Atharva Fulay - 1853414943
INF 551 Final Project
5/10/2020

Applying Decision Trees and Random Forests to Predict Poisonous Mushrooms

Abstract

Decision Trees and Random Forests are used to classify data based on its attributes. In this paper, I've outlined how I created decision tree and random forest models and compared their respective results when used to predict whether a mushroom is poisonous when eaten or if it is safely edible. The models use a training and test dataset to validate the results. Both, the decision tree model and the random forest model, are able to accurately classify 100% of poisonous mushrooms when there is no cap on the max_depth. When there is a cap, though, the random forest shows a slight advantage.

Introduction

In INF 552, we were taught about various machine learning methods including Decision Trees. Leveraging decision trees to make predictions or classifications is a popular method of machine learning today. However, decision trees are prone to becoming overfit to the training data. Pruning is a method that can be used to avoid overfitting, but could be coming at the cost of the overall prediction accuracy. Another solution that computer scientists found is using randomness to help mitigate this issue. Thus, one popular machine learning analysis that is built off the core of Decision Tree classifier (DTC) and leverages randomness is a Random Forest. Through this randomness, the Random Forest classifier (RFC) is less likely to suffer from overfitting while often resulting in a higher accuracy. The analysis in this paper demonstrates the use of both classifiers, and tests the advantages of random forests.

Data Exploration and Preparation

To apply random forests to a real world example, I analyzed a set of over 8,000 mushrooms [Resource 1]. There are 4,208 records for edible mushrooms and 3,916 for poisonous. The attributes of this dataset are `['edible-poisonous', 'cap-shape', 'cap-surface', 'cap-color', 'bruises', 'odor', 'gill-attachment', 'gill-spacing', 'gill-size', 'gill-color', 'stalk-shape', 'stalk-root', 'stalk-surface-above-ring', 'stock-shape-below-ring', 'stalk-color-above-ring', 'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number', 'ring-type', 'spore-print-color', 'population', 'habitat']` [Resource 2]. The objective is to determine which mushrooms are poisonous and which are safe to eat (`'edible-poisonous'`).

Initially, I explored the data and noticed there are numerous values for each attribute. Even though there are some records where the `'stalk root'` is unknown, I included that as a feature, and factorized it as unknown. Finally, I factorized the rest of the data points, since all of the fields are texts which map to the nature of their attribute.

Next, though any true application of this dataset is out of left field, it is important to determine the practical use of this data. In this case, one possible situation would be if

one were camping and were bet by a friend to eat a mushroom they encountered on a trail. Assuming knowledge of this model, one could determine if it is safe to eat or not. Because of this situation, and for general practice, I fed both models all fields and factorized values except `'edible-poisonous'` and had the models predict it.

In order to run these models efficiently, I used the sklearn library [Resource 3][Resource 4][Resource 5]. Sklearn offers comprehensive support and documentation for both DTC and RFC as well as other methods used in the process.

The data is shuffled to disassociate any possible patterns. In an effort to use as much of the dataset as possible, I didn't only do the typical training and test data sets. Rather, I include a K-Fold cross validation as well. This was done not necessarily to find the best parameters, but to see how well each model does on this dataset. I used a k of 10, which effectively splits up the data into 10 sections and trains it with 9 of the 10, tests on one, and then rotates through all 10. This way, we are able to make use of all the data points. Since this dataset was quite rich, there is no major concern about overfitting. The RFC and DTC are both run with this data to generate the models.

Experiments and Results

Both classifier models are easily able to classify 100% of the edible and poisonous mushrooms. Typically both model types needed 7-8 levels (depth) to achieve the perfect mark. To make analysis more interesting, I experimented with the parameters of both model types before they achieved the 100% score. Some of the experiments in detail below are: using entropy, max_depth, max_features, and class weights.

Model comparisons

The RFC models are generated with 100 decision trees all using entropy as their method of calculating which attributes to use at each node. DTC is also using entropy as the method of choosing the attribute. The difference between using entropy and gini were negligible.

Since the models were both returning 100% accuracies by depth 8, I began my experiments by including the max_depth parameter, which limits how deep the trees can go. A good depth for comparisons ranges between 3 and 5. This is where the models are still able to produce accuracies of over 90%, but are not 100%. By limiting the depth of the tree, I am generalizing it for a broader set of mushrooms. If this dataset was more complex, then there could be potential cases of overfitting, which is also where this depth-limitation would have helped the models from suffering from that. This parameter is also present in the following analyses to compare the two model types.

Next, as a part of experimenting with the data, I attempted to set the max_features as well to compare the RFC to DTC. There were two main considerations for max_features: None or 'sqrt'. Sqrt will take the square root of the number of features and use at most that many when constructing the RFC and DTC, while None would tell the classifiers to use all available features. For some of my analyses, I intentionally used 'sqrt' for comparison purposes only because None provided identical results.

The final aspect I experimented with was class weights. To think about this intuitively, it is much better to not eat an edible mushroom than to eat a mushroom thinking it is edible but it is really poisonous. Thus, instead of weighing the edible and poisonous with the same weight, it makes more sense to highly value

knowing for sure which mushrooms are poisonous. That way, one can be confident about eating the mushroom and not being poisoned. To account for this, I placed a high weight of 100 on the poisonous label and 1 on the edible label. This should ensure that any errors made by both model types would err on the side of caution. In other words, the models would rather predict that a mushroom is poisonous even if it isn't than predicting the mushroom is not poisonous even if it is.

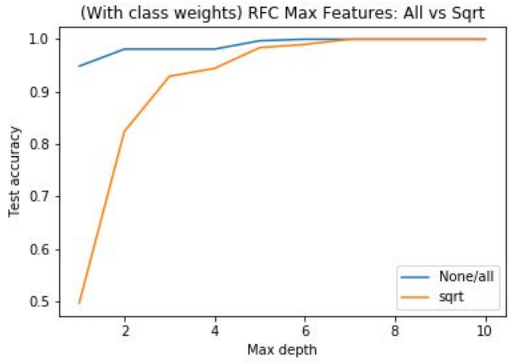
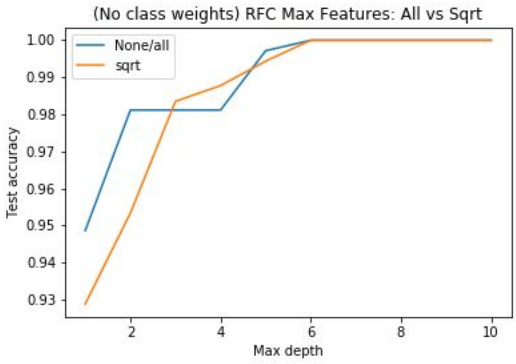


Figure 1. (above) Using a weight of 100 on poisonous, a comparison between using all features and only the square root number of features.

Figure 2. (below) With no special weight classes, all vs square root number of features for RFC. For the most part, using all features proves advantageous.



Figures 3 and 4 - Same setup as figures 1 and 2 except for DTC. Using all the features for DTC proves advantageous as well. Results are much more consistent than sqrt.

Figures 5 and 6 - Comparing RFC to DTC. Although the DTC catches up quickly, it is more inconsistent in its accuracy at lower max_depths than the RFC. Figure 6 (next page) is the same as Figure 5, but now with no class weights.

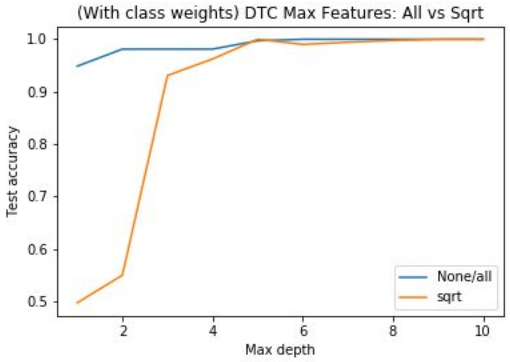


Figure 3 above.

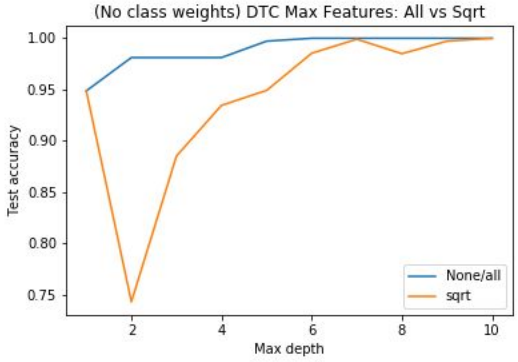


Figure 4 above.

Figure 5 above.

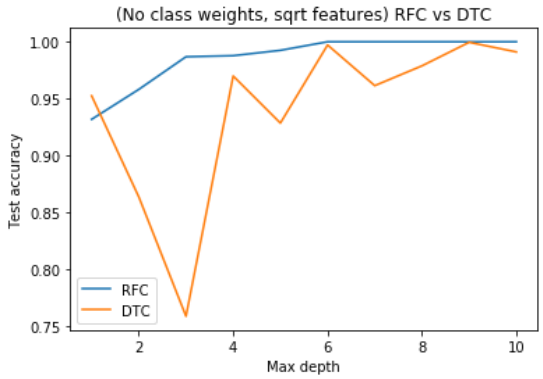


Figure 6 above.

By visualizing the analyses through Figures 1-6, one can see various differences between the models. Firstly, when viewing Figures 1 and 2 together and Figures 3 and 4 together (where the only difference is the usage of class weights), None/all resulted in higher test accuracies than 'sqrt' for the most part. This makes sense since the model can use all the available features instead of only a limited selection. Though it may be hard to tell, the blue lines in Figures 1 and 2 are almost the exact same values. Same for the blue lines in Figures 3 and 4.

This brings up the second observation: class weights seemed to give the 'sqrt' models a hard time for both model types. Using the weights was a way to ensure that the models err on the side of caution, but at lower depths that resulted in erratic model behavior. Since there were a limited number of features that could be used, it made both models more inconsistent when attempting to ensure no false positives.

Next, it comes down to comparing the DTC to RFC. One quick note, as I mentioned above, I used 'sqrt' for max_features only for comparison purposes. When using None, both models provided similar results (to the point where the difference was not visible). Looking at Figures 5 and 6, it is clear the RFC is more consistent than DTC. The lines are much smoother, and do not often bounce around in terms of test accuracy. This partially shows the advantage of random forests being more sturdy and steady than a single decision tree.

Furthermore, when comparing these models using the K-Fold cross validation, the RFC fairly consistently beat out the DTC in overall test accuracy (0.994 to 0.953). This is, of course assuming the max_depth is between 3 and 5. It seems that the RFC is simply more consistent than the DTC. Again, this makes intuitive sense since the RFC is more robust and is able to work better for generalization.

These analyses are not able to truly test the DTC's proneness to easily overfit with this dataset. Because the features provide rich discrete data, the DTC and RFC didn't need to go deep to produce accurate results. However, one can see that the RFC is more consistent in its results when paired with the suboptimal parameters. Altogether this displays the consistency and robustness that random forests provide even though it produces similar results to the decision tree models.

However, it must be acknowledged that the decision tree model performed well. With optimal parameters, the decision tree produced accuracies identical to those from random forests. DTC's shortcomings were visible when under not the best conditions (compared to RFC). But under the best conditions, it still is able to produce great results all while incurring significantly less computation.

RFC and DTC Specifics

Focusing only on RFC, the most important feature, by far, is odor. As you can see in Figure 7 (next page), odor takes a commanding 0.40 on average for importance level. Next best was ring type, resulting in about 0.07.

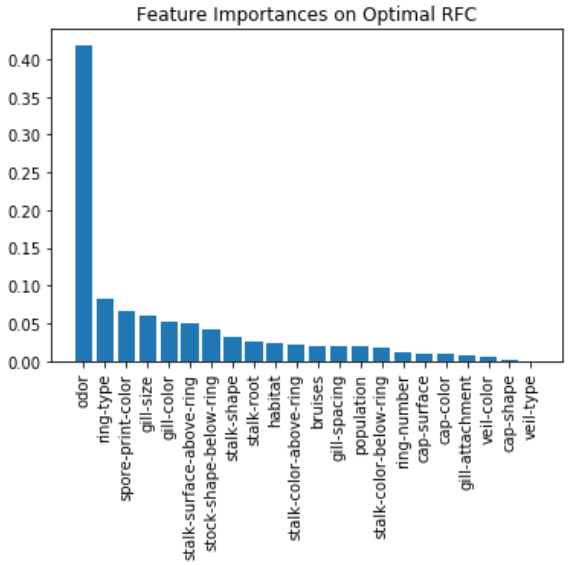
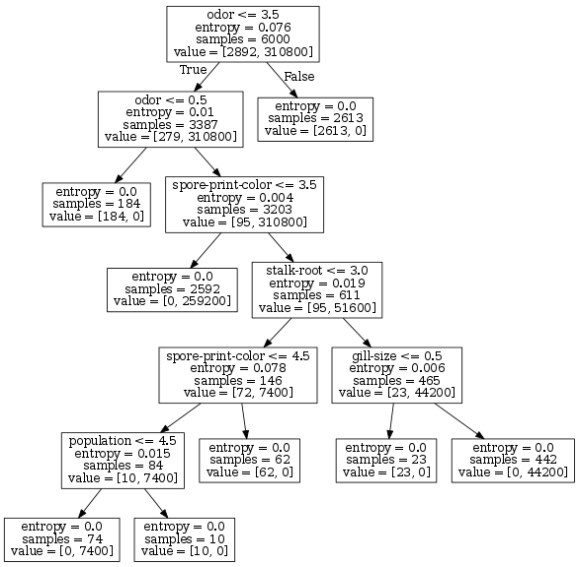


Figure 7 (above). This displays the feature importances for the optimal RFC. Odor is the most important feature, by far, having an average value of 0.40.

Figure 8 (below). The optimal DTC model looks something like this. Obviously it may vary if you run it on your own due to sampling of the dataset, however, it should look fairly similar to this. Note, this is with the high class weight on poisonous, which is why the tree is narrow.



Next, focusing only on the optimal DTC, one can see the simplicity of the tree in Figure 8. This model has the heavy weight on poisonous, which is why the tree is quite narrow compared to when the weights are evenly balanced.

Since the optimal models are producing 100% results, it is obvious what the confusion matrix would look like. However, when using suboptimal conditions, both models error on false negatives rather than false positives. Again, this is intentional since it would be better to not eat an edible mushroom than it would be to eat a poisonous one.

Both model types are able to accurately identify 100% of the poisonous mushrooms, but have a few that are false negatives.

Max-depth capped RFC:

Predicted	edible	poisonous
Actual		
edible	4040	168
poisonous	0	3916

Max-depth capped DTC:

Predicted	edible	poisonous
Actual		
edible	3845	363
poisonous	0	3916

Due to the fact that the data is shuffled and then partitioned, there is a small variance in the accuracies/optimal result. When running my analyses, I chose the best accuracy on a consistent basis. Therefore, if you choose to run the code yourself, there is a chance that the results

will not match exactly as these analyses display. Though, the optimal choices and visuals should have similar outputs.

Conclusion

All in all, both models are able to accurately predict 100% of the poisonous mushrooms with ease. In a real application, if a model was to tell a user that eating a particular mushroom could be deadly is pretty fascinating. Though unlikely, it could prove as insightful information when out in nature.

Even though the random forest model is thought to be advantageous over the decision tree classifiers, decision trees are still very useful and applicable to many datasets. In this case, the decision tree's simplicity produced accuracies more or less equivalent to the random forest model despite RFC's complexity and depth. Though, the random forest models did display more robustness when under suboptimal conditions than decision trees under the same parameters.

Citations

[Resource 1] Data was sourced from:
https://archive.ics.uci.edu/ml/datasets/Mushroom
Mushroom records drawn from The Audubon Society Field Guide to North American Mushrooms (1981). G. H. Lincoff (Pres.), New York: Alfred A. Knopf

Repository: Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[Resource 2] Full data mapping from the data source: `(classes: edible=e, poisonous=p)`

1. cap-shape	bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s
2. cap-surface	fibrous=f, grooves=g, scaly=y, smooth=s
3. cap-color	brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y
4. bruises?	bruises=t, no=f
5. odor	almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
6. gill-attachment	attached=a, descending=d, free=f, notched=n
7. gill-spacing	close=c, crowded=w, distant=d
8. gill-size	broad=b, narrow=n
9. gill-color	black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y
10. stalk-shape	enlarging=e, tapering=t
11. stalk-root	bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?
12. stalk-surface-above-ring	fibrous=f, scaly=y, silky=k, smooth=s
13. stalk-surface-below-ring	fibrous=f, scaly=y, silky=k, smooth=s
14. stalk-color-above-ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
15. stalk-color-below-ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
16. veil-type	partial=p, universal=u
17. veil-color	brown=n, orange=o, white=w, yellow=y
18. ring-number	none=n, one=o, two=t
19. ring-type	cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z

20. spore-print-color	black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y
21. population	abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y
22. habitat	grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d

[Resource 3]
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[Resource 4]
https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[Resource 5]
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html