

ĐẠI HỌC HUẾ

**KHOA KỸ THUẬT VÀ CÔNG NGHỆ**

🙠🙟🕮🙝🙢

****

**BÁO CÁO ĐỒ ÁN**

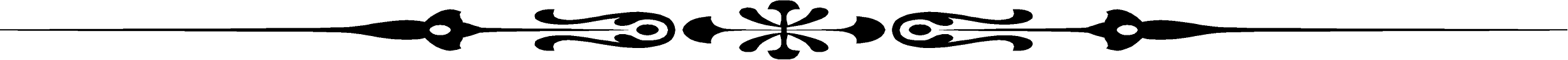
**Học kỳ II, năm học 2022 - 2023**

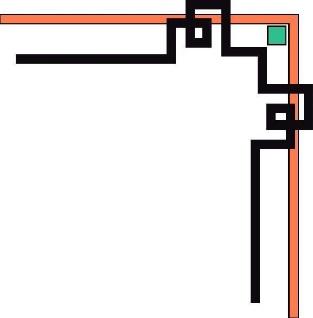
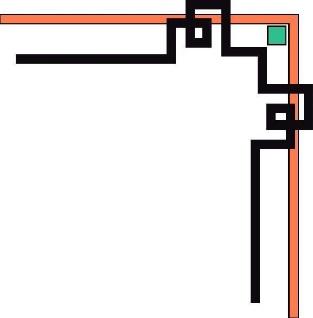
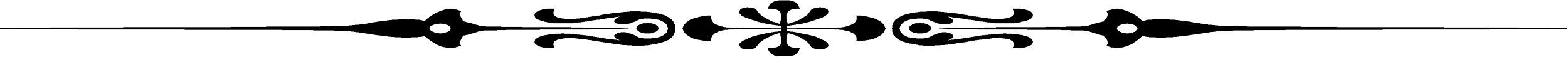
**Học phần: Các thuật toán tối ưu cho phân tích dữ liệu**

**Đề tài: SVM & Ứng dụng SVM dự đoán giao dịch gian lận**

**Sinh viên thực hiện:** .......**Trần Văn Tuấn Phong**...................

|  |
| --- |
| **Số phách**  *(Do hội đồng chấm thi ghi)* |

**Thừa Thiên Huế, tháng 05 năm 2023**



ĐẠI HỌC HUẾ

**KHOA KỸ THUẬT VÀ CÔNG NGHỆ**

🙠🙟🕮🙝🙢

****

**(Bìa phụ 2)**

**BÁO CÁO ĐỒ ÁN**

**Học kỳ II, năm học 2022 - 2023**

**Học phần:** Các thuật toán tối ưu cho phân tích dữ liệu

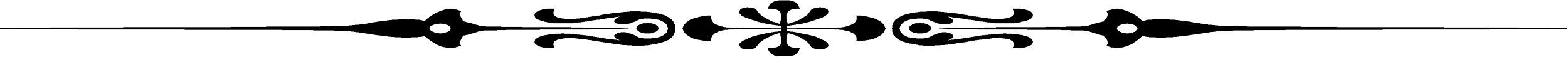
**Giảng viên hướng dẫn: Hoàng Trọng Lợi**

**Lớp: Khoa học dữ liệu & Trí tuệ nhân tạo**

**Sinh viên thực hiện: Trần Văn Tuấn Phong**

*(ký tên và ghi rõ họ tên)*

|  |
| --- |
| **Số phách**  *(Do hội đồng chấm thi ghi)* |

**Thừa Thiên Huế, tháng 05 năm 2023**

**MỤC LỤC**

[A. TỔNG QUAN LÝ THUYẾT VỀ SVM 1](#_Toc135389893)

[I. Giới thiệu về SVM: 1](#_Toc135389894)

[II. Thuật toán SVM và cách hoạt động: 1](#_Toc135389895)

[1. Hàm mất mát của SVM: 1](#_Toc135389896)

[2. Đường biên và lề trong SVM: 2](#_Toc135389897)

[3. Bài toán tối ưu SVM: 5](#_Toc135389898)

[a. Hệ điều kiện KKT: 5](#_Toc135389899)

[b. Dự báo nhãn: 6](#_Toc135389900)

[4. Kernel trong SVM: 6](#_Toc135389901)

[III. Thuật toán SVM trong thư viện Scikit-learn: 7](#_Toc135389902)

[B. ỨNG DỤNG SVM DỰ ĐOÁN GIAN LẬN THẺ TÍN DỤNG 8](#_Toc135389903)

[I. Mô tả bài toán: 8](#_Toc135389904)

[II. Ứng dụng SVM dự đoán phân loại thẻ tín dụng 8](#_Toc135389905)

[1. Cài đặt thư viện: 8](#_Toc135389906)

[2. Đọc dataset creditcard.csv: 8](#_Toc135389907)

[3. Mô tả các chỉ số thông kê của dataset: 9](#_Toc135389908)

[4. Hiển thị số tiền gian lận theo thời gian: 9](#_Toc135389909)

[5. Mối tương quan giữa các features: 10](#_Toc135389910)

[6. Kỹ thuật Undersampling: 10](#_Toc135389911)

[7. Tạo hàm vẽ Confusion Matrix: 11](#_Toc135389912)

[Tài liệu tham khảo 15](#_Toc135389913)

**DANH MỤC TỪ VIẾT TẮT**

|  |  |
| --- | --- |
| **SVM** | **Support Vector Machine** |
| **CNN** | **Convolutional Neural Network** |
| **KKT** | **Karush Kuhn Tucker** |
| **RBF** | **Radial Basis Function** |
| **TN** | **True Negative** |
| **TP** | **True Positive** |
| **FN** | **False Negative** |
| **FP** | **False Positive** |

**DANH MỤC HÌNH ẢNH**

[*Hình 1: Hình dạng của hàm cost1() và cost2() 7*](#_Toc135389410)

[*Hình 2: Hình ảnh về tập dữ liệu trong bài toán phân loại nhị phân mà các lớp là phân tuyến 8*](#_Toc135389411)

[*Hình 3: Hình ảnh các trường hợp của đường biên phân chia với các phương khác nhau đối với một tập dữ liệu phân tuyến 9*](#_Toc135389412)

[*Hình 4: Biểu đồ thể hiện số tiền gian lận theo thời gian 14*](#_Toc135389413)

[*Hình 5: Biểu đồ thể hiện mối tương quan giữa các thành phần trong dataset 15*](#_Toc135389414)

[*Hình 6: Biểu đồ phân loại các giao dịch của tập test 17*](#_Toc135389415)

[*Hình 7: Biểu đồ phân loại các giao dịch sau khi tính toán lại phương thức phù hợp. 18*](#_Toc135389416)

1. TỔNG QUAN LÝ THUYẾT VỀ SVM
2. Giới thiệu về SVM:

SVM là viết tắt của cụm từ support vector machine. Đây là một thuật toán khá hiệu quả trong lớp các bài toán phân loại nhị phân và dự báo của học có giám sát. Thuật toán này có ưu điểm là hoạt động tốt đối với những mẫu dữ liệu có kích thước lớn và thường mang lại kết quả vượt trội so với lớp các thuật toán khác trong học có giám sát.

Ưu điểm của SVM đó là:

* Đây là thuật toán hoạt động hiệu quả với không gian cao chiều
* Thuật toán tiêu tốn ít bộ nhớ vì chỉ sử dụng các điểm trong tập hỗ trợ để dự báo trong hàm quyết định*.*
* Chúng ta có thể tạo ra nhiều hàm quyết định từ những hàm kernel khác nhau. Thậm chí sử dụng đúng kernel có thể giúp cải thiện thuật toán lên đáng kể.

Chính vì tính hiệu quả mà SVM thường được áp dụng nhiều trong các tác vụ phân loại và dự báo, cũng như được nhiều công ty ứng dụng và triển khai trên môi trường production. Chúng ta có thể liệt kê một số ứng dụng của thuật toán SVM đó là:

* Mô hình chuẩn đoán bệnh. Dựa vào biến mục tiêu là những chỉ số xét nghiệm lâm sàng, thuật toán đưa ra dự báo về một số bệnh như tiểu đường, suy thận, máu nhiễm mỡ,…
* Trước khi thuật toán CNN và Deep Learning bùng nổ thì SVM là lớp mô hình cực kì phổ biến trong phân loại ảnh.
* Mô hình phân loại tin tức. Xác định chủ đề của một đoạn văn bản, phân loại cảm xúc văn bản, phân loại thư rác.
* Mô hình phát hiện gian lận.

1. Thuật toán SVM và cách hoạt động:
2. Hàm mất mát của SVM:

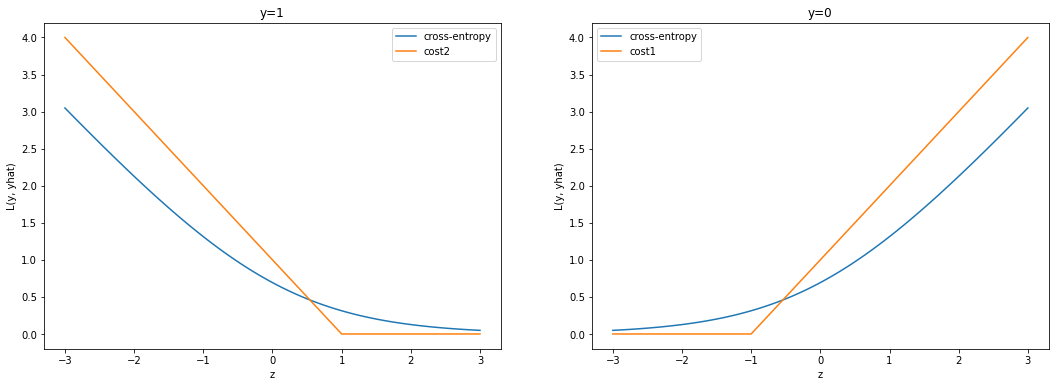
Trong hồi qui Logistic chúng ta đã làm quen với hàm mất mát*(*loss function*)* dạng:

Trong đó phân phối xác suất được ước tính dựa trên hàm Sigmoid theo công thức

Đường biên phân loại của hồi qui Logistic là một siêu phẳng có phương trình **wTx:**

Trong SVM, ta tìm cách xấp xỉ hàm mất mát dạng cross-entropy của Logistic bằng một hàm mà chỉ phạt những điểm ở gần đường biên thay vì phạt những điểm ở xa đường biên bằng cách đưa mức phạt về 0.

Cụ thể đó là hai hàm phạt cost1() và cost2() tương ứng với y=0 và y=1 như bên dưới:



Hình 1: Hình dạng của hàm cost1() và cost2()

Hai hàm này thể hiện chi phí phải bỏ ra nếu phân loại sai các nhãn lần lượt thuộc 0 hoặc 1. Dạng tổng quát của chúng là max(0,t) còn được gọi là hàm **hingloss**. Đây là một trong những hàm mất mát gặp khá nhiều trong machine learning.

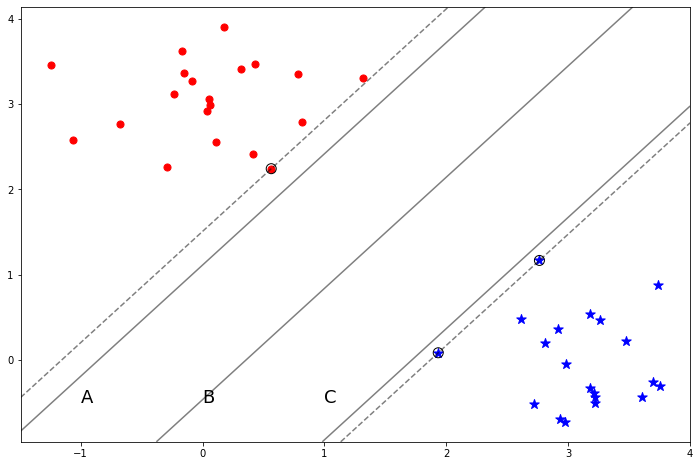
Theo các hàm mất mát mới này, chúng ta bỏ qua việc phạt phân loại sai những điểm nằm xa đường biên. Đối với những điểm nằm gần đường biên nhất thì mới ảnh hưởng tới hàm mất mát. Tập hợp những điểm nằm gần đường biên sẽ giúp xác định đường biên và được gọi là tập tập hỗ trợ (support vector).

Như vậy sau khi thay đổi hàm phạt ta thu được hàm mất mát mới dạng:

1. Đường biên và lề trong SVM:

Giả sử tập dữ liệu huấn luyện Z bao gồm N điểm dữ liệu. Trong đó điểm dữ liệu thứ i là Zi = (xi,yi) với xi ∈ Rd là vecto đầu vào và yi là biến mục tiêu thuộc một trong hai giá trị {-1, 1}. Tập dữ liệu này được giả định là phân tuyến.

Bên dưới là hình ảnh tập dữ liệu phân tuyến, đường biên và lề trong thuật toán SVM



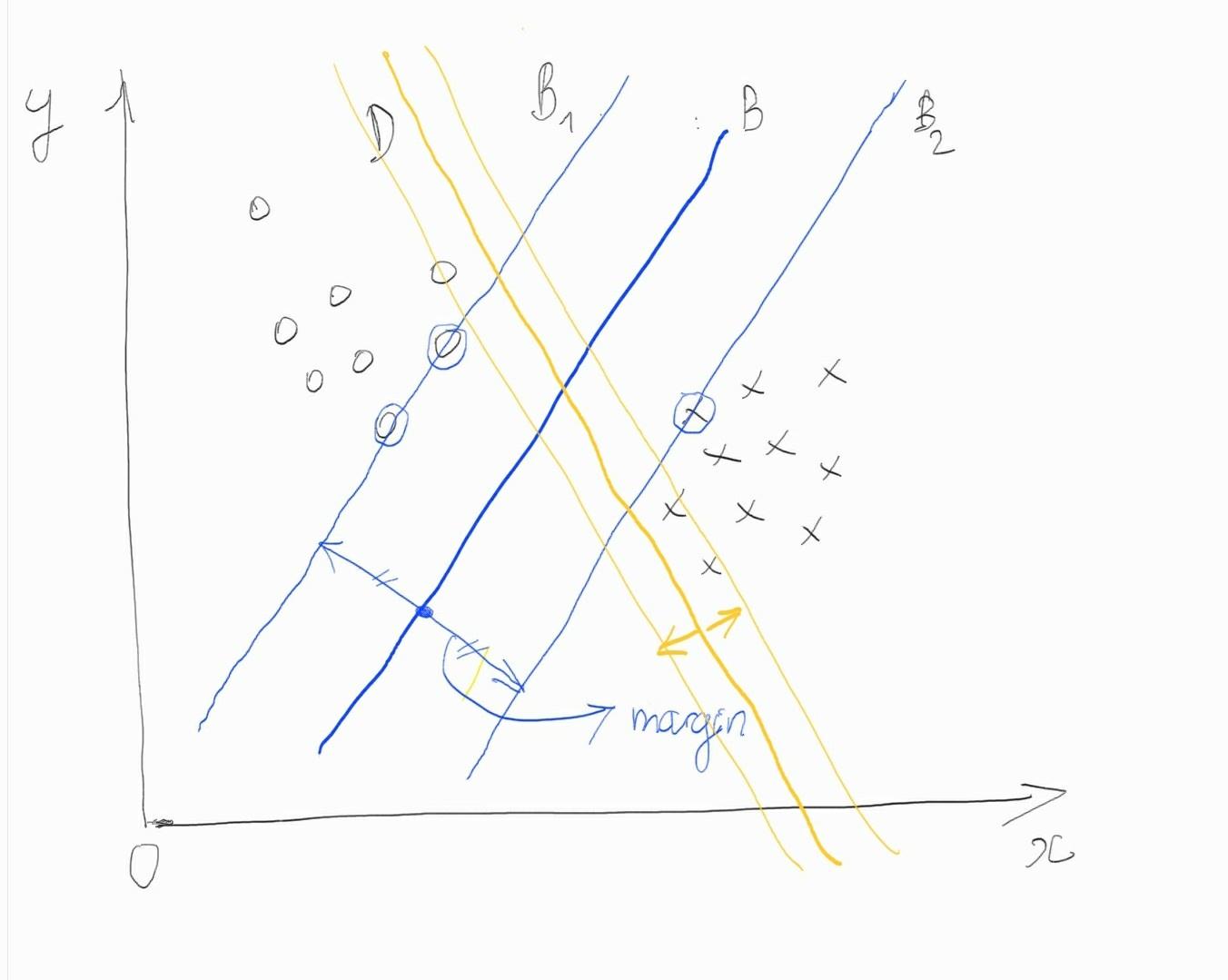
Hình 2: Hình ảnh về tập dữ liệu trong bài toán phân loại nhị phân mà các lớp là phân tuyến

Ba đường thẳng A, B, C ở ví dụ trên là ba đường biên phân chia song song và có cùng phương. Trong ba đường biên phân chia thì đường biên B là công bằng nhất vì chúng cách đều các điểm gần nhất thuộc hai lớp. Còn lựa chọn A và C sẽ không công bằng vì chúng ta sẽ dễ thiên vị một lớp hơn lớp còn lại.

Như vậy để cho công bằng thì đường biên phải luôn **nằm chính giữa** và **cách đều** các điểm gần nhất với nó. Đồng thời đối với bài toán Hard-Margin SVM thì tập dữ liệu là phân tuyến nên đường biên cần phải phân loại đúng mọi điểm dữ liệu. Ta coi độ rộng của đường biên là lề (margin). Ngoài ra tập hợp những điểm nằm sát đường biên nhất thì được gọi là tập hỗ trợ. Những điểm này sẽ hỗ trợ tìm ra đường biên vì những đường thẳng nét đứt đi qua chúng song song với đường biên.

Trong không gian hai chiều thì đường biên là một đường thẳng. Trong không gian 3 chiều chúng sẽ là một mặt phẳng (plane). Trong không gian nhiều hơn 3 chiều chúng ta gọi đường biên phân chia là siêu phẳng (hyperplane).

Mục tiêu của SVM đó là tìm ra một siêu phẳng (hyperplane) trong không gian d chiều làm đường biên phân chia sao cho độ rộng **lề** của chúng là lớn nhất vì khi phân chia theo đường biên này thì các nhóm là tách biệt nhất.



Hình 3: Hình ảnh các trường hợp của đường biên phân chia với các phương khác nhau đối với một tập dữ liệu phân tuyến

⁕ Khoảng cách từ một điểm đến siêu phẳng

Giả sử phương trình của đường biên phân chia hai điểm dữ liệu là:

b là hệ số tự do, w là các vecto hệ số, x là vecto quan sát đầu vào.

Khoảng cách từ một điểm bất kì Zi(xi, yi) tới đường biên siêu phẳng (**H**) là:

⁕ Tìm đường biên có lề lớn nhất

Tập hợp các điểm nằm gần nhất với một đường biên sẽ giúp xác định phương trình đường biên nên chúng còn được gọi là tập hợp các điểm hỗ trợ (support points), ký hiệu là ***S***. Trong hình 3 thì các điểm được khoanh tròn chính là các điểm thuộc tập hỗ trợ. Để tìm ra đường biên có độ rộng lề là lớn nhất thì chúng ta cần tối đa hoá khoảng cách từ các điểm thuộc tập hỗ trợ tới đường biên. Điều này tương đương với giải bài toán tối ưu:

Khi nhân vào phương trình đường biên với một hệ số k thì đường biên không thay đổi. Do đó khoảng cách từ mọi điểm tới đường biên không thay đổi. Tức là khoảng cách từ các điểm thuộc tập hỗ trợ tới đường biên không thay đổi và dẫn tới độ rộng của lề là không thay đổi. Điều đó cũng đồng nghĩa với luôn tìm được một cách nhân với k sao cho đường biên:

Như vậy bài toán tối ưu trở thành:

Subject:

Để đơn giản hóa bài toán thì ta có thể nghịch đảo hàm mục tiêu để chuyển thành:

Subject:

1. Bài toán tối ưu SVM:
2. Hệ điều kiện KKT:

Giả sử chúng ta đang cần giải quyết bài toán tối ưu:

Thoả mãn hệ ĐK ràng buộc:

Bài toán tối ưu có hàm mục tiêu và hệ điều kiện ràng buộc còn được gọi là bài toán gốc (primal problem). Để giải trực tiếp bài toán gốc là tương đối khó nên chúng ta sẽ chuyển sang giải bài toán tối ưu trên hàm đối ngẫu Lagrange (Lagrange Dual Function).

Trong đó  >0 ∀i và  >0 ∀j,những hệ số này còn được gọi là những nhân tử Lagrange (Lagrange Multipliers).

Bằng cách thiết lập phương trình đạo hàm bậc nhất của hàm Lagrange kết hợp với hệ điều kiện ràng buộc tuyến tính và một số điều kiện khác ta có thể thu được nghiệm của bài toán tối ưu. Hệ điều kiện này được gọi là Karush-Kuhn-Tucker conditions (viết tắt là KKT conditions). Trong tối ưu có rất nhiều các bài toán được giải quyết thông qua hệ điều kiện KKT. Đây là một trong những tiêu chuẩn tìm nghiệm khá hiệu quả và thường được sử dụng trong những bài toán tối ưu có điều kiện ràng buộc. Hệ điều kiện KKT bao gồm các nhóm điều kiện:

* Stationary:
* Complenatery slackness:
* Primal Feasibility:
* Dual Feasibility:

1. Dự báo nhãn:

Nhãn của một quan sát trong mô hình SVM sẽ phụ thuộc vào dấu của đường biên:

Trong trường hợp thì điểm được dự báo nhãn 1 và trái lại là nhãn -1

1. Kernel trong SVM:

Trong SVM, kernel là một phần quan trọng của thuật toán, đóng vai trò chính trong việc xác định hàm biến đổi không gian (transformation function) để phân tách các lớp dữ liệu không tuyến tính thành các lớp tuyến tính.

Kernel trong SVM giúp chúng ta ánh xạ dữ liệu từ không gian ban đầu (input space) sang không gian cao chiều hơn (feature space) trong đó các lớp dữ liệu trở nên tách biệt tuyến tính. Thay vì tìm một siêu mặt phẳng tuyến tính trực tiếp trong không gian ban đầu, chúng ta có thể tìm một siêu mặt phẳng tuyến tính trong không gian cao chiều hơn để phân loại dữ liệu một cách hiệu quả.

Các loại kernel phổ biến được sử dụng trong SVM bao gồm:

* Linear Kernel: Đây là kernel tuyến tính đơn giản nhất. Nó được sử dụng khi dữ liệu có thể được phân loại tốt bằng một siêu mặt phẳng tuyến tính đơn giản trong không gian ban đầu.
* Polynomial Kernel: Kernel đa thức sử dụng hàm đa thức để ánh xạ dữ liệu vào không gian cao chiều hơn. Bậc của hàm đa thức được xác định bằng tham số degree.
* Gaussian RBF Kernel: Kernel RBF (Radial Basis Function) sử dụng hàm Gaussian để ánh xạ dữ liệu. Nó tạo ra một không gian phi tuyến tính vô hạn chiều và tham số gamma ảnh hưởng đến hình dạng của siêu mặt phẳng phân tách.
* Sigmoid Kernel: Kernel sigmoid sử dụng hàm sigmoid để ánh xạ dữ liệu. Nó cũng tạo ra một không gian phi tuyến tính và tham số coef0 ảnh hưởng đến hình dạng của siêu mặt phẳng phân tách.

Mục đích chính của việc sử dụng kernel trong SVM là để xử lý các bài toán phân loại không tuyến tính, trong đó các lớp dữ liệu không thể được phân tách hoàn toàn bằng một siêu mặt phẳng tuyến tính đơn giản. Kernel cho phép chúng ta sử dụng một hàm biến đổi phi tuyến tính để tạo ra một siêu mặt phẳng tuyến tính phức tạp hơn trong không gian cao chiều, giúp mô hình SVM phân loại dữ liệu một cách chính xác hơn.

1. Thuật toán SVM trong thư viện Scikit-learn:

Trong thư viện scikit-learn của Python, khi sử dụng thuật toán SVM, các thuộc tính quan trọng của đối tượng mô hình SVM là như sau:

* kernel: Thuộc tính này xác định loại hàm kernel được sử dụng trong SVM. Một số giá trị phổ biến cho thuộc tính này là "linear" (hàm tuyến tính), "rbf" (hàm Gaussian Radial Basis Function), "poly" (hàm đa thức), và "sigmoid" (hàm sigmoid).
* C: Thuộc tính này xác định tham số ứng với độ ràng buộc (regularization parameter) trong SVM. Giá trị của C kiểm soát sự cân bằng giữa việc tìm một siêu mặt phẳng tối ưu và việc tối thiểu hóa sự sai lệch (margin violations). Giá trị C càng lớn, mô hình SVM càng nhạy cảm với các điểm nằm ngoài siêu mặt phẳng và có thể dẫn đến overfitting.
* gamma: Thuộc tính này chỉ định tham số gamma trong hàm kernel (như RBF, poly, sigmoid). Giá trị gamma kiểm soát hình dạng của siêu mặt phẳng phân tách các điểm dữ liệu. Một giá trị gamma nhỏ tạo ra một siêu mặt phẳng mềm, trong khi một giá trị gamma lớn tạo ra một siêu mặt phẳng cứng hơn.
* degree: Thuộc tính này chỉ định bậc của hàm đa thức trong SVM với kernel đa thức. Nó xác định độ phức tạp của mô hình SVM đa thức. Giá trị degree mặc định là 3.
* coef0: Thuộc tính này chỉ định hệ số tự do trong các hàm kernel đa thức và sigmoid. Giá trị của coef0 ảnh hưởng đến hình dạng của siêu mặt phẳng và có thể được sử dụng để kiểm soát overfitting.

Đây là một số thuộc tính quan trọng khi sử dụng SVM trong Python. Các giá trị được chọn cho các thuộc tính này có thể ảnh hưởng đến hiệu suất và khả năng tổng quát hóa của mô hình SVM. Thông qua việc điều chỉnh các thuộc tính này, bạn có thể tùy chỉnh mô hình SVM để phù hợp với bài toán cụ thể và tối ưu hóa kết quả của mô hình.

1. ỨNG DỤNG SVM DỰ ĐOÁN GIAN LẬN THẺ TÍN DỤNG
2. Mô tả bài toán:

Bài toán sử dụng dataset Credit Card Fraud Detection (phát hiện gian lận thẻ tín dụng). Dataset này chứa thông tin về các giao dịch thẻ tín dụng và được sử dụng để phân loại các giao dịch là giao dịch bình thường hoặc giao dịch gian lận.

Dataset Credit Card Fraud Detection có các thuộc tính như số tiền giao dịch, địa điểm, thời gian giao dịch và các thông tin liên quan. Với SVM, ta có thể huấn luyện một mô hình để phân loại các giao dịch thành hai lớp: giao dịch bình thường và giao dịch gian lận.

Bằng cách sử dụng dataset này, SVM có thể học các mẫu và quy tắc phân loại từ các giao dịch đã được gán nhãn. Sau đó, mô hình SVM có thể được sử dụng để dự đoán xem một giao dịch mới có khả năng là giao dịch gian lận hay không.

Tuy nhiên, đối với các bài toán phát hiện gian lận, thường cần xử lý các tập dữ liệu không cân bằng (imbalanced data) với tỷ lệ giao dịch gian lận thấp. Trong trường hợp này, bạn có thể áp dụng các kỹ thuật như oversampling (tăng cường dữ liệu giao dịch gian lận) hoặc undersampling (giảm thiểu dữ liệu giao dịch bình thường) để cải thiện hiệu suất của mô hình SVM trong việc phát hiện gian lận.

1. Ứng dụng SVM dự đoán phân loại thẻ tín dụng
2. Cài đặt thư viện:

import pandas as pd

import numpy as np

from sklearn import preprocessing

from sklearn.metrics import confusion\_matrix

from sklearn import svm

import itertools

import matplotlib.pyplot as plt

import matplotlib.mlab as mlab

import seaborn

1. Đọc dataset creditcard.csv:

from google.colab import drive

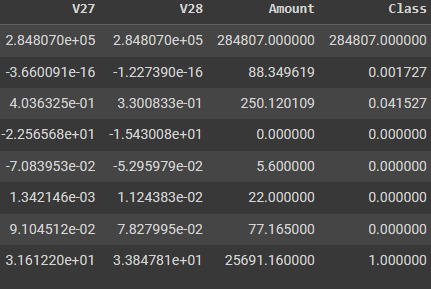
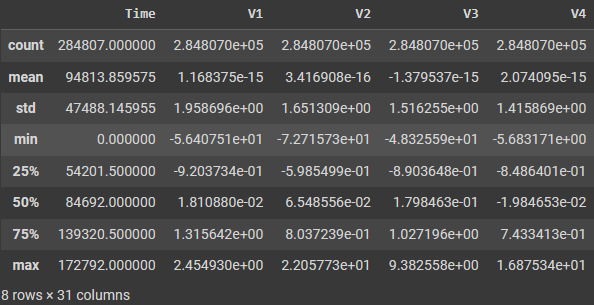
drive.mount('/content/drive')

data = pd.read\_csv('/content/drive/MyDrive/CreditcardSVM/creditcard.csv') # Đọc file csv

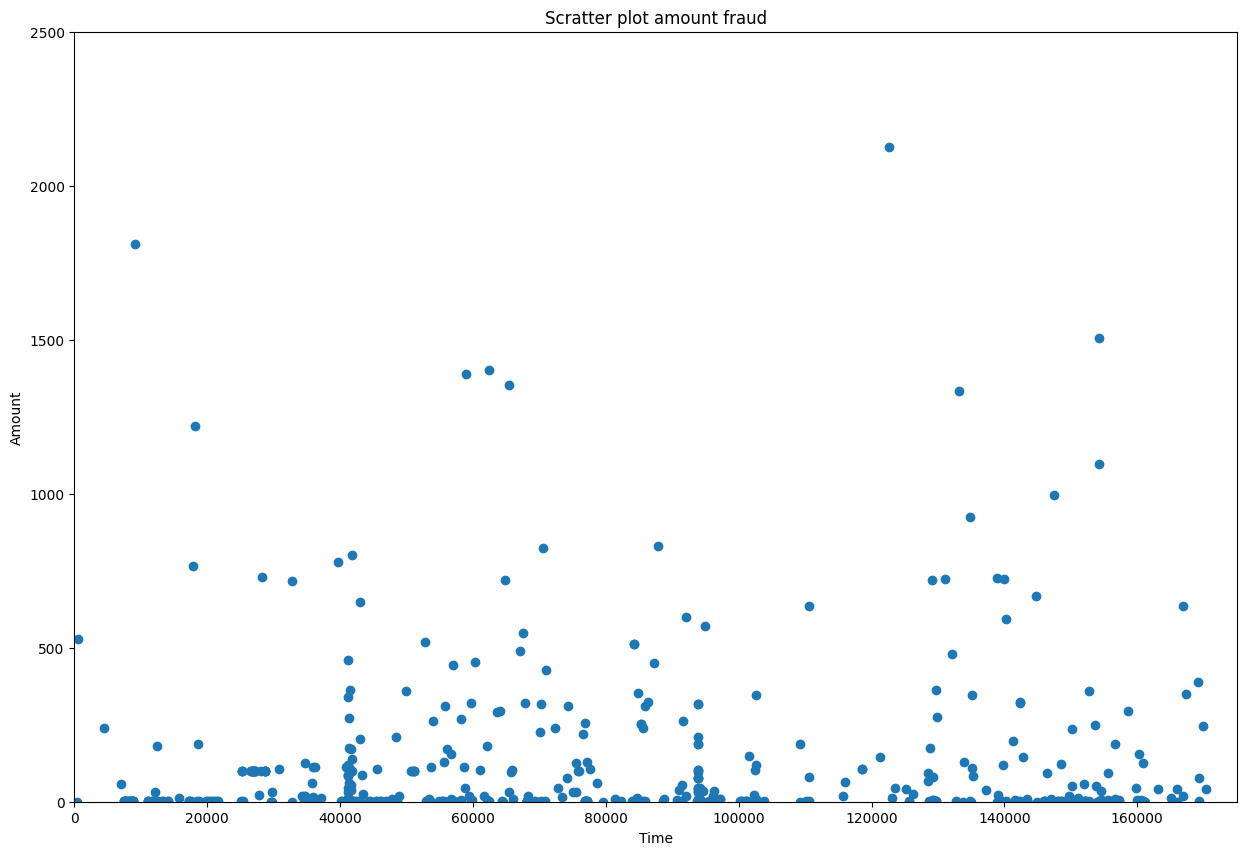
df = pd.DataFrame(data)

1. Mô tả các chỉ số thông kê của dataset:

df.describe()



1. Hiển thị số tiền gian lận theo thời gian:



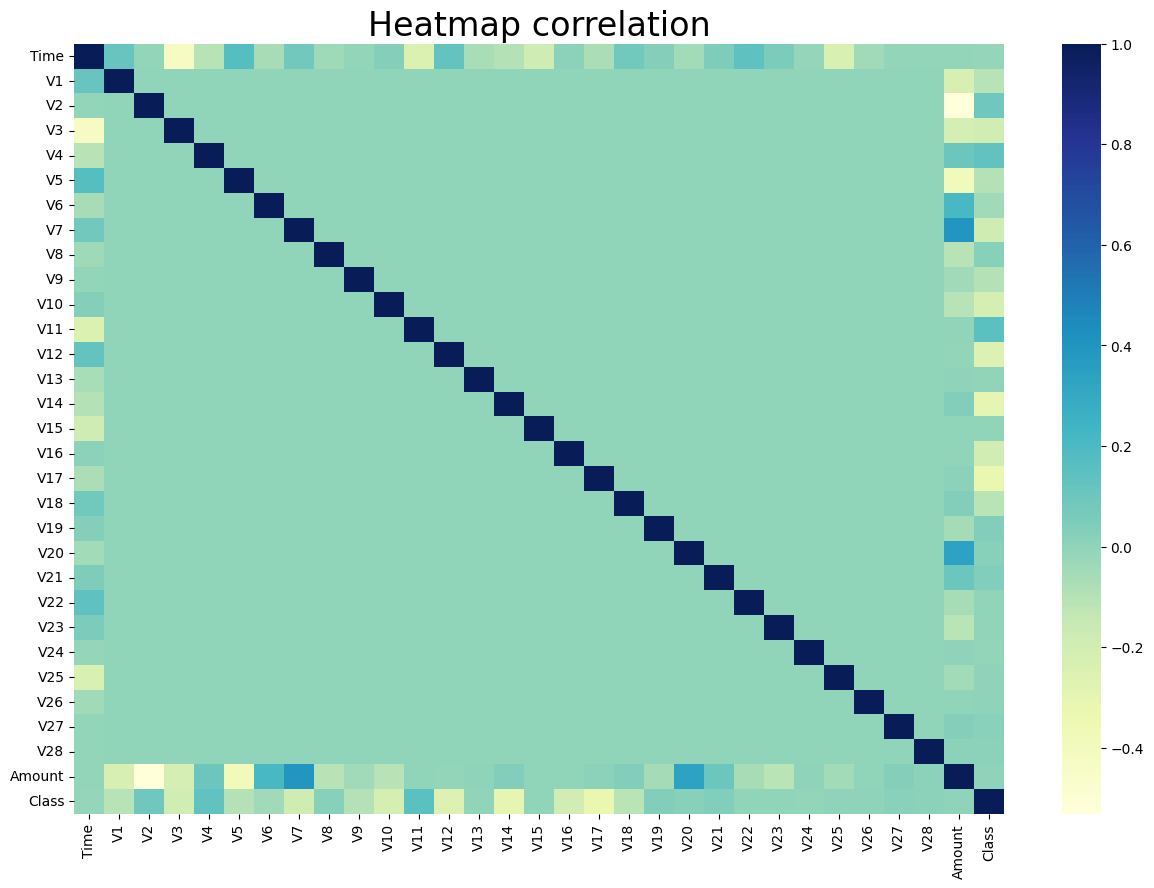
Hình 4: Biểu đồ thể hiện số tiền gian lận theo thời gian

Nhận xét:

* Thời gian không ảnh hưởng tới tần suất gian lận.
* Phần lớn các vụ lừa đảo là số tiền nhỏ.
* Chỉ có 9 trên 492 giao dịch lừa đảo là có số tiền lừa đảo lớn hơn 1000$.
* Ta có thể thấy dataset này không cân bằng (chỉ 492 giao dịch gian lận so với 284315 giao dịch bình thường) nên rất có thể bài toán sẽ gán nhãn cho mọi giao dịch là không gian lận.

⇒ Để giải quyết vấn đề này, ta có thể sử dụng nguyên tắc underssampling hoặc oversampling. Chỉ nên sử dụng undersampling nếu chúng ta có thể chắc rằng một vài bộ giao dịch bình thường được chọn đại diện cho toàn bộ giao dịch bình thường của dataset.

1. Mối tương quan giữa các features:



Hình 5: Biểu đồ thể hiện mối tương quan giữa các thành phần trong dataset

Theo hình mô tả trên, ta có thể thấy hầu hết các tính năng không tương quan với nhau. Điều này chứng minh thực tế là một PCA đã được thực hiện trước đó trên tập dữ liệu.

Điều thường được thực hiện trên một tập dữ liệu lớn là giảm kích thước. Bằng cách chọn các kích thước quan trọng nhất, có khả năng giải thích hầu hết vấn đề, do đó tiết kiệm được một lượng thời gian đáng kể đồng thời ngăn ngừa độ chính xác giảm quá nhiều.

Tuy nhiên, trong trường hợp này do PCA đã được thực hiện trước đó, nếu việc giảm kích thước có hiệu quả thì PCA không được tính toán theo cách hiệu quả nhất. Một cách khác để nói là không nên tính toán giảm kích thước trên tập dữ liệu mà PCA đã được tính toán chính xác.

1. Kỹ thuật Undersampling:

Kỹ thuật Undersampling trong bài toán phân loại dữ liệu liên quan đến việc giảm số lượng mẫu của lớp đa số (majority class) để tạo ra một tập dữ liệu cân bằng hơn giữa lớp đa số và lớp thiểu số (minority class). Điều này giúp giảm hiện tượng mất cân bằng dữ liệu và cải thiện khả năng phân loại của mô hình.

Các bước thực hiện kỹ thuật Undersampling là như sau:

* Xác định lớp đa số và lớp thiểu số: Đầu tiên, xác định lớp đa số và lớp thiểu số trong tập dữ liệu.
* Xác định tỷ lệ undersampling: Xác định tỷ lệ undersampling, tức là tỷ lệ giữa số lượng mẫu của lớp thiểu số và lớp đa số sau khi áp dụng undersampling. Thông thường, tỷ lệ này được lựa chọn để đảm bảo cân bằng giữa hai lớp, nhưng cần lưu ý không quá mất mát thông tin quan trọng trong quá trình này.
* Áp dụng undersampling: Thực hiện undersampling bằng cách loại bỏ một số mẫu từ lớp đa số sao cho số lượng mẫu của lớp đa số sau undersampling bằng tỷ lệ đã xác định ở bước trước.
* Xử lý mẫu thiếu: Nếu sau khi undersampling, lớp thiểu số vẫn còn thiếu mẫu, bạn có thể xử lý vấn đề này bằng cách sử dụng các phương pháp khác như oversampling hoặc tổng hợp mẫu (synthetic sample generation).
* Huấn luyện mô hình: Sử dụng tập dữ liệu đã được undersampling để huấn luyện mô hình phân loại. Điều này giúp mô hình học được cân bằng giữa hai lớp và có khả năng phân loại tốt hơn đối với lớp thiểu số.

Kỹ thuật Undersampling giúp cải thiện khả năng phân loại của mô hình trong trường hợp dữ liệu mất cân bằng nghiêm trọng, khi lớp thiểu số bị bỏ qua hoặc bị đánh giá không chính xác do ảnh hưởng mạnh từ lớp đa số. Tuy nhiên, nó cũng có thể dẫn đến mất mát thông tin và có thể không phù hợp trong các trường hợp dữ liệu ít hoặc lớp thiểu số chứa những thông tin quan trọng đặc biệt.

1. Tạo hàm vẽ Confusion Matrix:

def plot\_confusion\_matrix(cm, classes,

                          title='Confusion matrix',

                          cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)

    plt.title(title)

    plt.colorbar()

    tick\_marks = np.arange(len(classes))

    plt.xticks(tick\_marks, classes, rotation=45)

    plt.yticks(tick\_marks, classes)

    fmt = 'd'

    thresh = cm.max() / 2.

    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

        plt.text(j, i, format(cm[i, j], fmt),

                 horizontalalignment="center",

                 color="white" if cm[i, j] > thresh else "black")

    plt.tight\_layout()

    plt.ylabel('True label')

    plt.xlabel('Predicted label')

Khởi tạo mô hình SVM với kernel tuyến tính

classifier = svm.SVC(kernel='linear')

Huấn luyện mô hình

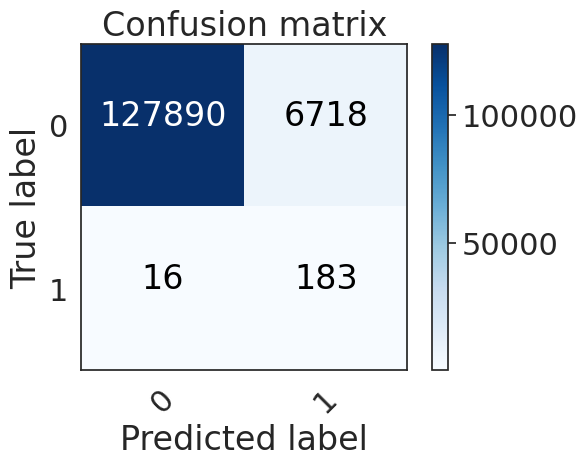
classifier.fit(X\_train, y\_train)

Thử nghiệm tập test

prediction\_SVM\_all = classifier.predict(X\_test\_all)

cm = confusion\_matrix(y\_test\_all, prediction\_SVM\_all)

plot\_confusion\_matrix(cm,class\_names)



Hình 6: Biểu đồ phân loại các giao dịch của tập test

Theo đồ thị, ta có:

* True Negative (TN) - Số lượng giao dịch bình thường (negative class) được phân loại chính xác là bình thường: cm[0][0] = 127890
* False Positive (FP): Số lượng giao dịch bình thường (negative class) bị phân loại sai thành giao dịch gian lận (positive class): cm[0][1] = 6718
* False Negative (FN): Số lượng giao dịch gian lận (positive class) bị phân loại sai thành giao dịch bình thường (negative class): cm[1][0] = 16
* True Positive (TP): Số lượng giao dịch gian lận (positive class) được phân loại chính xác là gian lận.: cm[1][1] = 183

Trong trường hợp này, ta sẽ cố gắng giảm thiểu số lỗi trong kết quả dự đoán. Lỗi nằm trên đường chéo của confusion matrix.

Tiêu chí ta đưa ra là độ chính xác của real fraud sẽ gấp 4 lần so với độ chính xác chung.

print('Our criterion give a result of '

      + str( ( (cm[0][0]+cm[1][1]) / (sum(cm[0]) + sum(cm[1])) + 4 \* cm[1][1]/(cm[1][0]+cm[1][1])) / 5))

Our criterion give a result of 0.9256878128355694

print('We have detected ' + str(cm[1][1]) + ' frauds / ' + str(cm[1][1]+cm[1][0]) + ' total frauds.')

print('\nSo, the probability to detect a fraud is ' + str(cm[1][1]/(cm[1][1]+cm[1][0])))

print("the accuracy is : "+str((cm[0][0]+cm[1][1]) / (sum(cm[0]) + sum(cm[1]))))

We have detected 183 frauds / 199 total frauds.

So, the probability to detect a fraud is 0.919597989949748 the accuracy is : 0.9500471043788528

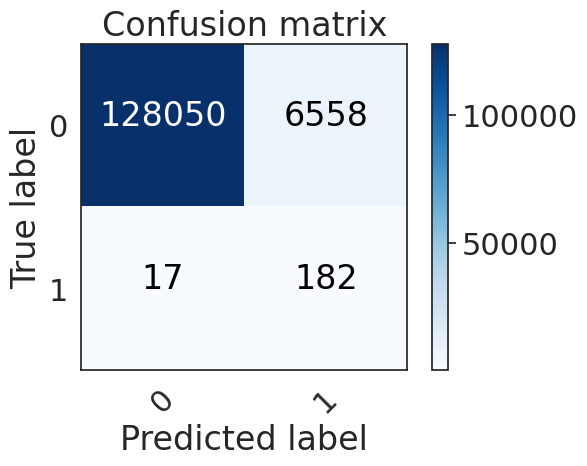
Tính toán lại phương thức phù hợp, vì ta đã áp dụng kỹ thuật undersampling nên kích thước của dữ liệu có sự thay đổi

classifier.fit(X\_train\_rank, y\_train)

prediction\_SVM = classifier.predict(X\_test\_all\_rank)

cm = confusion\_matrix(y\_test\_all, prediction\_SVM)

plot\_confusion\_matrix(cm,class\_names)



Hình 7: Biểu đồ phân loại các giao dịch sau khi tính toán lại phương thức phù hợp.

print('Our criterion give a result of '

      + str( ( (cm[0][0]+cm[1][1]) / (sum(cm[0]) + sum(cm[1])) + 4 \* cm[1][1]/(cm[1][0]+cm[1][1])) / 5))

Our criterion give a result of 0.9219036051279416

print('We have detected ' + str(cm[1][1]) + ' frauds / ' + str(cm[1][1]+cm[1][0]) + ' total frauds.')

print('\nSo, the probability to detect a fraud is ' + str(cm[1][1]/(cm[1][1]+cm[1][0])))

print("the accuracy is : "+str((cm[0][0]+cm[1][1]) / (sum(cm[0]) + sum(cm[1]))))

We have detected 182 frauds / 199 total frauds.

So, the probability to detect a fraud is 0.914572864321608 the accuracy is : 0.9512265683532755

**Tài liệu tham khảo**

Huynh Chi Trung. (n.d.). *viblo*. Retrieved from https://viblo.asia/p/gioi-thieu-ve-support-vector-machine-svm-6J3ZgPVElmB

Savan Patel. (n.d.). *https://medium.com*. Retrieved from https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72

Tiep Vu Huu. (n.d.). *machinelearningcoban*. Retrieved from https://machinelearningcoban.com/2017/04/09/smv/

*wikipedia.org*. (n.d.). Retrieved from https://en.wikipedia.org/wiki/Support\_vector\_machine

ĐẠI HỌC HUẾ

**KHOA KỸ THUẬT VÀ CÔNG NGHỆ**

🙠🙟🕮🙝🙢

**PHIẾU ĐÁNH GIÁ ĐỒ ÁN/TIỂU LUẬN/BÀI TẬP LỚN**

**Học kỳ II, năm học 2022 - 2023**

|  |  |
| --- | --- |
| **Cán bộ chấm thi 1** | **Cán bộ chấm thi 2** |
| **Nhận xét:**                    **Điểm đánh giá của CBCT1:**  Bằng số:  Bằng chữ: | **Nhận xét:**                    **Điểm đánh giá của CBCT2:**  Bằng số:  Bằng chữ: |

Điểm kết luận:

Bằng số:

Bằng chữ:

*Thừa Thiên Huế, ngày tháng năm 2023*

|  |  |
| --- | --- |
| **Cán bộ chấm thi 1**  *(Ký và ghi rõ họ và tên)* | **Cán bộ chấm thi 2**  *(Ký và ghi rõ họ và tên)* |