Mushroom Edibility Prediction

Macmillan Jacobson

2023-03-27

Abstract

This paper attempts to use observations about the stems of mushrooms to predict if the mushroom is edible or poisonous. We use 6 categorical variables that can be easily observed upon simply looking at a given mushroom. The motivation of this report is two fold. The first is to create a useful tool to help foragers identify which mushrooms they should take home to cook and which they should leave out in the forest. The second motivation is to examine the efficacy of sophisticated learning algorithms like random forest and AdaBoost in contrast with more a more basic and well understood algorithm like logistic regression. In our analysis we find that random forest and AdaBoost both achieve a 97.65% accuracy score on the testing dataset while the logistic regression algorithm achieves 85.48% accuracy. Clearly, the more sophisticated methods outperform the more basic method. Furthermore, we have created a useful model to help (with some error), predict the edibility of mushrooms found in the wild using simple observations.

Introduction

In this report, we will attempting to create a model that can inform a user if a given mushroom is edible. To accomplish this task we will use three classification methods. Two of these classification methods will be more advanced and one more basic in order to illustrate the efficacy of the more advanced methods. As we constantly encounter mushrooms in our world, and as some mushrooms are perfectly safe and even delicious to eat while some will kill the consumer in minutes, it would be quite handy to have a way to make some simple observations about a mushroom's stalk to determine if the given mushroom is edible. We will do just that in this paper.

Problem Statement and Data Sources

We will attempt, via three classification methods, to generate a model that can accurately predict, based on information about a mushroom's stalk, whether it is edible or not. The 6 categorical variables we will use to predict edibility are stalk shape, the root of the stalk, the colors of the mushrooms in various places and the surface texture in various places. The data here comes from Kaggle and can be found here: https://www.kaggle.com/datasets/uciml/mushroom-classification?resource=download.

Proposed Methodology

The first method we will use to classify mushrooms as poisonous or edible is a random forest classifier. Random forests are known to be very good, rivaling even neural networks for classification purposes. As such, it makes sense that we would use a random forest here. The second method we will use is boosting. Boosting is also known to be effective in classification algorithms and so we will employ it here to see how it compares to our random forest. Finally, we will use a more basic method, logistic regression, to classify our mushrooms. We will do this partly as a sanity check and partly to see if the more sophisticated methods are, in fact, superior.

Analysis and Results

The dataset we are working with contains 8124 observations. Each of these observations contains 6 categorical variables and in the chart below, we can see the number of categories per variable. For the meanings of each of the values in each category, please refer to the definitions in the kaggle link above.

```
##              stalk.shape               stalk.root stalk.surface.above.ring
##                        2                        5                        4
## stalk.surface.below.ring   stalk.color.above.ring   stalk.color.below.ring
##                        4                        9                        9
```

Note that the chart above leaves out or response variable, "class". "Class" is simply 'e' or 'p' indicating edible or poisonous. Because all of our variables are categorical, we will treat each column as a factor. This, essentially turns our 6 predicting variables into 33 boolean predicting variables.

NOTE: In all of our classification methods, we will use an 20/80 test/train split for our dataset.

Random Forest

Below is the output from our random forest classifier tuned with a 5-fold cross validation scheme. We use the 'caret' package in R to create this model.

```
## Loading required package: ggplot2

## Loading required package: lattice

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 14
##
##         OOB estimate of  error rate: 2.58%
## Confusion matrix:
##      e    p class.error
## e 3266  107  0.03172250
## p   59 3004  0.01926216

## [1] "stalk.surface.above.ringk" "stalk.surface.below.ringk"
## [3] "stalk.shapet"              "stalk.rootb"
## [5] "stalk.roote"
```

We can see that our random forest hyperparameters end up being 500 trees with a minimum of 14 variables per tree. Note that since we have factored each of our predicting variables, this is 14 of 33 possible. Our resulting random forest has an error rate of 2.39%. 1.64% for our poisonous mushrooms and 3.11% for our edible mushrooms. Additionally, we can see the 5 most important factors in our predictions.

```
## Accuracy on testing data: 97.99%
```

Finally, our random forest achieves a 97.65% accuracy score on our testing set.

Boosting

We will build an AdaBoosting model using 5-fold cross validation for tuning our parameters. We use the 'adabag' library in r to build our model. Below are the results of our model after training and predicting on our testing dataset. As we can see in the results, AdaBoosting achieves an accuracy of 97.65% which is identical to that of random forest.

```
## Loading required package: rpart
```

```
## Loading required package: foreach

## Loading required package: doParallel

## Loading required package: iterators

## Loading required package: parallel

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   e   p
##          e 814  21
##          p  13 840
##
##                Accuracy : 0.9799
##                  95% CI : (0.972, 0.986)
##     No Information Rate : 0.5101
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9597
##
##  Mcnemar's Test P-Value : 0.2299
##
##             Sensitivity : 0.9843
##             Specificity : 0.9756
##          Pos Pred Value : 0.9749
##          Neg Pred Value : 0.9848
##              Prevalence : 0.4899
##          Detection Rate : 0.4822
##    Detection Prevalence : 0.4947
##       Balanced Accuracy : 0.9799
##
##        'Positive' Class : e
##
```

Logistic Regression

The final method we will use is logistic regression.

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##
## Call:
## glm(formula = class ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3817  -0.5642   0.0000   0.5162   1.6651
##
## Coefficients: (2 not defined because of singularities)
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 34.23207  743.35858   0.046 0.963270
## stalk.shapet                -0.32682    0.10092  -3.239 0.001201 **
## stalk.rootb                 -1.16279    0.10048 -11.572  < 2e-16 ***
## stalk.rootc                -20.41806  533.71271  -0.038 0.969483
## stalk.roote                 -1.82152    0.12880 -14.142  < 2e-16 ***
```

```
## stalk.rootr                -38.40576 3220.87196  -0.012 0.990486
## stalk.surface.above.ringk    0.68961    0.19420   3.551 0.000384 ***
## stalk.surface.above.rings   -0.14709    0.14565  -1.010 0.312562
## stalk.surface.above.ringy  -54.72303 4267.87672  -0.013 0.989770
## stalk.surface.below.ringk    1.54800    0.18374   8.425  < 2e-16 ***
## stalk.surface.below.rings    0.46099    0.13615   3.386 0.000710 ***
## stalk.surface.below.ringy   18.44869 3104.41615   0.006 0.995258
## stalk.color.above.ringc    -13.38624 3785.47099  -0.004 0.997179
## stalk.color.above.ringe    -36.65256 1151.86789  -0.032 0.974616
## stalk.color.above.ringg    -35.78800  710.01018  -0.050 0.959800
## stalk.color.above.ringn      0.01688  781.49130   0.000 0.999983
## stalk.color.above.ringo    -54.11204 1123.42501  -0.048 0.961583
## stalk.color.above.ringp    -17.09028  542.20368  -0.032 0.974855
## stalk.color.above.ringw    -17.42640  542.20369  -0.032 0.974360
## stalk.color.above.ringy     58.40429 6491.97706   0.009 0.992822
## stalk.color.below.ringc           NA         NA      NA       NA
## stalk.color.below.ringe    -35.60203 1123.38477  -0.032 0.974718
## stalk.color.below.ringg    -34.78620  678.78809  -0.051 0.959128
## stalk.color.below.ringn    -16.37790  508.52442  -0.032 0.974307
## stalk.color.below.ringo           NA         NA      NA       NA
## stalk.color.below.ringp    -16.16549  508.52440  -0.032 0.974640
## stalk.color.below.ringw    -16.26758  508.52440  -0.032 0.974480
## stalk.color.below.ringy    -16.37790 4334.98071  -0.004 0.996986
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8907.3  on 6435  degrees of freedom
## Residual deviance: 4103.1  on 6410  degrees of freedom
## AIC: 4155.1
##
## Number of Fisher Scoring iterations: 18
```

As we can see, our logistic regression calculation identified several important variables as significant and they are similar to those identified by random forest. Now lets see how our model predicts on the testing set.

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
##
## Accuracy on testing data: 83.18%
```

Logistic regression achieves an 85.48% prediction accuracy on our testing set. While this is not horrible, it is clearly severely outperformed by random forest and boosting.

Conclusion

We have built a model that can, with 97.65% accuracy, determine whether or not a given mushroom is edible. While this is obviously not perfect, it is quite good. Interestingly, we found that AdaBoosting and Random Forest algorithms achieve the same accuracy. Further analysis should be conducted to determine why that is the case in this dataset. Furthermore, we can see that the more sophisticated algorithms greatly outperform the more basic models like logistic regression.

Lessons Learned

I learned that AdaBoosting is a useful and viable algorithm for classification. The status quo, in my experience, has been to use random forests or neural networks. It is interesting to see a third algorithm compete with the two gold standard algorithms.

Appendix: All code for this report

```
df = read.csv('mushrooms.csv')[,c(1, 11:16)]
apply(df[,2:7], 2, function(x) length(unique(x)))
set.seed(1)
df[sapply(df, is.character)] <- lapply(
  df[sapply(df, is.character)], as.factor)

sample <- sample(c(TRUE, FALSE), nrow(df), replace=TRUE, prob=c(0.8,0.2))
train  <- df[sample, ]
test   <- df[!sample, ]
library(caret)
library(dplyr)
library(ggplot2)
set.seed(1)
rf <- train(
  class~.,
  data=train,
  method='rf',
  trControl=trainControl(method='cv', number=5),
  metric='Accuracy')
rf$finalModel
var_imp <- varImp(rf, scale=TRUE)$importance
var_imp <- data.frame(variables=row.names(var_imp), importance=var_imp$Overall)
var_imp[order(-var_imp$importance),][1:5,1]
y_hats <- predict(object=rf, newdata=test[,-1])
accuracy <- mean(y_hats == test[,1])*100
cat('Accuracy on testing data: ', round(accuracy, 2), '%',  sep='')
library('adabag')
model_adaboost <- boosting(class~., data=train, boos=TRUE, mfinal=50)

pred_test = as.data.frame(predict(model_adaboost, test)$votes)
colnames(pred_test) = c('e', 'p')
labels = colnames(pred_test)[apply(pred_test,1,which.max)]
cm = confusionMatrix(test$class, as.factor(labels))
cm
set.seed(1)
lr = glm(class~., data=train, family='binomial')
summary(lr)
y_hats <- predict(lr,newdata=test[,-1],type='response')
y_hats <- ifelse(y_hats > 0.5,'p','e')
accuracy <- mean(y_hats == test[,1])*100
cat('\nAccuracy on testing data: ', round(accuracy, 2), '%',  sep='')
```