Statistical Learning: Assessed Practical 3

Kai Lawson-McDowall

14/05/2021

1) Fit Random Forest models using each possible input on its own to predict edibility. Evaluate the quality of fit by using the predict function to calculate the predicted class for each mushroom (edible or poisonous) (hint, you need type='response'). Which input fits best? (i.e. which classifies the most mushrooms correctly?)

There are five different categorical variables in the data frame "Mushrooms". For each variable, I created a random forest model using both the default number of trees (ntree = 500) and the number of variables randomly sampled as candidates at each split (mtry = sqrt(p), where p is the number of variables in a classification problem, in this case, p =1 and mtry = 1). I created training (n = 5686) and test datasets (n = 4047) from the original mushrooms (n = 8124) by randomly sampling with replacement (a key element in bagging). It should also be noted that the training and testing datasets are randomly sampled, but that the set.seed() command was used for reproducibility

The random forests were trained on the training data, and their predictive accuracy in determining the class for each mushroom determine via the predict(). The number of mushrooms classified correctly out of 4047 in the test set from least to most accurate was:

```
-Height: 2089 (51.6%)
-Cap Shape: 2287 (56.51%)
-Cap Surface: 2353 (58.14%)
-Cap Color: 2419 (59.77%)
-Odor: 3990 (98.59%)
```

From these initial tests, it is clear that Odor is the most accurate predictor of whether or not a mushroom is edible or poisonous. Height, conversely, is the least accurate predictor of Edibility and is only slightly better than random classification (which has a predictive accuracy of ~50%). Cap Shape, Cap Surface, and Cap color all show relatively similar levels of predictive accuracy.

2) Using cross-validation, perform a model selection to determine which features are useful for making predictions using a Random Forest. As above, use the number of mushrooms correctly classified as the criterion for deciding which model is best. You might try to find a way to loop over all 32 possible models (ignore the possibility of no input variables. Hint: you can use allCombs in the dagR package to generate all combinations of the numbers 1 to n). Or select features 'greedily', by picking one at a time to add to the model. Present your results in the most convincing way you can.

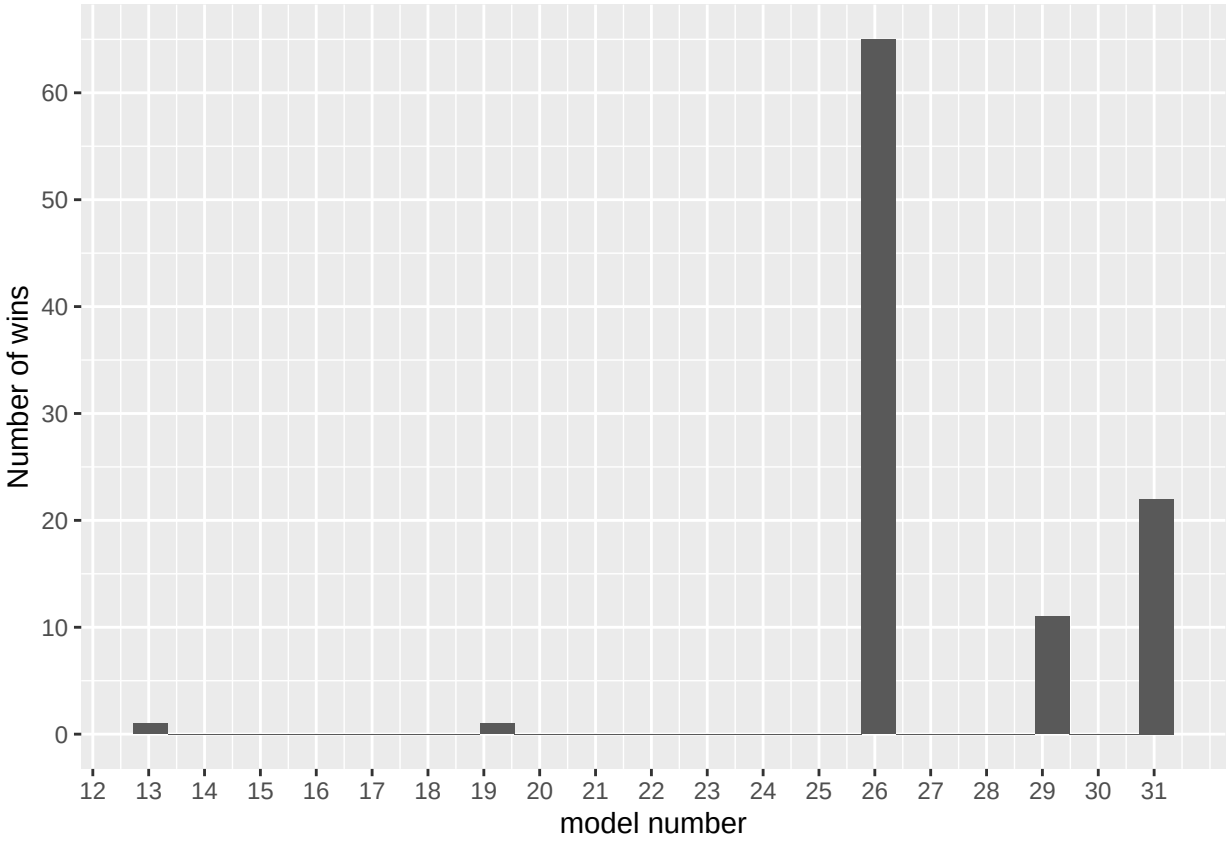
In order to perform cross-validation for the 31 potential random forest models (excluding a model with no input variables), I created a vector that contained all possible model formulas. I then ran this through a nested loop which performed cross-validation 100 times. Firstly, this nested loop randomly subset the original mushroom data into 60% training and 40% testing data (using a random uniform distribution to assign data to either set) for each loop, creating new datasets each time for all random forests to be trained and tested on. The default parameters for the number of trees (ntree = 500) and the number of variables randomly sampled as candidates at each split (sqrt(p) for classification, where p is the number of variables).

Following this, the predictive accuracy of each model was measured. The model with the highest number of correctly identified mushrooms in the testing set was then saved in the vector "winner" under its model number. Lastly, the mean predictive accuracy (proportion of correctly identified mushrooms from the test set) was taken for each model and placed in a data frame and denoted as "accuracy", where the rows, e.g. "V1" correspond to model 1, "V2" to model 2 and so forth.

The histogram results below indicate that model 26: Edible ~ CapShape + CapSurface + CapColor + Odor was the best model for predicting edibility, winning in 70% of cross-validation runs and correctly predicted over 99.15% of mushrooms in the testing set. This is also reflective of question 1, which shows that these four explanatory variables are the best predictors. Model 31, which contained all variables won 22 times, whereas Model 29: Edible ~ CapShape + CapColor + Odor + Height won 11 times. Model 19: Edible ~ CapShape + CapColor + Odor and model 13: Edible ~ CapColor + Odor, each won once (it should be noted the number of wins will likely change upon re-running of the cross-validation).

Interestingly, all of these models contained Odor, which, based on question 1, was previously shown to be the strongest predictor of edibility. Conversely, Height once again seemed to be a poor predictor. Based on the "results" data frame, all models contained four inputs correctly predicted with 98% accuracy, whereas the model containing Height was approximately 71% accurate. This seems to be a trend for all models, with the inclusion of Height as an explanatory variable reducing predictive power and Odor improving it.

```
##                                                          Model          Accuracy
## V1                                          Edible ~ CapShape 0.561533704390847
## V2                                       Edible ~ CapSurface 0.574314574314574
## V3                                          Edible ~ CapColor 0.582148010719439
## V4                                              Edible ~ Odor 0.985157699443414
## V5                                            Edible ~ Height 0.516182230467945
## V6                                Edible ~ CapShape + CapSurface 0.615543186971758
## V7                                  Edible ~ CapShape + CapColor 0.632034632034632
## V8                                      Edible ~ CapShape + Odor 0.985157699443414
## V9                                    Edible ~ CapShape + Height 0.560915275200989
## V10                               Edible ~ CapSurface + CapColor 0.671407957122243
## V11                                   Edible ~ CapSurface + Odor 0.985776128633271
## V12                                 Edible ~ CapSurface + Height 0.574314574314574
## V13                                     Edible ~ CapColor + Odor 0.988249845392703
## V14                                   Edible ~ CapColor + Height 0.582148010719439
## V15                                       Edible ~ Odor + Height 0.985157699443414
## V16                    Edible ~ CapShape + CapSurface + CapColor 0.692434549577407
## V17                        Edible ~ CapShape + CapSurface + Odor 0.985157699443414
## V18                      Edible ~ CapShape + CapSurface + Height 0.615543186971758
## V19                          Edible ~ CapShape + CapColor + Odor 0.988455988455988
## V20                        Edible ~ CapShape + CapColor + Height 0.624407338693053
## V21                            Edible ~ CapShape + Odor + Height 0.985157699443414
## V22                        Edible ~ CapSurface + CapColor + Odor 0.985157699443414
## V23                      Edible ~ CapSurface + CapColor + Height 0.653473510616368
## V24                          Edible ~ CapSurface + Odor + Height 0.985157699443414
## V25                            Edible ~ CapColor + Odor + Height 0.985157699443414
## V26          Edible ~ CapShape + CapSurface + CapColor + Odor 0.990311276025562
## V27        Edible ~ CapShape + CapSurface + CapColor + Height 0.702741702741703
## V28            Edible ~ CapShape + CapSurface + Odor + Height 0.987425273139559
## V29              Edible ~ CapShape + CapColor + Odor + Height 0.989692846835704
## V30            Edible ~ CapSurface + CapColor + Odor + Height 0.987425273139559
## V31 Edible ~ CapShape + CapSurface + CapColor + Odor + Height 0.990105132962276
```



3) Would you use this classifier if you were foraging for mushrooms? Discuss with reference to factors that you identified as important and the probability of poisoning yourself.

The best random forest classifier, Edible ~ CapShape + CapSurface + CapColor + Odor, correctly identified 99.15% of all mushrooms using the four most important individual factors in determining a mushrooms edibility status. In layman's terms, the chance of this classifier misidentifying a mushroom is less than 1/100.

Although this could be considered a robust predictor of edibility,a key real-life consideration deciding whether or not to use this classifier is the degree to which particular mushrooms are poisonous. For instance, although one mushroom might be classed as "poisonous," it may only cause unpleasant symptoms, whereas another mushroom might be lethal if eaten. Similarly, although a mushroom might be edible, it may not taste particularly good and should therefore be avoided. The lack of information regarding the severity of the poisonous status would make me hesitant to use it. Regarding producing random forests, greater sub-classifications of how edible and how poisonous these mushrooms are would be beneficial.

Furthermore, a key consideration when using this classifier would be the number of mushrooms being picked. Essentially, the more mushrooms picked, the greater the probability of a false positive (poisonous as edible) increases. As picking and eating only one poisonous mushroom could be dangerous, I would be skeptical using it for gathering a large number of mushrooms for the increasing probability of a false positive.

In terms of how the models could be altered to improve prediction accuracy, we can also consider random forest models' parameters, such as the of trees constituting the random forests. Although the default of 500 trees was used, in general, the predictive accuracy of random forests increases with the number of trees, and a greater number of trees does not cause the random forest to overfit (Hastie et al., 2016). Eventually, however, a point will be reached where the relative increase in computational time for learning these additional trees is not worth the increased predictive accuracy. Although more trees are needed in generating random forests

for larger datasets, some work indicates that choosing the largest number of trees which is computationally viable, this number has been suggested to be as low as 64-128 trees (Oshiro et al., 2012), indicating that our default of 500 is sufficient to produce accurate random forest classifiers.

Mtry, which determines the number of variables randomly sampled as candidates at each split, or the 'depth' of the trees, appears to be a more complex hyperparameter to tune. For instance, although the number of trees does not cause overfitting, more depth to the model can result in too rich a model, which incurs unnecessary variance (Hastie et al., 2016). However, the running of the cross-validation allows overfitting to be avoided, which is not necessarily true for out-of-bag (OOB) error alone (The average error for each calculated using predictions from the trees that do not contain in their respective bootstrap sample). In terms of tuning mtry, the tuneRF() command in the randomForest package could be used on each model by selecting the mtry, resulting in the lowest OOB error estimate.

Work Cited:

Hastie, T., Friedman, J. and Tisbshirani, R., 2017. The Elements of statistical learning. New York: Springer.

Oshiro, T., Perez, P. and Baranauskas, J., 2012. How Many Trees in a Random Forest?. Machine Learning and Data Mining in Pattern Recognition, pp.154-168.

R Code:

1)

```
mushrooms  <- read.csv(
"C:\\Users\\Kai\\Desktop\\Statistical learning\\Assessed Practical 3\\mushrooms.csv")

#had to indent the code so it could fit on the page

install.packages("randomForest")
library(randomForest)
as.data.frame(mushrooms)
mushrooms[sapply(mushrooms, is.character)] <- lapply(
  mushrooms[sapply(mushrooms, is.character)], as.factor)
# converts all variables in the mushroom dataframe to factors.


#training and test datasets
set.seed(5833)
training <- sample(nrow(mushrooms), 0.7*nrow(mushrooms), replace = TRUE)
#70% training 30% testing.

trainingset <- mushrooms[training,]
#creating the training set

testset <- mushrooms[-training,]
# creating the testing set

as.data.frame(testset)
#4047 rows

#Cap Shape:
ShapeRF <- randomForest(Edible ~ CapShape, data = trainingset)
ShapePred = predict(ShapeRF, newdata = testset, type = "response")
```

```r
ShapePred <- as.data.frame(ShapePred)
sum(ShapePred$ShapePred == testset$Edible)

# Cap Surface:
SurfaceRF <- randomForest(Edible ~ CapSurface, data = trainingset)
SurfacePred = predict(SurfaceRF, newdata = testset, type = "response")
SurfacePred <- as.data.frame(SurfacePred)
sum(SurfacePred$SurfacePred == testset$Edible)

# Cap Color :
ColorRF <- randomForest(Edible ~ CapColor, data = trainingset)
ColorPred = predict(ColorRF, newdata = testset, type = "response")
ColorPred <- as.data.frame(ColorPred)
sum(ColorPred$ColorPred == testset$Edible)

# Odor:
OdorRF <- randomForest(Edible ~ Odor, data = trainingset)
OdorPred = (predict(OdorRF, newdata =testset, type = "response"))
OdorPred <- as.data.frame(OdorPred)
sum(OdorPred$OdorPred == testset$Edible)

# Height
HeightRF <- randomForest(Edible ~ Height, data = trainingset)
HeightPred = predict(HeightRF, newdata =testset, type = "response")
HeightPred <- as.data.frame(HeightPred) # the number predicted as edible.
sum(HeightPred$HeightPred == testset$Edible)
```

2)

```r
#creating all possible models:
#using dagR

library(dagR)
library(tidyr)
library(randomForest)
library(ggplot2)


mushrooms[sapply(mushrooms, is.character)] <- lapply(
  mushrooms[sapply(mushrooms, is.character)], as.factor)
# converts all to factors


#Creating the models:

mushroom.names <- as.vector(colnames(mushrooms[2:6]))
# creates a vector of the explanatory variables

combinations <- allCombs(x = mushroom.names, force = 5)
#creates a matrix with possible combinations

combinations <- combinations[-1,-1]
#removes the "empty" row containing no variables
```

```r
combinations.list <- as.list(as.data.frame(t(combinations)))
#creates a vector of vectors

#for loop to create to format the models correctly:

for (i in 1:length(combinations.list)){
  combinations.list[[i]] <- combinations.list[[i]][!is.na(combinations.list[[i]])]
  # removes the NA's and concentates

  combinations.list[[i]] <- paste(combinations.list[[i]], collapse = " + ")
  # creates +'s between explanatory variables

  combinations.list[[i]] <-c("Edible ~",combinations.list[[i]])
  #adds the output variable, edible.

  combinations.list[[i]] <- paste(combinations.list[[i]], collapse = " ")
  # combines them into one formula
}
formulas <- c(combinations.list, recursive = TRUE)
# turns it into a single vector, "V1" corresponds to model 1, i.e. Edible~CapShape


#Nested Loop

formulas <- c(combinations.list, recursive = TRUE)
current_model <- vector('list', length(formulas))
#creates an empty list to store the random forest results.
winner = rep(NA, 10)

#1st loop:
winner = rep(NA, 100)
for (iteration in 1:100){

trainRows<-runif(nrow(mushrooms))>0.60
#randomly put aside 60% of the data

training<-mushrooms[trainRows,]
# about 60%

testing <-mushrooms[!trainRows,]
# about 40% testing

prediction_accuracy = rep(NA, length(formulas))
#Replicate elements of vectors and lists

#2nd loop
for(i in seq_along(formulas)) {
current_model <-  randomForest(formula = as.formula(formulas[i]), data = training)
predicted = predict(current_model, testing, type = "response") #
prediction_accuracy[i] <- (sum(
  (predicted== testing$Edible)/nrow(testing), na.rm = TRUE))
}
```

```
winner[iteration] = which.max(prediction_accuracy)
}

#dataframe containing the models and their mean
#prediction accuracy over 100 cross-validation runs.

results <- cbind(formulas,as.numeric(prediction_accuracy))
results <- as.data.frame(results)
results

#Histogram:

winner <- as.data.frame(winner)
ggplot(winner, aes(x = winner)) + geom_histogram() +
  scale_x_continuous(name ="model number", breaks = seq(1, 31, by=1)) +
  scale_y_continuous(name ="Number of wins", breaks = seq(0, 100, by=10))
```