non-null   int64
27  Walc        1044 non-null   int64
28  health      1044 non-null   int64
29  absences    1044 non-null   int64
30  G1          1044 non-null   int64
31  G2          1044 non-null   int64
32  G3          1044 non-null   int64
33  subject     1044 non-null   object
34  remarks     1044 non-null   object
dtypes: int64(16), object(19)
memory usage: 285.6+ KB

```

Hình 2. 29

4. Phân chia dữ liệu thành tập train và tập evaluation

```
# Step 4: Define Target Variable
# Assume we classify "G3" as Pass (1) if >=10, else Fail (0)
data_normalized['G3'] = data_normalized['G3'].apply(lambda x: 1 if x >= 10 else 0)
X = data_normalized.drop(columns=['G3']) # Features

y_clf = data_normalized['G3'] # Target variable
y_reg = data['G3'] # Target
# Step 5: Train-Test Split
X_train, X_test, y_train_clf, y_test_clf, y_train_reg, y_test_reg = train_test_split(X, y_clf, y_reg, test_size=0.2, random_state=42)
```

Hình 2. 30

- Chuyển đổi cột G3 trong data_normalized thành nhãn phân loại Pass (1) nếu điểm G3 lớn hơn hoặc bằng 10, và Fail (0) nếu thấp hơn 10.
- X là tập dữ liệu đặc trưng (features) không bao gồm cột G3.
- y_clf là biến mục tiêu đã chuẩn hóa cho bài toán phân loại (Pass/Fail).
- y_reg là biến mục tiêu gốc của G3 (dùng cho hồi quy).

Trong đó:

- train_test_split chia X và hai biến mục tiêu (y_clf cho phân loại và y_reg cho hồi quy) thành hai tập: tập huấn luyện (80%) và tập kiểm tra (20%).
- random_state=42 để đảm bảo việc chia tách nhất quán mỗi khi chạy lại.
- Kết quả là ba cặp tập huấn luyện và kiểm tra: X_train, X_test cho đặc trưng; y_train_clf, y_test_clf cho mô hình phân loại; và y_train_reg, y_test_reg cho mô hình hồi quy.

5. Thực hiện phân loại và hồi quy

5.1 Thực hiện phân loại

Các bài toán được thực hiện trong phần này là: LogisticRegression, DecisionTreeClassifier, NaiveBayesClassifier.

5.1.1 LogisticRegression

Tổng quan: là một thuật toán học máy trong lĩnh vực phân loại, được sử dụng để dự đoán kết quả của một biến phụ thuộc (nhãn) nhị phân hoặc đa lớp dựa trên một tập các biến độc lập (đặc trưng). Được sử dụng phổ biến trong học có giám sát, nó thuộc

nhóm các mô hình hồi quy, nhưng thay vì dự đoán giá trị liên tục như hồi quy tuyến tính, Logistic Regression dự đoán xác suất của một lớp cụ thể, sau đó phân loại dựa trên xác suất này.

- Cơ chế hoạt động: Logistic Regression không dùng hàm hồi quy tuyến tính để dự đoán, mà thay vào đó là hàm sigmoid (hoặc logistic) để biến đổi đầu ra thành giá trị nằm trong khoảng từ 0 đến 1. Công thức hàm sigmoid là:

$$f(s) = \frac{1}{1 + e^{-s}} \triangleq \sigma(s)$$

- Lợi ích và hạn chế:

- Lợi ích: Đơn giản, dễ triển khai, không yêu cầu nhiều tài nguyên tính toán, và dễ dàng giải thích kết quả.
- Hạn chế: Không hoạt động tốt với dữ liệu phi tuyến tính mà không có biến đổi thích hợp, dễ bị ảnh hưởng bởi các outliers.

- Logistic Regression tuy đơn giản nhưng hiệu quả, đặc biệt trong các bài toán phân loại với dữ liệu tuyến tính hoặc có thể biến đổi tuyến tính.

Thực hiện phân loại :

```
[ ] model = LogisticRegression()
    model.fit(X_train, y_train_clf)
    # Evaluate the model
    y_pred = model.predict(X_test)
    accuracy_lgt = accuracy_score(y_test_clf, y_pred)
    report = classification_report(y_test_clf, y_pred)

    print(f"Accuracy: {accuracy_lgt}")
    print("Classification Report:")
    print(report)
```

Hình 2. 31

→ Kết quả:

⇒ Accuracy: 0.8421052631578947

Classification Report:

	precision	recall	f1-score	support
0	0.86	0.53	0.66	60
1	0.84	0.97	0.90	149
accuracy			0.84	209
macro avg	0.85	0.75	0.78	209
weighted avg	0.85	0.84	0.83	209

Hình 2. 32

Nhận xét:

- **Độ chính xác (Accuracy):** Mô hình đạt độ chính xác là 84.2%, tức là mô hình dự đoán đúng khoảng 84% tổng số mẫu thử nghiệm.

- **Precision, Recall và F1-Score:**

- Lớp 0: Precision là 0.86, nghĩa là trong các dự đoán là 0, có 86% là đúng. Recall là 0.53, có nghĩa là mô hình chỉ tìm đúng được 53% các mẫu thực sự thuộc lớp 0. F1-score là 0.66, cho thấy một sự cân bằng giữa precision và recall, nhưng không cao do recall khá thấp.
- Lớp 1: Precision là 0.84, recall là 0.97, và F1-score là 0.90. Điều này cho thấy mô hình rất tốt trong việc dự đoán lớp 1, với recall cao.

- **Macro Average và Weighted Average:**

- Macro avg: Trung bình các chỉ số của cả hai lớp, không quan tâm đến số lượng mẫu. Các chỉ số này cho thấy rằng mô hình hoạt động tốt hơn với lớp 1, nhưng kết quả cho lớp 0 lại khá thấp, ảnh hưởng đến hiệu suất tổng thể.
- Weighted avg: Các chỉ số này có tính đến số lượng mẫu trong mỗi lớp. Vì lớp 1 có nhiều mẫu hơn, weighted avg gần với kết quả của lớp 1.

5.1.2 DecisionTreeClassifier

Tổng quan: DecisionTreeClassifier là một thuật toán học máy dùng trong bài toán phân loại có giám sát, hoạt động dựa trên cấu trúc của cây quyết định (decision tree). Trong mô hình này, dữ liệu sẽ được phân loại thông qua việc chia nhỏ liên tiếp các đặc trưng của dữ liệu, đến khi đạt được các phân nhóm nhỏ có độ thuần nhất cao.

- Cơ chế hoạt động: DecisionTreeClassifier xây dựng một cây phân loại từ trên xuống dưới. Mỗi nút trong cây tương ứng với một điều kiện trên một đặc trưng, và mỗi nhánh của cây là kết quả của điều kiện đó. Việc chia nhỏ dữ liệu được thực hiện dựa trên các tiêu chí chọn đặc trưng tốt nhất.

- Cấu trúc cây quyết định gồm:

- Nút gốc (Root Node): Nút đầu tiên, nơi bắt đầu chia dữ liệu.
- Nút quyết định (Decision Node): Nút trung gian, nơi dữ liệu được chia tiếp theo điều kiện dựa trên một đặc trưng.
- Nút lá (Leaf Node): Nút cuối cùng, chứa nhãn dự đoán.

- Lợi ích và hạn chế

- Lợi ích: dễ hiểu, trực quan và dễ diễn giải kết quả, không yêu cầu chuẩn hóa dữ liệu, hoạt động tốt với dữ liệu phi tuyến.
- Hạn chế: dễ bị quá khớp (overfitting) nếu cây quá sâu hoặc có quá nhiều nhánh, nhạy cảm với nhiễu trong dữ liệu, có thể tạo ra cây phức tạp và không tổng quát.

Thực hiện phân loại bằng DecisionTreeClassifier:

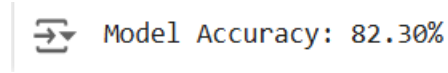
```
[ ] clf = DecisionTreeClassifier(random_state=42)
    clf.fit(X_train, y_train_clf)

    #Evaluate the model
    y_pred = clf.predict(X_test)
    accuracy_dt = accuracy_score(y_test_clf, y_pred)

    print(f"Model Accuracy: {accuracy_dt * 100:.2f}%")
```

Hình 2. 33

→ Kết quả:



Hình 2. 34

Nhận xét:

- Độ chính xác (Accuracy): Mô hình Decision Tree đạt độ chính xác là 85.65%

5.1.3 NaiveBayesClassifier:

Tổng quan: là một thuật toán phân loại có giám sát dựa trên Định lý Bayes, với giả định "ngây thơ" rằng các đặc trưng của dữ liệu là độc lập với nhau. Điều này có nghĩa là mỗi đặc trưng của dữ liệu đóng góp vào xác suất của lớp nhãn độc lập với các đặc trưng khác. Naive Bayes đặc biệt hữu ích cho các bài toán phân loại văn bản và có tốc độ nhanh, hiệu quả khi xử lý với lượng dữ liệu lớn.

- Cơ chế hoạt động: Naive Bayes dựa trên Định lý Bayes

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- Lợi ích và hạn chế

- Lợi ích: nhanh, tính toán đơn giản và yêu cầu ít tài nguyên, hoạt động tốt với các bài toán phân loại văn bản, như phân loại email spam và phân tích cảm xúc, hiệu quả với dữ liệu có tính độc lập giữa các đặc trưng.
- Hạn chế: hiệu suất giảm khi các đặc trưng phụ thuộc lẫn nhau, cần giả định dữ liệu phù hợp với phân phối xác suất cụ thể (như phân phối Gaussian trong Gaussian Naive Bayes).

Thực hiện phân loại với NaiveBayesClassifier:

```
[ ] nb_clf = GaussianNB()
    nb_clf.fit(X_train, y_train_clf)

    y_pred = nb_clf.predict(X_test)
    accuracy_nb = accuracy_score(y_test_clf, y_pred)

    print(f"Model Accuracy: {accuracy_nb * 100:.2f}%")
```

Hình 2. 35

→ Kết quả:

↔ Model Accuracy: 82.30%

Hình 2. 36

5.2 Thực hiện hồi quy

Các bài toán được thực hiện trong phần này là: LinearRegression, SVR, K-Nearest Neighbor.

5.2.1 LinearRegression

Tổng quan: LinearRegression là một thuật toán học máy thuộc loại hồi quy, được sử dụng để dự đoán giá trị của một biến phụ thuộc (nhãn) dựa trên một hoặc nhiều biến độc lập (đặc trưng). Thuật toán này giả định rằng có một mối quan hệ tuyến tính giữa các đặc trưng đầu vào và biến đầu ra, và cố gắng tìm một đường thẳng để dự đoán giá trị đầu ra một cách tốt nhất.

- Cơ chế hoạt động: Linear Regression tìm cách tối thiểu hóa sai số dự đoán giữa giá trị dự đoán và giá trị thực tế bằng cách xác định các hệ số tối ưu (weights) cho các đặc trưng. Đường hồi quy tuyến tính có dạng:

$$f(\mathbf{x}) = w_1x_1 + w_2x_2 + w_3x_3 + w_0$$

Trong đó:

y: Giá trị dự đoán (biến đầu ra).

w: Các hệ số hồi quy của từng đặc trưng.

x : Các biến đầu vào (đặc trưng).

w_0 : Hệ số điều chỉnh (bias).

- Lợi ích và hạn chế

- Lợi ích: đơn giản, dễ hiểu và dễ triển khai, không yêu cầu quá nhiều tài nguyên tính toán, dễ dàng diễn giải kết quả, hữu ích trong các bài toán mà giải thích mô hình là quan trọng.
- Hạn chế: không hoạt động tốt với các mối quan hệ phi tuyến tính, dễ bị ảnh hưởng bởi các ngoại lệ (outliers) và nhiễu trong dữ liệu, giả định mối quan hệ tuyến tính giữa các biến đầu vào và đầu ra, điều này không phải lúc nào cũng đúng trong thực tế.

Thực hiện hồi quy bằng Linear Regression:

```

modelRe = LinearRegression()
modelRe.fit(X_train, y_train_reg)

y_pred = modelRe.predict(X_test)
mse_ln = mean_squared_error(y_test_reg, y_pred)
r2 = r2_score(y_test_reg, y_pred)
print("LinearRegression")
print(f"Mean Squared Error: {mse_ln}")
print(f"R^2 Score: {r2}")

```

Hình 2. 37

→ Kết quả:

```

LinearRegression
Mean Squared Error: 3.0973651766402974
R^2 Score: 0.79966356659595

```

Hình 2. 38

Nhận xét:

- Mean Squared Error (MSE): MSE là 3.097, cho thấy mức độ sai số trung bình bình phương giữa dự đoán của mô hình và giá trị thực tế.

- R^2 Score: R^2 score là 0.80, có nghĩa là mô hình giải thích được khoảng 80% phương sai của dữ liệu kiểm tra. Đây là một R^2 khá cao, cho thấy mô hình có khả năng giải thích tốt cho mối quan hệ giữa biến đầu vào và biến mục tiêu.

Kết luận: Mô hình Linear Regression đã thể hiện khá tốt với R^2 đạt 0.80 và MSE ở mức chấp nhận được, tuy nhiên có thể cần điều chỉnh hoặc thử mô hình khác nếu độ chính xác cao hơn là yêu cầu chính.

5.2.2 SVR:

Tổng quan: SVR (Support Vector Regression) là một thuật toán học máy dùng cho bài toán hồi quy, dựa trên khái niệm cơ bản của máy vector hỗ trợ (Support Vector Machine - SVM). Mục tiêu của SVR là tìm một siêu phẳng để dự đoán giá trị của biến đầu ra dựa trên các biến đầu vào, trong đó nó cố gắng giữ sai số trong một ngưỡng nhất định gọi là epsilon, và tối ưu hóa khoảng cách của các điểm dữ liệu gần siêu phẳng nhất.

- Cơ chế hoạt động: Trong SVR, thay vì chỉ tối thiểu hóa sai số giữa giá trị dự đoán và giá trị thực, thuật toán cố gắng tối ưu hóa theo cách để tất cả các điểm dữ liệu nằm trong một dải băng có chiều rộng ϵ xung quanh siêu phẳng. SVR tìm kiếm siêu phẳng với điều kiện:

- Tối thiểu hóa sai số nhưng giữ trong phạm vi ϵ .
- Giảm thiểu độ phức tạp của mô hình bằng cách tối thiểu hóa biên độ giữa các support vector và siêu phẳng.

- Lợi ích và hạn chế

- Lợi ích: có khả năng xử lý tốt các dữ liệu phi tuyến khi sử dụng kernel, hiệu quả với các dữ liệu có nhiễu nhờ dải băng ϵ cho phép kiểm soát sai số, tổng quát hóa tốt cho dữ liệu mới nhờ các support vectors.

- Hạn chế: phức tạp và đòi hỏi tài nguyên tính toán cao, đặc biệt với dữ liệu lớn và phi tuyến, lựa chọn tham số (như ϵ và các tham số kernel) cần cẩn trọng để tránh overfitting hoặc underfitting, khó diễn giải hơn so với hồi quy tuyến tính.

Thực hiện hồi quy bằng SVR:

```
[ ] from sklearn.svm import SVR
    from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
    # Step 6: Train SVR Model
    # Using RBF kernel for regression
    svr_model = SVR(kernel='rbf', C=1.0, epsilon=0.1)
    svr_model.fit(X_train, y_train_reg)

    # Step 7: Make Predictions and Evaluate Model
    y_pred = svr_model.predict(X_test)

    # Evaluation Metrics
    mse_svr = mean_squared_error(y_test_reg, y_pred)
    mae = mean_absolute_error(y_test_reg, y_pred)
    r2 = r2_score(y_test_reg, y_pred)

    print(f"Mean Squared Error: {mse_svr}")
    print(f"Mean Absolute Error: {mae:.2f}")
    print(f"R2 Score: {r2:.2f}")
```

Hình 2. 39

→ Kết quả:

```
⇒ Mean Squared Error: 6.210977854585553
   Mean Absolute Error: 1.57
   R2 Score: 0.60
```

Hình 2. 40

SVR là một thuật toán mạnh mẽ và linh hoạt cho bài toán hồi quy, đặc biệt khi dữ liệu có cấu trúc phức tạp hoặc phi tuyến. Tuy nhiên, do độ phức tạp cao, SVR phù hợp nhất khi làm việc với dữ liệu có quy mô vừa phải và có yêu cầu tổng quát hóa cao.

5.2.3 K - Nearest Neighbor:

Tổng quan: K-Nearest Neighbor (KNN) là một thuật toán học máy đơn giản và dễ hiểu, được sử dụng cho cả bài toán phân loại và hồi quy trong học có giám sát. Thuật toán này không yêu cầu xây dựng mô hình hoặc giả thiết về phân phối dữ liệu mà dựa trên khoảng cách giữa các điểm dữ liệu trong không gian đặc trưng để đưa ra dự đoán, do đó KNN còn được gọi là phương pháp "lazy learning".

- Cơ chế hoạt động: Nguyên lý cơ bản của KNN là tìm KKK điểm dữ liệu gần nhất (neighbors) của điểm cần dự đoán và sử dụng các điểm đó để đưa ra quyết định phân loại hoặc hồi quy. Các bước của KNN bao gồm:

- Tính khoảng cách giữa điểm cần dự đoán và tất cả các điểm dữ liệu trong tập huấn luyện. Khoảng cách phổ biến nhất là khoảng cách Euclid, nhưng cũng có thể dùng các khoảng cách khác như Manhattan hoặc Minkowski.
- Chọn K điểm gần nhất: Xác định KKK điểm dữ liệu gần nhất (với giá trị KKK thường được xác định trước).
- Với phân loại: KNN dự đoán nhãn của điểm cần dự đoán dựa trên nhãn của đa số các điểm láng giềng gần nhất.
- Với hồi quy: KNN dự đoán giá trị đầu ra bằng cách trung bình giá trị của KKK điểm gần nhất.

- Lợi ích và hạn chế

- Lợi ích: đơn giản, dễ triển khai, không yêu cầu giả định phức tạp về dữ liệu, khả năng mô hình hóa tốt cho các bài toán không tuyến tính, có thể áp dụng cho cả bài toán phân loại và hồi quy.
- Hạn chế: tính toán chậm với dữ liệu lớn, vì phải tính khoảng cách với toàn bộ tập dữ liệu, độ chính xác phụ thuộc vào lựa chọn tham số KKK và cách đo khoảng cách, không hoạt động tốt khi các đặc trưng có độ chênh lệch lớn về giá trị, do đó thường cần chuẩn hóa dữ liệu.

Thực hiện hồi quy với K-Nearest Neighbor:


```
[ ] from sklearn.neighbors import KNeighborsRegressor
    from sklearn.metrics import mean_squared_error, r2_score

    knn_regressor = KNeighborsRegressor(n_neighbors=5)
    knn_regressor.fit(X_train, y_train_reg)

    y_pred = knn_regressor.predict(X_test)
    mse_knn = mean_squared_error(y_test_reg, y_pred)
    rmse = np.sqrt(mse_knn)
    r2 = r2_score(y_test_reg, y_pred)

    print(f"Model MSE: {mse_knn:.2f}")
    print(f"Model RMSE: {rmse:.2f}")
    print(f"Model R^2 Score: {r2:.2f}")
```

Hình 2. 41

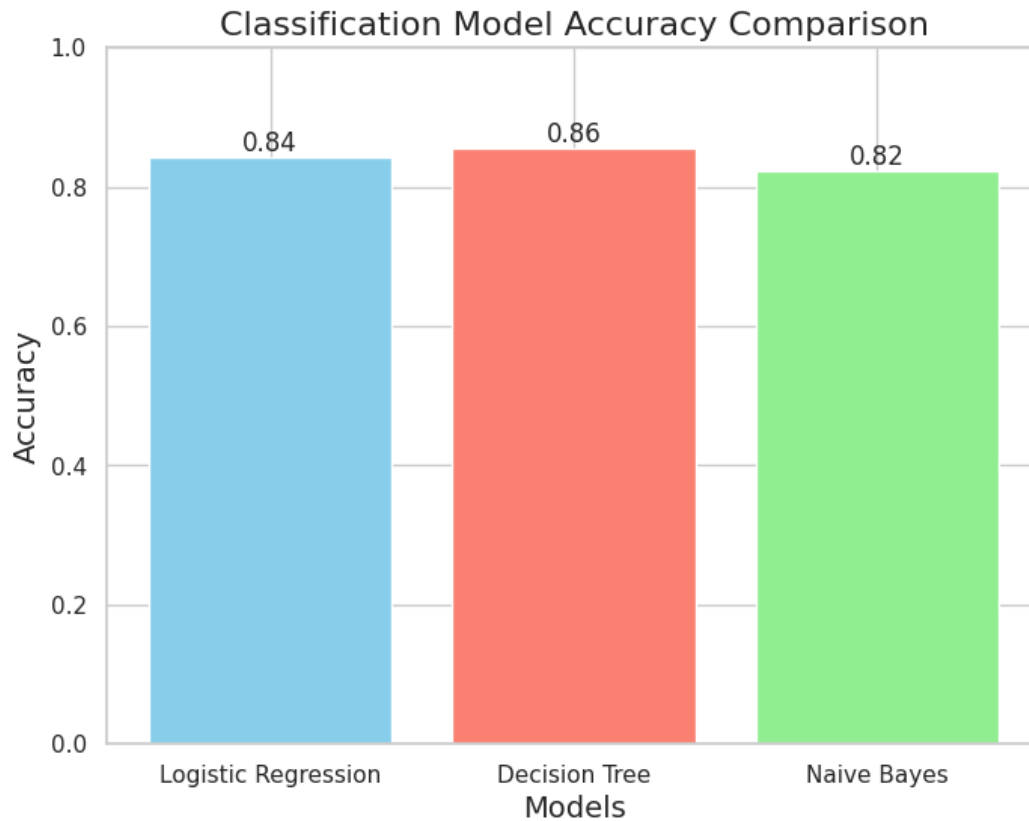


```
Model MSE: 12.00
Model RMSE: 3.46
Model R^2 Score: 0.22
```

Hình 2. 42

6. Đánh giá kết quả và so sánh các phương pháp

6.1 So sánh các phương pháp phân loại với nhau:



Hình 2. 43

Nhận xét về kết quả so sánh các mô hình phân loại:***1. Độ chính xác của các mô hình:***

- Logistic Regression: Đạt độ chính xác 84.21%, cho thấy mô hình này có hiệu suất khá tốt trong việc phân loại dữ liệu. Mô hình Logistic Regression thường phù hợp với các bài toán có ranh giới tuyến tính giữa các lớp.

- Decision Tree: Đạt độ chính xác cao nhất là 85.65%. Decision Tree có khả năng chia dữ liệu theo các quy tắc phức tạp và phù hợp hơn nếu dữ liệu không tuyến tính. Độ chính xác cao nhất của Decision Tree ở đây cho thấy mô hình này nắm bắt tốt các đặc trưng trong dữ liệu.

- Naive Bayes: Đạt độ chính xác 82.30%, thấp nhất trong ba mô hình. Naive Bayes giả định các đặc trưng độc lập với nhau, điều này có thể không hoàn toàn đúng với bộ dữ liệu, dẫn đến độ chính xác thấp hơn.

2. So sánh mô hình qua biểu đồ:

- Từ biểu đồ, có thể thấy rằng Decision Tree có độ chính xác cao nhất, tiếp theo là Logistic Regression và cuối cùng là Naive Bayes.

- Tuy rằng sự khác biệt giữa Logistic Regression và Decision Tree không lớn, nhưng Decision Tree vẫn cho thấy khả năng phân loại tốt hơn một chút trong trường hợp này.

3. Đánh giá tổng quan:

- Decision Tree là mô hình tốt nhất với dữ liệu hiện tại. Tuy nhiên, mô hình này có nguy cơ overfitting nếu không được kiểm soát tốt độ sâu. Bạn có thể sử dụng các kỹ thuật như pruning để hạn chế overfitting.

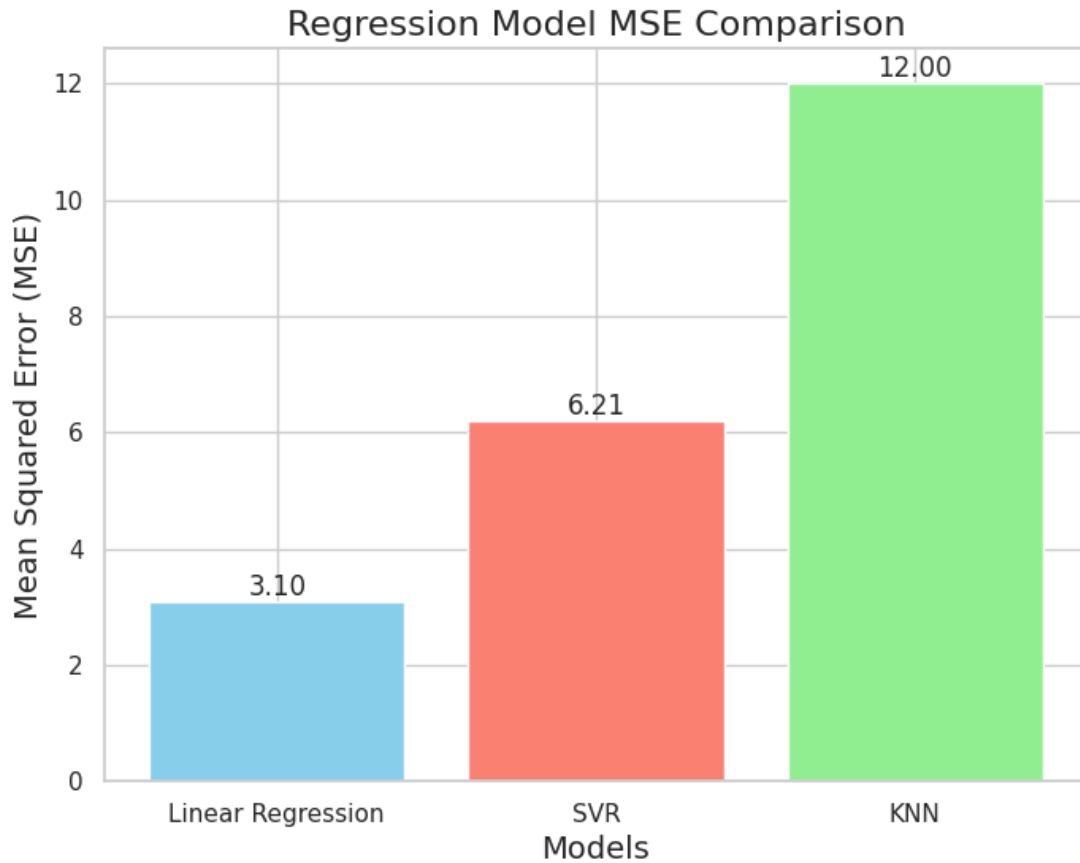
- Logistic Regression vẫn là một lựa chọn tốt nếu cần một mô hình đơn giản, dễ hiểu, và nếu bài toán có ranh giới phân loại tương đối tuyến tính.

- Naive Bayes có thể không phải là lựa chọn tối ưu cho dữ liệu này do giả định các đặc trưng độc lập, nhưng vẫn có thể hữu ích nếu yêu cầu tính toán nhanh với độ chính xác chấp nhận được.

4. Kết luận:

Decision Tree đạt hiệu suất tốt nhất và có thể được ưu tiên nếu hiệu suất là yêu cầu chính. Tuy nhiên, nếu đơn giản hóa và khả năng giải thích mô hình được ưu tiên, Logistic Regression cũng là một lựa chọn hợp lý.

6.2 So sánh các phương pháp hồi quy với nhau:



Hình 2. 44

Nhận xét về kết quả so sánh các mô hình hồi quy:

1. Mean Squared Error (MSE) của các mô hình:

- Linear Regression: Có MSE thấp nhất là 3.10, cho thấy mô hình hồi quy tuyến tính có khả năng dự đoán tốt nhất trong ba mô hình với độ sai lệch thấp hơn so với SVR và KNN. Điều này cho thấy mô hình Linear Regression phù hợp với dữ liệu và có thể đã nắm bắt được mối quan hệ tuyến tính giữa các đặc trưng đầu vào và biến mục tiêu.

- Support Vector Regressor (SVR): Có MSE là 6.21, cao hơn gần gấp đôi so với Linear Regression. Mô hình SVR thường thích hợp hơn với các dữ liệu có mối quan hệ phi tuyến tính nhẹ, nhưng trong trường hợp này, mô hình Linear Regression lại thể hiện tốt hơn.

- K-Nearest Neighbors Regressor (KNN): Có MSE cao nhất là 12.00, cho thấy mô hình này không hoạt động tốt với dữ liệu hiện tại. Điều này có thể do KNN dễ bị ảnh hưởng bởi các điểm nhiễu (outliers) hoặc phân bố của dữ liệu. KNN thường hiệu quả hơn với các tập dữ liệu phi tuyến tính và có ít nhiễu.

2. So sánh các mô hình qua biểu đồ:

- Biểu đồ MSE cho thấy sự khác biệt rõ rệt giữa các mô hình. Linear Regression có MSE thấp nhất, trong khi KNN có MSE cao nhất.

- Kết quả này chỉ ra rằng Linear Regression là lựa chọn tối ưu nhất trong ba mô hình cho tập dữ liệu hiện tại, đặc biệt nếu có mối quan hệ tuyến tính mạnh.

3. Đánh giá tổng quan:

- Linear Regression vượt trội nhất trong ba mô hình, cho thấy khả năng giải thích tốt với độ sai số thấp nhất. Nếu bộ dữ liệu có mối quan hệ tuyến tính, đây là lựa chọn rất hợp lý.

- SVR có thể được cải thiện bằng cách tinh chỉnh các tham số, như kernel hoặc C, để tìm ra thiết lập tối ưu hơn.

- KNN có thể chưa phù hợp với dữ liệu này, nhưng có thể điều chỉnh số lượng láng giềng k hoặc chuẩn hóa dữ liệu để cải thiện kết quả.

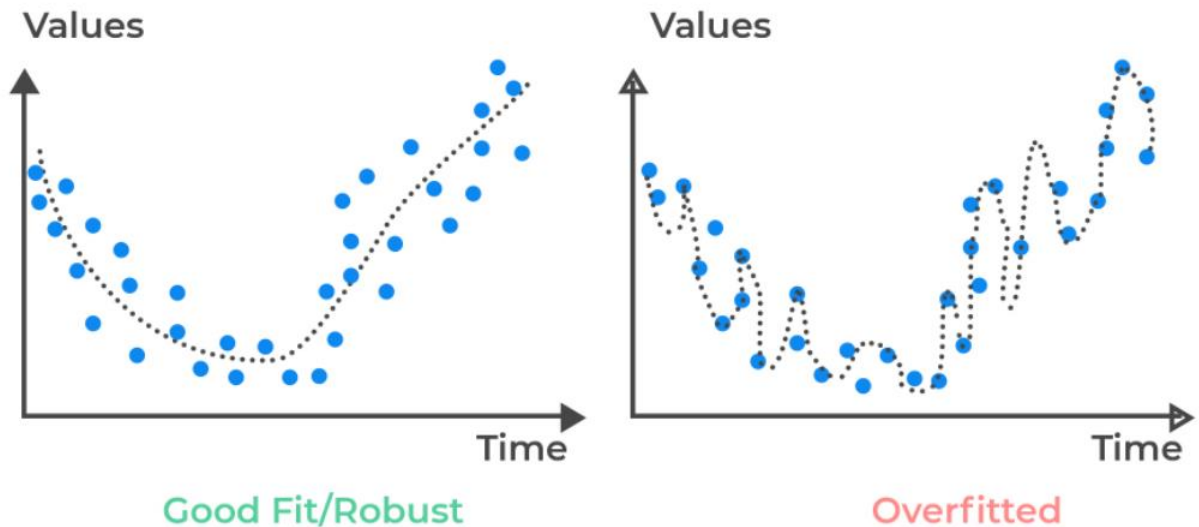
4. Kết luận:

- Linear Regression có MSE thấp nhất và hiệu suất tốt nhất, là lựa chọn hàng đầu với dữ liệu này.

- Nếu cần thử nghiệm thêm, có thể tối ưu tham số cho SVR hoặc thử các mô hình hồi quy phi tuyến tính khác nếu dữ liệu có mối quan hệ phức tạp hơn.

CHƯƠNG 3 – OVERFITTING

1. Vấn đề Overfitting:



Hình 3. 1

Nguồn : [Overfitting and Methods of Addressing it - CFA, FRM, and Actuarial Exams Study Notes](#)

Overfitting là một hiện tượng khi mà mô hình ta huấn luyện quá khớp với dữ liệu huấn luyện vì mô hình hình học thuộc quá nhiều dữ liệu mà ta đã huấn luyện. Việc này sẽ ảnh hưởng đến mô hình hoạt động rất tốt trên dữ liệu train mà lại không hiệu quả đối với dữ liệu mà ta kiểm tra và dữ liệu mới.

2. Hiện tượng overfitting xảy ra khi :

Dữ liệu ít: Mô hình khá phức tạp nhưng ta chỉ huấn luyện mô hình với tập dữ liệu quá nhỏ.

Không áp dụng regularization: Mô hình huấn luyện không được áp dụng các kỹ thuật regularization để kiểm soát kích thước và độ phức tạp dẫn đến việc mô hình sẽ học nhiều và kỹ đối với các chi tiết của dữ liệu khi huấn luyện.

Độ phức tạp mô hình cao: Mô hình có số lượng tham số lớn dẫn đến mô hình sẽ học thuộc quá nhiều chi tiết cụ thể thay vì học những đặc trưng tổng quát cần thiết hơn.

3. Cách nhận dạng khi mô hình bị overfitting:

Khi huấn luyện mô hình, tập dữ liệu sẽ được chia làm hai: Dữ liệu huấn luyện và dữ liệu kiểm tra. Dữ liệu huấn luyện sẽ đảm nhận vai trò để huấn luyện mô hình và dữ liệu kiểm tra sẽ đo lường độ chính xác của mô hình sau khi đã được huấn luyện.

Overfitting sẽ rất dễ nhận dạng khi dựa vào độ chính xác khi bạn huấn luyện mô hình. Nếu độ chính xác của mô hình khá hoàn hảo trên dữ liệu huấn luyện nhưng khi ta áp dụng mô hình này lên dữ liệu mới (dữ liệu kiểm tra) thì mô hình có độ chính xác giảm hơn so với ban đầu.

VD: Mô hình có độ chính xác trên dữ liệu huấn luyện khá cao, như 97%. Nhưng mô hình lại có độ chính xác trên dữ liệu kiểm tra thấp hơn rõ như 73%. Qua hai thông số trên thì ta có thể nhận định rằng mô hình này đã bị overfitting.

4. Phương pháp giải quyết overfitting:

- Prunning: Đây là phương pháp được dùng đặc biệt cho giảm overfitting của decision tree. Phương pháp này nhắm tới mục tiêu làm cho cây trở nên đơn giản hơn bằng cách loại bỏ đi những phần không mang lại khả năng dự đoán đáng kể. Phương pháp prunning này sẽ làm giảm đi kích thước của cây nhưng lại không làm giảm đi độ chính xác khi dự đoán của mô hình. Có hai loại prunning được sử dụng rộng rãi để giảm overfitting này:

- Pre-prunning: Để ngăn chặn overfitting khi mà mô hình được huấn luyện quá khớp với dữ liệu huấn luyện, từ đó sẽ ảnh hưởng đến khi mô hình tiếp xúc với dữ liệu mới thì phương pháp pre-prunning sẽ sử dụng kỹ thuật “Early Stopping” nhằm ngăn chặn trước khi mô hình được huấn luyện trở nên quá phức tạp.

- Post-pruning: Ở phương pháp này, các nút và cây con sẽ được thay thế bằng các lá để giảm độ phức tạp, qua đó sẽ làm giảm đi kích thước mà còn cải thiện độ chính xác khi dự đoán trên tập dữ liệu mới.

- K-Fold cross validation: Phương pháp này được sử dụng phổ biến để lấy mẫu để đánh giá mô hình học máy khi mà dữ liệu không được nhiều. Cách thường được sử dụng của phương pháp này đó là sử dụng k như một biến đại diện cho số nhóm dữ liệu hay còn được gọi là tập con được chia ra. Phương pháp sẽ huấn luyện và xác thực mô hình k lần nhưng mỗi lần sẽ sử dụng một tập con khác nhau để xác thực, qua đó sẽ đảm bảo rằng mô hình sẽ không quá phụ thuộc vào bất kỳ phần nào của dữ liệu, ngăn chặn kịp thời tình trạng overfitting.

- Regularization: Một nhược điểm lớn của cross-validation là số lượng training runs tỉ lệ thuận với k . Điều đáng nói là mô hình polynomial như trên chỉ có một tham số cần xác định là bậc của đa thức. Trong các bài toán Machine Learning, lượng tham số cần xác định thường lớn hơn nhiều, và khoảng giá trị của mỗi tham số cũng rộng hơn nhiều, chưa kể đến việc có những tham số có thể là số thực. Như vậy, việc chỉ xây dựng một mô hình thôi cũng là đã rất phức tạp rồi. Có một cách giúp số mô hình cần huấn luyện giảm đi nhiều, thậm chí chỉ một mô hình. Cách này có tên gọi chung là regularization. Regularization, một cách cơ bản, là thay đổi mô hình một chút để tránh overfitting trong khi vẫn giữ được tính tổng quát của nó (tính tổng quát là tính mô tả được nhiều dữ liệu, trong cả tập training và test). Một cách cụ thể hơn, ta sẽ tìm cách di chuyển nghiệm của bài toán tối ưu hàm mất mát tới một điểm gần nó. Hướng di chuyển sẽ là hướng làm cho mô hình ít phức tạp hơn mặc dù giá trị của hàm mất mát có tăng lên một chút.

5. Thực hiện code bổ sung giải pháp overfitting

5.1 Classification Overfitting

Code giải quyết Overfitting của LogisticRegression:

```
[ ] #LogisticsRegression OverFitting
    param_grid = {'C': [0.01, 0.1, 1, 10, 100]}
    grid = GridSearchCV(LogisticRegression(random_state=42), param_grid, cv=5)
    grid.fit(X_train, y_train_clf)

    # Best model
    model = grid.best_estimator_
    # Step 6: Evaluate the model
    y_pred = model.predict(X_test)
    accuracy_lgt_ovf = accuracy_score(y_test_clf, y_pred)
    report = classification_report(y_test_clf, y_pred)

    print(f"Best Regularization Parameter: {grid.best_params_['C']}")
    print(f"Model Accuracy: {accuracy_lgt_ovf * 100:.2f}%")
```

➡ Best Regularization Parameter: 10
Model Accuracy: 86.60%

Hình 3. 2

Code giải quyết Overfitting của DecisionTree

```
#Decision Tree OverFitting
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
param_grid = {
    'max_depth': [3, 5, 10, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 5, 10],
    'max_leaf_nodes': [10, 20, 30, None]
}

grid_search = GridSearchCV(DecisionTreeClassifier(random_state=42), param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train_clf)

clf = grid_search.best_estimator_

y_pred = clf.predict(X_test)
accuracy_dt_ovf = accuracy_score(y_test_clf, y_pred)

print("Best parameters found: ", grid_search.best_params_)
print(f"Model Accuracy: {accuracy_dt_ovf * 100:.2f}%")
```

➡ Best parameters found: {'max_depth': 5, 'max_leaf_nodes': 10, 'min_samples_leaf': 5, 'min_samples_split': 2}
Model Accuracy: 87.56%

Hình 3. 3

:

Code giải quyết Overfitting của Gaussian

```
[ ] param_grid = {
    'var_smoothing': np.logspace(-12, -8, num=10) # Try different smoothing values
}

# Set up GridSearchCV with k-fold cross-validation
grid_search = GridSearchCV(GaussianNB(), param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train_clf)

# Best parameters found
print("Best Parameters:", grid_search.best_params_)

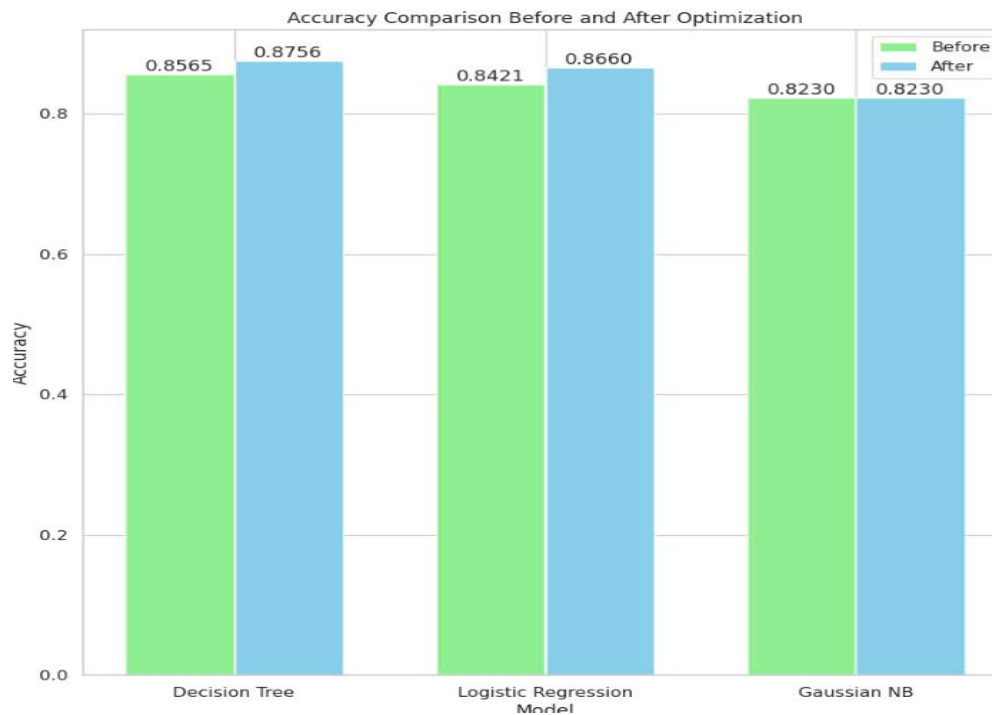
# Evaluate the best model on test set
best_nb_model = grid_search.best_estimator_
y_pred = best_nb_model.predict(X_test)
accuracy_nb_ovf = accuracy_score(y_test_clf, y_pred)

print(f"Test Set Accuracy with Best Parameters: {accuracy_nb_ovf * 100}%")
```

➡ Best Parameters: {'var_smoothing': 1e-12}
 Test Set Accuracy with Best Parameters: 82.29665071770334%

Hình 3. 4

So sánh kết quả trước và sau khi dùng overfitting:



Hình 3. 5

Nhận xét:**1. Decision Tree:**

- Trước điều chỉnh: Độ chính xác của Decision Tree là 85.65%, đã là cao nhất trong các mô hình ban đầu.

- Sau điều chỉnh: Độ chính xác tăng lên 87.56%, cao nhất trong tất cả các mô hình sau điều chỉnh.

Nhận xét: Điều chỉnh tham số (ví dụ, kiểm soát độ sâu cây hoặc sử dụng pruning) giúp Decision Tree tổng quát hóa tốt hơn trên tập dữ liệu thử nghiệm, dẫn đến hiệu suất tăng thêm. Điều này cho thấy Decision Tree đã khắc phục phần nào vấn đề overfitting và đã nắm bắt tốt hơn các đặc điểm quan trọng của dữ liệu.

2. Logistic Regression:

- Trước điều chỉnh: Độ chính xác là 84.21%.

- Sau điều chỉnh: Đạt được độ chính xác 86.60% sau khi chọn tham số C tối ưu, giúp Logistic Regression gần bắt kịp Decision Tree.

Nhận xét: Việc điều chỉnh tham số C giúp Logistic Regression cân bằng tốt hơn giữa độ phức tạp của mô hình và tính tổng quát, hạn chế overfitting. Hiệu suất cải thiện của Logistic Regression cho thấy điều chỉnh này đặc biệt hiệu quả trong các mô hình hồi quy logistic.

3. Gaussian Naive Bayes:

- Trước và sau điều chỉnh: Độ chính xác vẫn giữ nguyên ở 82.30%.

- Nhận xét: Gaussian Naive Bayes không bị ảnh hưởng nhiều bởi overfitting vì giả định các đặc trưng độc lập. Mô hình này dường như đã đạt giới hạn của nó với dữ liệu hiện tại, do đó không có thay đổi sau điều chỉnh.

Tổng quan về hiệu quả của việc điều chỉnh để tránh overfitting:

- Decision Tree và Logistic Regression đều cho thấy sự cải thiện đáng kể về độ chính xác sau khi áp dụng điều chỉnh. Điều này chứng tỏ rằng các tham số điều chỉnh

đóng vai trò quan trọng trong việc tối ưu hóa các mô hình phân loại phức tạp, giúp giảm overfitting.

- Gaussian Naive Bayes không thay đổi sau điều chỉnh, phản ánh giới hạn của mô hình khi giả định về tính độc lập giữa các đặc trưng không hoàn toàn phù hợp với dữ liệu.

Kết luận:

Điều chỉnh để tránh overfitting đã cải thiện rõ rệt hiệu suất của Decision Tree và Logistic Regression, với Decision Tree hiện là mô hình hiệu quả nhất ở mức 87.56%. Điều này nhấn mạnh tầm quan trọng của việc điều chỉnh tham số, đặc biệt khi sử dụng các mô hình dễ overfit trên dữ liệu phức tạp.


5.2 Regression Overfitting

Code giải quyết overfitting của LinearRegression:

```
[ ] #LinearRegression OverFitting
poly_model = make_pipeline(PolynomialFeatures(degree=1), LinearRegression())
poly_model.fit(X_train, y_train_reg)
# Cross-Validation
cv_scores = cross_val_score(poly_model, X_train, y_train_reg, cv=5, scoring='r2')
print("Cross-Validation R^2 Scores:", cv_scores)
print("Mean CV R^2 Score:", cv_scores.mean())

y_pred_poly = poly_model.predict(X_test)
mse_poly_ovf = mean_squared_error(y_test_reg, y_pred_poly)
r2_poly = r2_score(y_test_reg, y_pred_poly)

print("Polynomial Regression (degree=1)")
print(f"Mean Squared Error: {mse_poly_ovf}")
print(f"R^2 Score: {r2_poly}")
```

 Cross-Validation R^2 Scores: [0.82891327 0.75108693 0.84105252 0.90336951 0.86678345]
Mean CV R^2 Score: 0.8382411376722503
Polynomial Regression (degree=1)
Mean Squared Error: 3.097365176640295
R^2 Score: 0.7996635665959501

Hình 3. 6

Code giải quyết overfitting của SVR:

```

▶ param_grid = {
    'C': [0.1, 1.0, 10.0],
    'epsilon': [0.01, 0.1, 0.5],
    'kernel': ['rbf', 'linear']
}

# Set up GridSearchCV with 5-fold cross-validation directly
grid_search = GridSearchCV(SVR(), param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train_reg)

# Best parameters found
print("Best Parameters:", grid_search.best_params_)

# Use the best model to make predictions on the test set
best_svr_model = grid_search.best_estimator_
y_pred = best_svr_model.predict(X_test)

# Calculate evaluation metrics
mse_svr_ovf = mean_squared_error(y_test_reg, y_pred)
mae = mean_absolute_error(y_test_reg, y_pred)
r2 = r2_score(y_test_reg, y_pred)

# Print results
print(f"Mean Squared Error: {mse_svr_ovf:.2f}")
print(f"Mean Absolute Error: {mae:.2f}")
print(f"R2 Score: {r2:.2f}")

```

⇒ Best Parameters: {'C': 10.0, 'epsilon': 0.01, 'kernel': 'linear'}
Mean Squared Error: 3.06
Mean Absolute Error: 0.91
R2 Score: 0.80

Hình 3. 7

Code giải quyết overfitting của K-Nearest Neighbor:

```
[ ] from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

param_grid = {
    'n_neighbors': [3, 5, 7, 9, 11],
    'weights': ['uniform', 'distance'],
    'p': [1, 2]
}


grid_search = GridSearchCV(KNeighborsRegressor(), param_grid, cv=10, scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train_reg)

# Get the best parameters from the grid search
print("Best Parameters:", grid_search.best_params_)

# Evaluate the model with the best parameters on the test set
best_knn_model = grid_search.best_estimator_
y_pred = best_knn_model.predict(X_test)

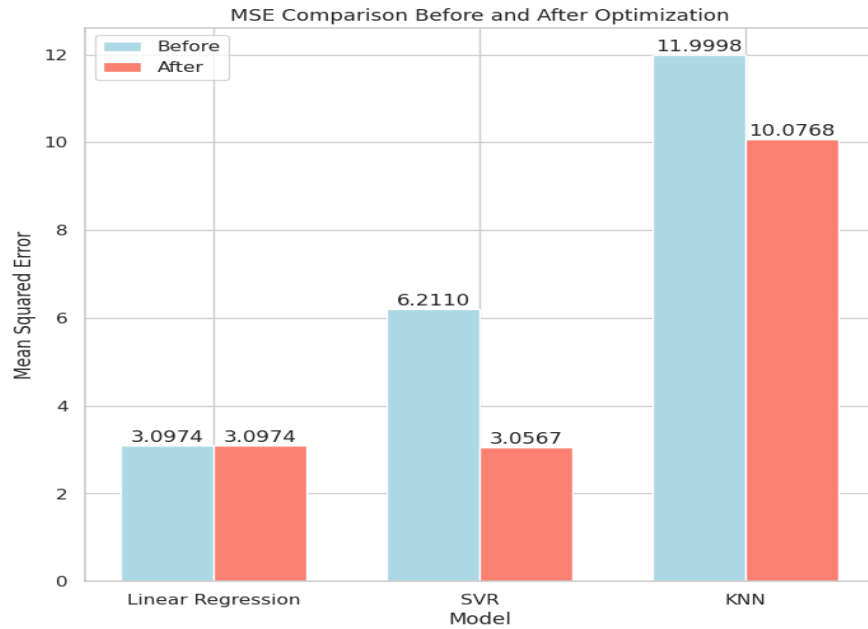
# Calculate MSE, RMSE, and R^2 score for the test set
mse_knn_ovf = mean_squared_error(y_test_reg, y_pred)
rmse = np.sqrt(mse_knn)
r2 = r2_score(y_test_reg, y_pred)

# Print results
print(f"Model MSE: {mse_knn_ovf:.2f}")
print(f"Model RMSE: {rmse:.2f}")
print(f"Model R^2 Score: {r2:.2f}")
```

 Best Parameters: {'n_neighbors': 11, 'p': 1, 'weights': 'distance'}
 Model MSE: 10.08
 Model RMSE: 3.46
 Model R^2 Score: 0.35

Hình 3. 8

So sánh kết quả trước và sau khi dùng overfitting:



Hình 3. 9

Nhận xét:**1. Linear Regression:**

- Trước và sau điều chỉnh: Mean Squared Error (MSE) vẫn giữ nguyên ở 3.0974.

- Nhận xét: Linear Regression không thay đổi (hoặc thay đổi nhỏ) vì mô hình này thường có ít nguy cơ overfitting với dữ liệu tuyến tính. Đây là mô hình đơn giản và không điều chỉnh quá phức tạp, nên việc điều chỉnh tham số (nếu có) sẽ ít ảnh hưởng đến hiệu suất của nó trong bài toán này.

2. Support Vector Regressor (SVR):

- Trước điều chỉnh: MSE là 6.2110, cao hơn đáng kể so với Linear Regression.

- Sau điều chỉnh: MSE giảm xuống 3.0567, thấp hơn cả Linear Regression, cho thấy mô hình đã cải thiện đáng kể sau khi điều chỉnh.

Nhận xét: Việc điều chỉnh tham số (như kernel, C, hoặc gamma) giúp SVR tối ưu hóa mối quan hệ phi tuyến tính trong dữ liệu và giảm sai số. Điều này cho thấy SVR

có thể là một lựa chọn mạnh mẽ cho dữ liệu này sau khi điều chỉnh để kiểm soát overfitting, đặc biệt khi dữ liệu có sự phức tạp và không hoàn toàn tuyến tính.

3. *K-Nearest Neighbors Regressor (KNN)*:

- Trước điều chỉnh: MSE là 11.9998, cao nhất trong ba mô hình.
- Sau điều chỉnh: MSE giảm xuống 10.0768 sau khi điều chỉnh, nhưng vẫn cao hơn so với Linear Regression và SVR.

Nhận xét: Kết quả cải thiện nhẹ cho thấy rằng điều chỉnh tham số k hoặc các biện pháp tiền xử lý dữ liệu (như chuẩn hóa) giúp KNN giảm sai số. Tuy nhiên, do KNN dễ bị ảnh hưởng bởi outliers và tính chất cục bộ của dữ liệu, hiệu suất vẫn chưa tốt như Linear Regression hoặc SVR.

Tổng quan về hiệu quả của việc điều chỉnh để tránh overfitting:

- SVR cho thấy sự cải thiện đáng kể nhất sau điều chỉnh, với MSE giảm mạnh từ 6.2110 xuống 3.0567, biến nó thành mô hình tốt nhất. Điều này cho thấy các tham số của SVR có thể điều chỉnh tốt để kiểm soát overfitting và phù hợp với dữ liệu phi tuyến tính.
- KNN đã giảm MSE nhưng vẫn không đạt hiệu quả cao, thể hiện rằng mô hình này có giới hạn nhất định với dữ liệu này.

Kết luận:

SVR hiện là mô hình hiệu quả nhất sau khi điều chỉnh, với MSE thấp nhất (3.0567). Việc điều chỉnh để tránh overfitting đã mang lại hiệu quả rõ rệt cho SVR, cho thấy nó là lựa chọn mạnh mẽ nhất cho bài toán này.

CHƯƠNG 4 – FEATURE SELECTION USING CORRELATION ANALYSIS

1. Giới thiệu về Feature Selection (Trích chọn đặc trưng)

- Feature Selection là một quá trình chọn lọc một tập con chứa các thuộc tính liên quan để sử dụng trong quá trình xây dựng mô hình. Trong học máy và xử lý dữ liệu, đây là một kỹ thuật nhằm chọn ra các đặc trưng (features) quan trọng nhất từ tập dữ liệu đầu vào, giúp mô hình học máy tối ưu hóa hiệu suất và giảm bớt chi phí tính toán.

- Mục tiêu của feature selection:

- Đơn giản hóa model, dễ sử dụng, dễ thông hiểu hơn.
- Giảm thời gian huấn luyện mô hình.
- Tránh curse of dimensionality (lời nguyền số chiều).
- Làm data tương thích hơn với mô hình.
- Nhúng các thông tin với tính chất đối xứng bên trong không gian input.

- Ý tưởng: loại bỏ những cột, những feature hoặc redundant (thừa thãi), hoặc irrelevant (không liên quan).

1.1 Các phương pháp chính

- Feature Selection có 3 phương pháp chính:

- Filter Method (Phương pháp lọc)
- Wrapper Method (Phương pháp đóng gói)
- Embedded Method (Phương pháp nhúng)

1.2 Feature selection using correlation analysis

- *Feature selection using correlation analysis* là phương pháp lọc, dựa trên các thống kê của dữ liệu như hệ số tương quan (correlation) để đánh giá mức độ liên quan của từng đặc trưng với biến mục tiêu. Các đặc trưng có giá trị tương quan thấp hoặc không có ảnh hưởng lớn đến biến mục tiêu sẽ bị loại bỏ.

- Tính hệ số tương quan: Hệ số tương quan, chẳng hạn như Pearson, Spearman, hoặc Kendall, được tính giữa các đặc trưng và biến mục tiêu.

- Pearson Correlation: Dùng cho dữ liệu tuyến tính, hệ số này đo lường mức độ tuyến tính giữa hai biến.
- Spearman Correlation: Dùng khi dữ liệu không tuyến tính hoặc khi biến đầu vào có quan hệ đơn điệu.

- Sử dụng correlation analysis trong feature selection giúp loại bỏ đặc trưng không liên quan, giảm đa cộng tuyến, tăng hiệu quả tính toán, giảm overfitting, đơn giản hóa mô hình và tiền xử lý nhanh – hiệu quả.

2. Thực nghiệm

2.1 Tính ma trận tương quan và vẽ biểu đồ heatmap

- Ma trận tương quan là một bảng số liệu thể hiện mức độ liên hệ giữa các đặc trưng trong tập dữ liệu. Các giá trị được tính toán dựa trên mối quan hệ giữa từng cặp đặc trưng, từ đó cho thấy chúng ảnh hưởng đến nhau như thế nào.

- Sau khi có ma trận tương quan, chúng ta sử dụng biểu đồ heatmap để dễ dàng quan sát mức độ liên quan giữa các đặc trưng. Heatmap là một dạng biểu đồ màu sắc, với mỗi màu thể hiện một mức độ tương quan nhất định. Các đặc trưng có liên hệ chặt chẽ với nhau sẽ nổi bật trên biểu đồ, giúp ta nhận diện những cặp đặc trưng có khả năng trùng lặp thông tin hoặc những đặc trưng không đóng góp nhiều giá trị.

- Màu sắc trong heatmap biểu thị mức độ tương quan:

- Màu tối (có thể là đỏ hoặc xanh đậm) thường biểu thị mối quan hệ tương quan mạnh mẽ (gần 1 hoặc -1).
- Màu sáng hơn (màu vàng nhạt hoặc xanh nhạt) biểu thị tương quan yếu hoặc không có mối quan hệ.

- Mối quan hệ giữa các đặc trưng và mục tiêu: Heatmap có thể giúp bạn dễ dàng nhận diện những đặc trưng nào có sự liên kết mạnh với biến mục tiêu, ví dụ như sự tương quan giữa các đặc trưng như failures, G1, G2 với G3 (Điểm cuối khóa học).

```
[84] data_math = pd.read_csv('student-mat.csv', delimiter=';')
data_por = pd.read_csv('student-por.csv', delimiter=';')

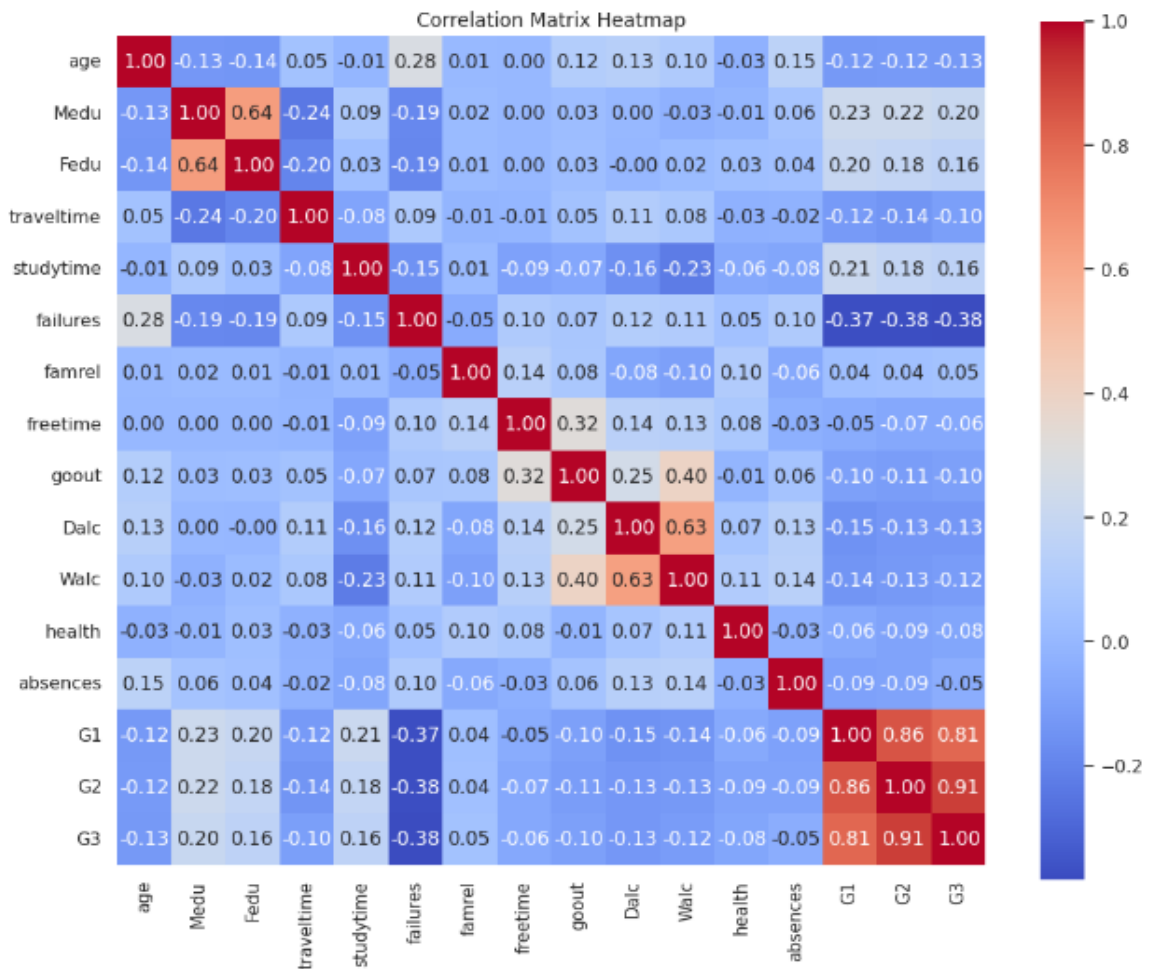
data_math['subject'] = 'math'
data_por['subject'] = 'portuguese'

combined_data = pd.concat([data_math, data_por], ignore_index=True)

numerical_features = combined_data.select_dtypes(include=['int64', 'float64']).columns.tolist()
corr_matrix = combined_data[numerical_features].corr()

plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f', cbar=True, square=True)
plt.title('Correlation Matrix Heatmap')
plt.show()
```

Hình 4. 1



Hình 4. 2

2.2 Tính toán hệ số tương quan giữa target và các đặc trưng còn lại

```

▶ target = abs(corr_matrix['G3'])
  print(target)

↗ age      0.125282
  Medu     0.201472
  Fedu     0.159796
  traveltime 0.102627
  studytime 0.161629
  failures  0.383145
  famrel    0.054461
  freetime  0.064890
  goout     0.097877
  Dalc      0.129642
  Walc      0.115740
  health    0.080079
  absences  0.045671
  G1        0.809142
  G2        0.910743
  G3        1.000000
  Name: G3, dtype: float64

```

Hình 4. 3

- Danh sách này sẽ hiển thị hệ số tương quan giữa biến mục tiêu và các đặc trưng, giúp bạn nhận biết những đặc trưng có mối quan hệ mạnh với biến mục tiêu và có thể ưu tiên chúng khi xây dựng mô hình học máy.

2.3 Chọn đặc trưng

```

▶ correlations = corr_matrix['G3'].abs().drop('G3')

threshold = 0.3
strong_corr_features = correlations[correlations > threshold].index.tolist()
print("Features strongly correlated with G3:", strong_corr_features)

↗ Features strongly correlated with G3: ['failures', 'G1', 'G2']

```

Hình 4. 4

- Sử dụng ma trận tương quan để chọn ra các đặc trưng có mối tương quan mạnh với biến mục tiêu G3, bằng cách thiết lập một ngưỡng cụ thể.

Absolute value	Interpretation
0.8 to 1	Very strong relationship
0.6 to 0.8	Strong relationship
0.4 to 0.6	Moderate relationship
0.2 to 0.4	Weak relationship
0 to 0.2	Very weak correlation or no correlation

Hình 4. 5

- Mức độ tương quan dựa trên giá trị tuyệt đối của hệ số tương quan Pearson: Khi thiết lập ngưỡng (như 0.3 trong đoạn mã), ta có thể tập trung vào các đặc trưng có mối quan hệ ít nhất là yếu hoặc trung bình với biến mục tiêu, loại bỏ những đặc trưng có mối quan hệ rất yếu hoặc không có mối quan hệ (giá trị từ 0 đến 0.2).

2.4 Huấn luyện lại mô hình

```

correlation_threshold = 0.3
data_normalized['G3'] = data_normalized['G3'].apply(lambda x: 1 if x >= 10 else 0)
correlation_matrix = data_normalized.corr()
correlation_with_target = correlation_matrix['G3'].drop('G3')
selected_features = correlation_with_target[correlation_with_target.abs() > correlation_threshold].index

print("Các đặc trưng được chọn với ngưỡng tương quan > 0.3:")
print(selected_features)

X = data_normalized[selected_features] # Tập đặc trưng đã chọn
y_clf = data_normalized['G3']         # Biến mục tiêu cho phân loại
y_reg = data['G3']                    # Biến mục tiêu gốc cho hồi quy

# Chia tập Train-Test cho cả hai loại bài toán
X_train, X_test, y_train_clf, y_test_clf, y_train_reg, y_test_reg = train_test_split(
    X, y_clf, y_reg, test_size=0.2, random_state=42)

print("Số lượng đặc trưng được chọn:", len(selected_features))

```

Các đặc trưng được chọn với ngưỡng tương quan > 0.3:
 Index(['failures', 'G1', 'G2'], dtype='object')
 Số lượng đặc trưng được chọn: 3

Hình 4. 6

- Xây dựng lại một DataFrame mới với các đặc trưng được chọn dựa trên ngưỡng tương quan, tạo ra các bộ dữ liệu đầu vào và nhãn mục tiêu mới, chia dữ liệu lại và huấn luyện mô hình, tương tự như các bước đã thực hiện trước đó.

2.5 Áp dụng vào mô hình Classification

```
[71] # LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train_clf)
# Evaluate the model
y_pred = model.predict(X_test)
accuracy_lgt_fs = accuracy_score(y_test_clf, y_pred)
report = classification_report(y_test_clf, y_pred)

print(f"Model Accuracy: {accuracy_lgt_fs * 100:.2f}%")
```

⇒ Model Accuracy: 83.73%

```
[72] # DecisionTree
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train_clf)

#Evaluate the model
y_pred = clf.predict(X_test)
accuracy_dt_fs = accuracy_score(y_test_clf, y_pred)

print(f"Model Accuracy: {accuracy_dt_fs * 100:.2f}%")
```

⇒ Model Accuracy: 88.04%

```
[73] # NB Classifier
nb_clf = GaussianNB()
nb_clf.fit(X_train, y_train_clf)

y_pred = nb_clf.predict(X_test)
accuracy_nb_fs = accuracy_score(y_test_clf, y_pred)

print(f"Model Accuracy: {accuracy_nb_fs * 100:.2f}%")
```

⇒ Model Accuracy: 87.08%

Hình 4. 7

```

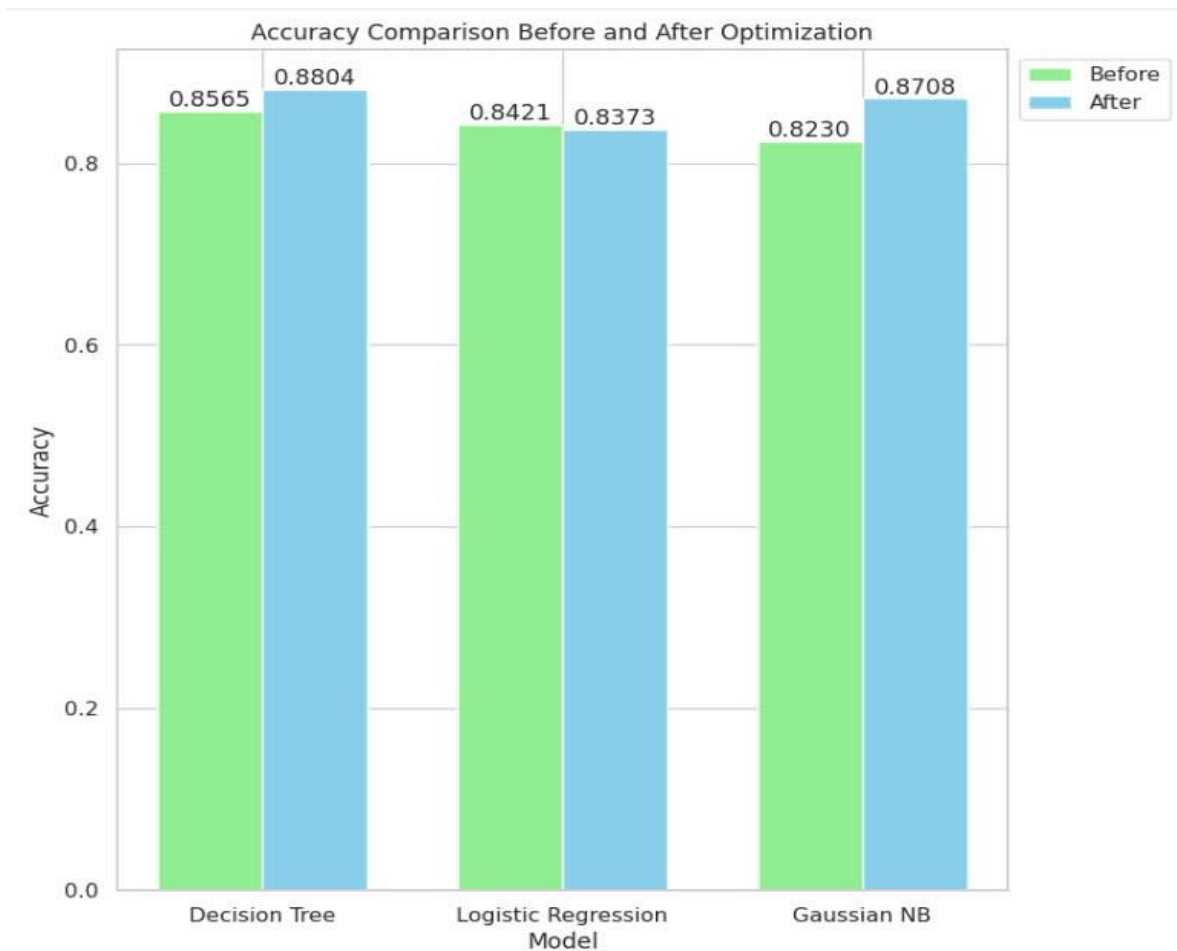
▶ print(f"{'Model':<20} | {'Before':<10} | {'After':<10}")
print("-" * 46)

print(f"{'DecisionTree':<20} | {accuracy_dt:<10.4f} | {accuracy_dt_fs:<10.4f}")
print(f"{'LogisticRegression':<20} | {accuracy_lgt:<10.4f} | {accuracy_lgt_fs:<10.4f}")
print(f"{'GaussianNB':<20} | {accuracy_nb:<10.4f} | {accuracy_nb_fs:<10.4f}")

```

Model	Before	After
DecisionTree	0.8565	0.8804
LogisticRegression	0.8421	0.8373
GaussianNB	0.8230	0.8708

Hình 4. 8



Hình 4. 9

- Nhận xét:

- DecisionTree: Tăng từ 0.8565 lên 0.8804, cải thiện hiệu suất nhờ loại bỏ đặc trưng không liên quan.
- LogisticRegression: Giảm nhẹ từ 0.8421 xuống 0.8373, có thể do mất đi thông tin quan trọng khi chọn đặc trưng.
- GaussianNB: Tăng từ 0.8230 lên 0.8708, cải thiện đáng kể nhờ giảm nhiễu và làm rõ mối quan hệ giữa các đặc trưng.

- Kết luận:

- Việc áp dụng phân tích tương quan có tác động tích cực đến một số mô hình (như DecisionTree và GaussianNB) nhờ vào việc loại bỏ các đặc trưng không cần thiết, giúp cải thiện độ chính xác.
- Tuy nhiên, LogisticRegression không có sự cải thiện rõ rệt và thậm chí có thể giảm nhẹ hiệu suất, điều này có thể phản ánh sự mất đi thông tin quan trọng từ các đặc trưng bị loại bỏ.

2.6 Áp dụng vào mô hình Regression

```
[74] # LinearRegression
modelRe = LinearRegression()
modelRe.fit(X_train, y_train_reg)

y_pred = modelRe.predict(X_test)
mse_ln_fs = mean_squared_error(y_test_reg, y_pred)
r2 = r2_score(y_test_reg, y_pred)
print("LinearRegression")
print(f"Mean Squared Error: {mse_ln_fs}")
```



LinearRegression
Mean Squared Error: 2.9939701395187566



```
# SVR
svr_model = SVR(kernel='rbf', C=1.0, epsilon=0.1)
svr_model.fit(X_train, y_train_reg)

# Step 7: Make Predictions and Evaluate Model
y_pred = svr_model.predict(X_test)

# Evaluation Metrics
mse_svr_fs = mean_squared_error(y_test_reg, y_pred)
mae = mean_absolute_error(y_test_reg, y_pred)
r2 = r2_score(y_test_reg, y_pred)

print(f"Mean Squared Error: {mse_svr_fs}")
```



Mean Squared Error: 3.3056527100862634

```
[90] # K-Nearest Neighbor
knn_regressor = KNeighborsRegressor(n_neighbors=5)
knn_regressor.fit(X_train, y_train_reg)

y_pred = knn_regressor.predict(X_test)
mse_knn_fs = mean_squared_error(y_test_reg, y_pred)
rmse = np.sqrt(mse_knn)
r2 = r2_score(y_test_reg, y_pred)

print(f"Model MSE: {mse_knn_fs}")
```



Model MSE: 3.6704306220095693

Hình 4. 10

```

▶ print(f"{'Model':<20} | {'Before':<10} | {'After':<10}")
  print("-" * 46)

  print(f"{'LinearRegression':<20} | {mse_ln:<10.4f} | {mse_ln_fs:<10.4f}")
  print(f"{'SVR':<20} | {mse_svr:<10.4f} | {mse_svr_fs:<10.4f}")
  print(f"{'KNN':<20} | {mse_knn:<10.4f} | {mse_knn_fs:<10.4f}")

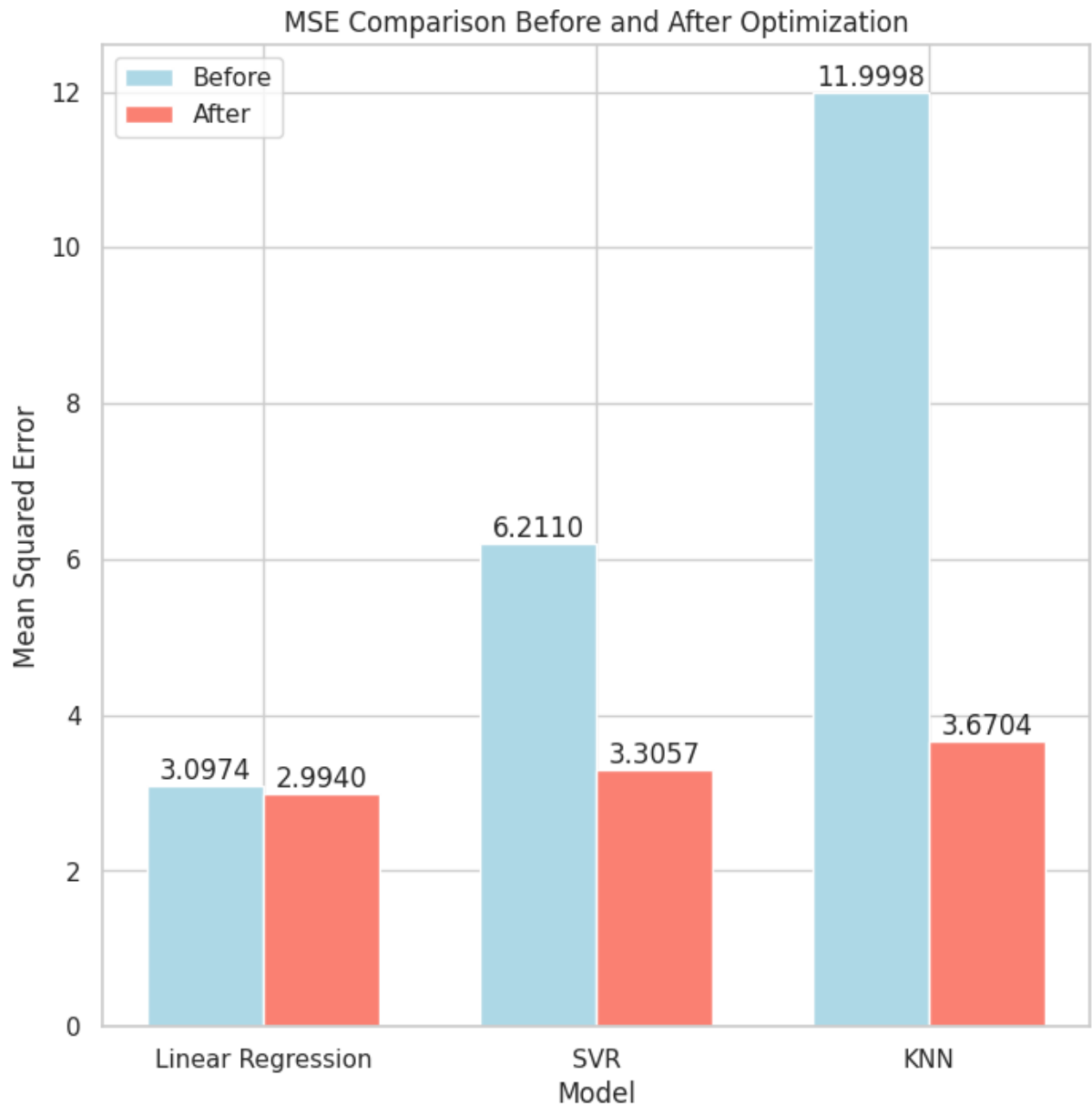
```

```

⇌
Model | Before | After
-----
LinearRegression | 3.0974 | 2.9940
SVR | 6.2110 | 3.3057
KNN | 11.9998 | 3.6704

```

Hình 4. 11



Hình 4. 12

- Nhận xét:

- LinearRegression: Giảm từ 3.0974 xuống 2.9940, cải thiện nhẹ MSE nhờ chọn đặc trưng quan trọng.
- SVR: Giảm mạnh từ 6.2110 xuống 3.3057, cải thiện rõ rệt MSE nhờ giảm nhiễu và tập trung vào các đặc trưng quan trọng.

- KNN: Giảm từ 11.9998 xuống 3.6704, cải thiện đáng kể MSE nhờ chọn đặc trưng hiệu quả và giảm nhiễu.

- Kết luận:

- Việc áp dụng phân tích tương quan có tác động tích cực đến các mô hình hồi quy (SVR và KNN), giúp cải thiện rõ rệt hiệu suất và giảm thiểu MSE.
- LinearRegression có cải thiện nhẹ, cho thấy phân tích tương quan giúp mô hình hoạt động ổn định hơn, mặc dù không có sự thay đổi lớn.

3. Tổng kết

- Ưu điểm:

- Phân tích tương quan giúp loại bỏ các đặc trưng có tương quan thấp với biến mục tiêu hoặc có sự tương quan cao với nhau (đa cộng tuyến), giúp giảm thiểu nhiễu và cải thiện hiệu suất của mô hình.
- Giúp mô hình tập trung vào các yếu tố thực sự ảnh hưởng đến kết quả
- Cải thiện tốc độ huấn luyện và tối ưu hóa tài nguyên tính toán

- Nhược điểm:

- Có thể dẫn đến việc mất thông tin quan trọng mà mô hình có thể sử dụng để cải thiện dự đoán.
- Không phù hợp với tất cả mô hình: Một số mô hình có thể hoạt động tốt hơn khi có nhiều đặc trưng hơn (như Random Forest hoặc Neural Networks), vì chúng có khả năng học các mối quan hệ phức tạp.
- Đối với các mô hình như Logistic Regression, việc loại bỏ các đặc trưng có thể khiến mô hình mất đi một số thông tin quan trọng, từ đó làm giảm hiệu suất của mô hình.

- Phân tích tương quan là một công cụ hữu ích trong việc tối ưu mô hình, nhưng không phải lúc nào cũng mang lại kết quả tốt. Việc sử dụng nó cần được cân nhắc kỹ

lượng dựa trên đặc điểm của dữ liệu và mô hình, vì đôi khi việc loại bỏ đặc trưng có thể làm mất đi thông tin quan trọng và giảm hiệu suất.

TÀI LIỆU THAM KHẢO

Tiếng Việt

1. Trích chọn đặc trưng (Feature selection),” Khoá Học. [Online]. Available: <https://tek4.vn/khoa-hoc/machine-learning-co-ban/trich-chon-dac-trung-feature-selection>
2. “Tìm hiểu Overfitting.” [Online]. Available: <https://machinelearningcoban.com/2017/03/04/overfitting/>
3. “Underfitting and Overfitting in Machine Learning.” [Online]. Available: <https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/>
4. “Overfitting and Regularization in ML.” [Online]. Available: <https://www.geeksforgeeks.org/overfitting-and-regularization-in-ml/>

Tiếng Anh

5. A. Demeusy, “Pearson correlation: Methodology, Limitations & Alternatives — Part 1 : Methodology,” Medium, Nov. 09, 2024. [Online]. Available: <https://medium.com/@anthony.demeusy/pearson-correlation-methodology-limitations-alternatives-part-1-methodology-42abe8f1ba90>
6. “Feature selection techniques in machine learning,” StrataScratch. <https://www.stratascratch.com/blog/feature-selection-techniques-in-machine-learning/>
7. SuNT, “SUNT’s blog |AI in Practical.” https://tiensu.github.io/blog/91_data_prepeation_for_ml_feature_selection_6/