ALY 6040 Module 2 Technique Practice

Student's name: Mohammad Hossein Movahedi

Assignment title: Module 2 Technique Practice

Course number and title: ALY6040 71368 Data Mining Applications SEC 09 Fall 2022 CPS [TOR-A-HY]

Term: 202315_A Fall 2022 CPS Quarter First Half

Instructor's name: Hootan Kamran, Ph.D.

Oct 2, 2022,

Introduction

In this assignment, I will go through the code and data provided to mine data using the decision tree method. The final result is classifying and distinguishing toxic mushrooms based on different attributes of mushrooms.

There are five main steps in this project.

1. Doing EDA
2. Finding splits
3. Dividing data into testing and training groups
4. Creating a decision tree
5. Examining the tree and validation

Code walkthrough

Cleaning the Data (EDA)

The first part of the Code is Just Installing and loading the Libraries.

```
print('Mohammad Hossein Movahedi')
print('Module 2 Technique Practice')

#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)
```

In the next part, we load the dataset and look into it's structure

```
#Reading the data set as a dataframe

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

```
# structure of the data
str(mushrooms)
```

The result is shown below

```
> str(mushrooms)
tibble [8,124 × 23] (S3: tbl_df/tbl/data.frame)
 $ class                    : chr [1:8124] "p" "e" "e" "p" ...
 $ cap-shape                : chr [1:8124] "x" "x" "b" "x" ...
 $ cap-surface              : chr [1:8124] "s" "s" "s" "y" ...
 $ cap-color                : chr [1:8124] "n" "y" "w" "w" ...
 $ bruises                  : chr [1:8124] "t" "t" "t" "t" ...
 $ odor                     : chr [1:8124] "p" "a" "l" "p" ...
 $ gill-attachment          : chr [1:8124] "f" "f" "f" "f" ...
 $ gill-spacing             : chr [1:8124] "c" "c" "c" "c" ...
 $ gill-size                : chr [1:8124] "n" "b" "b" "n" ...
 $ gill-color               : chr [1:8124] "k" "k" "n" "n" ...
 $ stalk-shape              : chr [1:8124] "e" "e" "e" "e" ...
 $ stalk-root               : chr [1:8124] "e" "c" "c" "e" ...
 $ stalk-surface-above-ring: chr [1:8124] "s" "s" "s" "s" ...
 $ stalk-surface-below-ring: chr [1:8124] "s" "s" "s" "s" ...
 $ stalk-color-above-ring  : chr [1:8124] "w" "w" "w" "w" ...
 $ stalk-color-below-ring  : chr [1:8124] "w" "w" "w" "w" ...
 $ veil-type                : chr [1:8124] "p" "p" "p" "p" ...
 $ veil-color               : chr [1:8124] "w" "w" "w" "w" ...
 $ ring-number              : chr [1:8124] "o" "o" "o" "o" ...
 $ ring-type                : chr [1:8124] "p" "p" "p" "p" ...
 $ spore-print-color        : chr [1:8124] "k" "n" "n" "k" ...
 $ population               : chr [1:8124] "s" "n" "n" "s" ...
 $ habitat                  : chr [1:8124] "u" "g" "m" "u" ...
```

As can be seen, the dataset is loaded as a tribble table, and it has 23 columns and 8124 rows. All of the columns are formatted as “chr.”

After this, we look at null data and duplicates and delete duplicates.

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

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

Now, we can move to the next part

Finding splits

```
#counting perfect splits

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

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

The apply function used in the first line applies the FUN function on every columns except for first column with is the target (class of mushroom)

The FUN function first counts the occurrence of each level in every column and and two levels of class column which is being poisons or edible.
Then it checks for the levels that don't connect with any level of class and count them.

So the final results of number.perfect.splits is equal to the number of splits that each column needs to fully separate toxic mushrooms from edible ones.

So the final result is shown below

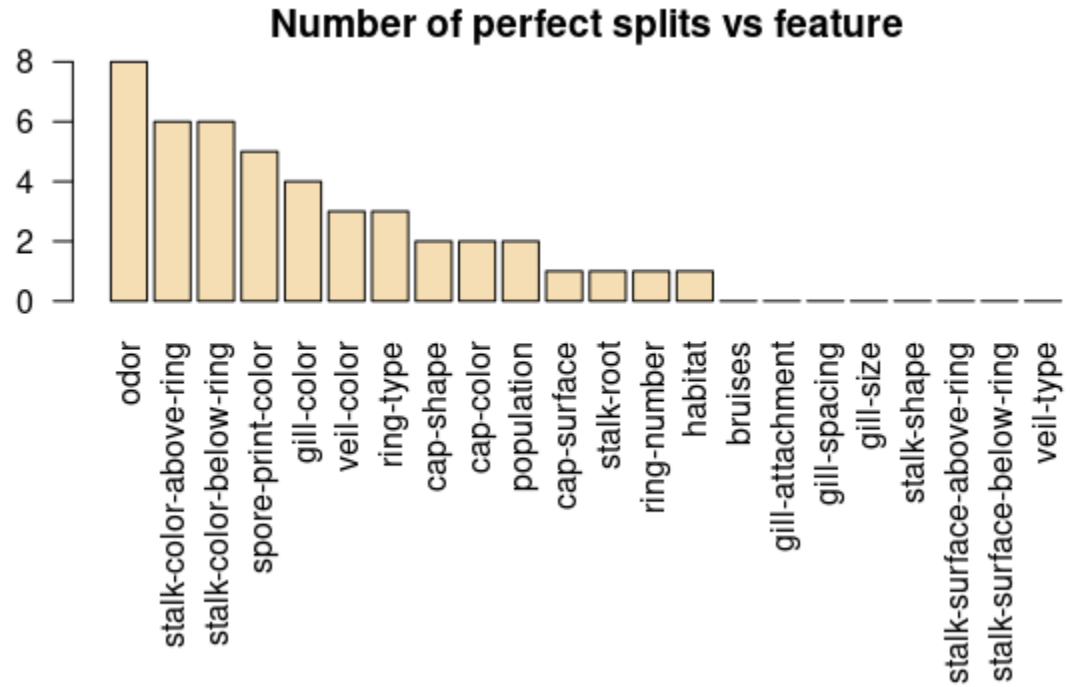
```
number.perfect.splits
               cap-shape              cap-surface                cap-color
                       2                        1                        2
                 bruises                     odor          gill-attachment
                       0                        8                        0
            gill-spacing               gill-size               gill-color
                       0                        0                        4
             stalk-shape              stalk-root stalk-surface-above-ring
                       0                        1                        0
stalk-surface-below-ring   stalk-color-above-ring   stalk-color-below-ring
                       0                        6                        6
               veil-type              veil-color             ring-number
                       0                        3                        1
               ring-type       spore-print-color               population
                       3                        5                        2
                 habitat
                       1
```

As can be seen, some attributes like stalk-shape can't split the data based of only them

```
# 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")
```

In this part we order the output of last part and plot it for visual effect.



The graph above shows the number of perfect splits that can be done with each attribute. As can be seen, some of them are zero, and the largest amount belongs to odor with eight perfect splits.

At this point, we can start splitting the data.but first, we need to create our training and testing dataset

Data slicing

```
#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 code above splices the data by random putting 80% of data in training set and the rest in testing set to avoid overfitting

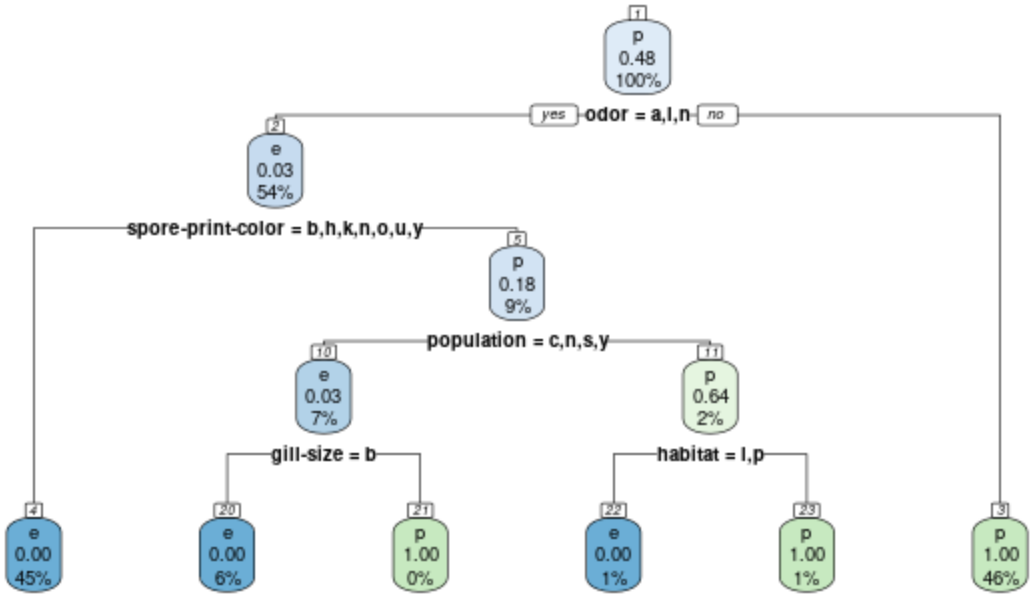
Creating the decision tree

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


# 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)
```

The creation fo decision tree is done by rpart() function. It first needs a penalty matrix that has default values. Then the function works and creates the decision tree on the training dataset, and at last, we plot the final results.



As can be seen, the odor alone did a good job detecting 46% of poisonous mushrooms. The spore-print color can distinguish 45% of edible mushrooms initially found edible by odor.

```
# choosing the best complexity parameter "cp" to prune the tree
cp.optim <- tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"]
```

What this part does is find which number of splitting has the smallest xerror and the find the CP of that split and return it.

```
# tree pruning using the best complexity parameter. For more in
tree <- prune(tree, cp=cp.optim)
```

we used the collected CP to prune the tree, which doesn't change our tree because we got lucky.

Validation

At last we test the tree on the testing dataset

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

We used the tree to predict the class based on attributes. And then we calculate the accuracy of the model

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

The resulting confusion matrix is shown below

```
> confusionMatrix(t)
Confusion Matrix and Statistics

   pred
      e   p
  e 829   0
  p   0 795

               Accuracy : 1
                 95% CI : (0.9977, 1)
    No Information Rate : 0.5105
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.5105
         Detection Rate : 0.5105
   Detection Prevalence : 0.5105
      Balanced Accuracy : 1.0000

       'Positive' Class : e
```

The tree has worked amazingly on our testing dataset and predicted with 100% sensitivity and Specificity meaning there where no error at all.

Conclusion

We got so lucky in this specific example. A decision tree is a potent tool; however, accuracy is pure luck, and maybe with different random numbers and initial data selection, we get different results.

References

Pluralsight.com. (2020). *Explore R Libraries: Rpart | Pluralsight*. [online] Available at: https://www.pluralsight.com/guides/explore-r-libraries:-rpart [Accessed 28 Sep. 2022].

Guru99. (2020). *Decision Tree in R: Classification Tree with Example*. [online] Available at: https://www.guru99.com/r-decision-trees.html [Accessed 28 Sep. 2022].

Tutorialspoint.com. (2022). *R - Decision Tree*. [online] Available at: https://www.tutorialspoint.com/r/r_decision_tree.htm [Accessed 28 Sep. 2022].

Appendix

```
print('Mohammad Hossein Movahedi')
print('Module 2 Technique Practice')

#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)

#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




#counting perfect splits

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

```
})

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

# 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)


# 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"]
```

```r
# 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)
```