STA 663: Graduate Project

Brian N. White

4/24/2020

Summary

For this project I asked whether or not one could classify a mushroom as poisonous, or not, using morphological and ecological information about the mushroom. To answer this question I fit a random forest to a mushroom data set obtained from kaggle.com. The parameter I chose to tune was the number of trees used in the bagging process. The model performed extremely well, with a ten-fold CV estimate of the test error, as measured by gain capture, of 1. Thus, the model was likely not overfit. Of note, the model perfectly classified the mushrooms in the full data set. This is not surprising given the 'tree-like' manner in which mycologists classify mushrooms.

The Data

The data I used for the project can found via the following link:

- https://www.kaggle.com/uciml/mushroom-classification

This data set consists of 8124 observations on 23 variables. Each observation is a one of 23 species of gilded mushroom from the Agaricus and Lepiota Family. The variable 'class' tells us whether or not a mushroom is poisonous or not. The remaining 22 variables correspond to morphological and ecological characteristics of mushrooms. Note that every variable in this data set is categorical. This information was simulated using the 1981 edition of The Audubon Society Field Guide to North American Mushrooms.

Research Question

- Can one predict whether or not a mushroom is poisonous given morphological and ecological data? (i.e. a classification problem)

The Model

I decided to use a random forest to answer my research question. This model was chosen for the following three reasons:

1. Mycologists use dichotomous keys to identify mushroom species. A dichotomous key is, in fact, a decision tree. Thus, a tree-based model is a natural choice for the mushroom data.
2. A random forest can be used for classification. Further, the potential predictors are categorical and tree-based methods can easily handle this.
3. We recently studied tree-based methods and so I decided to solidify the material further by implementing a tree-based model on a new data set.

Note, I decided not to use a vanilla decision tree as I assumed that the performance would be inferior to a bagged tree or a boosted tree. Beyond, that my choice to use a random forest as opposed to a boosted tree was arbitrary.

I chose 'class' as my response and the the remaining 22 morphological/ecological variables as my predictors. My model is a random forest so I set $mtry = \sqrt{p}$ where $p = 22$. The parameter 'trees' was tuned using a grid of values specified in the code-chunks below.

Fitting the Model

I began by loading the data and necessary packages. Note that I chose to use the tidymodels framework with the 'ranger' library to implement the random forest.

```
library("rsample")
library("ranger")
library("tidymodels")

mushroom=read.csv("data/mushrooms.csv")
```

Next, I determined the proportion of poisonous and non-poisonous mushrooms in the data set. There are about the same number of poisonous mushrooms in the data as there are non-poisonous.

```
summary(mushroom$class)
```

```
##    e    p
## 4208 3916
```

```
summary(mushroom$class)[1]/sum(summary(mushroom$class))
```

```
##         e
## 0.5179714
```

```
summary(mushroom$class)[2]/sum(summary(mushroom$class))
```

```
##         p
## 0.4820286
```

Next, I prepped the data for ten-fold cross-validation.

```
set.seed(7)

mush_cv=vfold_cv(mushroom, v=10)
```

I then specified the model, set some preprocessing criteria via a recipe and tuned the parameter 'trees'. For my recipe I first removed variables in the data that were highly sparse and unbalanced. These were variables that only realized one of their levels. In addition, I converted the nominal data in the data to numeric binary model terms. These steps were necessary to transform the data into a form that the 'rand_forest' command could use (Thanks Lucy!).

To measure the efficacy of the model I calculated the ten-fold CV estimate of the test error, measured via gain capture, for each of the tree values within my grid. The optimal number of trees was found to be 300.

Note that the 'gain_capture' is a ratio from 0 to 1. As per the documentation, it is a ratio in which the perfect scenario is the denominator and the model performance is the numerator.

```
set.seed(7)

#recipe to adjust constant variable
mush_recipe=recipe(class~., data=mushroom) %>%
```

```
  step_nzv(all_predictors()) %>%
  step_dummy(all_predictors())

#random forest specification for tuning
rf_tune=rand_forest(mode="classification",
                    mtry=floor(sqrt(22)),
                              trees=tune())  %>%
  set_engine("ranger",
             importance="impurity")

#grid of 'tree' values to be consdiered in the tuning process
grid=expand.grid(trees=c(10, 25, 50, 100, 200, 300))

#the random forest is tuned
mush_tune=tune_grid(rf_tune,
                     mush_recipe,
                     grid=grid,
                     resamples=mush_cv,
                     metrics=metric_set(gain_capture, accuracy))

#results of tuning
mush_tune %>%
  collect_metrics() %>%
  filter(.metric == "gain_capture") %>%
  arrange(desc(mean))
```

```
## # A tibble: 6 x 6
##   trees .metric      .estimator  mean     n  std_err
##   <dbl> <chr>        <chr>      <dbl> <int>    <dbl>
## 1    10 gain_capture binary         1    10 6.62e-17
## 2    25 gain_capture binary         1    10 6.62e-17
## 3    50 gain_capture binary         1    10 6.62e-17
## 4   100 gain_capture binary         1    10 6.62e-17
## 5   200 gain_capture binary         1    10 6.62e-17
## 6   300 gain_capture binary         1    10 6.62e-17
```

```
#the optimal number of trees, with respect to the gini index, is determined
best_tree <- mush_tune %>%
  select_best(metric = "gain_capture") %>%
  pull()
```

Model Evaluation

The tuned random forest has a ten-fold CV estimate of gain capture of 1. In other words, the model is classifying observations in a perfect manner. I was sceptical of this result at first; however, upon further reflection this was not too surprising given that the morphological and environmental features present in the data are what mycologists use in their dichotomous keys to identify mushroom species.

With these facts in mind I feel confident that the random forest is not overfit. As such, I fit my tuned random forest on the full data set and examined the corresponding confusion matrix, keeping in mind that the model will likely perform slightly worse on new data.

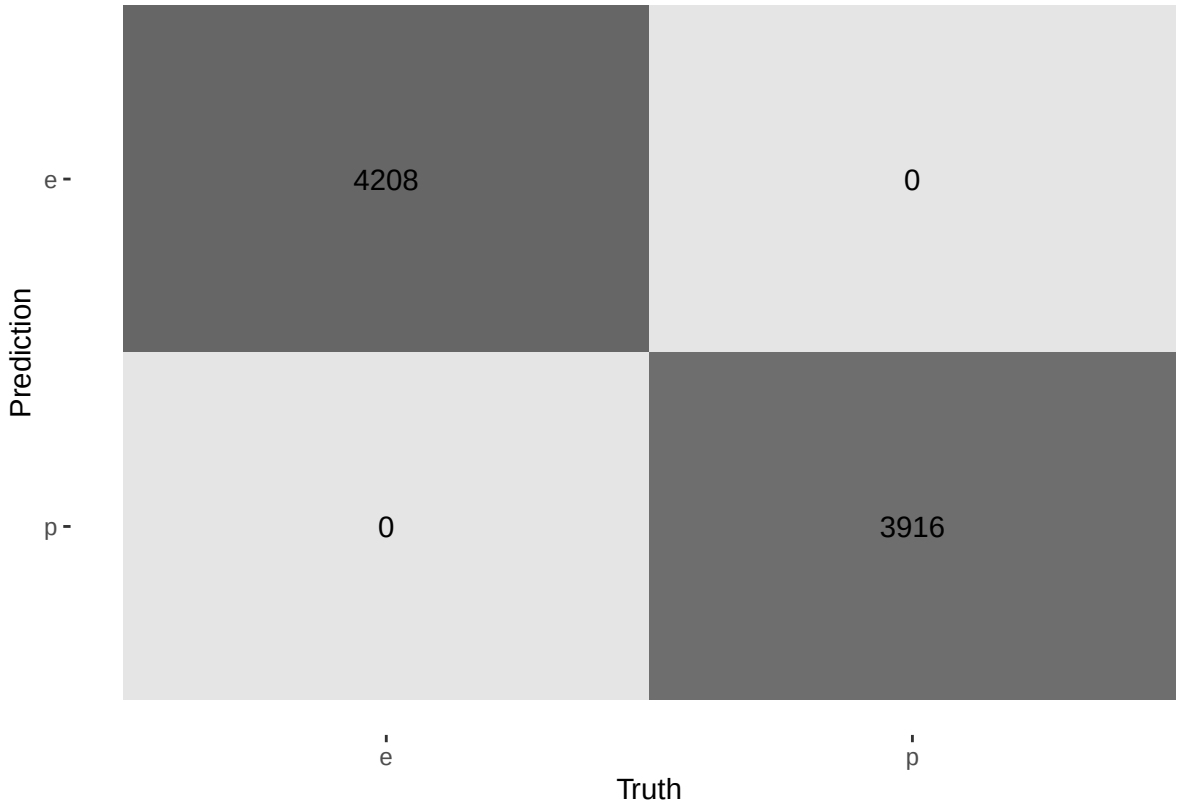
```
set.seed(7)
```

```
#create modified 'recipe' applied mushroom data set for use with fit function
mush_juice=prep(mush_recipe) %>%
  juice

#random forest specification with the tuned number of trees
rf_best=rand_forest(mode="classification",
                    mtry=floor(sqrt(22)),
                               trees=best_tree) %>%
  set_engine("ranger",
             importance="impurity")

#the model is fit to the data
mush_best=fit(rf_best,
              class~.,
                data=mush_juice)

#generates the confusion matrix for the tuned model on the data
mush_best %>%
  predict(new_data = mush_juice) %>%
  bind_cols(mush_juice) %>%
  conf_mat(truth = class, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```



I calculated the false positive and false negative rates of classification below. Note, that 'poisonous' is defined to be the 'positive' result in this setting. Observe that the model perfectly classified the mushrooms in the data set.

```
x=predict(mush_best, mush_juice)
```

```
mushroom1=mutate(mush_juice, predicted_class=x$.pred_class)

mushroom1 %>%
  summarise(fpr =
              sum(class == "e" & predicted_class == "p") /
              sum(class == "e"),
            fnr =
              sum(class == "p" & predicted_class == "e") /
              sum(class  == "p"))
```

```
## # A tibble: 1 x 2
##     fpr   fnr
##   <dbl> <dbl>
## 1     0     0
```

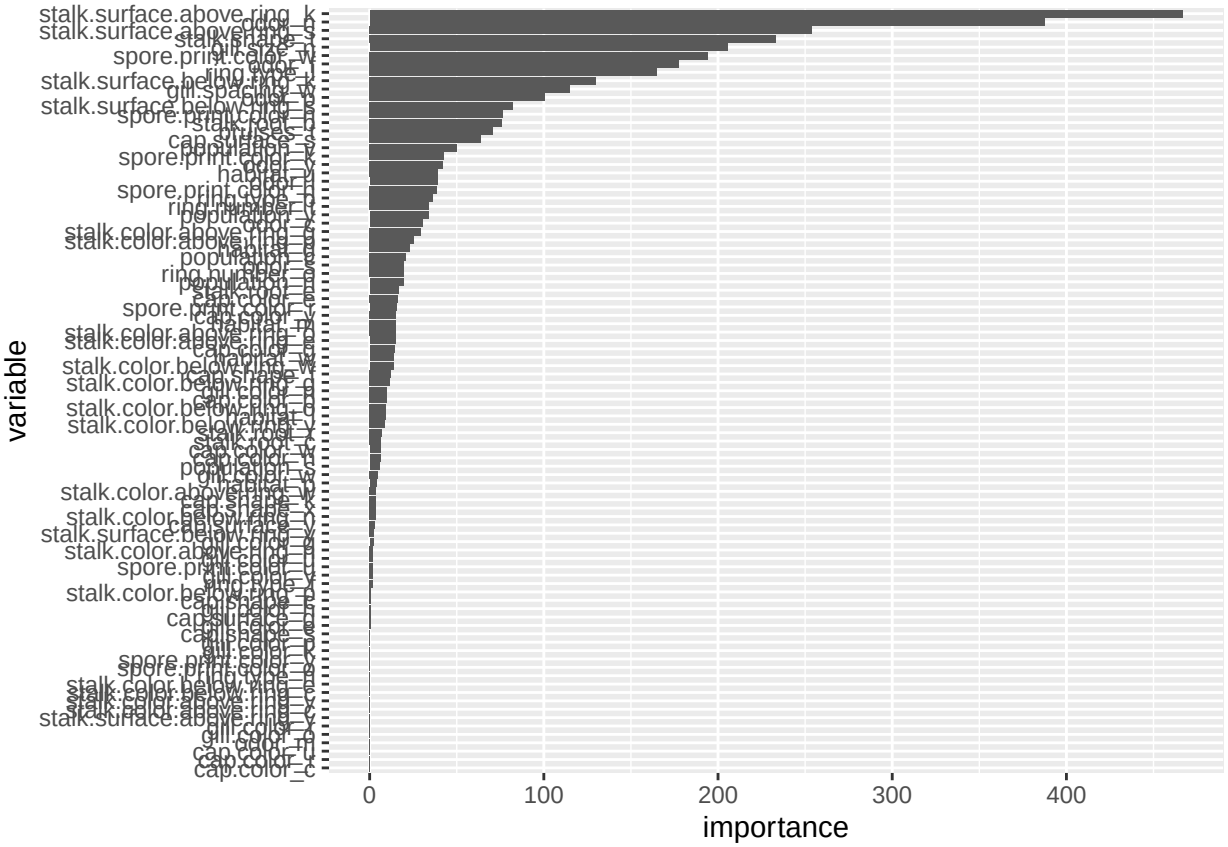
For fun (Variable Importance)

Although I was only asked to consider the predictive performance of the model, I decided, for completeness, to examine the relative importance of the predictors. In the plot below, 'importance' corresponds to the total amount that the Gini Index is decreased by splits of a given predictor, averaged over the 300 trees. Of the variables I considered, the most important, by far, appears to be 'stalk surfance above ring' followed by 'odor'. It would be interesting to consult a mycologist to see whether or not the features are some of the first variables considered in the dichotomous keys that are used for mushroom classification.

```
#variable importance
var_imp=ranger::importance(mush_best$fit)

#data frame created for use in plot
var_imp_df <- data.frame(
  variable = names(var_imp),
  importance = var_imp
  )

#modify data frame so that plot will present variables ordered by importance
var_imp_df %>%
  mutate(variable = factor(variable,
   levels = variable[order(var_imp_df$importance)])) %>%
ggplot(aes(x = variable, y = importance)) +
  geom_col() +
  coord_flip()
```



Future considerations

I would be curious to see how well a vanilla decision tree performs on this data set given the connection to dichotomous keys. If the predictive performance was preserved it woulld have an interpretive advantage over the random forest. Further, it would open up the opportunity to compare the decision tree with, and perhaps inform, those present in the dichotomous keys used by mycologists. Last, but not least, I would be interested in seeing how this model, fit on simulated data, performs on non-simulated data.