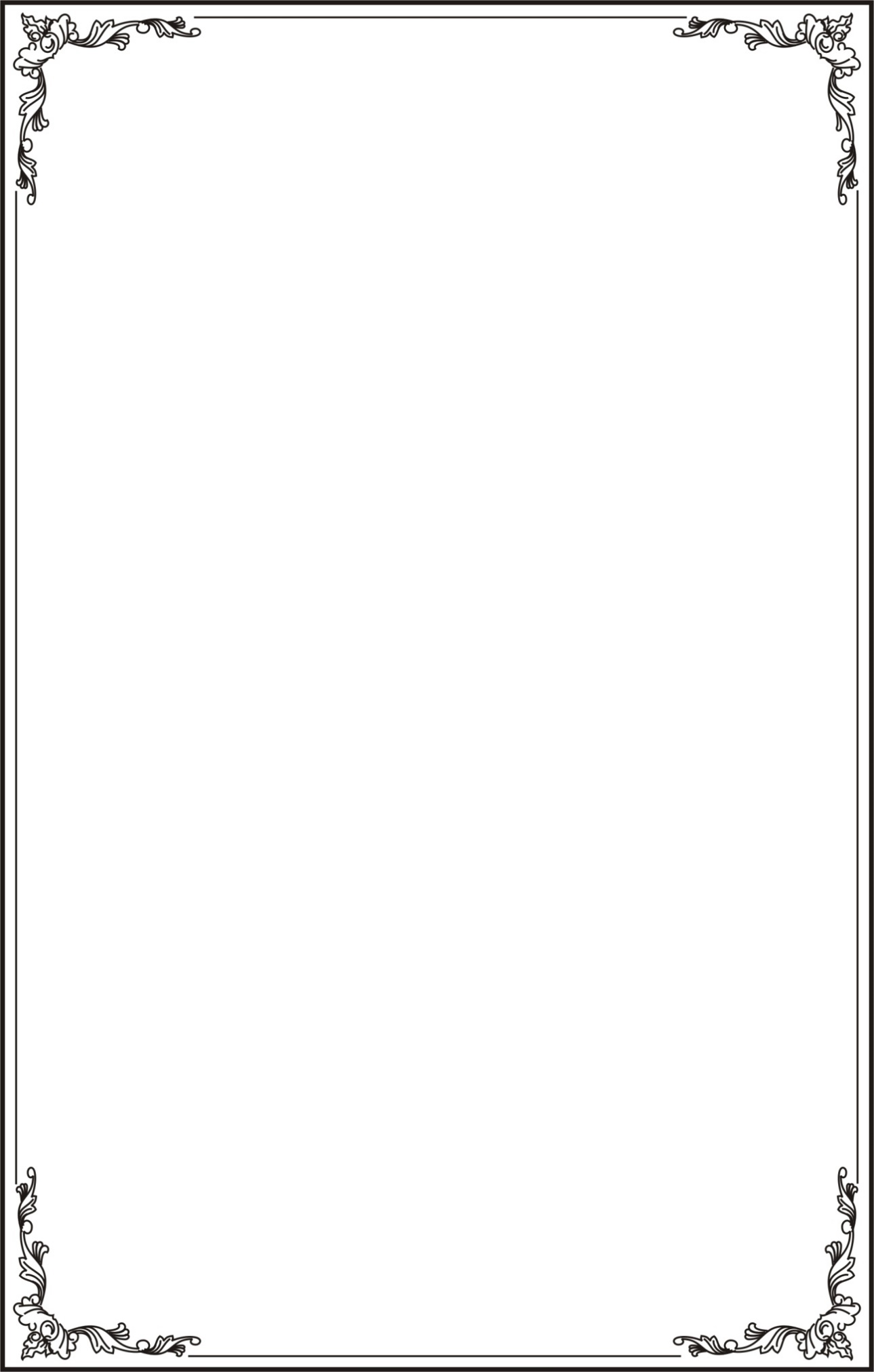
**Logo, icon

Description automatically generated**

**Mã lớp học phần : MALE431984\_22\_1\_04**

**GVHD: Ths.Trần Tiến Đức**

**Sinh viên thực hiện:**

**20110618 Trần Ngô Bích Du**

**20110223 Đào Thị Thanh Vi**

**MÔN HỌC: MACHINE LEARNING**

**Báo cáo ĐỒ ÁN CUỐI KỲ**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

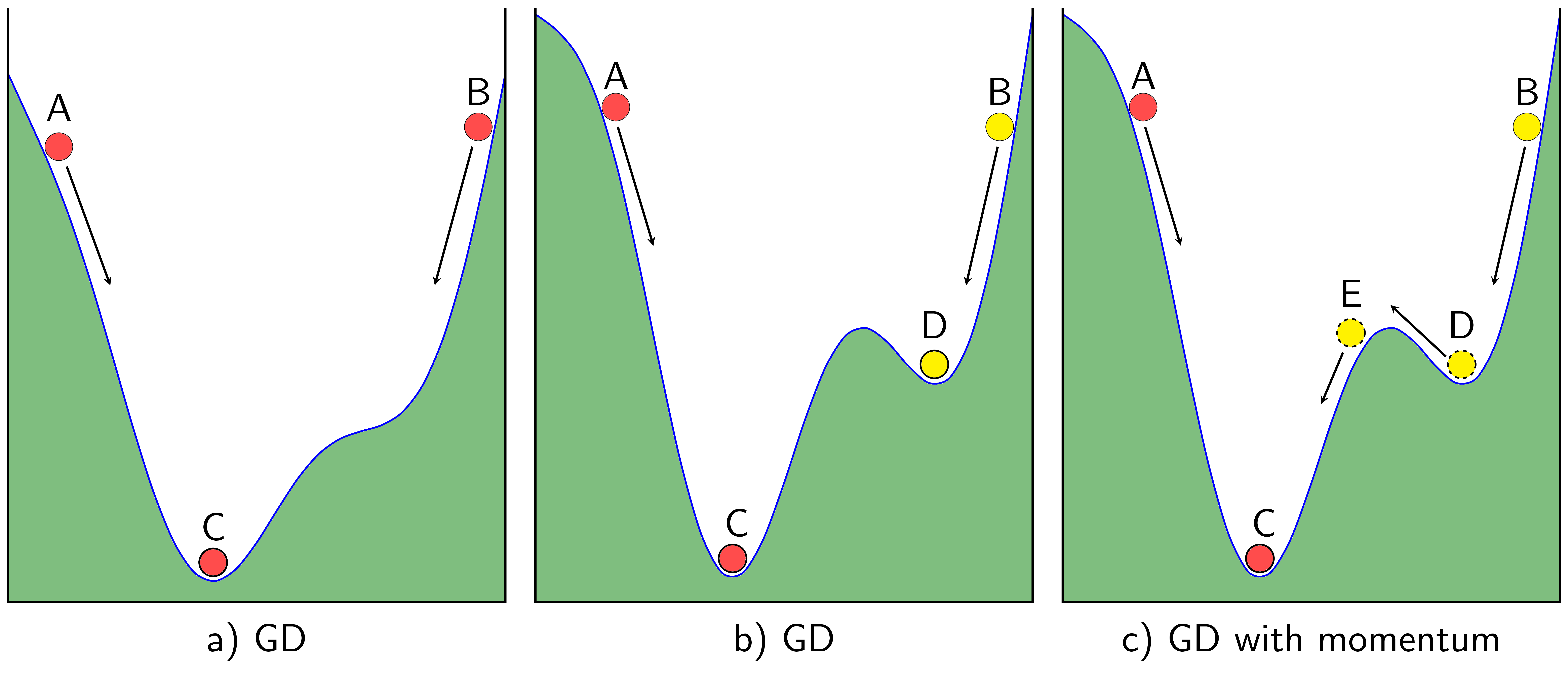
*Thủ Đức, tháng 12 năm 2022*

# Cơ sở lý thuyết

## Hồi quy – Regression

* 1. Hồi quy là gì?
* Đó là việc mô hình hóa và xây dựng các quy tắc tổng quát để mô tả thế giới. Về mặt bản chất, các công thức Newton hay các phương trình nhiệt động,...và tất cả các phương trình được xây dựng trong khoa học đều là bài toán hồi quy. Nghĩa là từ một tập các dữ liệu quan sát được, các nhà khoa học sẽ tìm cách để xây dựng các công thức biểu diễn để mô tả và tổng quát hóa tự nhiên, phân biệt dữ liệu thành các giá trị thực liên tục thay vì sử dụng các lớp hoặc các giá trị rời rạc.  
  Ví dụ: Xây dựng hàm dự đoán giá nhà, giá cổ phiếu theo thời gian hoặc các biến đầu vào khác.
* Đường thẳng phù hợp nhất với số liệu phải được lựa chọn sao cho giá trị của tổng bình phương các độ lệch (khoảng cách) theo phương thẳng đứng giữa các điểm và đường thẳng là nhỏ nhất.
* Ưu điểm:   
  Nhanh chóng để mô hình hóa và đặc biệt hữu ích khi mối quan hệ được mô hình hóa không quá phức tạp và nếu bạn không có nhiều dữ liệu.   
  Hồi quy tuyến tính là đơn giản để hiểu, nó rất có giá trị cho các quyết định kinh doanh.
* Nhược điểm:  
  Hạn chế đầu tiên của linear regression là nó rất nhạy cảm với nhiễu (sensitive to noise). Vì vậy, trước khi thực hiện linear regression, các nhiễu cần phải được loại bỏ. Bước này được gọi là tiền xử lý (pre-processing). Hoặc hàm mất mát có thể thay đổi một chút để tránh việc tối ưu các nhiễu bằng cách sử dụng Huber loss.  
  Hạn chế thứ hai của linear regression là nó không biễu diễn được các mô hình phức tạp.
  1. Một vài kỹ thuật hồi quy
     1. Linear Regression (Hồi quy tuyến tính)
* Hồi quy tuyến tính" là một phương pháp thống kê để hồi quy dữ liệu với biến phụ thuộc có giá trị liên tục trong khi các biến độc lập có thể có một trong hai giá trị liên tục hoặc là giá trị phân loại. Nói cách khác "Hồi quy tuyến tính" là một phương pháp để dự đoán biến phụ thuộc (Y) dựa trên giá trị của biến độc lập (X). Nó có thể được sử dụng cho các trường hợp chúng ta muốn dự đoán một số lượng liên tục.
  + 1. Random Forest Regression (Rừng ngẫu nhiên)
* Random forest là thuật toán học có giám sát, có thể giải quyết cả bài toán Regression (Hồi quy) và Classification (Phân loại).
* Random Forest sẽ xây dựng nhiều cây quyết định bằng thuật toán Decision Tree, tuy nhiên mỗi cây quyết định sẽ khác nhau (có yếu tố random). Sau đó kết quả dự đoán được tổng hợp từ các cây quyết định.
* Cây quyết định (Decision Tree) là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật. Các thuộc tính của đối tượngncó thể thuộc các kiểu dữ liệu khác nhau như Nhị phân (Binary), Định danh (Nominal), Thứ tự (Ordinal), Số lượng (Quantitative) trong khi đó thuộc tính phân lớp phải có kiểu dữ liệu là Binary hoặc Ordinal.

## Gradient Descient - Giảm dần đạo hàm

* Gradient Descient là một thuật toán tối ưu hóa tuân theo độ dốc âm của hàm mục tiêu để xác định vị trí cực tiểu của hàm.
* Trong Machine Learning nói riêng và Toán Tối Ưu nói chung, Gradient Descient thường được dùng để tìm giá trị cực tiểu của một hàm số nào đó, ví dụ như các hàm mất mát trong hai bài Linear Regression và K-means Clustering. Nhìn chung, việc tìm global minimum của các hàm mất mát trong Machine Learning là rất phức tạp, thậm chí là bất khả thi. Thay vào đó, người ta thường cố gắng tìm các điểm local minimum, và ở một mức độ nào đó, coi đó là nghiệm cần tìm của bài toán Gradient.
* Momentum – Động lượng:
  + Mometum là một thuật toán mở rộng được ra đời nhằm khắc phục việc nghiệm của GD rơi vào một điểm local minimum không mong muốn.
  + Ví dụ sự khác nhau:  
      
    Thuật toán GD thường được ví với tác dụng của trọng lực lên một hòn bi đặt trên một mặt có dạng như hình một thung lũng giống như hình 1a) dưới đây. Bất kể ta đặt hòn bi ở A hay B thì cuối cùng hòn bi cũng sẽ lăn xuống và kết thúc ở vị trí C.  
    Tuy nhiên, nếu như bề mặt có hai đáy thung lũng như Hình 1b) thì tùy vào việc đặt bi ở A hay B, vị trí cuối cùng của bi sẽ ở C hoặc D. Điểm D là một điểm local minimum chúng ta không mong muốn. Nếu suy nghĩ một cách vật lý hơn, giả sử vận tốc ban đầu lớn hơn nữa, bi có thể vượt dốc tới điểm E rồi lăn xuống C như trong Hình 1c). Đây chính là điều chúng ta mong muốn.

## K-Nearest Neighbors (KNN)

* 1. KNN là gì?

KNN là một trong những thuật toán học có giám sát đơn giản nhất được sử dụng nhiều trong khai phá dữ liệu và học máy. Ý tưởng của thuật toán này là nó không học một điều gì từ tập dữ liệu học (nên KNN được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán nhãn của dữ liệu mới. Lớp (nhãn) của một đối tượng dữ liệu mới có thể dự đoán từ các lớp (nhãn) của k hàng xóm gần nó nhất.

Một cách ngắn gọn, KNN là thuật toán đi tìm đầu ra của một điểm dữ liệu mới bằng cách chỉ dựa trên thông tin của K điểm dữ liệu trong training set gần nó nhất (K-lân cận), không quan tâm đến việc có một vài điểm dữ liệu trong những điểm gần nhất này là nhiễu.

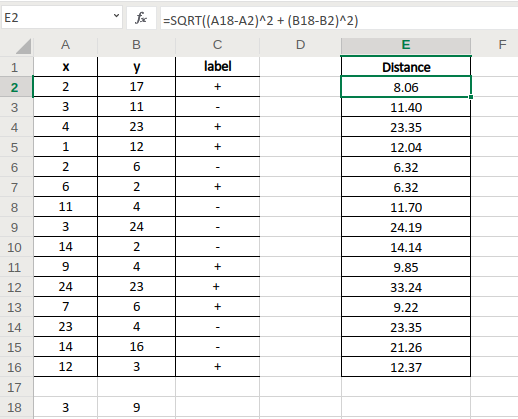
Ví dụ: bài toán Classification với 3 classes: Đỏ, Lam, Lục. Mỗi điểm dữ liệu mới (test data point) sẽ được gán label theo màu của điểm mà nó thuộc về. Trong hình này, có một vài vùng nhỏ xem lẫn vào các vùng lớn hơn khác màu. Ví dụ có một điểm màu Lục ở gần góc 11 giờ nằm giữa hai vùng lớn với nhiều dữ liệu màu Đỏ và Lam. Điểm này rất có thể là nhiễu. Dẫn đến nếu dữ liệu test rơi vào vùng này sẽ có nhiều khả năng cho kết quả không chính xác.

Map

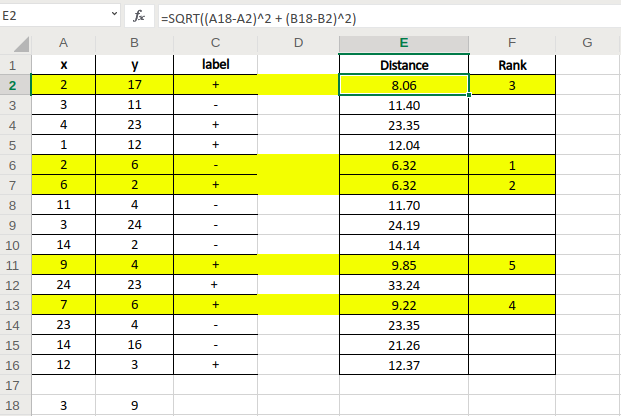
Description automatically generated

* 1. Các bước thực hiện trong KNN
* Ta có D là tập các điểm dữ liệu đã được gắn nhãn và A là dữ liệu chưa được phân loại.
* Đo khoảng cách (Euclidian, Manhattan, Minkowski, Minkowski hoặc Trọng số) từ dữ liệu mới A đến tất cả các dữ liệu khác đã được phân loại trong D.
* Chọn K (K là tham số mà bạn định nghĩa) khoảng cách nhỏ nhất.
* Kiểm tra danh sách các lớp có khoảng cách ngắn nhất và đếm số lượng của mỗi lớp xuất hiện.
* Lấy đúng lớp (lớp xuất hiện nhiều lần nhất).
* Lớp của dữ liệu mới là lớp mà bạn đã nhận được ở bước 5.

Ví dụ: Giả sử ta có tập dữ liệu D có gắn nhãn gồm 15 điểm như ảnh:



* Điểm cần dự đoán nhãn A(3,9)
* Ta tính khoảng cách từ điểm A đến các điểm dữ liệu trong D bằng công thức Euclidian.
* Ta chọn K= 5, và tìm ra 5 điểm có khoảng cách gần với điểm A nhất.
* Trong 5 điểm ta thấy có 4 điểm mang nhãn (+) và 1 điểm mang nhãn (-).
* Vậy ta có thể đưa ra kết luận là điểm A cần dự đoán mang nhãn (+).



* 1. Ưu nhược điểm của KNN

Ưu điểm

* Thuật toán đơn giản, dễ dàng triển khai.
* Độ phức tạp tính toán nhỏ.
* Xử lý tốt với tập dữ liệu nhiễu

Nhược điểm

* Với K nhỏ dễ gặp nhiễu dẫn tới kết quả đưa ra không chính xác
* Cần nhiều thời gian để thực hiện do phải tính toán khoảng cách với tất cả các đối tượng trong tập dữ liệu.
* Cần chuyển đổi kiểu dữ liệu thành các yếu tố định tính.

## Overfitting

* 1. Overfitting là gì?

Overfitting là hiện tượng mô hình tìm được quá khớp với dữ liệu training. Việc quá khớp này có thể dẫn đến việc dự đoán nhầm nhiễu, và chất lượng mô hình không còn tốt trên dữ liệu test nữa. Dữ liệu test được giả sử là không được biết trước, và không được sử dụng để xây dựng các mô hình Machine Learning.

Map

Description automatically generated

Về cơ bản, overfitting xảy ra khi mô hình quá phức tạp để mô phỏng training data. Điều này đặc biệt xảy ra khi lượng dữ liệu training quá nhỏ trong khi độ phức tạp của mô hình quá cao. Trong ví dụ trên đây, độ phức tạp của mô hình có thể được coi là bậc của đa thức cần tìm. Trong Multi-layer Perceptron, độ phức tạp của mô hình có thể được coi là số lượng hidden layers và số lượng units trong các hidden layers.

* 1. Những kỹ thuật giúp tránh Overfitting
* Validation: Phương pháp đơn giản nhất là trích từ tập training data ra một tập con nhỏ và thực hiện việc đánh giá mô hình trên tập con nhỏ này. Tập con nhỏ được trích ra từ training set này được gọi là validation set. Lúc này, training set là phần còn lại của training set ban đầu. Train error được tính trên training set mới này, và có một khái niệm nữa được định nghĩa tương tự như trên validation error, tức error được tính trên tập validation.

Chart, line chart

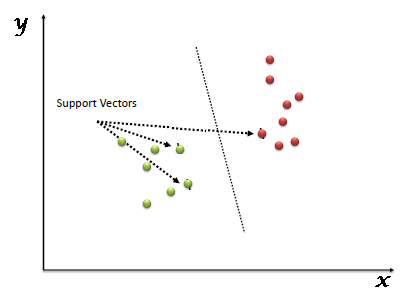
Description automatically generated

* Cross-validation: Cross validation là một cải tiến của validation với lượng dữ liệu trong tập validation là nhỏ nhưng chất lượng mô hình được đánh giá trên nhiều tập validation khác nhau. Một cách thường đường sử dụng là chia tập training ra k tập con không có phần tử chung, có kích thước gần bằng nhau. Tại mỗi lần kiểm thử, được gọi là run, một trong số k tập con được lấy ra làm validate set. Mô hình sẽ được xây dựng dựa vào hợp của k – 1 tập con còn lại. Mô hình cuối được xác định dựa trên trung bình của các train error và validation error. Cách làm này còn có tên gọi là k-fold cross validation. Khi k bằng với số lượng phần tử trong tập training ban đầu, tức mỗi tập con có đúng 1 phần tử, ta gọi kỹ thuật này là leave-one-out.
* Ngoài các phương pháp đã nêu ở trên, với mỗi mô hình, nhiều phương pháp tránh overfitting khác cũng được sử dụng. Điển hình là Dropout trong Deep Neural Networks mới được đề xuất gần đây. Một cách ngắn gọn, dropout là một phương pháp tắt ngẫu nhiên các units trong Networks. Tắt tức cho các unit giá trị bằng không và tính toán feedforward và backpropagation bình thường trong khi training. Việc này không những giúp lượng tính toán giảm đi mà còn làm giảm việc overffitng. Tôi xin được quay lại vấn đề này nếu có dịp nói sâu về Deep Learning trong tương lai.
  1. Kết luận
* Một mô hình mô tốt là mộ mô hình có tính tổng quát, tức mô tả được dữ liệu cả trong lẫn ngoài tập training. Mô hình chỉ mô tả tốt dữ liệu trong tập training được gọi là overfitting.
* Để tránh overfitting, có rất nhiều kỹ thuật được sử dụng, điển hình là cross-validation và regularization. Trong Neural Networks, weight decay và dropout thường được dùng.

## Support Vector Machine (SVM)

* 1. SVM là gì?

SVM là một thuật toán giám sát, nó có thể sử dụng cho cả việc phân loại hoặc đệ quy. Tuy nhiên nó được sử dụng chủ yếu cho việc phân loại. Trong thuật toán này sẽ vẽ đồ thị dữ liệu là các điểm trong n chiều (ở đây n là số lượng các tính năng) với giá trị của mỗi tính năng sẽ là một phần liên kết. Sau đó thực hiện tìm "đường bay" (hyper-plane) phân chia các lớp. Hyper-plane nó chỉ hiểu đơn giản là 1 đường thẳng có thể phân chia các lớp ra thành hai phần riêng biệt.



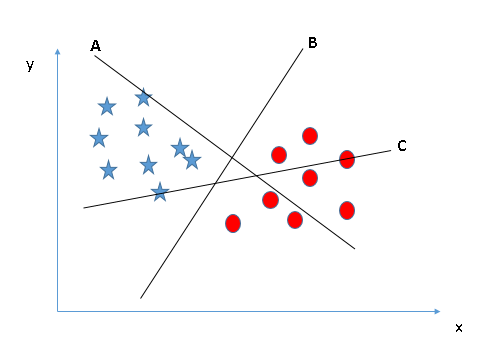
Support Vectors hiểu một cách đơn giản là các đối tượng trên đồ thị tọa độ quan sát, Support Vector Machine là một biên giới để chia hai lớp tốt nhất.

* 1. SVM làm việc như thế nào?

Để xác định "Làm sao để vẽ-xác định đúng hyper-plane", cần tuân theo các tiêu chí sau:

* Identify the right hyper-plane (Scenario-1):

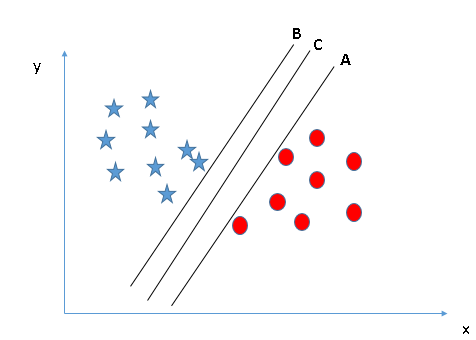
Ở đây, có 3 đường hyper-lane (A,B and C). Bây giờ đường nào là hyper-lane đúng cho nhóm ngôi sao và hình tròn.



Quy tắc số một để chọn 1 hyper-lane, chọn một hyper-plane để phân chia hai lớp tốt nhất. Trong ví dụ này chính là đường B.

* Identify the right hyper-plane (Scenario-2):

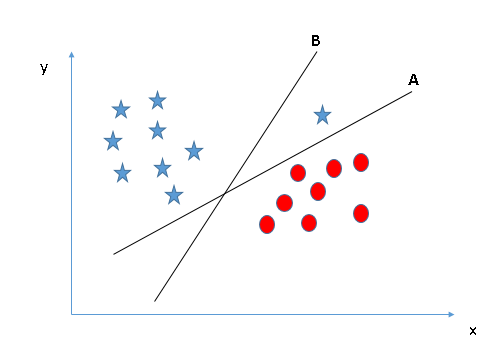
Ở đây cũng có 3 đường hyper-plane (A,B và C), theo quy tắc số 1, chúng đều thỏa mãn.



Quy tắc thứ hai chính là xác định khoảng cách lớn nhất từ điểu gần nhất của một lớp nào đó đến đường hyper-plane. Khoảng cách này được gọi là "Margin", Hãy nhìn hình bên dưới, trong đấy có thể nhìn thấy khoảng cách margin lớn nhất đấy là đường C. Cần nhớ nếu chọn lầm hyper-lane có margin thấp hơn thì sau này khi dữ liệu tăng lên thì sẽ sinh ra nguy cơ cao về việc xác định nhầm lớp cho dữ liệu.

* Identify the right hyper-plane (Scenario-3):

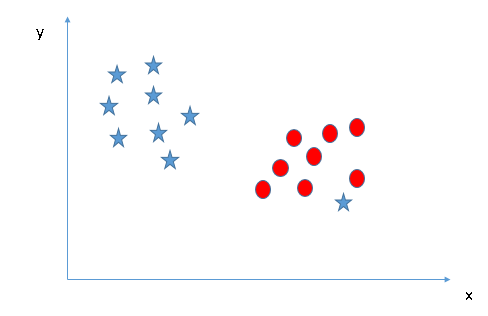
Sử dụng các nguyên tắc đã nêu trên để chọn ra hyper-plane cho trường hợp sau:



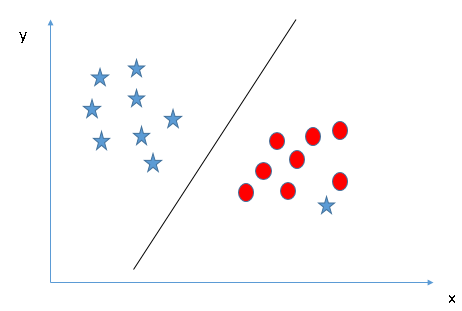
Có thể có một vài người sẽ chọn đường B bởi vì nó có margin cao hơn đường A, nhưng đấy sẽ không đúng bởi vì nguyên tắc đầu tiên sẽ là nguyên tắc số 1, chúng ta cần chọn hyper-plane để phân chia các lớp thành riêng biệt. Vì vậy đường A mới là lựa chọn chính xác.

* Can we classify two classes (Scenario-4):

Tiếp theo hãy xem hình bên dưới, không thể chia thành hai lớp riêng biệt với 1 đường thẳng, để tạo 1 phần chỉ có các ngôi sao và một vùng chỉ chứa các điểm tròn.

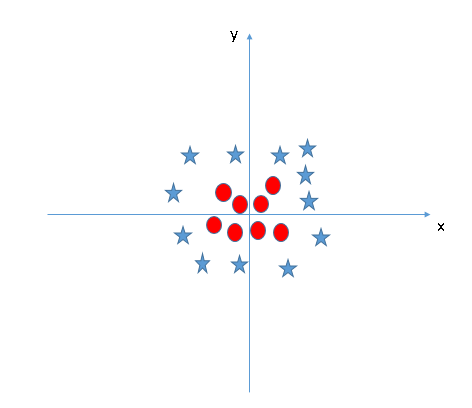


Ở đây sẽ chấp nhận, một ngôi sao ở bên ngoài cuối được xem như một ngôi sao phía ngoài hơn, SVM có tính năng cho phép bỏ qua các ngoại lệ và tìm ra hyper-plane có biên giới tối đa . Do đó có thể nói, SVM có khả năng mạnh trong việc chấp nhận ngoại lệ.

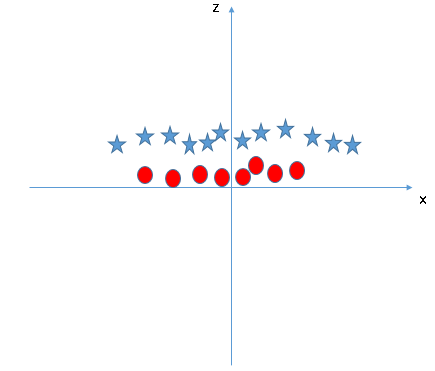


* Find the hyper-plane to segregate to classes (Scenario-5):

Trong trường hợp dưới đây, không thể tìm ra 1 đường hyper-plane tương đối để chia các lớp, vậy làm thế nào để SVM phân tách dữ liệu thành hai lớp riêng biệt? Cho đến bây giờ chúng ta chỉ nhìn vào các đường tuyến tính hyper-plane.



SVM có thể giải quyết vấn đề này, Khá đơn giản, nó sẽ được giải quyết bằng việc thêm một tính năng, Ở đây chúng ta sẽ thêm tính năng z = x^2+ y^2. Bây giờ dữ liệu sẽ được biến đổi theo trục x và z như sau:



Trong sơ đồ trên, các điểm cần xem xét là:

* Tất cả dữ liệu trên trục z sẽ là số dương vì nó là tổng bình phương x và y
* Trên biểu đồ các điểm tròn đỏ xuất hiện gần trục x và y hơn vì thế z sẽ nhỏ hơn => nằm gần trục x hơn trong đồ thị (z,x) Trong SVM, rất dễ dàng để có một siêu phẳng tuyến tính (linear hyper-plane) để chia thành hai lớp, Nhưng một câu hỏi sẽ nảy sinh đấy là, chúng ta có cần phải thêm một tính năng phân chia này bằng tay hay không. Không, bởi vì SVM có một kỹ thuật được gọi là kernel trick ( kỹ thuật hạt nhân), đây là tính năng có không gian đầu vào có chiều sâu thấm và biến đổi nó thành không gian có chiều cao hơn, tức là nó không phân chia các vấn đề thành các vấn đề riêng biệt, các tính năng này được gọi là kernel. Nói một cách đơn giản nó thực hiện một số biết đổi dữ liệu phức tạp, sau đó tìm ra quá trình tách dữ liệu dựa trên các nhãn hoặc đầu ra mà chúng ra đã xác định trước.
  1. Kết luận về bài toán

Là một kĩ thuật phân lớp khá phổ biến, SVM thể hiện được nhiều ưu điểm trong số đó có việc tính toán hiệu quả trên các tập dữ liệu lớn. Có thể kể thêm một số Ưu điểm của phương pháp này như:

* Xử lý trên không gian số chiều cao: SVM là một công cụ tính toán hiệu quả trong không gian chiều cao, trong đó đặc biệt áp dụng cho các bài toán phân loại văn bản và phân tích quan điểm nơi chiều có thể cực kỳ lớn.
* Tiết kiệm bộ nhớ: Do chỉ có một tập hợp con của các điểm được sử dụng trong quá trình huấn luyện và ra quyết định thực tế cho các điểm dữ liệu mới nên chỉ có những điểm cần thiết mới được lưu trữ trong bộ nhớ khi ra quyết định.
* Tính linh hoạt - phân lớp thường là phi tuyến tính. Khả năng áp dụng Kernel mới cho phép linh động giữa các phương pháp tuyến tính và phi tuyến tính từ đó khiến cho hiệu suất phân loại lớn hơn.

Nhược điểm:

* Bài toán số chiều cao: Trong trường hợp số lượng thuộc tính (p) của tập dữ liệu lớn hơn rất nhiều so với số lượng dữ liệu (n) thì SVM cho kết quả khá tồi.
* Chưa thể hiện rõ tính xác suất: Việc phân lớp của SVM chỉ là việc cố gắng tách các đối tượng vào hai lớp được phân tách bởi siêu phẳng SVM. Điều này chưa giải thích được xác suất xuất hiện của một thành viên trong một nhóm là như thế nào. Tuy nhiên hiệu quả của việc phân lớp có thể được xác định dựa vào khái niệm margin từ điểm dữ liệu mới đến siêu phẳng phân lớp mà chúng ta đã bàn luận ở trên.

Kết luận: SVM là một phương pháp hiệu quả cho bài toán phân lớp dữ liệu. Nó là một công cụ đắc lực cho các bài toán về xử lý ảnh, phân loại văn bản, phân tích quan điểm. Một yếu tố làm nên hiệu quả của SVM đó là việc sử dụng Kernel function khiến cho các phương pháp chuyển không gian trở nên linh hoạt hơn.

# Công nghệ sử dụng

## Git

* Git là một hệ thống quản lý phiên bản phân tán (có tên tiếng anh là Distributed Version Control System – DVCS) được ra đời vào năm 2005 hiện đang được sử dụng vô cùng phổ biến. Git cung cấp cho mỗi một lập trình viên kho lưu trữ riêng cho toàn bộ lịch sử thay đổi.
* Git có thể giúp bạn giải quyết triệt để và nhanh chóng các vấn đề như:

Lưu lại những phiên bản khác nhau của mã nguồn dự án phần mềm.

Khôi phục lại các mã nguồn từ một phiên bản bất kỳ khác.

Hỗ trợ so sánh dễ dàng hơn giữa các phiên bản.

Phát hiện được những vị trí, những phần mà người khác đã chỉnh sửa làm phát sinh lỗi.

Khôi phục lại những tập tin đã bị mất đi.

Thử nghiệm và mở rộng các tính năng của dự án một cách dễ dàng mà không làm ảnh hưởng đến các phiên bản chính.

Hỗ trợ phối hợp thực hiện dự án trong một nhóm mang lại nhiều hiệu quả hơn.

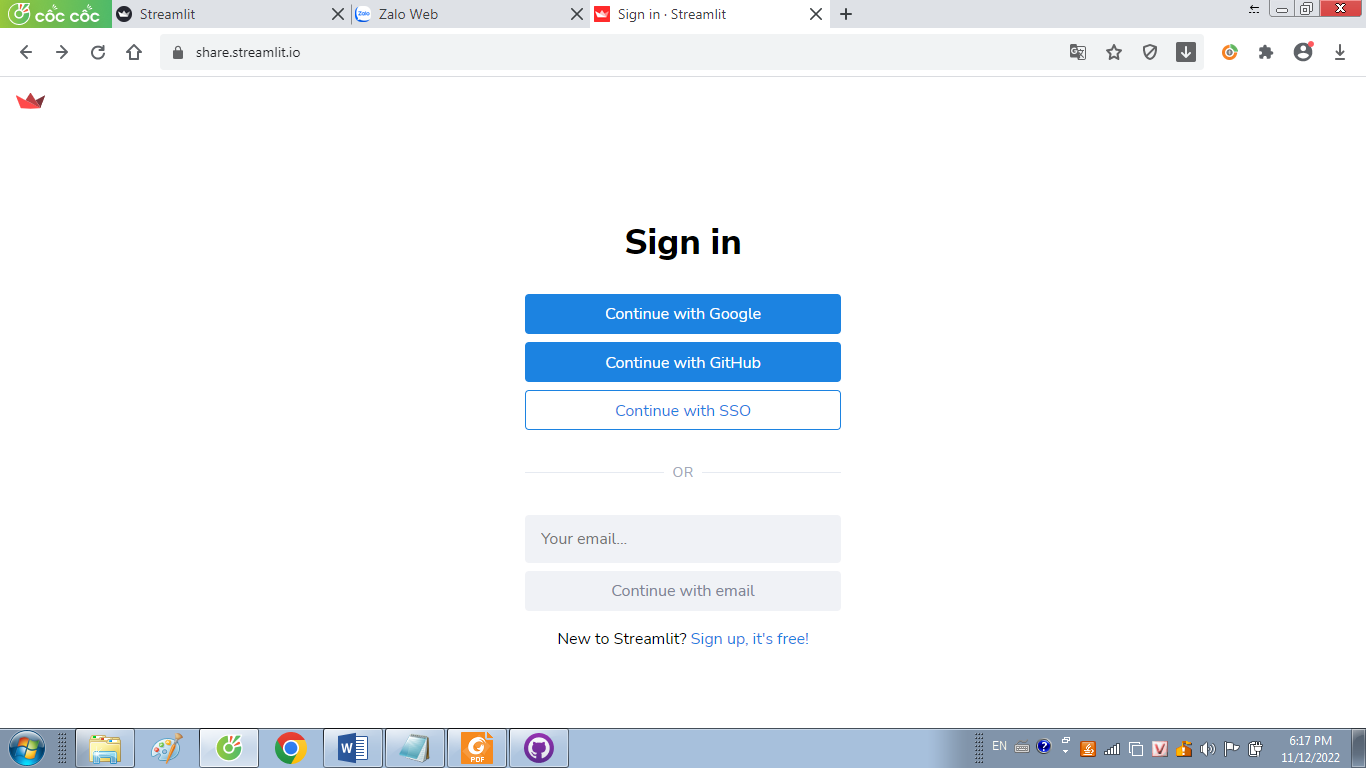
Git đảm bảo không có xung đột code giữa các lập trình viên trong một nhóm.

Chỉ cần có clone mã nguồn từ kho chứa hoặc clone một phiên bản thay đổi nào đó từ kho chứa hoặc một nhánh nào đó từ kho chứa là lập trình viên có thể bắt tay vào làm việc mọi lúc mọi nơi.

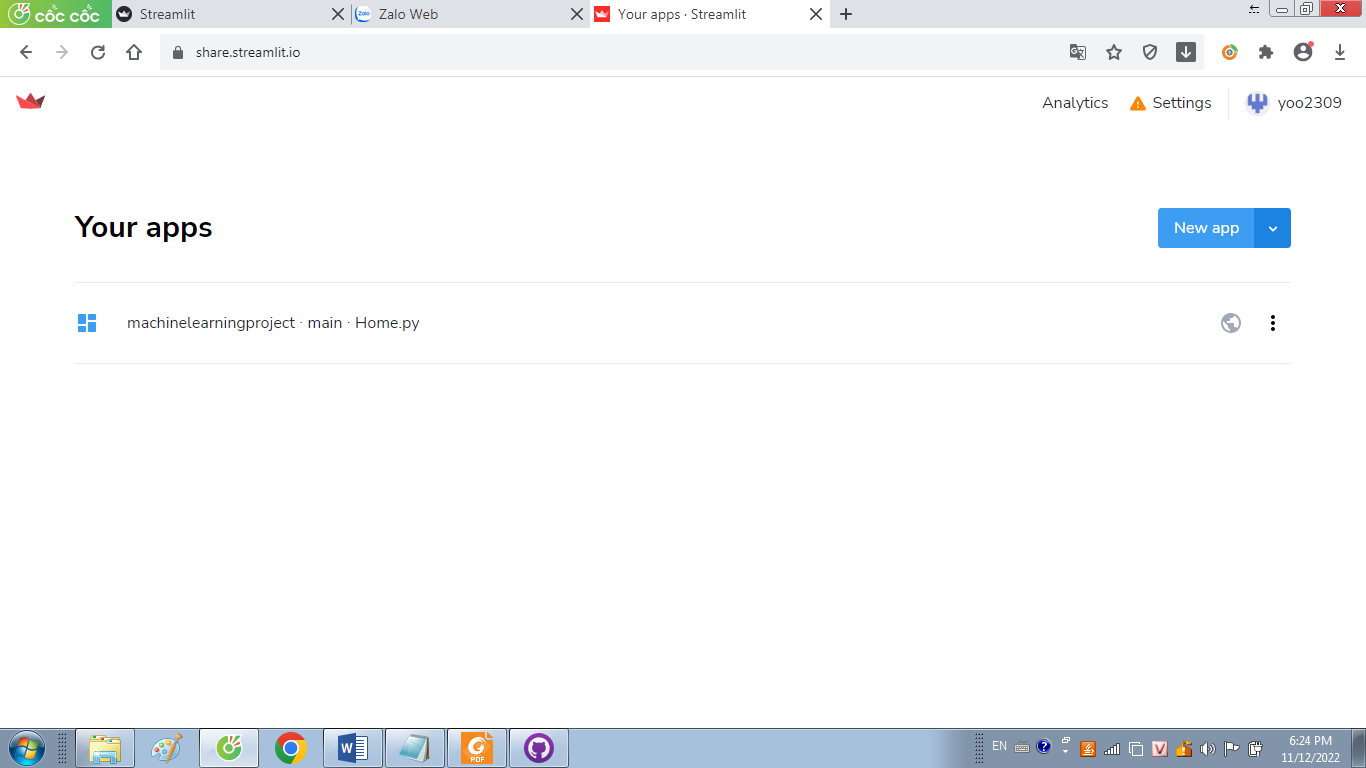
* Ứng dụng vào đồ án:
* Tạo Reponsitory – kho lưu trữ để lưu trữ dữ liệu để các thành viên thuận tiện trao đổi, lưu trữ dữ liệu và deploy web để có thể truy cập từ các máy bên ngoài.

## Streamlit

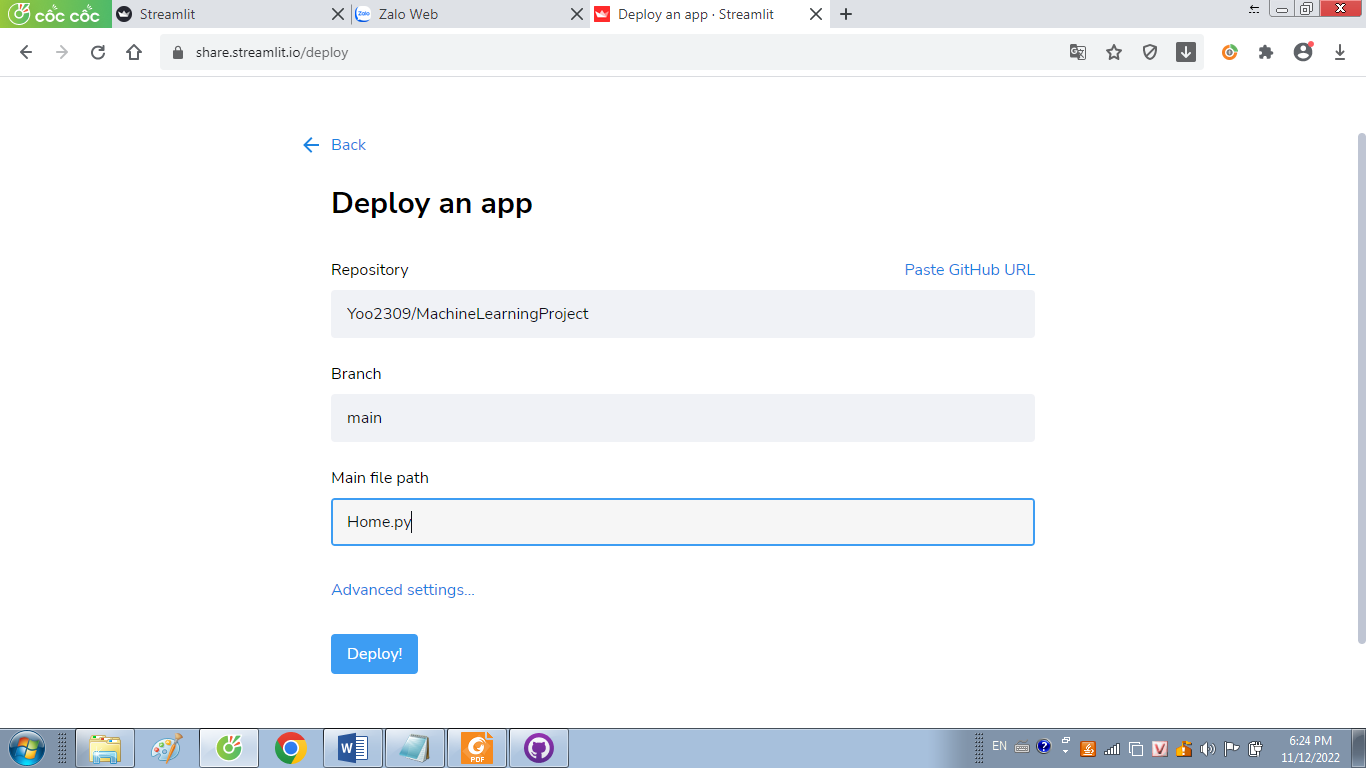
* Streamlit là một khung ứng dụng web dựa trên ngôn ngữ Python để trực quan hóa dữ liệu và phân tích kết quả một cách hiệu quả và linh hoạt hơn. Đây là một thư viện nguồn mở hỗ trợ các nhà khoa học dữ liệu, nhà phân tích dữ liệu phát triển bảng điều khiển trực quan hóa Machine Learning (ML) nhanh chóng. Chúng ta có thể xây dựng và triển khai các ứng dụng dữ liệu mạnh mẽ chỉ với một vài dòng code.
* Ứng dụng vào đồ án:
* Tạo trang web cho đồ án:
* Deploy web để có thể truy cập từ internet
  + Bước 1: Vào trang Streamlit <https://share.streamlit.io/> và đăng nhập với tài khoản Git:

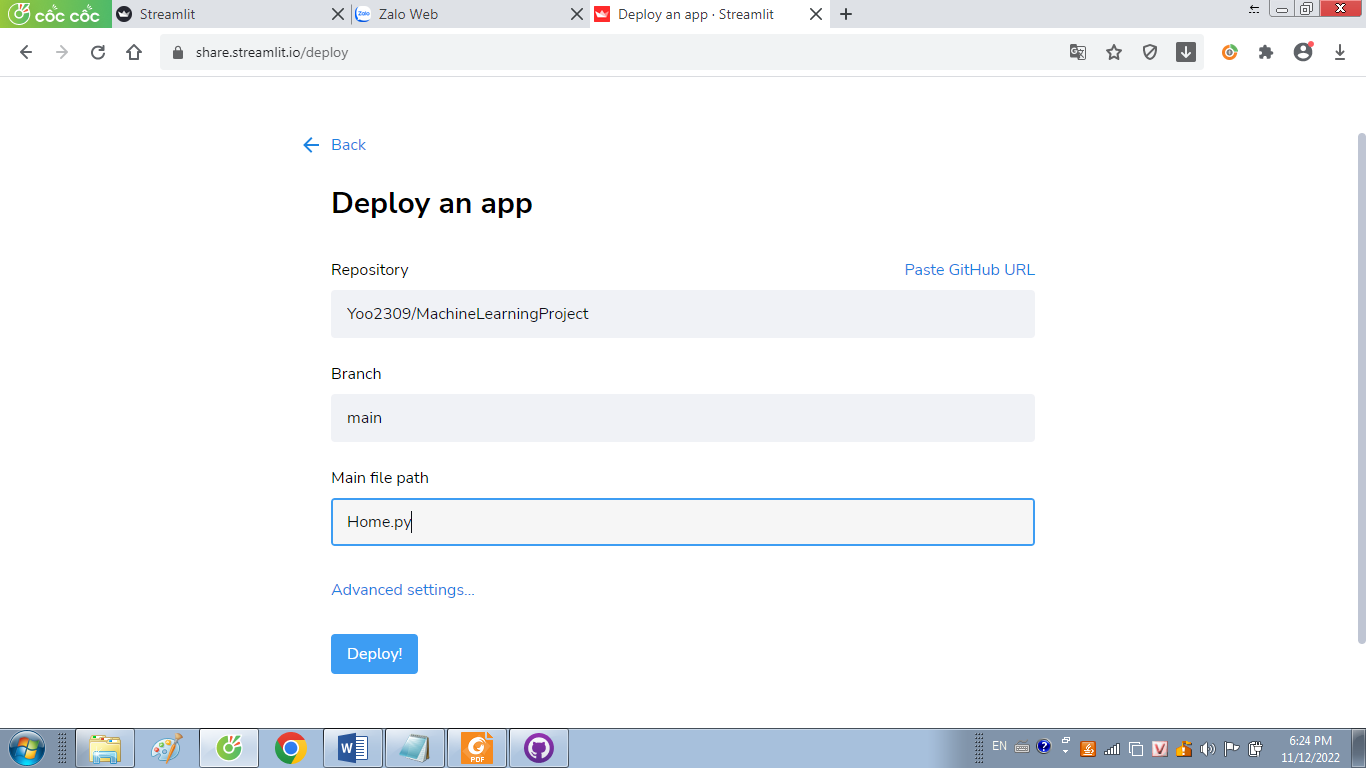


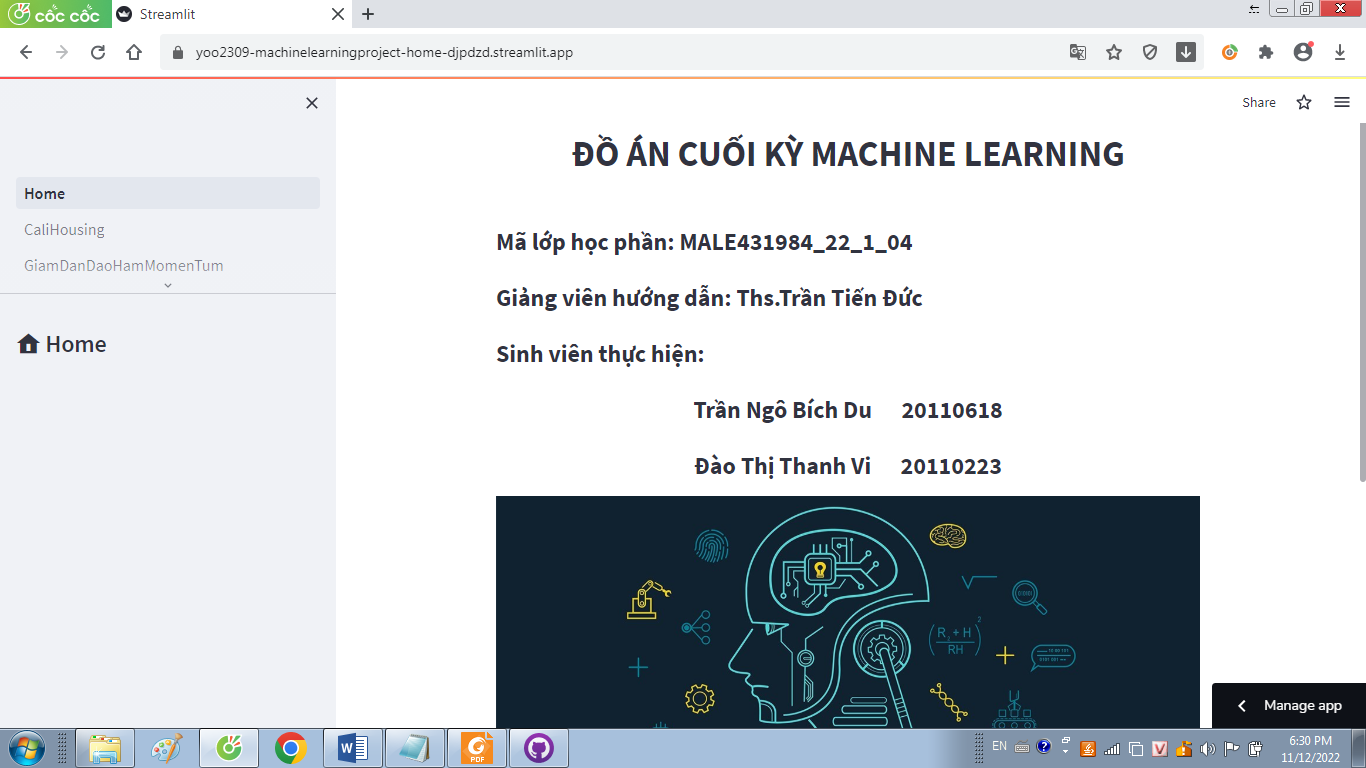
* + Bấm New App:



* + Điền các thông tin tương ứng của Repository muốn deploy:



* + Ấn  sẽ hoàn thành việc deploy trang web
  + Trang web sau khi deploy:



# Trang web cho môn học

1. Tìm hiểu về Framework Streamlit
2. Xây dựng trang web cho môn học Machine Learning sử dụng ngôn ngữ Python và framework Streamlit

# TÀI LIỆU THAM KHẢO

[1] <https://viblo.asia/p/gioi-thieu-ve-support-vector-machine-svm-6J3ZgPVElmB>

[2] https://machinelearningcoban.com/2017/04/09/smv/

[3] https://en.wikipedia.org/wiki/Support\_vector\_machine

[4] https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory

[5] https://trituenhantao.io/kien-thuc/van-de-overfitting-underfitting-trong-machine-learning/

[6] https://viblo.asia/p/knn-k-nearest-neighbors-1-djeZ14ejKWz

[]