Coursework
Data Science Development (CMM535)

Andrew Tait, *1504693@rgu.ac.uk*

April 22, 2018

---

1 Data Exploration

1.1 Dataset Choice

The dataset that has been chosen for this part of the coursework is Mushroom. This is available on the UCI repository. The set was chosen because of it's adequate instance size and number of attributes.
http://archive.ics.uci.edu/ml/datasets/Mushroom

1.2 Problem Statement and Data Exploration

The main purpose of the Mushroom dataset is to identify which characteristics (attributes) determine if a particular mushroom species is editable or poisonous.
Therefore the aim of this assignment is to build a predictive model to predict if a certain type of Mushroom is ediable or not.
To start off the data explortation I will first import the required librarys.

```
#Import packages
library(randomForest)
library(e1071)
library(caret)
library(ggplot2)
library(gridExtra)
library(caret)
library(rpart.plot)
library(RColorBrewer)
library(plyr)
library(dplyr)
library(doParallel)
```

Then set the working directory to the Coursework project folder path:

```
setwd("~/CMM535 Data Science Development/Coursework/CMM535_Coursework")
```

In order to import the dataset, I used a third party helper function, which can be viewed at (Figure 4.1) . The helper function not only set the attributes names but the instances names as well. Since all the data is represented a single character, it converts them into their string equivalent.

```r
#helper function
source('helper_functions.r')

#Import datasets using helper function

mushroom <- fetchAndCleanData()
```

Now that the dataset is imported, it is time to do some data exploration and analysis.
Number of rows in the dataset:

```r
#Number of rows in the dataset
nrow(mushroom)
```

```
## [1] 8124
```

Number of columns (features) in the dataset:

```r
ncol(mushroom)
```

```
## [1] 23
```

Summary of the Mushroom dataset:

```r
#Summary of the Mushroom dataset
str(mushroom)
```

```
## 'data.frame': 8124 obs. of  23 variables:
##  $ Edible               : Factor w/ 2 levels "Edible","Poisonous": 2 1 1 2 1 1 1 1 2 1 ...
##  $ CapShape             : Factor w/ 12 levels "b","c","f","k",..: 9 9 7 9 9 9 7 7 9 7 ...
##  $ CapSurface           : Factor w/ 8 levels "f","g","s","y",..: 8 8 8 7 8 7 8 7 7 8 ...
##  $ CapColor             : Factor w/ 20 levels "b","c","e","g",..: 11 20 19 19 14 20 19 19 19 20 ...
##  $ Bruises              : Factor w/ 4 levels "f","t","True",..: 3 3 3 3 4 3 3 3 3 3 ...
##  $ Odor                 : Factor w/ 18 levels "a","c","f","l",..: 17 10 11 17 16 10 10 11 17 10 ...
##  $ GillAttachment       : Factor w/ 6 levels "a","f","Attached",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ GillSpacing          : Factor w/ 5 levels "c","w","Close",..: 3 3 3 3 4 3 3 3 3 3 ...
##  $ GillSize             : Factor w/ 4 levels "b","n","Broad",..: 4 3 3 4 3 3 3 3 4 3 ...
##  $ GillColor            : Factor w/ 24 levels "b","e","g","h",..: 13 13 14 14 13 14 17 14 20 17 ...
##  $ StalkShape           : Factor w/ 4 levels "e","t","Enlarging",..: 3 3 3 3 4 3 3 3 3 3 ...
##  $ StalkRoot            : Factor w/ 12 levels "?","b","c","e",..: 9 7 7 9 9 7 7 7 9 7 ...
##  $ StalkSurfaceAboveRing: Factor w/ 8 levels "f","k","s","y",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ StalkSurfaceBelowRing: Factor w/ 8 levels "f","k","s","y",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ StalkColorAboveRing  : Factor w/ 18 levels "b","c","e","g",..: 17 17 17 17 17 17 17 17 17 17 ...
##  $ StalkColorBelowRing  : Factor w/ 18 levels "b","c","e","g",..: 17 17 17 17 17 17 17 17 17 17 ...
##  $ VeilType             : Factor w/ 3 levels "p","Partial",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ VeilColor            : Factor w/ 8 levels "n","o","w","y",..: 7 7 7 7 7 7 7 7 7 7 ...
##  $ RingNumber           : Factor w/ 6 levels "n","o","t","None",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ RingType             : Factor w/ 13 levels "e","f","l","n",..: 11 11 11 11 7 11 11 11 11 11 ...
##  $ SporePrintColor      : Factor w/ 18 levels "b","h","k","n",..: 10 11 11 10 11 10 10 11 10 10 ...
##  $ Population           : Factor w/ 12 levels "a","c","n","s",..: 10 9 9 10 7 9 9 10 11 10 ...
##  $ Habitat              : Factor w/ 14 levels "d","g","l","m",..: 12 8 10 12 8 8 10 10 8 10 ...
```

Now that some basic data exploration is covered, next to inspect the dataset a bit further. Starting with the class (Edible) distribution in the mushroom dataset, see (Figure 1)

```r
#Class Distribution
barplot(table(mushroom$Edible))
```

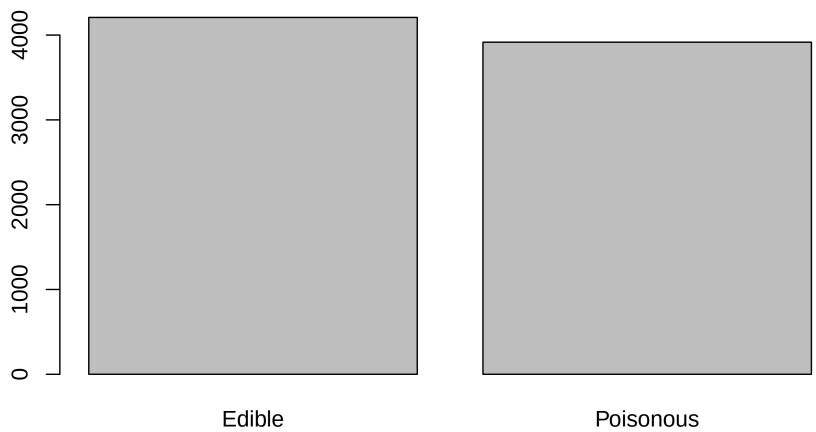

Figure 1: Barplot of Class Distribution

Next is to analyse if there is a correlocation between the CapShape and CapSurface of a mushroom and whether it is Edible or Poisonous. Which is shown in the plot ((Figure 2)) below.

```r
#Comparisons of CapShape and CapSurface with Edible or Poisonous
ggplot(mushroom,aes(x=CapShape, y=CapSurface, color=Edible)) +
                       geom_jitter(alpha=0.3) +
                       scale_color_manual(breaks = c('Edible','Poisonous'),
                                          values=c('darkgreen','red'))
```

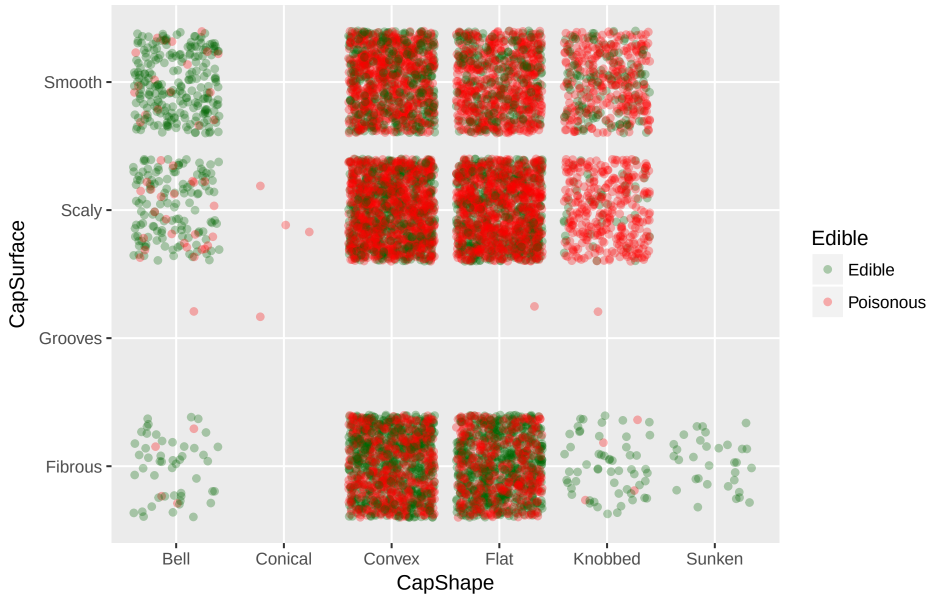

Figure 2: Comparisons of CapShape and CapSurface with Edible or Poisonous in Mushroom Dataset

```
#Comparisons of StalkSurfaceAboveRing and StalkSurfaceBelowRing with Edible or Poisionous
ggplot(mushroom,aes(x=StalkSurfaceAboveRing, y=StalkSurfaceBelowRing, color=Edible)) +
  geom_jitter(alpha=0.3) +
  scale_color_manual(breaks = c('Edible','Poisonous'), values=c('darkgreen','red'))
```

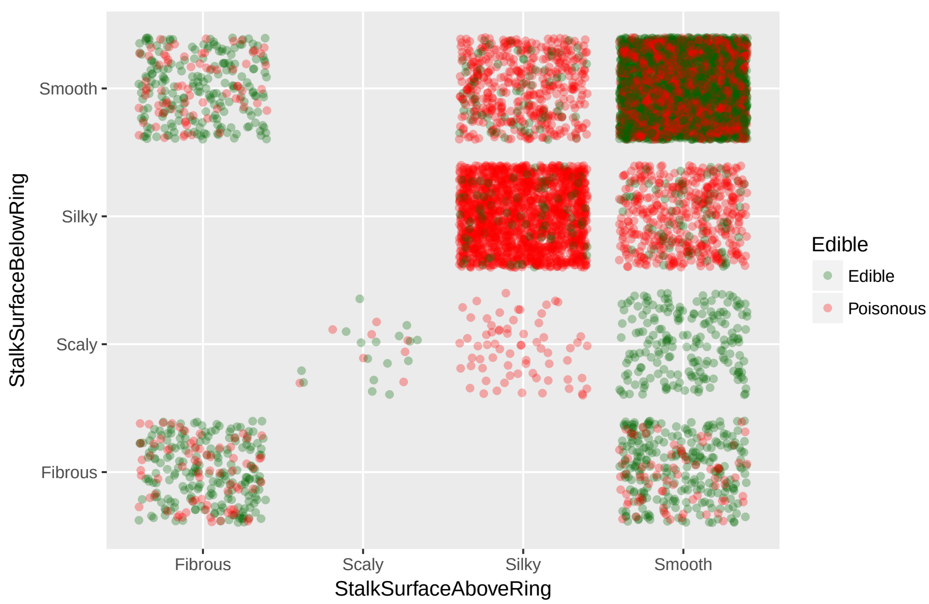

Figure 3: Comparisons of StalkSurfaceAboveRing and StalkSurfaceBelowRing with Edible or Poisionous

1.3 Pre-Processing

Before the Mushroom dataset can be processed by a classification model(s), some pre-processing is required. While the helper function should take out all missing values, lets valdiate this before continuing.

```r
#Class Distribution
table(complete.cases (mushroom))
```

```
##
## TRUE
## 8124
```

As shown above, there is not any missing values in the dataset.

2 Modeling and Classifcation

2.1 Divide into training and testing subset

When it came to dividing the mushroom dataset into training and testing subsets, I decided to go with the conventional 70 percent training and 30 percent testing split as a starting point/baseline.

```
#Divide the datset into 70% training and 30% testing.
inTrain <- createDataPartition(y=mushroom$Edible, p=0.7, list=FALSE)

#Assign indexes to split the Mushroom dataset into training and testing
training <- mushroom[inTrain,]
testing <- mushroom[-inTrain,]
```

2.2 Build Classifier

For the initial classifier I decided to go with the kNN Classifer as it has proven to be a good baseline in previous labs and exercises in R.
Before the classification begins, parallel processing is enabled to speed up this process.

```
#Setup Parallel processing to speed up classification modelling
cl <- makeCluster(detectCores(), type='PSOCK')
registerDoParallel(cl)
```

The train control is set to cross-validation with 5 folds:

```
#set train control to cross-validation with 5 folds
train_control<- trainControl(method="cv", number=5)
```

```
#First set the seed for reproducibility
set.seed(1)

#train model using kNN
kNNModel <- train(Edible ~ ., data = training,
                  trControl = train_control,
                  tuneLength = 20,
                  method = "knn"
)
```

Once the knn Model is complete, it's time to analyse the results, first with a print of the kNNModel as shown below.

```
#Show the kNN model results
kNNModel
```

```
## k-Nearest Neighbors
##
## 5688 samples
##   22 predictor
##    2 classes: 'Edible', 'Poisonous'
##
## No pre-processing
```

```
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 4551, 4549, 4551, 4551, 4550
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.9992964  0.9985907
##    7  0.9989446  0.9978860
##    9  0.9987687  0.9975336
##   11  0.9984169  0.9968289
##   13  0.9980654  0.9961249
##   15  0.9978895  0.9957725
##   17  0.9980654  0.9961249
##   19  0.9975377  0.9950681
##   21  0.9973618  0.9947159
##   23  0.9975377  0.9950683
##   25  0.9971863  0.9943648
##   27  0.9966591  0.9933090
##   29  0.9959561  0.9919015
##   31  0.9954287  0.9908461
##   33  0.9950768  0.9901412
##   35  0.9945492  0.9890853
##   37  0.9940215  0.9880294
##   39  0.9933181  0.9866213
##   41  0.9920878  0.9841591
##   43  0.9908568  0.9816959
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

Next is a confusion matrix is created by predicting the accuracy against the testing subset.

```
#Predict the accuracy of the kNN Model against the testing set
predictkNN <- predict(kNNModel,testing)
confusionMatrix(predictkNN, testing$Edible)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Edible Poisonous
##   Edible      1262         0
##   Poisonous      0      1174
##
##                Accuracy : 1
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.5181
##     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 : 0.5181
##          Detection Rate : 0.5181
##    Detection Prevalence : 0.5181
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : Edible
##
```

2.3 Improve Model Performance

2.3.1 C5.0 Model

```
#First set the seed for reproducibility
set.seed(1)

#train the model using c5.0
c50Model<- train(Edible~., data=training,
                 trControl=train_control,
                 tuneLength=5,
                 method="C5.0"
)
```

```
#Show the c50Model results
 c50Model
```

```
## C5.0
##
## 5688 samples
##   22 predictor
##    2 classes: 'Edible', 'Poisonous'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 4551, 4549, 4551, 4551, 4550
## Resampling results across tuning parameters:
##
##   model  winnow  trials  Accuracy   Kappa
##   rules  FALSE    1      1.0000000  1.0000000
##   rules  FALSE   10      1.0000000  1.0000000
##   rules  FALSE   20      1.0000000  1.0000000
##   rules  FALSE   30      1.0000000  1.0000000
##   rules  FALSE   40      1.0000000  1.0000000
##   rules  FALSE   50      1.0000000  1.0000000
##   rules  FALSE   60      1.0000000  1.0000000
##   rules  FALSE   70      1.0000000  1.0000000
##   rules  FALSE   80      1.0000000  1.0000000
##   rules  FALSE   90      1.0000000  1.0000000
##   rules   TRUE    1      0.9991216  0.9982406
##   rules   TRUE   10      1.0000000  1.0000000
##   rules   TRUE   20      1.0000000  1.0000000
##   rules   TRUE   30      1.0000000  1.0000000
##   rules   TRUE   40      0.9996485  0.9992961
##   rules   TRUE   50      0.9996485  0.9992961
##   rules   TRUE   60      0.9996485  0.9992961
##   rules   TRUE   70      0.9996485  0.9992961
##   rules   TRUE   80      0.9996485  0.9992961
##   rules   TRUE   90      0.9996485  0.9992961
##   tree   FALSE    1      1.0000000  1.0000000
##   tree   FALSE   10      1.0000000  1.0000000
##   tree   FALSE   20      1.0000000  1.0000000
```

```
##   tree   FALSE   30      1.0000000  1.0000000
##   tree   FALSE   40      1.0000000  1.0000000
##   tree   FALSE   50      1.0000000  1.0000000
##   tree   FALSE   60      1.0000000  1.0000000
##   tree   FALSE   70      1.0000000  1.0000000
##   tree   FALSE   80      1.0000000  1.0000000
##   tree   FALSE   90      1.0000000  1.0000000
##   tree    TRUE    1      0.9994728  0.9989440
##   tree    TRUE   10      1.0000000  1.0000000
##   tree    TRUE   20      1.0000000  1.0000000
##   tree    TRUE   30      1.0000000  1.0000000
##   tree    TRUE   40      1.0000000  1.0000000
##   tree    TRUE   50      1.0000000  1.0000000
##   tree    TRUE   60      1.0000000  1.0000000
##   tree    TRUE   70      1.0000000  1.0000000
##   tree    TRUE   80      1.0000000  1.0000000
##   tree    TRUE   90      1.0000000  1.0000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were trials = 1, model = rules
##  and winnow = FALSE.
```

```r
predictC50 <- predict(c50Model, testing)
confusionMatrix(predictC50,testing$Edible)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Edible Poisonous
##   Edible      1262         0
##   Poisonous      0      1174
##
##                Accuracy : 1
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.5181
##     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 : 0.5181
##          Detection Rate : 0.5181
##    Detection Prevalence : 0.5181
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : Edible
##
```

2.3.2 Random forest Model

```
#First set the seed for reproducibility
set.seed(1)

# train the model using random forest
RFModel<- train(Edible~., data=training,
                trControl=train_control,
                method="rf",
                tuneLength =10,
                metric = 'Accuracy'
)
```

```
RFModel
```

```
## Random Forest
##
## 5688 samples
##   22 predictor
##    2 classes: 'Edible', 'Poisonous'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 4551, 4549, 4551, 4551, 4550
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##     2   0.8957470  0.7896431
##    26   1.0000000  1.0000000
##    50   1.0000000  1.0000000
##    75   1.0000000  1.0000000
##    99   1.0000000  1.0000000
##   123   0.9998241  0.9996477
##   148   0.9998241  0.9996477
##   172   0.9996484  0.9992958
##   196   0.9996484  0.9992958
##   221   0.9996484  0.9992958
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 26.
```

```
predictRF <- predict(RFModel,testing)
confusionMatrix(predictRF, testing$Edible)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Edible Poisonous
##   Edible      1262         0
##   Poisonous      0      1174
```

```
##
##                Accuracy : 1
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.5181
##     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 : 0.5181
##          Detection Rate : 0.5181
##    Detection Prevalence : 0.5181
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : Edible
##
```

2.3.3 Comparison of all Models

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Edible Poisonous
##   Edible      1262         0
##   Poisonous      0      1174
##
##                Accuracy : 1
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.5181
##     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 : 0.5181
##          Detection Rate : 0.5181
##    Detection Prevalence : 0.5181
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : Edible
##
```

3 Fine-grained Model

3.1 Clustering Dataset

3.1.1 Pre-processing

```
normalizeData <- function (x ) {
return ( (x-min(x) ) / ( max(x)- min(x) ))
}
```

```
#Copy the dataset before pre-processing
dfNew <- mushroom
dfNew$VeilType <- NULL

#Convert the dataframe to numeric values
dfNew[,2:22] = lapply(dfNew[,2:22], as.numeric)

##Then use the normalise function from above
dfN <- as.data.frame(lapply(dfNew[,-1], normalizeData))
```

3.2 Adapting your Model

4 Appendix

4.1 Mushroom Dataset Helper Function

I used a helper function to import the dataset, it helps with assigning the correct column and row names to the dataset. It also removes any missing values from the dateset.
https://github.com/stoltzmaniac/Mushroom-Classification/blob/master/helper_functions.R

Figure 4: fetchAndCleanData Function for Mushroom dataset

```
fetchAndCleanData = function(){

  # All of this code is from
  # https://rstudio-pubs-static.s3.amazonaws.com/125760_358e4a6802c94fa29e2a9ab49f45df94.html

  mushrooms = read.table("data/agaricus-lepiota.data", header = FALSE, sep = ",")



  #create a data frame with only the required columns
  shrooms = mushrooms

  #column names are added
  colnames(shrooms) = c("Edible",
                        "CapShape",
                        "CapSurface",
                        "CapColor",
                        "Bruises",
                        "Odor",
                        "GillAttachment",
                        "GillSpacing",
                        "GillSize",
                        "GillColor",
                        "StalkShape",
                        "StalkRoot",
                        "StalkSurfaceAboveRing",
                        "StalkSurfaceBelowRing",
                        "StalkColorAboveRing",
                        "StalkColorBelowRing",
                        "VeilType",
                        "VeilColor",
                        "RingNumber",
                        "RingType",
                        "SporePrintColor",
                        "Population",
                        "Habitat")


  #Edible
  shrooms$Edible = as.character(shrooms$Edible)
  shrooms$Edible[shrooms$Edible == "e"] = "Edible"
  shrooms$Edible[shrooms$Edible == 'p'] = "Poisonous"
  shrooms$Edible = factor(shrooms$Edible)


  # Edible
  #levels(shrooms$Edible) = c(levels(shrooms$Edible), c("Poisonous","Edible"))
  #shrooms$Edible[shrooms$Edible == "p"] = "Poisonous"
  #shrooms$Edible[shrooms$Edible == "e"] = "Edible"

  #CapShape
  levels(shrooms$`CapShape`) = c(levels(shrooms$`CapShape`), c("Bell","Conical","Convex","Flat","Knobbed","Sunken"))
  shrooms$`CapShape`[shrooms$`CapShape` == "b"] = "Bell"
  shrooms$`CapShape`[shrooms$`CapShape` == "c"] = "Conical"
  shrooms$`CapShape`[shrooms$`CapShape` == "x"] = "Convex"
  shrooms$`CapShape`[shrooms$`CapShape` == "f"] = "Flat"
  shrooms$`CapShape`[shrooms$`CapShape` == "k"] = "Knobbed"
  shrooms$`CapShape`[shrooms$`CapShape` == "s"] = "Sunken"

  #CapSurface
  levels(shrooms$`CapSurface`) = c(levels(shrooms$`CapSurface`), c("Fibrous", "Grooves", "Scaly", "Smooth"))
  shrooms$`CapSurface`[shrooms$`CapSurface` == "f"] = "Fibrous"
  shrooms$`CapSurface`[shrooms$`CapSurface` == "g"] = "Grooves"
  shrooms$`CapSurface`[shrooms$`CapSurface` == "y"] = "Scaly"
  shrooms$`CapSurface`[shrooms$`CapSurface` == "s"] = "Smooth"

  #CapColor
  levels(shrooms$`CapColor`) = c(levels(shrooms$`CapColor`), c("Brown", "Buff", "Cinnamon", "Gray", "Green", "Pink", "Purple", "Red", "White", "
        Yellow"))
  shrooms$`CapColor`[shrooms$`CapColor` == "n"] = "Brown"
  shrooms$`CapColor`[shrooms$`CapColor` == "b"] = "Buff"
  shrooms$`CapColor`[shrooms$`CapColor` == "c"] = "Cinnamon"
  shrooms$`CapColor`[shrooms$`CapColor` == "g"] = "Gray"
  shrooms$`CapColor`[shrooms$`CapColor` == "r"] = "Green"
  shrooms$`CapColor`[shrooms$`CapColor` == "p"] = "Pink"
  shrooms$`CapColor`[shrooms$`CapColor` == "u"] = "Purple"
  shrooms$`CapColor`[shrooms$`CapColor` == "e"] = "Red"
  shrooms$`CapColor`[shrooms$`CapColor` == "w"] = "White"
  shrooms$`CapColor`[shrooms$`CapColor` == "y"] = "Yellow"

  # Bruises
  levels(shrooms$Bruises) = c(levels(shrooms$Bruises), c("True","False"))
  shrooms$Bruises[shrooms$Bruises == "t"] = "True"
  shrooms$Bruises[shrooms$Bruises == "f"] = "False"

  #Odor
```

```r
levels(shrooms$Odor) = c(levels(shrooms$Odor), c("Almond", "Anise", "Creosote", "Fishy", "Foul", "Musty", "None", "Pungent", "Spicy"))
shrooms$Odor[shrooms$Odor == "a"] = "Almond"
shrooms$Odor[shrooms$Odor == "l"] = "Anise"
shrooms$Odor[shrooms$Odor == "c"] = "Creosote"
shrooms$Odor[shrooms$Odor == "y"] = "Fishy"
shrooms$Odor[shrooms$Odor == "f"] = "Foul"
shrooms$Odor[shrooms$Odor == "m"] = "Musty"
shrooms$Odor[shrooms$Odor == "n"] = "None"
shrooms$Odor[shrooms$Odor == "p"] = "Pungent"
shrooms$Odor[shrooms$Odor == "s"] = "Spicy"

# GillAttachment
levels(shrooms$GillAttachment) = c(levels(shrooms$GillAttachment), c("Attached","Descending","Free","Notched"))
shrooms$GillAttachment[shrooms$GillAttachment == "a"] = "Attached"
shrooms$GillAttachment[shrooms$GillAttachment == "d"] = "Descending"
shrooms$GillAttachment[shrooms$GillAttachment == "f"] = "Free"
shrooms$GillAttachment[shrooms$GillAttachment == "n"] = "Notched"

# GillSpacing
levels(shrooms$GillSpacing) = c(levels(shrooms$GillSpacing), c("Close","Crowded","Distant"))
shrooms$GillSpacing[shrooms$GillSpacing == "c"] = "Close"
shrooms$GillSpacing[shrooms$GillSpacing == "w"] = "Crowded"
shrooms$GillSpacing[shrooms$GillSpacing == "d"] = "Distant"

# GillSize
levels(shrooms$GillSize) = c(levels(shrooms$GillSize), c("Broad","Narrow"))
shrooms$GillSize[shrooms$GillSize == "b"] = "Broad"
shrooms$GillSize[shrooms$GillSize == "n"] = "Narrow"

# GillColor
levels(shrooms$GillColor) = c(levels(shrooms$GillColor), c("Black","Brown","Buff","Chocolate","Gray","Green","Orange","Pink","Purple","Red","
      White","Yellow"))
shrooms$GillColor[shrooms$GillColor == "k"] = "Black"
shrooms$GillColor[shrooms$GillColor == "n"] = "Brown"
shrooms$GillColor[shrooms$GillColor == "b"] = "Buff"
shrooms$GillColor[shrooms$GillColor == "h"] = "Chocolate"
shrooms$GillColor[shrooms$GillColor == "g"] = "Gray"
shrooms$GillColor[shrooms$GillColor == "r"] = "Green"
shrooms$GillColor[shrooms$GillColor == "o"] = "Orange"
shrooms$GillColor[shrooms$GillColor == "p"] = "Pink"
shrooms$GillColor[shrooms$GillColor == "u"] = "Purple"
shrooms$GillColor[shrooms$GillColor == "e"] = "Red"
shrooms$GillColor[shrooms$GillColor == "w"] = "White"
shrooms$GillColor[shrooms$GillColor == "y"] = "Yellow"

# StalkShape
levels(shrooms$StalkShape) = c(levels(shrooms$StalkShape), c("Enlarging","Tapering"))
shrooms$StalkShape[shrooms$StalkShape == "e"] = "Enlarging"
shrooms$StalkShape[shrooms$StalkShape == "t"] = "Tapering"

# StalkRoot
levels(shrooms$StalkRoot) = c(levels(shrooms$StalkRoot), c("Bulbous","Club","Cup","Equal","Rhizomorphs","Rooted","Missing"))
shrooms$StalkRoot[shrooms$StalkRoot == "b"] = "Bulbous"
shrooms$StalkRoot[shrooms$StalkRoot == "c"] = "Club"
shrooms$StalkRoot[shrooms$StalkRoot == "u"] = "Cup"
shrooms$StalkRoot[shrooms$StalkRoot == "e"] = "Equal"
shrooms$StalkRoot[shrooms$StalkRoot == "z"] = "Rhizomorphs"
shrooms$StalkRoot[shrooms$StalkRoot == "r"] = "Rooted"
shrooms$StalkRoot[shrooms$StalkRoot == "?"] = "Missing"

# StalkSurfaceAboveRing
levels(shrooms$StalkSurfaceAboveRing) = c(levels(shrooms$StalkSurfaceAboveRing), c("Fibrous","Scaly","Silky","Smooth"))
shrooms$StalkSurfaceAboveRing[shrooms$StalkSurfaceAboveRing == "f"] = "Fibrous"
shrooms$StalkSurfaceAboveRing[shrooms$StalkSurfaceAboveRing == "y"] = "Scaly"
shrooms$StalkSurfaceAboveRing[shrooms$StalkSurfaceAboveRing == "k"] = "Silky"
shrooms$StalkSurfaceAboveRing[shrooms$StalkSurfaceAboveRing == "s"] = "Smooth"

# StalkSurfaceBelowRing
levels(shrooms$StalkSurfaceBelowRing) = c(levels(shrooms$StalkSurfaceBelowRing), c("Fibrous","Scaly","Silky","Smooth"))
shrooms$StalkSurfaceBelowRing[shrooms$StalkSurfaceBelowRing == "f"] = "Fibrous"
shrooms$StalkSurfaceBelowRing[shrooms$StalkSurfaceBelowRing == "y"] = "Scaly"
shrooms$StalkSurfaceBelowRing[shrooms$StalkSurfaceBelowRing == "k"] = "Silky"
shrooms$StalkSurfaceBelowRing[shrooms$StalkSurfaceBelowRing == "s"] = "Smooth"

# StalkColorAboveRing
levels(shrooms$StalkColorAboveRing) = c(levels(shrooms$StalkColorAboveRing), c("Brown","Buff","Cinnamon","Gray","Orange","Pink","Red","White","
      Yellow"))
shrooms$StalkColorAboveRing[shrooms$StalkColorAboveRing == "n"] = "Brown"
shrooms$StalkColorAboveRing[shrooms$StalkColorAboveRing == "b"] = "Buff"
shrooms$StalkColorAboveRing[shrooms$StalkColorAboveRing == "c"] = "Cinnamon"
shrooms$StalkColorAboveRing[shrooms$StalkColorAboveRing == "g"] = "Gray"
shrooms$StalkColorAboveRing[shrooms$StalkColorAboveRing == "o"] = "Orange"
shrooms$StalkColorAboveRing[shrooms$StalkColorAboveRing == "p"] = "Pink"
shrooms$StalkColorAboveRing[shrooms$StalkColorAboveRing == "e"] = "Red"
shrooms$StalkColorAboveRing[shrooms$StalkColorAboveRing == "w"] = "White"
shrooms$StalkColorAboveRing[shrooms$StalkColorAboveRing == "y"] = "Yellow"


# StalkColorBelowRing
levels(shrooms$StalkColorBelowRing) = c(levels(shrooms$StalkColorBelowRing), c("Brown","Buff","Cinnamon","Gray","Orange","Pink","Red","White","
      Yellow"))
shrooms$StalkColorBelowRing[shrooms$StalkColorBelowRing == "n"] = "Brown"
shrooms$StalkColorBelowRing[shrooms$StalkColorBelowRing == "b"] = "Buff"
shrooms$StalkColorBelowRing[shrooms$StalkColorBelowRing == "c"] = "Cinnamon"
shrooms$StalkColorBelowRing[shrooms$StalkColorBelowRing == "g"] = "Gray"
shrooms$StalkColorBelowRing[shrooms$StalkColorBelowRing == "o"] = "Orange"
shrooms$StalkColorBelowRing[shrooms$StalkColorBelowRing == "p"] = "Pink"
shrooms$StalkColorBelowRing[shrooms$StalkColorBelowRing == "e"] = "Red"
shrooms$StalkColorBelowRing[shrooms$StalkColorBelowRing == "w"] = "White"
shrooms$StalkColorBelowRing[shrooms$StalkColorBelowRing == "y"] = "Yellow"

# VeilType
levels(shrooms$VeilType) = c(levels(shrooms$VeilType), c("Partial","Universal"))
shrooms$VeilType[shrooms$VeilType == "p"] = "Partial"
shrooms$VeilType[shrooms$VeilType == "u"] = "Universal"

# VeilColor
```

```r
  levels(shrooms$VeilColor) = c(levels(shrooms$VeilColor), c("Brown","Orange","White","Yellow"))
  shrooms$VeilColor[shrooms$VeilColor == "n"] = "Brown"
  shrooms$VeilColor[shrooms$VeilColor == "o"] = "Orange"
  shrooms$VeilColor[shrooms$VeilColor == "w"] = "White"
  shrooms$VeilColor[shrooms$VeilColor == "y"] = "Yellow"

  # RingNumber
  levels(shrooms$RingNumber) = c(levels(shrooms$RingNumber), c("None","One","Two"))
  shrooms$RingNumber[shrooms$RingNumber == "n"] = "None"
  shrooms$RingNumber[shrooms$RingNumber == "o"] = "One"
  shrooms$RingNumber[shrooms$RingNumber == "t"] = "Two"

  # RingType
  levels(shrooms$RingType) = c(levels(shrooms$RingType), c("Cobwebby","Evanescent","Flaring","Large","None","Pendant","Sheathing","Zone"))
  shrooms$RingType[shrooms$RingType == "c"] = "Cobwebby"
  shrooms$RingType[shrooms$RingType == "e"] = "Evanescent"
  shrooms$RingType[shrooms$RingType == "f"] = "Flaring"
  shrooms$RingType[shrooms$RingType == "l"] = "Large"
  shrooms$RingType[shrooms$RingType == "n"] = "None"
  shrooms$RingType[shrooms$RingType == "p"] = "Pendant"
  shrooms$RingType[shrooms$RingType == "s"] = "Sheathing"
  shrooms$RingType[shrooms$RingType == "z"] = "Zone"

  # SporePrintColor
  levels(shrooms$SporePrintColor) = c(levels(shrooms$SporePrintColor), c("Black","Brown","Buff","Chocolate","Green","Orange","Purple","White","
        Yellow"))
  shrooms$SporePrintColor[shrooms$SporePrintColor == "k"] = "Black"
  shrooms$SporePrintColor[shrooms$SporePrintColor == "n"] = "Brown"
  shrooms$SporePrintColor[shrooms$SporePrintColor == "b"] = "Buff"
  shrooms$SporePrintColor[shrooms$SporePrintColor == "h"] = "Chocolate"
  shrooms$SporePrintColor[shrooms$SporePrintColor == "r"] = "Green"
  shrooms$SporePrintColor[shrooms$SporePrintColor == "o"] = "Orange"
  shrooms$SporePrintColor[shrooms$SporePrintColor == "u"] = "Purple"
  shrooms$SporePrintColor[shrooms$SporePrintColor == "w"] = "White"
  shrooms$SporePrintColor[shrooms$SporePrintColor == "y"] = "Yellow"

  # Population
  levels(shrooms$Population) = c(levels(shrooms$Population), c("Abundnant","Clustered","Numerous","Scattered","Several","Solitary"))
  shrooms$Population[shrooms$Population == "a"] = "Abundnant"
  shrooms$Population[shrooms$Population == "c"] = "Clustered"
  shrooms$Population[shrooms$Population == "n"] = "Numerous"
  shrooms$Population[shrooms$Population == "s"] = "Scattered"
  shrooms$Population[shrooms$Population == "v"] = "Several"
  shrooms$Population[shrooms$Population == "y"] = "Solitary"

  # Habitat
  levels(shrooms$Habitat) = c(levels(shrooms$Habitat), c("Grasses","Leaves","Meadows","Paths","Urban","Waste","Woods"))
  shrooms$Habitat[shrooms$Habitat == "g"] = "Grasses"
  shrooms$Habitat[shrooms$Habitat == "l"] = "Leaves"
  shrooms$Habitat[shrooms$Habitat == "m"] = "Meadows"
  shrooms$Habitat[shrooms$Habitat == "p"] = "Paths"
  shrooms$Habitat[shrooms$Habitat == "u"] = "Urban"
  shrooms$Habitat[shrooms$Habitat == "w"] = "Waste"
  shrooms$Habitat[shrooms$Habitat == "d"] = "Woods"

  return(shrooms)
}
```

Figure 5: clustData function for clustering Mushroom dataset

```
    #Clustering Function
clustData <- function (df,ClassIndex,kmeansClasses = rep(0,unique(df[,ClassIndex]))) {
  # use split function to split the dataset according to the class label
  # a set of dataframes each representing a class label will be stored
  # in dfs list()
  dfs <- split (df, df[,ClassIndex])
  # create empty list
  clustList <- list()
  n <- length(dfs)
  for (i in 1:length(kmeansClasses)){
    # Cluster according to all features excluding the label
    if (kmeansClasses[i]>1 & kmeansClasses[i]< nrow(dfs[[i]])){
      clustList[[i]] <- kmeans(dfs[[i]][,-ClassIndex],kmeansClasses[i])
      #plotcluster(clustList[[i]], clustList[[i]]$cluster)
      dfs[[i]]$cluster <- paste0((dfs[[i]][,ClassIndex]),
                                 "_","c",clustList[[i]]$cluster)
    }
    else {
      dfs[[i]]$cluster = paste0((dfs[[i]][,ClassIndex]),
                                "_c0")
    }
  }
  # put all list elements in a dataframe and return it
  # note that ldply() require the library plyr
  allClusteredElements <- ldply (dfs, data.frame)
  # drop the first column 'id' resulting from ldply
  allClusteredElements <- allClusteredElements[,-1]
  allClusteredElements <- allClusteredElements[,-ClassIndex]
  return(allClusteredElements)
}
```