Module 2 Individual Assignment: Technique Practice

William Scott Breckwoldt

College of Professional Studies: Analytics, Northeastern University

ALY 6040:

Dr. Marcus Ellis

June 9th, 2022

Module 2 Individual Assignment: Technique Practice

Code Walkthrough

#Installing libraries

install.packages('rpart') – powerful machine learning library in R used for building classification and regression trees. It implements recursive partitioning.

install.packages('caret') – Classification And REgression Training package that contains functions for the model training process of complex regression and classification problems

install.packages('rpart.plot') – used for plotting rpart trees

install.packages('rattle') - R Analytical Tool To Learn Easily provides data mining functions

install.packages('readxl') – used for loading Microsoft Excel files

#Loading libraries

```
library(rpart,quietly = TRUE)
library(caret,quietly = TRUE)
library(rpart.plot,quietly = TRUE)
library(rattle)
library(readxl)
```

Here we are installing and loading the packages and data necessary for completing the assignment. Some of the packages are loaded with 'quietly = TRUE'. This disables messages when loading packages.

#Setting the Working Directory

```
setwd("C:/Users/Scott/Desktop/ALY6040")
```

I set the working directory to the 'ALY6040' folder in my desktop.

#Reading the data set as a dataframe

```
mushrooms <- read_excel("mushrooms.xlsx")
```

I read the provided excel file from the working directory.

structure of the data

```
str(mushrooms)
```

The output of this command tells us that our dataset is a [8,124 x 23] tibble with all 23 columns being the character data type.

```
# number of rows with missing values

nrow(mushrooms) - sum(complete.cases(mushrooms))
```

By subtracting the number of rows from the rows with complete cases, or in other words, no missing values, we get a result of 0. Therefore, there are no rows with missing data.

```
# deleting redundant variable `veil.type` (should be 'veil-type')

mushrooms$'veil-type' <- NULL
```

This command removes the 'veil-type' variable.

```
#analyzing the odor variable

table(mushrooms$class,mushrooms$odor)
```

This command tells us the frequency of each unique odor variable per unique class variable in a table format. Class e mushrooms have only odors of a,l,n

```
number.perfect.splits <- apply(X=mushrooms[-1], MARGIN = 2, FUN = function(col){
  t <- table(mushrooms$class,col)
  sum(t == 0)
})
```

This builds a table of perfect splits for the decision nodes in our decision tree.

```
# Descending order of perfect splits

order <- order(number.perfect.splits,decreasing = TRUE)

number.perfect.splits <- number.perfect.splits[order]
```

This sets the number.perfect.splits in descending order.

```
# Plot graph
par(mar=c(10,2,2,2))
barplot(number.perfect.splits,
    main="Number of perfect splits vs feature",
    xlab="",ylab="Feature",las=2,col="wheat")
```

This returns a bar plot with the number of perfect splits per variable.

```
#data splicing
set.seed(12345)
train <- sample(1:nrow(mushrooms),size = ceiling(0.80*nrow(mushrooms)),replace = FALSE)
# training set
mushrooms_train <- mushrooms[train,]
# test set
mushrooms_test <- mushrooms[-train,]
```

The above commands give us a training and test set, which will be used for logistic regression. The size of the training set is 80% of the mushrooms dataset, and the test set is 20% of the dataset.

```
# penalty matrix
penalty.matrix <- matrix(c(0,1,10,0), byrow=TRUE, nrow=2)
```

This return a penalty matrix with values in rows one being 0 and 1 and values in row two being 10 and 0. The penalty matrix is used for classification splitting. This matrix assigns a penalty that is 10 times greater if a poisonous mushroom is classified as edible.

```
# building the classification tree with rpart
tree <- rpart(class~.,
        data=mushrooms_train,
        parms = list(loss = penalty.matrix),
        method = "class")
```

Class is our target variable. The ~ means that we are using all other variables as predictors. We set the data equal to our training data and the parameter equal to a list containing the penalty matrix.

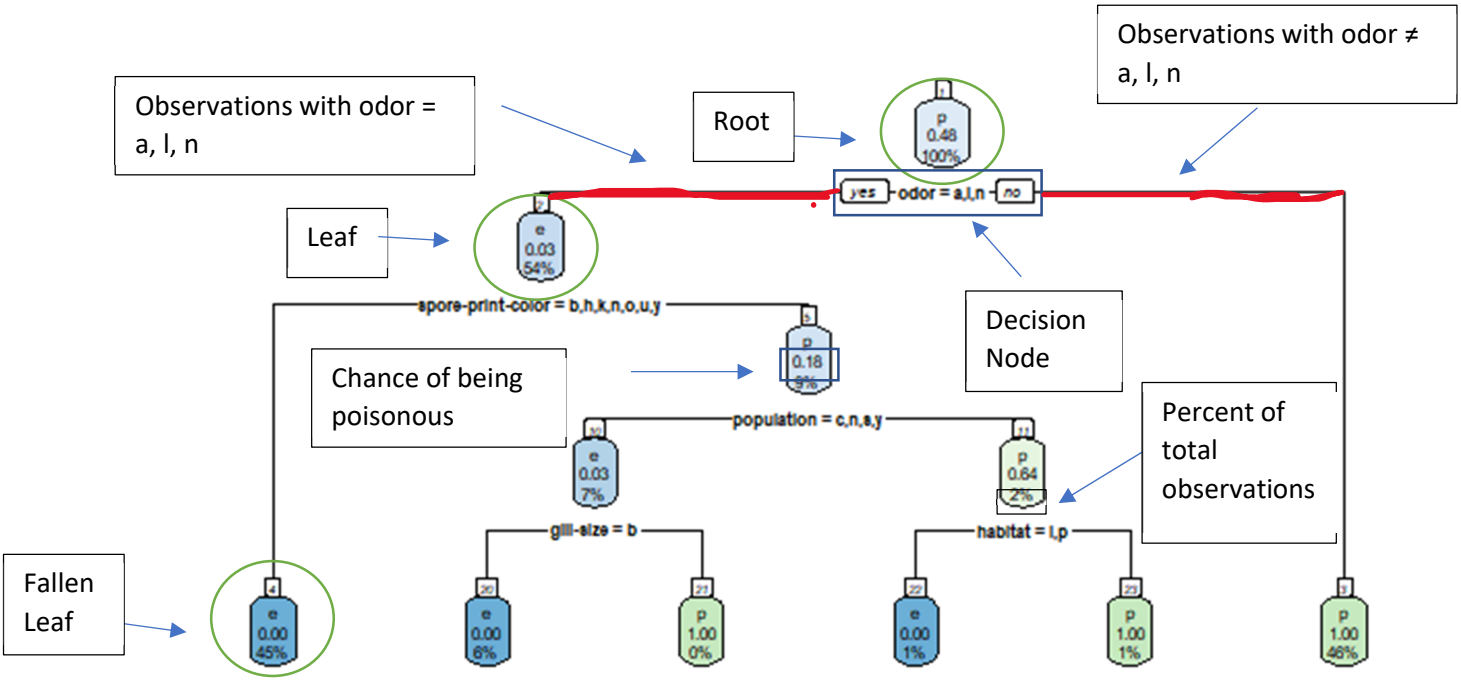
Visualize the decision tree with rpart.plot

rpart.plot(tree, nn=TRUE)

This command gives us the following visual for our decision tree. I have labeled the tree accordingly.

Figure 1

Decision Tree



choosing the best complexity parameter "cp" to prune the tree

cp.optim <- tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"]

This function returns the optimal cp value associated with the minimum error.

tree prunning using the best complexity parameter. For more in

```
tree <- prune(tree, cp=cp.optim)
```

This function creates our optimal decision tree.

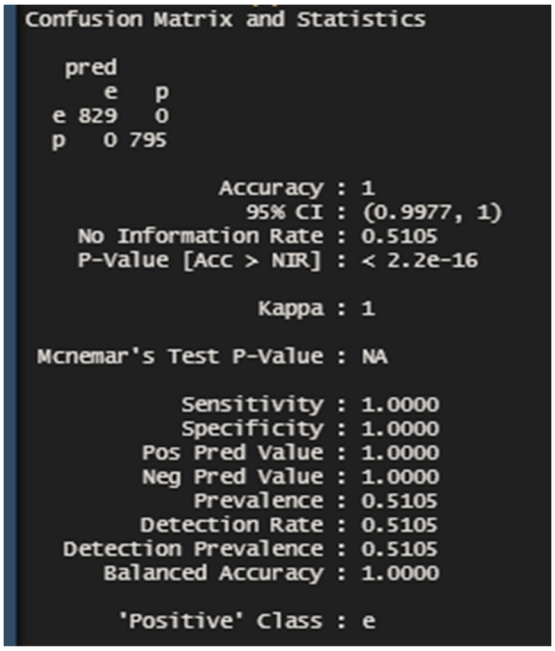
```
#Testing the model
pred <- predict(object=tree,mushrooms_test[-1],type="class")
```

Now that we have trained the regression tree model, we would like to make some predictions. We set our object equal to the inverse of our test dataset, which will be what we are predicting for.

```
#Calculating accuracy
t <- table(mushrooms_test$class,pred)
confusionMatrix(t)
```

Figure 2

Confusion Matrix and Statistics



According to the results, there are 829 true negatives and 795 true positives with 0 false negatives and positives. This tells us that our model is 100% accurate.

Sensitivity tells us the true positive rate or the portion of the positive class or e that was correctly predicted.

Specificity tells us the true negative rate or the portion of the class p that was correctly predicted.

Positive predictive value, also referred to as precision, tells us the number of the positive class or e correctly predicted as a portion of the total positive class predictions made.

Negative predictive value tells us the number of the negative class or p that was correctly predicted as a portion of the total negative class predictions made.

Prevalence tells us how often the positive class occurred in the sample.

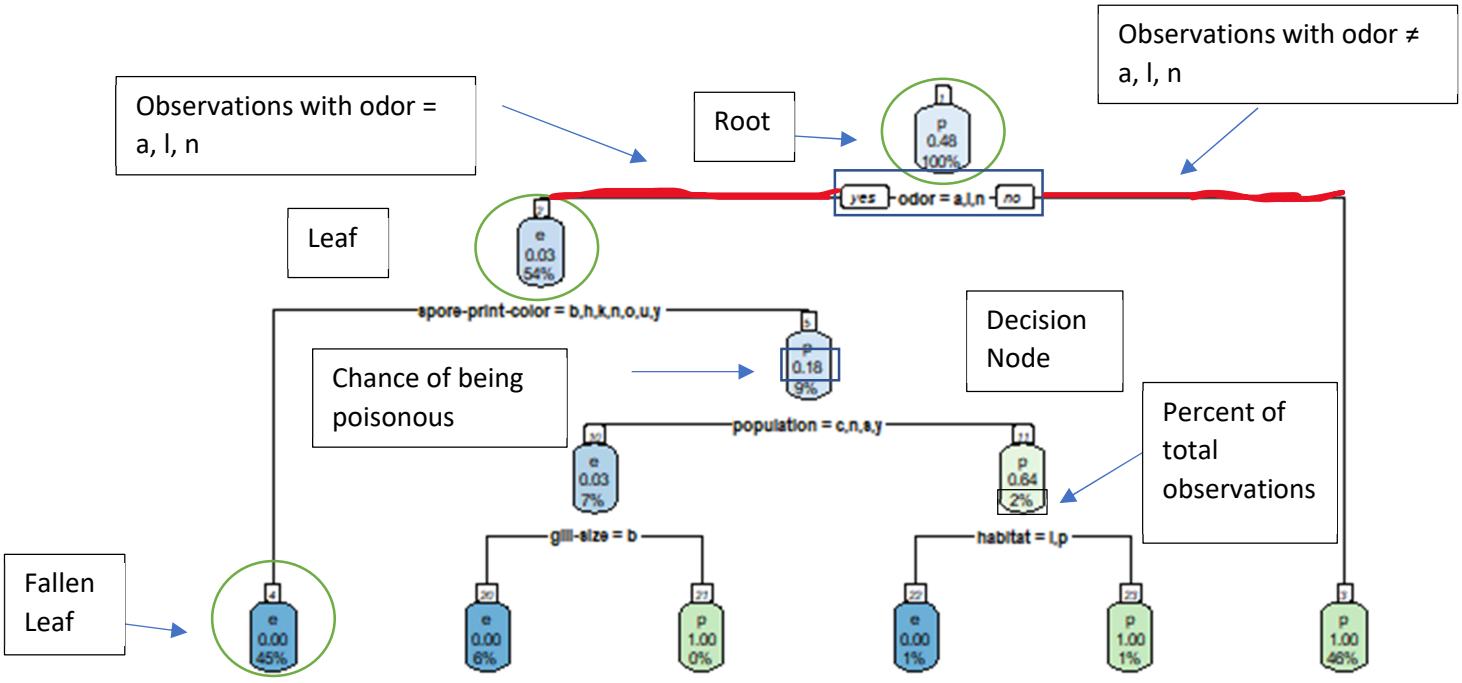
Detection rate shows the portion of correct positive class predictions made as a portion of all predictions.

Detection prevalence shows the number of positive class prediction made as a proportion of all predictions

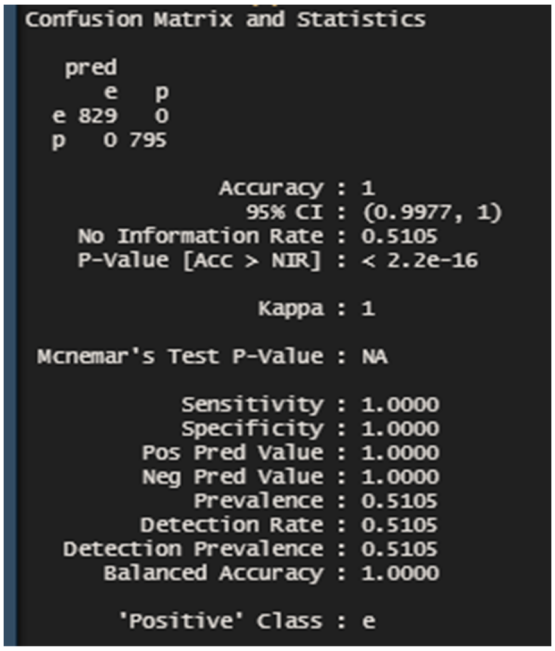
Balance accuracy take the average of the true positive and true negative rates or the sensitivity and specificity.

Analysis

Based on the data and the output provided in the code walkthrough, we have determined a 100% way of predicting whether a mushroom is poisonous or edible. Let us take another look at the decision tree.



The root tells us that 48% of the mushrooms in our dataset were poisonous. Our fallen leaves, being that they have either a 0 or 100% of being poisonous based on however many variables is very informative to foragers.



The confusion matrix, which tells us the performance of our model, suggests the same. That our model was very accurate.

Interpretation and Recommendations

Although the R code suggests the model was 100% accurate, it may be difficult to determine the exact, population, color, gill-size, etc. when out in the wilderness. If I were to become a forager, I would find it worthwhile know the area and the species of mushrooms found there before picking up anything that may harm me and/or whoever I am with. However, this dataset and the decision tree we created may be useful for anyone looking to start foraging for mushrooms.

Appendix: R Code

#Installing libraries

install.packages('rpart')

install.packages('caret')

install.packages('rpart.plot')

install.packages('rattle')

install.packages('readxl')

```
#Loading libraries
library(rpart,quietly = TRUE)
library(caret,quietly = TRUE)
library(rpart.plot,quietly = TRUE)
library(rattle)
library(readxl)



setwd("C:/Users/Scott/Desktop/ALY6040")
#Reading the data set as a dataframe

mushrooms <- read_excel("mushrooms.xlsx")

# structure of the data
str(mushrooms)



# number of rows with missing values
nrow(mushrooms) - sum(complete.cases(mushrooms))

# deleting redundant variable `veil.type`
mushrooms$'veil-type' <- NULL

View(mushrooms)

#analyzing the odor variable
table(mushrooms$class,mushrooms$odor)
```

```
number.perfect.splits <- apply(X=mushrooms[-1], MARGIN = 2, FUN = function(col){
  t <- table(mushrooms$class,col)
  sum(t == 0)
})
 number.perfect.splits

# Descending order of perfect splits
order <- order(number.perfect.splits,decreasing = TRUE)
number.perfect.splits <- number.perfect.splits[order]

# Plot graph
par(mar=c(10,2,2,2))
barplot(number.perfect.splits,
        main="Number of perfect splits vs feature",
        xlab="",ylab="Feature",las=2,col="wheat")


#data splicing
set.seed(12345)
train <- sample(1:nrow(mushrooms),size = ceiling(0.80*nrow(mushrooms)),replace = FALSE)
# training set
mushrooms_train <- mushrooms[train,]
# test set
mushrooms_test <- mushrooms[-train,]


# penalty matrix
```

```
penalty.matrix <- matrix(c(0,1,10,0), byrow=TRUE, nrow=2)

penalty.matrix
# building the classification tree with rpart
tree <- rpart(class~.,
              data=mushrooms_train,
              parms = list(loss = penalty.matrix),
              method = "class")


# Visualize the decision tree with rpart.plot
rpart.plot(tree, nn=TRUE)


# choosing the best complexity parameter "cp" to prune the tree
cp.optim <- tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"]
# tree prunning using the best complexity parameter. For more in
tree <- prune(tree, cp=cp.optim)


#Testing the model
pred <- predict(object=tree,mushrooms_test[-1],type="class")


#Calculating accuracy
t <- table(mushrooms_test$class,pred)
confusionMatrix(t)
```