GVHD: LÊ NGỌC THÀNH, NGUYỄN NGỌC THẢO

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
THÀNH PHỐ HỒ CHÍ MINH

BÀI TẬP 3: PHÂN LỚP DỮ LIỆU

I. THÔNG TIN SINH VIÊN

Họ và tên: **TRẦN NHẬT HUY**

Mssv: **1612272**

Email: nhathuy13598@gmail.com

Sđt: **0354 878 677**

II. BẢNG BÁO CÁO CÔNG VIỆC

STT	CÁC CÂU HỎI	MỨC ĐỘ HOÀN THÀNH	GHI CHÚ
1	Thiết lập bảng thống kê	100%	
2	Báo cáo hiệu quả 2 thuật toán	100%	
3	Vẽ đồ thị thể hiện độ chính xác phân lớp	100%	
4	Tham số numFolds có vai trò gì trong J48	100%	
5	Hiệu quả tỉa nhánh giảm lỗi trên cây quyết định	100%	
6	Đánh giá độ chính xác của mô hình được chọn	100%	
7	Mô tả lý thuyết về thuật toán được chọn	100%	
8	Báo cáo bộ tham số được dùng để thực nghiệm	100%	

III. CHI TIẾT BÀI LÀM

1. Bảng thống kê độ chính xác

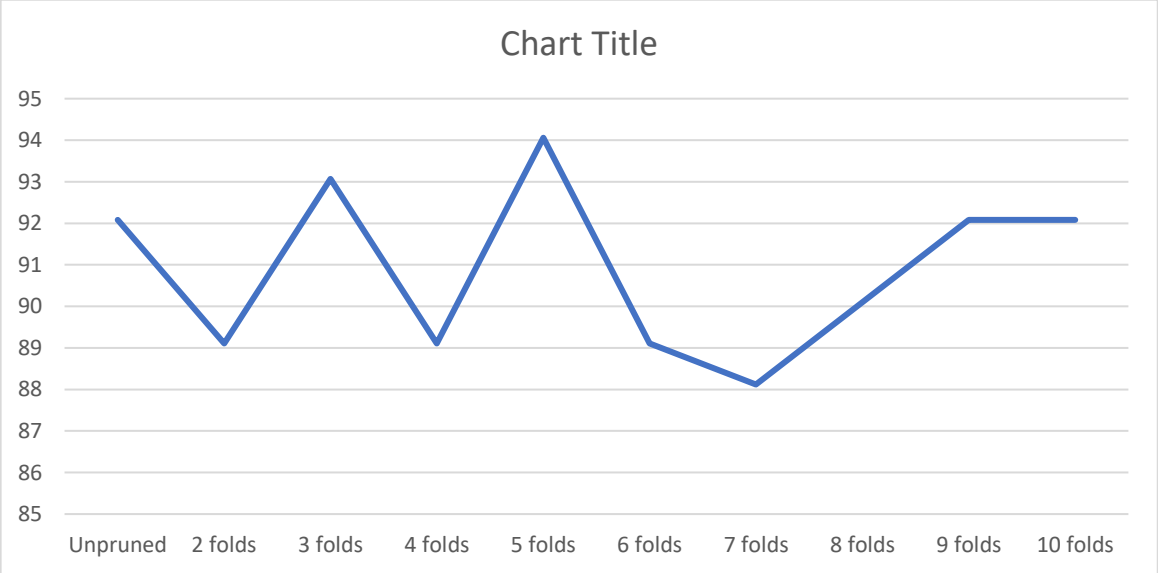
Giải thuật	Accuracy	Detailed Accuracy By Class							
		Class edible				Class poisonous			
		TP Rate	FP Rate	Precision	Recall	TP Rate	FP Rate	Precision	Recall
LR	99.7537%	0.998	0.003	0.998	0.998	0.997	0.002	0.997	0.997
J48	99.8768%	1.000	0.003	0.998	1.000	0.997	0.000	1.000	0.997
IBk (KNN=1)	99.8768%	0.998	0.000	1.000	0.998	1.000	0.002	0.997	1.000
IBk (KNN=4)	99.1379%	0.998	0.005	0.995	0.988	0.995	0.012	0.998	0.995

2. Hiệu quả của hai giải thuật A và B

Với bảng đã tìm được, ta xác định được: **A** là thuật toán **J48**, **B** là thuật toán **IBk (KNN = 1)**

Tuy hai thuật toán **A** và **B** có độ chính xác như nhau nhưng chúng ta phải xét lại bài toán mà chúng ta cần giải. Bài toán này dùng để phân biệt 2 loại nấm: **ăn được** và **có độc**. Vậy nếu một loại nấm **có độc** nhưng thuật toán lại dự đoán là **ăn được** thì sẽ nghiêm trọng hơn việc một loại nấm **ăn được** nhưng thuật toán dự đoán là **có độc**. Vậy chúng ta sẽ tập trung vào **TP rate** cho lớp **poisonous**, tỉ lệ nấm **có độc** được dự đoán là **có độc**, thuật toán nào có **TP rate** cho lớp **poisonous** lớn hơn thuật toán còn lại thì đó là thuật toán tốt hơn. Ở bài này thì thuật toán **B** là thuật toán tốt hơn

3. Đồ thị thể hiện độ chính xác



4. Tham số numFolds

Tham số **numFolds** trong **J48** được dùng để xác định số lượng dữ liệu dùng cho việc tỉa nhánh. Nếu set **numFolds = k** có nghĩa là ta lấy **k folds** để tỉa nhánh và các fold còn lại để xây dựng cây. Sử dụng tham số **numFolds** sẽ giúp cho cây quyết định đơn giản, gọn gàng và trực quan hơn

5. Hiệu quả tỉa nhánh giảm lỗi trên cây quyết định

Dựa vào đồ thị ta thấy việc tỉa nhánh thường làm cho độ chính xác giảm. Tuy nhiên, việc tỉa nhánh sẽ làm cho cây quyết định đơn giản hơn, từ đó tránh hiện tượng **overfitting** rất dễ xảy ra khi lựa chọn mô hình cây quyết định

6. Đánh giá độ chính xác của mô hình được chọn

Ở đây, em sẽ chọn thuật toán **Random Forest**

			Detailed Accuracy By Class	
	Accuracy		95.8468%	
Class	TP Rate	FP Rate	Precision	Recall
A	0,997	0,001	0,987	0,997
B	0,950	0,005	0,890	0,950
C	0,961	0,001	0,983	0,961
D	0,967	0,004	0,913	0,967
E	0,961	0,002	0,944	0,961
F	0,934	0,002	0,948	0,934
G	0,935	0,002	0,951	0,935
H	0,872	0,002	0,947	0,872
I	0,936	0,001	0,968	0,936
J	0,937	0,001	0,963	0,937
K	0,942	0,003	0,922	0,942
L	0,962	0,000	0,993	0,962
M	0,984	0,001	0,968	0,984
N	0,962	0,001	0,971	0,962
O	0,949	0,002	0,940	0,949
P	0,954	0,001	0,963	0,954
Q	0,962	0,003	0,938	0,962
R	0,940	0,003	0,924	0,940
S	0,965	0,001	0,976	0,965
T	0,980	0,001	0,986	0,980
U	0,977	0,001	0,972	0,977
V	0,954	0,001	0,965	0,954
W	0,990	0,001	0,987	0,990
X	0,971	0,002	0,957	0,971
Y	0,981	0,001	0,984	0,981
Z	0,983	0,000	0,988	0,983

7. Mô tả giải thuật được chọn

Random Forest là một thuật toán học có giám sát. **Random Forest** tạo ra một rừng với mỗi cây là một cây quyết định và tạo nó ra một cách ngẫu nhiên. **Random Forest** có thể được sử dụng cho bài toán phân lớp và bài toán regression. Nếu chúng ta đưa một tập huấn luyện gồm các đặc trưng và nhãn, thì thuật toán cây quyết định như **J48** sẽ tạo ra duy nhất một cây để dự đoán. Đối với **Random Forest**, thuật toán này sẽ ngẫu nhiên chọn ra các subset khác nhau, mỗi subset sẽ tạo ra một cây quyết định

Ưu điểm:

- Có thể được dùng cho bài toán Classification và Regression
- Có thể sử dụng ngay cả khi có missing values
- Chống overfitting

Khuyết điểm:

- Tốt cho bài toán Classification nhưng không tốt cho Regression
- Ít có khả năng kiểm soát được thuật toán làm gì
- Nếu số lượng cây muốn tạo lớn thì sẽ khiến thuật toán chạy chậm và không phù hợp cho việc dự đoán theo thời gian thực

Thuật toán **Random Forest** hoạt động như sau: Với mỗi cây trong rừng, chúng ta chọn ra một mẫu bootstrap thứ i $S^{(i)}$ từ S. Sau đó, chúng ta tạo ra cây quyết định từ mẫu bootstrap thứ i này với thuật toán tạo cây quyết định được chỉnh sửa. Thuật toán tạo cây quyết định được chỉnh sửa như sau: với mỗi node của cây, thay vì chúng ta sẽ kiểm tra toàn bộ đặc trưng để quyết định xem đặc trưng nào dùng để chia cây thì chúng ta chọn ngẫu nhiên tập đặc trưng con $f \subseteq F$, với F là tập các đặc trưng. Cây quyết định sẽ được chia dựa vào đặc trưng tốt nhất trong f thay vì tập F. Việc quyết định đặc trưng nào để chia cây thường tốn chi phí nhất.

Tập huấn luyện: $S := (x_1, y_1), \ldots, (x_n, y_n)$

Tập đặc trưng: F

Số lượng cây trong rừng: B

Hàm *RANDOMFOREST(S, F)* dùng để tạo rừng với rừng khởi tạo H là rỗng

Hàm *RANDOMIZEDTREELEARN(S, F)* dùng để tạo cây dựa vào mẫu bootstrap $S^{(i)}$, f là tập đặc trưng con của F, f nhỏ hơn F lớn rất nhiều

Algorithm 1 Random Forest

Precondition: A training set $S := (x_1, y_1), \ldots, (x_n, y_n)$, features F, and number of trees in forest B.

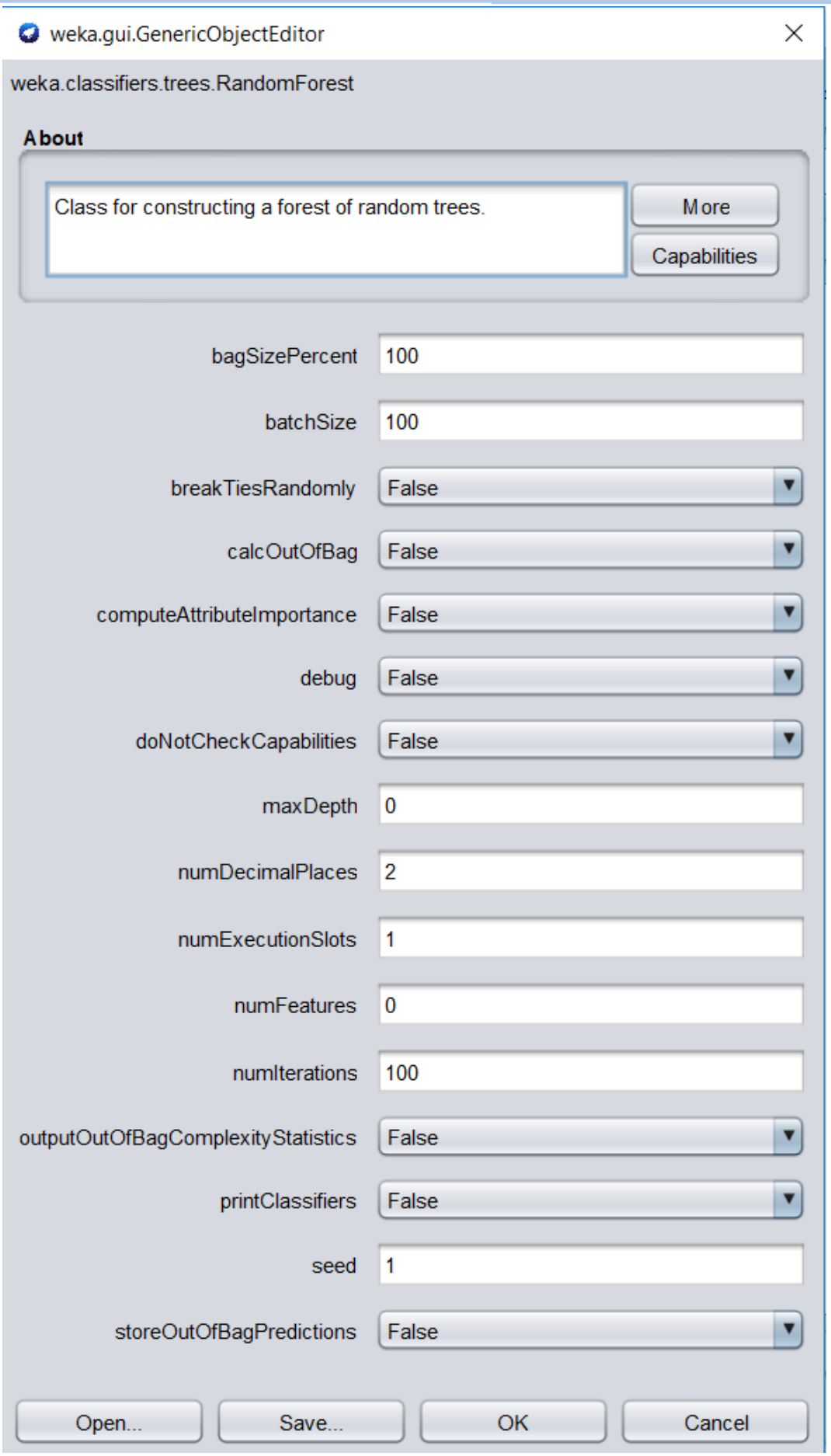
```
function RandomForest(S, F)
    H ← ∅
    for i ∈ 1, ..., B do
        S^(i) ← A bootstrap sample from S
        h_i ← RandomizedTreeLearn(S^(i), F)
        H ← H ∪ {h_i}
    end for
    return H
end function
function RandomizedTreeLearn(S, F)
    At each node:
        f ← very small subset of F
        Split on best feature in f
    return The learned tree
end function
```

Để dự đoán một mẫu chưa biết đưa vào, chúng ta sẽ duyệt từng cây trong rừng và thống kê xem số lượng các cây trả về cùng một lớp nào lớn nhất thì mẫu đó sẽ được phân loại vào lớp này

Lý do chọn thuật toán **Random Forest**: **Random Forest** là thuật toán nằm trong nhóm **Ensemble Methods**. Kỹ thuật này sử dụng kết hợp nhiều model lại với nhau để tăng độ chính xác. Đối với dataset này, thuật toán **KNN** với **k = 1** đã cho một kết quả cực kỳ tốt và theo kết quả thực nghiệm được nghiên cứu trên nhiều bài báo thì thuật toán **Random Forest** luôn cho một kết quả tốt hơn thuật toán **KNN** với hầu như mọi dataset

8. Báo cáo bộ tham số

Bộ tham số đã sử dụng


weka.gui.GenericObjectEditor
weka.classifiers.trees.RandomForest
About
Class for constructing a forest of random trees.
More
Capabilities
bagSizePercent
100
batchSize
100
breakTiesRandomly
False
calcOutOfBag
False
computeAttributeImportance
False
debug
False
doNotCheckCapabilities
False
maxDepth
0
numDecimalPlaces
2
numExecutionSlots
1
numFeatures
0
numIterations
100
outputOutOfBagComplexityStatistics
False
printClassifiers
False
seed
1
storeOutOfBagPredictions
False
Open...
Save...
OK
Cancel

maxDepth: là độ sâu của cây, **0** nghĩa là độ sâu không giới hạn. Nếu set một giá trị khác **0** sẽ khiến cho thuật toán giảm độ chính xác vì với **Random Forest** ta muốn cây được xây dựng một cách tối đa

numFeatures: Số lượng features được chọn ngẫu nhiên từ tập features gốc. Số lượng này phải cực nhỏ so với số lượng feature gốc. Số lượng càng ít thì thời gian gian xây dựng cây sẽ giảm. Nếu set là 0 thì số lượng feature được chọn sẽ tùy vào dataset

numIterations: Số lượng cây trong rừng. Số này càng lớn thì càng chính xác nhưng thời gian xây dựng rừng và phân loại sẽ lâu