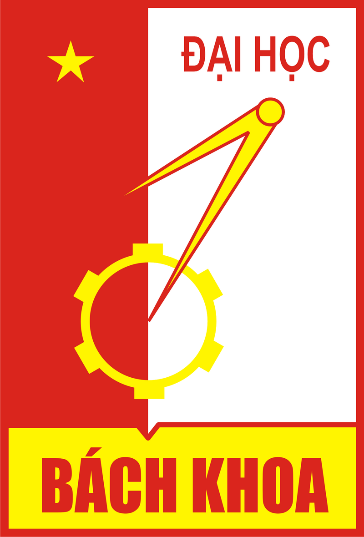
**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN ĐIỆN – ĐIỆN TỬ**



**ĐỒ ÁN III:**

**ĐỀ TÀI: TÌM HIỂU CÁC THUẬT TOÁN MACHINE LEARNING VÀ ỨNG DỤNG VÀO DỰ ĐOÁN GIÁ NHÀ**

**Giảng viên hướng dẫn:** PGS.TS Nguyễn Hữu Phát

**Sinh viên thực hiện:**

Trần Ngọc Đức – 20192788

***Hà Nội, Tháng 1 năm 2024***

**PHỤ LỤC**

[GIỚI THIỆU ĐỀ TÀI 5](#_Toc157006109)

[**I. Bối cảnh:** 5](#_Toc157006110)

[**II. Mục đích:** 5](#_Toc157006111)

[**III. Nhiệm vụ:** 5](#_Toc157006112)

[CƠ SỞ LÝ THUYẾT 6](#_Toc157006113)

[**I.** **Một số thuật toán Machine Learning:** 6](#_Toc157006114)

[**1.** **Các thuật toán Hồi quy:** 6](#_Toc157006115)

[**1.1** **Hồi quy tuyến tính đơn biến** 6](#_Toc157006116)

[**1.1.1** **Thuật toán** 6](#_Toc157006117)

[**1.1.2** **Ứng dụng** 12](#_Toc157006118)

[**1.2** **Hồi quy đa thức** 17](#_Toc157006119)

[**1.2.1** **Thuật toán** 17](#_Toc157006120)

[**1.2.2** **Ứng dụng** 18](#_Toc157006121)

[**1.3** **Support Vector Regression:** 20](#_Toc157006122)

[**1.3.1** **Thuật toán:** 20](#_Toc157006123)

[**1.3.2** **Ứng dụng** 24](#_Toc157006124)

[**2.** **Các thuật toán phân loại:** 26](#_Toc157006125)

[**2.1 Logistic Regression** 26](#_Toc157006126)

[**2.1.1 Thuật toán:** 26](#_Toc157006127)

[**2.1.2 Ứng dụng:** 33](#_Toc157006128)

[**2.2. K – Nearest Neighbors:** 35](#_Toc157006129)

[**2.2.1. Thuật toán:** 35](#_Toc157006130)

[**2.3.2. Ứng dụng:** 38](#_Toc157006131)

[**2.3. Support Vector Machine** 40](#_Toc157006132)

[**2.3.1. Thuật toán** 40](#_Toc157006133)

[**2.3.2. Ứng dụng** 44](#_Toc157006134)

[**2.4 Random Forest:** 47](#_Toc157006135)

[**2.4.1. Thuật toán:** 47](#_Toc157006136)

[**2.4.2 Ứng dụng:** 49](#_Toc157006137)

[**3. Các thuật toán phân cụm:** 52](#_Toc157006138)

[**3.1. K-Means Clustering:** 52](#_Toc157006139)

[**3.1.1 Thuật toán:** 52](#_Toc157006140)

[**3.1.2 Ứng dụng:** 58](#_Toc157006141)

[**II. Các thư viện của python hỗ trợ trong Machine Learning:** 62](#_Toc157006142)

[**1. Pandas:** 62](#_Toc157006143)

[**1.1. Giới thiệu:** 62](#_Toc157006144)

[**1.2 Ứng dụng:** 66](#_Toc157006145)

[**2. Numpy:** 66](#_Toc157006146)

[3. Sklearn: 71](#_Toc157006147)

[**3.1. Giới thiệu:** 71](#_Toc157006148)

[**3.2. Các thuật toán trong thư viện:** 71](#_Toc157006149)

[**4. Matplotlib:** 73](#_Toc157006150)

[**4.1. Giới thiệu:** 73](#_Toc157006151)

[**4.2. Thành phần:** 73](#_Toc157006152)

[ỨNG DỤNG 74](#_Toc157006153)

# GIỚI THIỆU ĐỀ TÀI

## **I. Bối cảnh:**

Số tiền dành cho mua nhà là không nhỏ, vì vậy việc người mua nhà quan tâm không chỉ nằm ở việc lựa chọn một ngôi nhà có thể đáp ứng đầy đủ các nhu cầu mà còn xem giá cả ngôi nhà có thực sự phù hợp với mình hay không.

Việc đánh giá giá trị của ngôi nhà không phải là việc dễ dàng mà nó đòi hỏi sự hiểu biết về thị trường bất động sản (một thị trường vốn rất biến động) cũng như sự hiểu biết thật sự tường tận về các thuộc tính của bất động sản đó.

Những kiến thức này thường chỉ được lưu trữ bởi các đại lý kinh doanh bất động sản. Nếu chúng ta có thể sử dụng kiến thức này bằng cách thu thập dữ liệu, sử dụng các dữ liệu có sẵn, tận dụng khả năng tính toán của các thuật toán, chương trình máy tính để giúp đưa ra quyết định mà không cần phải phụ thuộc vào các chuyên gia.

Để có thể ước lượng giá bất động sản, ngoài các phương pháp truyền thống, trên thế giới đã và đang nghiên cứu, áp dụng rộng rãi các phương pháp có sử dụng đến các mô hình toán học để xác định giá trị.

Đối với khu vực thành phố, do dân cư đông đúc, việc sống trong các căn hộ, chung cư đã trở nên không còn xa lạ, trở thành một trong những loại hình đầu tư phổ biến bậc nhất. Vì vậy, đối tượng mà em hướng tới nghiên cứu chính là áp dụng mô hình học máy nhằm dự đoán giá trị nhà ở, chung cư một cách tương đối chính xác.

## **II. Mục đích:**

Đề tài có hai mục đích chính:

* Tìm hiểu về Machine Learning và các thuật toán thường được sử dụng trong Machine Learning
* Tiến hành huấn luyện mô hình máy học nhằm dự đoán một cách tương đối giá nhà ở, chung cư tại TP. Hà Nội

# CƠ SỞ LÝ THUYẾT

1. **Một số thuật toán Machine Learning:**
2. **Các thuật toán Hồi quy:**
   1. **Hồi quy tuyến tính đơn biến**
      1. **Thuật toán**

Là một trong những thuật toán đầu của Machine Learning, thuộc nhóm Supervised Learning (Học có giám sát)

Hồi quy tuyến tính đơn thức là một mô hình hồi quy gồm hai biến là biến độc lập và biến phụ thuộc, nó liên quan đến các điểm mẫu trong không gian hai chiều sao cho từ một biến độc lập ta có thể tìm ra một biến phụ thuộc tương ứng nhờ vào một hàm tuyến tính (một đường thẳng) chính xác nhất có thể nhằm dự đoán các giá trị dựa trên hàm tuyến tính để tìm ra kết quả. Hàm dự đoán liên quan đến một yếu tố dự đoán duy nhất.

Mô hình hồi quy mô tả mối quan hệ giữa các biến bằng cách vẽ một đường thẳng để khít với bộ dữ liệu nhất có thể. Các mô hình hồi quy tuyến tính sử dụng một đường thẳng, trong khi các mô hình hồi quy logistic và phi tuyến tính sử dụng một đường cong. Hồi quy cho phép ước tính cách một biến phụ thuộc thay đổi khi các biến độc lập thay đổi.

Hồi quy đơn thức đơn biến nghĩa là ta tìm một đường thẳng (công thức) để fit nhất với tập dữ liệu đã có.

Hồi quy tuyến tính đơn biến được sử dụng để ước tính mối quan hệ giữa hai biến định lượng. Sử dụng hồi quy tuyến tính đơn biến khi bạn muốn biết:

* Mối quan hệ chặt chẽ như thế nào giữa hai biến số (ví dụ mối quan hệ giữa lượng mưa và xói mòn đất)
* Giá trị của biến phụ thuộc tại một giá trị nhất định của biến độc lập (VD: lượng đất xói mòn ở một lượng mưa nhất định)

*Ví dụ:*

Một nhà nghiên cứu về xã hội quan tâm đến mối quan hệ giữa thu nhập và hạnh phúc (thu nhập bao nhiêu thì độ hạnh phúc của chúng ta như thế nào). Họ khảo sát 500 người có thu nhập từ 15 nghìn USD/năm đến 75 nghìn USD/năm và yêu cầu họ xếp hạng mức độ hạnh phúc của họ trên thang điểm từ 1 đến 10.

Biến độc lập (thu nhập) và biến phụ thuộc (hạnh phúc) của bạn đều là định lượng (biến độc lập thu nhập quy định về biến độc lập hạnh phúc), vì vậy ta có thể thực hiện phân tích hồi quy để xem mối quan hệ tuyến tính giữa chúng.

Nếu có nhiều hơn một biến độc lập, sử dụng hồi quy tuyến tính đa biến để thay thế thay vì hồi quy tuyến tính đơn biến

Đi tìm mô hình hồi quy tuyến tính đơn biến là đi tìm mô hình có dạng theo công thức:

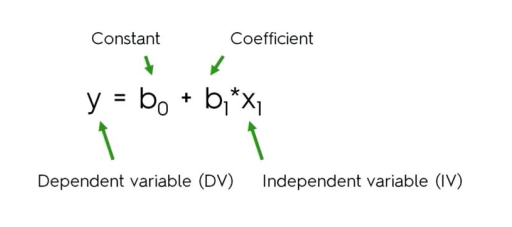


Trong đó: là biến phụ thuộc

là biến độc lập

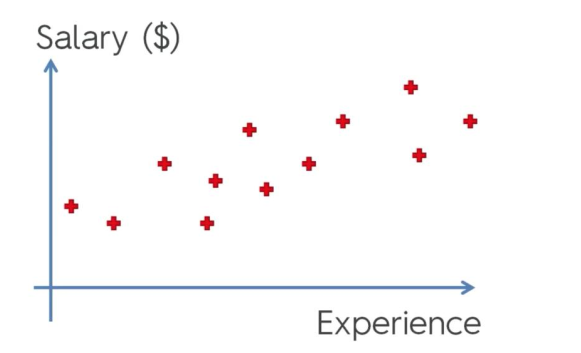
là hằng số

là hệ số

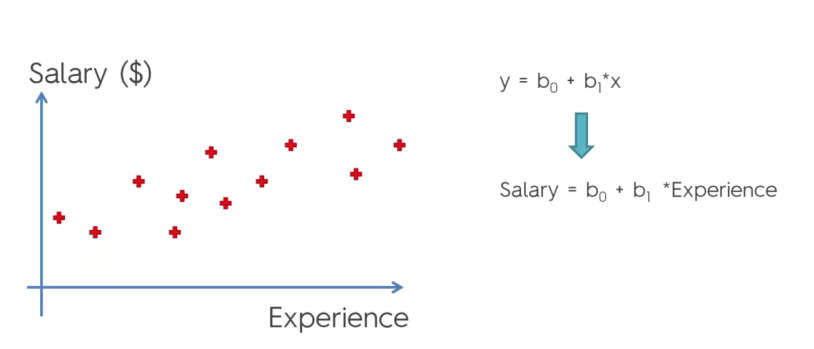


Chẳng hạn như lương sẽ phụ thuộc vào số kinh nghiệm. Khi ta làm việc lâu, lương sẽ tăng lên do kinh nghiệm này một nâng lên

Chiếu từ số năm kinh nghiệm lên điểm dữ liệu rồi từ điểm dữ liệu chiếu qua trục tung ta thấy được số lương tương ứng



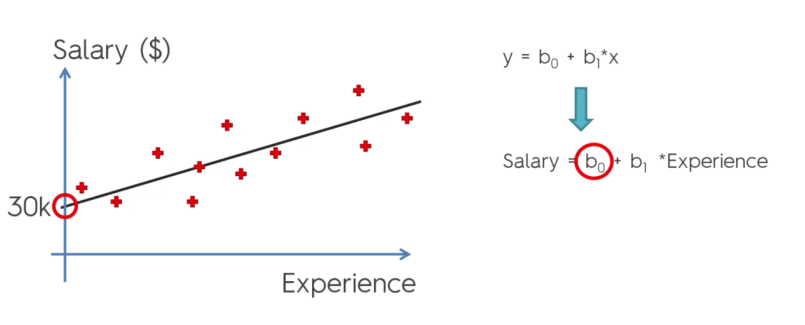
Trong trường hợp này thì Salary sẽ là và Experience là , số lương sẽ phụ thuộc vào số năm kinh nghiệm. Experience là biến độc lập và Salary là biến phụ thuộc



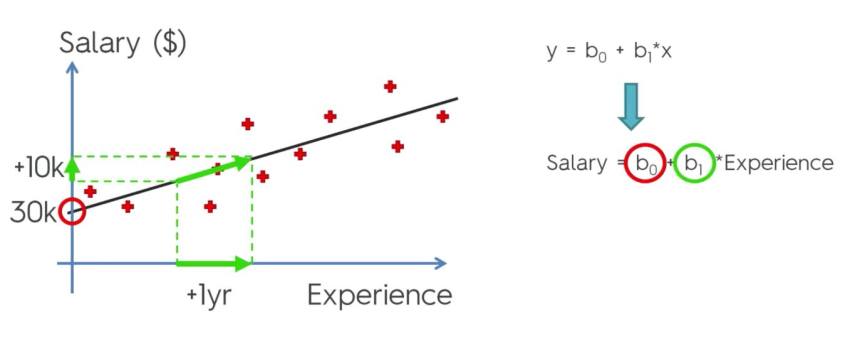
Thuật toán sẽ tìm vô số các đường thẳng và nó tìm đến khi nào khoảng cách giữa các điểm dữ liệu thật đến đường thẳng (đường thẳng dự đoán) là bé nhất.



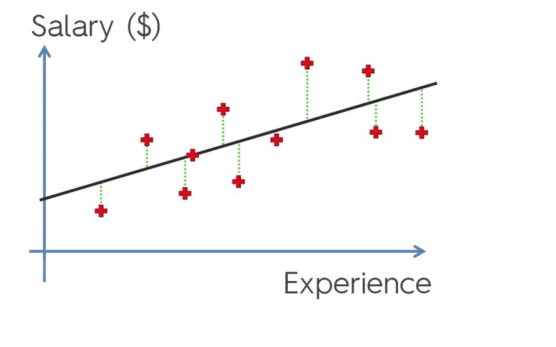
Hằng số là nơi mà đường thẳng dự đoán cắt trục tung (ở đây là số lương). Trong trường hợp này đường thẳng dự đoán cắt trục tung ở điểm khoanh đỏ (30k $/năm)



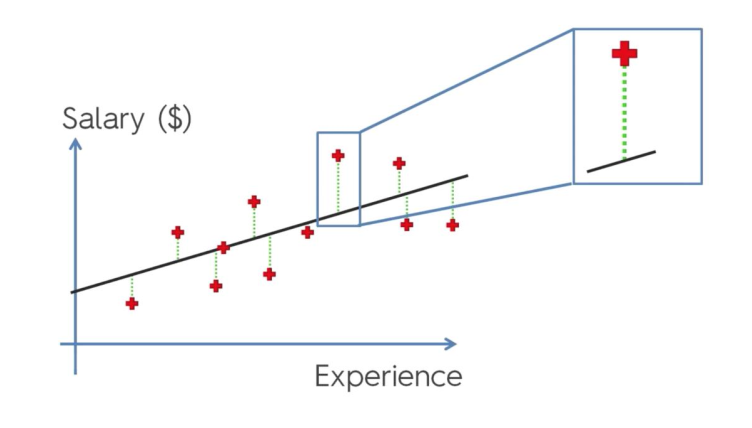
Còn là hệ số của phương trình, ở đây là độ dốc của đường thẳng. Trong biểu đồ minh họa bên dưới, thông qua độ dốc ta có thể thấy rằng trong khoảng thời gian một năm (+1 yr), đối chiếu lên trục tung thông qua phương trình ta tìm được khoảng tăng của Salary là +10k $

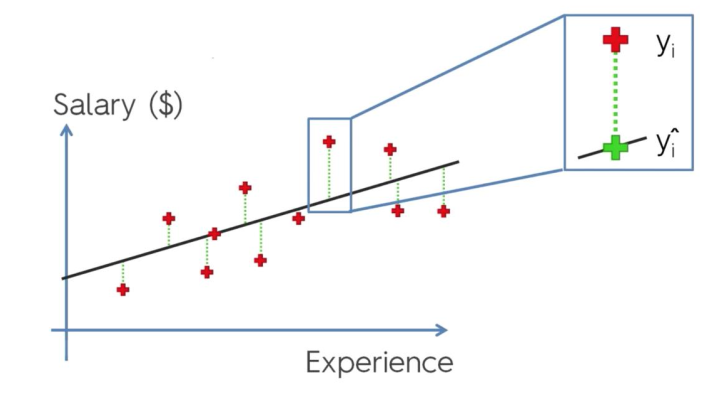


Từ các điểm dữ liệu thật vẽ đường thẳng song song với trục tung về phía đường thẳng được thuật toán tạo ra.



Khoảng cách từ điểm dữ liệu đến đường thẳng được gọi là phương sai

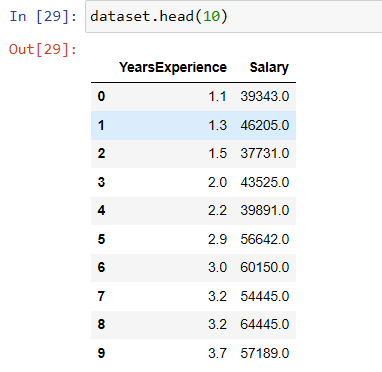




* + 1. **Ứng dụng**

Tìm hiểu thuật toán trên bộ dữ liệu tương ứng

Tập dữ liệu gồm 30 mẫu, cột đầu là số năm kinh nghiệm (Years Experience) và cột theo sau là lương (Salary) dựa trên số năm kinh nghiệm tương ứng. Số năm kinh nghiệm là biến độc lập, còn lương là biến phụ thuộc



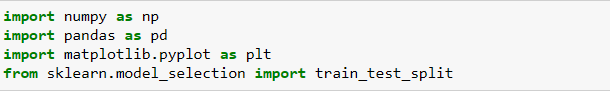
Nhìn vào tập dữ liệu, ta nhận xét được số năm kinh nghiệm tăng đồng nghĩa với việc số lương cũng tăng theo, hai giá trị này tăng theo chiều tuyến tính và tương quan với nhau

Trực quan hóa bộ dữ liệu bằng thư viện matplotlib của Python:



Dữ liệu tăng dần tuyến tính, theo chiều tăng của số năm kinh nghiệm, mức lương sẽ tăng theo

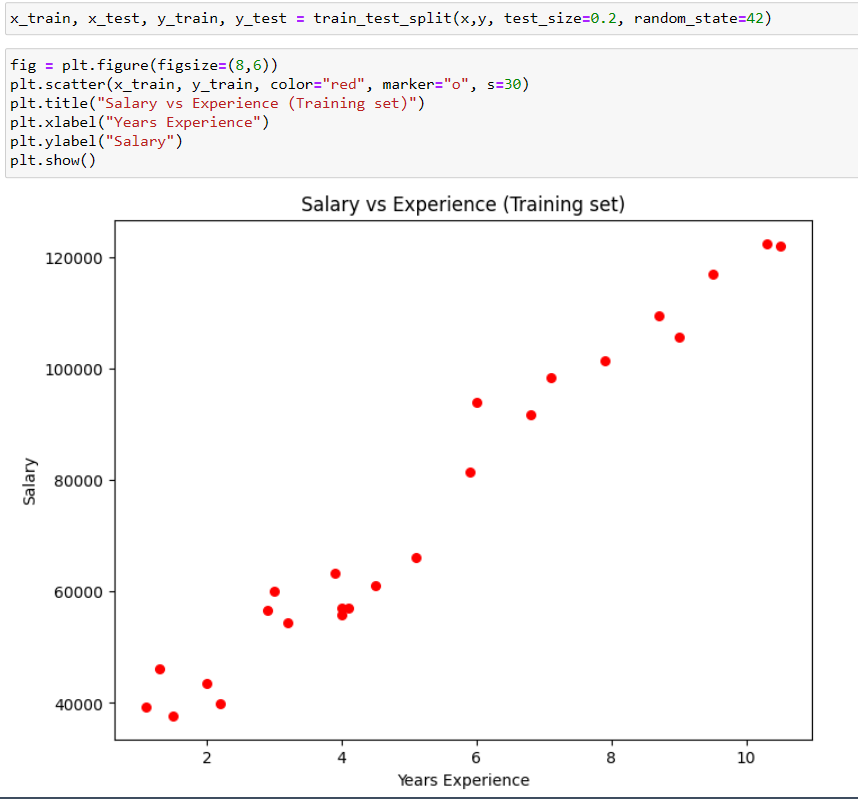
Đầu tiên, cần phải import các thư viện cần sử dụng



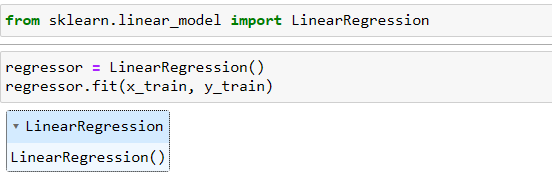
Đọc file dữ liệu vào, file dữ liệu ở đây là “Salary\_Data.csv”. Tách ra thành hai trường, trường dữ liệu x và y, trong đó x là số năm kinh nghiệm còn y là số lương



Tách mỗi phần dữ liệu ra làm 2 phần là train và test, tổng cộng ta có 4 phần dữ liệu: x\_train, x\_test, y\_train, y\_test. Trong đó x\_train và y\_train dùng để huấn luyện dữ liệu còn x\_test, y\_test dùng để kiểm tra kết quả đầu ra của dữ liệu



Tiếp theo, ta sẽ tiến hành training trên tập dữ liệu đã tách ra



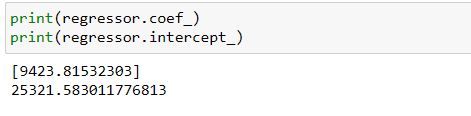
Sau đó ta trực quan hóa dữ liệu và xem tập training



Trực quan hóa trên tập dữ liệu test để xem và đối chiếu kết quả

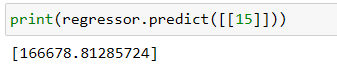


Qua đó ta tìm được hệ số và hằng số của phương trình:



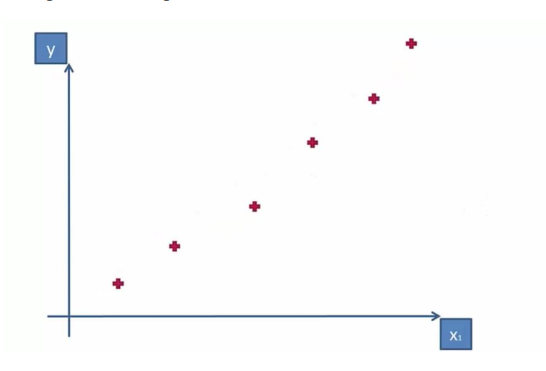
Vậy phương trình từ tập dữ liệu huấn luyện trên bộ dataset là:

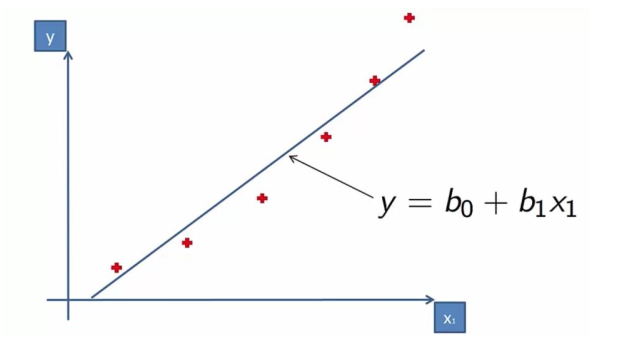
Thử dự đoán lương của nhân viên có 15 năm kinh nghiệm:



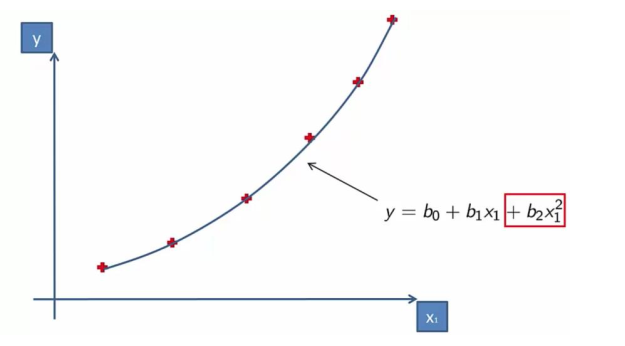
* 1. **Hồi quy đa thức**
     1. **Thuật toán**

Giả sử ta có tập dữ liệu

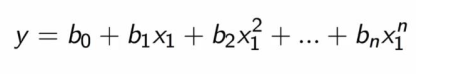
Nếu như ta sử dụng thuật toán hồi quy đơn thức



Khi đó đường thẳng hồi quy đơn thức này sẽ không fit được hoàn toàn tập dữ liệu. Chính vì vậy mà hồi quy đa thức ra đời để giải quyết trường hợp này



Công thức tổng quát của hồi quy đa thức:



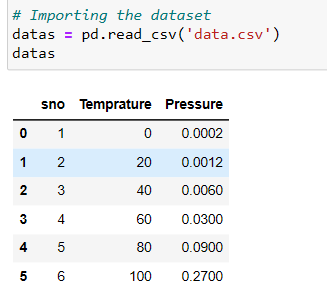
Trong đó:

: Biến phụ thuộc Hệ số hồi quy

: Hằng số hồi quy : Biến độc lập

* + 1. **Ứng dụng**

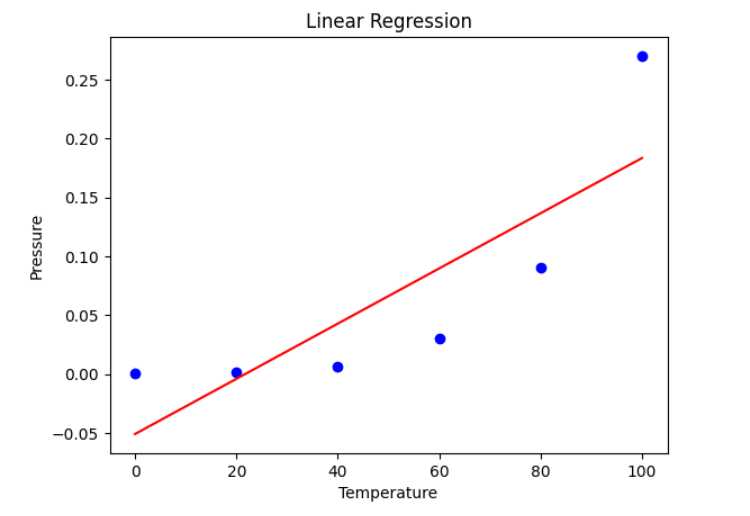
Dự đoán áp suất từ dữ liệu về nhiệt độ



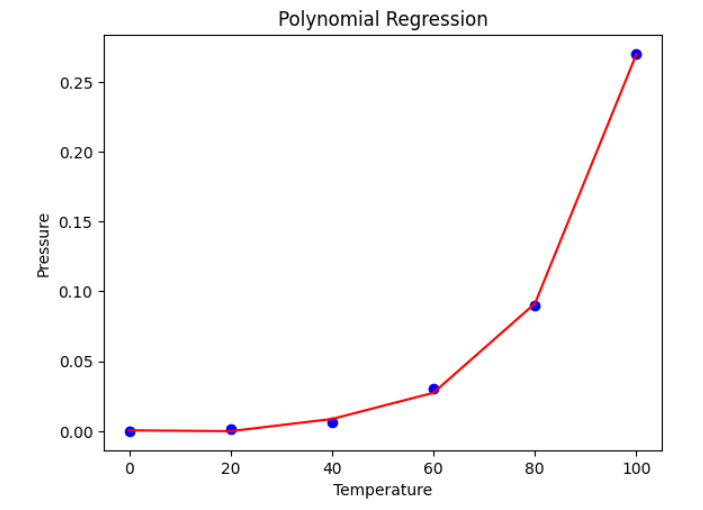
Chia tập dữ liệu làm 2 phần



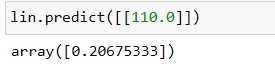
Train bộ dữ liệu khi sử dụng hồi quy đơn thức, ta được:



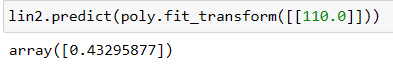
Khi sử dụng hồi quy đa thức, ta có kết quả:



Kết quả khi sử dụng hồi quy đơn thức:



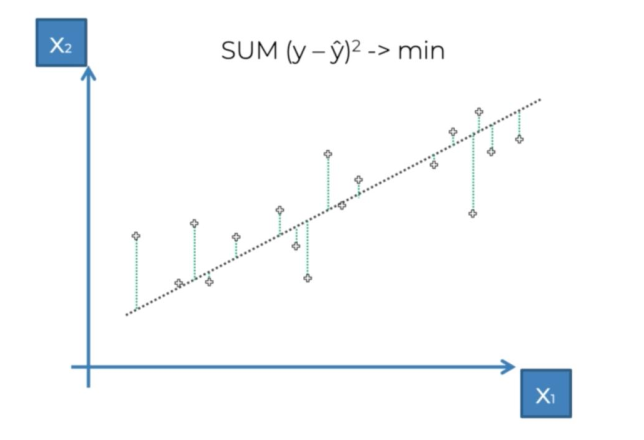
Kết quả khi sử dụng hồi quy đa thức:



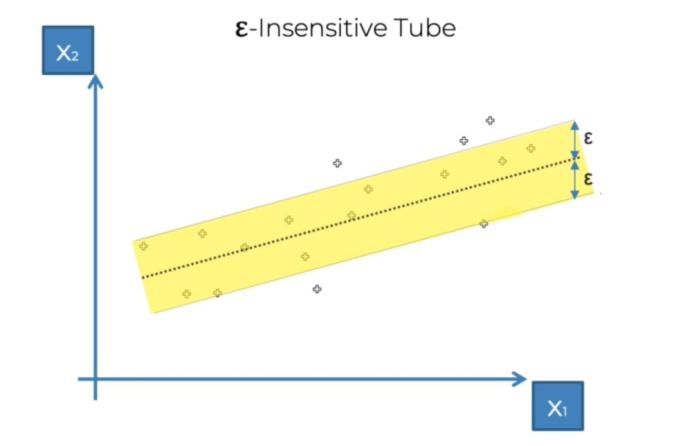
* 1. **Support Vector Regression:**
     1. **Thuật toán:**

Mô hình Support Vector Regression (SVR) là mô hình với cơ chế hồi quy của mô hình Support Vector Machine (SVM) - một thuật toán học máy có giám sát.

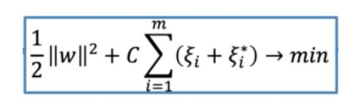
Nếu như hồi quy đơn thức sẽ vẽ một đường thẳng sao cho khoảng cách đến đường dữ liệu này là bé nhất

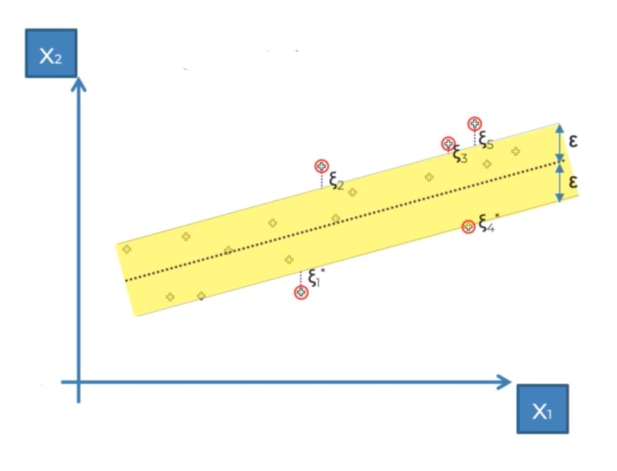


Thì Support Vector Regression (SVR) sẽ vẽ một cái tube với bề dày mỗi bên là

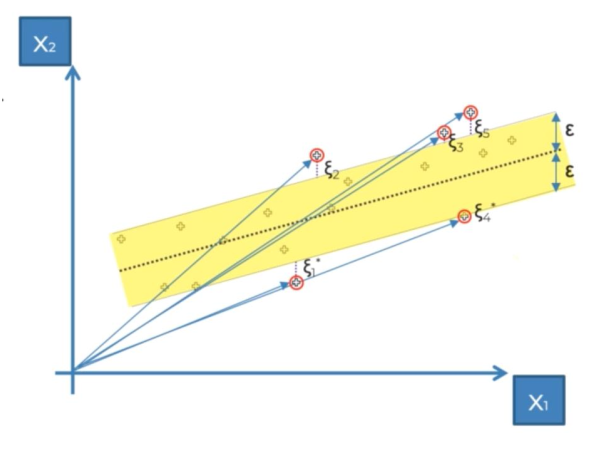


Thuật toán sẽ không quan tâm đến những điểm dữ liệu có trong cái tube mà chỉ quan tâm đến những điểm dữ liệu ở bên ngoài. Và hình dạng của cái tube này được hình thành dựa vào tổng các khoảng cách của các điểm bên ngoài sao cho tổng khoảng cách này là bé nhất theo công thức:

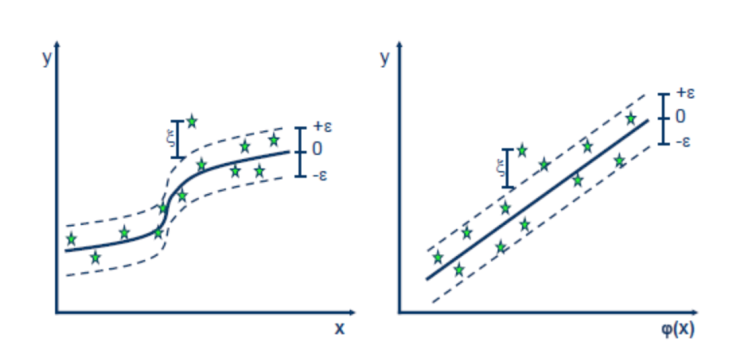




Và những điểm ở ngoài có thể tạo thành các vector. Các vector này có vai trò hình thành nên hình dạng của tube

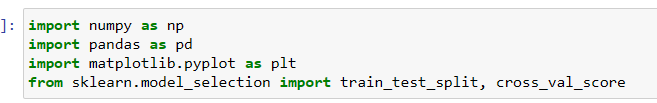


SVR được chia làm 2 loại là Linear và Non-Linear. Linear SVR sẽ được dùng cho những tập dữ liệu tuyến tính, còn những tập dữ liệu không tuyến tính thì dùng non-linear SVR

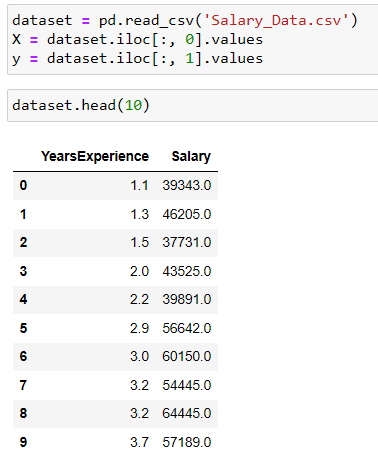


1. *Non-Linear SVR b) Linear SVR* 
   * 1. **Ứng dụng**

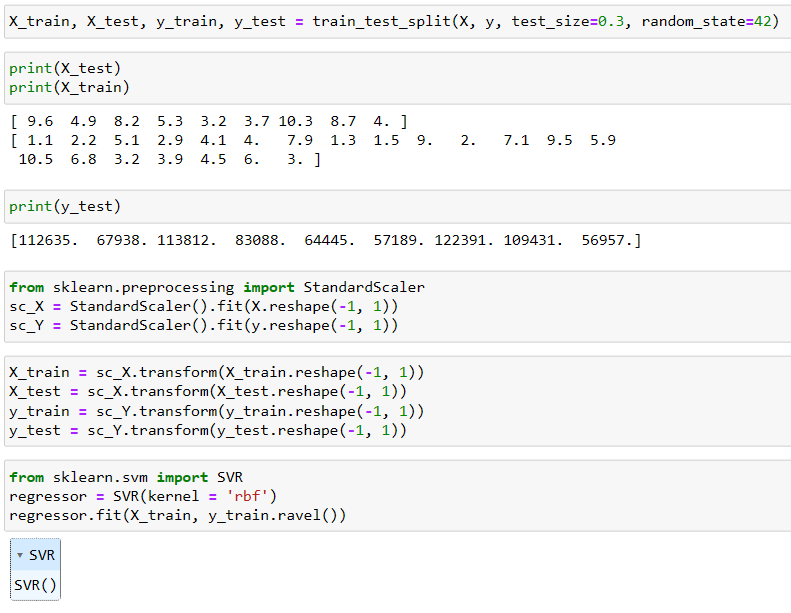
Đầu tiên ta khai báo các thư viện cần sử dụng



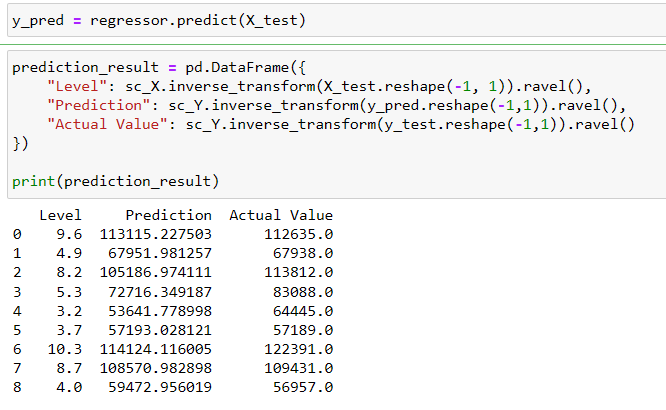
Sử dụng bộ dataset “Salary\_Data.csv”



Chia bộ dữ liệu ra thành các trường X\_train, X\_test, y\_train, y\_test và tiến hành huấn luyện:



Tiến hành dự đoán lương theo “Level”:



1. **Các thuật toán phân loại:**

#### **2.1 Logistic Regression**

##### **2.1.1 Thuật toán:**

*Logistic Regression* là một loại thuật toán *Supervised Learning* tính toán mối quan hệ giữa các feature trong input và output dựa trên hàm logistic/sigmoid.

Logistic Regression là phương pháp hồi quy thông dụng áp dụng cho các biến mục tiêu không phải là biến định lượng liên tục (continous variable). Logistic Regression được ứng dụng để dự báo và phân loại giá trị của biến mục tiêu dựa trên dữ liệu có 2 giá trị, ví dụ như phân loại khách hàng có khả năng trả nợ để cấp phát thẻ tín dụng (biến mục tiêu có 2 giá trị có hoặc không có khả năng trả nợ).

Khác với hồi quy tuyến tính, kết quả của phương trình hồi quy logistic là xác suất, dựa vào xác suất để quyết định giá trị sau cùng của biến y. Đối với hồi quy logistic, biến y có 2 giá trị (có hoặc không, thành công hay thất bại,…). Thông thường theo thông lệ thì sẽ gán y = 0 cho các kết quả “không” và y = 1 cho các kết quả ngược lại.

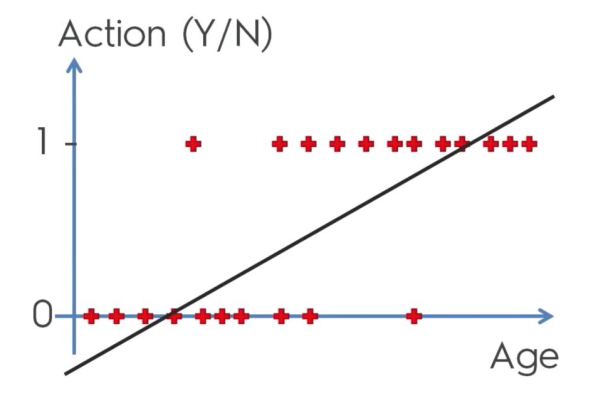
Với p là biến phụ thuộc, xác suất khả năng y xảy ra 0 hoặc 1 (chịu ảnh hưởng của biến x) là biến chúng ta sẽ dự báo, x là biến độc lập (biến tác động lên biến phụ thuộc). là giá trị ước lượng cho p khi x đạt giá trị 0, dùng để xác định giá trị trung bình của p tăng hay giảm khi x tăng.

Logistic Regression được ứng dụng trong nhiều ngành và lĩnh vực khác nhau như:

* Dự báo phân loại email spam hay không spam
* Dự báo khả năng trả nợ hay không trả nợ của khách hàng
* Dự báo tình trạng khối u ung thư là ác tính hay lành tính trong y học

Giả sử như có một người làm trong công ty và người này gửi email để đề nghị khách hàng mua sản phẩm của bên họ (quần áo, mỹ phẩm,…), và họ nhận được phản hồi 2 kết quả là “Trả lời” và “Không trả lời” dựa trên độ tuổi.

Giá trị output đầu ra chỉ có giá trị 0 (Không trả lời) và 1 (Trả lời) và được phân bố như hình

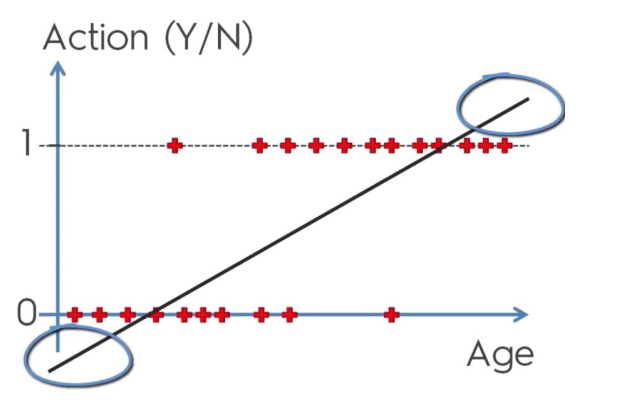


Ở đây, trục tung sẽ biểu thị cho kết quả dự đoán còn trục hoành là độ tuổi, dựa vào độ tuổi dự đoán xem khách hàng mà ta đề nghị có phản hồi email hay không.

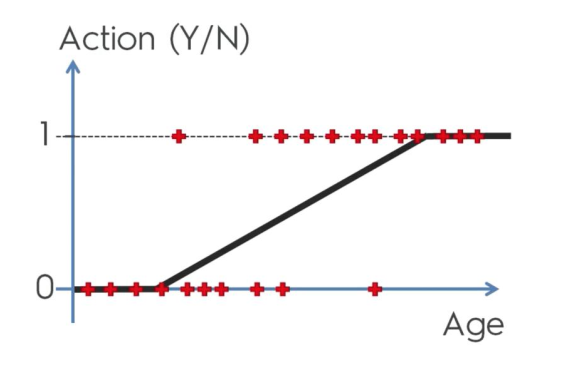
Nếu như ta dùng thuật toán Linear Regression, đường thẳng này sẽ không thể fit được tất cả các điểm dữ liệu, và đường thẳng khi dùng thuật toán hồi quy đơn thức chỉ có giá trị nằm trong đoạn [0, 1] là hợp lệ



Phần đường thẳng nằm ngoài 2 giá trị 0 và 1 (theo trục tung) sẽ không hợp lệ



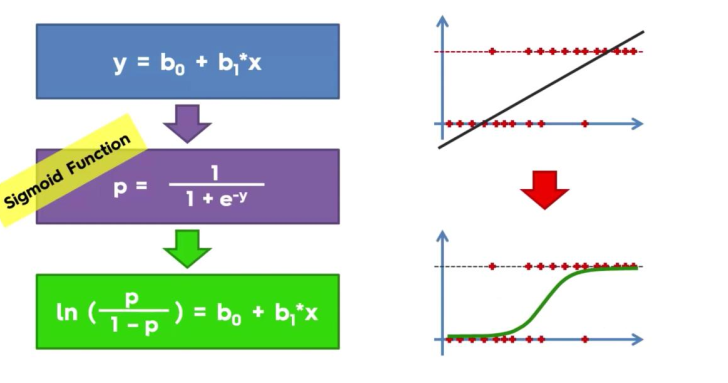
Ta thay thế 2 phần đường thẳng trên bằng 2 đường thẳng trên trục tung như hình bên dưới sẽ giúp fit với tập dữ liệu



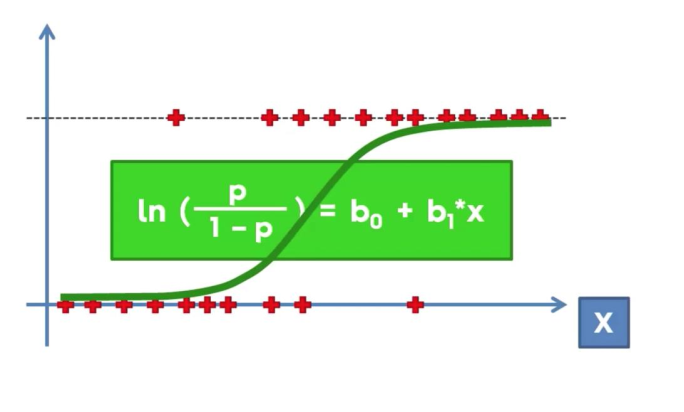
Ta có hàm sigmoid :

Công thức hồi quy đơn thức:

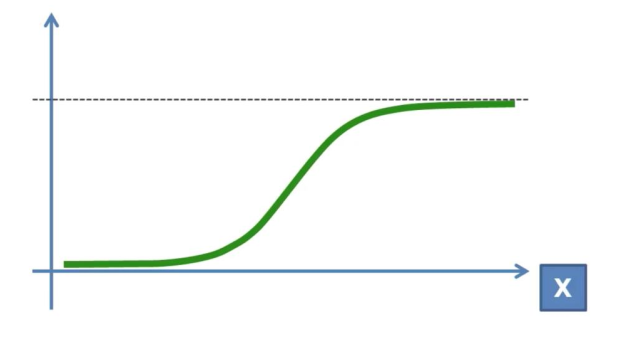
Lấy 2 biểu thức trên thay thế vào nhau ta được công tức của hồi quy logistic

****

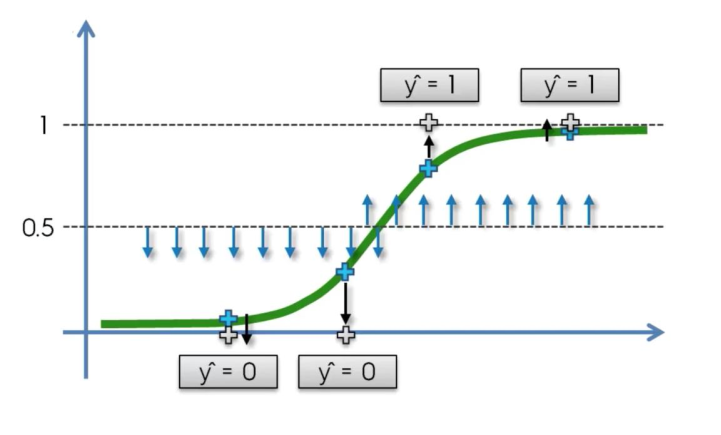
Dựa vào tập dữ liệu quan sát và dùng công thức của Linear Regression ta tìm được đường màu xanh fit nhất với tập dữ liệu



Vậy đường màu xanh là Logistic Regression được sử dụng để dự đoán khả năng có hay không của hành động, sự việc nào đó.



**Lưu ý:** Thuật toán chỉ tìm ra output p là 0 hoặc 1. Thuật toán sẽ vẽ đường thẳng trung vị có p = 0.5, những giá trị được chiếu qua trục tung có p < 0.5 sẽ quy về 0, những giá trị có p 0.5 sẽ quy về 1

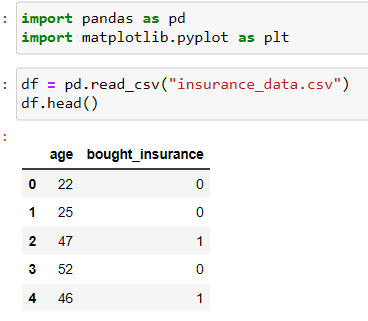


Mô hình logistic được sử dụng để dự đoán xác suất của một lớp hoặc sự kiện nhất định (đạt hay không đạt, thắng hay thua,…). Nó có thể được mở rộng để áp dụng với một số lớp sự kiện như xác định xem một hình ảnh có chứa mèo, chó, sư tử,… Mỗi đối tượng được phát hiện trong hình ảnh sẽ được gán một giá trị từ 0 đến 1, với tổng xác suất của hai biến đối lập là 1.

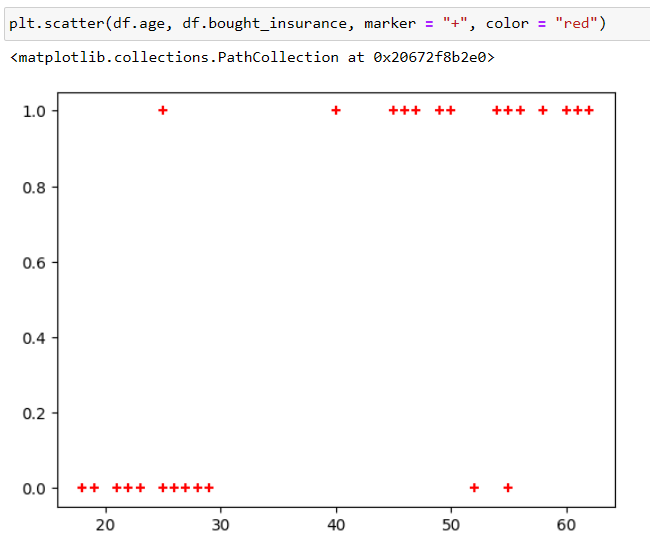
##### **2.1.2 Ứng dụng:**

Chúng ta sẽ dự đoán xem một người có quyết định mua bảo hiểm nhân thọ hay không dựa vào độ tuổi của họ sử dụng thuật toán Linear Regression. Ta sẽ sử dụng bộ dữ liệu “insurance\_data.csv” cho ví dụ này

Đầu tiên, ta khai báo các thư viện cần thiết và đọc file dataset

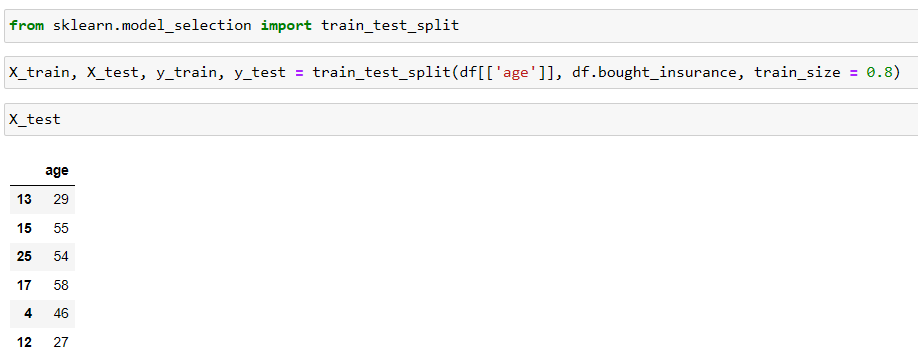


Trực quan hóa dữ liệu bằng matplotlib

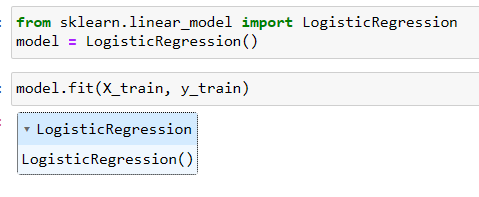


Ta có thể thấy dữ liệu tạo thành hình dạng một hàm Sigmoid, một hàm đặc trưng của thuật toán Logistic Regression

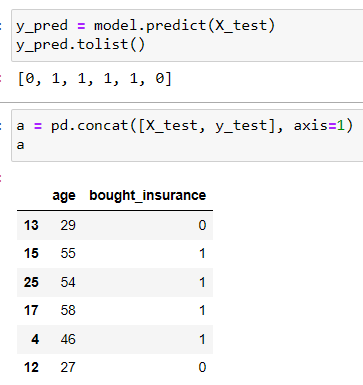
Ta tiến hành bộ dữ liệu thành các tập Train và Test



Sau đó ta sẽ bắt đầu huấn luyện mô hình



Kiểm tra lại trên tập Test



#### **2.2. K – Nearest Neighbors:**

##### **2.2.1. Thuật toán:**

Phương pháp này tốn nhiều công sức khi đưa ra các tập huấn luyện lớn. Tuy nhiên ngày càng trở nên phổ biến khi khả năng tính toán ngày càng tăng

Được sử dụng rộng rãi trong lĩnh vực nhận dạng mẫu và ước tính thống kê

*K – Nearest Neighbors* 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 đó 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 Neighbors* có thể áp dụng được vào cả hai loại của bài toán Supervised learning là Classification và Regression

Trong bài toán *Regression,* đầu ra của một điểm dữ liệu sẽ bằng chính đầu ra của điểm dữ liệu đã biết gần nhất (với K = 1), hoặc là trung bình có trọng số của đầu ra của những điểm gần nhất, hoặc bằng một mối quan hệ dựa trên khoảng cách tới các điểm gần nhất đó

Trong bài toán *Classification,* label của một điểm dữ liệu mới được suy ra trực tiếp từ K điểm dữ liệu gần nhất trong training set.

|  |  |
| --- | --- |
| **Ưu điểm** | **Nhược điểm** |
| * Độ phức tạp tính toán của quá trình training là bằng 0 * Việc dự đoán kết quả của dữ liệu mới rất đơn giản * Không cần giả sử gì về phân phối của các class | * Nhạy cảm với nhiễu khi K nhỏ * Thời gian tính toán tăng theo độ lớn của bộ dữ liệu training set và test set |

**Các bước hoạt động:**

Bao gồm 6 bước:

Bước 1: Chọn số K các điểm lân cận

Bước 2: Tính toán khoảng cách Euclidean của K điểm lân cận đó

Bước 3: Lấy các điểm lân cận gần nhất cho mỗi khoảng cách Euclidean được tính

Bước 4: Trong số K điểm lân cận đó, đếm số điểm dữ liệu cho mỗi loại

Bước 5: Gán điểm dữ liệu mới vào loại mà số điểm lân cận là lớn nhất

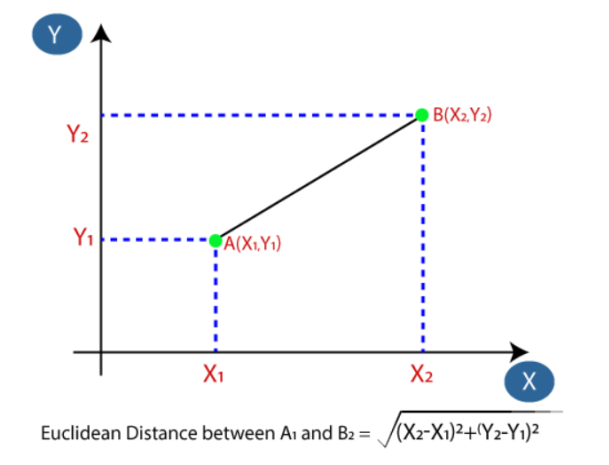
Bước 6: Hoàn tất mô hình

**Ví dụ:**

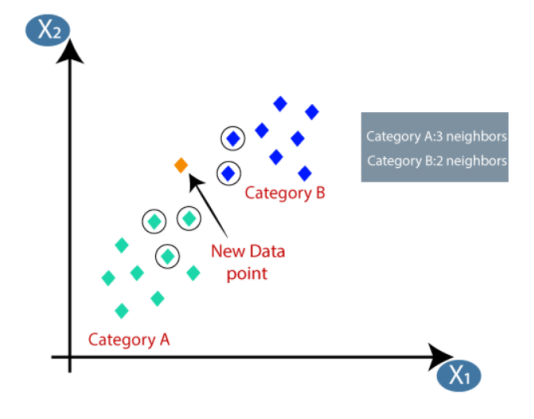
Giả sử chúng ta có điểm dữ liệu mới và ta cần đặt chúng vào nhóm theo yêu cầu theo như hình:



* Đầu tiên, ta sẽ chọn số lượng điểm lân cận, ở đây chọn K = 5
* Kế đến, chúng ta sẽ tính khoảng cách Euclidean giữa các điểm dữ liệu đó



* Bằng cách tính khoảng cách Euclidean, ta tìm được các điểm lân cận gần nhất, 3 điểm cho loại A và 2 điểm cho loại B:

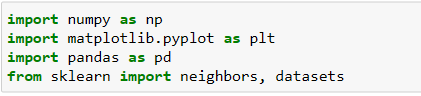


* Như ta thấy, có 3/5 điểm lân cận với điểm dữ liệu cần xét là loại A, chính vì vậy nên điểm dữ liệu cũng thuộc vào loại A

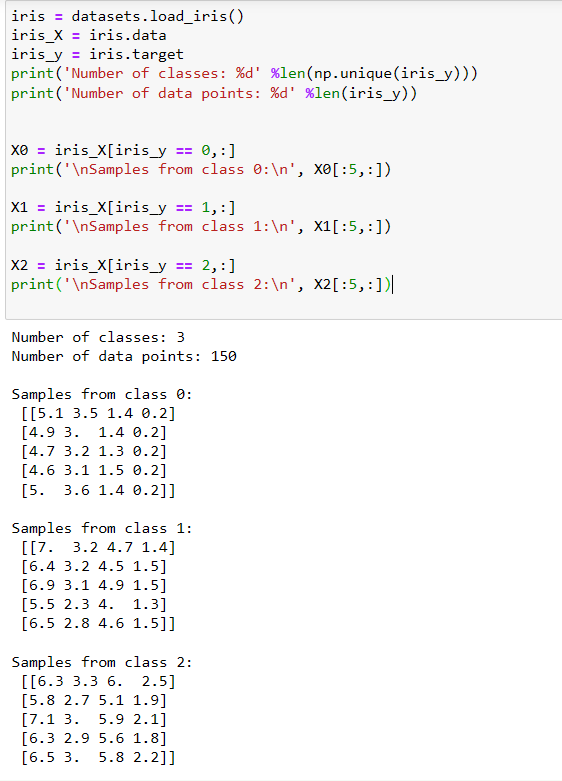
##### **2.3.2. Ứng dụng:**

Chúng ta sẽ sử dụng bộ dữ liệu [Iris flower dataset](https://en.wikipedia.org/wiki/Iris_flower_data_set) và tách ra thành 2 phần training set và test set. Thuật toán KNN sẽ dựa vào thông tin ở training set để dự đoán xem mỗi dữ liệu trong test set tương ứng với loại hoa nào.

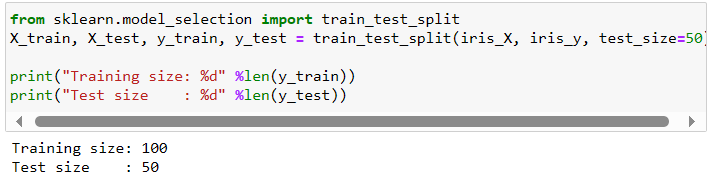
Trước tiên ta cần khai báo thư viện:



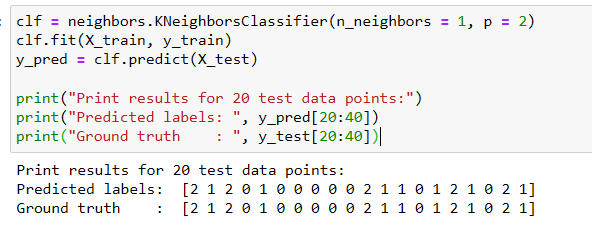
Tiếp theo, ta load dữ liệu và hiện thị vài dữ liệu mẫu



Giả sử ta muốn dùng 50 điểm dữ liệu cho test set, 100 điểm còn lại cho training set:

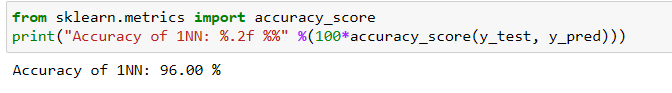


Xét với trường hợp đơn giản k = 1, tức với mỗi điểm test data, ta chỉ xét 1 điểm training data gần nhất và lấy label của điểm đó để dự đoán cho điểm test này:



Ta có thể thấy label dự đoán cho 20 điểm dữ liệu này hoàn toàn giống với label thật của test data.

Đánh giá độ chính xác của mô hình:

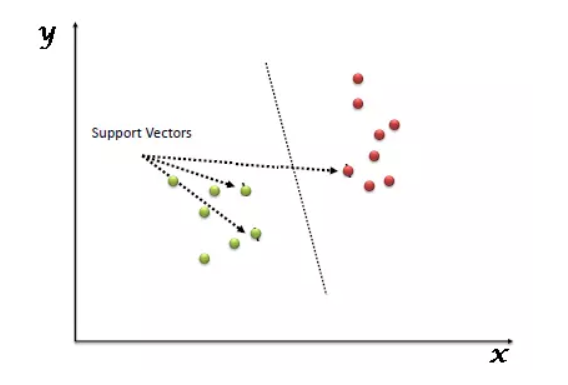


#### **2.3. Support Vector Machine**

##### **2.3.1. Thuật toán**

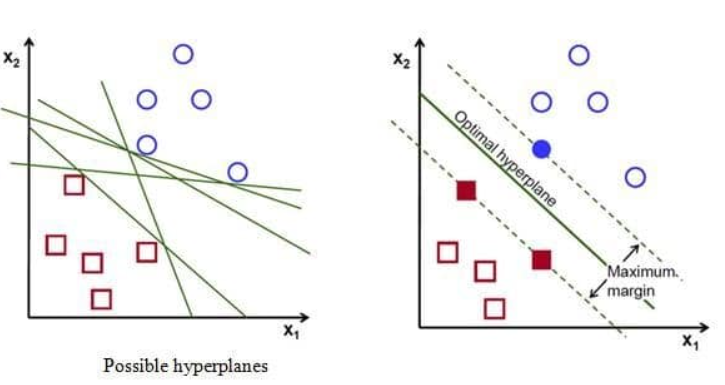
**Khái niệm:**

*SVM* là một thuật toán giám sát, nó có thể sử dụng cho cả việc phân loại và hồi quy. Tuy nhiên nó được sử dụng chủ yếu cho việc phân loại

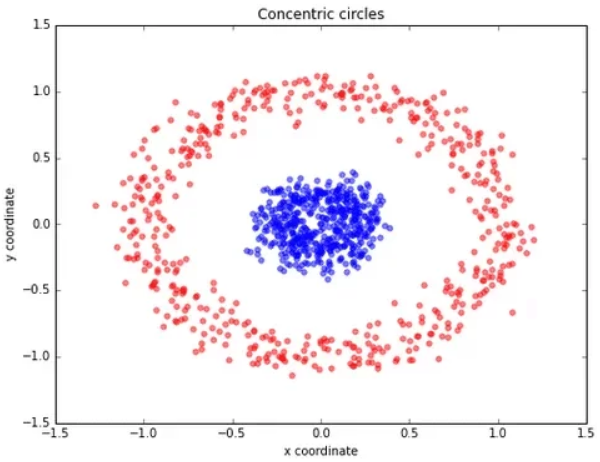


*Support Vectors* hiểu đơn giản là các đối tượng trên đồ thị tọa độ quan sát, *Support Vector Machine (SVM)* sẽ xác định một *siêu phẳng* để chia hai lớp tốt nhất. Trong một mặt phẳng (2 chiều) thì *siêu phẳng* là một đường thẳng, trong không gian 3 chiều thì *siêu phẳng* là một mặt phẳng

Để phân chia hai lớp dữ liệu, rõ ràng là có rất nhiều siêu phẳng có thể làm được, dù vậy, mục tiêu của ta là tìm ra siêu phẳng có lề rộng nhất (tức là có khoảng cách tới các điểm của hai lớp là lớn nhất)



**Vấn đề: Nếu dữ liệu không thể phân chia tuyết tính thì sao?**



Nhìn vào hình ảnh trên, dữ liệu được phân tách phi tuyến nên chúng ta không thể vẽ một đường thẳng đề phân loại các điểm dữ liệu đỏ và xanh. Để giải quyết vấn đề này, có 2 giải pháp:

1. Lề mềm (Soft margin)

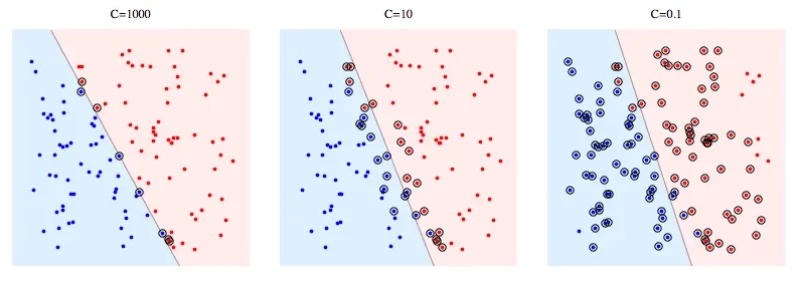
Thuật toán này cho phép SVM mắc một số lỗi nhất định và giữ cho lề càng rộng càng tốt để các điểm khác vẫn có thể được phân loại chính xác. Nói một cách khác, nó cân bằng giữa việc phân loại sai và tối đa hóa lề

Có 2 kiểu phân loại sai có thể xảy ra:

* + Dữ liệu nằm ở đúng bên nhưng phạm vào lề
  + Dữ liệu nằm ở sai bên

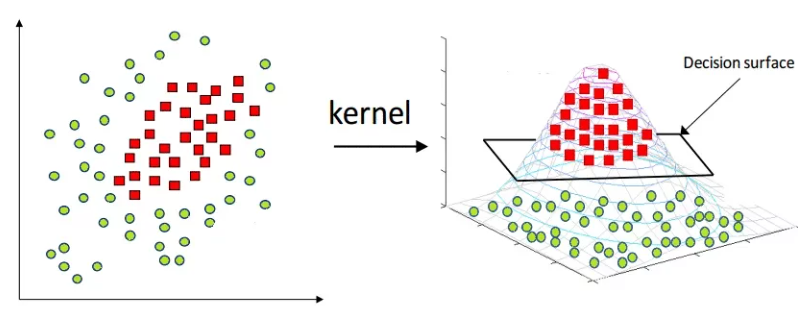
**Mức độ chấp nhận lỗi:**

Mức độ chấp nhận lỗi hay còn gọi là tham số phạt (C) là một siêu tham số quan trọng trong SVM. C càng lớn có nghĩa SVM càng bị phạt nặng khi thực hiện phân loại sai. Do đó, lề càng hẹp và càng ít vectơ hỗ trợ được sử dụng



1. Thủ thuật Kernel (Kernel tricks)

Một *Kernel* là một hàm ánh xạ dữ liệu từ không gian ít chiều hơn sang không gian nhiều chiều hơn, từ đó ta tìm được siêu phẳng phân tách dữ liệu



***Ứng dụng trong thực tế:***

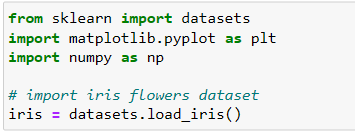
SVM là một phương pháp hiệu quả cho bài toán phân lớp dữ liệu. Nó là một công cụ đắc lực cho các bài toán về xử lý ảnh, phân loại văn bản, nhận diện khuôn mặt.

Một yếu tố làm nên hiệu quả của SVM là việc sử dụng Kernel function khiến cho các phương pháp chuyển không gian trở nên linh hoạt hơn.

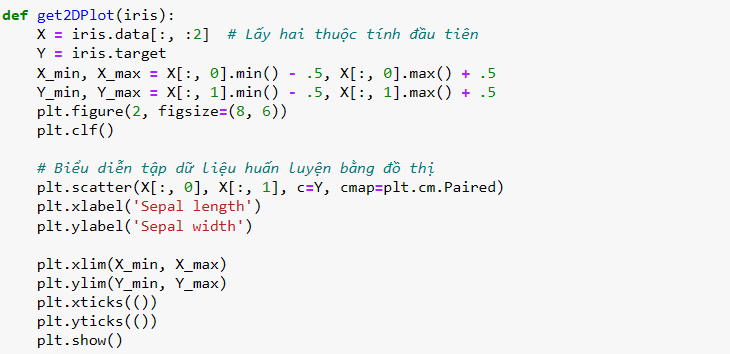
|  |  |
| --- | --- |
| **Ưu điểm** | **Nhược điểm** |
| 1. *Xử lý không gian số chiều cao:* SVM là một công cụ tính toán hiệu quả trong không gian chiều cao, trong đó đặc biệt áp dụng các bài toán phân loại văn bản và phân tích quan điểm nơi chiều có thể lớn 2. *Tiết kiệm bộ nhớ:* Do chỉ có một tập 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 3. *Tính linh hoạt – phân lớp thường là phi tuyến:* Khả năng áp dụng Kernel mới cho phép linh động giữa các phương pháp tuyến tính và phi tuyến từ đó cho hiệu suất phân loại lớn hơn | 1. *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 kém 2. *Chưa thể hiện rõ tính xác suất:* Việc phân lớp của SVM chỉ là việc cố gắng tách hai 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 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. |

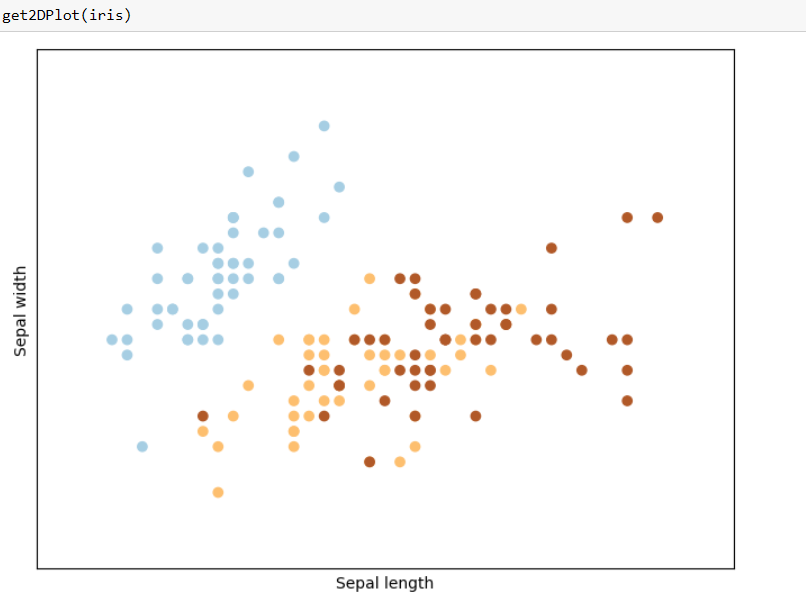
##### **2.3.2. Ứng dụng**

Chúng ta sẽ sử dụng bộ dữ liệu [Iris flower dataset](https://en.wikipedia.org/wiki/Iris_flower_data_set), với mỗi một mẫu hoa bao gồm 4 thuộc tính là chiều dài và chiều rộng của đài hoa và cánh hoa



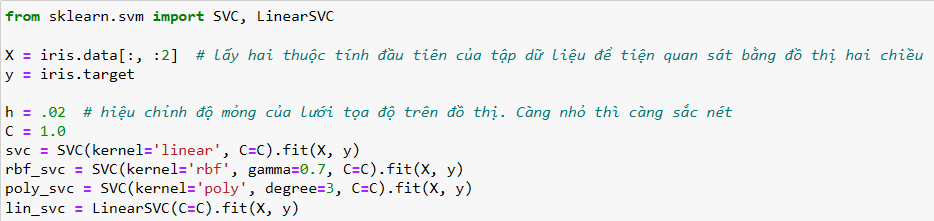
Mỗi điểm dữ liệu bao gồm 4 thuộc tính như đã nói ở trên. Tuy nhiên để biểu diễn trong đồ thị 2 chiều cần chọn 2 thuộc tính để biểu diễn, ở đây ta sẽ chọn chiều dài và chiều rộng của đài hoa



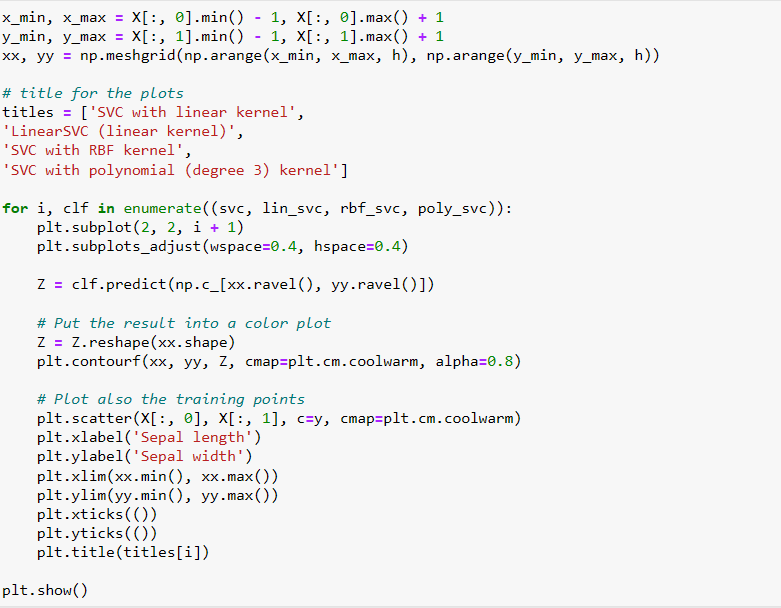


Phân lớp sử dụng SVM với các Kernel khác nhau, bao gồm:

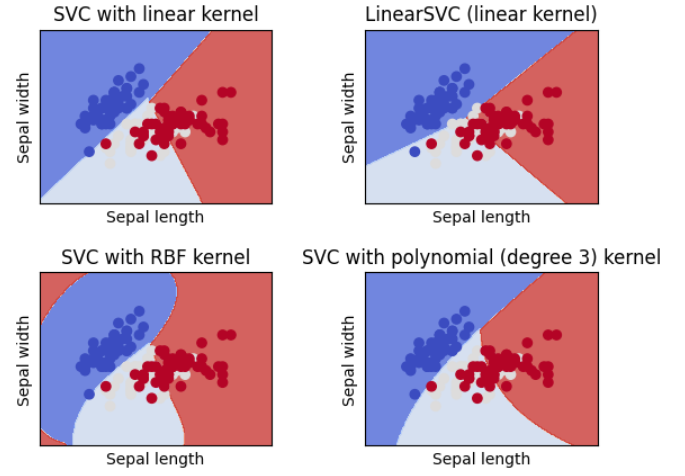
* + SVC with linear Kernel
  + LinearSVC
  + SVC with RBF Kernel
  + SVC with polynomial (degree 3) Kernel



Biểu diễn các mô hình thu được



Sau khi biểu diễn các đồ thị, ta có thể so sánh hiệu quả phân lớp của các Kernel khác nhau:



#### **2.4 Random Forest:**

##### **2.4.1. Thuật toán:**

**Khái niệm:**

Random Forests là thuật toán học có giám sát (Supervised Learning). Nó có thể được sử dụng cho cả phân loại và hồi quy. Nó cũng là thuật toán linh hoạt và dễ sử dụng nhất.

Random Forests tạo ra cây quyết định trên các mẫu dữ liệu được chọn ngẫu nhiên, được dự đoán từ mỗi cây và chọn giải pháp tốt nhất bằng cách bỏ phiếu.

Random Forests có nhiều ứng dụng, chẳng hạn như công cụ đề xuất, phân loại hình ảnh và lựa chọn tính năng. Nó có thể được sử dụng để phân loại các ứng viên cho vay trung thành, xác định hoạt động gian lận và dự đoán các bệnh.

**Thuật toán:**

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 mình sẽ làm như sau:

1. Lấy ngẫu nhiên n dữ liệu từ bộ dữ liệu với kỹ 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 kỹ thuật này thì tập n dữ liệu mới của mình có thể có những dữ liệu bị trùng nhau.
2. 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
3. Dùng thuật toán Decision Tree để xây dựng cây quyết định với bộ dữ liệu ở bước 2.

Do quá trình 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,* mình hay 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 có thể tách

**Ưu, nhược điểm:**

1. *Ưu điểm:*

Thuật toán có thể được sử dụng trong cả hai vấn đề phân loại và hồi quy. Random Forests cũng có thể xử lý các giá trị còn thiếu. Có hai cách để xử lý các giá trị này:

* Sử dụng các giá trị trung bình để thay thế các biến liên tục
* Tính toán mức trung bình gần kề của các giá trị bị thiếu

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

1. *Nhược điểm:*

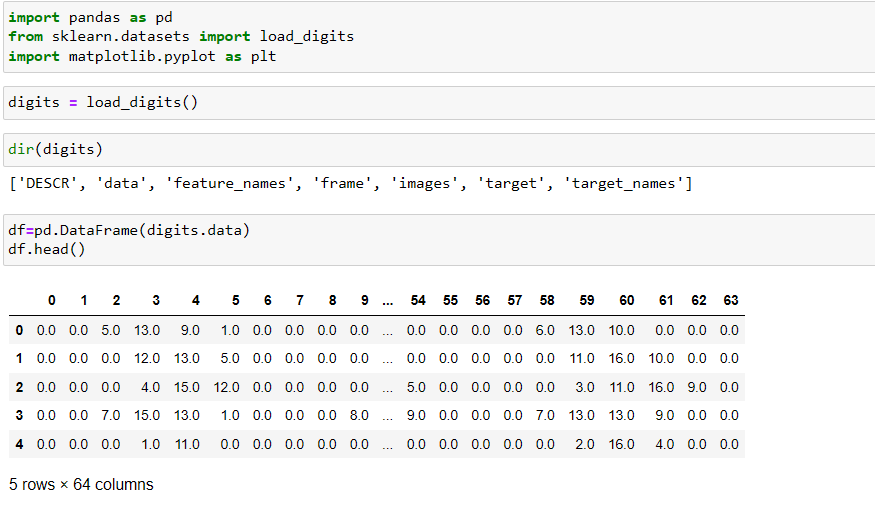
*Random Forest* chậm tạo ra dự đoán bởi nó có nhiều cây quyết định. Bất cứ khi nào nó đưa ra dự đoán, tất cả các cây đều phải đưa ra dự đoán cho cùng một đầu vào cho trước và sau đó thực hiện bỏ phiếu trên đó. Toàn bộ quá trình này tốn khá nhiều thời gian.

Mô hình khó hiểu hơn so với *Decision Tree,* nơi bạn có thể dễ dàng đưa ra quyết định bằng cách đi theo đường dẫn trong cây.

##### **2.4.2 Ứng dụng:**

Chúng ta sẽ sử dụng bộ dữ liệu digits có sẵn trong module datasets của thư viện sklearn để dự đoán chữ số viết tay.

Đầu tiên ta sẽ load dataset lên:

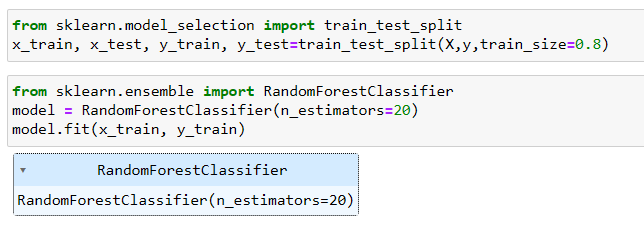


Sử dụng thư viện matplotlib để trực quan hóa mẫu dữ liệu trong dataset:

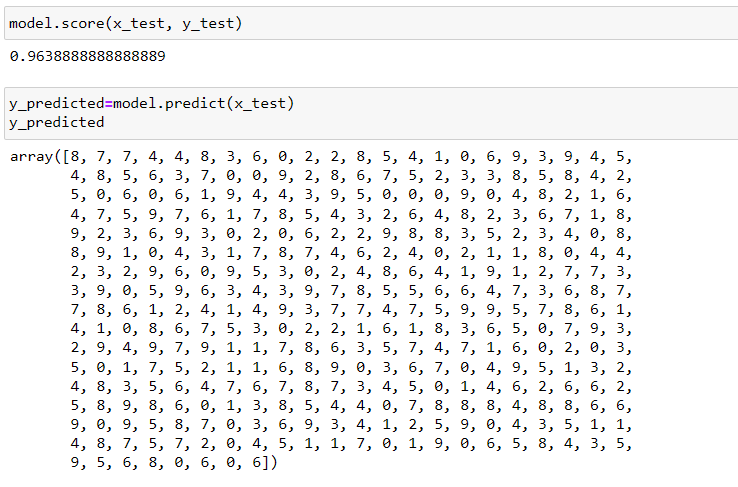


Tiến hành phân tách các tập huấn luyện và kiểm thử, sau đó huấn luyện cho model sử dụng lớp RandomForestClassifier hỗ trợ cho thuật toán Random Forest

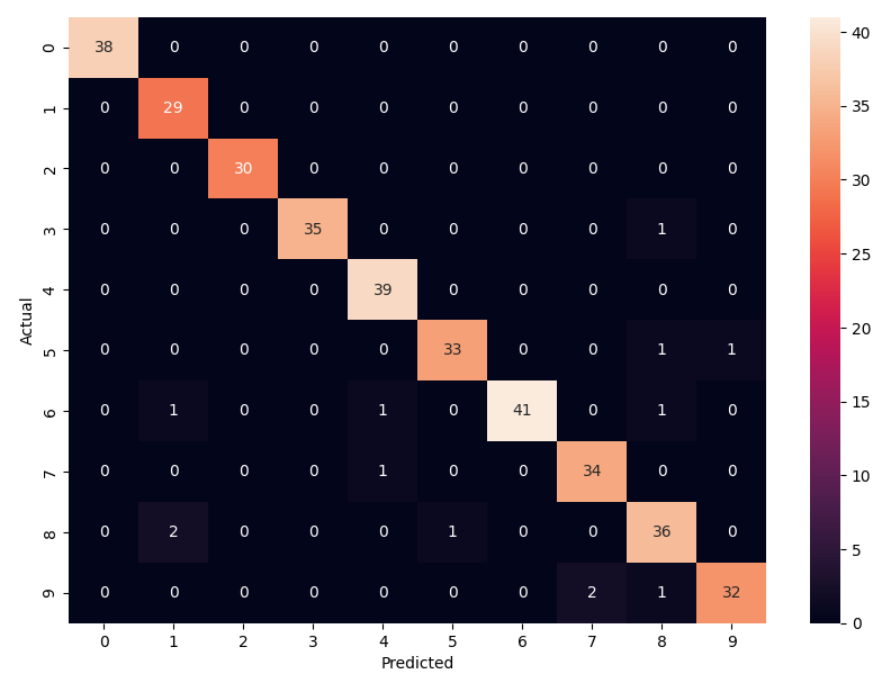




Kết quả đạt được:



Confusion Matrix:

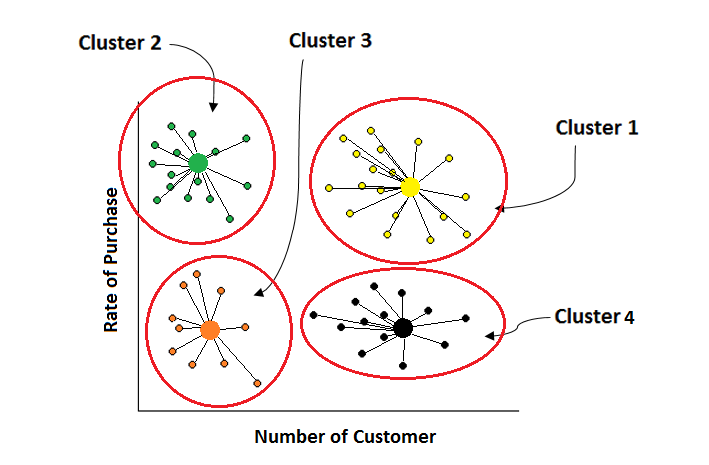


### **3. Các thuật toán phân cụm:**

#### **3.1. K-Means Clustering:**

##### **3.1.1 Thuật toán:**

Là một thuật toán trong Unsupervised Learning, K-Means Clustering là thuật toán quan trọng và được sử dụng phổ biến trong kỹ thuật Clustering. Tư tưởng chính của thuật toán K-Means là tìm cách phân nhóm các đối tượng (objects) đã cho vào K cụm (K là số các cụm được xác định trước, K nguyên dương) sao cho tổng bình phương khoảng cách giữa các đối tượng đến tâm nhóm (centroid) là nhỏ nhất.



Thuật toán:

B1: Chọn số lượng K cụm

B2: Tương ứng với số cụm, chọn những điểm trung tâm (không cần chọn trùng với những điểm có trong tập dữ liệu)

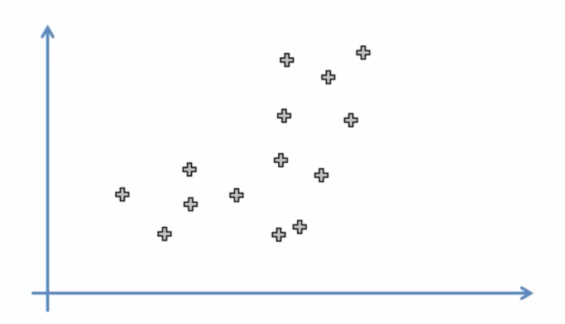
B3: Chia những điểm dữ liệu tương ứng với những điểm trung tâm (điểm dữ liệu gần điểm trung tâm nào thì điểm dữ liệu thuộc cụm trung tâm đó)

B4: Tính toán và thay đổi điểm trung tâm của mỗi cụm

B5: Chia lại những điểm dữ liệu vào chụm tương ứng. Nếu không có điểm dữ liệu nào bị chia lại thì hoàn thành thuật toán, còn không quay lại bước 4.

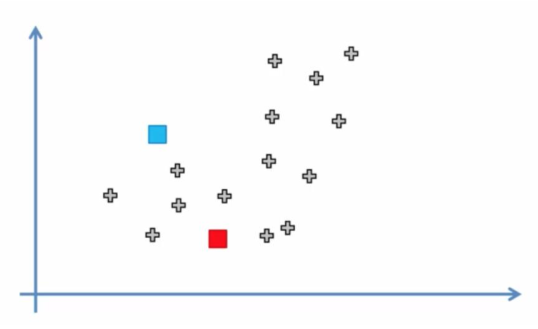
***Ví dụ:***

Tập dữ liệu như hình vẽ dưới



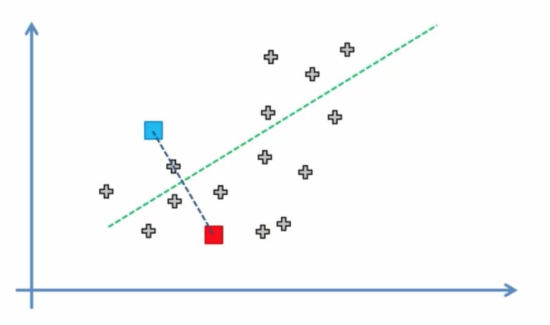
Bước đầu tiên ta chọn số cụm K = 2

Sau đó có bao nhiêu cụm thì chọn bấy nhiêu điểm trung tâm, điểm trung tâm này có thể lấy bất kỳ ở đâu, không cần thiết phải là điểm có trong tập dữ liệu

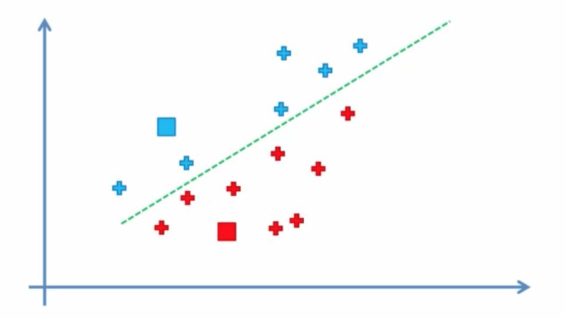


Phân chia những điểm dữ liệu dựa vào những điểm trung tâm này nó sẽ hình thành K cụm

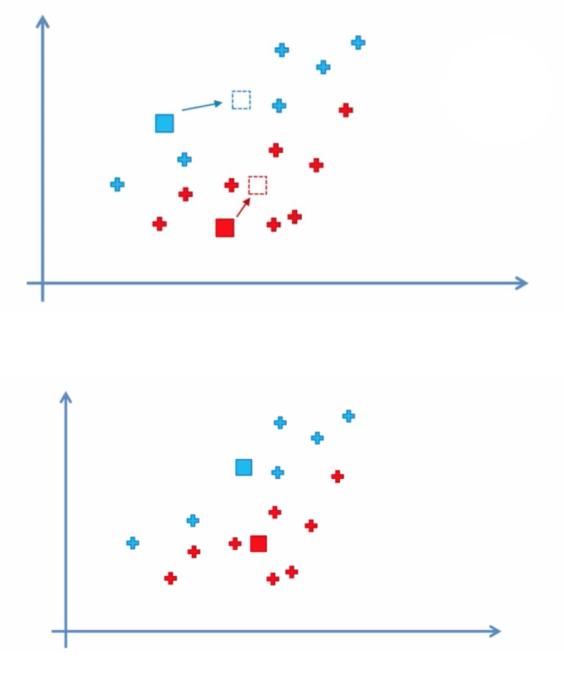
Ví dụ như lấy một điểm, xét xem điểm đó gần điểm nào hơn trong 2 điểm trung tâm xanh và đỏ (khoảng cách euclidean). Nếu điểm đó gần điểm xanh hơn thì nó thuộc cụm xanh và ngược lại.



Trong trường hợp này thì những điểm phía trên đường thẳng xanh lá sẽ gần điểm trung tâm màu xanh hơn, và những điểm dữ liệu phía dưới đường thẳng sẽ gần điểm màu đỏ hơn



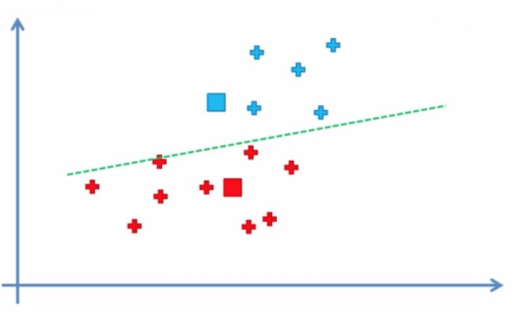
Sau đó tính toán và dời điểm trung tâm trước đó sang vị trí trung tâm sau khi đã chia cụm



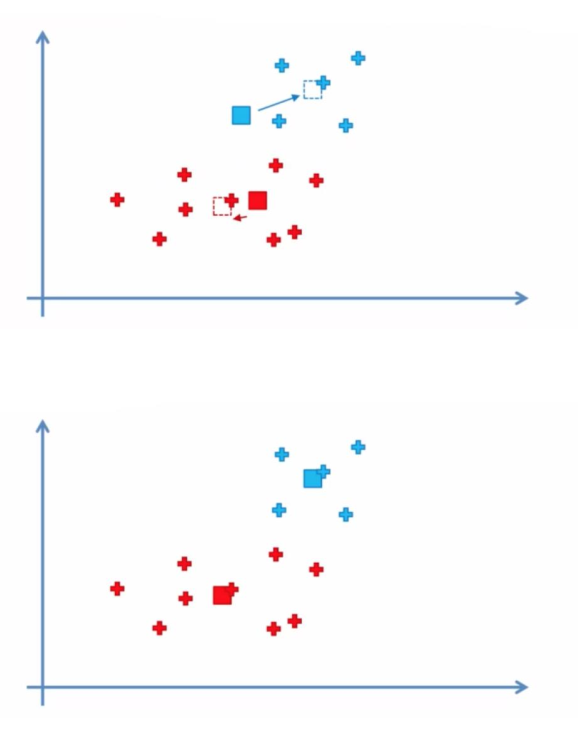
Sau khi đã dịch chuyển điểm dữ liệu tới điểm trung tâm mới (như trên hình) thì thuật toán sẽ phân chia những điểm dữ liệu thành các cụm dựa vào điểm trung tâm mới



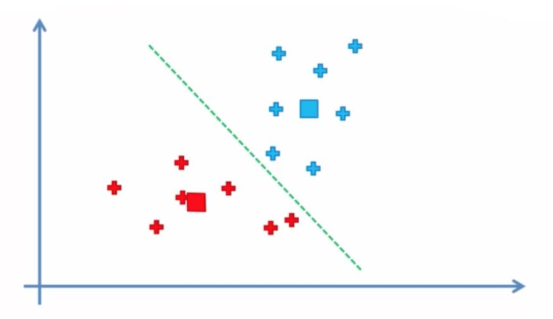
Ta thấy 2 điểm xanh dương dưới đường thẳng xanh lá giờ gần điểm trung tâm màu đỏ và điểm màu đỏ nằm trên đường thẳng giờ gần với điểm xanh dương. Do đó 2 điểm màu xanh dương sẽ chuyển thành màu đỏ và điểm màu đỏ sẽ chuyển sang màu xanh dương



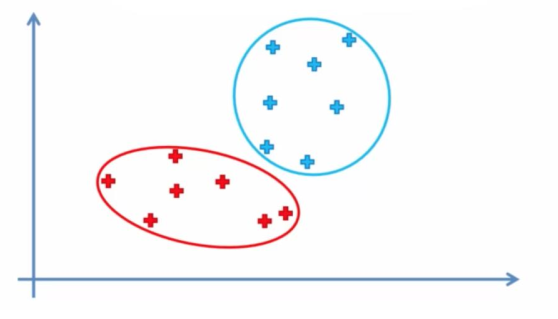
Tiếp tục tính toán và di chuyển điểm trung tâm trước đó tới vị trí điểm trung tâm của tập dữ liệu mới vừa phân cụm lại:



Sau đó tiếp tục chia những điểm dữ liệu dựa trên điểm trung tâm mới và lặp lại cho đến khi không có điểm dữ liệu nào bị chuyển sang cụm khác nữa thì thuật toán sẽ dừng



Bỏ điểm trung tâm đi, ta sẽ có 2 cụm sau khi hoàn thành thuật toán



***Ưu, nhược điểm:***

Thuật toán K-Means có ưu điểm là đơn giản, dễ hiểu và cài đặt

Tuy nhiên, một số hạn chế của K-Means là hiệu quả của thuật toán phụ thuộc vào việc chọn số nhóm K (phải xác định trước) và chi phí cho thực hiện vòng lặp tính toán khoảng cách lớn khi số cụm K và dữ liệu phân cụm lớn.

***Ứng dụng trong thực tế:***

*Marketing:* Xác định các nhóm khách hàng (khách hàng tiềm năng, phân loại và dự đoán hành vi khách hàng,…) sử dụng sản phẩm hay dịch vụ của công ty để giúp công ty có chiến lược kinh doanh tốt hơn

*Biology:* Phân chia nhóm động vật và thực vật dựa vào các đặc tính của chúng

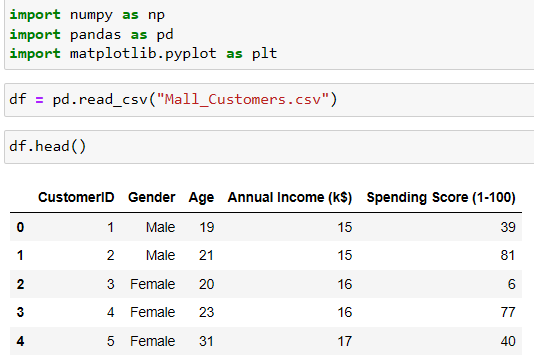
*Insurance, Finace:* Phân nhóm các đối tượng sử dụng bảo hiểm và các dịch vụ tài chính, dự đoán xu hương (trend) của khách hàng, phát hiện gian lận tài chính

*Website:* Phân loại tài liệu, phân loại người dùng trang web,…

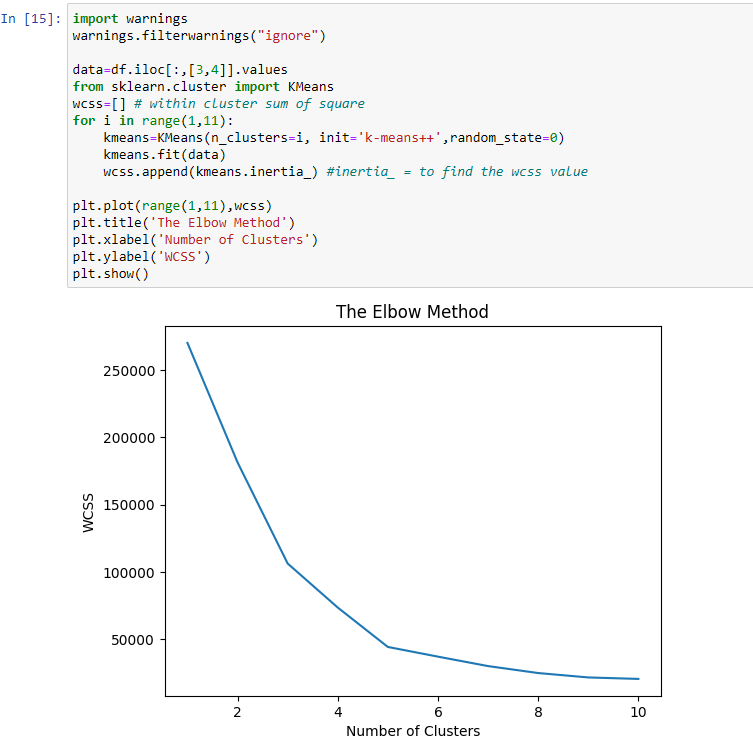
##### **3.1.2 Ứng dụng:**

Bài toán đặt ra có thu nhập hàng năm và điểm tiêu xài của người dùng, một công ty muốn phân chia đối tượng thành các nhóm để có chiến lược khuyến mãi riêng cho từng đối tượng

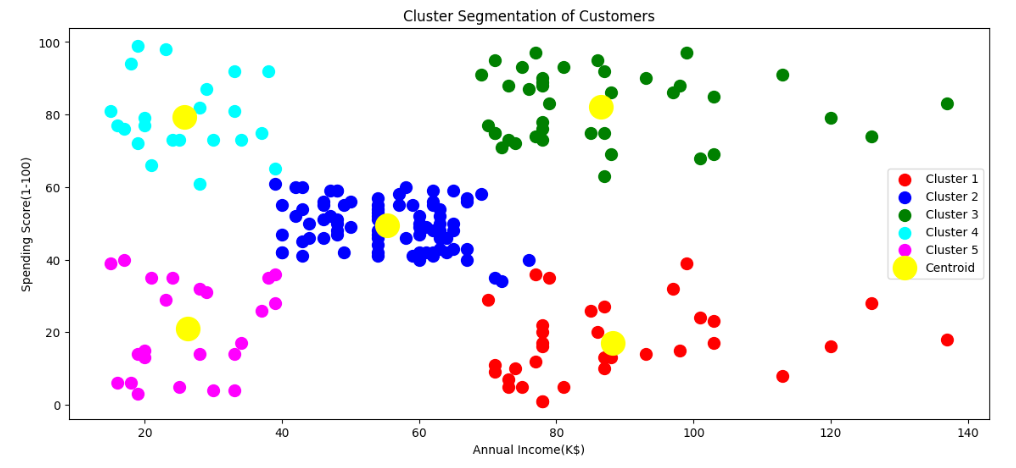
Ở đây ta sử dụng bộ dataset “Mall\_Customers.csv”, đầu tiên ta khai báo thư viện và đọc dữ liệu đầu vào



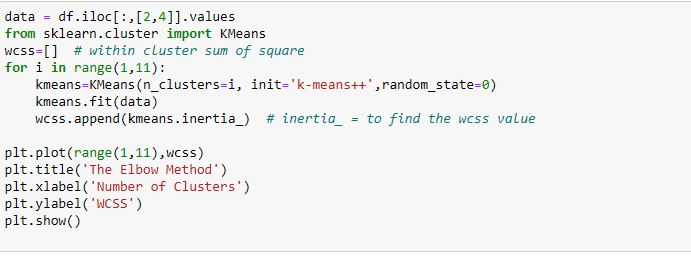
* Dùng Elbow Method để tìm số cụm tối ưu nhất dựa theo **thu nhập trung bình hàng năm**

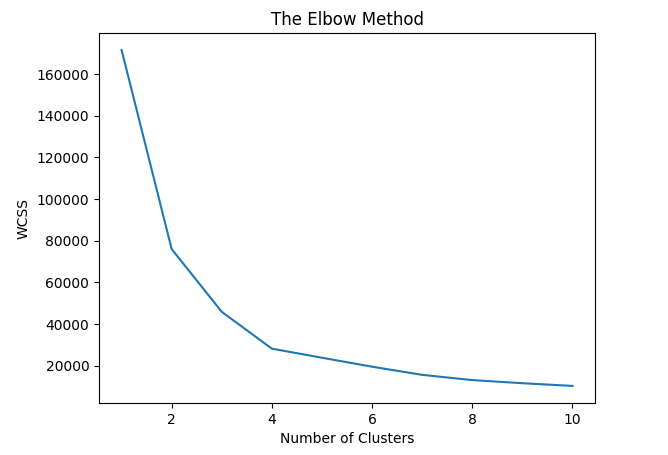


Kết quả thu được sau khi train bộ dữ liệu

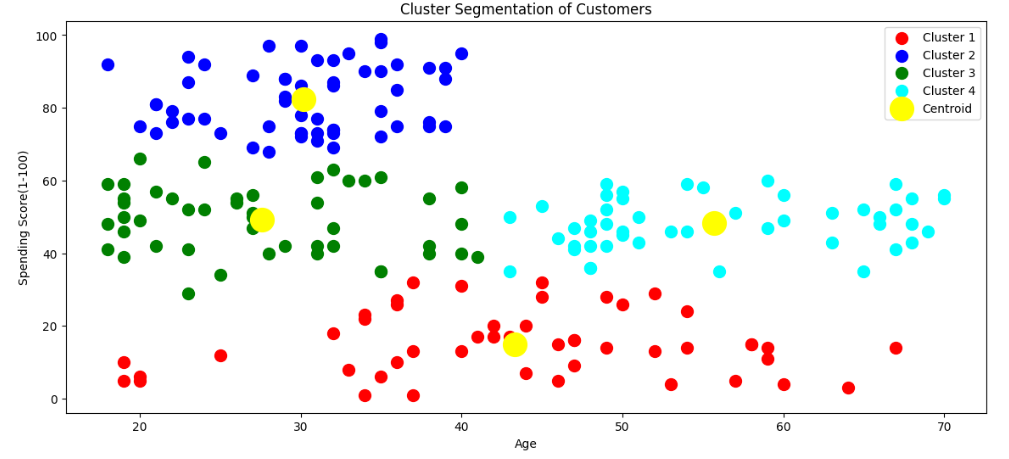


* Dùng Elbow Method để tìm số cụm tối ưu nhất dựa theo **độ tuổi khách hàng**





Kết quả sau khi train bộ dữ liệu:



## **II. Các thư viện của python hỗ trợ trong Machine Learning:**

### **1. Pandas:**

#### **1.1. Giới thiệu:**

Pandas là một thư viện Python toàn diện; một nguồn lực để thực hiện phân tích và thao tác dữ liệu; bất kỳ loại xử lý, phân tích, lọc và tổng hợp dữ liệu nào. Thư viện này được xây dựng dựa trên ngôn ngữ lập trình Python và có thể được sử dụng cho bất kỳ quy trình thu thập thông tin chi tiết từ dữ liệu nào.

Trong nghiên cứu khoa học dữ liệu, Pandas là một trong những công cụ quan trọng trong việc hỗ trợ, xử lý và phân tích dữ liệu với mã nguồn mở nhanh, mạnh, linh hoạt và dễ sử dụng.

***Thành phần cốt lõi của Pandas:***

Hai thành phần chính của Pandas là Series và DataFrame.

Một Series về cơ bản là một cột và một DataFrame là một bảng đa chiều được tạo thành từ một tập hợp các Chuỗi (Series).



Tạo DataFrames ngay bằng Python là điều cần biết và khá hữu ích khi thử nghiệm các phương pháp và chức năng mới trong docs của Pandas.

Có rất nhiều các để tạo mới một DataFrame, một trong những lựa chọn tuyệt vời là sử dụng dict.

*Ví dụ:* Chúng ta có một quầy bán táo và cam, giờ ta cần có một cột cho mỗi loại trái cây và một hàng cho mỗi lần mua hàng của khách hàng. Theo mục đích đó, ta sẽ có lệnh như sau:

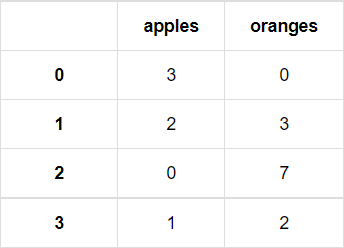
data = { 'apples': [3, 2, 0, 1], 'oranges': [0, 3, 7, 2]}

Sau đó chuyển lệnh này đến phương thức khởi tạo DataFrame:

purchases = pd.DataFrame(data)

purchases

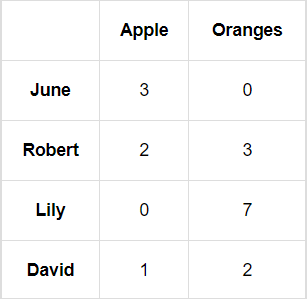
*Kết quả:*



Mỗi cặp khóa: giá trị (key:value) trong data tương ứng với một cột trong kết quả được xuất ra từ DataFrame. Các chỉ số của Data Frame này được cung cấp cho chúng ta thông qua các số từ 0-3, tuy vậy, chúng ta cũng có thể tạo riêng các chỉ số khi khởi tạo Data Frame.

*Ví dụ:* chúng ta có thể lấy tên khách hàng làm chỉ mục như sau: purchases = pd.DataFrame(data, index=['June', 'Robert', 'Lily', 'David']) purchases

Kết quả:



Giờ đây, chúng ta có thể xác định đơn hàng của khách thông qua việc sử dụng tên của họ:

purchases.loc['June']

Kết quả:

apples 3

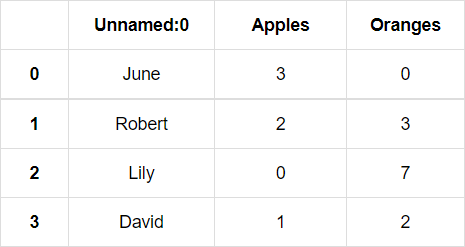
oranges 0

Name: June, dtype: int6

***Đọc dữ liệu từ file CSV bằng Pandas:***

Với các file CSV, ta sẽ cần một dòng dòng đơn để tải dữ liệu:

df = pd.read\_csv('purchases.csv')



CSV không có các chỉ số như Dataframes nên tất cả những gì ta cần là chỉ định index\_cox khi đọc:

Kết quả:

df = pd . read\_csv ( ' Purchase.csv ' , index\_col = 0 ) df



Ở đây chỉ mục được đặt là cột không. Chúng ta sẽ thấy rằng hầu hết các CSV không bao giờ có cột chỉ mục và do đó, ta sẽ không cần lo lắng quá nhiều về bước này.

#### **1.2 Ứng dụng:**

Thông qua thư viện này, chúng ta có thể làm quen với các dữ liệu của mình bằng cách sắp xếp, phân tích và biến đổi chúng. Chẳng hạn như:

* + - * Tính toán số liệu thống kê, trả lời các câu hỏi về dữ liệu như giá trị trung bình, tối đa, tối thiểu của mỗi cột. Cột A có tương quan với cột B không? Sự phân bố dữ liệu trong cột C trông như thế nào?...
      * Làm sạch dữ liệu bằng cách thực hiện những việc như xóa các giá trị bị thiếu và lọc các hàng và cột theo một số tiêu chí.
      * Trực quan hóa dữ liệu với sự trợ giúp từ Matplotlib, biểu đồ thanh, đường kẻ, biểu đồ,....
      * Lưu trữ các dữ liệu đã được làm sạch, chuyển đổi chúng thành CSV, tệp hoặc các cơ sở dữ liệu.

### **2. Numpy:**

*Numpy (Numeric Python)*: là một thư viện toán học rât phổ biến và mạnh mẽ của *Python*. *NumPy* được trang bị các hàm số đã được tối ưu, cho phép làm việc hiệu quả với ma trận và mảng, đặc biệt là dữ liệu ma trận và mảng lớn với tốc độ xử lý nhanh hơn nhiều lần khi chỉ sử dụng *Python* đơn thuần.

***Các loại dữ liệu trong Numpy:***

*NumPy* hỗ trợ nhiều kiểu số hơn nhiều so với *Python*. Bảng sau đây cho thấy các kiểu dữ liệu vô hướng khác nhau được xác định trong *NumPy*.

|  |  |
| --- | --- |
| **STT** | **Loại dữ liệu & mô tả** |
| 1 | bool\_ - Boolean (Đúng hoặc Sai) được lưu trữ dưới dạng byte |
| 2 | int\_ - Kiểu số nguyên mặc định (giống C long; thường là int64 hoặc int32) |
| 3 | intc - Giống hệt với int C (thường là int32 hoặc int64) |
| 4 | intp - Số nguyên được sử dụng để lập chỉ mục (giống như C ssize\_t; thông thường là int32 hoặc int64) |
| 5 | int8 - Byte (-128 đến 127) |
| 6 | int16 - Số nguyên (-32768 đến 32767) |
| 7 | int32 - Số nguyên (-2147483648 đến 2147483647) |
| 8 | int64 - Số nguyên (-9223372036854775808 đến 9223372036854775807) |
| 9 | uint8 - Số nguyên không dấu (0 đến 255) |

|  |  |
| --- | --- |
| 10 | uint16 - Số nguyên không dấu (0 đến 65535) |
| 11 | uint32 - Số nguyên không dấu (0 đến 4294967295) |
| 12 | uint64 - Số nguyên không dấu (0 đến 18446744073709551615) |
| 13 | float\_ - Viết tắt cho float64 |
| 14 | float16 - float: bit dấu, số mũ 5 bit, phần định trị 10 bit |
| 15 | float32 - float: bit dấu, số mũ 8 bit, phần định trị 23 bit |
| 16 | float64 - float: bit dấu, số mũ 11 bit, phần định trị 52 bit |
| 17 | complex\_ - Viết tắt cho complex128 |
| 18 | complex64 - Số phức, được biểu diễn bằng hai số thực 32 bit (thành phần thực và ảo) |
| 19 | complex128 - Số phức, được biểu thị bằng hai số thực 64 bit (thành phần thực và ảo) |

***Đối tượng kiểu dữ liệu (dtype):***

Một đối tượng kiểu dữ liệu mô tả diễn giải khối bộ nhớ cố định tương ứng với một mảng, tùy thuộc vào các khía cạnh sau:

* + - * Loại dữ liệu (số nguyên, đối tượng float hoặc Python)
      * Kích thước của dữ liệu
      * Thứ tự Byte (little-endian hoặc big-endian)
      * Trong trường hợp kiểu có cấu trúc, tên của các trường, kiểu dữ liệu của từng trường và một phần của khối bộ nhớ được lấy bởi từng trường.
      * Nếu kiểu dữ liệu là một mảng con, hình dạng và kiểu dữ liệu của nó

Thứ tự byte được quyết định bằng cách thêm tiền tố '<' hoặc '>' vào kiểu dữ liệu. '<' có nghĩa là mã hóa có giá trị nhỏ (ít quan trọng nhất được lưu trữ ở địa chỉ nhỏ nhất). '>' có nghĩa là mã hóa là big-endian (byte quan trọng nhất được lưu trữ ở địa chỉ nhỏ nhất).

Một đối tượng dtype được xây dựng bằng cú pháp sau:

numpy.dtype(object, align, copy)

Các thông số là :

* **Đối tượng** - Được chuyển đổi thành đối tượng kiểu dữ liệu
* **Căn chỉnh** - Nếu đúng, hãy thêm phần đệm vào trường để làm cho trường tương tự như *C-struct*
* **Sao chép** - Tạo một bản sao mới của đối tượng dtype. Nếu sai, kết quả là tham chiếu đến đối tượng kiểu dữ liệu nội trang

***Array Indexing:***

*NumPy* cung cấp một số cách để truy xuất phần tử trong mảng.

*Indexing và slicing*: Mỗi thành phần trong mảng 1 chiều tương ứng với một chỉ số. Chỉ số trong *NumPy*, cũng giống như chỉ số trong python, bắt đầu bằng 0. Nếu mảng 1 chiều có n phần tử thì các chỉ số chạy từ 0 đến n - 1. Và tương tự như list trong python, *NumPy* arrays cũng có thể được cắt (slicing).

***Boolean array indexing:***

Cho phép bạn chọn ra các phần tử tùy ý của một mảng, thường được sử dụng để chọn ra các phần tử thỏa mãn điều kiện nào đó.

***Ứng dụng:***

* Các phép toán toán học và logic trên mảng.
* Các biến đổi Fourier và các quy trình để thao tác shape.
* Các phép toán liên quan đến đại số tuyến tính. NumPy tích hợp sẵn các hàm cho đại số tuyến tính và tạo số ngẫu nhiên.

***NumPy - Sự thay thế tốt cho MatLab***

NumPy thường được sử dụng cùng với các gói như SciPy (Python Scientific) và Matplotlib (thư viện vẽ đồ thị). Sự kết hợp này được sử dụng rộng rãi để thay thế cho MatLab, một nền tảng phổ biến cho tính toán kỹ thuật. Tuy nhiên, Python thay thế cho MatLab hiện được xem như một ngôn ngữ lập trình hoàn thiện và hiện đại hơn. Điều quan trọng hơn cả là Numpy là một thư viện mã nguồn mở, miễn phí so với MatLab là một thư viện mã nguồn đóng và phải trả phí.

# 3. Sklearn:

#### **3.1. Giới thiệu:**

Scikit-learn (Sklearn) là thư viện mạnh mẽ nhất dành cho các thuật toán học máy được viết trên ngôn ngữ Python. Thư viện cung cấp một tập các công cụ xử lý các bài toán machine learning và statistical modeling gồm: classification, regression, clustering, và dimensionality reduction.

Thư viện được cấp phép bản quyền chuẩn FreeBSD và chạy được trên nhiều nền tảng Linux. Scikit-learn được sử dụng như một tài liệu để học tập.

Scikit-learn hỗ trợ mạnh mẽ trong việc xây dựng các sản phẩm. Nghĩa là thư viện này tập trung sâu trong việc xây dựng các yếu tố: dễ sử dụng, dễ code, dễ tham khảo, dễ làm việc, hiệu quả cao.

Mặc dù được viết cho Python nhưng thực ra các thư viện nền tảng của scikit-learn lại được viết dưới các thư viện của C để tăng hiệu suất làm việc. Ví dụ như: Numpy(Tính toán ma trận), LAPACK, LibSVM và Cython.

#### **3.2. Các thuật toán trong thư viện:**

Thư viện tập trung vào việc mô hình hóa dữ liệu. Nó không tập trung vào việc truyền tải dữ liệu, biến đổi hay tổng hợp dữ liệu. Những công việc này dành cho thư viện *Numpy và Panda.*

Sau đâu là một số nhóm thuật toán được xây dựng bởi thư viện *scikit-learn*:

* **Clustering**: Nhóm thuật toán *Phân cụm dữ liệu không gán nhãn*. Ví dụ thuật toán

*KMeans.*

* **Cross Validation:** Kiểm thử chéo, đánh giá độ hiệu quả của thuật toán học giám sát sử dụng dữ liệu kiểm thử *(validation data)* trong quá trình huấn luyện mô hình.
* **Datasets**: Gồm nhóm các Bộ dữ liệu được tích hợp sẵn trong thư viện. Hầu như các bộ dữ liệu đều đã được chuẩn hóa và mang lại hiệu suất cao trong quá trình huấn luyện như *iris, digit,* ...
* **Dimensionality Reduction**: Mục đích của thuật toán này là để Giảm số lượng thuộc tính quan trọng của dữ liệu bằng các phương pháp như tổng hợp, biểu diễn dữ liệu và lựa chọn đặc trưng. Ví dụ thuật toán *PCA (Principal component analysis)*.
* **Ensemble methods**: Các Phương pháp tập hợp sử dụng nhiều thuật toán học tập để có được hiệu suất dự đoán tốt hơn so với bất kỳ thuật toán học cấu thành nào.
* **Feature extraction**: Trích xuất đặc trưng. Mục đích là để định nghĩa các thuộc tình với dữ liệu hình ảnh và dữ liệu ngôn ngữ.
* **Feature selection**: Trích chọn đặc trưng. Lựa chọn các đặc trưng có ý nghĩa trong việc huấn luyện mô hình học giám sát.
* **Parameter Tuning**: Tinh chỉnh tham số. Các thuật toán phục vụ việc lựa chọn tham số phù hợp để tối ưu hóa mô hình.
* **Manifold Learning**: Các thuật toán học tổng hợp và Phân tích dữ liệu đa chiều phức tạp.
* **Supervised Models**: Học giám sát. Mảng lớn các thuật toán học máy hiện nay. Ví dụ như *linear models*, *discriminate analysis*, *naive bayes, lazy methods, neural networks, support vector machines và decision tree*

### **4. Matplotlib:**

#### **4.1. Giới thiệu:**

Matplotlib là một trong những thư viện giúp trực quan hóa dữ liệu để chúng ta có thể thực hiện các suy luận thống kê cần thiết. Nó là một công cụ rất mạnh mẽ khi làm việc với python, numpy trong việc vẽ đồ thị, trong đó module được dùng nhiều nhất chính là Pyplot, một module với mã nguồn mở sử dụng ngôn ngữ Python, cung cấp giao diện tương tự như Matlab.

#### **4.2. Thành phần:**

Các thành phần của một MATPLOTLIB figure:

* **Figure**: Là một khung cửa sổ chứa tất cả những gì chúng ta vẽ trên đó.
* **Axes *(những khung nhỏ hơn để vẽ hình lên đó)***: Thành phần chính của một figure. Một figure có thể chứa một hoặc nhiều axes. Có thể hiểu, trong khi figure chỉ là khung chứa thì chính các axes mới thật sự là nơi các hình vẽ được vẽ lên.
* **Axis**: Là dòng số giống như các đối tượng và đảm nhiệm việc tạo các giới hạn biểu đồ.
* **Artist**: Mọi thứ mà ta có thể nhìn thấy trên figure là một artist như Text objects, Line2D objects, collection objects. Hầu hết các Artists được gắn với Axes

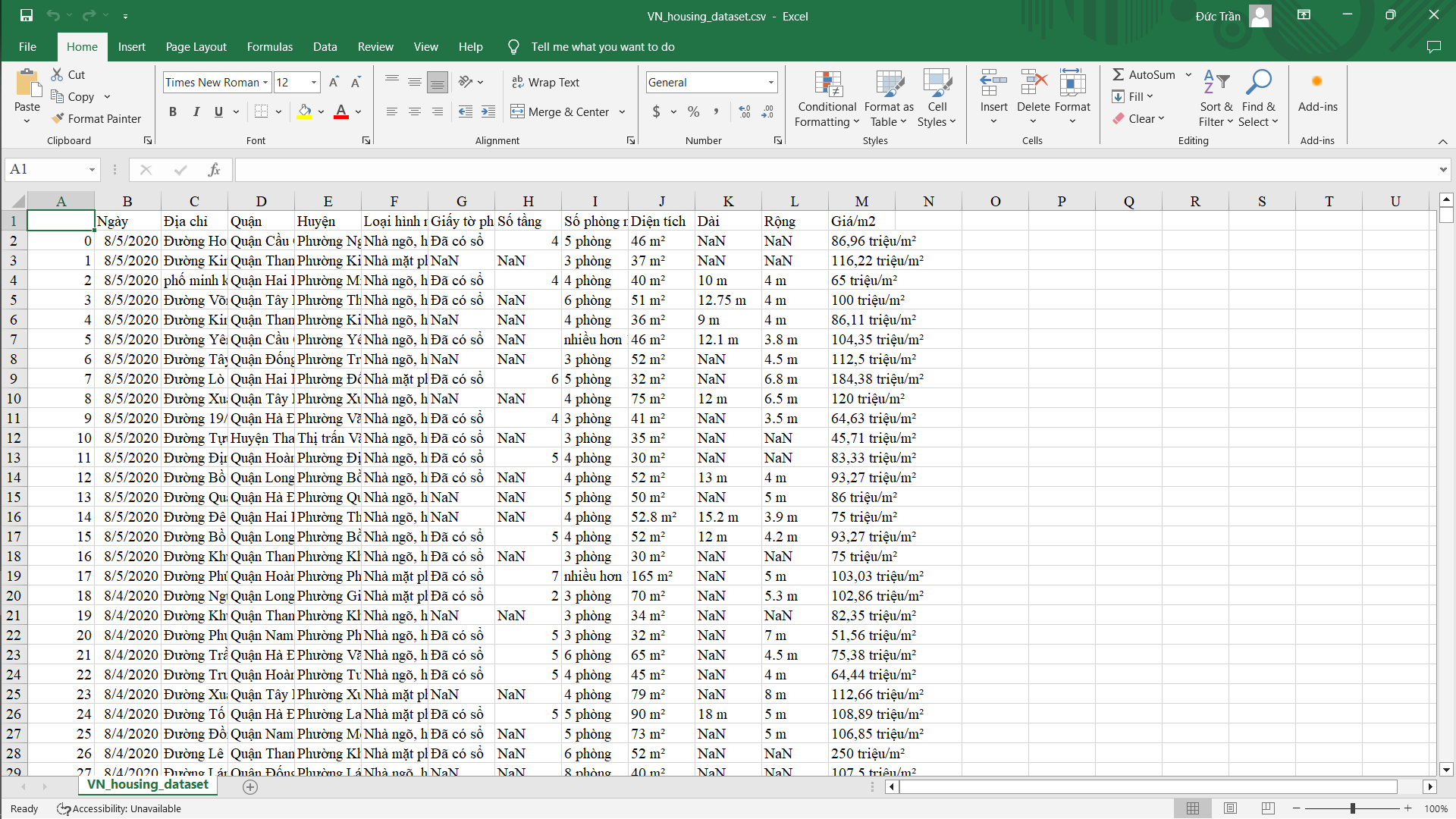
# ỨNG DỤNG

## **I. Thu thập dữ liệu:**

Bộ dữ liệu em sử dụng trông đề tài này được lấy từ những dữ liệu bất động sản trên website [**Alo Nhà Đất**](https://alonhadat.com.vn/) vào tháng 8 năm 2020 bởi [**Le Anh Duc**](https://www.kaggle.com/ladcva)

## **II. Tiền xử lý dữ liệu:**

Bộ dữ liệu bao gồm 82497 dòng dữ liệu chứa các trường thông tin như địa chỉ, loại hình nhà, số phòng, diện tích, …

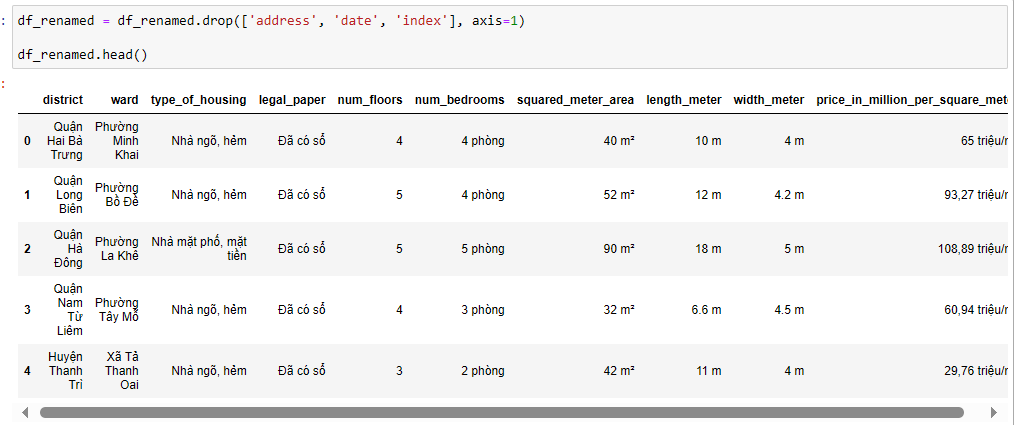


Bởi vì bộ dữ liệu này có rất nhiều dòng dữ liệu bị thiếu các thông tin như số tầng, giấy tờ pháp lý, … nên ta cần phải loại bỏ những dòng này đi để bộ dữ liệu chỉ còn các dòng có đầy đủ các trường thông tin:

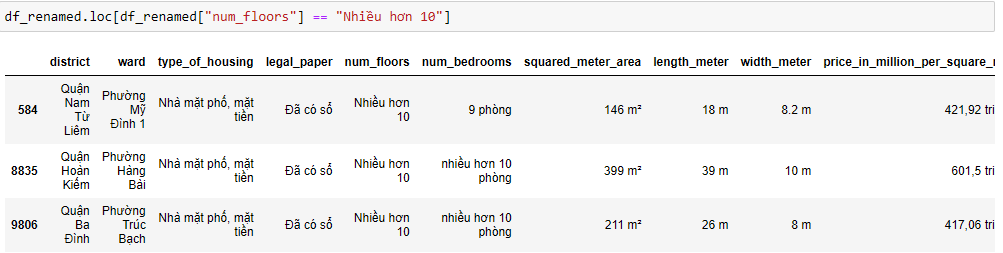


Sau khi tiến hành lọc các trường bị thiếu, bộ dữ liệu còn 11473 dòng

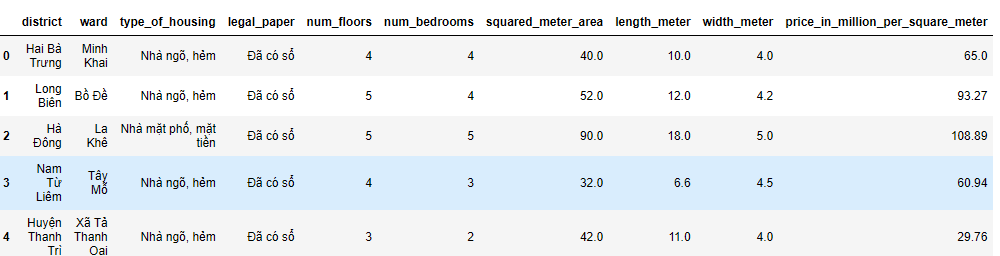
Tiếp theo ta loại bỏ đi các trường dữ liệu không quá cần thiết trong việc tiến hành huấn luyện:



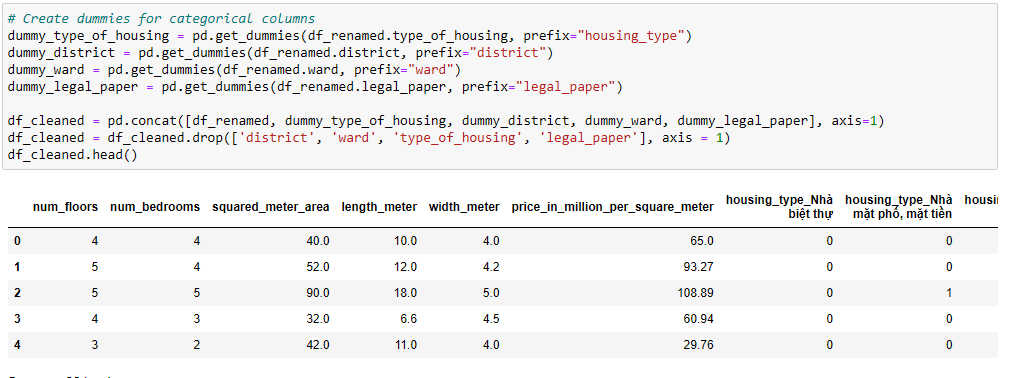
Ta có thể thấy tại các trường ‘num\_floors’ (số tầng) và ‘num\_bedrooms’ (số phòng ngủ) có các giá trị ‘Nhiều hơn 10’, các giá trị này sẽ làm cho việc đánh giá trở nên khó khăn hơn nên ta cũng loại bỏ đi các dòng dữ liệu chứa các giá trị trên:



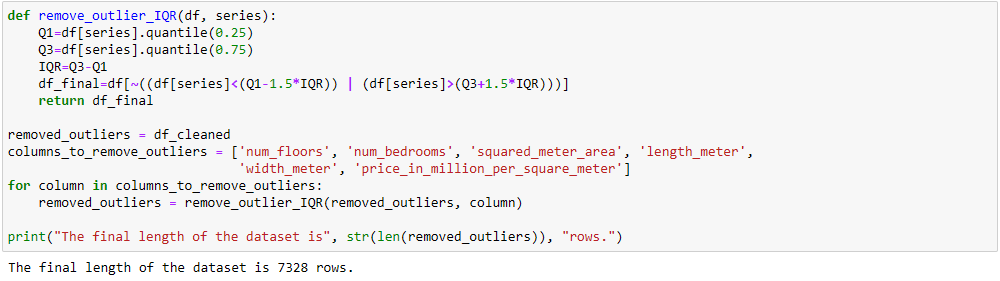
Bởi vì mô hình chỉ có thể hiểu dữ liệu của mình thông qua các con số nên ta cần phải chuyển dữ liệu dạng chữ sang dạng số để mô hình có thể hiểu được.



Tiếp theo ta sẽ tạo ra các Dummy Variable (biến giả) để chứa các trường thông tin ***district, ward, type\_of\_housing*** và ***legal\_paper***

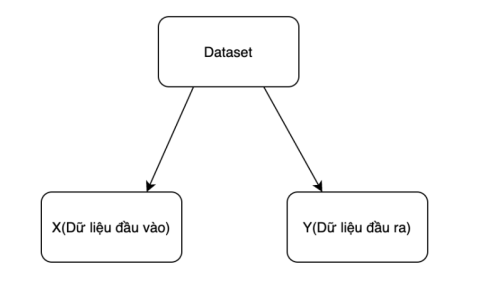


Cuối cùng, ta sẽ lọc các outliers (dữ liệu ngoại lai) để mô hình có thể tiến hành dự đoán giá nhà một cách chính xác hơn, khi đó ta sẽ được một bộ dữ liệu bao gồm 7328 dòng dữ liệu

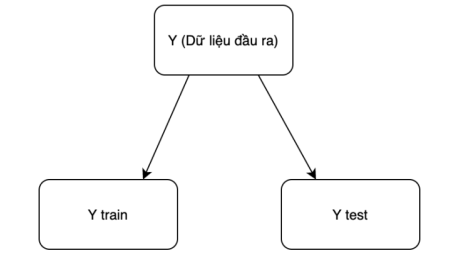
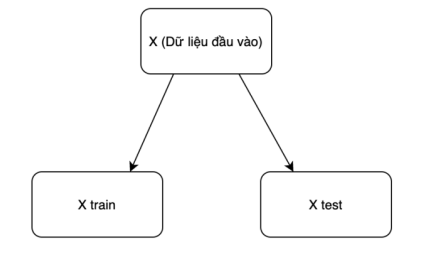


## **III. Huấn luyện mô hình:**

Đầu tiên, ta chia tập dữ liệu ban đầu thành 2 tập dữ liệu: Dữ liệu đầu vào và dữ liệu đầu ra, trong đó dữ liệu đầu vào là các trường thông tin của bộ dữ liệu và dữ liệu đầu ra là giá nhà (house price)



Sau đó ta tách dữ liệu đầu vào thành 2 phần: X\_train, X\_test

Tương tự ta tách dữ liệu đầu ra thành 2 phần: Y\_train và Y\_test

Sau đó ta tiến hành chuẩn hóa dữ liệu. Sở dĩ việc chúng ta cần phải chuẩn hóa dữ liệu là tại vì các trường như legal\_paper chỉ có giá trị 0 và 1, trong khi giá trị của trường diện tích lại lớn hơn rất nhiều (từ 10 ~ 100) khiến cho các vector dữ liệu có sự chênh lệch lớn, vì vậy nên ta phải tiến hành chuẩn hóa dữ liệu hay còn gọi là ***Feature Scaling***

Có 2 cách thường sử dụng để chuẩn hóa dữ liệu là Standardization và Normalization. Normalization được áp dụng trong trường hợp tập dữ liệu phân phối đều hết ở các cột phân phối, và những giá trị bị giảm chiều sẽ nằm trong miền [0,1]. Còn Standardization thì áp dụng cho tất cả các trường hợp bao gồm những trường hợp của Normalization, những giá trị thường sẽ nằm trong miền [-3, 3].

Trong trường hợp này ta sẽ sử dụng phương pháp chuẩn hóa Standardization bởi bộ dữ liệu không phân phối đều



Ta sẽ tạo hai mô hình Artificial Neural Network (ANN) và Random Forest (RF) và tiến hành huấn luyện cả hai mô hình trên bộ dữ liệu trên.

Trên cả hai mô hình ta sẽ sử dụng Grid Search để tối ưu các siêu tham số của các mô hình.

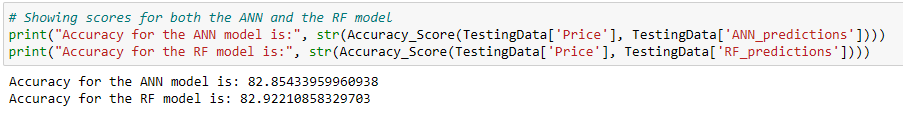
## **IV. Đánh giá kết quả:**

## Ta so sánh kết quả giữa 2 mô hình Artificial Neural Network (ANN) và Random Forest (RF) so với kết quả thực tế:

## 

Độ chính xác đạt được của mô hình ANN là 82.85%

Độ chính xác đạt được của mô hình RF là 82.92%

****

**Danh mục tham khảo:**

[What Do Normalization and Standardization Mean? When to Normalize Data and When to Standardize Data? | by Akalbir Singh Chadha | Becoming Human: Artificial Intelligence Magazine](https://becominghuman.ai/what-does-feature-scaling-mean-when-to-normalize-data-and-when-to-standardize-data-c3de654405ed)

[Dealing with Outliers Using the IQR Method - Analytics Vidhya](https://www.analyticsvidhya.com/blog/2022/09/dealing-with-outliers-using-the-iqr-method/)

[3.2. Tuning the hyper-parameters of an estimator — scikit-learn 1.4.0 documentation](https://scikit-learn.org/stable/modules/grid_search.html)

[Building a Random Forest Model: A Step-by-Step Guide (analyticsvidhya.com)](https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/)

[ANN for Data Science | Basics Of Artificial Neural Network (analyticsvidhya.com)](https://www.analyticsvidhya.com/blog/2021/07/understanding-the-basics-of-artificial-neural-network-ann/)

**Source code:**