

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH



BIG DATA CLUB

BÁO CÁO

ASSIGNMENT 3 - NHÓM 1

STUDENT PERFORMANCE ANALYSIS

GV hướng dẫn: Hoàng Lê Hải Thanh
SV thực hiện: Nguyễn Đặng Anh Khoa – K20 KHMT
Nguyễn Đức An – K20 KHMT
Huỳnh Nguyên Phúc – K21 KHMT
Phạm Đức Minh – K21 KTMT

Tp. Hồ Chí Minh, Tháng 8/2022

Mục lục

1	Phần mở đầu	2
1.1	Giới thiệu đề tài	2
1.2	Ý nghĩa đề tài	2
1.3	Phạm vi đề tài	2
1.4	Sản phẩm của nhóm	3
2	Các mô hình Machine Learning sử dụng	4
2.1	Mô hình hồi quy tuyến tính	4
2.1.1	Linear Regression	4
2.2	Mô hình hồi quy tuyến tính chính quy hóa (regularization)	6
2.2.1	LASSO	6
2.2.2	RIDGE	7
2.3	Mô hình phân loại	8
2.3.1	Decision Tree	8
2.3.2	Random Forest	10
2.3.3	Support Vector Machine	12
2.3.4	AdaBoosting	14
3	Áp dụng các mô hình Machine Learning vào dự đoán điểm số và phân loại sinh viên	15
3.1	Đọc dữ liệu	15
3.2	Tiền xử lý dữ liệu	15
3.2.1	Đổi tên các column	15
3.2.2	Xử lý missing value	16
3.2.3	Xử lý outliers	17
3.2.4	Mã hóa đặc trưng	20
3.2.5	Chia dữ liệu thành tập huấn luyện và tập kiểm nghiệm	21
3.3	Huấn luyện mô hình	21
3.3.1	Regression Model	21
3.3.2	Classification Model	25
4	Demo App E-learning Analysis System	34
4.1	Giao diện	34
4.2	Sản phẩm	39
5	Định hướng mở rộng	40
6	Tài liệu tham khảo	40

1 Phần mở đầu

1.1 Giới thiệu đề tài

Tình hình dịch bệnh COVID-19 căng thẳng ở phạm vi toàn cầu đã làm đình trệ các hoạt động mua bán, sản xuất và đặc biệt là hoạt động giáo dục của nhiều quốc gia trong đó có Việt Nam. Khó khăn sinh ra từ đại dịch cũng là động lực để các công cụ hỗ trợ chuyển đổi việc học từ hình thức trực tiếp (offline) sang trực tuyến (online) phát triển rầm rộ như: các hệ thống quản lý giáo dục (Learning Management System - LMS) như Moodle được triển khai ở tất cả các cấp bậc giáo dục từ phổ thông tới đại học.

Xuất phát từ tình hình thực tế trên, nhóm 1 - Big Data Club (HCMUT) đã thực hiện xây dựng mô hình đánh giá hiệu quả học tập của người học dựa trên các thông tin về điểm số được cung cấp từ các hệ thống LMS (như Moodle) kết hợp một số thông tin của cá nhân người học thu thập từ khảo sát và thông tin lai lịch... Từ đó, phát triển công cụ dự đoán kết quả học tập của người học dựa trên các thông tin được cung cấp từ người dùng.

1.2 Ý nghĩa đề tài

Với tốc độ triển khai của giáo dục trực tuyến, lượng dữ liệu được sinh ra từ các hoạt động giáo dục của các tổ chức vô cùng lớn và rất có giá trị trong các nghiên cứu để xuất nâng cao chất lượng hoạt động giảng dạy quốc gia và trên toàn thế giới. Điểm chung của các tập dữ liệu này là kích thước rất lớn, sinh ra nhanh chóng và liên tục trong quá trình học tập trực tuyến. Tuy vậy, hiện nay các phương pháp tận dụng nguồn dữ liệu này còn hạn chế dẫn đến các hoạt động kiểm soát hành vi học tập trực tuyến chưa thật sự đạt được hiệu quả tiềm năng của nó.

Nhằm đánh giá được hiệu quả học trực tuyến của các học sinh, sinh viên trên các nền tảng elearning và cung cấp cho giảng viên những thông tin cần thiết để đề ra lộ trình học tập phù hợp cho mỗi cá nhân, nhóm đã triển khai một dự án nhỏ để thu gom, xử lý, phân tích trên các tập dữ liệu này, từ đó xây dựng nên một công cụ thông minh dựa trên các mô hình học máy trong phân tích dữ liệu với mục đích đánh giá và dự đoán kết quả học tập dựa trên lịch sử hoạt động của người dùng.

1.3 Phạm vi đề tài

1. Phạm vi nghiên cứu

- Nhóm sử dụng tập dữ liệu có sẵn từ nguồn [UCI Machine Learning Repository](#) và sử dụng các mô hình học máy để huấn luyện mô hình phân tích và dự đoán

kết quả học tập của người dùng.

- Lý do sử dụng tập dữ liệu:
 - Tập dữ liệu thuộc lĩnh vực giáo dục, có các đặc trưng quen thuộc, dễ hình dung và dễ thấy được mối quan hệ giữa chúng bằng cảm quan thông thường.
 - Tập dữ liệu này đa dạng về thuộc tính (33 thuộc tính) để chọn lọc đặc trưng và xây dựng mô hình học máy. Các thuộc tính của tập dữ liệu có thể dễ dàng thu thập bổ sung từ thực tế.

2. Phạm vi kiến thức

- **Các mô hình Regression**
 - Linear Regression
 - LASSO Regression
 - Ridge Regression
- **Các mô hình Classification**
 - Decision Tree Classifier
 - Random Forest Classifier
 - Support Vector Machine
 - AdaBoost Classifier
- Hệ thống **Elearning Analysis System** của nhóm được hiện thực bằng streamlit, đây là một framework sử dụng ngôn ngữ chính là Python và hỗ trợ việc xây dựng website cho các ứng dụng về Data Science, AI, tham khảo thêm ở đây: <https://docs.streamlit.io/>

1.4 Sản phẩm của nhóm

- **Dataset:** <http://archive.ics.uci.edu/ml/datasets/Student+Performance>
- **EDA:** [EDA.ipynb](#)
- **ML Algorithms:** [MLAlgorithms.ipynb](#)
- **Landing Page:** http://anduckhmt146.me/BDC_Assignment1/
- **Source Github:** https://github.com/KhoaNgex/education_ML_web
- **Heroku App:** <https://elearning-analysis-system.herokuapp.com/>

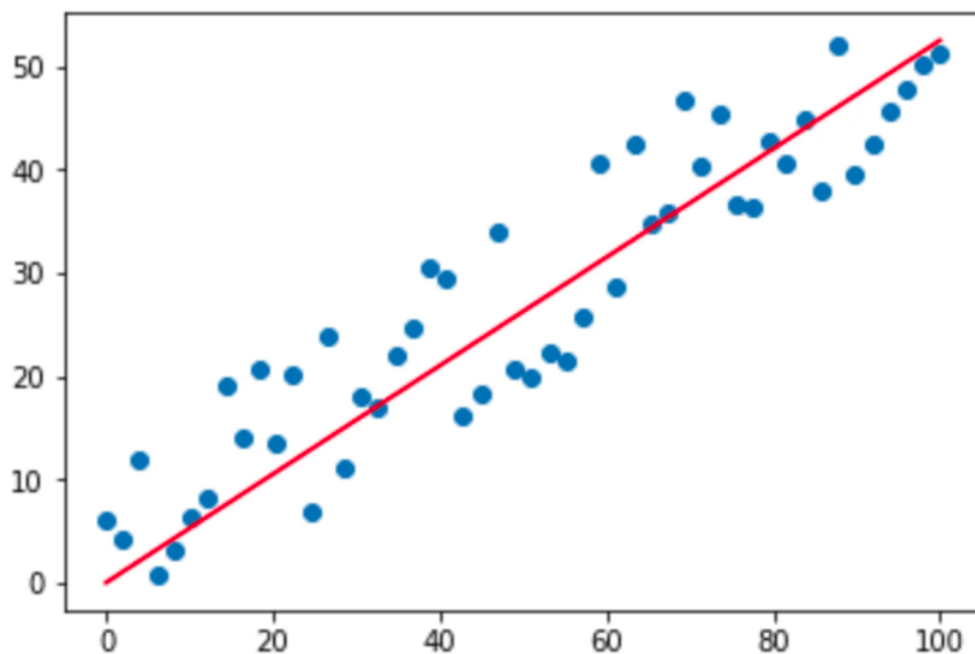
2 Các mô hình Machine Learning sử dụng

2.1 Mô hình hồi quy tuyến tính

2.1.1 Linear Regression

1. Mục tiêu

- Giải thuật dùng để dự đoán biến phụ thuộc liên tục (continuous dependent variable) dựa vào các biến độc lập đầu vào (independent variables).
- Output của Linear Regression là một khoảng giá trị liên tục (continuous value), ví dụ như bài toán dự đoán giá nhà, dự đoán tiền lương, dự đoán phổ điểm,...



Hình 1: Linear Regression

2. Ý tưởng chính

- a. **Regression line:** phương trình đường thẳng dùng để dự đoán biến phụ thuộc Y từ các biến độc lập X

Ví dụ, $Y = B_0 + B_1X + \epsilon$. Trong đó

- B_0 là bias
- B_1 là hệ số góc
- Y là biến phụ thuộc, và phải là giá trị liên tục
- X là các biến độc lập, có thể là giá trị liên tục hoặc giá trị phân loại
- ϵ có thể là một giá trị bất kỳ nào đó.

- b. **Loss function** Tìm các hệ số sao cho giá trị thực Y và giá trị mô hình dự đoán \hat{Y} có sai khác nhau nhỏ nhất.

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (y_i - \bar{\mathbf{x}}_i \mathbf{w})^2$$

Hình 2: Loss Function

Trong đó

- \mathbf{w} = vector (cột) hệ số cần phải tối ưu
- $\bar{\mathbf{x}}$ = vector (hàng) dữ liệu đầu vào mở rộng

Mục tiêu: Tìm vector hệ số \mathbf{w} sao cho giá trị hàm mất mát (sai số) là nhỏ nhất.
Giá trị \mathbf{w} đó gọi là điểm tối ưu (optimal point)

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \mathcal{L}(\mathbf{w})$$

Hình 3: Vector \mathbf{w}

Giải phương trình đạo hàm bằng 0, ta thu được điểm tối ưu có dạng:

$$\mathbf{w} = \mathbf{A}^\dagger \mathbf{b} = (\bar{\mathbf{X}}^T \bar{\mathbf{X}})^\dagger \bar{\mathbf{X}}^T \mathbf{y}$$

Hình 4: Nghiệm tối ưu của \mathbf{w}

3. Lợi ích và hạn chế

(a) Lợi ích

- Áp dụng tốt với những tập dữ liệu linearly separable.
- Đơn giản, dễ hiểu, dễ thực hiện.

(b) Hạn chế

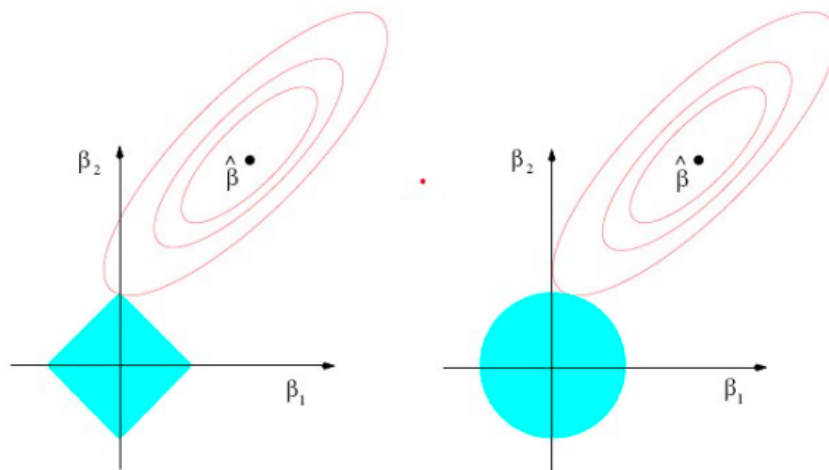
- Rất **nhạy cảm với nhiễu** (sensitive to noise), cần loại bỏ nhiễu (outlier) khi tiền xử lý dữ liệu.
- Dễ bị overfitting.

2.2 Mô hình hồi quy tuyến tính chính quy hóa (regularization)

2.2.1 LASSO

1. Mục tiêu

- Trong hồi qui Lasso, thay vì sử dụng thành phần điều chuẩn là norm chuẩn bậc hai thì chúng ta sử dụng norm chuẩn bậc 1.



Hình 5: Minh họa hình học hàm mục tiêu và miền giới hạn của LASSO (trái) và Ridge (phải)

2. Ý tưởng chính

- Bài toán tối ưu đối với hàm hồi qui Lasso tương đương với bài toán tối ưu với điều kiện ràng buộc về độ lớn của hàm mục tiêu:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N} \|\bar{\mathbf{X}}\mathbf{w} - \mathbf{y}\|_2^2 + \alpha \|\mathbf{w}\|_1$$

$$\text{subject : } \|\mathbf{w}\|_1 < C, C > 0$$

Hình 6: Hàm hồi quy LASSO

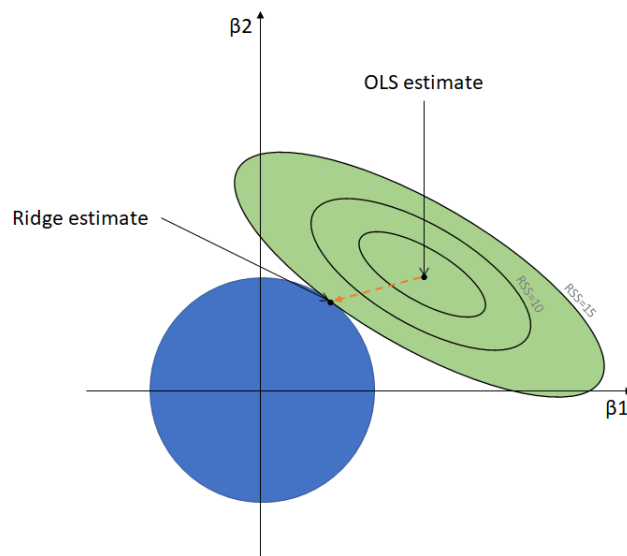
- Thành phần điều chuẩn norm bậc 1 cũng có tác dụng như một sự kiểm soát áp đặt lên hệ số ước lượng. Khi muốn gia tăng sự kiểm soát, chúng ta sẽ gia tăng hệ số α để mô hình trở nên bớt phức tạp hơn. Cũng tương tự như hồi qui Ridge chúng ta cùng phân tích tác động của α
 - Trường hợp $\alpha = 0$, thành phần điều chuẩn bị tiêu giảm và chúng ta quay trở về bài toán hồi quy tuyến tính

- Trường hợp α nhỏ thì vai trò của thành phần điều chuẩn trở nên ít quan trọng. Mức độ kiểm soát quá khớp của mô hình sẽ trở nên kém hơn.
- Trường hợp α lớn chúng ta muốn gia tăng mức độ kiểm soát lên độ lớn của các hệ số ước lượng.

2.2.2 RIDGE

1. Mục tiêu

- Một mục tiêu tiên quyết để có thể áp dụng được mô hình vào thực tiễn đó là chúng ta cần giảm thiểu hiện tượng quá khớp. Để thực hiện được mục tiêu đó, mô hình được huấn luyện được kì vọng sẽ nắm bắt được qui luật tổng quát từ tập huấn luyện (train dataset) mà qui luật đó phải đúng trên những dữ liệu mới mà nó chưa được học. Thông thường tập dữ liệu mới đó được gọi là tập kiểm tra (test dataset). Đây là một tập dữ liệu độc lập được sử dụng để đánh giá mô hình.



Hình 7: Circle and Ridge Estimate

2. Ý tưởng chính

- Giả định dữ liệu đầu vào bao gồm N quan sát là những cặp các biến đầu vào và biến mục tiêu $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$. Quá trình hồi qui mô hình sẽ tìm kiếm một véc tơ hệ số ước lượng $w = [w_0, w_1, \dots, w_p]$ sao cho tối thiểu hoá hàm mất mát dạng MSE:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 = \frac{1}{N} \|\bar{\mathbf{X}}\mathbf{w} - \mathbf{y}\|_2^2$$

Hình 8: Hàm mất mát MSE

- Giá trị hàm mất mát MSE chính là trung bình của tổng bình phương phần dư. Phần dư chính là chênh lệch giữa giá trị thực tế và giá trị dự báo. Tối thiểu hoá hàm mất mát nhằm mục đích làm cho giá trị dự báo ít chênh lệch so với giá trị thực tế, giá trị thực tế còn được gọi là ground truth.
- Tham khảo thêm tại đây: https://phamdinhkhanh.github.io/deepai-book/ch_ml/RidgedRegression.html

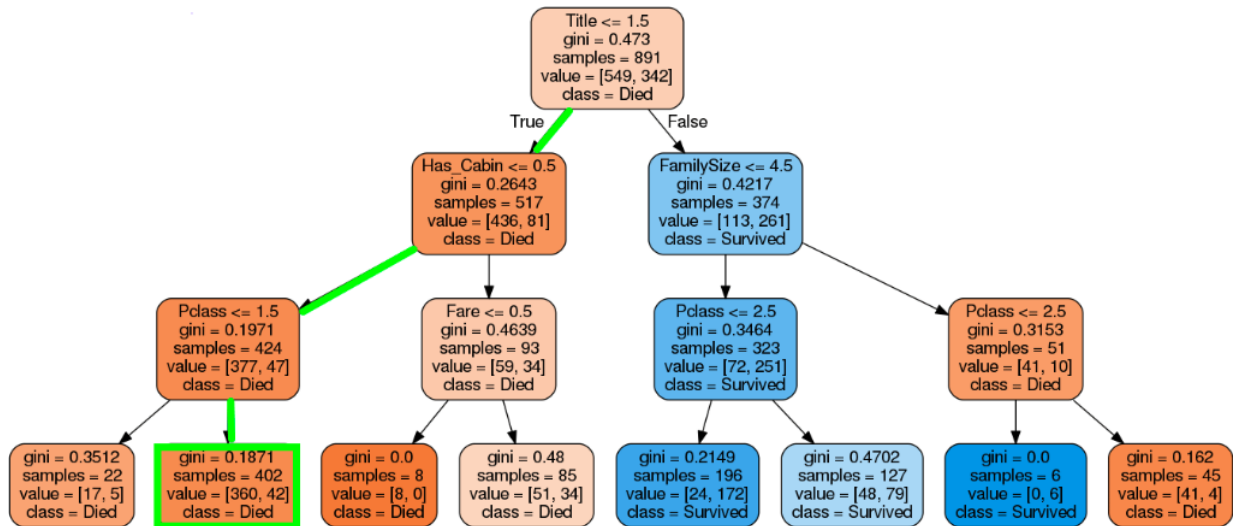
2.3 Mô hình phân loại

2.3.1 Decision Tree

1. Mục tiêu

- Decision Trees là một non-parametric supervised learning sử dụng cho các bài toán về classification và regression.
- Mục đích là sử dụng để xây dựng một model dự đoán giá trị dựa vào các thuộc tính và ràng buộc của dữ liệu.

2. Ý tưởng chính

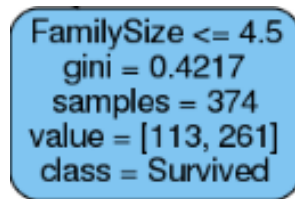


Hình 9: Decision Tree

Có hai loại node:

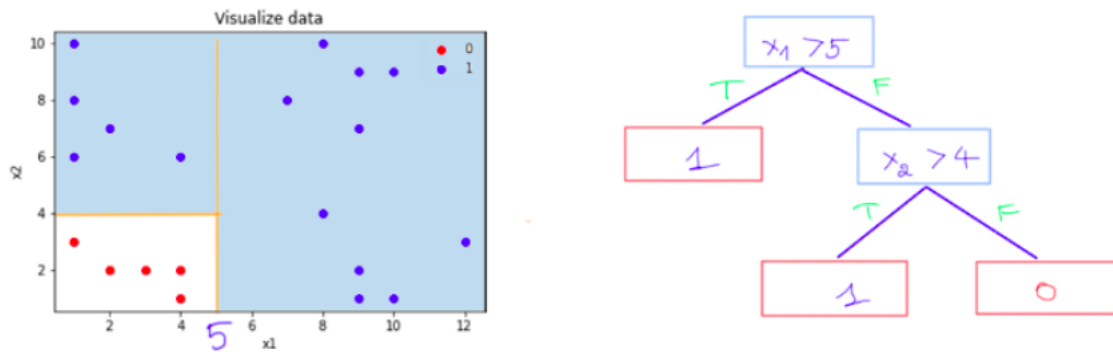
- **Node điều kiện:** bao gồm điều kiện kiểm tra
- **Node lá:** kết quả dự đoán

Giải thích sample node:



Hình 10: Sample Node

- Dòng 1 là điều kiện để tách node.
- Dòng 2 là chỉ số gini ở node nó.
- Dòng 3 là tổng số dữ liệu ở node đó.
- Dòng 4 Số lượng dữ liệu là Died và Survived, lần lượt là 113 và 261 nhận xét: $113 + 261 = 374$ (tổng số sample).
- Dòng 5 là kết quả dự đoán, do $261 > 113$, nên dự đoán là Survived.



Hình 11: Visualize Data Prediction

Xây dựng Decision Tree

- Đề xuất một số tiêu chí để quyết định điều kiện phân tách dữ liệu thành 2 phần mà dữ liệu mỗi phần có tính phân tách hơn dữ liệu ban đầu, dựa vào chỉ số để đánh giá: **entropy, information gain (ID3), gini index (CART)**
- Dựa vào các tiêu chí thiết kế model decision tree cho phù hợp.
- Một số thuật toán thường được sử dụng: ID3, entropy, information gain, CART (Classification And Regression Tree)

3. Lợi ích và hạn chế

(a) Lợi ích

- Áp dụng tốt với tập dữ liệu kích thước lớn.
- Dễ hiểu, dễ thực hiện hay visualization.
- Không bị ảnh hưởng bởi outliers.

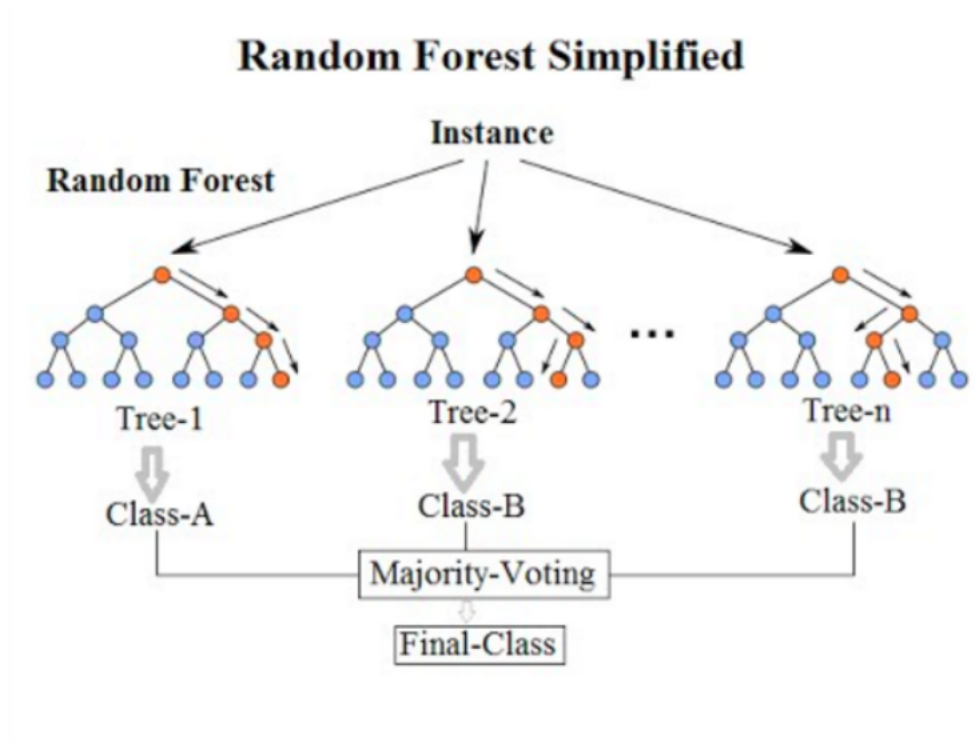
(b) Hạn chế

- Độ chính xác thấp, có xu hướng overfitting.
- Không quản lý được kích thước, có thể trở nên rất phức tạp.
- Mô hình dễ bị ảnh hưởng bởi sự thay đổi của tập huấn luyện.

2.3.2 Random Forest

1. Mục tiêu

- Xây dựng nhiều cây quyết định bằng thuật toán Decision Tree, tuy nhiên mỗi cây quyết định sẽ khác nhau (random). Sau đó kết quả dự đoán được tổng hợp từ các cây quyết định.



Hình 12: Random Forest

2. Ý tưởng chính

- Xây dựng: Dataset có n dữ liệu (sample) và mỗi dữ liệu có d thuộc tính (feature)
 - Lấy ngẫu nhiên n dữ liệu: Bootstrapping (random sampling with replacement) xây dựng tập n dữ liệu mới có thể có những dữ liệu bị trùng nhau
 - Chọn ngẫu nhiên ở k thuộc tính ($k < n$)
 - Dùng Decision Tree để xây dựng cây quyết định cho bộ dữ liệu vừa tạo.
 - Các thuộc tính cần chú ý: số lượng cây quyết định sẽ xây dựng, số lượng thuộc tính dùng để xây dựng cây, độ sâu tối đa, số phần tử tối thiểu trong 1 node.

3. Lợi ích và hạn chế

(a) Lợi ích

- Hiếm khi xảy ra overfitting và độ chính xác cao hơn so với Decision Tree.
- Với RF, mỗi cây quyết định đều có những yếu tố ngẫu nhiên:
 - Lấy ngẫu nhiên dữ liệu để xây dựng cây quyết định.
 - Lấy ngẫu nhiên các thuộc tính để xây dựng cây quyết định.
 - Do đó, kết quả cuối cùng của thuật toán Random Forest lại tổng hợp từ nhiều cây quyết định, thế nên thông tin từ các cây sẽ bổ sung thông tin cho nhau, dẫn đến mô hình có low bias và low variance

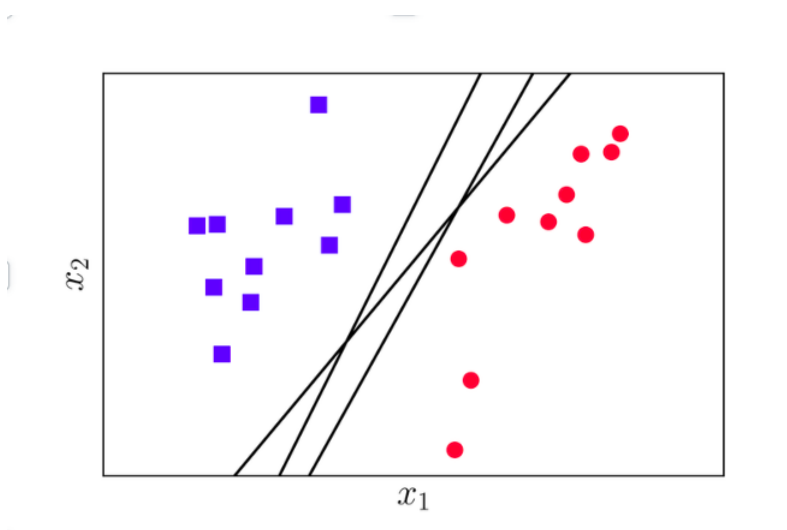
(b) **Hạn chế**

- Tốc độ training chậm. Khó visualization.

2.3.3 Support Vector Machine

1. Mục tiêu

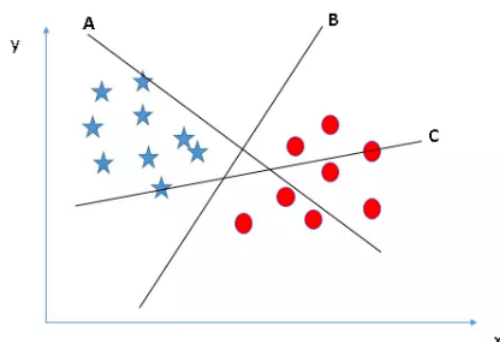
- Xem mỗi điểm dữ liệu là một điểm trong không gian n chiều (n là số đặc trưng), giá trị của mỗi đặc trưng ứng với giá trị của điểm tại 1 chiều cụ thể.
- Mục tiêu: Tìm 1 hyper-plane phân chia các lớp



Hình 13: Support Vector Machine

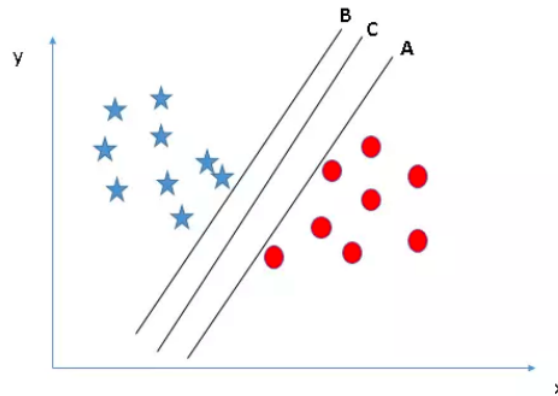
2. Ý tưởng chính

- **Tiêu chí 1:** Phân chia chính xác nhất.



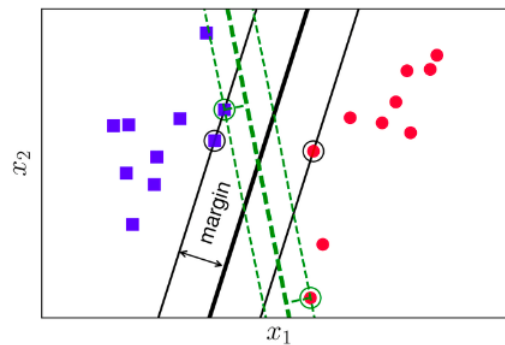
Hình 14: Phân chia chính xác nhất

- **Tiêu chí 2:** Phân chia công bằng nhất.



Hình 15: Phân chia công bằng nhất

- Khoảng cách từ điểm gần nhất của mỗi class tới đường phân chia là như nhau. Khoảng cách như nhau này được gọi là margin (lề)
- **Tiêu chí 3:** Phân chia văn minh nhất.



Hình 16: Phân chia văn minh nhất

- Xét hai cách phân chia bởi đường nét liền màu đen và đường nét đứt màu lục, rõ đường nét liền màu đen tốt hơn vì nó tạo ra một margin rộng hơn.
- Việc margin rộng hơn sẽ mang lại hiệu ứng phân lớp tốt hơn vì sự phân chia giữa hai classes là rạch ròi hơn \Rightarrow cần tìm hyper-plane sao cho margin là lớn nhất.

3. Lợi ích và hạn chế

(a) Lợi ích

- Xử lý trên không gian số chiều cao. Ví dụ bài toán phân loại văn bản.
- Linh hoạt, có thể áp dụng cho cả các bài toán non-linearly separable bằng cách áp dụng các hàm kernel.

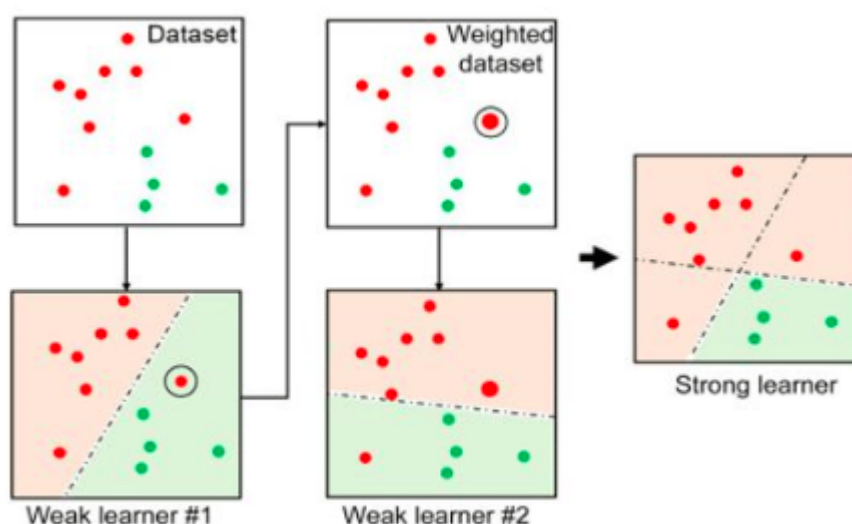
(b) Hạn chế

- Tuy nhiên, với số lượng thuộc tính (p) của tập dữ liệu lớn hơn rất nhiều so với số lượng dữ liệu (n) thì SVM sẽ thường cho kết quả không tốt.

2.3.4 AdaBoosting

1. Mục tiêu

- Cải thiện kết quả bằng cách sử dụng nhiều mô hình kết hợp với nhau để cùng giải quyết một vấn đề .
- Về ý tưởng, ta sẽ xây dựng một mô hình từ dữ liệu huấn luyện, sau đó các mô hình tiếp theo được tạo, cố gắng sửa các lỗi từ mô hình đầu tiên. Các mô hình được thêm vào cho đến khi tập huấn luyện được dự đoán hoàn hảo hoặc số lượng mô hình đạt ngưỡng cực đại.



Hình 17: AdaBoosting

2. Ý tưởng chính

- AdaBoost được sử dụng với các cây quyết định có độ sâu nhỏ. Sau khi cây đầu tiên được tạo, hiệu suất của cây trên mỗi mẫu huấn luyện được sử dụng làm thông tin để quyết định cây tiếp theo sẽ tập trung vào mẫu huấn luyện nào. Dữ liệu huấn luyện khó dự đoán sẽ được đánh trọng số lớn hơn so với các trường hợp khác.
- Các mô hình được tạo lần lượt, hiệu suất của mô hình trước sẽ ảnh hưởng đến cách mô hình sau được xây dựng.

3 Áp dụng các mô hình Machine Learning vào dự đoán điểm số và phân loại sinh viên

3.1 Đọc dữ liệu

Đọc dữ liệu trực tiếp từ Google Drive, lưu ý thêm dataset vào drive trước khi đọc dữ liệu.

```
# load datasets
from google.colab import drive
drive.mount('/content/gdrive')
mat = pd.read_csv("gdrive/My Drive/student-mat.csv", sep=',')
por = pd.read_csv("gdrive/My Drive/student-por.csv", sep=',')

# merge datasets
df = pd.concat([mat,por])
```

3.2 Tiền xử lý dữ liệu

3.2.1 Đổi tên các column

Đặt lại tên cho các cột của dataframe và chuyển đổi thuộc tính điểm cuối kì (nhân) từ kiểu giá trị **numeric** sang kiểu giá trị **categorical**.

```
# rename column labels
df.columns =
['school', 'sex', 'age', 'address', 'family_size', 'parents_status',
'mother_education', 'father_education', 'mother_job', 'father_job',
'reason', 'guardian', 'commute_time', 'study_time', 'failures',
'school_support', 'family_support', 'paid_classes', 'activities',
'nursery', 'desire_higher_edu', 'internet', 'romantic', 'family_quality',
'free_time', 'go_out', 'weekday_alcohol_usage', 'weekend_alcohol_usage',
'health', 'absences', 'period1_score', 'period2_score', 'final_score']

# convert final_score to categorical variable
# Good:15~20 Fair:10~14 Poor:0~9
df['final_grade'] = 'na'
df.loc[(df.final_score >= 15) & (df.final_score <= 20), 'final_grade'] =
    'good'
df.loc[(df.final_score >= 10) & (df.final_score <= 14), 'final_grade'] =
```


'fair'

```
df.loc[(df.final_score >= 0) & (df.final_score <= 9), 'final_grade'] =  
'poor'
```

Dataset sau khi đổi tên column

	school	sex	age	address	family_size	parents_status	mother_education	father_education	mother_job	father_job	...	free_time	go_out	weekday_alcohol_usage	weeken
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	3	4	1.0	
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	3	3	1.0	
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	3	2	2.0	
3	GP	F	15	U	GT3	T	4	2	health	services	...	2	2	1.0	
4	GP	F	16	U	GT3	T	3	3	other	other	...	3	2	1.0	

5 rows × 34 columns

Hình 18: Dữ liệu sau khi đổi tên column

3.2.2 Xử lý missing value

Kiểm tra sơ bộ các thông tin của dataframe để phát hiện các cột chứa giá trị khuyết (missing value) và định hình phương pháp xử lý.

Trong trường hợp này, nhóm lựa chọn giải pháp xóa đi các cột có giá trị khuyết.

```
# drop columns with NaN  
df = df.dropna(axis=1)  
df.info()
```

Sau khi xử lý missing value, lúc này dữ liệu của nhóm còn lại **1034 entries** và **28 columns**.

```
<class 'pandas.core.frame.DataFrame'  
RangeIndex: 1044 entries, 0 to 1043  
Data columns (total 30 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   family_size           1044 non-null   object  
5   parents_status        1044 non-null   object  
6   mother_job            1044 non-null   object  
7   father_job            1044 non-null   object  
8   guardian              1044 non-null   object  
9   commute_time          1044 non-null   int64  
10  failures              1044 non-null   int64  
11  school_support         1044 non-null   object  
12  family_support         1044 non-null   object  
13  paid_classes           1044 non-null   object  
14  activities             1044 non-null   object  
15  nursery               1044 non-null   object  
16  desire_higher_edu      1044 non-null   object  
17  internet               1044 non-null   object  
18  romantic               1044 non-null   object  
19  family_quality         1044 non-null   int64  
20  free_time              1044 non-null   int64  
21  go_out                 1044 non-null   int64  
22  weekday_alcohol_usage  1044 non-null   int64  
23  weekend_alcohol_usage  1044 non-null   int64  
24  health                 1044 non-null   int64  
25  absences               1044 non-null   int64  
26  period1_score          1044 non-null   int64  
27  period2_score          1044 non-null   int64  
28  final_score            1044 non-null   int64  
29  final_grade            1044 non-null   object  
dtypes: int64(13), object(17)  
memory usage: 244.8+ KB
```

Hình 19: Dữ liệu sau khi đã xử lý missing value

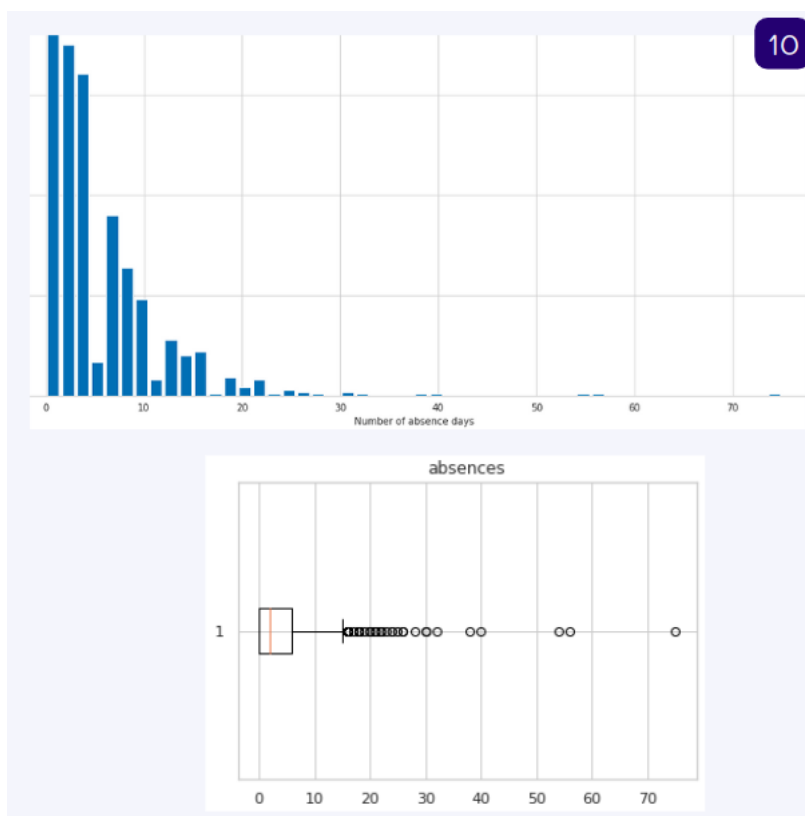
3.2.3 Xử lý outliers

Sau khi xử lý missing values, để đảm bảo kết quả train có độ chính xác cao hơn và hạn chế sai lệch trong xây dựng mô hình, nhóm sẽ tiến hành loại bỏ outliers.

Trong dataset của nhóm, cần xử lý outliers trường **absences**.

```
# histogram
plt.figure(figsize=(15,15))
plt.hist(df.absences, bins=50, rwidth=0.8)
plt.xlabel('Number of absence days')
plt.ylabel('Number of students')
plt.show()
# use boxplot
fig1, ax1 = plt.subplots()
ax1.set_title('absences')
ax1.boxplot(df.absences, vert=False)
```

Trong các đồ thị biểu diễn trường absences của dataset, chúng ta có thể nhận thấy rõ có xuất hiện nhiều outliers cần được xử lý.



Hình 20: Phân phối dữ liệu theo trường absences trước khi xử lý outliers

Để xử lý outliers, nhóm chọn giải pháp xử lý là xác định **IQR** (Inter Quantile Range) nơi phạm vi phần lớn dữ liệu tập trung và xử lý các ngoại lệ nằm ngoài đoạn này bằng phương pháp **Quantile based flooring and capping**.

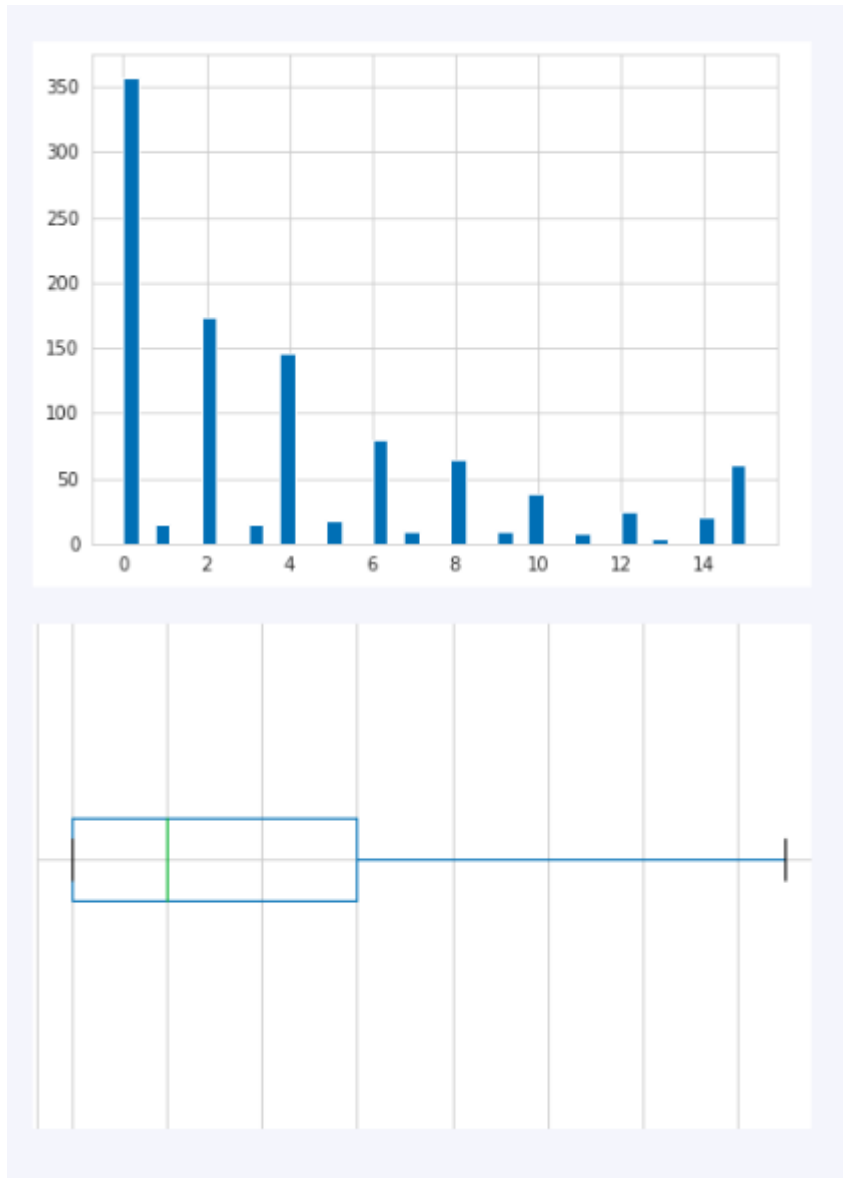
```
# use Quantile based flooring and capping method to remove outliers of
'absence' attribute
from typing import Tuple
from sklearn.base import BaseEstimator, TransformerMixin

def find_boxplot_boundaries(
    col: pd.Series, whisker_coeff: float = 1.5
) -> Tuple[float, float]:
    """Find minimum and maximum in boxplot.
    Args:
        col: a pandas series of input.
        whisker_coeff: whisker coefficient in box plot
    """
    Q1 = col.quantile(0.25)
    Q3 = col.quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - whisker_coeff * IQR
    upper = Q3 + whisker_coeff * IQR
    return lower, upper

class BoxplotOutlierClipper(BaseEstimator, TransformerMixin):
    def __init__(self, whisker_coeff: float = 1.5):
        self.whisker = whisker_coeff
        self.lower = None
        self.upper = None

    def fit(self, X: pd.Series):
        self.lower, self.upper = find_boxplot_boundaries(X, self.whisker)
        return self

    def transform(self, X):
        return X.clip(self.lower, self.upper)
```



Hình 21: Phân phối dữ liệu theo trường *absences* sau khi xử lý outliers

Tương tự, chúng ta sẽ tiếp tục xử lý outliers ở **period1_score** and **period2_score**

```
# remove outliers of period1_score and period2_score
def score_outliers_remove(column_name):
    global df
    categories = df[column_name].value_counts().to_frame()
    categories = categories.reset_index()
    categories.columns = ['unique_values', 'counts']
    drop_value = list(categories[categories["counts"] < 5].unique_values)

    for i in range(len(drop_value)):
        df = df[df[column_name] != drop_value[i]]
```

```
score_outliers_remove('period1_score')
score_outliers_remove('period2_score')

print(df.period1_score.value_counts())
print(df.period2_score.value_counts())
```

3.2.4 Mã hóa đặc trưng

Tiến hành **mã hóa đặc trưng** (Label Encoder) để chuyển các dữ liệu về dạng **numeric** giúp các mô hình có thể hiểu được quan hệ đại số giữa các biến.

```
from sklearn.preprocessing import LabelEncoder

object_attribute = ['school', 'sex', 'address', 'family_size',
                   'parents_status', 'mother_job', 'father_job', 'reason',
                   'school_support', 'family_support', 'paid_classes', 'activities',
                   'desire_higher_edu', 'internet']
for attribute in object_attribute:
    le_name = "le_" + attribute
    globals()[le_name] = LabelEncoder()
    df[attribute] = globals()[le_name].fit_transform(df[attribute])
```

```
----- school
[0 1]
----- sex
[0 1]
----- address
[1 0]
----- family_size
[0 1]
----- parents_status
[0 1]
----- mother_job
[0 1 2 3 4]
----- father_job
[4 2 3 1 0]
----- reason
[0 2 1 3]
----- school_support
[1 0]
----- family_support
[0 1]
----- paid_classes
[0 1]
----- activities
[0 1]
----- desire_higher_edu
[1 0]
----- internet
[0 1]
```

Hình 22: Label Encoder các thuộc tính của dữ liệu

3.2.5 Chia dữ liệu thành tập huấn luyện và tập kiểm nghiệm

Tiến hành chia dữ liệu thành **tập huấn luyện và tập kiểm nghiệm** (train/test), ở đây nhóm quyết định chia theo tỷ lệ **75% train, 25% test**.

```
# dataset train_test_split
from sklearn.model_selection import train_test_split
X = df.drop(['final_grade', 'final_score'], axis=1)
y = df.final_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
                                                    random_state=1)
```

3.3 Huấn luyện mô hình

3.3.1 Regression Model

1. Linear Regression

(a) Khởi tạo mô hình với thư viện sklearn

Đầu tiên, chúng ta sẽ import model **LinearRegression** từ thư viện **sklearn**

```
# create LR model and fit training dataset
from sklearn.linear_model import LinearRegression

modelLR = LinearRegression()
```

(b) Huấn luyện mô hình và dự đoán

- Sử dụng **tập dữ liệu train** fit vào model.

```
modelLR.fit(X_train, y_train)
```

- Sau khi train xong **modelLR**, sử dụng **tập dữ liệu x_test** để test output của mô hình vừa train, kết quả này được ghi vào **Y_pred**.

```
Y_pred = modelLR.predict(X_test)
```

```
Y_pred
array([[15.90951384,  9.94387041, 10.43128369, 17.37765641, 11.38974017,
        9.61702153, 11.78507446, 15.76653321, 14.411683,  7.82400205,
        7.49156179,  9.63473965, 12.3910635,  10.06910529, 12.25247902,
        7.34729839,  6.72291584, 13.63263031, 16.244661,  9.52330936,
        9.47127304, 15.89378925, 14.06653157, 14.40406753, 12.30935134,
        18.78345684, 12.51843689, 12.75835512, 14.08923464,  9.72749219,
        12.01329474,  9.42618422, 11.02036552, 13.62070992,  5.92172486,
        9.09171802, 13.48140246, 16.33103541, 12.43614874, 13.0693453,
        7.94457957,  8.74004261,  9.861865,  9.995736,  9.68579207,
        17.8046723, 17.21417287, 14.29678679, 14.57031445, 11.19300265,
        8.92178307, 15.26335188, 16.78799231, 17.26630975,  6.27779748,
        14.85463825, 14.88644999,  5.6733791,  9.73356648, -0.12420775,
        13.96623543, 12.99923272, 15.21028817, 13.01830237,  7.63588229,
        9.57046788, 13.83318541, 12.23273367,  9.40060944, 12.22282807,
        11.55789777,  7.3533585, 12.82969672,  7.62482394, -0.43490908,
        13.23056708, 17.10099955, 13.4156691, 15.59412835,  9.23300607,
        12.3853767,  9.05990629,  9.95489638, -0.14851383, 10.82433681,
        6.3868156,  6.79831884,  5.05120024,  6.56015361, 12.9352147,
        16.50360743,  7.71358423, 11.4675717, 10.25800174, 11.06125485,
        11.52897928, 10.40007581, 16.63854579,  7.27819101, 12.06904787,
        11.75728009, 10.58070345,  6.54597119,  6.02538816, 12.07474941,
        15.07485342,  7.08848496, 16.57197086, 12.82614865,  9.40130671,
        12.33531929, 12.33543464, 10.77726168,  9.95522886, 13.71471879,
        7.55082954, 18.72568874,  8.69087405, 10.03297775, 16.14532657,
        12.56670398,  8.34486216, 14.8486582,  9.3353748,  8.14833158,
        9.04232606, 16.12210932, 14.99988396, 14.97079913, 10.9660139,
        5.95191145, 10.94917972,  8.35857765, 10.32853258,  8.99276177,
        13.98211423, 11.19673331, 10.81715811, 10.69709732, 18.77492858,
        15.04638493, -0.33900374, 10.55221937, 12.1511509, 10.89831808,
        8.9790343,  6.7482405,  9.7516688, 12.38347708,  5.20246372,
        5.66981151, -0.53864456, 14.99680718, 15.85212206, 16.38648969,
        14.47459252,  6.76672413,  8.90230371, 14.63237403,  6.58291035,
        10.34783583, 15.51694031, 15.74289117, 12.39655956, 10.18140196,
        12.05478063, 12.9135331, 13.48106353,  0.12912272, 15.17423557,
        18.5501288, 10.7742372, 10.37401426, 12.64151163, 13.39383527,
        14.52552095, 12.20849588,  7.17558669,  8.25524432, 12.20659899,
        12.3128442, -0.12262445,  8.68175229, 12.26627123, 11.88456997,
        11.54840699, 11.71252275, 16.2436274, 12.44917596, 17.1606772,
        10.03021373, 12.36609129, 15.25306646,  8.50368228, 12.19352167,
        15.82671733, 13.60328624, 10.39196789, 11.25285714,  7.22037857,
        13.80588733, 17.94473442, 12.78078544, 10.77842113, 17.30878289,
        11.50084943, 13.27715035, 14.69189096, 11.85521901, 12.25762698,
        8.4011949,  9.10491914, 14.8426647, 10.49348531, 10.67788,
        12.19699884, 10.78129191, 15.47867293,  9.45756773,  9.99871604,
        12.50361364, 15.93287121, 10.67266275, 12.59795812,  9.65733841,
        10.82965325, 10.13003328, 14.59416319, 13.01025064,  8.32272226,
        10.75121365, 10.36695938,  6.14916025, 11.10404022, 14.1546394,
        11.7942338,  9.12107965, 13.09932503,  8.8877932,  7.49780833,
        12.49099889,  8.6940622, 18.11003966, 13.2678197,  9.16762721,
        11.85943011, 10.29738987,  9.97462993,  9.73037701, 12.25650791,
        9.48851462, 13.30907594, 13.55360211, 14.61666774, 11.75177408,
        17.68058648, 11.66047122, 15.13609316])
```

Hình 23: Kết quả Y_{pred} của modelLR

2. LASSO & RIDGE

(a) Khởi tạo mô hình với thư viện sklearn

Đầu tiên, chúng ta sẽ import model **LASSO & RIDGE** từ thư viện **sklearn**

```
# regularization for linear regressor
from sklearn.linear_model import Lasso, Ridge
```

(b) Huấn luyện mô hình và dự đoán

- Tiến hành train model và dự đoán.

```
lasso_reg = Lasso().fit(X_train, y_train)
ridge_reg = Ridge().fit(X_train, y_train)
```

```
lasso_p = lasso_reg.predict(X_test)
ridge_p = ridge_reg.predict(X_test)
```

- Tính các chỉ số **MSE**, **RMSE**, **R2** giữa các mô hình. Đánh giá thông qua các chỉ số RMSE, MSE, R2 Score đối với các mô hình Regression như sau:
 - **RMSE và MSE**: Càng thấp càng chính xác.
 - **R2 Score**: Càng gần 1 càng phù hợp.

```
from sklearn.metrics import mean_squared_error, r2_score
def calculate_metrics(y_test, y_pred):
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2_scors = r2_score(y_test, y_pred)

    print("MSE: ", mse)
    print("RMSE: ", rmse)
    print("R2_score: ", r2_scors)
```

```
[ ] calculate_metrics(y_test, Y_pred)
```

```
MSE: 1.3715982258401438
RMSE: 1.171152520314986
R2_score: 0.9006543341434604
```

```
[ ] calculate_metrics(y_test, lasso_p)
```

```
MSE: 1.3413765523347927
RMSE: 1.1581781177067683
R2_score: 0.9028433077226942
```

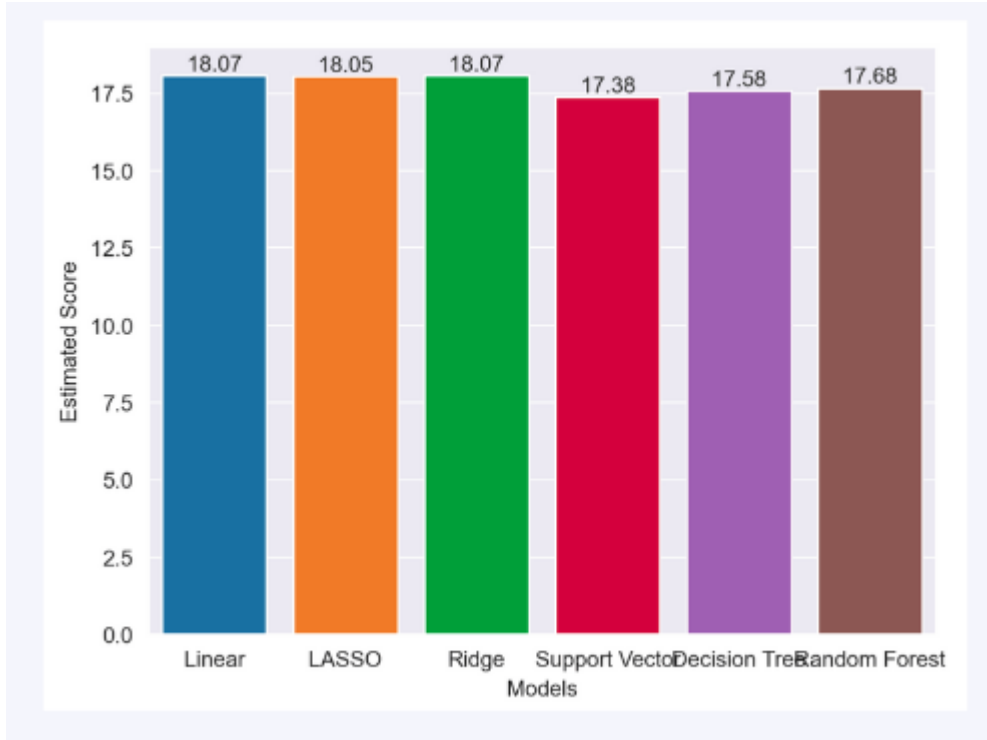
```
[ ] calculate_metrics(y_test, ridge_p)
```

```
MSE: 1.3699496167776126
RMSE: 1.1704484682281457
R2_score: 0.9007737438670724
```

Hình 24: Bảng so sánh kết quả RMSE, MSE, R2

3. Nhận xét và đánh giá

- **Nhận xét:** Ta thấy có sự sai lệch nhẹ về giá trị dự đoán khi sử dụng các mô hình khác nhau.



Hình 25: Bảng so sánh kết quả dự đoán của các Regression Model

- **Giải thích**
 - **Độ tương thích:** Sự phù hợp của dữ liệu với mỗi mô hình là khác nhau
 - **Độ chính xác:** Thể hiện qua các số liệu đánh giá mô hình, ví dụ như MSE, RMSE, R2 Score...như chúng ta đã tính ở phần trên.
 - **Overfitting:** Khi dữ liệu huấn luyện quá "khớp" với mô hình.

- **Đánh giá**



Hình 26: Bảng so sánh kết quả RMSE, MSE, R2

- **Kết luận:** Mô hình có khả năng ước lượng chính xác nhất là **Random Forest**.

3.3.2 Classification Model

1. Decision Tree

(a) **Khởi tạo mô hình với thư viện sklearn**

Khởi tạo mô hình **Decision Tree**

```
from sklearn.tree import DecisionTreeClassifier

# default value of criterion = "gini", min_samples_split = 2,
min_samples_leaf = 1
Model = DecisionTreeClassifier()
```

(b) **Huấn luyện mô hình và dự đoán**

- **Tiến hành train model**

```
Model.fit(X_train, y_train)
y_pred = Model.predict(X_test)
```

- **Non-Parameters Tunning**

```
MSE: 2.7141472868217056
RMSE: 1.6474669304182423
R2_score: 0.8034127163755544
```

Hình 27: RMSE, MSE, R2 của model Decision Tree

- **Parameters Tunning**

```
from matplotlib.legend_handler import HandlerLine2D
max_depths = np.linspace(1, 32, 32, endpoint=True)
min_samples_leafs = np.linspace(1, 50, 50,
                                endpoint=True).astype(int)
min_samples_splits = np.linspace(2, 50, 49,
                                endpoint=True).astype(int)
```

```
def dt_score_calculate(para_value_list, para_name):
    train_results = []
    test_results = []
    for element in para_value_list:
        if para_name == 'max_depth':
            dt = DecisionTreeClassifier(random_state=0,
                                        max_depth=element)
        elif para_name == 'min_samples_leaf':
            dt = DecisionTreeClassifier(random_state=0,
                                        min_samples_leaf=element)
        elif para_name == 'min_samples_split':
```

```
dt = DecisionTreeClassifier(random_state=0,
                             min_samples_split=element)

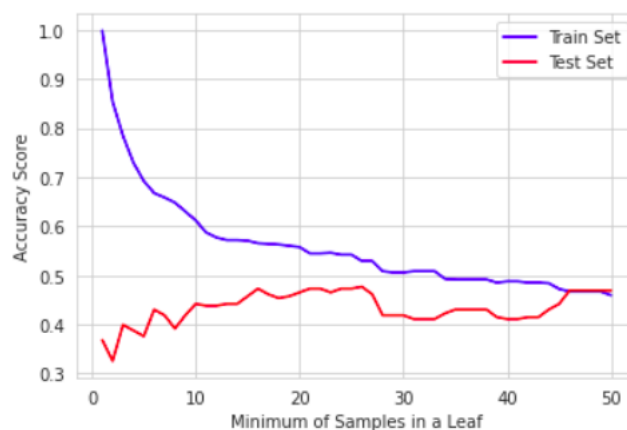
dt.fit(X_train, y_train)

train_score = dt.score(X_train, y_train)
train_results.append(train_score)

test_score = dt.score(X_test, y_test)
test_results.append(test_score)

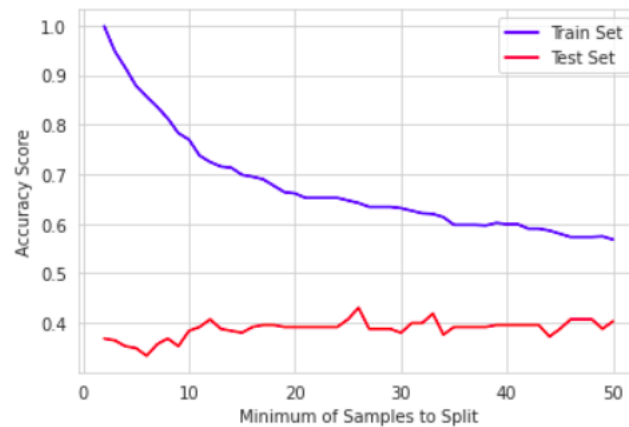
y_label = ""
if para_name == 'max_depth':
    y_label = 'Tree Depth'
elif para_name == 'min_samples_leaf':
    y_label = 'Minimum of Samples in a Leaf'
elif para_name == 'min_samples_split':
    y_label = 'Minimum of Samples to Split'
line1 = plt.plot(para_value_list, train_results, 'b',
                  label = "Train Set")
line2 = plt.plot(para_value_list, test_results, 'r',
                  label= "Test Set")
plt.ylabel("Accuracy Score")
plt.xlabel(y_label)
plt.legend()
plt.show()
```

i. Max depth



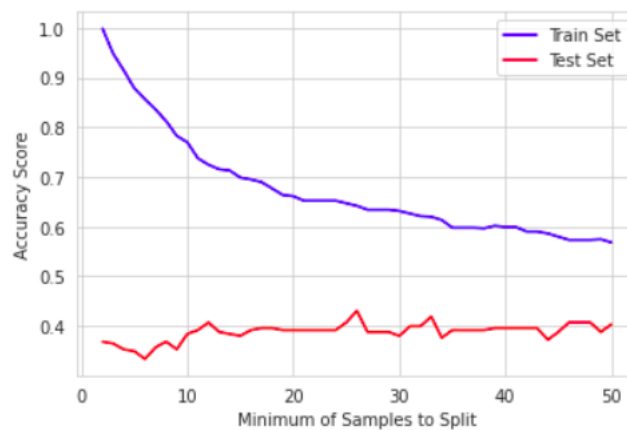
Hình 28: Max depth

ii. Min Samples Leaf



Hình 29: Min Samples Leaf

iii. Min Samples Split



Hình 30: Min Samples Split

• Grid Search

```
from sklearn.tree import DecisionTreeClassifier
# try grid search for hyperparameter tuning to find a better
# model which offers higher score for test set
from sklearn.model_selection import GridSearchCV
params = {'min_samples_leaf' : list(range(15,25)),
          'max_depth' : list(range(1,10)), 'min_samples_split' :
          list(range(30,35))}
dtc_grid = GridSearchCV(DecisionTreeClassifier(), params,
                        verbose=1, refit=True)
# fitting the model for grid search
dtc_grid.fit(X_train_c, y_train_c)
```

- **Output**

```
Prediction on test set:  
Accuracy Score: 0.8488372093023255  
Confusion Matrix:  
[[134  4  9]  
 [ 9 49  0]  
 [17  0 36]]  
Prediction on training set:  
Accuracy Score: 0.8952134540750324  
Confusion Matrix:  
[[438  7 17]  
 [19 120  0]  
 [38  0 134]]
```

Hình 31: *Accuracy Score & Confusion Matrix của Decision Tree*

2. Random Forest

(a) Khởi tạo mô hình với thư viện sklearn

```
from sklearn.ensemble import RandomForestClassifier  
max_depths = np.linspace(1, 32, 32, endpoint=True).astype(int)  
n_estimators = np.linspace(1, 40, 40, endpoint=True).astype(int)
```

(b) Huấn luyện mô hình và dự đoán

- **Tiến hành train model**

```
Model.fit(X_train, y_train)  
y_pred = Model.predict(X_test)
```

- **Store Calculation**

```
from matplotlib.legend_handler import HandlerLine2D  
max_depths = np.linspace(1, 32, 32, endpoint=True)  
min_samples_leafs = np.linspace(1, 50, 50,  
                                endpoint=True).astype(int)  
min_samples_splits = np.linspace(2, 50, 49,  
                                endpoint=True).astype(int)
```

```
def rf_score_calculate(para_value_list, para_name):  
    train_results = []  
    test_results = []  
    for element in para_value_list:
```

```
if para_name == 'n_estimators':
    rf = RandomForestClassifier(random_state=0,
                               n_estimators=element)
elif para_name == 'max_depth':
    rf = RandomForestClassifier(random_state=0,
                               max_depth=element)

rf.fit(X_train, y_train)

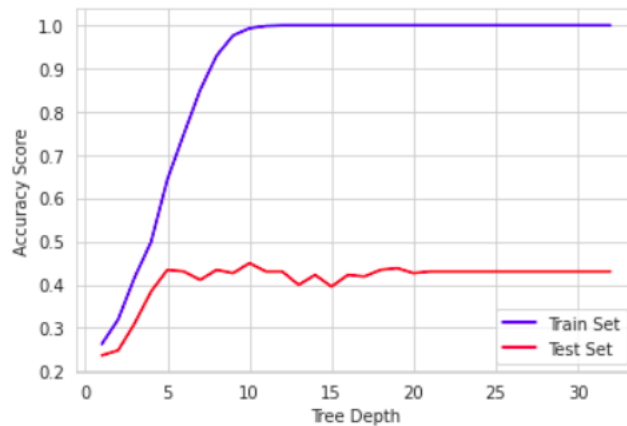
train_score = rf.score(X_train, y_train)
train_results.append(train_score)

test_score = rf.score(X_test, y_test)
test_results.append(test_score)

y_label = ""
if para_name == 'max_depth':
    y_label = 'Tree Depth'
elif para_name == 'n_estimators':
    y_label = 'Number of Trees'

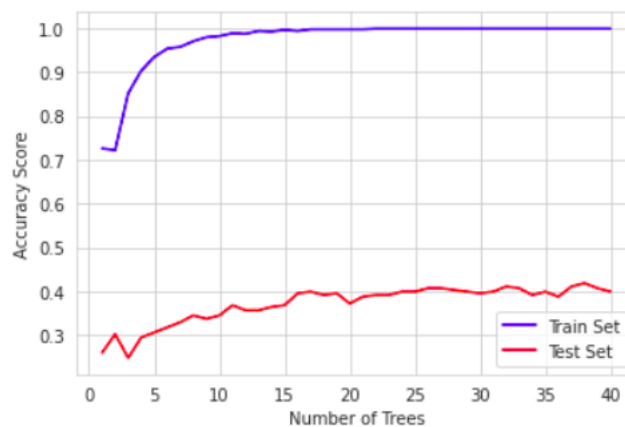
line1 = plt.plot(para_value_list, train_results, 'b', label
                 = "Train Set")
line2 = plt.plot(para_value_list, test_results, 'r', label=
                 "Test Set")
plt.ylabel("Accuracy Score")
plt.xlabel(y_label)
plt.legend()
plt.show()
```

i. Max depth



Hình 32: Max depth

ii. n_estimators



Hình 33: N_Estimators

• Grid Search

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

params = {'n_estimators' : list(range(20,30)), 'max_depth' :
          list(range(5,15))}
rfc_grid = GridSearchCV(RandomForestClassifier(), params,
                        verbose=1, refit=True)
# fitting the model for grid search
rfc_grid.fit(X_train_c, y_train_c)
```


- Output

```
Prediction on test set:  
Accuracy Score: 0.8875968992248062  
Confusion Matrix:  
[[139  2  6]  
 [ 7 51  0]  
 [ 14  0 39]]  
Prediction on training set:  
Accuracy Score: 0.9935316946959897  
Confusion Matrix:  
[[462  0  0]  
 [ 2 137  0]  
 [ 3  0 169]]
```

Hình 34: Accuracy Score & Confusion Matrix của Random Forest

3. Support Vector Machine

(a) Khởi tạo mô hình với thư viện sklearn

```
# Support Vector Machine  
from sklearn.svm import SVC  
from sklearn.model_selection import GridSearchCV  
  
params = {'kernel' : ['rbf','poly','sigmoid'], 'C' : [1,10,100]}  
svm_grid = GridSearchCV(SVC(), params, verbose=1, refit=True)
```

(b) Huấn luyện mô hình và dự đoán

- Tiến hành train model

```
# fitting the model for grid search  
svm_grid.fit(X_train_c, y_train_c)  
  
modelSVM = svm_grid.best_estimator_  
modelSVM.fit(X_train_c, y_train_c)  
svm_p = modelSVM.predict(X_test_c)  
svm_p_train = modelSVM.predict(X_train_c)
```

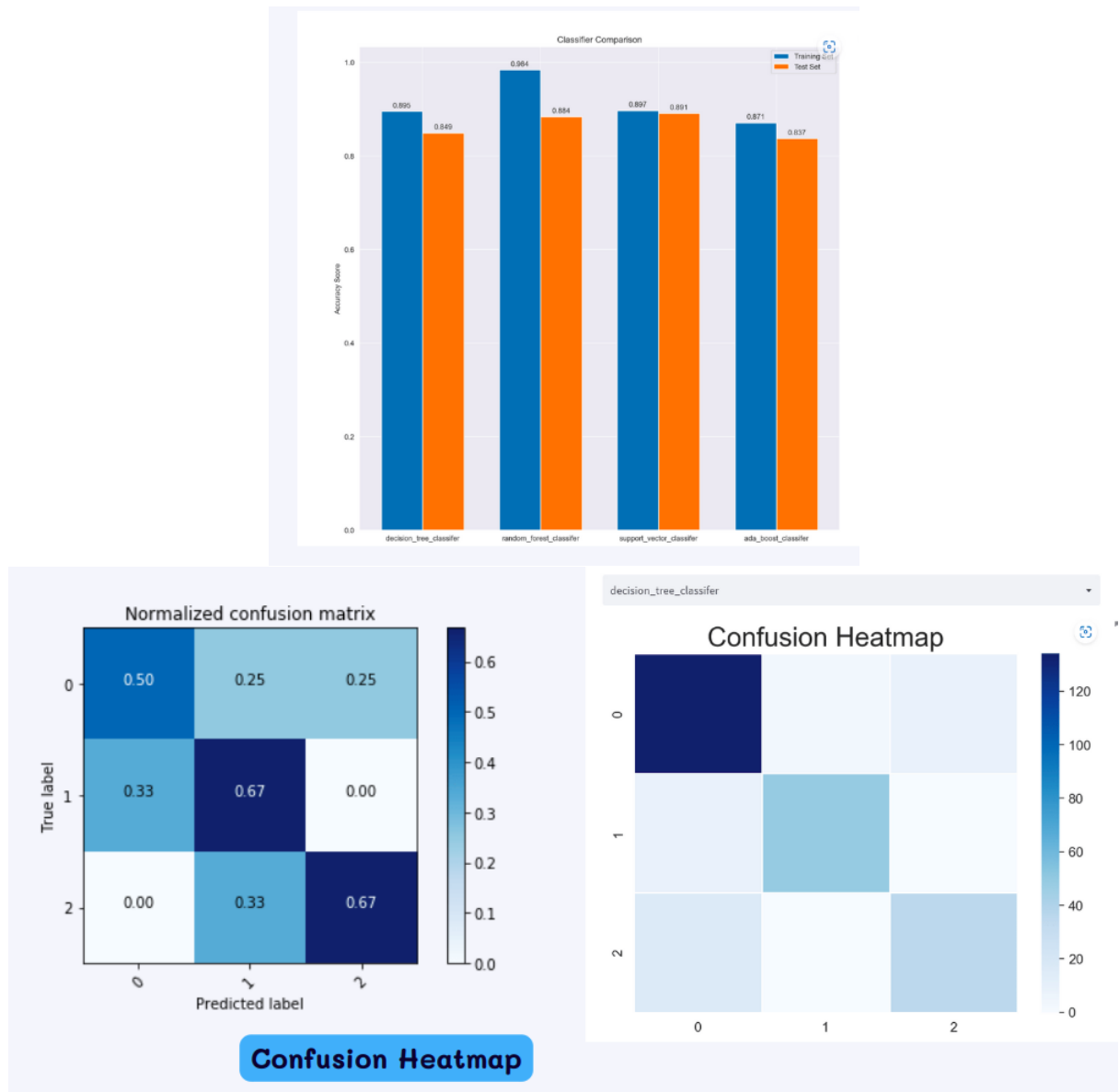
- **Output**

```
Prediction on test set:  
Accuracy Score: 0.8914728682170543  
Confusion Matrix:  
[[137  3  7]  
 [ 7 51  0]  
 [ 11  0 42]]  
Prediction on training set:  
Accuracy Score: 0.8965071151358344  
Confusion Matrix:  
[[430  8 24]  
 [ 17 122  0]  
 [ 31  0 141]]
```

Hình 35: Accuracy Score & Confusion Matrix của Support Vector Machine

4. Nhận xét và đánh giá

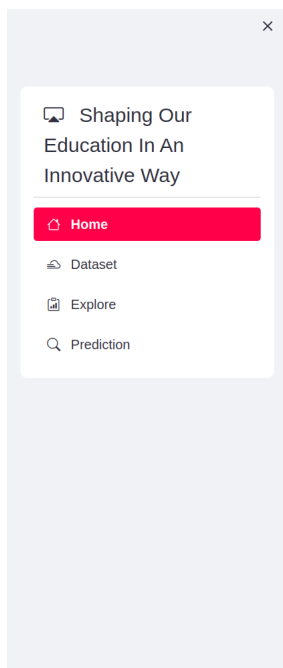
- **Nhận xét: Random Forest Classifier** có **Accuracy Score** trên training set cao nhất nhưng đối với test set thì **Accuracy Score** lại thấp, chứng tỏ nó bị overfitting nhiều nhất.
- **Giải thích**
 - **Accuracy Score**: Tỷ lệ giữa số điểm được dự đoán đúng và tổng số điểm trong tập dữ liệu kiểm thử
 - **Confusion Matrix**: Có bao nhiêu điểm dữ liệu thực sự thuộc vào một class, và được dự đoán là rơi vào một class?
 - So sánh Accuracy Score của mô hình trên training set và test set có thể chẩn đoán mô hình đang bị overfitting/underfitting.
- **Kết luận**: Mô hình có khả năng phân loại chính xác nhất là **Support Vector Machine**, đồng thời cũng ít bị overfitting nhất..



Hình 36: Bảng so sánh kết quả dự đoán của các Classification Model

4 Demo App E-learning Analysis System

4.1 Giao diện



Welcome to our project

The academic performance of student is usually stored in various formats like files, documents, records etc. The available data would be analyzed to extract useful information. It becomes difficult to analyze student data by applying statistical techniques or other traditional database management tools. Hence there is a need to develop an automated tool for student performance analysis that would analyze student performance and will guide them by displaying the areas where they need improvement, in order to contribute to a student's overall growth by generating a score card for the same.

The proposed system will display results of student performance on a single click action by the user, thus inducing automation and reducing efforts of staff in analyzing student performance manually. The proposed system finds out student trends on the basis of outcomes of students academic performance, strengths, weakness, hobbies and extra - curricular activities. Academic data includes unit test, students theory, practicals and term work marks. This data gathered will be processed by classification algorithm of data mining. A result from classification algorithm will be recognizing as Trend. This trend will help us to track where the students excel and where not and what are their abilities which can be enhanced. The analysis will summarize the outcome and will classify students based on the results.

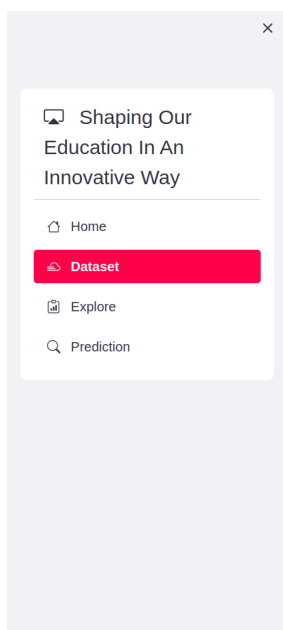
Result

Dataset: <http://archive.ics.uci.edu/ml/datasets/Student+Performance#>

Google Colab: [Assjgnment3_Group1.ipynb](#)

Landing Page: [Assjgnment1_Group1](#)

Hình 37: Home Page



Dataset

Upload CSV



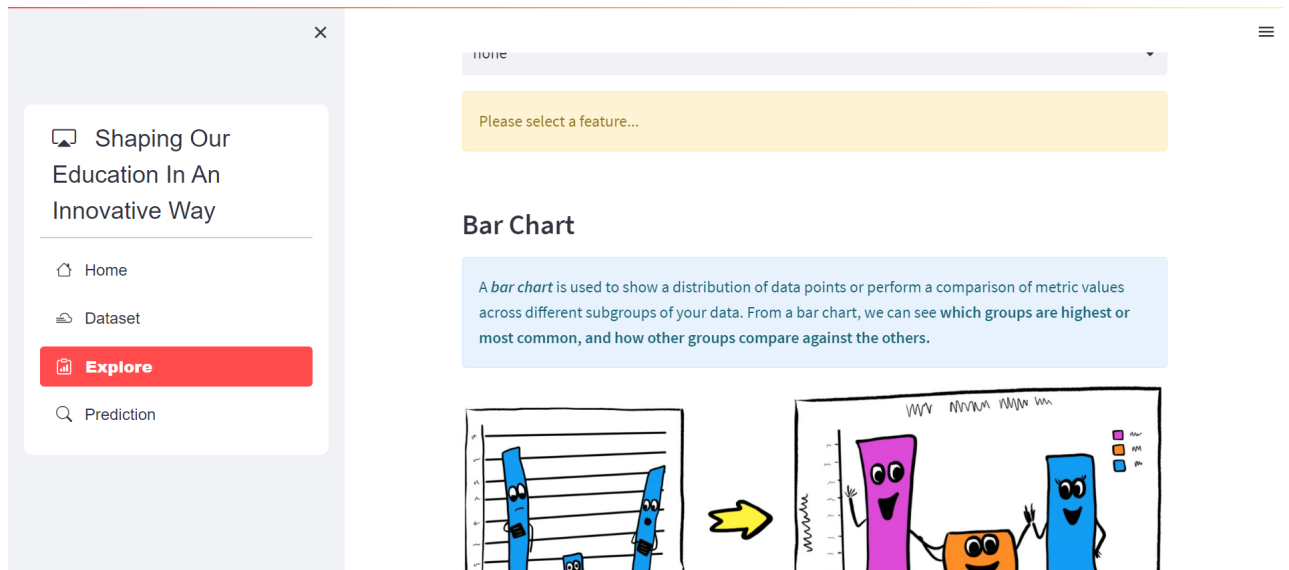
Drag and drop file here
Limit 200MB per file • CSV

Browse files

Process

Made with Streamlit

Hình 38: Dataset Page





×

Shaping Our Education In An Innovative Way

Home

Dataset

Explore

Prediction

Enter the second period score (assignment grade) of that student:

Assignment Score

0

1 Student Information You've just enter (after encoding):

	school	sex	age	address	family_size	parents_status	mother_education	father_education	m	
0	GP	F	18	U	LE3	T		0	0	te

☐ Display JSON Format

2 Student Final Score Estimator

Let's see how this student performs in final exam test!

Estimate Final Score

Hình 40: Prediction Page

×

Shaping Our Education In An Innovative Way


Home

Dataset

Explore

Prediction

Model Comparison





Compare Between Used Models

i In the Prediction Page, we offer a bunch of regressors and classifiers for you to choose the models you wanna use to make predictions about student performance based on your input information. *However*, once in a while the score estimated by different models results in relatively high difference. Why is that??

Basically, every model fits well with every dataset. Besides, the properties and complexity of each model also determine the accuracy when using the model to predict data of the dataset.

So, let's see which accuracy level each used model can reaches in making predictions with a random splitting way to divide our dataset into training set and test set!





Hình 41: *Model Comparison Page*



4.2 Sản phẩm

- **Source Github:** https://github.com/KhoaNgex/education_ML_web
- **Heroku App:** <https://elearning-analysis-system.herokuapp.com/>

5 Định hướng mở rộng

Từ việc đánh giá kết quả học tập, nhóm dự định mở rộng kết quả nghiên cứu theo hướng phát triển nên bộ công cụ đánh giá và dự đoán hiệu quả học tập đi kèm trên các hệ thống LMS. Bộ công cụ này có chức năng thu thập lịch sử hoạt động của người học thực tế trên hệ thống e-learning của trường đại học Bách Khoa nhằm tạo ra bộ dữ liệu mẫu để thực hiện việc khám phá và dự đoán bằng các mô hình mà dự án đề xuất.

Tuy nhiên cần lưu ý rằng tập dữ liệu nhóm sử dụng chỉ đủ lớn để phù hợp cho việc nghiên cứu, trong tương lai nhóm mong muốn được thu thập được lượng dữ liệu lớn và có độ chính xác cao, đồng thời áp dụng thêm các phương pháp regularization để hạn chế overfitting hay các phương pháp optimization và parameter tuning khác nhằm tối ưu hoá các mô hình đã được sử dụng.

Ngoài ra, trong tương lai nếu nhóm thu thập được các tập dữ liệu có kích thước vô cùng lớn, có thể xem xét áp dụng mô hình Học sâu (Deep learning) để tăng độ chính xác cho bộ công cụ.

6 Tài liệu tham khảo

1. DeepAI KhanhBlog, truy cập tại: https://phamdinhkhanh.github.io/deepai-book/ch_ml/RidgedRegression.html