# CHƯƠNG II: Bài toán học máy

## 1. Bài toán phân lớp dùng để phân loại giá điện thoại hay sử dụng kỹ thuật Logistic Regression

### 1.1 Mô tả bộ dữ liệu

**Nguồn gốc:**  
  
https://www.kaggle.com/iabhishekofficial/mobile-price-classification

**Thông tin tập dữ liệu:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Đặc điểm của Tập dữ liệu:** | Đa biến | **Số phiên bản:** | 1 | **Khu vực:** | Đời sống |
| **Đặc điểm thuộc tính:** | Phân loại | **Số thuộc tính:** | 20 | **Ngày được tặng** | 2017 |
| **Nhiệm vụ liên quan:** | Phân loại | **Giá trị bị mất?** | Đúng | **Số lượt truy cập web:** | 236k |

Bob đã thành lập công ty di động của riêng mình. Anh ta muốn đấu tranh gay gắt với các công ty lớn như Apple, Samsung, v.v. Anh ta không biết làm thế nào để ước tính giá điện thoại di động mà công ty mình tạo ra. Trong thị trường điện thoại di động cạnh tranh này, bạn không thể chỉ đơn giản giả định mọi thứ. Để giải quyết vấn đề này, anh ta thu thập dữ liệu bán điện thoại di động của nhiều công ty khác nhau. Bob muốn tìm hiểu một số mối quan hệ giữa các tính năng của điện thoại di động (ví dụ: - RAM, Bộ nhớ trong, v.v.) và giá bán của nó. Nhưng anh ấy không giỏi về Máy học. Vì vậy, anh ấy cần sự giúp đỡ của bạn để giải quyết vấn đề này. Trong bài toán này, bạn không phải dự đoán giá thực tế mà là một khoảng giá cho biết giá cao như thế nào

### 1.2 Mô tả bài toán

Bài toán sử dụng bộ dữ liệu các thuộc tính vật lý điện thoại để phân loại ra 4 loại giá điện thoại. Sử dụng bài toán học máy phân lớp. Áp dụng ký thuật Logistic Regression

2.1.1 Mô tả dữ liệu

|  |  |  |
| --- | --- | --- |
| STT | Thuộc tính | Mô tả |
| 1 | Battery | Interger (Lượng pin) |
| 2 | Bluetooth | Boolean (Có bluetooth hay không) |
| 3 | Clock speed | Float (Tốc độ bộ vi xử lý) |
| 4 | Dual sim | Boolean ( Có sim đôi hay không ) |
| 5 | Front camera | Boolean (Có sữa hay không) |
| 6 | 4G | Boolean (Có sim 4G hay không) |
| 7 | Internal memory | Integer (Bộ nhớ trong) |
| 8 | Mobile depth | Integer (Chiều sâu) |
| 9 | Weight of mobile | Float (trọng lượng) |
| 10 | Number of core | Integer (Số nhân) |
| 11 | Primary camera | Integer (Cam sau) |
| 12 | Pixel Height | Integer |
| 13 | Pixel Width | Integer |
| 14 | Ram | Integer |
| 15 | Screen Height | Float |
| 16 | Screen Width | Float |
| 17 | 3G | Boolean |
| 18 | Talk time | Integer |
| 19 | Touch Screen | Boolean |
| 20 | Wifi | Boolean |
| 21 | Price Range | Integer |

2.2 Bài toán

2.3.1 Xử lý dữ liệu

- Chia X, Y với

+ X : Là các thuộc tính Battery đến Wifi

+ Y : Là thuộc tính Price\_range

Code :

import pandas as pd

import numpy as np

# Xử lý dữ liệu

df = pd.read\_csv('train.csv')

df.head()

features = list(df.columns)

print(features)

features.remove('price\_range')

print(features)

X = df[features].values.astype(np.float32)

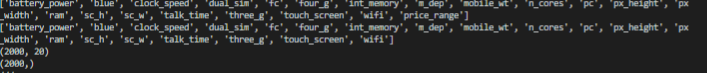
Y = df.price\_range

print(X.shape)

print(Y.shape)

print("///")

Kết quả :



-Chia tập train với tập test : chia tập dữ liệu ra thành 2 phần với tỷ lệ 50:50

Code:

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.5, random\_state = 0)

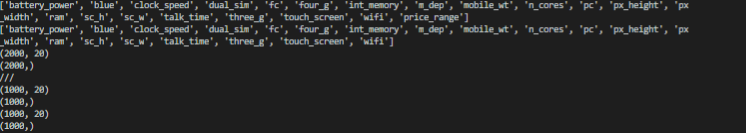
print(X\_train.shape)

print(Y\_train.shape)

print(X\_test.shape)

print(Y\_test.shape)

Kết quả

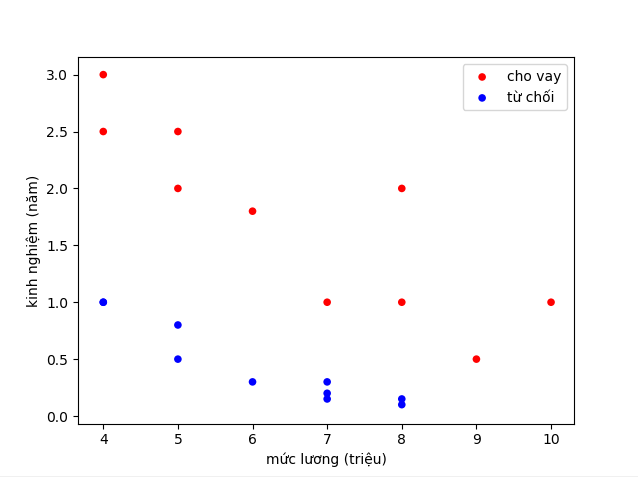
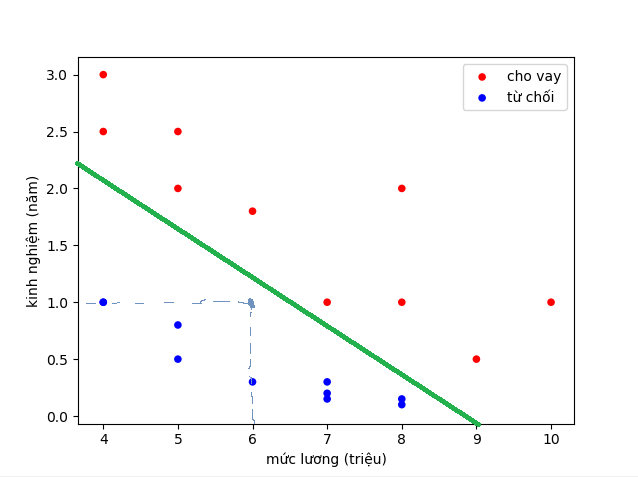


2.3.2 Thuật toán Logistic Regression

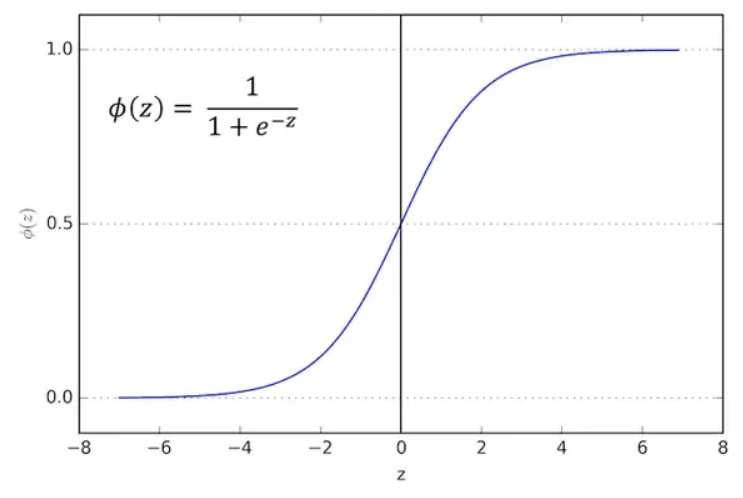
# 1. Giới thiệu

* Trong bài viết này, chúng ta sẽ thảo luận các khái niệm Logistic Regression và xem nó có thể giúp chúng ta xử lý các vấn đề thế nào.
* Logistic Regression là 1 thuật toán phân loại được dùng để gán các đối tượng cho 1 tập hợp giá trị rời rạc (như 0, 1, 2, ...). Một ví dụ điển hình là phân loại Email, gồm có email công việc, email gia đình, email spam, ... Giao dịch trực tuyến có là an toàn hay không an toàn, khối u lành tính hay ác tình. Thuật toán trên dùng hàm sigmoid logistic để đưa ra đánh giá theo xác suất. Ví dụ: Khối u này 80% là lành tính, giao dịch này 90% là gian lận, ...

# 2. Đặt vấn đề

* Ngân hàng bạn đang làm có chương trình cho vay ưu đãi cho các đối tượng mua chung cư. Tuy nhiên gần đây có nhiều chung cư hấp dẫn nên lượng hồ sơ người nộp cho chương trình ưu đãi tăng lên nhiều. Bình thường bạn có thể duyệt 10-20 hồ sơ một ngày để quyết định hồ sơ có được cho vay hay không, tuy nhiên gần đây bạn nhận được 1000-2000 hồ sơ mỗi ngày. Bạn không thể xử lý hết hồ sơ và bạn cần có một giải pháp để có thể dự đoán hồ sơ mới là có nên cho vay hay không. Sau khi phân tích, bạn nhận thấy có 2 yếu tố quyết định đến việc hồ sơ có được chấp nhận hay không, đó là mức lương và kinh nghiệm làm việc. Dưới đây là 1 đồ thị ví dụ 
* Về mặt logic, chúng ta có thể nghĩ ngay đến việc vẽ 1 đường thẳng phân chia các điểm xanh và đó, rồi đưa ra quyết định cho 1 điểm mới dựa vào đường thẳng đó. Ví dụ thế này: 
* Ví dụ đường xanh là đường phân chia. Dự đoán cho hồ sơ của người có mức lương 6 triệu và 1 năm kinh nghiệm là không chấp nhận
* Tuy nhiên, do ngân hàng đang gặp khó khăn nên hạn chế cho vay, ngân hàng yêu cầu hồ sơ đạt trên 80% mới cho vay. Bây giờ không chỉ dừng lại ở việc quyết định cho vay hay không, mà phải tìm xác suất hồ sơ đó cho vay là bao nhiêu.

# 3. Hàm sigmoid

* Giờ phải tìm xác suất cho vay của 1 hồ sơ, đương nhiên là giá trị trong đoạn [0, 1] rồi. Hàm mà luôn có giá trị trong đoạn [0, 1], liên tục mà lại dễ sử dụng thì đó là hàm sigmoid. 
* Nhận xét:
* Hàm liên tục và luôn đưa ra giá trị trong khoảng (0, 1)
* Có đạo hàm tại mọi điểm nên có thể dùng gradient descent

# 4. Thiết lập bài toán

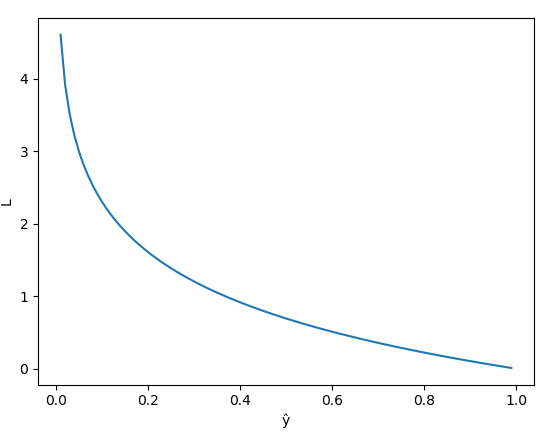
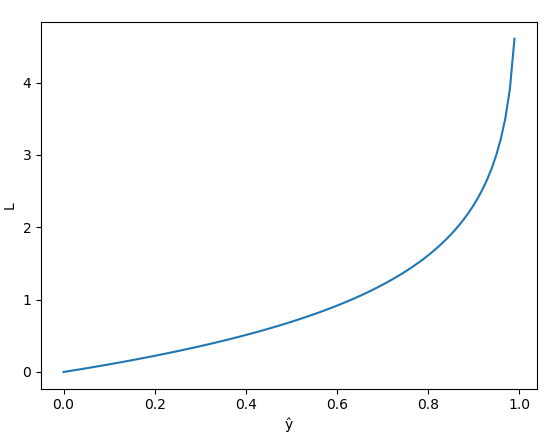
Về cơ bản thì chúng ta sẽ có các bước sau cho 1 bài toán Machine learning:

* Thiết lập model
* Thiết lập hàm mất mát Loss Function
* Tìm tham số bằng việc tối ưu loss function
* Dự đoán dữ liệu mới dựa vào loss function mới tìm được

## 4.1 Model

* Với dòng thức i trong dữ liệu, gọi https://images.viblo.asia/d9f97821-75be-401c-b635-0aeef1507415.png là lương và https://images.viblo.asia/e3d9e4d2-ce8d-4e6b-a23c-984d542207cc.png là kinh nghiệm làm việc của hồ sơ thứ i
* https://images.viblo.asia/8c0f8d7c-824e-43a0-8a2d-02a58a4ace0c.pnglà xác suất mà model dự đoán hồ sơ thứ i cho vay
* https://images.viblo.asia/792b0e6b-a144-4659-bdf7-bd87208050e7.pnglà xác suất mà model dự đoán hồ sơ thứ i không cho vay.
* Ta có ngay https://images.viblo.asia/81a18a8e-7c71-43e6-91f6-fd3225be6b90.png
* Hàm sigmoid là: https://images.viblo.asia/77820f3f-0631-4b07-9f01-541d4bc0a223.png
* Tương tự như hàm dự đoán trong Linear Regression là https://images.viblo.asia/5388405c-2ddb-4dea-8c6c-4db68e12892a.png, thì trong Logistic Regression ta có hàm dự đoán như sau: https://images.viblo.asia/43965a36-f47e-4f23-8d05-695a315f7ce5.png

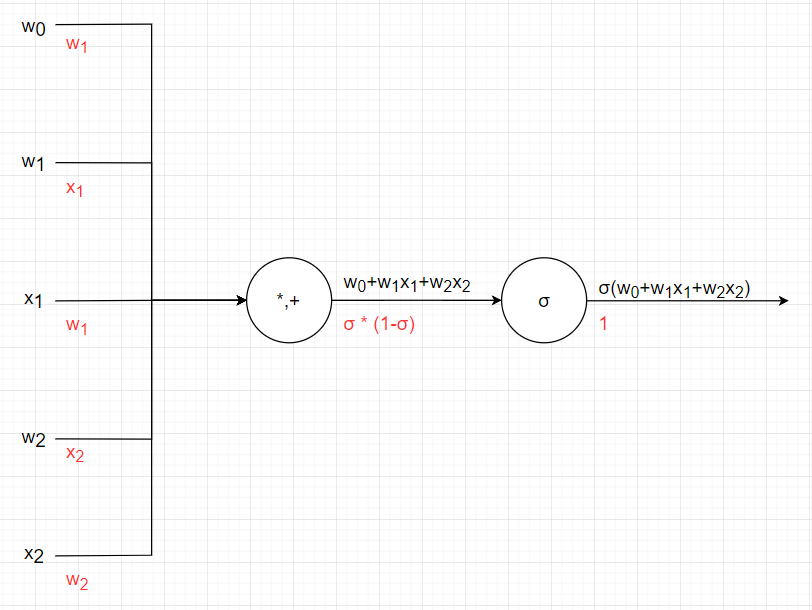
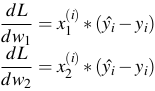
## 4.2 Loss Function - Hàm mất mát

* Bây giờ chúng ta cần 1 hàm để đánh giá độ tốt của model (tức làm dự đoán).
* Ta có nhận xét như sau: + Nếu hồ sơ thứ i là cho vay, tức https://images.viblo.asia/13db9712-1c4a-4510-8fd5-22584bf57492.png = 1 thì ta mong muốn https://images.viblo.asia/ed726ad9-c92c-43f4-afd6-6f0713d5320c.png càng gần 1 càng tốt hay model dự đoán xác suất cho hồ sơ thứ i vay càng cao càng tốt. + Nếu hồ sơ thứ i là không cho vay, tức https://images.viblo.asia/13db9712-1c4a-4510-8fd5-22584bf57492.png = 0 thì ta mong muốn https://images.viblo.asia/ed726ad9-c92c-43f4-afd6-6f0713d5320c.png càng gần 0 càng tốt hay model dự đoán xác suất cho hồ sơ thứ i vay càng thấp càng tốt.
* Với mỗi điểm (https://images.viblo.asia/96db60b5-c191-43f1-873e-0b685263a34d.png, https://images.viblo.asia/13db9712-1c4a-4510-8fd5-22584bf57492.png), ta gọi hàm loss function https://images.viblo.asia/ff288b90-04cc-417e-9b8f-5d18335f1446.png (Trong Machine learning, Deep leaning thì chúng ta hiểu log là ln nhé)
* Thử đánh giá hàm L nhé. Nếu https://images.viblo.asia/078a4431-8732-4724-a6f8-416c689488a7.png. Đây là đồ thị hàm loss trong trường hợp https://images.viblo.asia/13db9712-1c4a-4510-8fd5-22584bf57492.png = 1 
* Nhận xét: + Hàm L giảm từ 0 đến 1 + Khi model dự đoán https://images.viblo.asia/ed726ad9-c92c-43f4-afd6-6f0713d5320c.png = 1, tức giá trị dự đoán gần với giá trị thật https://images.viblo.asia/13db9712-1c4a-4510-8fd5-22584bf57492.png thì L nhỏ, xấp xỉ 0. + Khi model dự đoán https://images.viblo.asia/ed726ad9-c92c-43f4-afd6-6f0713d5320c.png = 0, tức giá trị dự đoán ngược lại với giá trị thật https://images.viblo.asia/13db9712-1c4a-4510-8fd5-22584bf57492.png thì L rất lớn.
* Ngược lại, nếu https://images.viblo.asia/43f2576b-0272-46af-847b-23fcf7d499be.png, ta có đồ thị sau 
* Nhận xét: + Hàm L tăng từ 0 đến 1 + Khi model dự đoán https://images.viblo.asia/ed726ad9-c92c-43f4-afd6-6f0713d5320c.png gần 0, tức giá trị dự đoán gần với giá trị thật https://images.viblo.asia/13db9712-1c4a-4510-8fd5-22584bf57492.png thì L nhỏ, xấp xỉ 0. + Khi model dự đoán https://images.viblo.asia/ed726ad9-c92c-43f4-afd6-6f0713d5320c.png gần 1, tức giá trị dự đoán ngược lại với giá trị thật https://images.viblo.asia/13db9712-1c4a-4510-8fd5-22584bf57492.png thì L rất lớn => Hàm L nhỏ khi giá trị model gần với giá trị thật và rất lớn khi model dự đoán sai, hay nói cách khác L càng nhỏ thì model dự đoán càng gần với giá trị thật. => Bài toán toán quy về tìm giá trị nhỏ nhất của L.
* Ta có hàm mất mát trên tất cả bộ dữ liệu như sau: https://images.viblo.asia/39f2e8f2-d03b-4fc3-917b-6fa8cae579b4.png

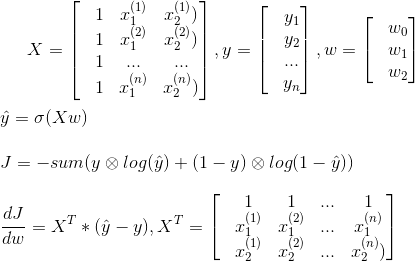
## 4.3 Tính đạo hàm phức tạp bằng kỹ thuật Chain Rule

* Chain rule là gì? Nếu z = f(y) và y = g(x) hay z = f(g(x)) thì https://images.viblo.asia/9134d3a9-1936-45a9-a95d-493a101fbe10.png
* Thử áp dụng tính đạo hàm của hàm sigmoid https://images.viblo.asia/77820f3f-0631-4b07-9f01-541d4bc0a223.png. https://images.viblo.asia/f085633c-27de-4eb6-83f3-7b5b79bc94e4.png =https://images.viblo.asia/d3ae2abf-fef7-45d3-a5f4-571d3ea8affc.png

## 4.4 Áp dụng gradient descent

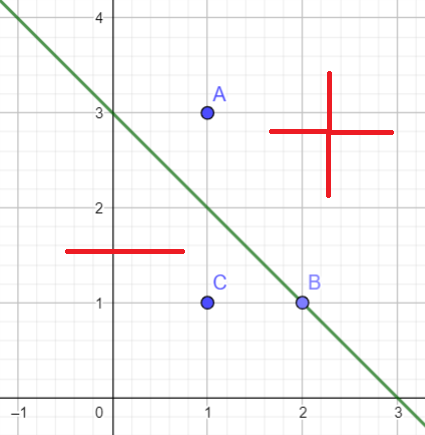
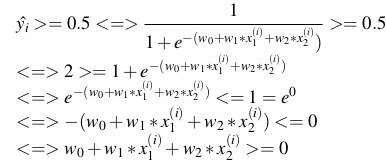
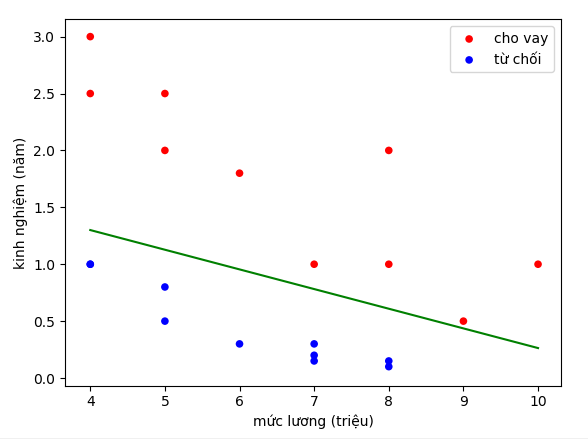
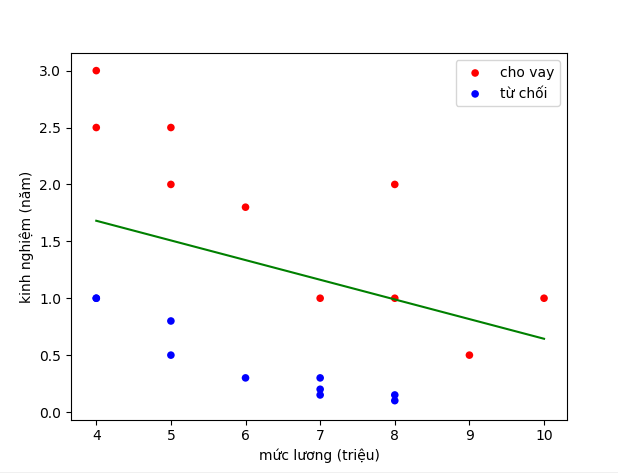
* Với mỗi điểm (https://images.viblo.asia/96db60b5-c191-43f1-873e-0b685263a34d.png, https://images.viblo.asia/13db9712-1c4a-4510-8fd5-22584bf57492.png), gọi hàm mất mát https://images.viblo.asia/ff288b90-04cc-417e-9b8f-5d18335f1446.png trong đó https://images.viblo.asia/51a71410-c471-41d0-907d-4c92ab8f559f.png là giá trị mà model dự đoán, còn yi là giá trị thật của dữ liệu.
* Áp dụng Chain rule ta có: https://images.viblo.asia/71e160e4-9ae0-48cd-9452-5c36456f1e56.png https://images.viblo.asia/3c00f0a7-0f33-4fcb-8033-2192e6db6291.png 
* Từ đồ thị ta thấy: https://images.viblo.asia/b7e9d3df-22bd-47f6-92ea-5b2e0ce08922.png https://images.viblo.asia/b7e9d3df-22bd-47f6-92ea-5b2e0ce08922.png https://images.viblo.asia/fff84b5c-446f-4991-b1db-2f82fd50d22d.png Do đó: https://images.viblo.asia/e725863b-eb0a-4e1f-809a-8f1dc0b0c89f.png
* Tương tự: 
* Đấy là trên 1 điểm dữ liệu, còn trên toàn bộ dữ liệu: https://images.viblo.asia/b4bfc1f8-9da3-44b4-9083-6c05e469c00b.png https://images.viblo.asia/9d7065a6-73b5-4703-935e-4e27db70d902.png https://images.viblo.asia/80d71951-a1a4-42ba-aef1-a1884b96e141.png

## 4.5 Biểu diễn bằng ma trận



* Sau khi thực hiện gradient descent ta tìm được w0, w1, w2. Với mỗi hồ sơ mới https://images.viblo.asia/54785d11-3317-45df-9d2b-d52028e1784c.png ta tính được phần trăm nên cho vay https://images.viblo.asia/d6f3634c-bdbb-4c8d-8fc2-2ffe7ec785da.png rồi so sánh với ngưỡng cho vay của công ty t (thường là 0.5, hoặc cao hơn là 0.8), nếu https://images.viblo.asia/36b5d293-6fea-42c0-9cc2-9a5c87647f72.png thì cho vay, không thì không cho vay.

## 4.6 Xây dựng đường thẳng phân chia

* Xét đường thẳng y = ax + b, thì f = y - (ax + b), ta có được 1 đường thẳng chia mặt phẳng là 2 phần, 1 phần f > 0, 1 phần f < 0 và các điểm trên đường thẳng thì f = 0. 
* Giả sử mốc chính giữa là 0.5 thì https://images.viblo.asia/ed726ad9-c92c-43f4-afd6-6f0713d5320c.png >= 0.5 thì cho vay, ngược lại thì không cho vay. 
* Tương tựhttps://images.viblo.asia/224c574e-b9cd-4afa-8c77-3bc8964ac0a5.png => đường thẳng https://images.viblo.asia/e16b7eb3-7175-4fac-b88b-a2d3f27d6b83.png là đường phân cách giữa các điểm cho vay và từ chối. 
* Trong trường hợp tổng quát t bất kỳ, https://images.viblo.asia/9c22a37c-4208-4437-ac34-18b42bfcfb1c.png <=> https://images.viblo.asia/4f87b5be-8038-4ce4-bfb0-291a80294b0b.png 
* Ta thấy khi t = 0.8 thì đường phân cách gần các điểm màu đỏ hơn so với t = 0.5, thậm chí 2 điểm màu đỏ trước đó được chấp nhận thì bây giờ lại bị loại bỏ.

# 5. Ứng dụng

* Dự đoán email có phải spam hay không
* Dự đoán giao dịch ngân hàng là gian lận hay không
* Dự đoán khối u lành hay ác tính
* Dự đoán khoản vay có trả được không
* Dự đoán khoản đầu tư vào start-up có sinh lãi hay không.

Code :

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.5, random\_state = 0)

print(X\_train.shape)

print(Y\_train.shape)

print(X\_test.shape)

print(Y\_test.shape)

Kết quả : 0.64

2.3.3 Đánh giá

-Accuracy score

Cách đơn giản và hay được sử dụng nhất là *accuracy* (độ chính xác). Cách đánh giá này đơn giản tính tỉ lệ giữa số điểm được dự đoán đúng và tổng số điểm trong tập dữ liệu kiểm thử.

Các thông số

### Precision

Xem xét trên tập dữ liệu kiểm tra xem có bao nhiêu dữ liệu được mô hình dự đoán đúng. Đây chính là chỉ số **accuracy - độ chính xác** như chúng ta sử dụng bên trên. Một cách toán học thì **precision** được biểu diển như sau:

Precision= Y(true)/Y(tổng)

### Recall

Chỉ số này còn được gọi là **độ bao phủ** tức là xem xét xem mô hình tìm được có khả năng **tổng quát hóa** như thế nào. Từ hai yếu tố **độ chính xác** và **độ bao phủ** phía trên người ta đưa ra một chỉ số khác gọi là **F1-Score**

**Recall= [Y(true) giao Y(tổng)]/Y(true)**

### F1-Score

Đây được gọi là một **trung bình điều hòa**(harmonic mean) của các tiêu chí Precision và Recall. Nó có xu hướng lấy giá trị gần với giá trị nào nhỏ hơn giữa 2 giá trị **Precision** và **Recall** và đồng thời nó có giá trị lớn nếu cả 2 giá trị **Precision** và **Recall** đều lớn. Chính vì thế **F1-Score** thể hiện được một cách khách quan hơn **performance** của một mô hình học máy.

F1- Score = 2x [(Precision.Recall)/(Precision+Recall)]

**-Confusion matrix**