ECE 6254 : Edible Mushrooms Identificator

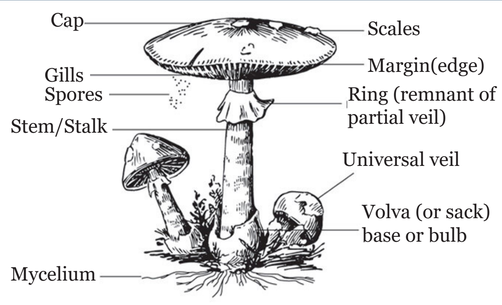
Machine Learning Project

By BENOIT T. (ECE Department), DERKAOUI H. (ECE Department), SOUBIGOU E. (ECE Department)

Abstract

Each year, a too large panel of foolhardy mushroom pickers are intoxicated, and in certain cases, pass away. Simultaneously, each year, drugstores and physicians must manage these cases, without the most modern tools. Therefore, in response to this situation, we have decided to develop a piece of machine learning software able to identify the harmfulness of a mushroom. The methods selected were the ***tree-Bagging***, the ***random forest***, and the ***boosting method***.

To understand our approach, we firstly propose to select a rich data set [of mushrooms] containing more than *8124 mushrooms* described by *23 characteristics* **(cf. Table 1)**. It is easy to notice that all the feature are qualitative data 1. Then, after treating the dataset, we assumed that the majority of people are not experts in mushrooms, and some features are hard to describe for *beginner "shroomers"*. Thus we decided to only keep features which are the easiest to differentiate for us **(c.f. in bold font Tab. 1)**.



cap-shape	**cap-surface**	**cap-color**	bruises
odor	gill-attachement	gill-spacing	gill-size
gill-color	stalk-shape	stalk-root	stalk-surface-above-ring
stalk-surface-below-ring	stalk-color-above-ring	**population**	veil-type
veil-color	ring-number	ring-type	spore-print-color

Table 1: List of the different characteristics

Preliminary Analyses & Naïve methode

Preliminary studies

For our first operation we report some informative points in visualizing features on each class edible/poisonous **(Tab. 2)**. Hence, we can noticy that the odor feature and the spore-print-color are interesting criteria to classify edible/poisonous mushrooms.
However, for shrooming beginners, these characteristics are too hard to distinguish. Hence we first decided not including them in our algorithm.

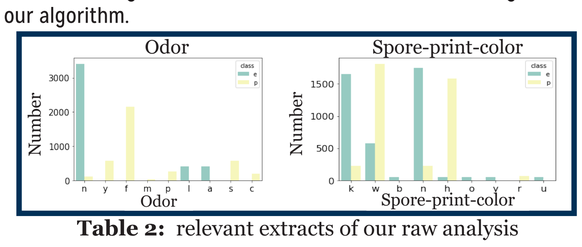


Table 2: relevant extracts of our raw analysis

A Naïve Approach: SVM

In the first place, we used as naïve method, a simple *SVM*. We optimized the value of C and obtained some interesting results **(c.f. Tab. 3)**.
We can then sum up by saying that even though we perform some good results the convergence time is excessively long. Therefore, due to this aspect, it is essential to apply another type of methods.

	deg 1	deg 2	deg 3
C	4.64	10^4	774
Score	0.832	0.90	0.96
Support vectors	2770	1591	625

Table 3: Caption

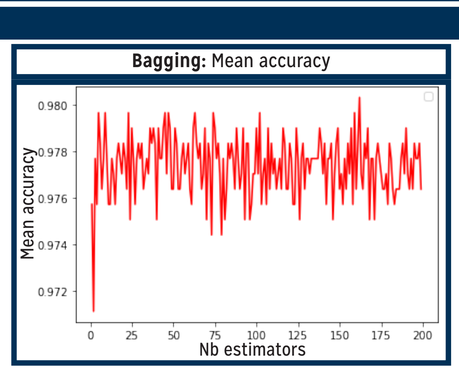
Method 1: Bagging

The *bagging method*, also known as «bootstrap aggregating», has as a goal to reduce the variance of a classifier by generating it thanks to the average of set of classifiers. The principles is to choose a number N of trees estimator that will be trained on N estimators and then compute the new classifier as the average of all these estimates:

$$h(x) = \frac{1}{N}\sum_{k=1}^{N} h_k(x) \text{ with } h_k \text{ the } k\text{th tree classifier}$$

The response of each tree depends on a set of predictor's values chosen independently and with the same distribution for all trees, which is a subset of the predictor values of the original data set.

Bagging: Mean accuracy



- Optimal score : 0.980
- Parameter : $N = 160$

Remark : Good results but high variance/ no convergence phenomenon

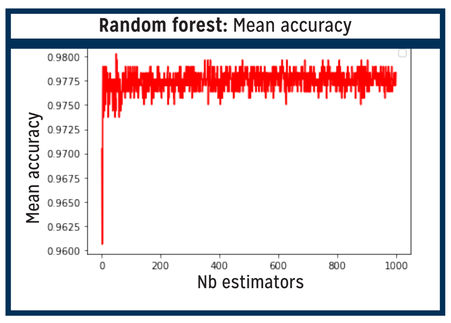
Method 2: Random Forest

Random forest changes the algorithm for the way that the sub-trees are learned so that the resulting predictions from all of the sub-trees have less correlation. The number of features that can be searched at each split point must be specified as a parameter to the algorithm : max features. In our case, we took the standard rule of thumb:

$$\text{max\_features} = \sqrt{number\_of\_input\_features}$$

For this classification problem, the *random forest method* defines a margin function that measures the extent to which the average number of votes for the correct class exceeds the average vote for any other class present in the dependent variable. This measure provides us not only with a convenient way of making predictions, but also with a way of associating a confidence measure with those predictions.
A leaf is the end node of a decision tree. A smaller leaf (i.e. a leaf containing a few sample) makes the model more prone to capturing noise in train data.

Random forest: Mean accuracy



- Optimal score : 0.981
- Parameters : $N = 32$ and $min\_sample\_leaf = 50$

Method 3: Boosting

The last approach employed was the *boosting method*, which involves that a set of "weak learner classifiers" are tuned to become a single "strong learner classifier". Factually, *boosting* could provide the final strong learner classifier, which is a merely weighted majority vote of weak classifiers. The formula of final classifier computed by the *boosting method* is the following:

$$G(x) = sign\left(\sum_{k=1}^{N} \alpha_k G_k(x)\right)$$

- $N = 160$: Number of estimators
- α_k : The learning rate
- $G_k(x)$: The base of Weak learner estimators

In the context of the problem, we studied the impacts of the number of estimators and the learning rate on the *Adaboost algorithm's score*.

Boosting: Holdout method's scores for a learning rate fixed to 0.9 (optimal value)

- Optimal score : 1.0
- Parameters : $N = 82$ and $\alpha_k = 0.9$

Conclusion

To sum it up, thanks to this investigation trough decision tree methods, we are pretty convinced that the implementation of a ***boosting method*** would be the best algorithm for our mobile application. Indeed, from the outcomes and scores provided, we notice that the *boosting* offers a reduced bias and variance, with a score of 1.0. Moreover, note that these results were obtained with the light dataset, where only the most obvious mushroom's features are treated.
However, we note that the *bagging* and the *random forest* offered also scores near to one. Although these facts, the *bagging* and *random forest* score are the lowest obtained, and, they provide potentially 2% of misclassifications. Furthermore, their variances are significantly large.
We also want to underline that by adding the other feature, the results were much better for each method and *boosting* stay the better one. Therefore, an implementation of boosting would provide a better security for any beginners or amateurs 2.0 shroomers.

1. **Boosting**
2. **Random forest**
3. **Bagging**

References

- Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning: data mining, inference and prediction. 2nd ed. Springer, 2009.
url: *http://www-stat.stanford.edu/~tibs/ElemStatLearn/*.

- Kevin P. Murphy. Machine Learning: A Probabilistic Perspective. The MIT Press, 2012.
isbn: 0262018020, 9780262018029.

- Mushroom Data Set, Mushroom records drawn from The Audubon Society Field Guide to North American Mushrooms (1981). G. H. Lincoff (Pres.), New York: Alfred A. Knopf.

https://archive.ics.uci.edu/ml/datasets/mushroom.

- North America Mycological Association (NAMA).
https://wwwnamyco.org/mushroom_poisoning_syndromes.php.

- scikit-learn : Machine Learning in Python.
http://scikit-learn.org/stable/index.html.

Acknowledgements

The authors acknowledge the support of David V. Anderson, the instructor of ECE 6254 and professor in the ECE department, and the members of the ECE 6254's educational.