# Báo cáo thực tập ngày 26/06/2024

Nguyễn Tiến Đạt - Thực tập sinh AI

## 1 Hạn chế của decision tree

Cây quyết định là một mô hình khá nổi tiếng hoạt động trên cả hai lớp bài toán phân loại và dự báo của học có giám sát, có độ chính xác khá cao. Tuy nhiên vẫn tồn tại những hạn chế lớn:

- **Overfitting:** Xảy ra khi số lượng các đặc trưng để hỏi lớn. Khi độ sâu của cây quyết định không được giới hạn thì có thể tạo ra các leaf-node chỉ có một vài quan sát.
- Với một bộ dataset lớn, độ sâu của cây quyết định khi bị giới hạn (để hạn chế overfitting) thường sẽ bỏ qua những biến quan trọng.
- Một mô hình thì chỉ tạo ra một kịch bản dự báo duy nhất, nếu mô hình không tốt thì kết quả sẽ bị lệch.

## 2 Bộ phân loại Biểu quyết (Voting)

Giả sử ta đã huấn luyện xong một mô hình giải quyết vấn đề nhị phân có xác suất trả về nhãn 1 là 0.6. Đây là một kết quả không quá cao và ta vẫn cần phải kiểm chứng, vì vậy ta xây dựng thêm nhiều mô hình hơn để tham vấn. Ta đánh giá trên 9 mô hình khác nhau và tiến hành *bầu cử* kết quả trả về giữa chúng. Kết quả trả về từ 9 mô hình thì có 8 mô hình dự báo nhãn 1 và một mô hình dự bão nhãn 0. Như vậy căn cứ vào kết quả *bầu cử* ta có thể tin cậy rằng nhãn dự báo ảnh 1 là đúng.

Ta đặt câu hỏi phức tạp cho hàng nghìn người được chọn ngẫu nhiên, và tổng hợp các câu trả lời của họ. Trong nhiều trường hợp, câu trả lời có được thường tốt hơn câu trả lời của một chuyên gia. Hiện tượng này được gọi là *trí tuệ đám đông*. Tương tự như vậy, nếu kết hợp một nhóm mô hình phân loại hoặc hồi quy, dự đoán tổng hợp thường sẽ tốt hơn dự đoán riêng lẻ của bộ dự đoán tốt nhất.

Một nhóm các bộ phân loại được gọi là *ensemble*, và vì thế kỹ thuật trên được gọi là *Học Ensemble*.

Đoạn mã sau khởi tạo và huấn luyện một bộ phận biết quyết với scikit-learn, là ensemble của ba bộ phận khác nhau (Tập huấn luyện là iris_data):

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import VotingClassifier

# Three model in ensemble learning
log_clf = LogisticRegression()
svm_clf = SVC()
tree_clf = DecisionTreeClassifier(max_depth=3)

voting_clf = VotingClassifier(
  estimators=[('lr', log_clf), ('svc', svm_clf), ('
    tree_clf', tree_clf)],
  voting='hard'
)

voting_clf.fit(X_train, y_train)

for clf in (log_clf, svm_clf, tree_clf, voting_clf):
  clf.fit(X_train, y_train)
  y_pred = clf.predict(X_test)
```

```
LogisticRegression 0.7
SVC 0.8666666666666667
DecisionTreeClassifier 0.8333333333333334
VotingClassifier 0.8666666666666667
```

Có thể thấy bộ phân loại biểu quyết hoạt động tốt hơn một chút so với mỗi bộ phân loại riêng lẻ.

## 3 Bagging

Một cách để có được các bộ phân loại đa dạng là sử dụng cùng một thuật toán huấn luyện cho model nhưng train chúng trên các tập con ngẫu nhiên của tập huấn luyện. Để các mẫu được sử dụng nhiều lần cho cùng một bộ dự đoán, ta sử dụng *bagging* (viết tắt của bootstrap aggregating).

Sau khi tất cả các bộ dự đoán riêng lẻ đã được huấn luyện, ensemble có thể đưa ra dự đoán trên một mẫu mới bằng cách tổng hợp các kết quả từ bộ dự đoán riêng lẻ. Hàm tổng hợp thường là *statistical mode* cho các tác vụ phân loại, hoặc lấy trung bình cho các tác vụ hồi quy. Mỗi bộ dự đoán riêng lẻ thường có độ chênh lệch cao hơn khi chúng được train trên tập huấn luyện gốc, tuy nhiên việc tổng hợp sẽ giảm cả độ chênh lệch lẫn phương sai.

Scikit-learn cung cấp một API cho phương pháp *bagging* với lớp *BaggingClassifier* hoặc *BaggingRegressor*.

Đoạn mã dưới đây huấn luyện một ensemble gồm 500 Cây Quyết Định: mỗi bộ phân loại được huấn luyện trên 100 mẫu ngẫu nhiên có hoàn lại từ tập huấn luyện. Tham số *n_jobs* chỉ định số lõi CPU mà scikit-learn có thể sử dụng để huấn luyện và dự đoán (-1 cho phép sử dụng tất cả các lõi)

```
from sklearn.ensemble import BaggingClassifier

bag_clf = BaggingClassifier(
  DecisionTreeClassifier(),
  n_estimators=500,
  max_samples=100,
  bootstrap=True,
  n_jobs=-1,
)
```

## 4 Đánh giá Out-of-bag

Phương pháp bỏ túi cho phép lấy mẫu lặp lại các quan sát trên tập huấn luyện nên sẽ có một lượng lớn những quan sát chưa được đưa vào các tập huấn luyện con. Tập hợp những mẫu này gọi là mẫu nằm ngoài túi (out of bag), được viết tắt là oob. Những dữ liệu này được lựa chọn ngẫu nhiên, độc lập và hoàn toàn không được học từ mô hình nên có thể được sử dụng để đánh giá mô hình tương đương với một tập kiểm tra. Trong **sklearn.ensemble.BaggingClassifier** chúng ta sử dụng thêm tuỳ chọn **oob_score=True** để đánh giá mô hình dựa trên các mẫu oob.

```
from sklearn.ensemble import BaggingClassifier

bag_clf = BaggingClassifier(
  DecisionTreeClassifier(),
  n_estimators=500,
  max_samples=100,
  bootstrap=True,
  oob_score=True,
  n_jobs=-1,
)

bag_clf.fit(X_train, y_train)

print('Out of bag accuracy: ', bag_clf.oob_score_)
```

```
Out of bag accuracy:  0.9666666666666667
```

Đánh giá từ các quan sát oob cho thấy mô hình đạt được độ chính xác là 96.67%. Còn trên tập kiểm tra độ chính xác đạt được là:

```
y_pred_bag = bag_clf.predict(X_test)
print('Out of bag accuracy: ', accuracy_score(y_test
    , y_pred_bag))
```

```
Out of bag accuracy:  0.9333333333333333
```

Như vậy kết quả đánh giá từ các mẫu *oob* và tập kiểm tra là gần như bằng nhau. Bởi bản chất đây là những quan sát độc lập mà mô hình chưa được học. Về cơ bản thì nó cũng tương tự như những quan sát trên tập kiểm tra. Khi đánh giá mô hình dựa trên tập *oob* sẽ tạo ra một kết quả khá khách quan giúp nhận biết hiện tượng *Overfitting*.

## 5 RANDOM FOREST

Mô hình *rừng cây* (Random Forest) sẽ áp dụng cả hai phương pháp Ensemble và Bagging. Thứ tự của quá trình tạo thành một mô hình *rừng cây* như sau:

- Lấy mẫu tái lập một cách ngẫu nhiên từ tập huấn luyện để tạo thành một tập dữ liệu con.
- Lựa chọn ra ngẫu nhiên *d* biến và xây dựng mô hình cây quyết định dựa trên những biến này và tập dữ liệu con ở bước 1. Chúng ta sẽ xây dựng nhiều cây quyết định nên bước 1 và 2 sẽ lặp lại nhiều lần.
- Thực hiện *bầu cử* hoặc *lấy trung bình* giữa các cây quyết định để đưa ra dự báo.

Kết quả dự báo từ mô hình rừng cây là sự kết hợp của nhiều cây quyết định nên chúng tận dụng được trí thông minh đám đông và giúp cải thiện độ chính xác so với chỉ sử dụng một mô hình cây quyết định.

Nếu như mô hình cây quyết định thường bị nhạy cảm với dữ liệu *ngoại lai (outlier)* thì mô hình rừng cây được huấn luyện trên nhiều tập dữ liệu con khác nhau, trong đó có những tập được loại bỏ dữ liệu ngoại lai, điều này giúp cho mô hình ít bị nhạy cảm với dữ liệu ngoại lai hơn.

Sự kết hợp giữa các cây quyết định giúp cho kết quả ít bị chệch và phương sai giảm. Như vậy chúng ta giảm thiểu được *overfitting* ở mô hình rừng cây, một điều mà mô hình cây quyết định thường xuyên gặp phải.

Cuối cùng các bộ dữ liệu được sử dụng từ những *cây quyết định* đều xuất phát từ dữ liệu huấn luyện nên quy luật học được giữa các *cây quyết định* sẽ gần tương tự như nhau và tổng hợp kết quả giữa chúng không có xu hướng bị chệch.

## 6 HUẤN LUYỆN RANDOM FOREST

Thay vì phải tạo một **BaggingClassifier** và truyền vào một **DecisionTreeClassifier**, sử dụng lớp **RandomForestClassifier** sẽ tiện lợi hơn và tối ưu hơn (tương tự có **RandomForestRegressor** cho tác vụ hồi quy).

Đoạn mã sau sử dụng tất cả lõi CPU sẵn có để huấn luyện model Random Forest Classifier với 500 cây quyết định (mỗi cây tối đa 16 nút):

```
from sklearn.ensemble import RandomForestClassifier

# H u n  l u y n  m  h nh  t r n  t p  train
rdf_clf = RandomForestClassifier(
    n_estimators=500,
    max_leaf_nodes = 16,
    n_jobs=-1
)

rdf_clf.fit(X_train, y_train)

#  D  b o  t r n  t p  test
y_pred = rdf_clf.predict(X_test)
scores = accuracy_score(y_pred, y_test)
print('RandomForest Accuracy: ', scores)
```

```
RandomForest Accuracy:  0.9
```