6) (2 điểm) Tìm hiểu về vấn đề Overfiting và trong chương trình trên đã áp dụng như thế nào? (phần này cần trình bày bằng slide)

* Overfitting đề cập đến một hành vi không mong muốn của một thuật toán học máy được sử dụng để tạo mô hình dự đoán. .Hay nói cách khác là mô hình rất hợp lý, rất khớp với tập huấn luyện nhưng khi đem ra dự đoán với dữ liệu mới thì lại không phù hợp. Nó ngụ ý rằng mô hình đã diễn giải quá mức các mẫu trong dữ liệu đào tạo và được điều chỉnh quá tốt.
* Một mô hình được coi là tốt (fit) nếu cả train và test đều thấp. Nếu train thấp nhưng test cao, ta nói mô hình bị overfitting. Nếu train error và test cao, ta nói mô hình bị underfitting.
* Nguyên nhân của Overfitting: do ta chưa đủ dữ liệu để đánh giá, dự đoán hoặc do tệp dữu liệu của ta quá phức tạp hoặc quá nhỏ. Mô hình bị quá phức tạp khi mà mô hình của ta sử dụng cả những nhiễu lớn trong tập dữ liệu để học, dẫn tới mât tính tổng quát của mô hình.
* Overfitting có thể giải quyết bằng các biện pháp sau:
  + Bằng cách chọn loại mô hình đơn giản hơn (mô hình đa thức bậc thấp hơn thay vì mạng nơ ron sâu phức tạp) hoặc kiến trúc mô hình khác (tức là kiểu mạng nơ ron khác)
  + Bằng cách giảm các tham số của mô hình ( mô hình đa thức bậc thấp hoặc mô hình tuyến tính thay vì mô hình đa thức bậc cao)
  + Bằng cách thêm các điểm dừng sớm vào mô hình để việc học được dừng lại ở tỷ lệ lỗi được nhắm mục tiêu.
  + Bằng cách giảm các số tính năng hoặc chọn tính năng phù hợp hơn cho sử dụng.
  + Đơn giản bằng cách thu thập thêm dữ liệu miễn là nó có sẵn và sức mạnh tính toán không phải là vấn đề
  + Bằng cách giảm nhiễu (sửa lỗi cân bằng và loại bỏ các ngoại lệ)
* Trong dữ liệu post-operative.data:
  + Ở đây ta sẽ thử thực hiện phân tích mô hình học máy và xem có bị đánh lừa bởi kết quả không
  + Ví dụ được thể hiện sau đây là thay đổi số lượng hàm xóm của k-các láng giềng gần nhất, mà ta có thể triển khai bằng cách sử dụng lớp KNeighborsClassifier định cấu hình thông qua tham số “n\_neighbors”:

# xác định danh sách để thu thập dữ liệu

train\_scores, test\_scores = list(), list()

values = [i for i in range(1, 61)]

# Chạy thay đổi láng giềng

for i in values:

modelKnn = KNeighborsClassifier(n\_neighbors=i)

modelKnn.fit(X\_train\_Knn, y\_train\_Knn)

# đánh giá trên tệp post operative

train\_yhat = modelKnn.predict(X\_train\_Knn)

train\_acc = accuracy\_score(y\_train\_Knn, train\_yhat)

train\_scores.append(train\_acc)

# đánh giá trên tệp dữ liệu test

test\_yhat = modelKnn.predict(X\_test\_Knn)

test\_acc = accuracy\_score(y\_test\_Knn, test\_yhat)

test\_scores.append(test\_acc)

# Tóm tắt

print('\_%d, train: %.3f, test: %.3f' % (i, train\_acc, test\_acc))

# Hình minh họa

pyplot.plot(values, train\_scores, '-o', label='Post-operative')

pyplot.plot(values, test\_scores, '-o', label='Test')

pyplot.legend()

pyplot.show()

* + Kết quả:

\_

1, train: 0.950, test: 0.667

\_2, train: 0.667, test: 0.778

\_3, train: 0.717, test: 0.667

\_4, train: 0.633, test: 0.852

\_5, train: 0.683, test: 0.667

\_6, train: 0.650, test: 0.778

\_7, train: 0.633, test: 0.667

\_8, train: 0.617, test: 0.815

\_9, train: 0.650, test: 0.704

\_10, train: 0.633, test: 0.815

\_11, train: 0.583, test: 0.815

\_12, train: 0.633, test: 0.815

\_13, train: 0.650, test: 0.852

\_14, train: 0.633, test: 0.815

\_15, train: 0.633, test: 0.815

\_16, train: 0.633, test: 0.852

\_17, train: 0.650, test: 0.889

\_18, train: 0.633, test: 0.852

\_19, train: 0.650, test: 0.852

\_20, train: 0.633, test: 0.852

\_21, train: 0.683, test: 0.815

\_22, train: 0.633, test: 0.852

\_23, train: 0.633, test: 0.852

\_24, train: 0.633, test: 0.852

\_25, train: 0.633, test: 0.852

\_26, train: 0.633, test: 0.852

\_27, train: 0.617, test: 0.815

\_28, train: 0.617, test: 0.852

\_29, train: 0.617, test: 0.815

\_30, train: 0.633, test: 0.852

\_31, train: 0.633, test: 0.852

\_32, train: 0.633, test: 0.852

\_33, train: 0.633, test: 0.852

\_34, train: 0.633, test: 0.852

\_35, train: 0.633, test: 0.852

\_36, train: 0.633, test: 0.852

\_37, train: 0.633, test: 0.852

\_38, train: 0.633, test: 0.852

\_39, train: 0.633, test: 0.852

\_40, train: 0.633, test: 0.852

\_41, train: 0.633, test: 0.852

\_42, train: 0.633, test: 0.852

\_43, train: 0.633, test: 0.852

\_44, train: 0.633, test: 0.852

\_45, train: 0.633, test: 0.852

\_46, train: 0.633, test: 0.852

\_47, train: 0.633, test: 0.852

\_48, train: 0.633, test: 0.852

\_49, train: 0.633, test: 0.852

\_50, train: 0.633, test: 0.852

\_51, train: 0.633, test: 0.852

\_52, train: 0.633, test: 0.852

\_53, train: 0.633, test: 0.852

\_54, train: 0.633, test: 0.852

\_55, train: 0.633, test: 0.852

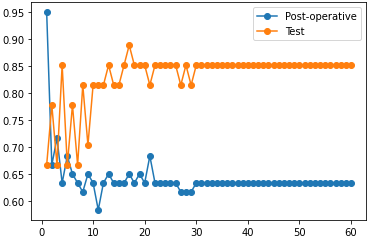
\_56, train: 0.633, test: 0.852

\_57, train: 0.633, test: 0.852

\_58, train: 0.633, test: 0.852

\_59, train: 0.633, test: 0.852

\_60, train: 0.633, test: 0.852



* Nhận xét:
  + - Chúng ta thấy rằng độ chính xác ở tệp dữ liệu train bắt đầu ở 0.95 gần như hoàn hảo và có xu hướng giảm dần với sự gia tăng của n láng giềng. Giữ mức ở n láng giềng = 30 và không có sự thay đổi sau đó.
    - Bên cạnh đó ta thấy rằng độ chính xác ở tệp dữ test được cài thiện dần với sự gia tăng của n láng giềng . Giữ mức ở n láng giềng = 30 và không có sự thay đổi sau đó.
    - Ở hình độ thị, ta thấy ban đầu 2 tệp dữ liệu tăng giảm. Sau đó thì 2 tệp dữ liệu gần như tách ra thành song song và không ảnh hưởng với nhau.
    - Qua đây ta thấy rằng dữ liệu post-operative có train thấp và test cao nên ta bị overfitting

Link tham khảo:

<https://codetudau.com/danh-gia-model-cua-machine-learning-precision-recall-bias-variance-cross-validation/>

<https://ichi.pro/vi/gioi-thieu-ve-overfitting-underfitting-va-data-mismatch-trong-xay-dung-he-thong-may-hoc-91195616954933>

<https://dominhhai.github.io/vi/2017/12/ml-overfitting/#1-2-qu%C3%A1-kh%E1%BB%9Bp-overfitting>

<https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html>

<https://machinelearningmastery.com/overfitting-machine-learning-models/>