

VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY  
UNIVERSITY OF TECHNOLOGY  
FACULTY OF COMPUTER SCIENCE AND ENGINEERING



MACHINE LEARNING (CO3117)

---

Assignment

# Math-Students Performance Data

---

Advisor: Võ Thanh Hùng, *CSE-HCMUT*  
Students: Tạ Gia Bảo - 2110795 (Group L02)  
Nguyễn Đức Bình - 2110817 (Group L02)  
Hong Anh Dũng - 2210572 (Group L02)  
Nguyễn Đăng Dũng - 2210581 (Group L02)

HO CHI MINH CITY, 28 APRIL 2025



## Contents

<b>1</b>	<b>Member list &amp; Workload</b>	<b>2</b>
<b>2</b>	<b>GIỚI THIỆU</b>	<b>2</b>
2.1	Mục tiêu . . . . .	2
2.2	Hướng tiếp cận . . . . .	2
<b>3</b>	<b>TỔNG QUAN DỮ LIỆU: Math-Students Performance Data</b>	<b>3</b>
3.1	Ngữ cảnh dữ liệu . . . . .	3
3.2	Cấu trúc dữ liệu . . . . .	3
3.3	Làm rõ dữ liệu . . . . .	5
3.4	Tiền xử lý dữ liệu . . . . .	12
<b>4</b>	<b>Xây dựng mô hình học máy</b>	<b>13</b>
4.1	Logistic Regression . . . . .	13
4.2	Decision Tree . . . . .	15
4.3	Random Forest . . . . .	16
4.4	Support Vector Machine . . . . .	17
4.5	K-Nearest Neighbors . . . . .	18
4.6	Gradient Boosting . . . . .	19
<b>5</b>	<b>So sánh và đánh giá</b>	<b>20</b>
5.1	Metric đánh giá . . . . .	20
5.2	Kết quả các mô hình . . . . .	21
5.3	Đánh giá . . . . .	21



## 1 Member list & Workload

No.	Fullname	Student ID	Problems	Percentage of work
1	Nguyễn Đức Bình	210817	- Data understanding - Feature selection	25%
2	Tạ Gia Bảo	2110795	Training and evaluate models: - Logistic Regression - Decision tree	25%
3	Hồng Anh Dũng	2210572	Training and evaluate models: - Random Forest - Support Vector Machine	25%
4	Nguyễn Đăng Dũng	2210581	Training and evaluate models: - K-Nearest Neighbors - Gradient Boosting	25%

## 2 GIỚI THIỆU

### 2.1 Mục tiêu

Bài tập lớn hướng đến áp dụng các kiến thức Machine learning thông qua việc xây dựng các mô hình phân loại nhằm dự đoán khả năng đậu/trượt của học sinh trong môn Toán dựa trên các đặc điểm cá nhân, gia đình và học tập. Mục tiêu cụ thể bao gồm:

1. Phân tích sâu các yếu tố ảnh hưởng đến thành tích học tập của học sinh trong môn Toán, xác định các biến có tương quan mạnh đến kết quả học tập của mỗi học sinh.
2. Xây dựng mô hình phân loại có độ chính xác cao để dự đoán học sinh nào có khả năng đậu (điểm  $G3 > 10$ ) và học sinh nào có nguy cơ trượt (điểm  $G3 \leq 10$ ).
3. Cung cấp những hiểu biết có giá trị về các yếu tố quyết định thành công trong học tập qua đó có thể được sử dụng để phát triển các chiến lược giáo dục hiệu quả.
4. Đánh giá hiệu suất của các thuật toán phân loại khác nhau và xác định mô hình phù hợp nhất cho bài toán này.

### 2.2 Hướng tiếp cận

Để đạt được các mục tiêu trên, nhóm áp dụng phương pháp tiếp cận dựa trên dữ liệu theo các bước sau:

**Phân tích khám phá dữ liệu (EDA):** Thực hiện phân tích toàn diện về phân phối của các biến, mối quan hệ giữa chúng và mối tương quan với biến mục tiêu.

**Lựa chọn đặc trưng định lượng:** Sử dụng các phương pháp thống kê như kiểm định t-test, Cohen's d và hệ số tương quan để xác định các biến số có ảnh hưởng đáng kể đến kết quả học tập của học sinh.

**Phân tích biến phân loại:** Sử dụng kiểm định Chi-square và độ đo Cramér's V để đánh giá mối liên hệ giữa các biến phân loại và kết quả đậu/trượt.

**Tiền xử lý dữ liệu:** Biến đổi các biến bị lệch, mã hóa biến phân loại và chuẩn hóa dữ liệu nếu phát hiện dữ liệu có phân phối không đều để chuẩn bị cho việc xây dựng mô hình.

**Lựa chọn mô hình:** Huấn luyện và đánh giá nhiều mô hình phân loại khác nhau bao gồm...

**Đánh giá mô hình:** Sử dụng các phương pháp đánh giá như precision-recall và confusion matrix...

**Nhận xét kết quả:** Phân tích...

## 3 TỔNG QUAN DỮ LIỆU: **Math-Students Performance Data**

### 3.1 Ngữ cảnh dữ liệu

Bộ dữ liệu **Math-Students.csv** chứa thông tin về học sinh trung học Bồ Đào Nha học môn Toán, được thu thập qua các cuộc khảo sát và hồ sơ học tập. Dữ liệu bao gồm nhiều khía cạnh của đời sống học sinh:

**Thông tin nhân khẩu học:** Trường học hiện tại, tuổi, giới tính, khu vực sinh sống (thành thị/nông thôn).

**Thông tin gia đình:** Quy mô gia đình, trình độ học vấn và nghề nghiệp của cha mẹ, tình trạng chung sống của cha mẹ, chất lượng mối quan hệ gia đình.

**Đặc điểm học tập:** Thời gian học tập hàng tuần, hỗ trợ giáo dục từ trường và gia đình, lịch sử trượt lớp.

**Sở thích và lối sống:** Hoạt động ngoại khóa, thời gian giải trí, mức độ tiêu thụ rượu, các mối quan hệ lãng mạn.

**Kết quả học tập:** Điểm số qua ba giai đoạn (G1, G2 và G3), với G3 là điểm cuối kỳ.

Mục tiêu chính của bộ dữ liệu là cung cấp cái nhìn toàn diện về các yếu tố ảnh hưởng đến thành tích học tập của học sinh, từ đó có thể dự đoán khả năng đậu/trượt môn Toán.

### 3.2 Cấu trúc dữ liệu

Bộ dữ liệu bao gồm 399 mẫu (học sinh) với 33 biến, không có giá trị khuyết (missing values) và không có bản ghi trùng lặp, điều này cho thấy chất lượng dữ liệu tốt.

```
1 print('Number of duplicate rows:', data.duplicated().sum())
2 print('Number of missing values:', data.isnull().sum().sum())
3 data.columns
```

Number of duplicate rows: 0

Number of missing values: 0

```
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'], dtype='object')
```

Các biến trong bộ dữ liệu được phân loại thành:

- 17 biến phân loại (object): như `school`, `sex`, `address`, `Mjob`, `Fjob`, `guardian`...

- 16 biến số (int64): như `age`, `Medu`, `Fedu`, `traveltime`, `studytime`, `failures`, `G1`, `G2`, `G3`...

Ý nghĩa của một số biến chính:

`school`: Trường học sinh theo học ('GP' - Gabriel Pereira hoặc 'MS' - Mousinho da Silveira)

`sex`: Giới tính ('F' - nữ hoặc 'M' - nam)

`age`: Tuổi của học sinh (từ 15 đến 22)

`address`: Loại nơi ở ('U' - thành thị hoặc 'R' - nông thôn)

`famsize`: Quy mô gia đình ('LE3' -  $\leq 3$  người hoặc 'GT3' -  $> 3$  người)

`Pstatus`: Tình trạng chung sống của cha mẹ ('T' - sống cùng nhau hoặc 'A' - sống riêng)

`Medu/Fedu`: Trình độ học vấn của mẹ/cha (0 - không có, 1 - tiểu học, 2 - cấp 2, 3 - cấp 3, 4 - đại học)

`Mjob/Fjob`: Nghề nghiệp của mẹ/cha ('teacher', 'health', 'services', 'at\_home', 'other')

`guardian`: Người giám hộ ('mother', 'father', 'other')

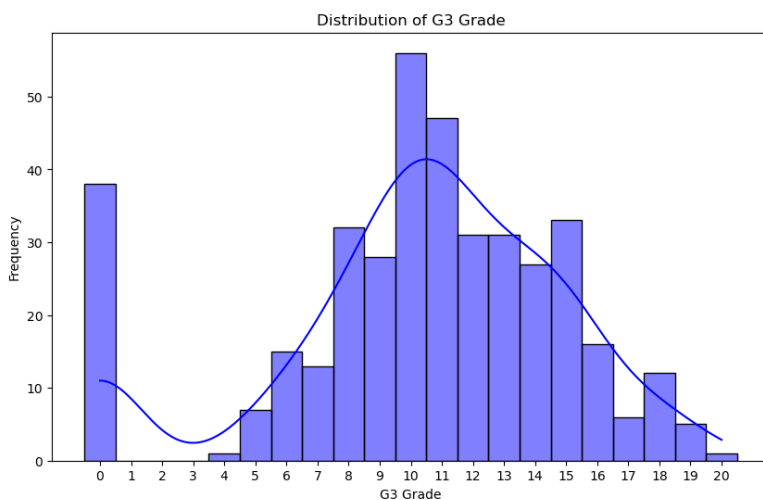
`traveltime`: Thời gian di chuyển từ nhà đến trường (1 -  $< 15$  phút, 2 - 15-30 phút, 3 - 30-60 phút, 4 -  $> 60$  phút)

`studytime`: Thời gian học tập hàng tuần (1 -  $< 2$  giờ, 2 - 2-5 giờ, 3 - 5-10 giờ, 4 -  $> 10$  giờ)

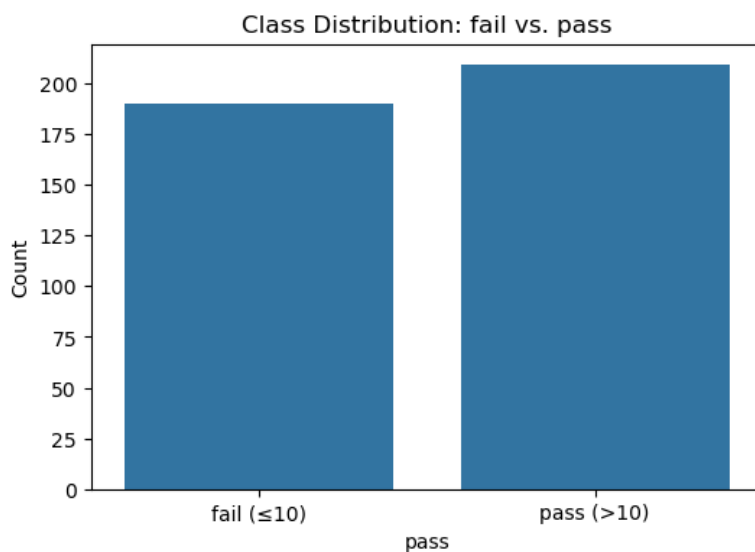
`failures`: Số lần trượt lớp trước đây

`G1`, `G2`, `G3`: Điểm số qua 3 giai đoạn (từ 0 đến 20)

Biến mục tiêu được tạo ra từ `G3` (điểm cuối kỳ): `pass`: 0 (fail) nếu `G3`  $\leq 10$ ; 1 (pass) nếu `G3`  $> 10$



Hình 1: Phân phối biến mục tiêu trên thang điểm 20



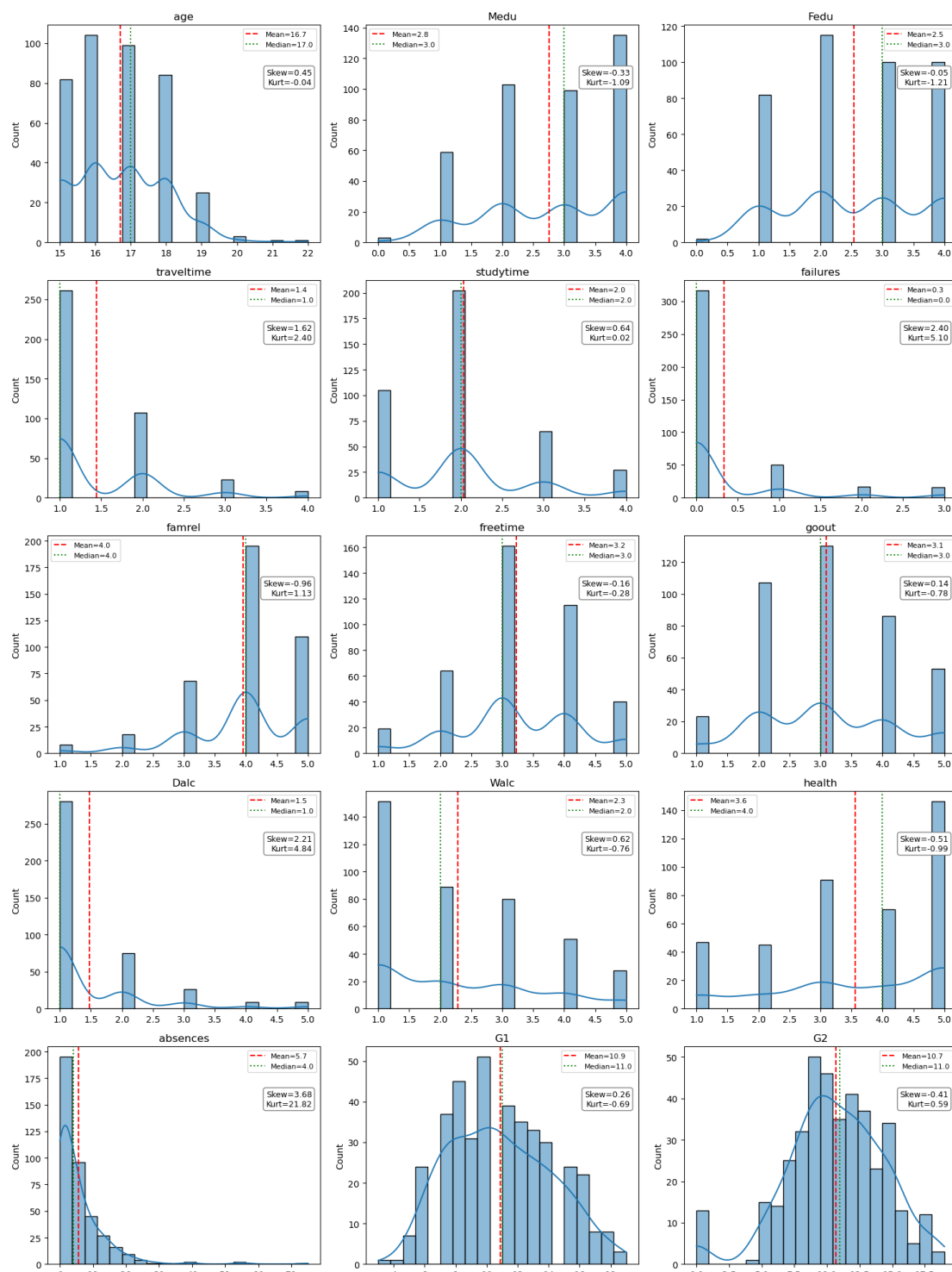
Hình 2: Phân phối biến mục tiêu pass/fail

Chúng ta thấy phân phối khá cân bằng với 190 học sinh trượt (47.6%) và 209 học sinh đậu (52.4%).

### 3.3 Làm rõ dữ liệu

Thông qua biểu đồ phân phối điểm G3 cho thấy phần lớn điểm tập trung trong khoảng 7-14, với một số lượng nhỏ học sinh đạt điểm rất cao (16-20). Điểm ngưỡng đậu/trượt là 10 tạo ra sự phân chia tương đối cân bằng. Sau đây chúng ta xét đến phân phối của các biến số số (numerical) với biến mục tiêu với phân tích skewness và kurtosis. (Hình 3)

Ta thấy đa số các biến số có phân phối đối xứng. Chỉ có một vài biến số lệch như: `traveltime` có phân phối lệch phải khá mạnh, đuôi dài — nhiều học sinh có thời gian đi học rất cao. `failures` lệch phải rất mạnh, đuôi rất dày — đa số học sinh không trượt, một số ít trượt nhiều lần. `absences` cực kỳ lệch phải và có nhiều outlier — một số bạn nghỉ rất nhiều ngày.

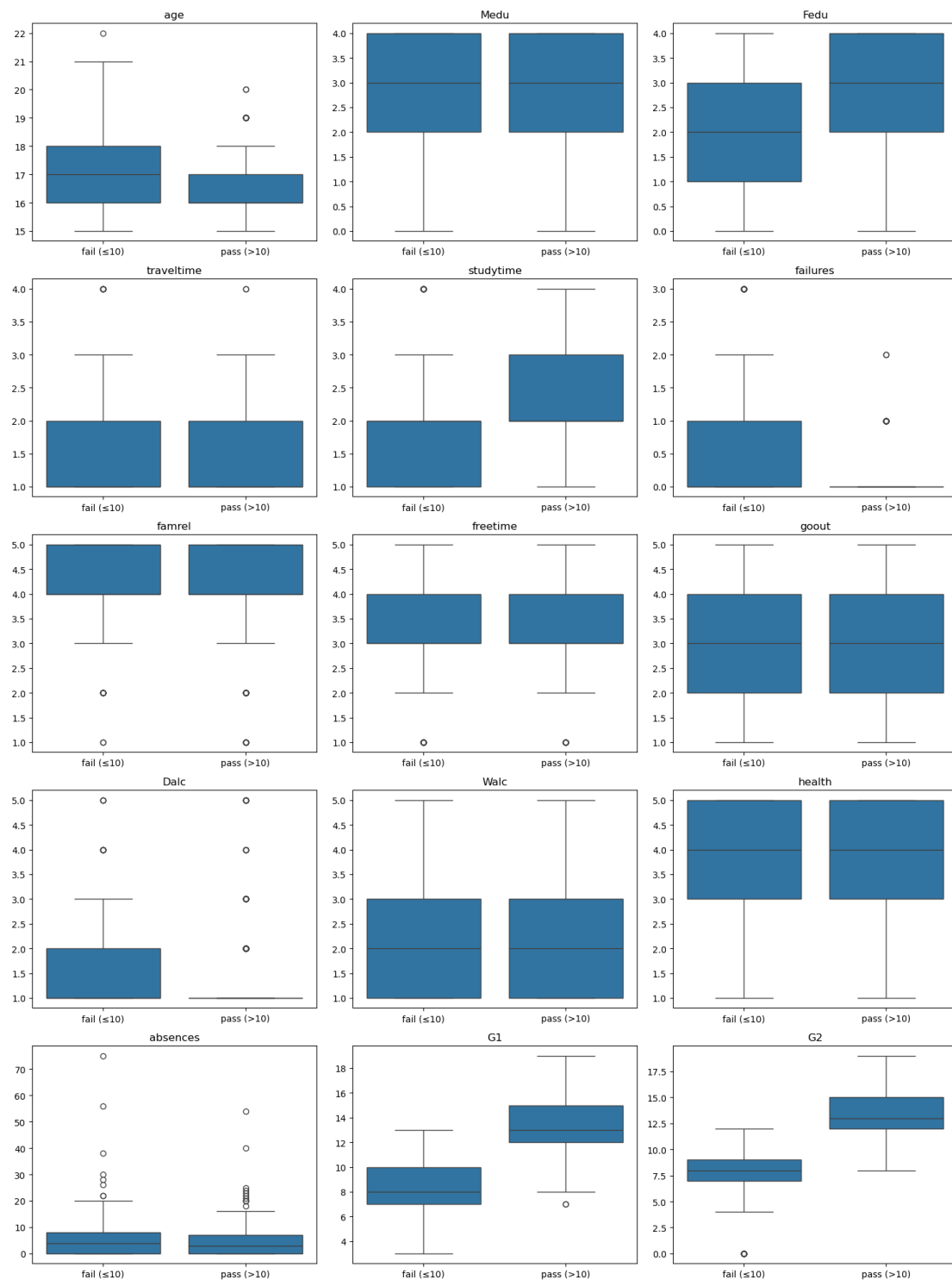


Hình 3: Phân phối biến numerical và điểm skewness và kurtosis



Tiếp đến ta cần xem xét giá trị của các biến này ảnh hưởng như thế nào đến biến mục tiêu bằng cách xem xét biểu đồ boxplot của 2 thuộc tính **pass/fail**. (Hình 4)





Hình 4: Biểu đồ Boxplot với biến mục tiêu

Dựa vào biểu đồ boxplot so sánh giữa các thuộc tính **pass** và **fail**, ta có thể thấy sơ bộ đa số các feature đều có giá trị trung bình như nhau giữa **fail** và **pass**. Điều này chứng tỏ các feature này là như nhau với dữ liệu của các bạn **pass** và các bạn **fail**. Có thể kết luận sơ bộ rằng các feature này có thể không đóng góp gì nhiều đối với kết quả **fail** và **pass** của các bạn. Bên cạnh đó, có các biểu đồ của một số feature có sự chênh lệch rõ như **G1**, **G2**, **Studytime** có thể các feature này sẽ mang ý nghĩa.

Phân tích độ tương quan dựa trên boxplot mới chỉ cho chúng ta cảm giác các biến có liên quan với nhau hay không. Muốn khẳng định chính xác cần có thang đo cụ thể:

```
1 def cohen_d(x, y):
2     """Compute Cohen's d for two 1D arrays."""
3     nx, ny = len(x), len(y)
4     sx, sy = x.std(ddof=1), y.std(ddof=1)
5     # pooled standard deviation
6     pooled = np.sqrt(((nx-1)*sx**2 + (ny-1)*sy**2) / (nx+ny-2))
7     return (x.mean() - y.mean()) / pooled
8
9 num_cols = data.select_dtypes(include='int64').drop(columns=['G3', 'pass'
10 ])
11 grp_fail = data[data['pass']==0]
12 grp_pass = data[data['pass']==1]
13
14 effects = {col: abs(cohen_d(grp_pass[col], grp_fail[col])) for col in
15             num_cols}
16 effects = pd.Series(effects).sort_values(ascending=False)
17
18 print("Cohen's d (absolute) sorted:")
19 print(effects)
```

```
Cohen's d (absolute) sorted:
G2          2.332734
G1          2.310316
failures    0.674149
age         0.328096
Fedu        0.315227
Medu        0.303321
goout       0.286221
traveltime  0.257741
Walc        0.254445
studytime   0.201314
Dalc        0.130574
absences    0.097147
famrel      0.051084
health      0.027625
freetime    0.002884
dtype: float64
```

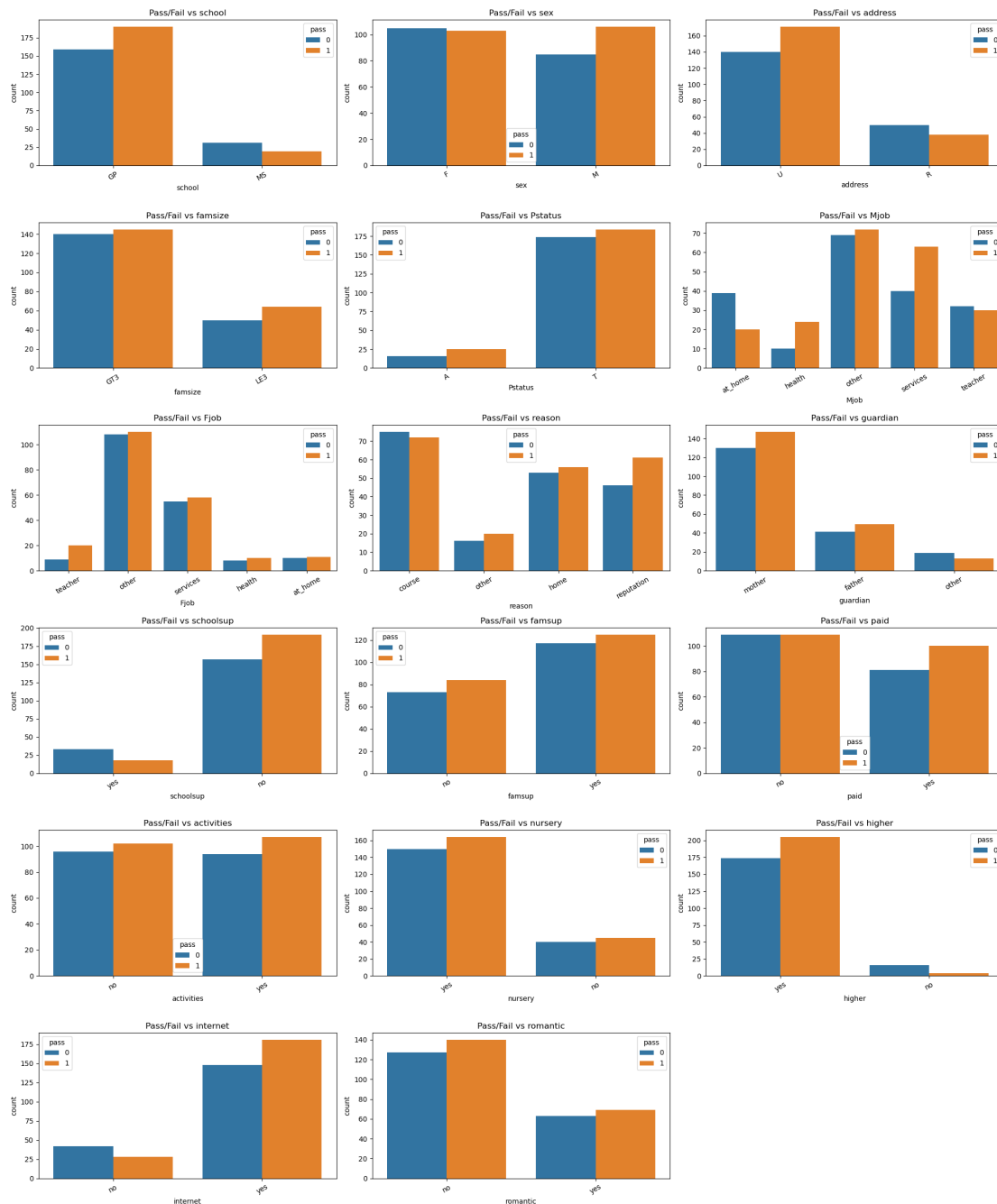
Có thể kết luận rằng các biến có ảnh hưởng lớn đến target có thể kể đến như **G1**, **G2**. Biến **failures** cũng có ảnh hưởng nhưng không lớn bằng **G1** và **G2**. Bên cạnh đó, các biến **age**, **Fedu**,



Medu, goout, traveltime, Walc cũng gây hiệu ứng nhỏ đối với kết quả dự đoán.

Tuy nhiên, so với dữ liệu boxplot, có thấy sự khác biệt về **studytime** giữa nhóm **pass** và **fail**, nhưng khi thực hiện kiểm tra độ tương quan thì lại thấy sự tương quan thấp đến kết quả. Điều này vẫn cho thấy học sinh dành nhiều thời gian học hơn thì có xu hướng “pass” nhiều hơn, nhưng khác biệt không quá rõ ràng như G1/G2. Nhưng có thể nhiều bạn tuy ít giờ luyện ở nhà nhưng vẫn có G1/G2 cao do học tốt trên lớp và thời gian học không phải cứ nhiều là hiệu quả.

Tiếp đến ta cùng xem xét các biến non-numerical ảnh hưởng như thế nào đến biến mục tiêu:



Hình 5: Phân phối biến non-numerical

Chúng ta có thể thấy không có sự khác biệt rõ ràng nào đến từ sự tương quan giữa **fail** và **pass** với các thuộc tính vì với các thuộc tính này thì tỉ lệ **pass** và **fail** của học sinh gần như là bằng nhau và không có sự chênh lệch lớn nào, chứng tỏ không có sự khác biệt rõ rệt đến mục tiêu dự đoán. Thực hiện kiểm tra lại thông qua thang đo Cramer's V cho kết quả:

```
1 def cramers_v(x, y):
2     table = pd.crosstab(x, y)
3     chi2 = chi2_contingency(table)[0]
4     n = table.sum().sum()
5     r, k = table.shape
6     return np.sqrt(chi2 / (n * (min(k, r) - 1)))
7
8 cramers_results = {col: cramers_v(data[col], data['pass']) for col in
9     cat_cols}
10 cramers_sorted = pd.Series(cramers_results).sort_values(ascending=False)
11 print("\nCramer's V for categorical variables:")
12 print(cramers_sorted)
```

```
      Cramér's V for categorical variables:
Mjob      0.201994
higher    0.137440
schoolsup  0.123458
internet   0.107751
school     0.101410
Fjob       0.095563
address    0.091927
guardian   0.070429
reason     0.066986
sex         0.054772
Pstatus    0.049974
paid        0.047279
famsize     0.042053
famsup      0.012963
activities  0.012187
nursery     0.000000
romantic    0.000000
dtype: float64
```

Thông qua thang đo Cramér's V lại thấy sự tương quan khá thấp. Điều này là hợp lý vì so với biểu đồ so sánh ở trên cũng nêu ra không có sự chênh lệch về tỉ lệ **fail/pass** đối với các thuộc tính.

Thế nên sau khi phân tích tương quan các biến ảnh hưởng đến mục tiêu có thể rút ra các feature chính đưa vào mô hình bao gồm:

```
1 selected_features = [
2     'G1', 'G2', 'failures', 'age', 'Fedu', 'Medu',
3     'goout', 'traveltime', 'Walc'
4 ]
```

### 3.4 Tiền xử lý dữ liệu

Qua phân tích phân phối các biến, nhóm thực hiện các bước tiền xử lý sau:

**Tạo biến mục tiêu:** Biến G3 (điểm cuối kỳ) được chuyển đổi thành biến nhị phân 'pass', giúp chuyển bài toán từ dự đoán điểm số sang phân loại đậu/trượt:

```
1 df['pass'] = (df['G3'] > 10).astype(int)
```

**Lựa chọn đặc trưng:** Từ phân tích định lượng của các biến numerical và non-numerical (Cohen's d và hệ số tương quan), chúng tôi xác định các đặc trưng quan trọng nhất cho mô hình:

```
1 selected_features = [  
2     'G1', 'G2', 'failures', 'age', 'Fedu', 'Medu',  
3     'goout', 'traveltime', 'Walc'  
4 ]
```

**Xử lý các biến bị lệch:** Một số biến như failures và traveltime có phân phối lệch phải mạnh (positive skewness cao), gây khó khăn cho các thuật toán học máy giả định phân phối chuẩn nên áp dụng phép biến đổi logarithmic để giảm độ lệch:

```
1 df['failures'] = np.log1p(df['failures'])  
2 df['traveltime'] = np.log1p(df['traveltime'])
```

Áp dụng StandardScaling và sau đó chia tỉ lệ train/test là 0.8/0.2

## 4 Xây dựng mô hình học máy

### 4.1 Logistic Regression

Mô hình Logistic Regression được huấn luyện bằng thư viện scikit-learn:

```
1 from sklearn.linear_model import LogisticRegression  
2 from sklearn.metrics import classification_report, confusion_matrix,  
   accuracy_score, ConfusionMatrixDisplay  
3 from Data_loader import load_data  
4 import matplotlib.pyplot as plt
```

Dữ liệu được tải từ file Math-Students.csv thông qua hàm load\_data. Sau đó, dữ liệu được tách thành tập train và tập test. Các tensors được chuyển thành mảng numpy để phù hợp với yêu cầu đầu vào của scikit-learn.

```
1 train_dataset, test_dataset = load_data('../data/Math-Students.csv')  
2  
3 X_train, y_train = train_dataset.tensors  
4 X_test, y_test = test_dataset.tensors  
5  
6 X_train = X_train.numpy()  
7 y_train = y_train.numpy()  
8 X_test = X_test.numpy()  
9 y_test = y_test.numpy()  
10  
11 model = LogisticRegression()  
12 model.fit(X_train, y_train)
```

Các tham số (hyperparameters) của Logistic Regression được thiết lập mặc định:

- `penalty='l2'`: Sử dụng chuẩn L2 (Ridge Regularization) để giảm overfitting.
- `solver='lbfgs'`: Phương pháp tối ưu hóa sử dụng thuật toán L-BFGS.
- `C=1.0`: Tham số nghịch đảo của regularization strength (giá trị nhỏ hơn sẽ regularize mạnh hơn).
- `max_iter=100`: Số lần lặp tối đa.

Để tinh chỉnh mô hình và tăng độ chính xác, có thể:

- Điều chỉnh `C` để kiểm soát mức độ regularization (ví dụ: thử `C=0.5` hoặc `C=2.0`).
- Thử các solver khác như `liblinear` hoặc `saga` nếu dữ liệu lớn hoặc sparse.
- Tăng `max_iter` nếu thuật toán chưa hội tụ.

Kiểm tra accuracy, confusion matrix và classification report để đánh giá hiệu suất model.

```
1 y_pred = model.predict(X_test)
2
3 print("=== Logistic Regression Evaluation ===")
4 print("Accuracy:", accuracy_score(y_test, y_pred) * 100, "%")
5 print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
6 print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
=== Logistic Regression Evaluation ===
Accuracy: 93.75 %
Classification Report:
              precision    recall  f1-score   support

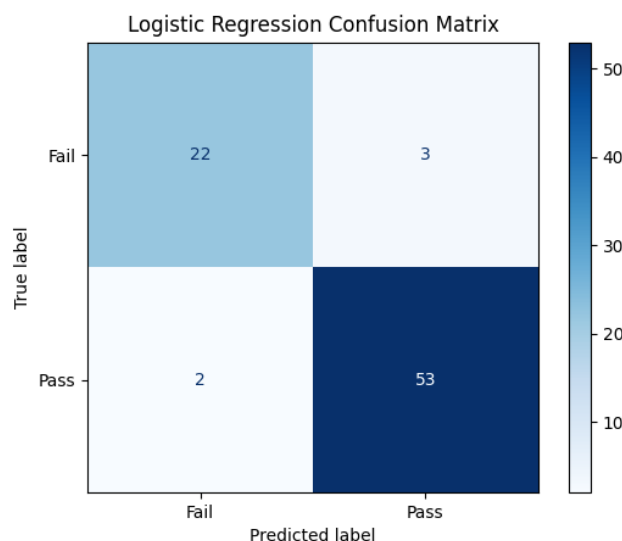
     0.0       0.92      0.88      0.90         25
     1.0       0.95      0.96      0.95         55

 accuracy          0.94
 macro avg         0.93
 weighted avg      0.94
```

Hình 6: Classification Report - Logistic Regression

Vẽ biểu đồ Confusion Matrix:

```
1 disp = ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(y_test,
2                               y_pred), display_labels=["Fail", "Pass"])
3 disp.plot(cmap='Blues')
4 plt.title('Logistic Regression Confusion Matrix')
5 plt.show()
```



Hình 7: Confusion Matrix - Logistic Regression

Mô hình Logistic Regression đạt độ chính xác **93.75%**. Qua confusion matrix, mô hình dự đoán đúng 22 mẫu Fail và 53 mẫu Pass, với tổng số lỗi nhỏ (5 mẫu). Precision và Recall đạt trên 90% cho cả hai lớp, cho thấy mô hình phân loại khá tốt và ổn định trên tập kiểm tra.

**Lưu ý:** Các phương pháp khác (Decision Tree, KNN, SVM, v.v.) chỉ thay đổi ở phần khởi tạo mô hình (`model = ...`), các bước còn lại được giữ nguyên.

## 4.2 Decision Tree

Tương tự xây dựng Decision Tree với `scikit-learn`:

```
1 from sklearn.tree import DecisionTreeClassifier
2 model = DecisionTreeClassifier(random_state=42)
```

Các tham số mặc định của Decision Tree gồm:

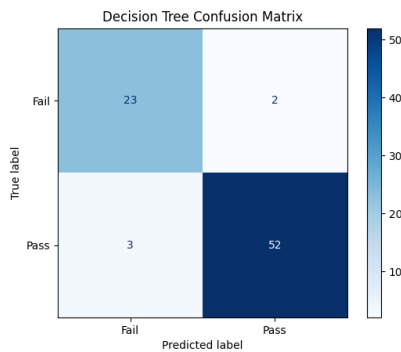
- `criterion='gini'`: Tiêu chí chia nhánh dựa trên chỉ số Gini.
- `max_depth=None`: Không giới hạn độ sâu của cây (có thể gây overfitting).
- `min_samples_split=2`: Số lượng mẫu tối thiểu để tách một node.

Các hướng tinh chỉnh mô hình:

- Giới hạn `max_depth` (ví dụ: 5 hoặc 10) để giảm overfitting.
- Thay đổi `criterion` sang `'entropy'` để thử nghiệm cách đánh giá khác.
- Tăng `min_samples_split` để yêu cầu nhiều mẫu hơn mới cho phép chia nhánh.



### Kết quả:



(a) Confusion Matrix - Decision Tree

=== Decision Tree Evaluation ===				
Accuracy: 93.75 %				
Classification Report:				
	precision	recall	f1-score	support
Fail	0.0	0.88	0.92	25
Pass	1.0	0.96	0.95	55
accuracy			0.94	80
macro avg	0.92	0.93	0.93	80
weighted avg	0.94	0.94	0.94	80

(b) Classification Report - Decision Tree

Hình 8: Kết quả mô hình Decision Tree

Mô hình Decision Tree cũng đạt độ chính xác **93.75%**. Dựa trên confusion matrix, mô hình dự đoán chính xác 23 mẫu Fail và 52 mẫu Pass, sai lệch nhỏ (5 mẫu). Precision và Recall của mô hình cao, với f1-score bằng 0.90 cho lớp Fail và 0.95 cho lớp Pass, cho thấy Decision Tree khai thác đặc trưng tốt để đưa ra dự đoán hợp lý.

## 4.3 Random Forest

Tương tự xây dựng Random Forest với `scikit-learn`:

```
1 from sklearn.ensemble import RandomForestClassifier
2 model = RandomForestClassifier()
```

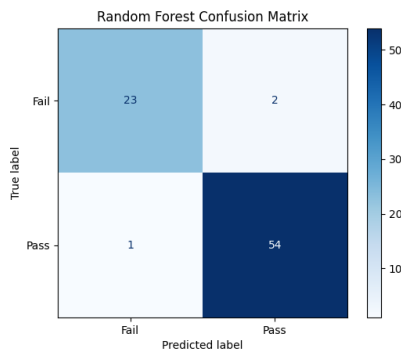
Một vài tham số mặc định của Random Forest gồm:

- `n_estimators=100`: Số lượng cây quyết định trong rừng.
- `criterion='gini'`: Tiêu chí phân chia nút dựa trên chỉ số Gini.
- `max_depth=None`: Cây phát triển tự do đến khi tất cả các lá thuần.
- `max_features='sqrt'`: Số lượng đặc trưng được xét tại mỗi node chia tách.

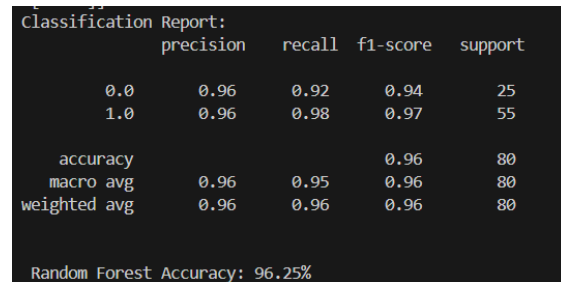
Các hướng tinh chỉnh accuracy của mô hình:

- Thay đổi `n_estimators` (ví dụ: 200 hoặc 300) để tăng tính ổn định.
- Giới hạn `max_depth` để tránh overfitting.
- Thử `max_features='log2'` hoặc giá trị tùy chỉnh để cải thiện hiệu quả.

### Kết quả:



(a) Confusion Matrix - Random Forest



Classification Report:				
	precision	recall	f1-score	support
0.0	0.96	0.92	0.94	25
1.0	0.96	0.98	0.97	55
accuracy			0.96	80
macro avg	0.96	0.95	0.96	80
weighted avg	0.96	0.96	0.96	80

Random Forest Accuracy: 96.25%

(b) Classification Report - Random Forest

Hình 9: Kết quả mô hình Random Forest

Mô hình Random Forest có độ chính xác **96.25%**. Dựa trên kết quả từ confusion matrix, mô hình Random Forest đã dự đoán đúng 23 mẫu thuộc lớp 0 (Fail) và 54 mẫu thuộc lớp 1 (Pass), với tổng số mẫu dự đoán sai chỉ là 3 trên 80 mẫu kiểm tra. Chỉ số precision và recall của mô hình đều đạt giá trị cao ở cả hai lớp, trong đó f1-score đạt 0.94 đối với lớp Fail và 0.97 đối với lớp Pass. Điều này cho thấy mô hình có khả năng khai thác hiệu quả các đặc trưng của dữ liệu để đưa ra các dự đoán chính xác và cân bằng. Độ chính xác tổng thể (accuracy) đạt 96.25%, phản ánh hiệu suất dự đoán xuất sắc và khả năng phân biệt rõ ràng giữa các nhóm đối tượng.

## 4.4 Support Vector Machine

Tương tự xây dựng Support Vector Machine với `scikit-learn`:

```
1 from sklearn.svm import SVC
2 model = SVC()
```

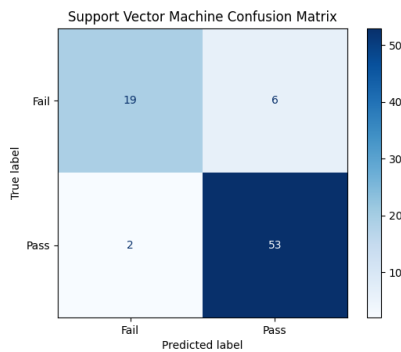
Các tham số mặc định của SVM gồm:

- `C=1.0`: Hệ số điều chỉnh biên phạt (penalty parameter).
- `kernel='rbf'`: Kernel mặc định là Gaussian Radial Basis Function.
- `gamma='scale'`: Tham số kernel coefficient, tự động tính dựa trên số đặc trưng.

Các hướng tinh chỉnh mô hình:

- Thử các kernel khác như `linear` hoặc `poly` tùy bài toán.
- Điều chỉnh `C` để kiểm soát mức độ regularization.
- Thay đổi `gamma` để mô hình nhạy hơn với sự thay đổi của dữ liệu.

## Kết quả:



(a) Confusion Matrix - Support Vector Machine

Classification Report:				
	precision	recall	f1-score	support
0.0	0.90	0.76	0.83	25
1.0	0.90	0.96	0.93	55
accuracy			0.90	80
macro avg	0.90	0.86	0.88	80
weighted avg	0.90	0.90	0.90	80

Support Vector Machine Accuracy: 90.00%

(b) Classification Report - Support Vector Machine

Hình 10: Kết quả mô hình Support Vector Machine

Mô hình Support Vector Machine có độ chính xác **90%**. Dựa trên confusion matrix, mô hình SVM dự đoán đúng 19 mẫu thuộc lớp 0 (Fail) và 53 mẫu thuộc lớp 1 (Pass), với tổng cộng 8 mẫu bị dự đoán sai. Precision và recall của mô hình đều đạt mức cao, trong đó f1-score là 0.83 cho lớp Fail và 0.93 cho lớp Pass. Tuy nhiên, recall của lớp Fail chỉ đạt 0.76, cho thấy mô hình có xu hướng bỏ sót một số mẫu thuộc nhóm này. Mặc dù vậy, độ chính xác tổng thể đạt 90%, phản ánh khả năng dự đoán tốt của mô hình, đặc biệt trên lớp chiếm ưu thế. Kết quả cho thấy SVM khai thác đặc trưng dữ liệu hiệu quả, tuy nhiên cần cân nhắc thêm các kỹ thuật cân bằng mẫu nếu muốn cải thiện độ nhạy cho lớp Fail.

## 4.5 K-Nearest Neighbors

Tương tự xây dựng K-Nearest Neighbors với `scikit-learn`:

```
1 from sklearn.neighbors import KNeighborsClassifier
2 model = KNeighborsClassifier()
```

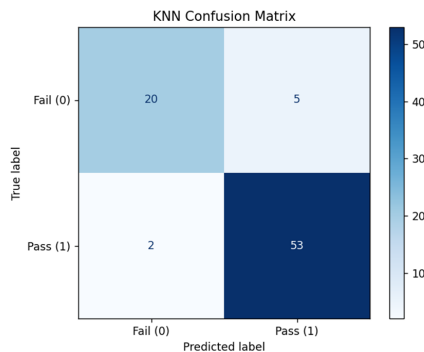
Các tham số mặc định của KNN gồm:

- `n_neighbors=5`: Số lượng láng giềng gần nhất để bỏ phiếu.
- `weights='uniform'`: Tất cả các láng giềng có trọng số bằng nhau.
- `algorithm='auto'`: Tự động chọn thuật toán tìm kiếm tối ưu (ball tree, kd tree,...).

Các hướng tinh chỉnh mô hình:

- Thử thay đổi `n_neighbors` (ví dụ: 3, 7) để tối ưu độ chính xác.
- Thử `weights='distance'` để các láng giềng gần hơn có ảnh hưởng lớn hơn.
- Chọn thuật toán cụ thể (`ball_tree`, `kd_tree`) để tăng tốc tính toán.

Kết quả:



(a) Confusion Matrix - K-Nearest Neighbors

KNN Test Accuracy: 91.25%

Classification Report:

	precision	recall	f1-score	support
0.0	0.91	0.80	0.85	25
1.0	0.91	0.96	0.94	55
accuracy			0.91	80
macro avg	0.91	0.88	0.89	80
weighted avg	0.91	0.91	0.91	80

(b) Classification Report - K-Nearest Neighbors

Hình 11: Kết quả mô hình K-Nearest Neighbors

Mô hình KNN có độ chính xác **91.25%**. Dựa trên kết quả từ confusion matrix, mô hình KNN đã dự đoán đúng 20 mẫu thuộc lớp Fail và 53 mẫu thuộc lớp Pass, với tổng số mẫu dự đoán sai là 7 trên 80 mẫu kiểm tra. Chỉ số precision và recall lần lượt là 0.91 và 0.80 cho lớp Fail, 0.91 và 0.96 cho lớp Pass. F1-score đạt 0.85 đối với lớp Fail và 0.94 đối với lớp Pass. Điều này cho thấy mô hình hoạt động khá tốt, đặc biệt là trong việc xác định các mẫu thuộc lớp Pass, mặc dù có bỏ sót một số mẫu của lớp Fail.

## 4.6 Gradient Boosting

Tương tự xây dựng Gradient Boosting với `scikit-learn`:

```
1 from sklearn.ensemble import GradientBoostingClassifier
2 model = GradientBoostingClassifier()
```

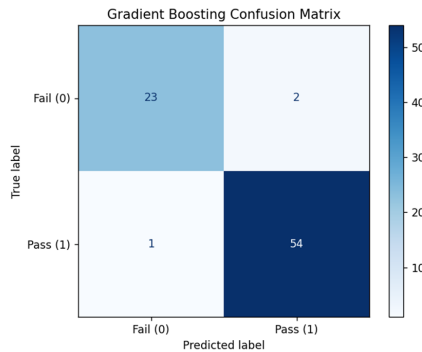
Các tham số mặc định của Gradient Boosting gồm:

- `n_estimators=100`: Số lượng cây tuần tự.
- `learning_rate=0.1`: Tốc độ học, càng nhỏ thì học càng chậm nhưng ổn định.
- `max_depth=3`: Độ sâu tối đa của mỗi cây con.

Các hướng tinh chỉnh mô hình:

- Giảm `learning_rate` kết hợp tăng `n_estimators` để cải thiện tổng quát hóa.
- Điều chỉnh `max_depth` để kiểm soát độ phức tạp mô hình.
- Sử dụng `subsample < 1.0` để thêm randomness, giúp giảm overfitting.

Kết quả:



(a) Confusion Matrix - Gradient Boosting

Gradient Boosting Test Accuracy: 96.25%

Classification Report:

	precision	recall	f1-score	support
0.0	0.96	0.92	0.94	25
1.0	0.96	0.98	0.97	55
accuracy			0.96	80
macro avg	0.96	0.95	0.96	80
weighted avg	0.96	0.96	0.96	80

(b) Classification Report - Gradient Boosting

Hình 12: Kết quả mô hình Gradient Boosting

Mô hình Gradient Boosting có độ chính xác **96.25%**. Dựa trên kết quả từ confusion matrix, mô hình Gradient Boosting đã dự đoán đúng 23 mẫu thuộc lớp Fail và 54 mẫu thuộc lớp Pass, với tổng số mẫu dự đoán sai chỉ là 3 trên 80 mẫu kiểm tra. Chỉ số precision và recall lần lượt là 0.96 và 0.92 cho lớp Fail, 0.96 và 0.98 cho lớp Pass. F1-score đạt 0.94 đối với lớp Fail và 0.97 đối với lớp Pass. Điều này cho thấy mô hình có khả năng khai thác hiệu quả các đặc trưng của dữ liệu để đưa ra các dự đoán rất chính xác và cân bằng giữa hai lớp.

## 5 So sánh và đánh giá

### 5.1 Metric đánh giá

Để đo lường và so sánh hiệu quả của các mô hình phân loại nhị phân trong bài toán dự đoán đậu/trượt (Pass/Fail), nhóm sử dụng các metric phổ biến sau đây. Các metric này được tính toán dựa trên các giá trị từ Confusion Matrix: True Positives (TP - dự đoán đúng Pass), True Negatives (TN - dự đoán đúng Fail), False Positives (FP - dự đoán nhầm Pass), và False Negatives (FN - dự đoán nhầm Fail).

**Accuracy** Là tỷ lệ tổng số dự đoán chính xác trên tổng số mẫu.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (1)$$

Mặc dù phổ biến, Accuracy có thể gây hiểu lầm khi dữ liệu mất cân bằng giữa các lớp.

**Precision** Đo lường độ chính xác của các dự đoán là 'Pass'. Tức là tỷ lệ mẫu được dự đoán là 'Pass' thực sự là 'Pass'.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2)$$

**Recall (Sensitivity)** Đo lường khả năng mô hình phát hiện đúng các mẫu 'Pass'. Tức là tỷ lệ mẫu thực sự là 'Pass' được mô hình dự đoán đúng.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (3)$$

**F1-Score** Là trung bình điều hòa (harmonic mean) của Precision và Recall. F1-Score cung cấp một cái nhìn cân bằng hơn về hiệu suất mô hình, đặc biệt hữu ích khi cả Precision và Recall đều quan trọng hoặc khi dữ liệu mất cân bằng.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN} \quad (4)$$

Để đánh giá một cách kỹ lưỡng hơn nhóm cũng sử dụng **Macro F1** (trung bình F1 của các lớp, không trọng số) và **Weighted F1** (trung bình F1 của các lớp, có trọng số theo số lượng mẫu mỗi lớp).

## 5.2 Kết quả các mô hình

Để đánh giá và so sánh hiệu quả của các mô hình đã xây dựng, nhóm tổng hợp các chỉ số đánh giá chính trên tập dữ liệu kiểm tra (test set) vào Bảng 1. Bao gồm các chỉ số Accuracy và F1-Score cho từng lớp (Fail - Lớp 0, Pass - Lớp 1) cũng như F1-Score trung bình (Macro và Weighted).

Bảng 1: Tổng hợp kết quả đánh giá các mô hình phân loại

Mô hình	Accuracy	F1 (Fail)	F1 (Pass)	F1 (Macro)	F1 (Weighted)
Logistic Regression	93.75%	0.90	0.95	0.93	0.94
Decision Tree	93.75%	0.90	0.95	0.93	0.94
Random Forest	<b>96.25%</b>	<b>0.94</b>	<b>0.97</b>	<b>0.96</b>	<b>0.96</b>
Support Vector Machine	90.00%	0.83	0.93	0.88	0.90
K-Nearest Neighbors	91.25%	0.85	0.94	0.89	0.91
Gradient Boosting	<b>96.25%</b>	<b>0.94</b>	<b>0.97</b>	<b>0.96</b>	<b>0.96</b>

## 5.3 Đánh giá

Dựa trên kết quả tổng hợp tại Bảng 1, Nhóm rút ra những nhận xét:

- Hiệu suất tốt nhất: Random Forest (RF) và Gradient Boosting (GB)** là 2 mô hình cho kết quả phân loại tốt nhất cùng đạt Accuracy là 96.25% và chỉ số F1-Score cũng cao nhất (cho cả hai lớp, macro avg, weighted avg) cho thấy khả năng dự đoán chính xác và cân bằng giữa việc xác định đúng học sinh đậu (pass) và học sinh trượt (fail) có thể lý giải vì các mô hình ensemble thường có khả năng khai thác tốt các mối quan hệ phi tuyến tính, giảm overfitting.
- Hiệu suất cân bằng: Logistic Regression (LR) và Decision Tree (DT)** 2 mô hình thu được hiệu suất khá tốt (Accuracy 93.75%) và độ cân bằng cao (F1-scores cao cho cả hai lớp 0.90-0.95) chứng tỏ dữ liệu có thể phân tách tương đối tốt bằng các mô hình tuyến tính hoặc tree-based đơn giản và đây là những lựa chọn đáng tin cậy nếu ưu tiên sự đơn giản hoặc khả năng diễn giải để người dùng thật sự thấy được logic bên trong mô hình.
- Các mô hình có hiệu suất thấp hơn: Support Vector Machine (SVM) và K-Nearest Neighbors (KNN)** có hiệu suất tổng thể thấp hơn. Đáng chú ý, cả hai mô hình này đều có chỉ số Recall thấp hơn đối với lớp 'Fail' (SVM: 0.76, KNN: 0.80) so với các mô hình khác (ví dụ: RF/GB có Recall lớp 'Fail' là 0.92). Điều này ngụ ý rằng SVM và KNN có xu hướng bỏ sót (dự đoán sai là 'Pass') nhiều học sinh có nguy cơ trượt hơn có thể do SVM bị ảnh hưởng bởi phân phối không đều hoặc đặc điểm của tập dữ liệu.

### Kết luận lựa chọn mô hình:

Dựa trên các phân tích về độ chính xác, độ cân bằng (thể hiện qua F1-scores), và khả năng xác định đúng các lớp:

- **Random Forest** và **Gradient Boosting** là những mô hình phù hợp nhất cho bài toán này với bộ dữ liệu và các đặc trưng đã chọn. Chúng cung cấp độ chính xác dự đoán cao nhất và khả năng phân loại cân bằng nhất giữa hai lớp.
- Nếu ưu tiên sự đơn giản và khả năng diễn giải, **Logistic Regression** hoặc **Decision Tree** cũng là những lựa chọn rất tốt với hiệu suất chỉ thấp hơn một chút so với RF/GB.
- **SVM** và **KNN**, mặc dù vẫn đạt được độ chính xác khá, cần được xem xét cẩn trọng, đặc biệt nếu mục tiêu là giảm thiểu việc bỏ sót học sinh ở nhóm 'Fail'. Có thể cần tinh chỉnh tham số (hyperparameter tuning) hoặc áp dụng các kỹ thuật xử lý dữ liệu mất cân bằng (nếu có) để cải thiện hiệu suất của chúng trên lớp này.

### Hướng phát triển trong tương lai:

- Thực hiện tinh chỉnh siêu tham số (Hyperparameter Tuning), đặc biệt cho RF và GB, để tối ưu hóa hiệu suất.
- Phân tích Feature Importance từ RF hoặc GB để xác định các yếu tố ảnh hưởng mạnh mẽ nhất đến kết quả học tập theo dự đoán của mô hình.
- Thử nghiệm lại với tập đặc trưng khác, ví dụ bao gồm cả các biến phân loại có giá trị Cramér's V cao sau khi được mã hóa phù hợp.

## References

- [1] F. Pedregosa et al., *Scikit-learn: Machine Learning in Python*, 2011.
- [2] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.
- [3] J. Friedman, *Greedy Function Approximation: A Gradient Boosting Machine*, 2001.
- [4] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer, 2009.
- [5] T. Cover and P. Hart, *Nearest Neighbor Pattern Classification*, 1967.