

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**



**Môn học:** Học máy

**Mã môn học: MALE431984\_22\_1\_03**

**Giảng viên hướng dẫn:** T.S Trần Tiến Đức

**Nhóm sinh viên thực hiện:**

***Thành viên:***

*Võ Ngọc Quý - 20110709*

*Vũ Nguyễn Trung Khang - 20110277*

*Hồ Thành Danh - 20110207*

**XÂY DỰNG TRANG WEB ỨNG DỤNG MACHINE LEARNING**

**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**MỤC LỤC**

[**1.** **MỞ ĐẦU** 4](#_Toc121691419)

[**1.1.** **LÝ DO CHỌN ĐỀ TÀI** 4](#_Toc121691420)

[**1.2.** **MỤC ĐÍCH NGUYÊN CỨU** 4](#_Toc121691421)

[**2.** **CÁC CHỨC NĂNG CHÍNH CỦA WEB MACHINE LEARNING** 5](#_Toc121691422)

[**2.1.** **Cali housing:** 5](#_Toc121691423)

[2.1.1. Mô hình hồi quy tuyến tính Linear Regression 5](#_Toc121691424)

[2.1.2. Mô hình cây quyết định Decision Tree Regression 7](#_Toc121691425)

[2.1.3. Mô hình thuật toán Random Forest Regression 9](#_Toc121691426)

[2.1.4. Mô hình thuật toán Random Forest Regression Grid Search CV 13](#_Toc121691427)

[2.1.5. Random Forest Regression Random Search CV tương tự 14](#_Toc121691428)

[2.1.6. Phân nhóm Median Income 15](#_Toc121691429)

[**2.2.** **GRADIENT DESCENT** 15](#_Toc121691430)

[Gradient Descent cho hàm 1 biến 15](#_Toc121691431)

[Gradient Descent cho hàm nhiều biến 17](#_Toc121691432)

[**2.3.** **NHẬN DẠNG TRÁI CÂY** 19](#_Toc121691433)

[**2.4.** **NHẬN DẠNG KHUÔN MẶT** 21](#_Toc121691434)

[**2.5.** **K-NEAREST NEIGHBORS (KNN)** 27](#_Toc121691435)

[**2.6.** **SUPPORT VECTOR MACHINE (SVM)** 30](#_Toc121691436)

[**3.** **TÀI LIỆU THAM KHẢO** 33](#_Toc121691437)

1. **MỞ ĐẦU**
   1. **LÝ DO CHỌN ĐỀ TÀI**

Dưới góc nhìn của [trí tuệ nhân tạo](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o), động lực chính học máy bởi là nhu cầu thu nhận tri thức (knowledge acquisition). Thật vậy, trong nhiều trường hợp ta cần kiến thức chuyên gia là khan hiếm (không đủ chuyên gia ngồi phân loại lừa đảo thẻ tín dụng của tất cả giao dịch hàng ngày) hoặc chậm vì một số nhiệm vụ cần đưa ra quyết định nhanh chóng dựa trên xử lý dữ liệu khổng lồ (trong mua bán chứng khoán phải quyết định trong vài khoảng khắc của giây chẳng hạn) và thiếu ổn định thì buộc phải cần đến máy tính. Ngoài ra, đại đa số dữ liệu sinh ra ngày nay chỉ phù hợp cho máy đọc (computer readable) tiềm tàng ngưồn kiến thức quan trọng. Máy học nghiên cứu cách thức để mô hình hóa bài toán cho phép máy tính tự động hiểu, xử lý và học từ dữ liệu để thực thi nhiệm vụ được giao cũng như cách đánh giá giúp tăng tính hiệu quả.

Ứng dụng tất cả các kiến thức đã học trên lớp để viết được một trang web ứng dụng machine learning để phục vụ người dùng trong việc phân tích tính toán.

* 1. **MỤC ĐÍCH NGUYÊN CỨU**

Triển khai và ứng dụng được các thuật toán machine learning đã được học để hiểu sâu và nắm bắt được các kiến thức cần thiết của môn học này.

1. **CÁC CHỨC NĂNG CHÍNH CỦA WEB MACHINE LEARNING**
   1. **Cali housing:**

Ứng dụng machine learning dùng để dự đoán giá nhà ở Cali.

Huấn luyện: Từ dữ liệu thuật toán sẽ học ra model.

Dự đoán: Dùng model học được từ bước trên dự đoán các giá trị mới.

Dùng các mẫu thử để dự đoán ra:

+Sai số bình phương trung bình – train

+Sai số bình phương trung bình – cross – vailidation

+Mean

+Standard deviation

+Sai số bình phương trung bình - test

**Vấn đề**

Đã có một tập dữ liệu gồm các thuộc tính của bất động sản và giá của chúng

Các thuộc tính như: số phòng ngủ, số phòng tắm, vị trí địa lý, năm xây dựng...và giá bán

Giờ muốn tìm một công thức nào đó để từ một bất động sản bất kì, cho biết một vài thông tin như số phòng ngủ, vị trí địa lý... có thể tính toán ra được giá bán

### **2.1.1. Mô hình hồi quy tuyến tính Linear Regression**

Đối với mô hình hồi quy tuyến tính, mô hình hóa sử dụng là dạng bậc nhất. Có nghĩa là chúng ta cần tính toán các hệ số Wi trong một biểu thức bậc nhất như sau:

Y = W0 + W1.X1 + W2.X2 + ... + Wn.Xn

Tức là trên đồ thị chúng ta đi tìm một đường thẳng (nếu trường hợp có 1 biến độc lập) hoặc một siêu phẳng (với trường hợp nhiều biến độc lập) đi qua tập hợp các điểm trong không gian thuộc tính mà thể hiện gần đúng nhất sự phân bố của tập dữ liệu. Trên phương diện tính toán, chúng ta đi tìm các hệ số Wi như trên sao cho lỗi hồi quy đạt được là nhỏ nhất. Lại nói về lỗi hồi quy người ta thường sử dụng Phương pháp bình phương nhỏ nhất (least square). Cái đầu tiên muốn nói gì thì nói chúng ta cần phải có một tập dữ liệu. Tập dữ liệu trong bài viết này các bạn có thể tìm thấy ở đây. File CSV này chứa thông tin về các bất động sản như số phòng ngủ, số phòng tắm, năm xây dựng... và giá bán tương ứng của nó. Chúng ta sẽ áp dụng lý thuyết về mô hình hồi quy để từ tập dữ liệu này, xây dựng một hàm sử dụng để định giá cho một bất động sản bất kì trong tương lai.

**Text

Description automatically generated with medium confidence**

**Text

Description automatically generated**

### **2.1.2. Mô hình cây quyết định Decision Tree Regression**

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.

Tóm lại, cho dữ liệu về các đối tượng gồm các thuộc tính cùng với lớp (classes) của nó, cây quyết định sẽ sinh ra các luật để dự đoán lớp của các dữ liệu chưa biết.

*Ưu/nhược điểm của thuật toán cây quyết định*

***Ưu điểm***

Cây quyết định là một thuật toán đơn giản và phổ biến. Thuật toán này được sử dụng rộng rãi bới những lợi ích của nó:

Mô hình sinh ra các quy tắc dễ hiểu cho người đọc, tạo ra bộ luật với mỗi nhánh lá là một luật của cây.

Dữ liệu đầu vào có thể là là dữ liệu missing, không cần chuẩn hóa hoặc tạo biến giả

Có thể làm việc với cả dữ liệu số và dữ liệu phân loại

Có thể xác thực mô hình bằng cách sử dụng các kiểm tra thống kê

Có khả năng là việc với dữ liệu lớn

***Nhược điểm***

Kèm với đó, cây quyết định cũng có những nhược điểm cụ thể:

Mô hình cây quyết định phụ thuộc rất lớn vào dữ liệu của bạn. Thạm chí, với một sự thay đổi nhỏ trong bộ dữ liệu, cấu trúc mô hình cây quyết định có thể thay đổi hoàn toàn.

Cây quyết định hay gặp vấn đề overfitting

**Text

Description automatically generated**

**Text

Description automatically generated**

### **2.1.3. Mô hình thuật toán Random Forest Regression**

Random forest là thuật toán supervised learning, có thể giải quyết cả bài toán regression và classification.

Random là ngẫu nhiên, Forest là rừng, nên ở thuật toán Random Forest mình 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.

Ở bước huấn luyện thì mình sẽ xây dựng nhiều cây quyết định, các cây quyết định có thể khác nhau (phần sau mình sẽ nói mỗi cây được xây dựng như thế nào).

Giả sử bộ dữ liệu của mình có n dữ liệu (sample) và mỗi dữ liệu có d thuộc tính (feature).

Để xây dựng mỗi cây quyết định ta sẽ làm như sau:

B1: Lấy ngẫu nhiên n dữ liệu từ bộ dữ liệu với kĩ thuật Bootstrapping, hay còn gọi là random sampling with replacement. Tức khi mình sample được 1 dữ liệu thì mình không bỏ dữ liệu đấy ra mà vẫn giữ lại trong tập dữ liệu ban đầu, rồi tiếp tục sample cho tới khi sample đủ n dữ liệu. Khi dùng kĩ thuật này thì tập n dữ liệu mới của mình có thể có những dữ liệu bị trùng nhau.

B2: Sau khi sample được n dữ liệu từ bước 1 thì mình chọn ngẫu nhiên ở k thuộc tính (k < n). Giờ mình được bộ dữ liệu mới gồm n dữ liệu và mỗi dữ liệu có k thuộc tính.

B3: Dùng thuật toán Decision Tree để xây dựng cây quyết định với bộ dữ liệu ở bước 2.

Do quá trính xây dựng mỗi cây quyết định đều có yếu tố ngẫu nhiên (random) nên kết quả là các cây quyết định trong thuật toán Random Forest có thể khác nhau.

Thuật toán Random Forest sẽ bao gồm nhiều cây quyết định, mỗi cây được xây dựng dùng thuật toán Decision Tree trên tập dữ liệu khác nhau và dùng tập thuộc tính khác nhau. Sau đó kết quả dự đoán của thuật toán Random Forest sẽ được tổng hợp từ các cây quyết định.

Khi dùng thuật toán Random Forest, ta để ý các thuộc tính như: số lượng cây quyết định sẽ xây dựng, số lượng thuộc tính dùng để xây dựng cây. Ngoài ra, vẫn có các thuộc tính của thuật toán Decision Tree để xây dựng cây như độ sâu tối đa, số phần tử tối thiểu trong 1 node để có thể tách.

*Tại sao thuật toán Random Forest tốt?*

Trong thuật toán Decision Tree, khi xây dựng cây quyết định nếu để độ sâu tùy ý thì cây sẽ phân loại đúng hết các dữ liệu trong tập training dẫn đến mô hình có thể dự đoán tệ trên tập validation/test, khi đó mô hình bị overfitting, hay nói cách khác là mô hình có high variance.

Thuật toán Random Forest gồm nhiều cây quyết định, 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 mỗi cây quyết định trong thuật toán Random Forest không dùng tất cả dữ liệu training, cũng như không dùng tất cả các thuộc tính của dữ liệu để xây dựng cây nên mỗi cây có thể sẽ dự đoán không tốt, khi đó mỗi mô hình cây quyết định không bị overfitting mà có thế bị underfitting, hay nói cách khác là mô hình có high bias. Tuy nhiên, 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, hay mô hình có kết quả dự đoán tốt.

Ý tưởng tổng hợp các cây quyết định của thuật toán Random Forest giống với ý tưởng của The Wisdom of Crowds được đề xuất bởi by James Surowiecki vào năm 2004. The Wisdom of Crowds nói rằng thông thường tổng hợp thông tin từ 1 nhóm sẽ tốt hơn từ một cá nhân. Ở thuật toán Random Forest tổng hợp thông tin từ 1 nhóm các cây quyết định và kết quả cho ra tốt hơn thuật toán Decision Tree với 1 cây quyết định.

**Text

Description automatically generated**

**Text

Description automatically generated**

### **2.1.4. Mô hình thuật toán Random Forest Regression Grid Search CV**

**Text

Description automatically generatedText

Description automatically generated**

### **2.1.5. Random Forest Regression Random Search CV tương tự**

**Text

Description automatically generatedText

Description automatically generated**

### **2.1.6. Phân nhóm Median Income**

* 1. **GRADIENT DESCENT**

### **Gradient Descent cho hàm 1 biến**

Trong Machine Learning nói riêng và Toán Tối Ưu nói chung, chúng ta thường xuyên phải tìm giá trị nhỏ nhất (hoặc đôi khi là lớn nhất) của một hàm số nào đó. Ví dụ như các hàm mất mát trong hai bài [Linear Regression](https://machinelearningcoban.com/2016/12/28/linearregression/) và [K-means Clustering](https://machinelearningcoban.com/2017/01/01/kmeans/). 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.

Các điểm local minimum là nghiệm của phương trình đạo hàm bằng 0. Nếu bằng một cách nào đó có thể tìm được toàn bộ (hữu hạn) các điểm cực tiểu, ta chỉ cần thay từng điểm local minimum đó vào hàm số rồi tìm điểm làm cho hàm có giá trị nhỏ nhất. Tuy nhiên, trong hầu hết các trường hợp, việc giải phương trình đạo hàm bằng 0 là bất khả thi. Nguyên nhân có thể đến từ sự phức tạp của dạng của đạo hàm, từ việc các điểm dữ liệu có số chiều lớn, hoặc từ việc có quá nhiều điểm dữ liệu.

Hướng tiếp cận phổ biến nhất là xuất phát từ một điểm mà chúng ta coi là gần với nghiệm của bài toán, sau đó dùng một phép toán lặp để tiến dần đến điểm cần tìm, tức đến khi đạo hàm gần với 0. Gradient Descent (viết gọn là GD) và các biến thể của nó là một trong những phương pháp được dùng nhiều nhất.

1. grad để tính đạo hàm
2. cost để tính giá trị của hàm số. Hàm này không sử dụng trong thuật toán nhưng thường được dùng để kiểm tra việc tính đạo hàm của đúng không hoặc để xem giá trị của hàm số có giảm theo mỗi vòng lặp hay không.
3. myGD1 là phần chính thực hiện thuật toán Gradient Desent nêu phía trên. Đầu vào của hàm số này là learning rate và điểm bắt đầu. Thuật toán dừng lại khi đạo hàm có độ lớn đủ nhỏ.

Xét f(x)=x^2 + 5\*sin(x) => f’(x)=2\*x + 5\*cos(x)

Text

Description automatically generated

Sau khi có các hàm cần thiết, thử tìm nghiệm với các điểm khởi tạo khác nhau là x0=−5 và x0=5

Text

Description automatically generatedText

Description automatically generated

Chart, histogram

Description automatically generated

Qua biểu đồ ta thấy

Solution x1 = -1.110667, cost = -3.246394, obtained after 11 iterations

Solution x2 = -1.110341, cost = -3.246394, obtained after 29 iterations

Từ hình minh họa trên ta thấy rằng ở hình bên trái, tương ứng với x0=−5, nghiệm hội tụ nhanh hơn, vì điểm ban đầu x0 gần với nghiệm x∗≈−1 hơn. Hơn nữa, với x0=5 ở hình bên phải, *đường đi* của nghiệm có chứa một khu vực có đạo hàm khá nhỏ gần điểm có hoành độ bằng 2. Điều này khiến cho thuật toán *la cà* ở đây khá lâu. Khi vượt qua được điểm này thì mọi việc diễn ra rất tốt đẹp.

### **Gradient Descent cho hàm nhiều biến**

Chúng ta tạo 1000 điểm dữ liệu được chọn gần với đường thẳng y=4+3x, hiển thị chúng và tìm nghiệm theo công thức:

Text

Description automatically generatedChart, scatter chart

Description automatically generated

Solution found by formula: w = [[ 4.00305242 2.99862665]]

Đường thẳng tìm được là đường có màu đỏ có phương trình y≈4+2.998

* 1. **NHẬN DẠNG TRÁI CÂY**

Dùng module keras của thư viện tensorflow, bs4, pillow, streamlit để chạy

Sample: Text

Description automatically generatedXử lý mẫu ảnh: Text

Description automatically generatedChạy chương trình:Text

Description automatically generated

Sau khi chạy thuật toán, kết quả:

A screenshot of a computer

Description automatically generated with medium confidence

* 1. **NHẬN DẠNG KHUÔN MẶT**

Nhận dạng mặt người (Face recognition) là một lĩnh vực nghiên cứu của ngành Computer Vision, và cũng được xem là một lĩnh vực nghiên cứu của ngành Biometrics (tương tự như nhận dạng vân tay – Fingerprint recognition, hay nhận dạng mống mắt – Iris recognition). Xét về nguyên tắc chung, nhận dạng mặt có sự tương đồng rất lớn với nhận dạng vân tay và nhận dạng mống mắt, tuy nhiên sự khác biệt nằm ở bước trích chọn đặt trưng (feature extraction) của mỗi lĩnh vực.

Trong khi nhận dạng vân tay và mống mắt đã đạt tới độ chín, tức là có thể áp dụng trên thực tế một cách rộng rãi thì nhận dạng mặt người vẫn còn nhiều thách thức và vẫn là một lĩnh vực nghiên cứu thú vị với nhiều người. So với nhận dạng vân tay và mống mắt, nhận dạng mặt có nguồn dữ liệu phong phú hơn (bạn có thể nhìn thấy mặt người ở bất cứ tấm ảnh, video clip nào liên quan tới con người trên mạng) và ít đòi hỏi sự tương tác có kiểm soát hơn (để thực hiện nhận dạng vân tay hay mống mắt, dữ liệu input lấy từ con người đòi hỏi có sự hợp tác trong môi trường có kiểm soát).

Hiện nay các phương pháp nhận dạng mặt được chia thành nhiều hướng theo các tiêu chí khác nhau: nhận dạng với dữ liệu đầu vào là ảnh tĩnh 2D (still image based FR) là phổ biến nhất, tuy nhiên tương lai có lẽ sẽ là 3D FR (vì việc bố trí nhiều camera 2D sẽ cho dữ liệu 3D và đem lại kết quả tốt hơn, đáng tin cậy hơn), cũng có thể chia thành 2 hướng là: làm với dữ liệu ảnh và làm với dữ liệu video.

Trên thực tế người ta hay chia các phương pháp nhận dạng mặt ra làm 3 loại: phương pháp tiếp cận toàn cục (global, như Eigenfaces-PCA, Fisherfaces-LDA), phương pháp tiếp cận dựa trên các đặc điểm cục bộ (local feature based, như LBP, Gabor wavelets) và phương pháp lai (hybrid, là sự kết hợp của hai phương pháp toàn cục và local feature). Phương pháp dựa trên các đặc điểm cục bộ đã được chứng minh là ưu việt hơn khi làm việc trong các điều kiện không có kiểm soát và có thể nói rằng lịch sử phát triển của nhận dạng mặt (A never ending story) là sự phát triển của các phương pháp trích chọn đặc trưng (feature extractrion methods) được sử dụng trong các hệ thống dựa trên feature based.

Các ứng dụng cụ thể của nhận dạng mặt dựa trên 2 mô hình nhận dạng: identification (xác định danh tính, bài toán 1-N), và verification (xác thực danh tính, bài toán 1-1). Trong bài toán identification, ta cần xác định danh tính của ảnh kiểm tra, còn trong bài toán verification ta cần xác định 2 ảnh có cùng thuộc về một người hay không.

Dùng thư viện opencv, matplotlib và module Image trong PIL

Text

Description automatically generated

Hàm nhận diện khuôn mặt:

Text

Description automatically generated

Text

Description automatically generated

Chạy kết quả kiểm thử:

Demo:

Graphical user interface, application

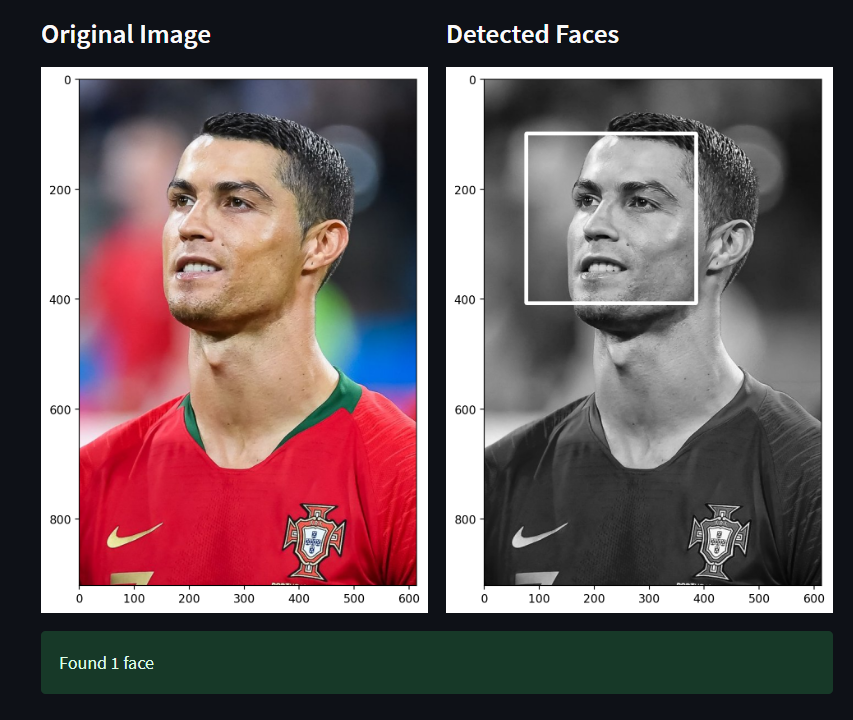
Description automatically generated

Mẫu thử khi load ảnh phải giới hạn 200MB, định dạng JPG, PNG, JPEG:

Graphical user interface, text

Description automatically generated

Xóa màu ảnh rồi bắt đầu nhận diện:



Tiếp tục tải mẫu thử có 2 khuôn mặt để kiểm tra thuật toán:

A picture containing text, monitor, screen, screenshot

Description automatically generated

* 1. **K-NEAREST NEIGHBORS (KNN)**

K-nearest neighbor là một trong những thuật toán supervised-learning đơn giản nhất (mà hiệu quả trong một vài trường hợp) trong Machine Learning. Khi training, thuật toán này không học một điều gì từ dữ liệu training (đây cũng là lý do thuật toán này được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới. K-nearest neighbor có thể áp dụng được vào cả hai loại của bài toán Supervised learning là Classification và Regression. KNN còn được gọi là một thuật toán Instance-based hay Memory-based learning.

KNN hoạt động với nguyên lý tương tự. Giá trị của một điểm dữ liệu được xác định bởi các điểm dữ liệu xung quanh nó.

***Quy trình làm việc của thuật toán KNN:***

Buớc 1: Xác định tham số k số láng giềng gần nhất.

Bước 2: Tính khoảng cách đối tượng cần phân lớp với tất cả các đối tượng trong training data

Bước 3: Sắp xếp khoảng cách theo thứ tự tăng dần và xác định k láng giềng gần nhất với đối tượng cần phân lớp

Bước 4: Lấy tất cả các lớp K của láng giềng gần nhất.

Bước 5: Dựa vào phân lớp của K để phân loại đối tượng cần phân lớp.

Các bước thực hiện:

+ Tạo bộ cơ sở dữ liệuText

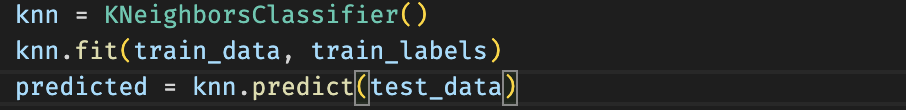
Description automatically generated+ Thí nghiệm

Khai báo thư viện**Text

Description automatically generated**

Tách training và test sets

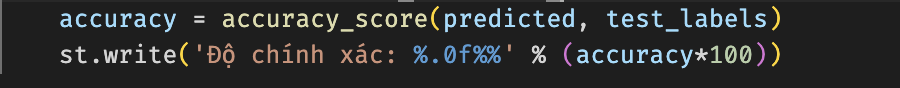
**Text

Description automatically generated**KNN dự đoán với K=5****Kết quả:

**Chart, scatter chart

Description automatically generated**

Đánh giá

****

* 1. **SUPPORT VECTOR MACHINE (SVM)**

Support Vector Machines (SVM) là kỹ thuật mới đối với việc phân lớp dữ liệu, là phương pháp học sử dụng không gian giả thuyết các hàm tuyến tính trên không gian đặc trƣng nhiều chiều, dựa trên lý thuyết tối ưu và lý thuyết thống kê. Trong kỹ thuật SVM không gian dữ liệu nhập ban đầu sẽ được ánh xạ vào không gian đặc trưng và trong không gian đặc trưng này mặt siêu phẳng phân chia tối ưu sẽ được xác định.

Bài toán phân lớp (Classification) và dự đoán (Prediction) là hai bài toán cơ bản và có rất nhiều ứng dụng trong tất cả các lĩnh vực như: học máy, nhận dạng, trí tuệ nhân tạo, v.v . Trong khóa luận này, chúng em sẽ đi sâu nghiên cứu phương pháp Support Vector Machines (SVM), một phương pháp rất hiệu quả hiện nay. Phương pháp SVM đƣợc coi là công cụ mạnh cho những bài toán phân lớp phi tuyến tính được các tác giả Vapnik và Chervonenkis phát triển mạnh mẽ năm 1995. Phương pháp này thực hiện phân lớp dựa trên nguyên lý Cực tiểu hóa Rủi ro có Cấu trúc SRM (Structural Risk Minimization), đƣợc xem là một trong các phương pháp phân lớp giám sát không tham số tinh vi nhất cho đến nay. Các hàm công cụ đa dạng của SVM cho phép tạo không gian chuyên đổi để xây dựng mặt phẳng phân lớp.

Các bước thực hiện:

+ Tạo bộ cơ sở dữ liệu

Text

Description automatically generated

+ Thí nghiệm

Khai báo thư viện

**Text

Description automatically generated**

Tách training và test sets

**Text

Description automatically generated**

Phân lớp sử dụng SVM với Kernel

**Graphical user interface, text, application

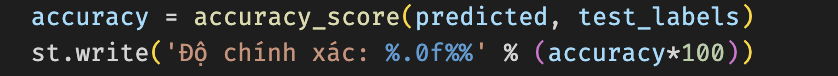
Description automatically generated**

Kết quả

**Chart, scatter chart

Description automatically generated**

Đánh giá:

****

1. **TÀI LIỆU THAM KHẢO**

Calihousing: <https://machinelearningcoban.com/tabml_book/ch_model/decision_tree.html>

Haarcascade OpenCV face recognize: <https://github.com/opencv/opencv/tree/master/data/haarcascades>

Recognize Fruit:

<https://github.com/nimadorostkar/Recognize-fruit>

SVM: <https://viblo.asia/p/ung-dung-support-vector-machine-trong-bai-toan-phan-loai-hoa-PdbGnLXBkyA>

KNN: <https://www.slideshare.net/trongthuy3/luan-van-nghien-cuu-thuat-toan-knearest-neighbor-haym-9d>

Gradient Descent: <https://machinelearningcoban.com/2017/01/12/gradientdescent/>