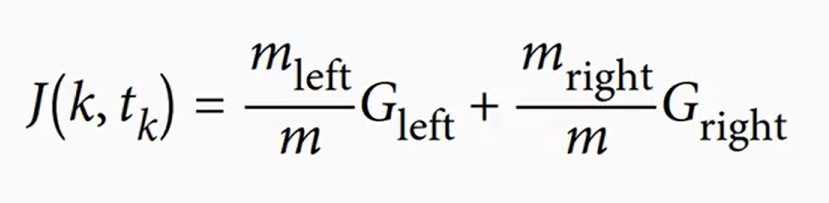
1. Mô tả cách thực hiện classification sử dụng decision trees.

Với decision trees ta sử dụng cây nhị phân để làm classification, dựa vào giá trị của các features để chia các sample ra các node khác nhau. Mỗi node sẽ chứa thông tin của số lượng sample tương thích, giá trị của feature (threshold), tên class,... Phân chia cho tới khi không chia được nữa (tới nút leaf) -> kết thúc phân loại và thu thập kết quả.

1. CART algorithm được dùng để làm gì? Mô tả các bước chính của nó.

CART algorithm dùng để xây dựng ra binary tree

* + Lựa chọn ra feature k và tính threshold tk để tách ra làm 2 nhánh. Sử dụng cost function để tính gini nhỏ nhất -> để chọn features và phù hợp



* + Tách cho đến khi giới hạn cây bằng với giá trị đã quy định (depth, số node) hoặc khi impurity không giảm được nữa -> dừng lại

1. Gini impurity và entropy là gì?
   * Gini impurity là phép đo độ không tinh khuyết (bị pha tạp) của 1 node, nếu giá trị bằng 0 -> node đó tinh khuyết và > 0 ta kết luận node đó bị tạp pha (chứa sample của class khác). Độ tạp pha tỉ lệ thuận với giá trị phép do.
   * Gini entropy cũng là phép đo độ không xem tạp, hỗn loạn của 1 node.

Kết luận chung: xây dựng ra tree là giống nhau, tuy nhiên phép đo inpurity là nhanh hơn vì không dùng log và phép đo entropy cho ra cây cân bằng hơn về 2 phía

1. Mô tả cách thực hiện regression sử dụng decision trees.

Giống với decision tree, ta cần phải xây dựng ra các node trong đó có chứa điều kiện để rẽ nhánh. Khác với decision tree, regression sẽ yêu cầu nút lá trả về value nào đó.

1. Thế nào là một parametric model và non-parametric model? Trong các thuật toán đã học (linear regression, polinominial regression, logistic regression, softmax regression, SVM, decision tree) cái nào là parametric cái nào là non-parametric models?
2. Nêu cách thực hiện regularization với decision tree.

Regularization với decision tree, ta chỉ cần thây đổi các paremeter bên dưới sao cho phù hợp

max\_depth: giảm độ sâu của cây khi bị overfitting

min\_samples\_split: giảm

min\_samples\_leaf: tăng -> 1 node leaf sẽ có sample -> ít phân nhánh -> giảm độ sâu

max\_leaf\_nodes: giảm -> sẽ giảm được độ sâu

max\_features: tăng

Tuy nhiên với các paremeter trên ta cần thực hiện giảm hoặc tăng sao cho phù hợp để tránh bị underfitting (depth = 1)

1. Nếu sử dụng decision tree (không dùng regularization) thì model được tạo có độ sâu khoảng bao nhiêu khi trained với 1000 samples (sử dụng CART algorithm)? Gợi ý: cây nhị phân với k leaf nodes sẽ có độ sâu bao nhiêu

Đối với cây nhị phân cân bằng độ sâu của nó xấp xĩ 2X = k leaf node, với x là giá trị đầu tiên trong tập xác định

Ví dụ 2X = 512-> x = 9 -> độ sâu bằng 9s

1. Nếu một decision tree bị overfitting thì ta nên tăng hay giảm max\_depth của nó? Tại sao?

Nếu decision tree bị overfitting thì ta nên giảm max\_depth vì depth càng cao độ sâu của cây càng sâu -> cây sẽ phức tạp, tốn nhiều bộ nhớ. Vì vậy ta phải giảm depth để cây đơn giản và ít tốn bộ nhớ. Tuy nhiên không nên giảm quá nhiều, nó sẽ gây ra underfitting.

1. Nếu một decision tree bị underfitting, thì ta có nên thực hiện feature scaling? Tại sao?

Thực hiện feature scaling là không cần thiết bởi vì sự phân nhánh của decision dựa vào giá trị của threshold nếu có giá trị khác biệt lớn thì nó vẫn tách ra 2 node khác nhau và ảnh hưởng đến underfitting và overfitting dựa chủ yếu vào các parameter của model. Ta giới hạn các tham số này sao cho phù hợp.

1. Nếu ta mất 1 giờ để huấn luyện một decision tree với training data có 1 triệu samples thì sẽ tốn bao nhiêu thời gian để huấn luyện với training data có 10 triệu samples? Gợi ý: computational complexity cho việc huấn luyện một decision tree là O(n × m log(m)), với n là số features, m là số instances.

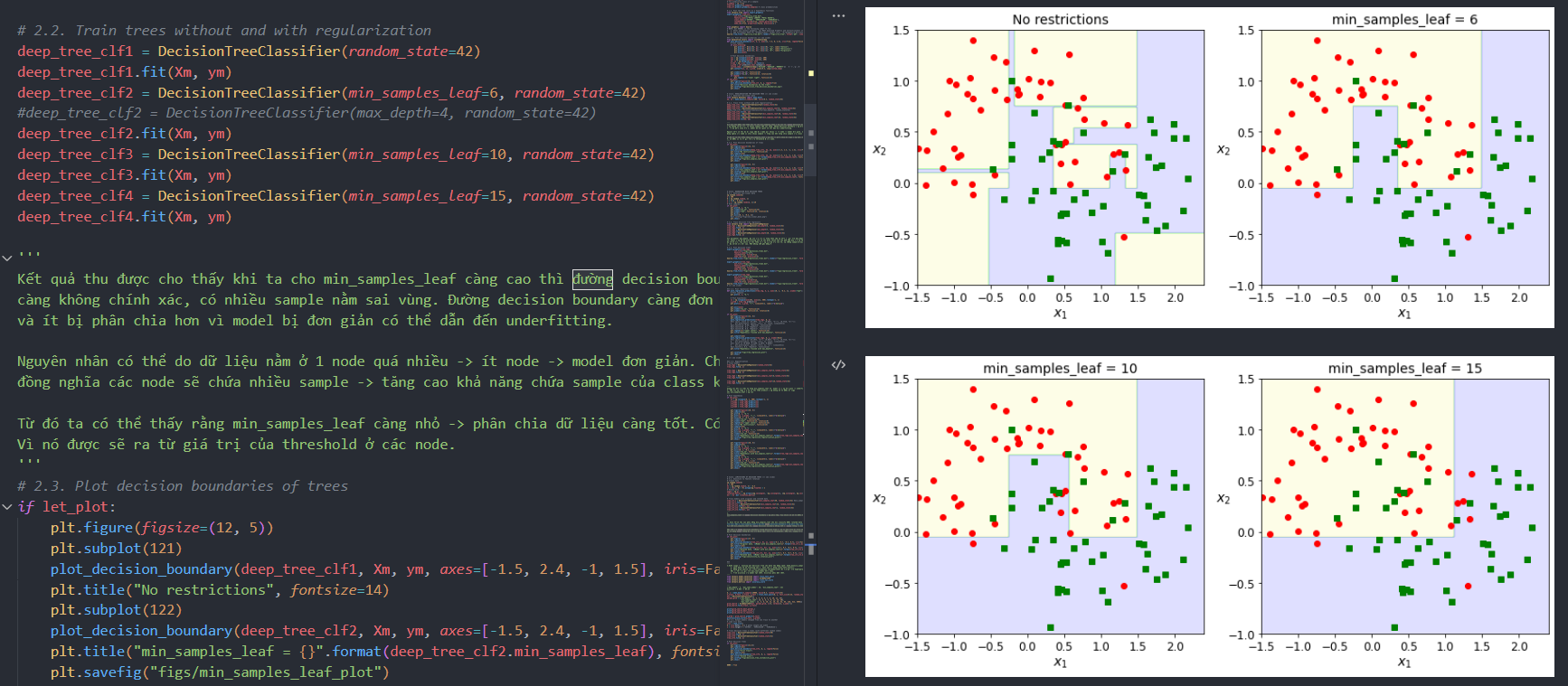
Ví dụ ta có n là giống nhau

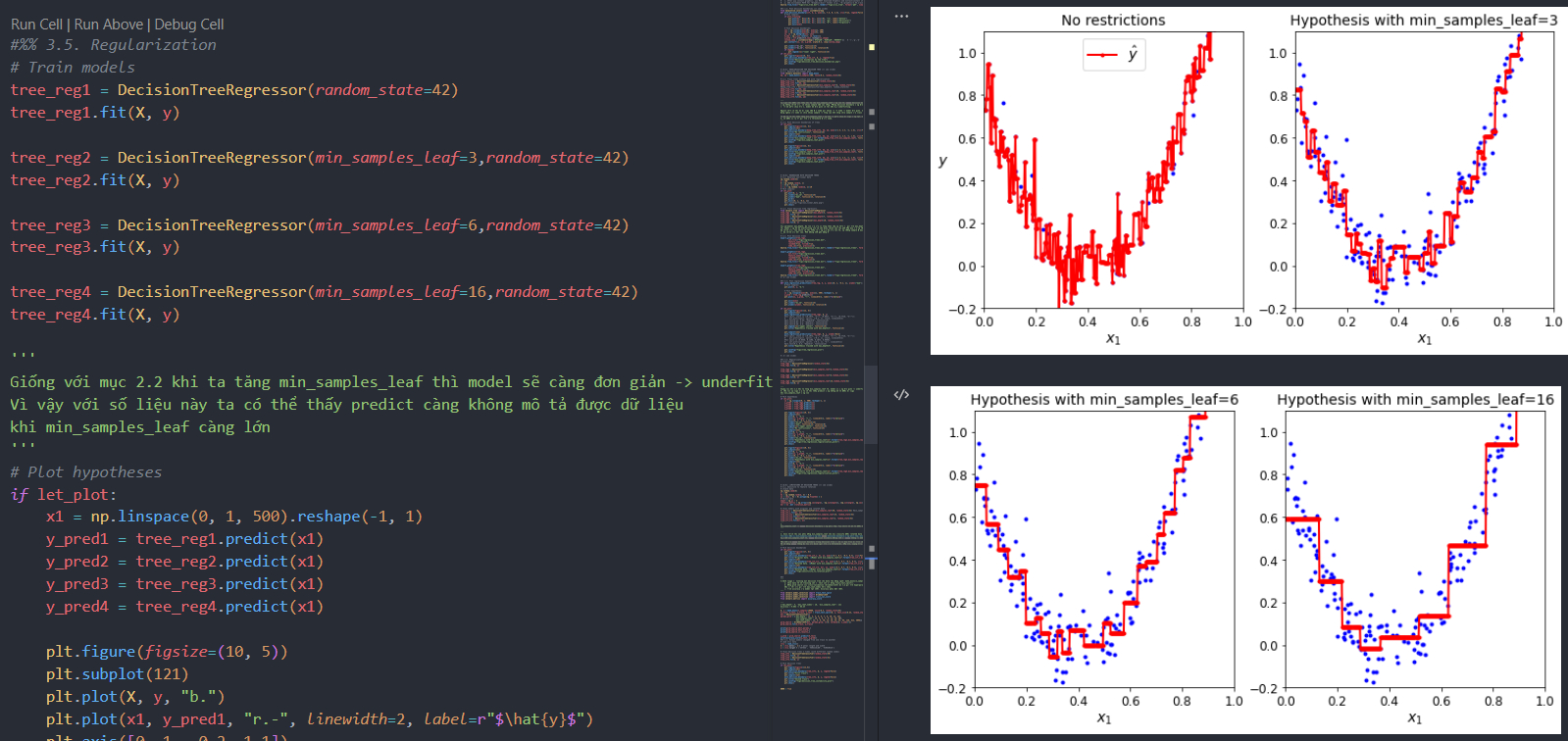
Thời gian với 10 triệu sample: => t = 700 phút

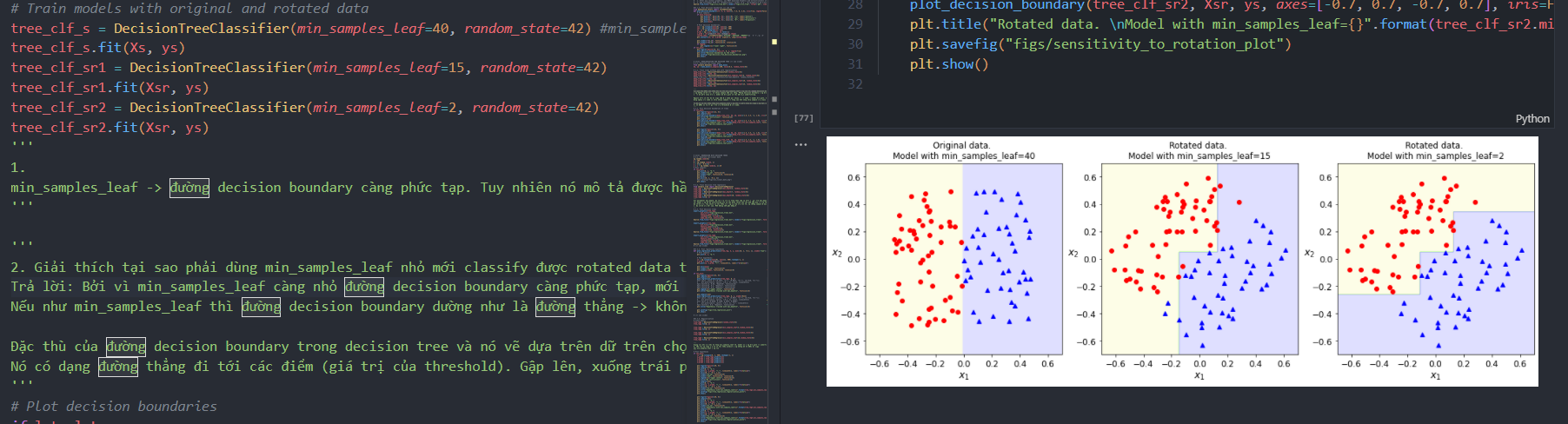
Kết luận ta tốn khoảng 11h40p để huấn luyện

*[Phần lập trình]*

1. Thay đổi các regularization hyperparameters trong **mục 2.2, 3.2, 3.5, 4.1**, và huấn luyện lại model. Giải thích kết quả thu được.



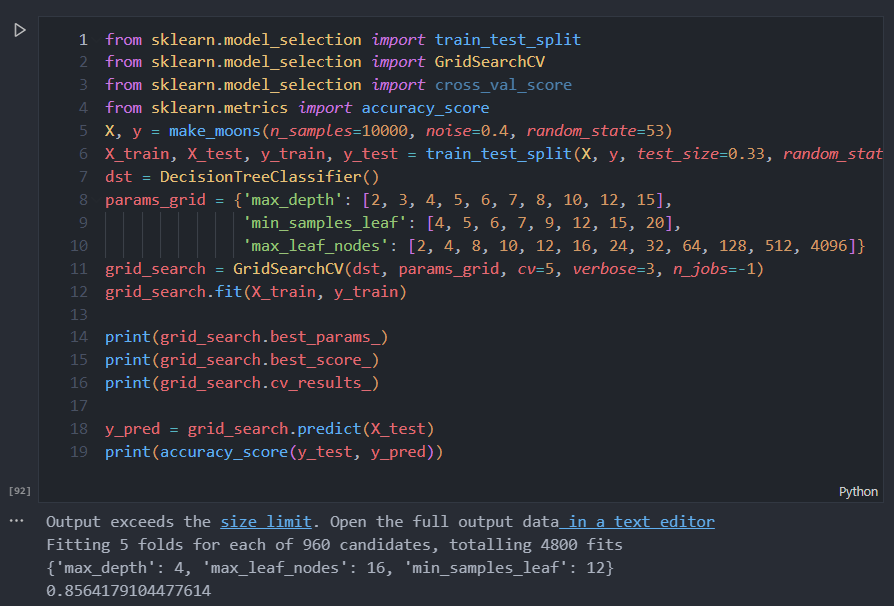




1. *[Không bắt buộc]* Giải thích tại sao phải dùng **min\_samples\_leaf nhỏ** mới classify được rotated data trong **mục 4.1**.   
   *Gợi ý:* vẽ decision trees được huấn luyện sử dụng rotated data.
2. Huấn luyện và tunning một decision tree cho data tạo bằng lệnh: make\_moons(n\_samples=10000, noise=0.4). Hướng dẫn và yêu cầu:  
   a. Chia data thành training set và test set sử dụng train\_test\_split().  
   b. Dùng grid search với cross-validation (GridSearchCV) để tìm giá trị hyperparameters cho DecisionTreeClassifier.   
   *Gợi ý:* thử các giá trị max\_leaf\_nodes khác nhau.   
   c. Tính accuracy của model tốt nhất. **Accuracy phải đạt >85%.**

{'max\_depth': 4, 'max\_leaf\_nodes': 16, 'min\_samples\_leaf': 12}

Accuracy = 0.852 -> 85.2%

**