**Báo cáo tìm hiểu tuần 1**

# 1, Hồi quy, phân lớp

- Đều là thuật toán học có giám sát, dữ liệu train đã được gắn nhãn.

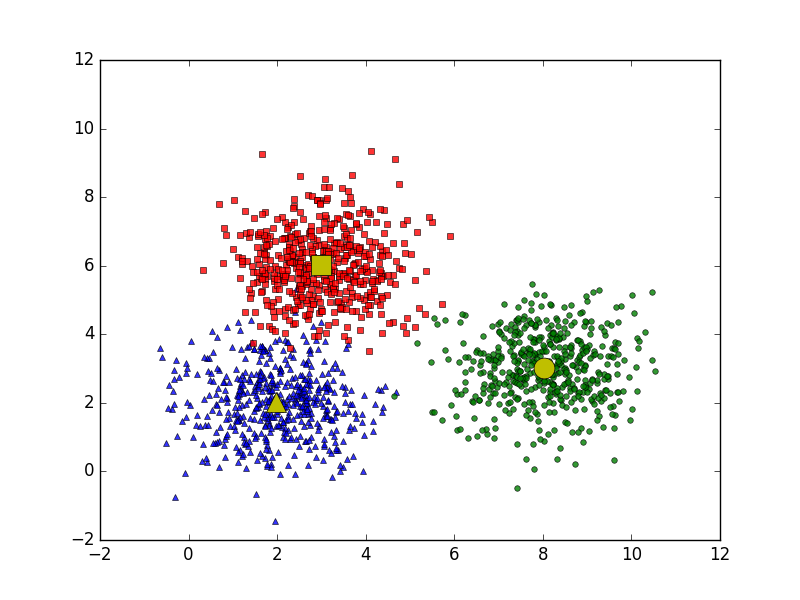
- Trong phân lớp (classification) nhãn là một số lượng hữu hạn các nhóm.

- Trong hồi quy (regression) nhãn thường là các số, liên tục và vô hạn.

# 2, Các thuật toán ML cơ bản:

## K-Means Cluster

Là thuật toán phân cụm dữ liệu chưa có nhãn (học không giám sát)



**Ý tưởng:** Tìm điểm trung tâm của các cụm dữ liệu, dựa vào đó để phân chia các điểm xung quanh thuộc cụm nào.

**Các bước thực hiện:**

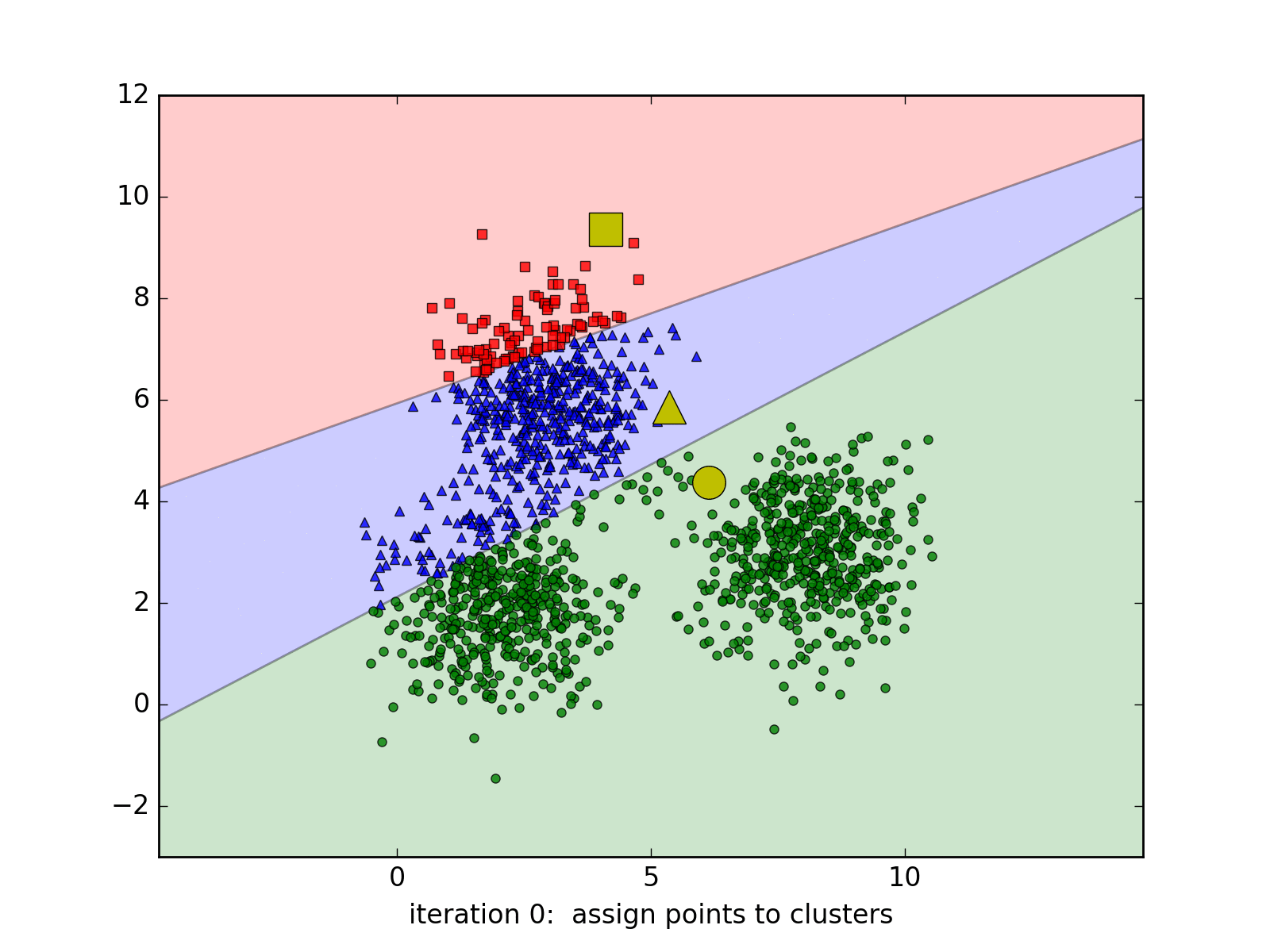
B1: Chọn ngẫu nhiên K điểm (tương ứng với số cụm muốn phân) làm center của mỗi cụm.

B2: Phân các điểm dữ liệu vào các cụm có center gần nó nhất.

B3: Nếu không có thay đổi cụm trong bước 2 thì dừng lại.

B4: Lấy trung bình cộng của tất cả các điểm trong cụm làm center mới. (Cập nhật lại center)

B5: Quay lại bước 2.



**Hạn chế:**

- Cần biết số lượng cụm -> Có thể dùng Elbow method để xác định cụm

- Nghiệm cuối phụ thuộc vào center được chọn ban đầu -> Chạy nhiều lần để lấy kết quả tốt nhất hoặc K-means++

- Cụm cần có số điểm bằng nhau.

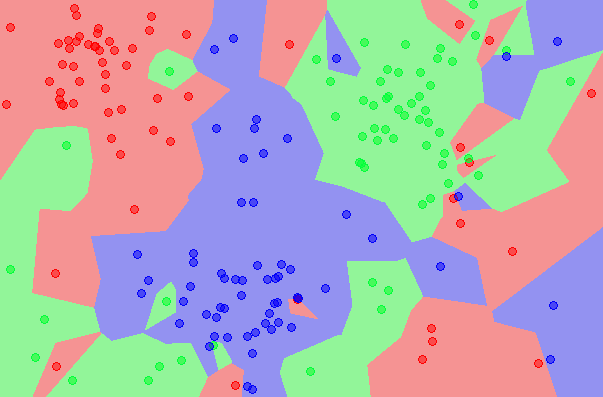
- Cụm cần có dạng hình tròn.

- Các cụm không bao nhau.

## 

## KNN (K-nearest neighbors)

Học có giám sát, thuộc loại lazy learning (không học gì từ dữ liệu, mọi tính toán được thực hiện khi cần dự đoán kết quả.)



**Ý tưởng:** Đầu ra được suy ra trực tiếp từ K điểm gần nhất trong training set. Có thể thể sử dụng major voting để thay đổi trọng số.

**Ưu điểm:**

- Độ phức tạp bằng 0.

- Việc dự đoán kết quả cho dữ liệu mới đơn giản.

- Không cần giả sử gì về phân phối của các class.

**Nhược điểm:**

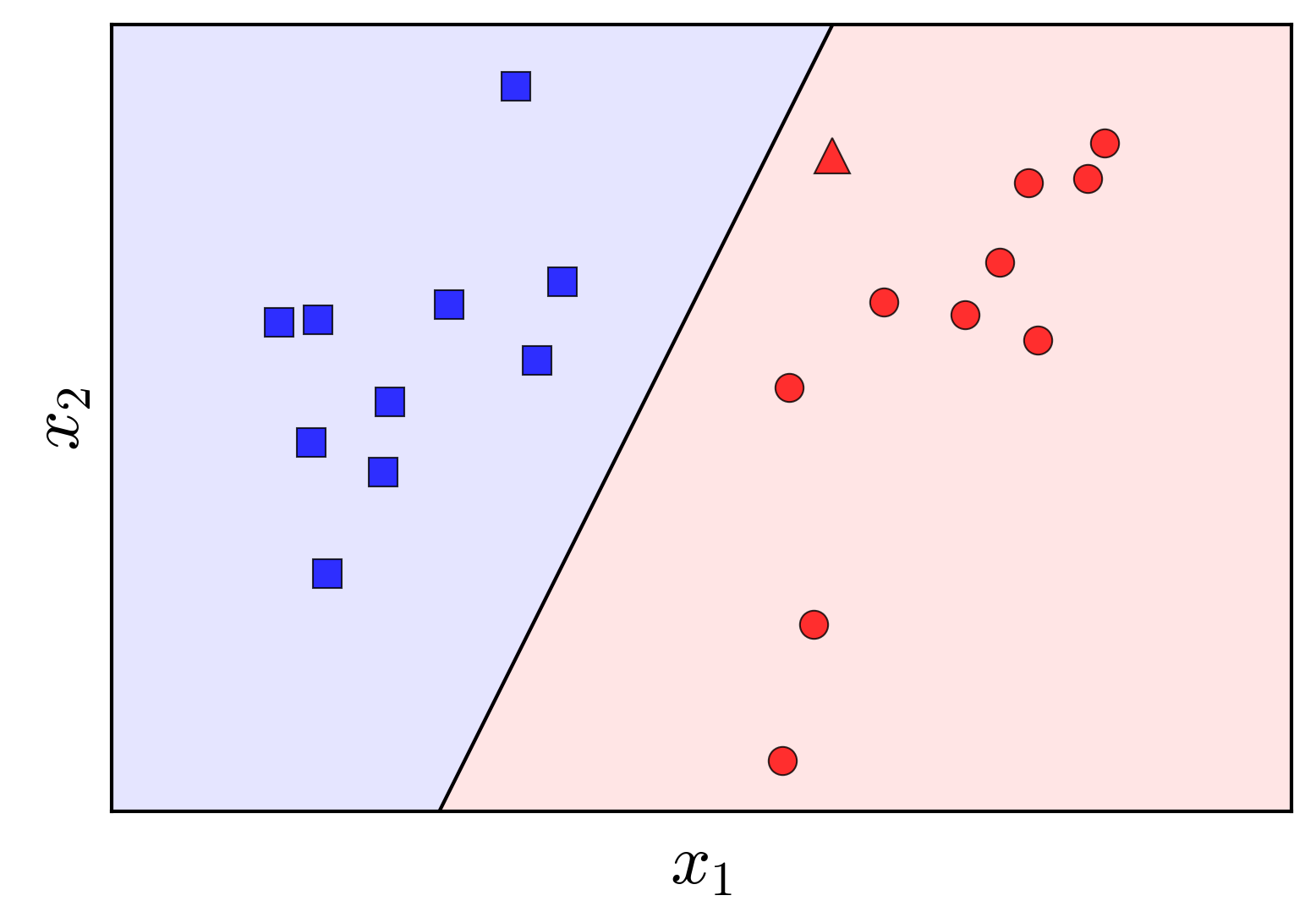
- Nhạy cảm với nhiễu khi K nhỏ.

- Tốn thời gian và bộ nhớ khi làm việc với dữ liệu lớn.

## 

## PLA (Perceptron Learning Algorithm)

Thuật toán có giám sát, đơn giản nhất cho 2 class (binary classification)



**Ý tưởng:** Chọn một nghiệm dự đoán nào đó, thay đổi nghiệm sau mỗi vòng lặp để có kết quả tốt hơn.

**Các bước thực hiện:**

B1: Chọn ngẫu nhiên một nghiệm phân chia 2 class.

B2: Cập nhật nghiệm nếu có điểm bị nhầm class.

B3: Kiểm tra xem còn điểm nào bị nhầm class không. Nếu còn, quay lại bước 2. Nếu không, dừng thuật toán.

**Hạn chế:**

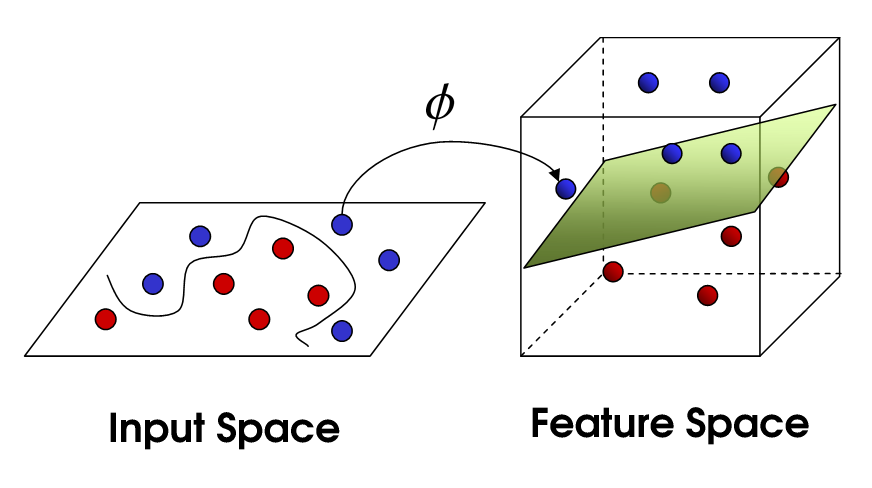
- Có vô số nghiệm khác nhau thỏa mãn.

- Đòi hỏi dữ liệu phải linearly separable (dữ liệu phải chia ra 2 lớp phân biệt hoàn toàn, không lẫn vào nhau) -> có thể sử dụng Pocket Algorithm (giới hạn số vòng lặp/ lưu lại số điểm bị misclassified và tìm số nhỏ nhất).

## 

## SVM (Support Vector Machine)

Là thuật toán học có giám sát, xây dựng một mặt siêu phẳng để tách riêng dữ liệu của 2 lớp riêng biệt.



Ý tưởng: Ánh xạ dữ liệu ban đầu vào không gian nhiều chiều hơn để xây dựng mặt phẳng phù hợp để phân chia dữ liệu sau đó ánh xạ dữ liệu đó về không gian ban đầu.

Ưu điểm:

- Tìm được mặt phẳng phân chia với cả dữ liệu non-linearly separable

- Ít overfitting

Nhược điểm:

- Sử dụng nhiều RAM.

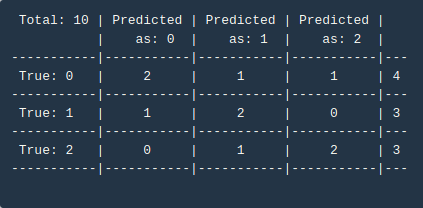
- Khó scale với dữ liệu quá lớn.

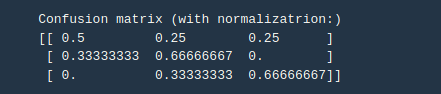
# 3, Đánh giá model

## Với mô hình classification

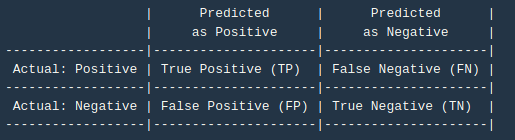
**Accuracy** = số điểm đoán đúng/ tổng số điểm

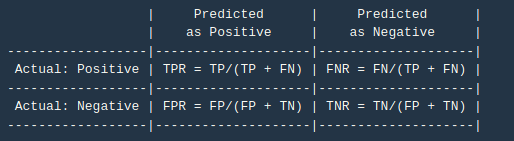
**Confusion matrix**





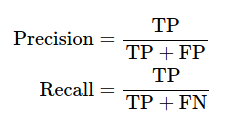
**True/False Positive/Negative**





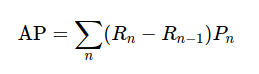
Mô hình tốt là khi FPR thấp, TPR cao.

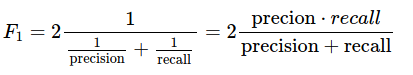
**Precision và Recall**

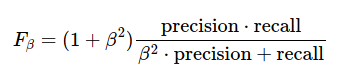
****

Mô hình tốt là mô hình có cả Precision và Recall cao.

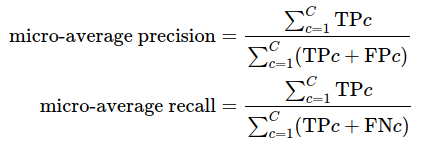
Có 2 cách đo chất lượng của bộ phân lớp dựa vào Precision và Recall là **Average Precision(AP)** và **F1-score**







Đối với bài toán có phân lớp nhiều lớp thì có 2 cách đánh giá dựa trên Precision và Recall là **micro-average** và **macro-average**

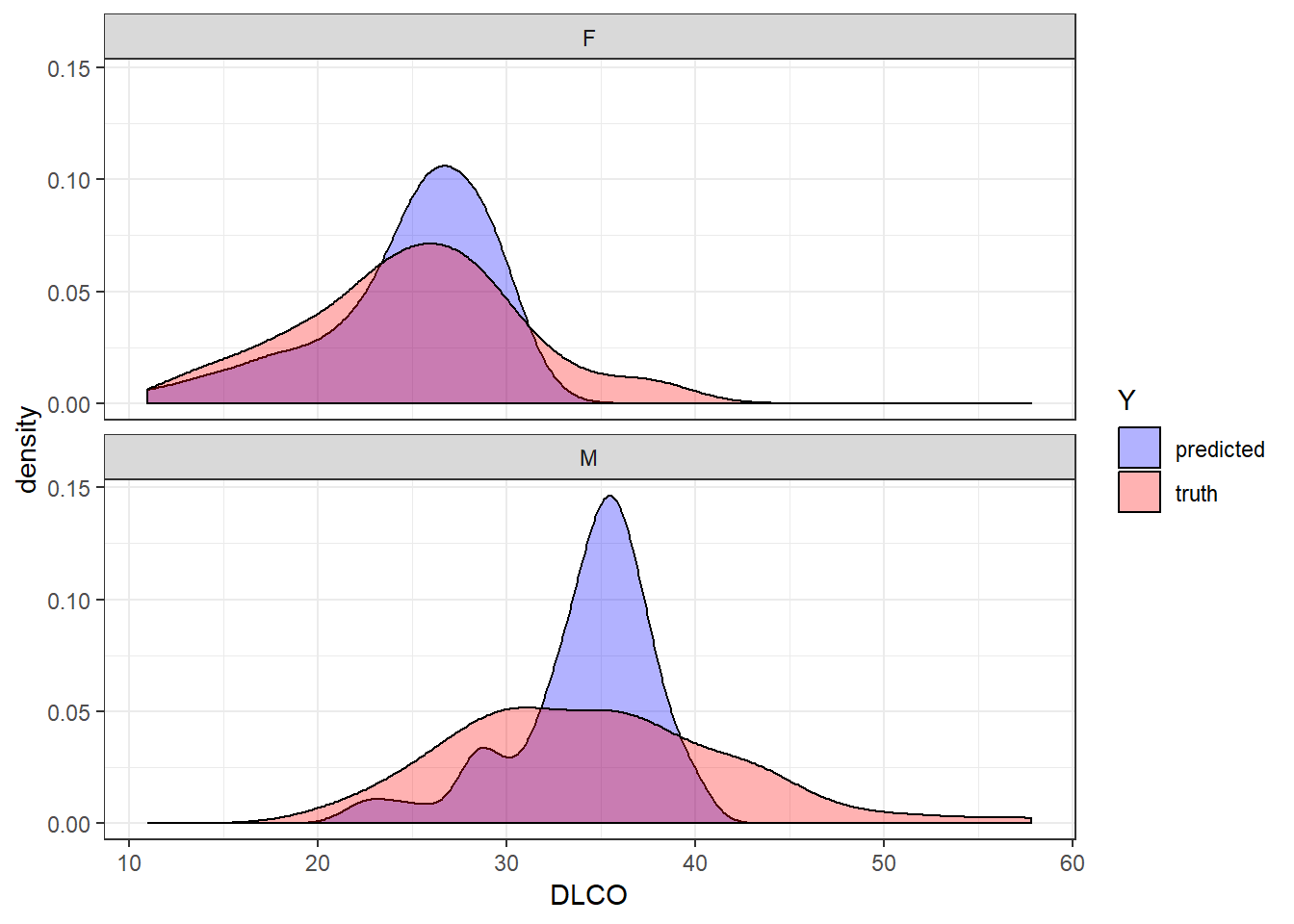
****

**macro-average precision = trung bình cộng các precision**

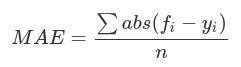
**macro-average recall = trung bình cộng các recall**

## Với mô hình regression

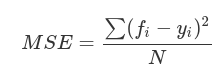
**Biểu đồ trực quan**



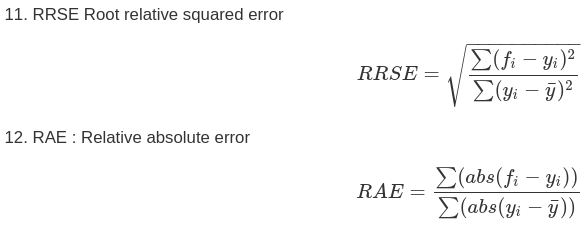
**Sai biệt tuyệt đối:** MAE, MEDAE, SAE, MAPE



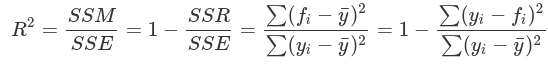
**Bình phương sai số:** MSE, MEDSE,SSE, RMSE, MSLE, RMSLE

****

**Sai biệt tương đối**

****

**Hệ số xác định (R2):** giải thích được bao nhiêu phần phương sai của biến kết quả trong mẫu. => Không đánh giá được độ chính xác.

****

**Tương quan tuyến tính giữa giá trị thực và tiên lượng**

# 4, Overfitting, Underfitting

## Overfitting

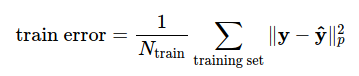
là hiện tượng mô hình tìm được quá khớp với dữ liệu train, có thể dẫn đến dự đoán sai đối với dữ liệu test.

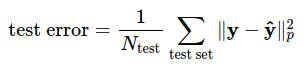
Overfitting xảy ra khi mô hình quá phức tạp và dữ liệu train quá ít.

## Underfitting

là hiện tượng ngược lại.

## Đánh giá chất lượng mô hình





## Các biện pháp khắc phục:

**Validation:** tách một phần dữ liệu train ra để test, phần đó được gọi là validation. Tìm mô hình để train error và validation error nhỏ nhất.

**Cross-validation:** khi lượng dữ liệu train ít, ta chia nó thành k tập con. Mỗi lần chạy train bằng k-1 tập con và dùng tập cuối để test. Từ đó đánh giá chất lượng mô hình.

**Early stopping:** Dừng lặp khi thấy validation error bắt đầu tăng.

**Thêm số hạng vào hàm mất mát**

**l2 regularization (weight decay)**

**Tikhonov regularization**

**Dropout:** Tắt ngẫu nhiên các units trong Networks.

# 5, Gradient Descent

Là một phương pháp tối ưu lặp với mục tiêu tìm tập các biến để tối ưu hóa models. Thường là tìm nghiệm của phương trình đạo hàm để hàm loss f(x) có giá trị nhỏ nhất.

## Các bước thực hiện:

B1: Khởi tạo nghiệm ngẫu nhiên.

B2: Cập nhật giá trị của nghiệm x = x - learning\_rate \* f'(x)

B3: Tính lại f(x). Nếu f(x) đủ nhỏ thì dừng, không thì quay lại bước 2.

## Các thuật toán tối ưu cho GD:

**Momentum:** Thêm vận tốc vào hàm.

**Nesterov Accelerated Gradient (NAG):** lấy f’(x) tạo thời điểm tiếp theo chứ không phải hiện tại.

## Biến thể của GD:

**Batch GD:** Như trên. Sử dụng mọi điểm dữ liệu để tính toán tại một thời điểm rồi mới cập nhật x.

**Stochastic Gradient Descent (SGD):** Chỉ sử dụng một điểm dữ liệu để tính toán tại một thời điểm và cập nhật x.

-> Giảm tốc độ thực hiện epoch (duyệt qua 1 lượt tất cả các điểm) nhưng chỉ cần thực hiện một lượng nhỏ epoch để thu được kết quả tốt.

-> Phù hợp với bài toán dữ liệu lớn, dữ liệu online

**Mini-batch Gradient Descent:**

Bắt đầu với việc chi toàn bộ dữ liệu ra làm các mini-batch có số điểm dữ liệu bằng nhau. Mỗi epoch sẽ lấy ra một mini-batch để thực hiện, sử dụng ***n*** điểm dữ liệu (n>1) để tính toán tại một thời điểm và cập nhật x.