Gambling With Nature's Probabilities: A Classification Analysis On Whether Mushrooms Are Poisonous Or Edible

Emeric Szaboky (PSTAT 231 Data Mining)

6/14/2017

Abstract

In this research study, two important questions regarding a dataset on wild mushrooms are addressed. The first question, a supervised learning problem, is whether or not a mushroom (observation) can be accurately classified according to the specific attributes collected in the dataset. The classification analysis is completed using two data mining classification techniques, Random Forest and Logistic Regression. The second question addressed in this study, an unsupervised learning problem, is whether or not categorical clustering methods can accurately group the collection of mushroom observations into clusters revealing their species and familial groupings. For the exploration of this question, two clustering methods have been chosen: ROCK (RObust Clustering using linKs) Algorithm and K-Modes (a modification of K-Means clustering method created for clustering categorical data). K-Modes has not yet been implemented, but will later be added to the report. In addition to the supervised and unsupervised learning analyses, Exploratory Data Analysis (EDA) and visualizations are also used in disecting the dataset and sifting out meaningful information.

The results of both the Random Forest Classification and the Logistic Regression Classifications were 100% accurate with zero percent error rate. This suggests that the attributes available in the dataset are virtually perfect predictors of class. This combination of attributes are shown to have very accurate predictive power. The covariates odor and spore.print.color were shown through the Random Forest Variable Importance Plot and EDA to be the most significant predictors. In combination, they were able to perfectly classify all mushrooms except for odorless mushrooms with a white spore print. The intuitive ROCK Algorithm for clustering with categorical attributes resulted in 21 clusters, 20 of which were pure clusters, in which all observations are either poisonous or edible. There was one mixed cluster, and a number of observations which were not part of a cluster. In analyzing the 21 clusters found, there is significant possibility that these 21 clusters represent 21 of the 23 documented mushroom species, the unclustered observations belonging to the final two species which are indistinguishable.

Introduction

This study, Gambling With Nature's Probabilities: A Classification Analysis On Whether Mushrooms Are Poisonous Or Edible, documents a classification analysis performed on a dataset about wild mushrooms from the University of California Irvine Machine Learning Repository. The dataset, called the "Mushroom Data Set," consists of 8124 observations (hypothetical samples) of mushroom sightings. The observations are sourced originally from The Audubon Society Field Guide to North American Mushrooms (1981). G. H. Lincoff (Pres.), New York: Alfred A. Knopf. The samples correspond to repetitions of 23 different mushroom species within either the Agaricus or Lepiota Families. The observations do not have Family grouping or species as documented attributes; it is unspecified which mushrooms belong to which groupings.

In the original dataset, the class variable had four class levels: definitely edible, definitely poisonous, unknown edibility, or not recommended. The latter two class levels, however, were combined with the poisonous class, making the class variable binary in factor level: edible or poisonous. This intruiging dataset offers much unique knowledge regarding the nature and tendencies of wild mushrooms (and other fungi). With deep

exploration, the data scientist is able to learn how to fairly accurately identify edible mushrooms, a first foot into the world of mycology and mushroom hunting. Although this guide may be a helpful resource and introduction to the understanding of fungi, it is not meant to be a simple rubric for the application of determining mushroom edibility. There will always be risk incorporated in ingestion of wild mushrooms.

The main problem in this study is a classification problem using the dataset's 23 attributes on whether the documented mushrooms are poisonous or edible. The methods used to approach this problem are Random Forest Classification and Logistic Regression Classification, as discussed in the Abstract. The secondhand problem focuses on finding clusters of observations that represent groupings based on mushroom species or Family. The categorical clustering method ROCK Algorithm is used in the investigation of this problem. Another categorical clustering method, K-Modes will be implemented later on. Both forms of classification analysis revealed that the covariate attributes in the dataset were perfect predictors of class. The ROCK Algorithm synthesized 21 clusters, 20 of which were pure clusters (either all poisonous or all edible). These 21 clusters could potentially be groupings aligning with the 23 documented species in the dataset. It also left 127 observations unclustered; these observations are indistinguishable or unable to be grouped with the conceptual linkage methods used in the ROCK Algorithm. Perhaps, these unclustered observations are the ambiguous odorless white spore print mushrooms which had so much difficulty being classified in the classification analysis.

Software programs R, R-Studio, and R Markdown were utilized for the compilation of this report. The packages installed for the completion of this study include plyr, dplyr, ggplot2, caret, randomForest, readr, caTools, rpart.plot, ROCR, class, tree, pROC, cba, and klaR. The following report will include important code excerpts used to run supervised and unsupervised learning analyses, documentation and mechanisms of machine learning methods, in-depth analysis of methods and results, documentation of unique findings discovered through EDA, insightful visualizations with intent of easy digestion of the meaning in the data, model selection techniques, and the overarching conclusion.

Wild Mushroom Dataset Characteristics

Attributes: 23

Observations: 8124

Families: Agaricus, Lepiota

Number of Species Documented: 23

Binary Class Variable: class(poisonous, edible)

Pre-processing

Libraries Utilized

```
library(plyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
library(ggplot2) # beautiful plots
library(caret)  # shortcut analysis methods
## Loading required package: lattice
library(randomForest)  # random Forest analysis
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
## The following object is masked from 'package:dplyr':
##
##     combine
library(readr) # CSV file I/O, e.g. the read_csv function
library(caTools)   # For stratified split
library(rpart.plot)
## Loading required package: rpart
library(ROCR) # ROC curves
## Loading required package: gplots
##
## Attaching package: 'gplots'
## The following object is masked from 'package:stats':
##
##     lowess
library(class)
library(tree)
library(pROC) # ROC curves
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
library(cba)  # package for ROCK clustering algorithm
## Loading required package: grid
```

```
## Loading required package: proxy
##
## Attaching package: 'proxy'
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
## The following object is masked from 'package:base':
##
##     as.matrix
```

```
library(klaR) # package for K-Modes categorical clustering
```

```
## Loading required package: MASS
##
## Attaching package: 'MASS'
## The following object is masked from 'package:dplyr':
##
##     select
```

Read In Data As "mush" / Create Copy Of Dataframe "mush.desc" As Descriptive Reference (Running Certain Analysis Methods On Dataset With Detailed Descriptors Required Significant RAM And Crashes R-Studio)

```
# Read in csv file as a data frame
mush = read.csv('mushrooms.csv')

# Make copy of dataframe labeled "mush.desc"; this will be our descriptive dataset for use in visualiza
mush.desc <- mush
```

Make Duplicate Dataframe "mush.desc" More Descriptive: Convert Shorthand Symbolic Labeling To True Descriptive Labeling

```
### Make Data Frame More Descriptive (check mush.desc for description reference)

#_____________________________________________________________________________________________________

# Alter Class Labels, more descriptive: edible; poisonous
mush.desc$class <- revalue(mush$class, c("e"="edible", "p"="poisonous"))

# Alter cap.shape Attribute (using plyr function revalue)
mush.desc$cap.shape <- revalue(mush$cap.shape, c("b"="bell", "c"="conical", "x"="convex", "f"="flat", "

# Alter cap.surface Attribute
mush.desc$cap.surface <- revalue(mush$cap.surface, c("f"="fibrous", "g"="grooves", "y"="scaly", "s"="sm

# Alter cap.color Attribute
mush.desc$cap.color <- revalue(mush$cap.color, c("n"="brown", "b"="buff", "c"="cinnamon", "g"="gray", "

# Alter bruises Attribute (binary factor covariate)
```

```
mush.desc$bruises <- revalue(mush$bruises, c("t"="bruises", "f"="none"))

# Alter odor Attribute
mush.desc$odor <- revalue(mush$odor, c("a"="almond", "l"="anise", "c"="creosote", "y"="fishy", "f"="foul

# Alter gill.attachment Attribute; "notched", "descending" not present
mush.desc$gill.attachment <- revalue(mush$gill.attachment, c("a"="attached", "f"="free","n"="notched","d
```

```
## The following `from` values were not present in `x`: n, d
```

```
# Alter gill.spacing Attribute; "distant" not present
mush.desc$gill.spacing <- revalue(mush$gill.spacing, c("w"="crowded", "c"="close","d"="distant"))
```

```
## The following `from` values were not present in `x`: d
```

```
# Alter gill.size Attribute (binary factor covariate)
mush.desc$gill.size <- revalue(mush$gill.size, c("b"="broad", "n"="narrow"))

# Alter gill.color Attribute
mush.desc$gill.color <- revalue(mush$gill.color, c("k"="black", "n"="brown", "b"="buff", "h"="chocolate

# Alter stalk.shape Attribute (binary factor covariate)
mush.desc$stalk.shape <- ifelse(mush$stalk.shape == "e","enlarging","tapering")

# Alter stalk.root Attribute; "cup", "rhizomporphs" not present
## missing vals -> labeled as factor level undocumented
mush.desc$stalk.root <- revalue(mush$stalk.root, c("b"="bulbous", "c"="club", "u"="cup", "e"="equal", "z
```

```
## The following `from` values were not present in `x`: u, z
```

```
# Alter stalk.surface.above.ring Attribute
mush.desc$stalk.surface.above.ring <- revalue(mush$stalk.surface.above.ring, c("f"="fibrous", "y"="scaly

# Alter stalk.surface.below.ring Attribute
mush.desc$stalk.surface.below.ring <- revalue(mush$stalk.surface.below.ring, c("f"="fibrous", "y"="scaly

# Alter stalk.color.above.ring Attribute
mush.desc$stalk.color.above.ring <- revalue(mush$stalk.color.above.ring, c("n"="brown", "b"="buff", "c"=

# Alter stalk.color.below.ring Attribute
mush.desc$stalk.color.below.ring <- revalue(mush$stalk.color.below.ring, c("n"="brown", "b"="buff", "c"=

# Alter veil.type Attribute (binary factor covariate); "universal" not present
mush.desc$veil.type <- revalue(mush$veil.type, c("p"="partial", "u"="universal"))
```

```
## The following `from` values were not present in `x`: u
```

```
# Alter veil.color Attribute
mush.desc$veil.color <- revalue(mush$veil.color, c("n"="brown", "o"="orange", "w"="white", "y"="yellow"

# Alter ring.number Attribute
mush.desc$ring.number <- revalue(mush$ring.number, c("n"="none", "o"="one", "t"="two"))

# Alter ring.type Attribute; "cobwebby", "sheathing", "zone" not present
mush.desc$ring.type <- revalue(mush$ring.type, c("c"="cobwebby", "e"="evanescent", "f"="flaring", "l"="
```

```
## The following `from` values were not present in `x`: c, s, z
```

```
# Alter spore.print.color Attribute
mush.desc$spore.print.color <- revalue(mush$spore.print.color, c("k"="black", "n"="brown", "b"="buff",

# Alter population Attribute
mush.desc$population <- revalue(mush$population, c("a"="abundant", "c"="clustered", "n"="numerous", "s"=

# Alter habitat Attribute
mush.desc$habitat <- revalue(mush$habitat, c("g"="grasses", "l"="leaves", "m"="meadows", "p"="paths", "

#-------------------------------------------------------------------------------------------------------
```

Missing values for the stalk.root attribute, originally labeled "?", were made to be their own factor level named "undocumented." Error messages are due to attribute factor levels with 0 observations in them.

Descriptive Dataframe Preview: "mush.desc"

```
# Dataframe Preview
mush.desc[1:15,]  # print first 15 rows of mush.desc
```

```
##        class cap.shape cap.surface cap.color bruises    odor
## 1  poisonous    convex      smooth     brown bruises pungent
## 2     edible    convex      smooth    yellow bruises  almond
## 3     edible      bell      smooth     white bruises   anise
## 4  poisonous    convex       scaly     white bruises pungent
## 5     edible    convex      smooth      gray    none    none
## 6     edible    convex       scaly    yellow bruises  almond
## 7     edible      bell      smooth     white bruises  almond
## 8     edible      bell       scaly     white bruises   anise
## 9  poisonous    convex       scaly     white bruises pungent
## 10    edible      bell      smooth    yellow bruises  almond
## 11    edible    convex       scaly    yellow bruises   anise
## 12    edible    convex       scaly    yellow bruises  almond
## 13    edible      bell      smooth    yellow bruises  almond
## 14 poisonous    convex       scaly     white bruises pungent
## 15    edible    convex     fibrous     brown    none    none
##    gill.attachment gill.spacing gill.size gill.color stalk.shape
## 1             free        close    narrow      black   enlarging
## 2             free        close     broad      black   enlarging
## 3             free        close     broad      brown   enlarging
## 4             free        close    narrow      brown   enlarging
## 5             free      crowded     broad      black    tapering
## 6             free        close     broad      brown   enlarging
## 7             free        close     broad       gray   enlarging
## 8             free        close     broad      brown   enlarging
## 9             free        close    narrow       pink   enlarging
## 10            free        close     broad       gray   enlarging
## 11            free        close     broad       gray   enlarging
## 12            free        close     broad      brown   enlarging
## 13            free        close     broad      white   enlarging
## 14            free        close    narrow      black   enlarging
## 15            free      crowded     broad      brown    tapering
##    stalk.root stalk.surface.above.ring stalk.surface.below.ring
## 1       equal                   smooth                   smooth
```

```
## 2        club                   smooth                   smooth
## 3        club                   smooth                   smooth
## 4       equal                   smooth                   smooth
## 5       equal                   smooth                   smooth
## 6        club                   smooth                   smooth
## 7        club                   smooth                   smooth
## 8        club                   smooth                   smooth
## 9       equal                   smooth                   smooth
## 10       club                   smooth                   smooth
## 11       club                   smooth                   smooth
## 12       club                   smooth                   smooth
## 13       club                   smooth                   smooth
## 14      equal                   smooth                   smooth
## 15      equal                   smooth                  fibrous
##    stalk.color.above.ring stalk.color.below.ring veil.type veil.color
## 1                   white                  white   partial      white
## 2                   white                  white   partial      white
## 3                   white                  white   partial      white
## 4                   white                  white   partial      white
## 5                   white                  white   partial      white
## 6                   white                  white   partial      white
## 7                   white                  white   partial      white
## 8                   white                  white   partial      white
## 9                   white                  white   partial      white
## 10                  white                  white   partial      white
## 11                  white                  white   partial      white
## 12                  white                  white   partial      white
## 13                  white                  white   partial      white
## 14                  white                  white   partial      white
## 15                  white                  white   partial      white
##    ring.number  ring.type spore.print.color population habitat
## 1          one    pendant             black  scattered   urban
## 2          one    pendant             brown   numerous grasses
## 3          one    pendant             brown   numerous meadows
## 4          one    pendant             black  scattered   urban
## 5          one evanescent             brown   abundant grasses
## 6          one    pendant             black   numerous grasses
## 7          one    pendant             black   numerous meadows
## 8          one    pendant             brown  scattered meadows
## 9          one    pendant             black    several grasses
## 10         one    pendant             black  scattered meadows
## 11         one    pendant             brown   numerous grasses
## 12         one    pendant             black  scattered meadows
## 13         one    pendant             brown  scattered grasses
## 14         one    pendant             brown    several   urban
## 15         one evanescent             black   abundant grasses
```

This alternate dataframe can be used for descriptively labeled visualizations.

Exploratory Data Analysis Part 1: Disecting The Data (Part Of Pre-processing)

Statistical Summary Of Mushroom Dataset

```
# Summarize Data
summary(mush.desc)  # Tally Summary of Number of Mushrooms In Each Factor Level of Each Covariate/Attri
```

```
##        class        cap.shape     cap.surface     cap.color
##  edible   :4208   bell   : 452   fibrous:2320   brown  :2284
##  poisonous:3916   conical:   4   grooves:   4   gray   :1840
##                   flat   :3152   smooth :2556   red    :1500
##                   knobbed: 828   scaly  :3244   yellow :1072
##                   sunken :  32                  white  :1040
##                   convex :3656                  buff   : 168
##                                                 (Other): 220
##     bruises          odor      gill.attachment  gill.spacing
##  none   :4748   none   :3528   attached: 210   close  :6812
##  bruises:3376   foul   :2160   free    :7914   crowded:1312
##                 spicy  : 576
##                 fishy  : 576
##                 almond : 400
##                 anise  : 400
##                 (Other): 484
##   gill.size        gill.color   stalk.shape              stalk.root
##  broad :5612   buff     :1728   Length:8124        undocumented:2480
##  narrow:2512   pink     :1492   Class :character   bulbous     :3776
##                white    :1202   Mode  :character   club        : 556
##                brown    :1048                      equal       :1120
##                gray     : 752                      rooted      : 192
##                chocolate: 732
##                (Other)  :1170
##  stalk.surface.above.ring stalk.surface.below.ring stalk.color.above.ring
##  fibrous: 552             fibrous: 600             white  :4464
##  silky  :2372             silky  :2304             pink   :1872
##  smooth :5176             smooth :4936             gray   : 576
##  scaly  :  24             scaly  : 284             brown  : 448
##                                                    buff   : 432
##                                                    orange : 192
##                                                    (Other): 140
##  stalk.color.below.ring   veil.type     veil.color   ring.number
##  white  :4384           partial:8124   brown :  96   none:  36
##  pink   :1872                          orange:  96   one :7488
##  gray   : 576                          white :7924   two : 600
##  brown  : 512                          yellow:   8
##  buff   : 432
##  orange : 192
##  (Other): 156
##       ring.type    spore.print.color     population      habitat
##  evanescent:2776   white    :2388    abundant : 384   woods  :3148
##  flaring   :  48   brown    :1968    clustered: 340   grasses:2148
##  large     :1296   black    :1872    numerous : 400   leaves : 832
##  none      :  36   chocolate:1632    scattered:1248   meadows: 292
##  pendant   :3968   green    :  72    several  :4040   paths  :1144
##                    buff     :  48    solitary :1712   urban  : 368
```

```
##                   (Other)   : 144                    waste   : 192
```

Number Of Factor Levels In Each Of The 23 Attributes

```
# Print Number of Factor Levels (possible unique values) for Each Covariate
apply(mush.desc, 2, function(x) length(unique(x)))
```

```
##                     class                 cap.shape               cap.surface
##                         2                         6                         4
##                cap.color                   bruises                     odor
##                        10                         2                         9
##          gill.attachment             gill.spacing                 gill.size
##                         2                         2                         2
##                gill.color              stalk.shape               stalk.root
##                        12                         2                         5
## stalk.surface.above.ring stalk.surface.below.ring   stalk.color.above.ring
##                         4                         4                         9
##   stalk.color.below.ring                veil.type               veil.color
##                         9                         1                         4
##              ring.number                ring.type        spore.print.color
##                         3                         5                         9
##               population                   habitat
##                         6                         7
```

From the output, we can see that the attribute veil.type has only one factor level. Because it's value is uniform (singular) for all observations (not binary or multi-level), veil.type is not a valuable predictor. It is unchanging and therefore irrelevant to the class (Y covariate) prediction. This attribute can be dropped from the dataset.

Nullify/Delete veil.type Attribute (on "mush" dataset, our dataset for analysis)

```
# Nullify/delete veil.type (on original dataset mush) Variable Since It Is Uniform (1 factor level) For
mush$veil.type <- NULL
```

Display Number Of Variables And Observations

```
# Print Number of Variables/Observations
ncol(mush)        # 22 variables (23 before veil.type was removed)
```

```
## [1] 22
```

```
nrow(mush)        # 8124 observations
```

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

There were originally 23 attributes (covariates). After removing veil.type, there are 22 existing attributes in the "mush" dataframe. There are 8124 observations.

Quanitify And Display Class (Y, Response: Poisonous, Edible) Distribution For Each Covariate Factor Level

```r
# DIY variable/factor-level importance measurement: allows us to see # edible, or # poisonous per each
mush.table <- lapply(seq(from=2, to=ncol(mush.desc)),
                     function(x) {table(mush.desc$class, mush.desc[,x])})
names(mush.table) <- colnames(mush.desc)[2:ncol(mush.desc)]

for(i in 1:length(mush.table)) {
    print("_________________________")
    print(names(mush.table)[i])
    print(mush.table[[i]])
}
```

```
## [1] "_________________________"
## [1] "cap.shape"
##
##             bell conical flat knobbed sunken convex
##   edible     404       0 1596     228     32   1948
##   poisonous   48       4 1556     600      0   1708
## [1] "_________________________"
## [1] "cap.surface"
##
##             fibrous grooves smooth scaly
##   edible       1560       0   1144  1504
##   poisonous     760       4   1412  1740
## [1] "_________________________"
## [1] "cap.color"
##
##             buff cinnamon  red gray brown pink green purple white yellow
##   edible      48       32  624 1032  1264   56    16     16   720    400
##   poisonous  120       12  876  808  1020   88     0      0   320    672
## [1] "_________________________"
## [1] "bruises"
##
##             none bruises
##   edible    1456    2752
##   poisonous 3292     624
## [1] "_________________________"
## [1] "odor"
##
##             almond creosote foul anise musty none pungent spicy fishy
##   edible       400        0    0   400     0 3408       0     0     0
##   poisonous      0      192 2160     0    36  120     256   576   576
## [1] "_________________________"
## [1] "gill.attachment"
##
##             attached free
##   edible         192 4016
##   poisonous       18 3898
## [1] "_________________________"
## [1] "gill.spacing"
##
##             close crowded
##   edible     3008    1200
##   poisonous  3804     112
## [1] "_________________________"
```

```
## [1] "gill.size"
##
##             broad narrow
##   edible     3920    288
##   poisonous  1692   2224
## [1] "________________________"
## [1] "gill.color"
##
##             buff  red gray chocolate black brown orange pink green purple
##   edible       0   96  248       204   344   936     64  852     0    444
##   poisonous 1728    0  504       528    64   112      0  640    24     48
##
##             white yellow
##   edible      956     64
##   poisonous   246     22
## [1] "________________________"
## [1] "stalk.shape"
##
##             enlarging tapering
##   edible         1616     2592
##   poisonous      1900     2016
## [1] "________________________"
## [1] "stalk.root"
##
##             undocumented bulbous club equal rooted
##   edible             720    1920  512   864    192
##   poisonous         1760    1856   44   256      0
## [1] "________________________"
## [1] "stalk.surface.above.ring"
##
##             fibrous silky smooth scaly
##   edible        408   144   3640    16
##   poisonous     144  2228   1536     8
## [1] "________________________"
## [1] "stalk.surface.below.ring"
##
##             fibrous silky smooth scaly
##   edible        456   144   3400   208
##   poisonous     144  2160   1536    76
## [1] "________________________"
## [1] "stalk.color.above.ring"
##
##             buff cinnamon  red gray brown orange pink white yellow
##   edible       0        0   96  576    16    192  576  2752      0
##   poisonous  432       36    0    0   432      0 1296  1712      8
## [1] "________________________"
## [1] "stalk.color.below.ring"
##
##             buff cinnamon  red gray brown orange pink white yellow
##   edible       0        0   96  576    64    192  576  2704      0
##   poisonous  432       36    0    0   448      0 1296  1680     24
## [1] "________________________"
## [1] "veil.type"
##
```

```
##             partial
##   edible       4208
##   poisonous    3916
## [1] "________________________"
## [1] "veil.color"
##
##             brown orange white yellow
##   edible       96     96  4016      0
##   poisonous     0      0  3908      8
## [1] "________________________"
## [1] "ring.number"
##
##             none  one  two
##   edible       0 3680  528
##   poisonous   36 3808   72
## [1] "________________________"
## [1] "ring.type"
##
##             evanescent flaring large none pendant
##   edible          1008      48     0    0    3152
##   poisonous       1768       0  1296   36     816
## [1] "________________________"
## [1] "spore.print.color"
##
##             buff chocolate black brown orange green purple white yellow
##   edible      48        48  1648  1744     48     0     48   576     48
##   poisonous    0      1584   224   224      0    72      0  1812      0
## [1] "________________________"
## [1] "population"
##
##             abundant clustered numerous scattered several solitary
##   edible         384       288      400       880    1192     1064
##   poisonous        0        52        0       368    2848      648
## [1] "________________________"
## [1] "habitat"
##
##             woods grasses leaves meadows paths urban waste
##   edible     1880    1408    240     256   136    96   192
##   poisonous  1268     740    592      36  1008   272     0
```

Fairly Accurate Rules (Correlations) Discovered:

1. cap.shape

- If cap.shape == knobbed, then mushrooms are more likely to be poisonous.
- If cap.shape == conical, then mushrooms are all poisonous. This rule is not very accurate, as there is a small sample of mushrooms with conical cap.shape in the dataset.
- If cap.shape == sunken, then mushrooms are all edible. This rule is not very accurate, as there is a small sample of mushrooms with sunken cap.shape in the dataset.
- If cap.shape == bell, then mushrooms are significantly more likely to be edible. A mushroom hunter should proceed with caution; however, the odds are in their favor.

2. cap.surface

- If cap.surface == fibrous, then mushrooms are more likely to be edible.
- If cap.surface == grooves, then mushrooms are all poisonous. This rule is not very accurate, as there is

a small sample of mushrooms with grooved cap.surface in the dataset.

3. cap.color

- If cap.color == buff, then mushrooms are slightly more likely to be poisonous.
- If cap.color == green, then mushrooms are all poisonous. This rule is not very accurate, as there is a small sample of mushrooms with green cap.color in the dataset.

- If cap.color == white, then mushrooms are slightly more likely to be edible.
- If cap.color == red and class == poisonous, then mushroom could potentially be an Amanita Muscaria mushroom (a poisonous psychoactive species in the Agaricus family). The existence of this species of mushroom is rumored to have inspired Santa Claus folklore.

4. bruises

- If bruises == bruises, then mushrooms are significantly more likely to be edible.
- If brusies == none, then mushrooms are more likely to be poisonous.

5. odor: A strong predictor of class; the odors correlated with poison are very logical. This draws an interesting perspective on smell and our evolutionary sensory decision-making to avoid things with "bad" or "unfavorable" smells. All "unfavorable" odors are completely correlated with the poisonous class.

- If odor == almond, then all mushrooms documented are edible.
- If odor == creosote, then all mushrooms documented are poisonous.
- If odor == foul, then all mushrooms documented are poisonous.
- If odor == anise, then all mushrooms documented are edible.
- If odor == musty, then all mushrooms documented are poisonous.
- If odor == none, then mushrooms are much more likely to be edible.
- If odor == pungent, then all mushrooms documented are poisonous.
- If odor == spicy, then all mushrooms documented are poisonous.
- If odor == fishy, then all mushrooms documented are poisonous.

6. gill.attachment

- If gill.attachment == attached, then mushrooms are very slightly more likely to be edible (however, there are not enough observations to accurately claim) [class imbalance]

7. gill.spacing

- If gill.spacing == crowded, then mushrooms are significantly more likely to be edible.

8. gill.size

- If gill.size == broad, then mushrooms are more likely to be edible. Mushrooms collectors must still proceed with caution.
- If gill.size == narrow, then mushrooms are significantly more likely to be poisonous.

9. gill.color

- If gill.color == buff, then all mushrooms documented are poisonous.
- If gill.color == red, then all mushrooms documented are edible.
- If gill.color == gray || gill.color == chocolate, then mushrooms are slightly more likely to be poisonous.
- If gill.color == black, then mushrooms are slightly more likely to be edible.
- If gill.color == brown, then mushrooms are more likely to be edible.
- If gill.color == orange, then all mushrooms documented are edible.
- If gill.color == green, then all mushrooms documented are poisonous
- If gill.color == purple, then mushrooms are more likely to be edible.
- If gill.color == white, then mushrooms are more likely to be edible.

10. stalk.shape

- No significant correlations here. This predictor is not of large importance.

11. stalk.root

- If stalk.root == club, then mushrooms are more likely to be edible.
- If stalk.root == equal, then mushrooms are more likely to be edible.
- If stalk.root == rooted, then all mushrooms documented are edible.
- If stalk.root == undocumented, then mushrooms are more likely to be poisonous.

12. stalk.surface.above.ring

- If stalk.surface.above.ring == silky, then mushrooms are significantly more likely to be poisonous.
- If stalk.surface.above.ring == smooth, then mushrooms are more likely to be edible.

13. stalk.surface.below.ring

- If stalk.surface.below.ring == silky, then mushrooms are significantly more likely to be poisonous.
- If stalk.surface.below.ring == smooth, then mushrooms are more likely to be edible.

14. stalk.color.above.ring

- If stalk.color.above.ring == buff, then all mushrooms documented are poisonous.
- If stalk.color.above.ring == cinnamon, then all mushrooms documented are poisonous.
- If stalk.color.above.ring == red, then all mushrooms documented are edible.
- If stalk.color.above.ring == gray, then all mushrooms documented are edible.
- If stalk.color.above.ring == brown, then mushrooms more likely to be poisonous.
- If stalk.color.above.ring == orange, then all mushrooms documented are edible.
- If stalk.color.above.ring == pink, then mushrooms are more likely to be poisonous.
- If stalk.color.above.ring == white, then mushrooms are more likely to be edible. Proceed with caution.
- If stalk.color.above.ring == yellow, then all mushrooms documented are poisonous.

15. stalk.color.below.ring

- If stalk.color.below.ring == buff, then all mushrooms documented are poisonous.
- If stalk.color.below.ring == cinnamon, then all mushrooms documented are poisonous.
- If stalk.color.below.ring == red, then all mushrooms documented are edible.
- If stalk.color.below.ring == gray, then all mushrooms documented are edible.
- If stalk.color.below.ring == brown, then mushrooms more likely to be poisonous.
- If stalk.color.below.ring == orange, then all mushrooms documented are edible.
- If stalk.color.below.ring == pink, then mushrooms are more likely to be poisonous.
- If stalk.color.below.ring == white, then mushrooms are slightly more likely to be edible. Proceed with caution.
- If stalk.color.below.ring == yellow, then all mushrooms documented are poisonous.

16. veil.color

- If veil.color == brown, then all mushrooms documented are edible.
- If veil.color == orange, then all mushrooms documented are edible.
- If veil.color == yellow, then all mushrooms documented are poisonous.

17. ring.number

- If ring.number == none, then all mushrooms documented are poisonous.
- If ring.number == two, then mushrooms are more likely to be edible.

18. ring.type

- If ring.type == evanescent, then mushrooms are slightly more likely to be poisonous.
- If ring.type == flaring, then all mushrooms documented are edible.
- If ring.type == large, then all mushrooms documented are poisonous.
- If ring.type == none, then all mushrooms documented are poisonous.

- If ring.type == pendant, then mushrooms are more likely to be edible.

19. spore.print.color: A good predictor.

- If spore.print.color == buff, then all mushrooms documented are edible.
- If spore.print.color == chocolate, then mushrooms are significantly more likely to be poisonous.
- If spore.print.color == black, then mushrooms are significantly more likely to be edible.
- If spore.print.color == brown, then mushrooms are significantly more likely to be edible.
- If spore.print.color == orange, then all mushrooms documented are edible.
- If spore.print.color == green, then all mushrooms documented are poisonous.
- If spore.print.color == purple, then all mushrooms documented are edible.
- If spore.print.color == white, then mushrooms are much more likely to be poisonous.
- If spore.print.color == yellow, then all mushrooms documented are edible.

20. population

- If population == abundant, then all mushrooms documented are edible.
- If population == clustered, then mushrooms are more likely to be edible.
- If population == numerous, then all mushrooms documented are edible.
- If population == scattered, then mushrooms are slightly more likely to be edible.
- If population == several, then mushrooms are more likely to be poisonous.
- If population == solitary, then mushrooms are more likely to be edible.

21. habitat

- If habitat == grasses, then mushrooms are more likely to be edible.
- If habitat == meadows, then mushrooms are more likely to be edible.
- If habitat == paths (trampled through nature), then mushrooms are significantly more likely to be poisonous.
- If habitat == urban, then mushrooms are more likely to be poisonous.
- If habitat == waste (feces), then all mushrooms documented are edible.

Classification (Supervised Learning)

Random Forest Classification

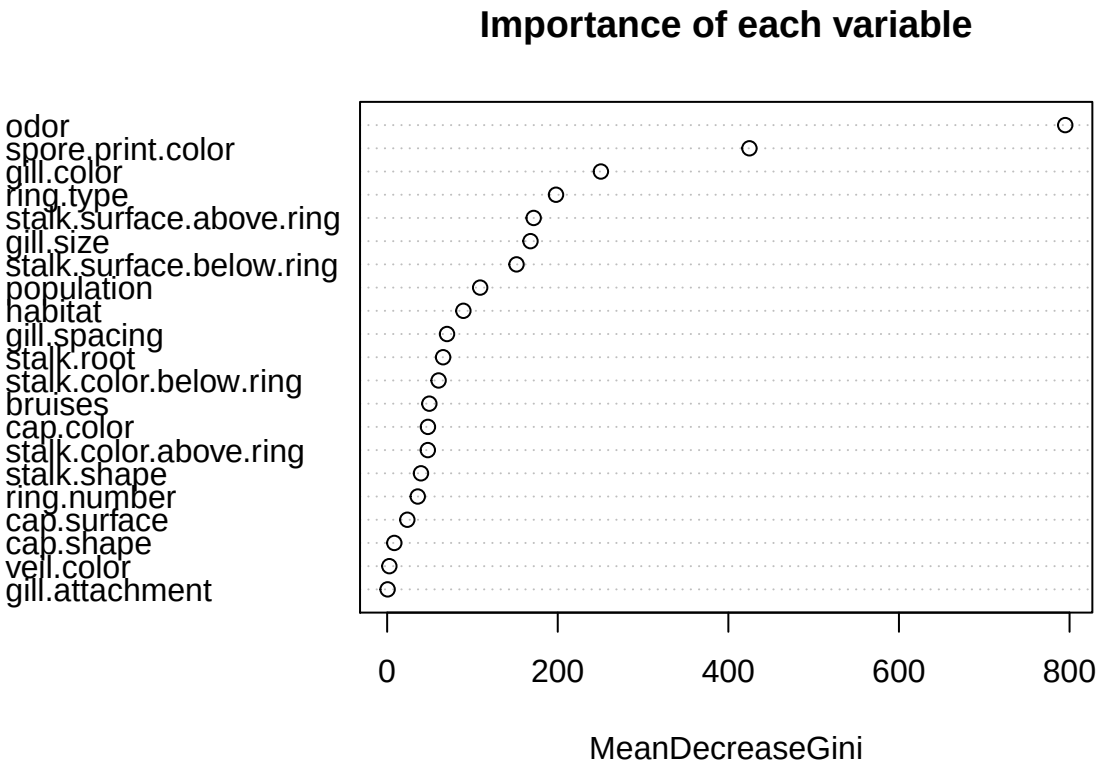
Details: Random Forest Classification is an ensemble learnig method, meaning it utilizes multiple learning algorithms to obtain more accurate predictons than those acquired by using only one algorithm. This method creates many decision trees (a "forest") using the specified training set. The class label outputted is decided by the mode (most common) of the classes or the mean prediction of of the individual trees.

Reasoning: I chose to use Random Forest rather than decision trees classification because Random Forest Classification protects against overfitting to the training set. In the mushroom dataset, there are many variables and most analysis methods utilizing all covariates will be prone to overfitting, in which weighting of covariates is too spread out in order to accurately determine covariate signficance. Random Forest is significantly more stable and robust as a method of analysis. Its Variable Importance Plot is also desirable and useful in the exploration of a categorical dataset.

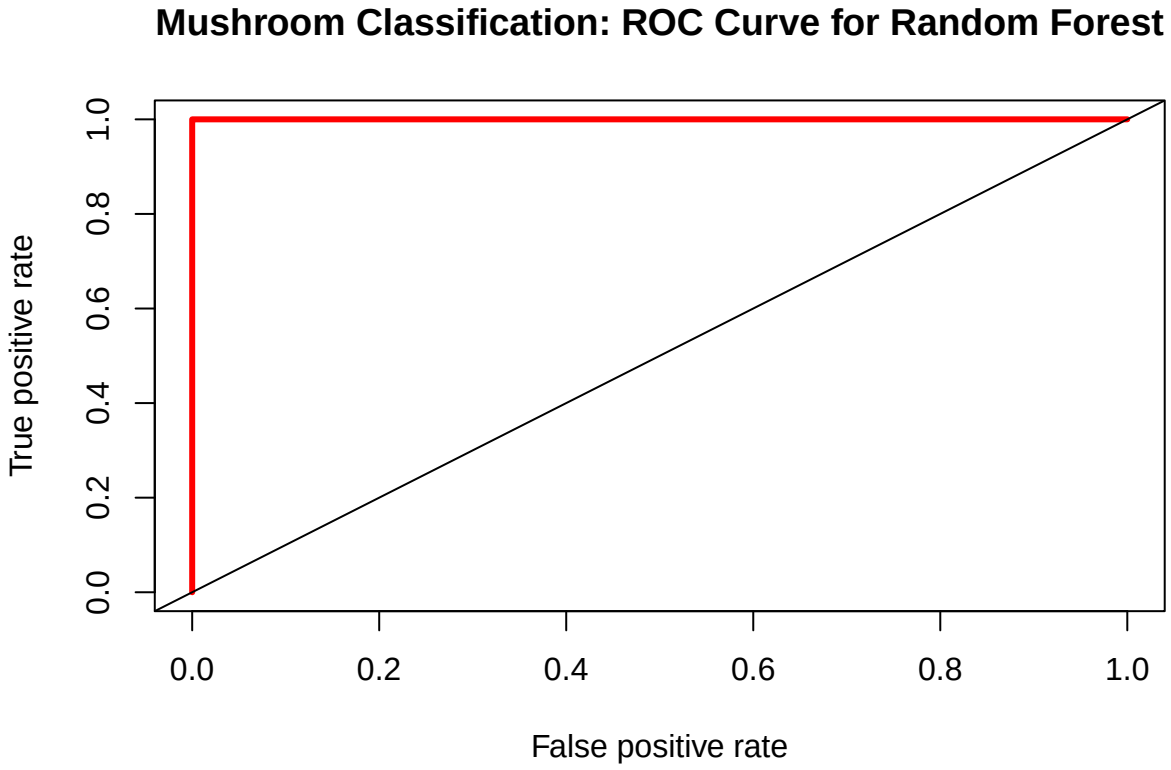
Manual Random Forest / ROC Curve

```
### Manual Random Forest for ROC Curve
set.seed(1)
data <- sample(2, nrow(mush), replace = T, prob = c(0.7, 0.3))  # split data into test set and training
traindata <- mush[data == 1,]
```

```r
testdata <- mush[data == 2,]

# Tree functions
varsTree <- class ~ cap.shape + cap.surface + cap.color + bruises + odor + gill.attachment + gill.spaci

# Applying the algorithm
treeRF <- randomForest(varsTree, data = traindata, ntree=100, proximity = T)

# Importance of each variable
mush.imp <- varImpPlot(treeRF, main = "Importance of each variable")
```




```r
# Class Prediction Object / ROC Curve
pred.rf = predict(treeRF, testdata, type="prob")
pred.rf = prediction(pred.rf[,2], testdata$class)
perf.rf = performance(pred.rf, measure="tpr", x.measure="fpr")
plot(perf.rf, col=2, lwd=3, main="Mushroom Classification: ROC Curve for Random Forest")
abline(0,1)
```



Analysis:

- The shape of the ROC curve illustrates a perfect class prediction with 0% error and 100% accuracy.
- The Variable Importance Plot produced by the manual Random Forest method specifies the top five most important variables to be odor, spore.print.color, gill.color, ring.type, and stalk.surface.above.ring.

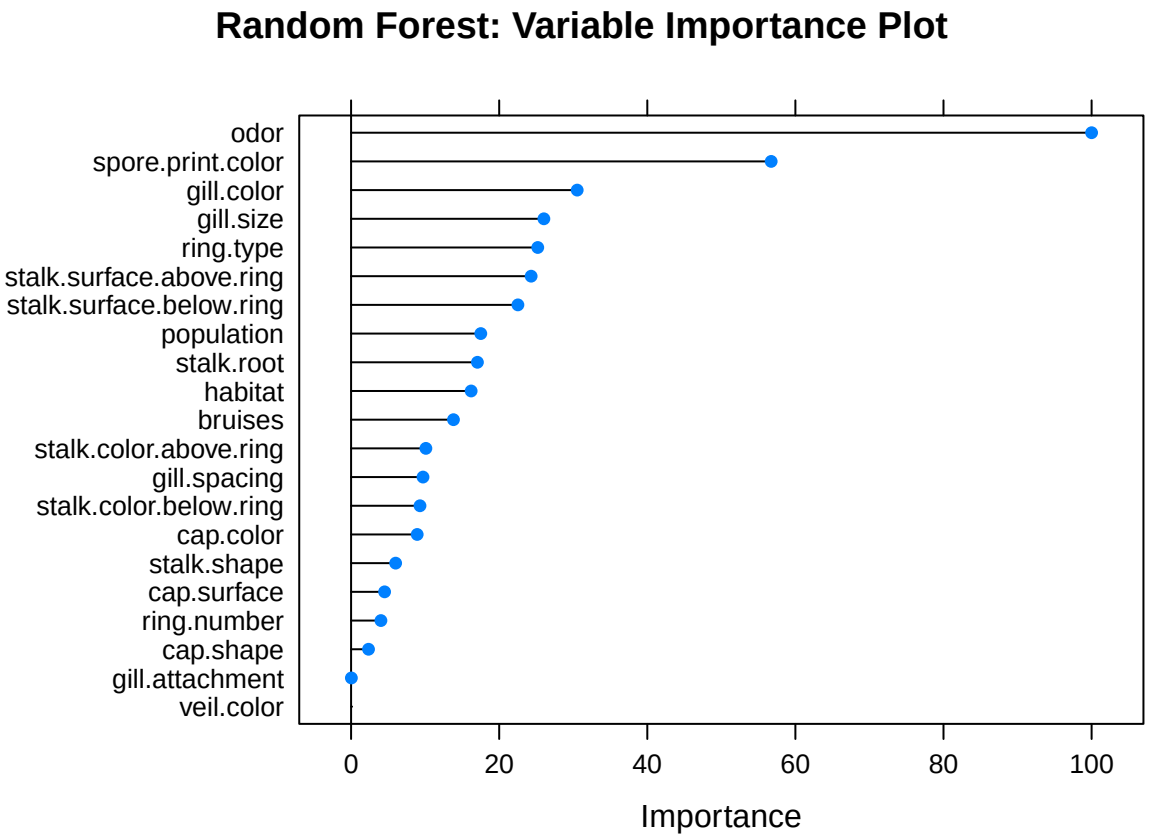
Alternate Random Forest Method Using Caret Package: Yields Preferred Variable Importance Plot

```
### More efficient/aesthetically-pleasing Random Forest Classification with cleaner Variable Importance

# Split data into test/training sets
set.seed(101)
diviz = sample.split(mush$class, SplitRatio = .7)  # 0.7 is the stratified split ratio of train to test
table(diviz)
```

```
## diviz
## FALSE  TRUE
##  2437  5687
```

```
mush.Xtrain = subset(mush, diviz == TRUE)       # create training subset of predictors (X)
mush.Xtest = subset(mush, diviz == FALSE)       # create testing  subset of predictors (X)

mush.Ytrain <- mush.Xtrain$class                #  create training Y vector
mush.Ytest <- mush.Xtest$class                  #  create testing  Y vector

mush.Xtrain$class <- NULL                       # nullify class vector so it can be re-predicted
mush.Xtest$class <- NULL
```

```r
# Create a stratified sample for 10 fold repeated cross validation
CV.10fold <- createMultiFolds(mush.Ytrain, k=10, times=2)       # 2 repetitions

# Create a control object for repeated CV in caret
mush.control <- trainControl(method="repeatedcv", number=10, repeats=2, index=CV.10fold)

# Model 1: Random Forest Model
RFmush.cv <- train(x=mush.Xtrain, y=mush.Ytrain, method="rf", trControl=mush.control, tuneLength=3)
# tuneLength = granularity/scale parameter

# Clean Variable Importance Plot
plot(varImp(RFmush.cv), main = "Random Forest: Variable Importance Plot")
```




```r
# Class Prediction Object On Test Set
mush.YpredictRF <- predict(RFmush.cv, mush.Xtest)

# Accuracy Evaluation Dataframe
acc.eval.rf <- data.frame(Orig = mush.Ytest, Pred = mush.YpredictRF)  # accuracy evaluation dataframe

# Confusion Matrix (error rate = 0%, 100% accuracy)
confusionMatrix(table(acc.eval.rf$Orig, acc.eval.rf$Pred))
```

```
## Confusion Matrix and Statistics
##
##
##         e    p
##   e 1262    0
```

```
##   p    0 1175
##
##                Accuracy : 1
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.5178
##     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 : 0.5178
##          Detection Rate : 0.5178
##    Detection Prevalence : 0.5178
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : e
##
```

```
# Random Forest: Variable Importance Plot confirms DIY variable/factor-level importance measurement
```

Analysis:

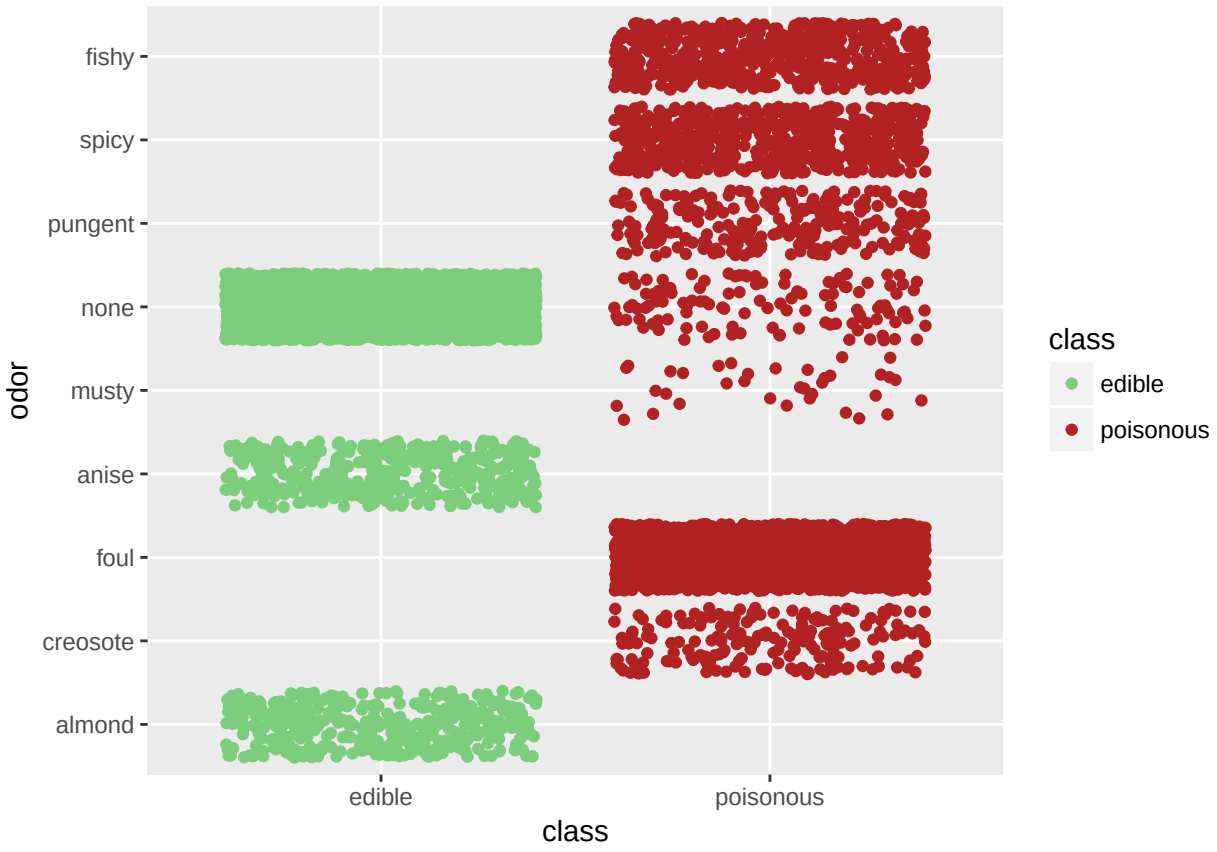
- The confusion matrix for Random Forest Classification method using the caret package materials shows a 100% accuracy in classification and a 0% error rate. Let's investigate the cause of this perfect classification rate using EDA visualization.
- Random Forest caret package shortcut method yields a Variable Importance Plot denoting the top five most important variables to be odor, spore.print.color, gill.color, gill.size, and ring.type. The plot is slightly different than the Variable Importance Plot created with the manual Random Forest method. This could be due a slight discontinuity between the two Random Forest algorithms.

Class Prediction Results (Random Forest Method)

Exploratory Data Analysis Part 2: Thoughtful Visualizations (Inspired By Random Forest Classification)

Plot Of odor vs. class

```
# Plot of odor against class using a ggplot geom_jitter shortcut
odor.plot <- ggplot(mush.desc, aes(class, odor))
odor.plot + geom_jitter(aes(color = class)) + scale_color_manual(values=c("palegreen3", "firebrick"))
```



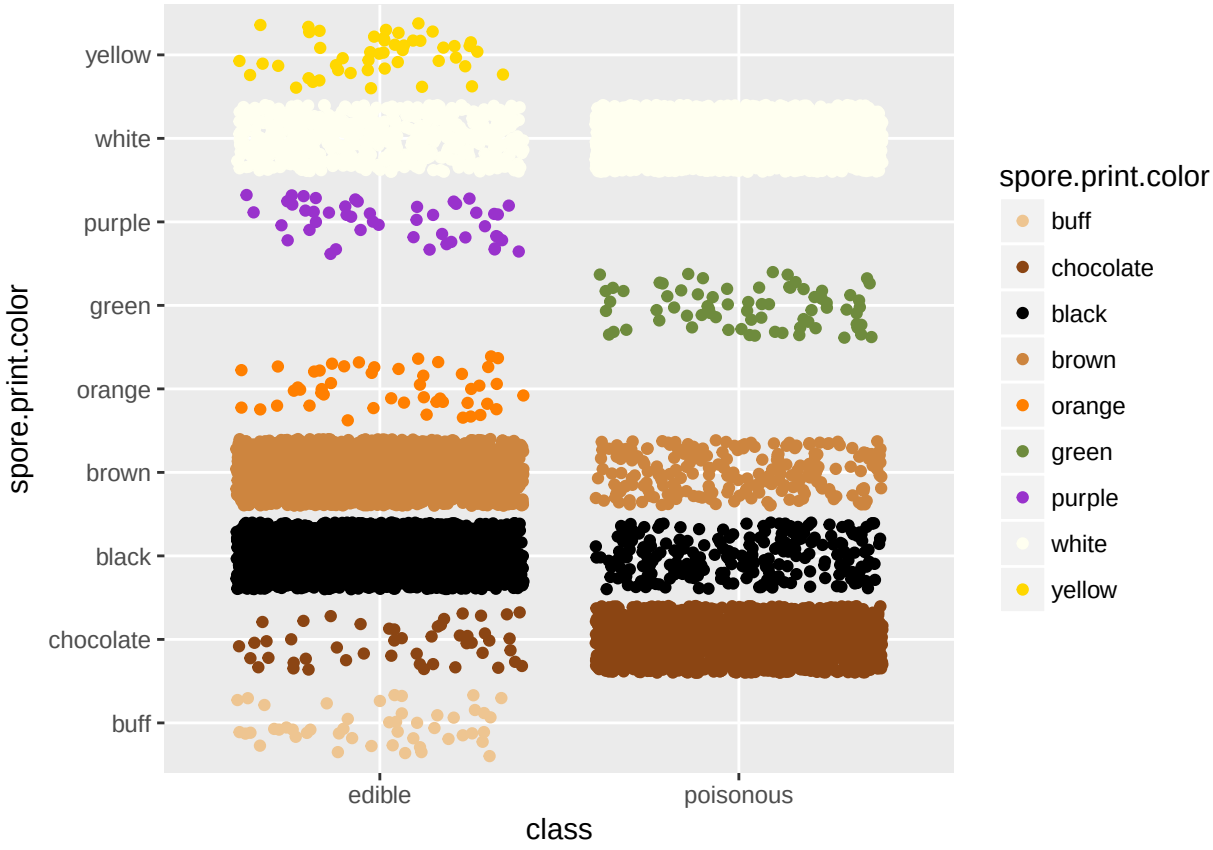
Analysis:

- It can be seen from the above visualization that the attribute odor nearly predicts all mushrooms classes except for those which are odorless. This single variable is the culprit behind the high accuracy in classification analysis.
- Mushrooms with odors “almond” and “anise” denote only edible mushrooms. According to the data, there is no risk associated with eating these mushrooms, if their smells can be properly identified.
- All unpleasant odors (“fishy”, “spicy”, “pungent”, “musty”, “foul”, “creosote”) are correlated only with poisonous mushrooms. According to the data, the risk in eating these mushrooms is 100%.
- Scent is a perfect predictor for classifying mushrooms as edible vs. poisonous, all except for odorless mushrooms, which are largely edible, but still in part poisonous (considerable risk). This makes an interesting statement about human cognition and sensory perception (sense of smell/taste particularly) being virtually a perfect predictor for classifying what to eat in nature. It might also be interesting to consider the evolutionary possiblity of humans having developed to perceive qualities of nature/food perfectly; even suggesting maybe a correlated evolution between humans and mushrooms. This essentially suggests that humans don’t need to be studying data on mushrooms at all; if one were to perform a proper sensory inspection of a fruiting body before putting it in our mouth or more importantly swallowing it, they would probably be okay. But the data is a beautiful validation of nature and human being’s instinctual knowledge.
- Idealogical support for the concept of correlated evolution between mushrooms and other living creatures: mushrooms only produce their fruiting body to spread spores and reproduce. The mushroom is not the central component of the organism. It must be costly in energy for a mushroom to synthesize poison; why would a mushroom spend extra energy creating a poison to protect itself from being eaten? A plant may produce a poison to protect its own livelihood; if it is eaten, it dies. However, the process of mushrooms being eaten actually spreads spores and allows the organism to grow. The organism which lies beneath the ground is undisturbed when the fruiting body is eaten. This trend leaves us with no reasoning for the evolution of poison in mushrooms, unless they evolved to synthesize poison as a direct

means of protection of an environment or combatting of an invasive species. This scenario would denote a correlated evolution.

Plot Of spore.print.color vs. class

```
# Plot of spore.print.color against class using a ggplot geom_jitter shortcut
spore.print.color.plot <- ggplot(mush.desc, aes(class, spore.print.color))
spore.print.color.plot + geom_jitter(aes(color = spore.print.color)) + scale_color_manual(values=c("bur
```

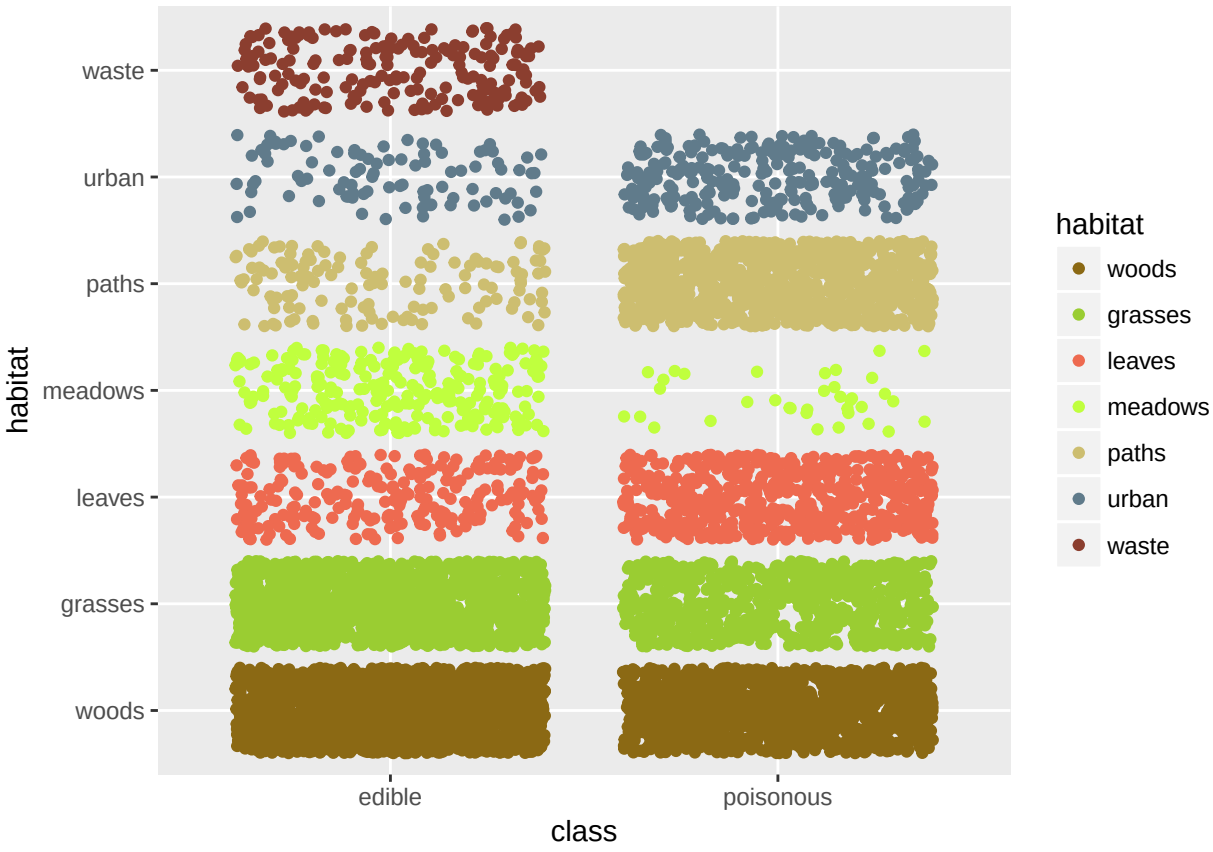


Analysis:

- According to the data, mushrooms with "buff", "orange", "purple", or "yellow" spore.print.color correspond only to edible mushrooms.
- Mushrooms with green spore.print.color correspond only to being poisonous.

Plot Of habitat vs. class

```
# Plot of habitat against class using a ggplot geom_jitter shortcut
habitat.plot <- ggplot(mush.desc, aes(class, habitat))
habitat.plot + geom_jitter(aes(color = habitat)) + scale_color_manual(values=c("goldenrod4","yellowgree
```



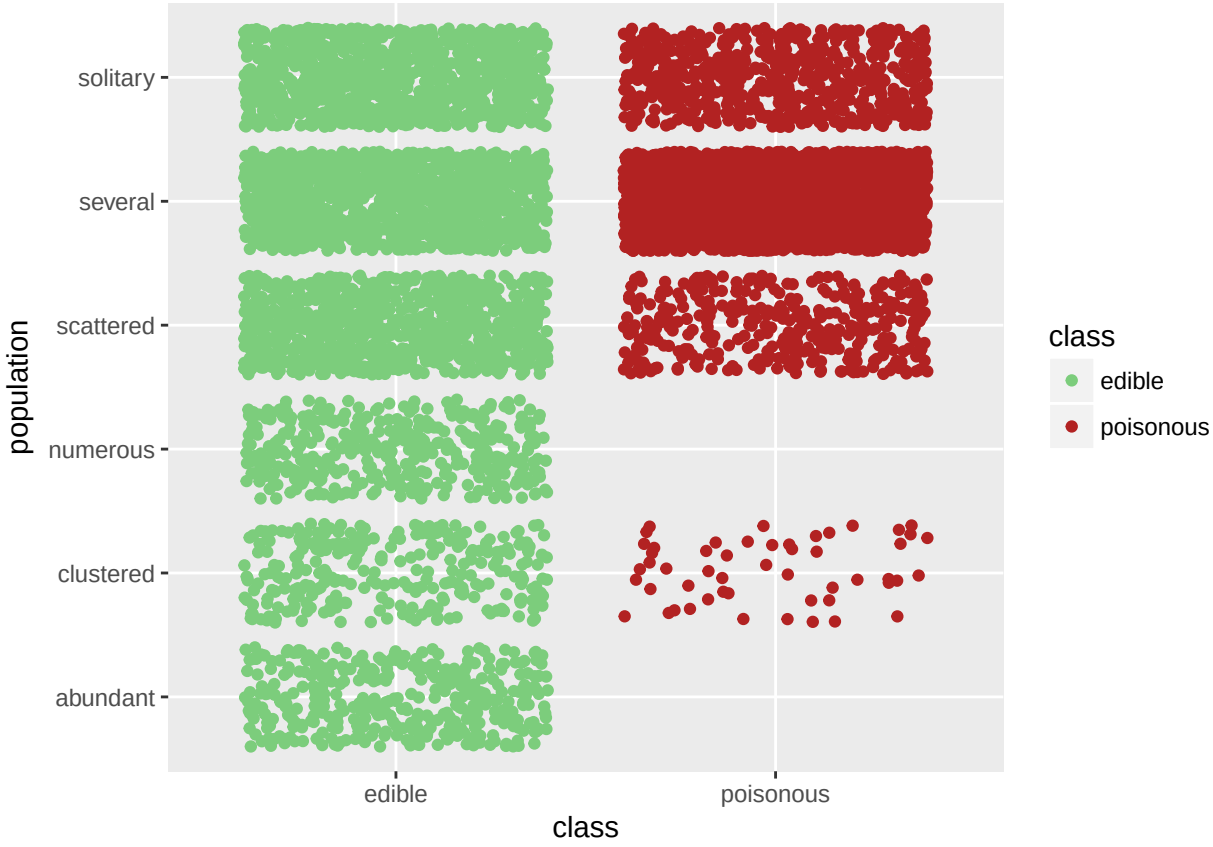
Analysis:

- Mushrooms on paths and in urban places possess a strong correlation to being poisonous. This is interesting because paths and urban places are both places where human beings have intervened in nature. Could it be possible that the homeostatic tendencies of nature influence the production of poisonous mushrooms in specific areas as a form of protection, retaliation, or even rehabilitation of those environments. If so, the Earth is far more symbiotically intelligent that humans can even comprehend.
- This concept is supported by a colleague's research into psychoactive mushrooms. His research presented findings showing that mushrooms containing the psychoactive chemical psilocybin are more plentiful in areas where nature has been disturbed. I have not yet acquired the permission, but I will eventually reference his research in this study. This phenomenological correlation of psychoactive/poisonous mushrooms to disturbed natural environments could potentially be explained by a tendency of the Earth to control homeostasis, shift and manipulate environmental evolution, and rehabilitate its dying environments. In the case of psychoactive mushrooms, which produce mental experiences of oneness with and appreciation of nature, the Earth is producing purposeful objects which shift consciousness towards environmentalism and protection of the Earth and away from exploitation. This could also be considered a homeostatic rehabilitation.

- Another unique correlation is that mushrooms that grow on waste (feces) are classified as always edible. This denotes a fully self-contained and self-sustainable bioligical cycle, a feedback loop. There is no waste; everything is used. How is it that mushrooms know where they need to grow in order to coerce the decomposition of decaying bio-matter? This phenomenon is also a sign of a correlated evolution between humans and mushrooms.

Plot Of population vs. class

```
# Plot of population against class using a ggplot geom_jitter shortcut
population.plot <- ggplot(mush.desc, aes(class, population))
population.plot + geom_jitter(aes(color = class)) + scale_color_manual(values=c("palegreen3", "firebric
```



Analysis:

- Singular or more sparse populatons of mushrooms seem to possess a much higher correlation to being poisonous (although there are still some edible), whereas larger populations of mushrooms possess a high correlation to edibility. Observations labeled as having abundant and numerous populations are classified as only edible. Evolutionarily, this trend could relate to edible mushrooms evolving to reproduce i larger clusters as a food source.

Table: odor (Nested Under class) vs. spore.print.color

```
# Plot table of odor (nested under class) against spore.print.color
odor.class.spore.table <- table(mush.desc$class, mush.desc$odor, mush.desc$spore.print.color)
ftable(odor.class.spore.table)
```

```
##                    buff chocolate black brown orange green purple white yellow
##
## edible    almond      0         0   176   200      0     0     24     0      0
##           creosote    0         0     0     0      0     0      0     0      0
##           foul        0         0     0     0      0     0      0     0      0
##           anise       0         0   176   200      0     0     24     0      0
##           musty       0         0     0     0      0     0      0     0      0
##           none       48        48  1296  1344     48     0      0   576     48
##           pungent     0         0     0     0      0     0      0     0      0
```

```
##           spicy        0         0     0     0      0     0       0     0       0
##           fishy        0         0     0     0      0     0       0     0       0
## poisonous almond       0         0     0     0      0     0       0     0       0
##           creosote     0         0    96    96      0     0       0     0       0
##           foul         0      1584     0     0      0     0       0   576       0
##           anise        0         0     0     0      0     0       0     0       0
##           musty        0         0     0     0      0     0       0    36       0
##           none         0         0     0     0      0    72       0    48       0
##           pungent      0         0   128   128      0     0       0     0       0
##           spicy        0         0     0     0      0     0       0   576       0
##           fishy        0         0     0     0      0     0       0   576       0
```

Analysis:

- Looking for values symmetrical across the center horizontal line of the table, we can see the only symmetrical (or same-scented-colored) value which lies under both edible and poisonous class levels is odorless white spore print mushrooms. With retaining only the two variables odor and spore.print.color, our model is virtually perfect at predicting class, except for those observations which are categorized as odorless and white spore print.
- In looking for the best model (simplest [least predictors] while still most explanatory), maybe it is possible to find a third covariate which in combination with odor and spore.print.color can rule out ambiguities for white spore print odorless mushrooms.

Logistic Regression With Categorical Attributes Classification

Details: Logistic Regression is a regression model in which the response (dependent) variable is categorical, often binary as in the case of The Mushroom Dataset. The glm function automatically takes non-binary categorical multiple-factor-level variables and converts them into binary dummy variables in order to perform a logistic regression to predict the binary class level: i.e. veil.color –> brown(0,1), orange(0, 1), white(0, 1), yellow(0, 1).

Reasoning: I chose to use Logistic Regression because it works decently well with categorical attributes. It is also able to accurately determine covariate significance in models with only a few covariates. This allows us to use a model reduction process that we were unable to use in Random Forest Classification. The model reduction can allow us to iteratively simplify our model until we have a high accuracy classification with the minimum amount of covariates included.

Full Model: Containing All Covariates

```
### Manual Logistic Regression

# Full Model: Containing All Covariates
mush.glmz = glm(class ~ cap.shape + cap.surface + cap.color + bruises + odor + gill.attachment + gill.sp
```

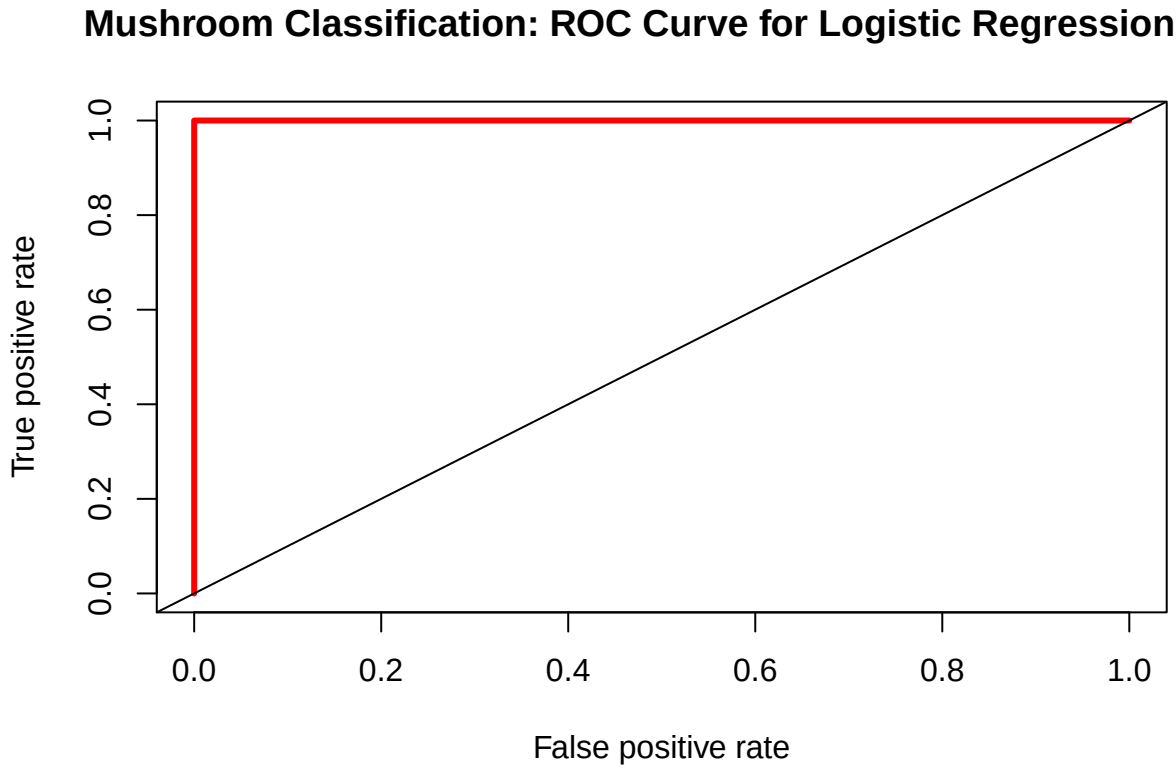
```
## Warning: glm.fit: algorithm did not converge
```

```
# Class Prediction Object / ROC Curve Code
pred.glm = predict(mush.glmz, mush, type="response")    # receive output of predicted probabilities
```

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

```
pred.glm = prediction(pred.glm, mush$class)   # comparing predicted probabilities to true class labels
perf.glm = performance(pred.glm, measure="tpr", x.measure="fpr")
```

```r
plot(perf.glm, col=2, lwd=3, main="Mushroom Classification: ROC Curve for Logistic Regression FM")
abline(a = 0, b = 1)
```




```r
# Statistical Summary
summary(mush.glmz)
```

```
##
## Call:
## glm(formula = class ~ cap.shape + cap.surface + cap.color + bruises +
##     odor + gill.attachment + gill.spacing + gill.size + gill.color +
##     stalk.shape + stalk.root + stalk.surface.above.ring + stalk.surface.below.ring +
##     stalk.color.above.ring + stalk.color.below.ring + veil.color +
##     ring.number + ring.type + spore.print.color + population +
##     habitat, family = "binomial", data = mush)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -2.409e-06  -2.409e-06  -2.409e-06   2.409e-06   2.409e-06
##
## Coefficients: (10 not defined because of singularities)
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -2.657e+01  3.088e+05       0        1
## cap.shapec                  8.901e-07  2.062e+05       0        1
## cap.shapef                 -3.438e-08  2.344e+04       0        1
## cap.shapek                  1.121e-07  2.548e+04       0        1
## cap.shapes                 -3.089e-08  8.032e+04       0        1
## cap.shapex                 -2.734e-08  2.245e+04       0        1
## cap.surfaceg               -1.067e-06  2.521e+05       0        1
## cap.surfaces               -6.067e-11  1.342e+04       0        1
## cap.surfacey               -1.153e-10  1.124e+04       0        1
```

```
## cap.colorc                -4.791e-10  7.787e+04       0        1
## cap.colore                 6.965e-10  3.625e+04       0        1
## cap.colorg                 2.196e-09  3.473e+04       0        1
## cap.colorn                -6.376e-10  3.544e+04       0        1
## cap.colorp                -4.226e-09  4.460e+04       0        1
## cap.colorr                -5.629e-09  1.307e+05       0        1
## cap.coloru                -5.632e-09  1.307e+05       0        1
## cap.colorw                -5.629e-09  3.494e+04       0        1
## cap.colory                -1.566e-08  3.719e+04       0        1
## bruisest                   2.657e+01  1.197e+05       0        1
## odorc                      2.657e+01  1.432e+05       0        1
## odorf                     -2.657e+01  3.585e+05       0        1
## odorl                      2.655e-12  2.518e+04       0        1
## odorm                      1.328e+02  5.639e+05       0        1
## odorn                     -7.970e+01  3.458e+05       0        1
## odorp                     -5.313e+01  4.306e+05       0        1
## odors                     -2.657e+01  3.591e+05       0        1
## odory                     -2.657e+01  3.591e+05       0        1
## gill.attachmentf          -6.626e-07  1.187e+05       0        1
## gill.spacingw              9.434e-08  5.140e+04       0        1
## gill.sizen                -7.970e+01  3.241e+05       0        1
## gill.colore               -2.657e+01  2.147e+05       0        1
## gill.colorg               -2.657e+01  2.126e+05       0        1
## gill.colorh               -2.657e+01  2.122e+05       0        1
## gill.colork               -2.657e+01  2.129e+05       0        1
## gill.colorn               -2.657e+01  2.125e+05       0        1
## gill.coloro               -2.657e+01  2.208e+05       0        1
## gill.colorp               -2.657e+01  2.120e+05       0        1
## gill.colorr               -2.657e+01  2.298e+05       0        1
## gill.coloru               -2.657e+01  2.131e+05       0        1
## gill.colorw               -2.657e+01  2.116e+05       0        1
## gill.colory               -2.657e+01  2.189e+05       0        1
## stalk.shapet              -5.313e+01  2.111e+05       0        1
## stalk.rootb               -2.657e+01  1.398e+05       0        1
## stalk.rootc               -1.594e+02  6.548e+05       0        1
## stalk.roote                2.657e+01  1.287e+05       0        1
## stalk.rootr               -1.860e+02  5.638e+05       0        1
## stalk.surface.above.ringk -1.275e-09  2.667e+04       0        1
## stalk.surface.above.rings -4.774e-11  2.144e+04       0        1
## stalk.surface.above.ringy  6.287e-07  2.929e+05       0        1
## stalk.surface.below.ringk  1.673e-10  2.667e+04       0        1
## stalk.surface.below.rings -5.745e-11  2.144e+04       0        1
## stalk.surface.below.ringy  2.657e+01  1.790e+05       0        1
## stalk.color.above.ringc           NA         NA      NA       NA
## stalk.color.above.ringe    1.690e-09  5.836e+04       0        1
## stalk.color.above.ringg    1.740e-09  3.101e+04       0        1
## stalk.color.above.ringn   -1.303e-07  2.423e+04       0        1
## stalk.color.above.ringo   -5.313e+01  2.702e+05       0        1
## stalk.color.above.ringp    1.790e-09  2.423e+04       0        1
## stalk.color.above.ringw    1.690e-09  2.763e+04       0        1
## stalk.color.above.ringy    1.594e+02  5.420e+05       0        1
## stalk.color.below.ringc           NA         NA      NA       NA
## stalk.color.below.ringe    2.253e-08  5.836e+04       0        1
## stalk.color.below.ringg    2.255e-08  3.101e+04       0        1
```

```
## stalk.color.below.ringn   -2.113e-07  2.423e+04       0        1
## stalk.color.below.ringo           NA         NA      NA       NA
## stalk.color.below.ringp    2.256e-08  2.423e+04       0        1
## stalk.color.below.ringw    2.253e-08  2.763e+04       0        1
## stalk.color.below.ringy   -2.084e-07  1.282e+05       0        1
## veil.coloro                1.035e-15  5.140e+04       0        1
## veil.colorw                       NA         NA      NA       NA
## veil.colory                       NA         NA      NA       NA
## ring.numbero               1.328e+02  5.188e+05       0        1
## ring.numbert                      NA         NA      NA       NA
## ring.typef                 5.313e+01  2.596e+05       0        1
## ring.typel                        NA         NA      NA       NA
## ring.typen                        NA         NA      NA       NA
## ring.typep                 2.657e+01  1.089e+05       0        1
## spore.print.colorh                NA         NA      NA       NA
## spore.print.colork         7.923e-13  7.362e+04       0        1
## spore.print.colorn         2.019e-14  7.269e+04       0        1
## spore.print.coloro         5.628e-14  7.269e+04       0        1
## spore.print.colorr         1.328e+02  3.472e+05       0        1
## spore.print.coloru        -3.012e-13  1.028e+05       0        1
## spore.print.colorw         7.970e+01  3.330e+05       0        1
## spore.print.colory         7.515e-23  7.269e+04       0        1
## populationc               -1.547e-09  6.210e+04       0        1
## populationn               -6.548e-14  3.598e+04       0        1
## populations               -5.162e-13  2.570e+04       0        1
## populationv               -1.547e-09  3.484e+04       0        1
## populationy               -1.495e-09  3.609e+04       0        1
## habitatg                  -6.249e-09  2.139e+04       0        1
## habitatl                  -1.380e-09  1.978e+04       0        1
## habitatm                  -6.253e-09  3.641e+04       0        1
## habitatp                  -1.196e-09  1.564e+04       0        1
## habitatu                  -2.275e-08  3.728e+04       0        1
## habitatw                          NA         NA      NA       NA
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1.1252e+04  on 8123  degrees of freedom
## Residual deviance: 4.7132e-08  on 8038  degrees of freedom
## AIC: 172
##
## Number of Fisher Scoring iterations: 25
```

Perfect class prediction; prone to overfitting.

Shortcut Logistic Regression Method Using Caret Library: Yields Logistic Regression Variable Importance Plot (Full Model)

```
### Shortcut Logistic Regression Method Using Caret Library

# Full Model: Containing All Covariates
mush.glm <- train(x = mush.Xtrain, y = mush.Ytrain, method="glm", trControl = mush.control, family = "b
```

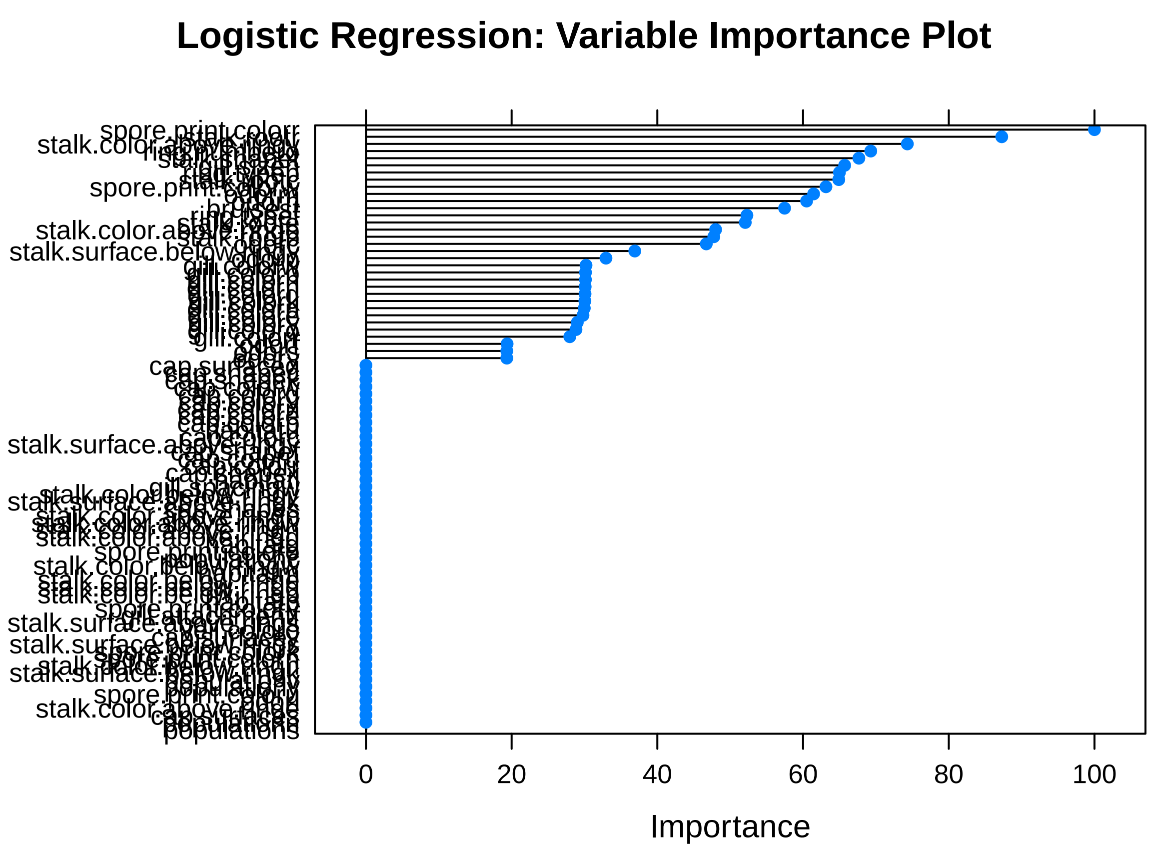
```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
```

```
## ifelse(type == : prediction from a rank-deficient fit may be misleading
## Warning: glm.fit: algorithm did not converge
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
## Warning: glm.fit: algorithm did not converge
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
## Warning: glm.fit: algorithm did not converge
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
## Warning: glm.fit: algorithm did not converge
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
## Warning: glm.fit: algorithm did not converge
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
## Warning: glm.fit: algorithm did not converge
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
## Warning: glm.fit: algorithm did not converge
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
## Warning: glm.fit: algorithm did not converge
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
## Warning: glm.fit: algorithm did not converge
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
## Warning: glm.fit: algorithm did not converge
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
## Warning: glm.fit: algorithm did not converge
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
## Warning: glm.fit: algorithm did not converge
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
## Warning: glm.fit: algorithm did not converge

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

## Warning: glm.fit: algorithm did not converge

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

## Warning: glm.fit: algorithm did not converge

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

## Warning: glm.fit: algorithm did not converge

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

## Warning: glm.fit: algorithm did not converge

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

## Warning: glm.fit: algorithm did not converge

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

## Warning: glm.fit: algorithm did not converge
```

```
plot(varImp(mush.glm), main="Logistic Regression: Variable Importance Plot")
```



```r
# Class Prediction Object
mush.YpredictGLM <- predict(mush.glm, mush.Xtest)
```

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

```r
# Accuracy Evaluation Dataframe
acc.eval.glm <- data.frame(Orig = mush.Ytest,Pred = mush.YpredictGLM)

# Confusion Matrix (error rate = 0%, 100% accuracy)
confusionMatrix(table(acc.eval.glm$Orig, acc.eval.glm$Pred))      # 100% accuracy confirmed
```

```
## Confusion Matrix and Statistics
##
##
##        e    p
##   e 1262    0
##   p    0 1175
##
##                Accuracy : 1
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.5178
##     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 : 0.5178
##          Detection Rate : 0.5178
##    Detection Prevalence : 0.5178
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : e
##
```

Reduced Model: Containing Top Five Covariates On Random Forest Variable Importance Plot

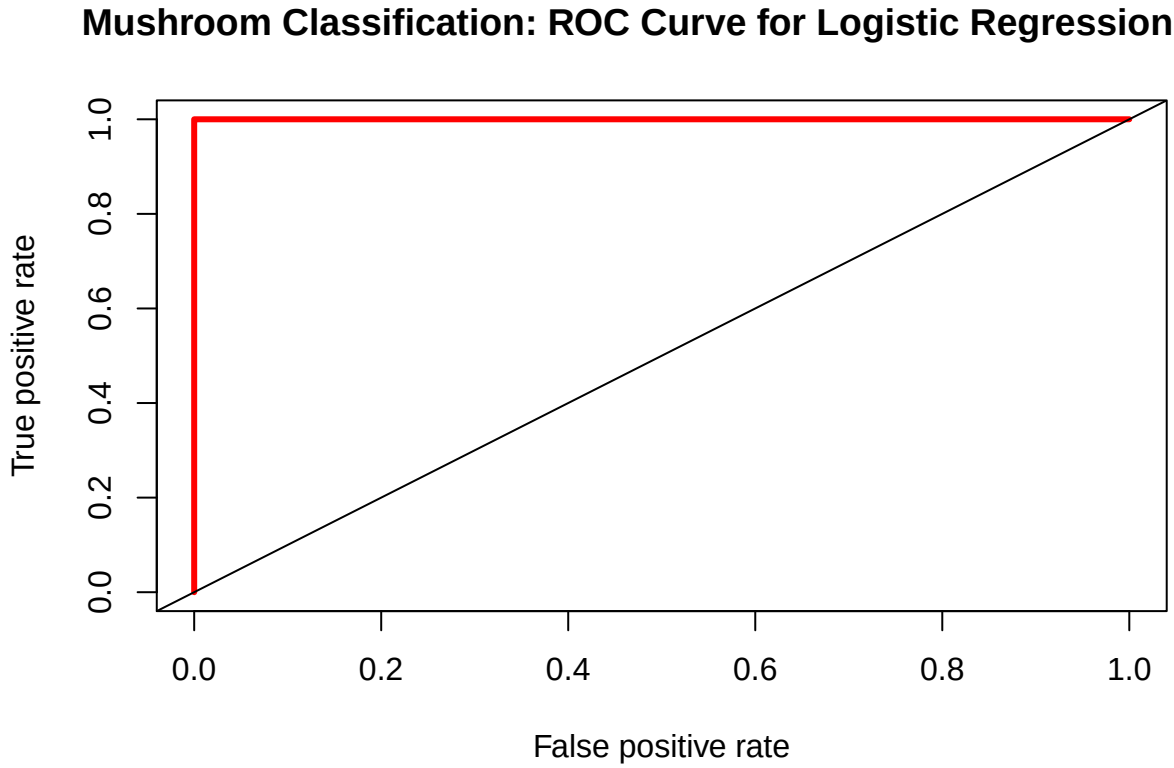
```r
### Manual Logistic Regression

# Reduced Model: Containing Top 5 Covariates On Random Forest Variable Importance Plot
mush.glmRM = glm(class ~ odor + spore.print.color + gill.color + gill.size + ring.type, data = mush, fa
```

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

```r
# Class Prediction Object / ROC Curve Code
pred.glmRM = predict(mush.glmRM, mush, type="response")  # receive output of predicted probabilities
```

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

```
pred.glmRM = prediction(pred.glmRM, mush$class)  # comparing predicted probabilities to true class labe
perf.glmRM = performance(pred.glmRM, measure="tpr", x.measure="fpr")
plot(perf.glm, col=2, lwd=3, main="Mushroom Classification: ROC Curve for Logistic Regression RM")
abline(a = 0, b = 1)
```



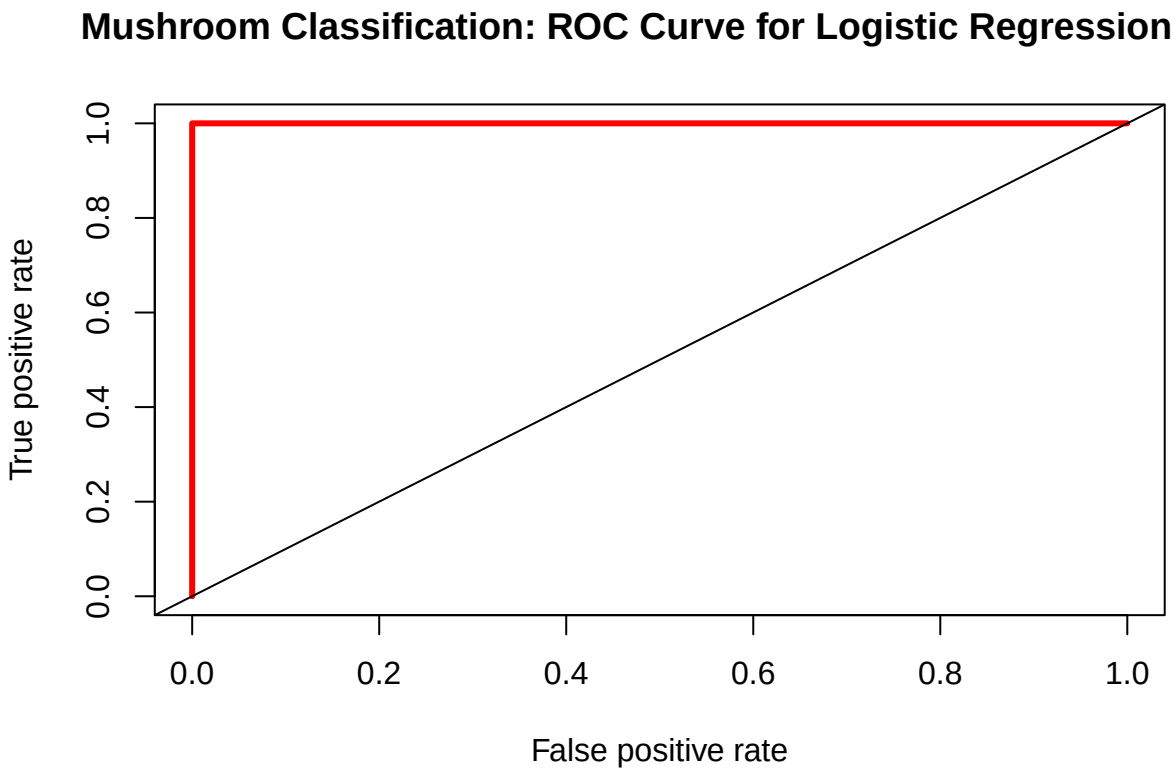
The Reduced Model was chosen according to the top five variables on the Random Forest Variable Importance Plot (caret version). This is because the Logistic Regression Full Model is significantly prone to overfitting and its Variable Importance Plot is therefore not very trustworthy.

Small Model: Containing Only Top Two Most Important Covariates, odor And spore.print.color

```
### Manual Logistic Regression

# Small Model: Containing Only Covariates odor And spore.print.color
mush.glmS = glm(class ~ spore.print.color + odor, data = mush, family = "binomial")
```

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

```
# Class Prediction Object / ROC Curve Code
pred.glmS = predict(mush.glmS, mush, type="response")   # receive output of predicted probabilities
pred.glmS = prediction(pred.glmS, mush$class) # comparing predicted probabilities to true class labels
perf.glmS = performance(pred.glmS, measure="tpr", x.measure="fpr")
plot(perf.glm, col=2, lwd=3, main="Mushroom Classification: ROC Curve for Logistic Regression SM")
abline(a = 0, b = 1)
```



The Small Model utilizes only the top two variables on the Random Forest Variable Importance Plot, odor and spore.print.color. These are also the variables shown through EDA to be the strongest predictors.

Class Prediction Results (Logistic Regression Method)

All Logistic Regression models had near perfectly accurate class prediction. The Full Model has a perfect class prediction but is prone to overfitting. The other two models have a satisfactory class prediction accuracy for their small number of covariates.

Model Assessment, Comparison and Selection

```
# AUC for Random Forest and Logistic Regression Models
auc = matrix(NA, nrow=1, ncol=4)
colnames(auc) <- c("Random Forest", "Logistic Regression FM", "Logistic Regression RM", "Logistic Regres

auc.rf = performance(pred.rf, "auc")@y.values
auc[,1] = auc.rf[[1]]
auc.glm = performance(pred.glm, "auc")@y.values
auc[1,2] = auc.glm[[1]]
auc.glmRM = performance(pred.glmRM, "auc")@y.values
auc[1,3] = auc.glmRM[[1]]
auc.glmS = performance(pred.glmS, "auc")@y.values
auc[1,4] = auc.glmS[[1]]

auc
```

```
##      Random Forest Logistic Regression FM Logistic Regression RM
```

```
## [1,]             1                       1              0.9999476
##      Logistic Regression SM
## [1,]              0.9991611
```

Analysis:

- The AUC (area under the ROC curve) values are very similar and are all close to 1, signifying a perfect or near perfect class prediction for all models.
- In order to select the best model, factors other than AUC must be taken into account.
- Based on the above AUC values and the Variable Importance Plots for Random Forest and Logistic Regression, I conclude the best model to be the Random Forest model. Although the AUCs for the Logistic Regression models are also favorable, the Variable Importance Plot for Logistic Regression is difficult to read because it is based on the Full Model containing all covariates; it also shows signs of overfitting. Because Logistic Regression splits each multiple-factor-level covariate into multiple binary covariates, the number of covariates incorporated in the Full Model can become very large, making it signficantly prone to overfitting. Classification can still be performed well, however, due to the wide spread of the weight distribution of the model on many covariates, this method has trouble deciding variable signficance unless it is reduced. The Random Forest model is most robust in incorporating all predictors.

- In case, of overfitting, I created two reduced Logistic Regression models, the Reduced Model and the Small Model. Out of the three Logistic Regression models, I would assume that one of the reduced models would be preferred to avoid potential overfitting since their AUC values are close enough to 1. The Small Model maintains an accurate AUC value of 0.9991611 with only the two covariates odor and spore.print.color.

Clustering (Unsupervised Learning)

Cluster Mushrooms Using ROCK Clustering (RObust Clustering using linKs) Algorithm

Details: ROCK Algorithm synthesizes clustered according to a dissimilarity measurement based on how many factor-levels each observation has in common with evey other observation (links).

Reasoning: I chose this method in order to cluster categorical data with the hypothesis that I might be able to synthesize clusters representing the 23 species of mushrooms documented.

```
# Searching For Clusters By Species: 23 Species Documented in Dataset, 21 pure clusters discovered with
# 2 clusters/species indistinguishable (NA)

set.seed(1)
x <- as.dummy(mush[-1])
mush.rc <- rockCluster(x[sample(dim(x)[1],1000),], n=10, theta=0.8)
```

```
## Clustering:
## computing distances ...
## computing links ...
## computing clusters ...
## rockMerge: terminated with 29 clusters
```

```
print(mush.rc)
```

```
##  data: x
##  beta: 0.2
## theta: 0.8
##   fun: dist
```

```
##  args: list(method = "binary")
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
## 214  34  39 206 152  77  35  97  26  28  10  41   1   2   9   1   1   9
##  19  20  21  22  23  24  25  26  27  28  29
##   2   3   5   1   1   1   1   1   1   1   1
```

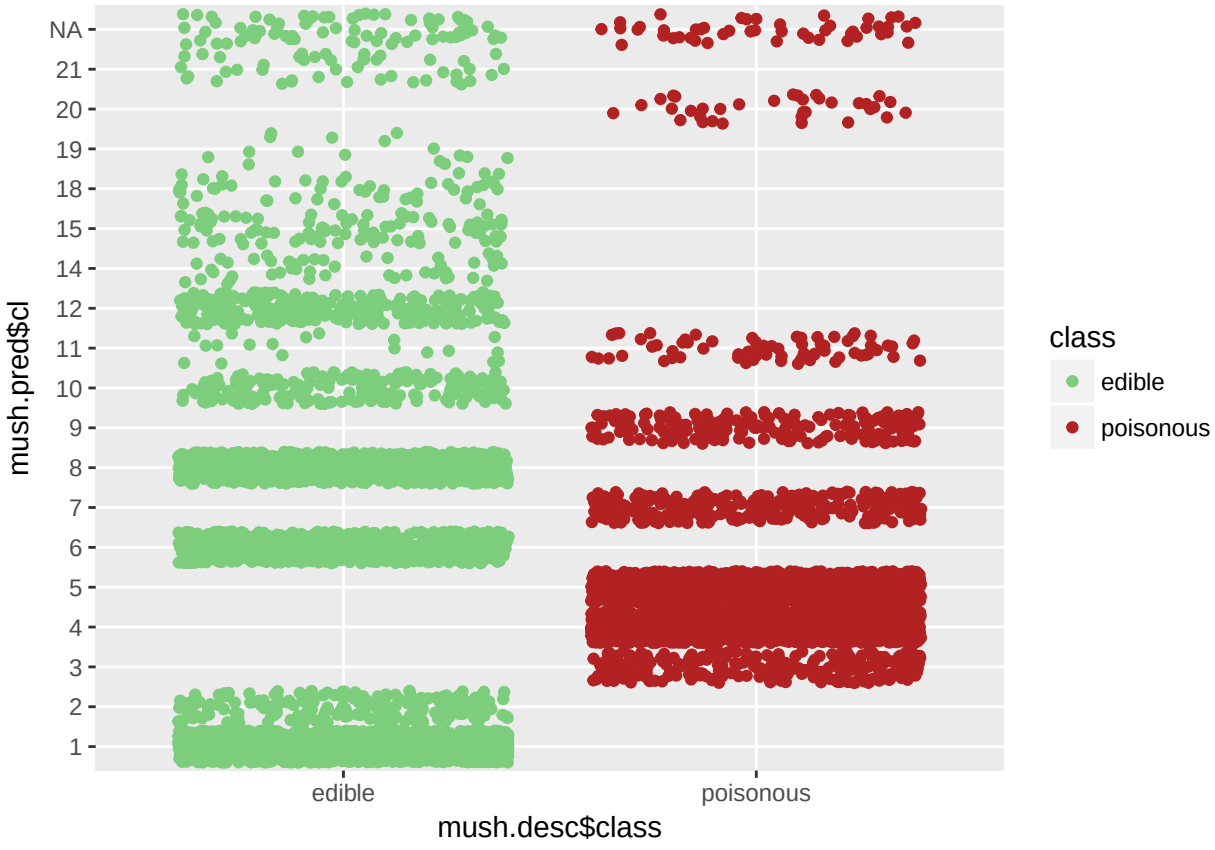
```
mush.pred <- predict(mush.rc, x)
```

```
## dropping 11 clusters
## computing distances ...
## computing classes ...
```

```
table(mush.desc$class, mush.pred$cl)
```

```
##
##                1    2    3    4    5    6    7    8    9   10   11   12
##   edible    1728  192    0    0    0  700    0  768    0  192   19  288
##   poisonous    0    0  256 1728 1296    0  288    0  192    0   72    0
##
##               14   15   18   19   20   21 <NA>
##   edible      47   88   48   16    0   44   78
##   poisonous    0    0    0    0   35    0   49
```

```
# jitter plot of rock clustered class predictions vs. actual class values
rock.plot <- ggplot(mush.desc, aes(mush.desc$class, mush.pred$cl))
rock.plot + geom_jitter(aes(color = class)) + scale_color_manual(values=c("palegreen3", "firebrick"))
```



Analysis:

- The ROCK Algorithm synthesized 21 clusters, 20 of which were pure clusters (either all poisonous or

all edible). These 21 clusters could potentially be groupings aligning with the 23 documented species in the dataset. It also left 127 observations unclustered; these observations are indistinguishable or unable to be grouped with the linkage methods used in the ROCK Algorithm.

It is possible that these unclustered observations are the ambiguous odorless white spore print mushrooms which had so much difficulty being classified in the classification analysis.

Clustering (K-Modes)

Details: Not yet implemented. K-Modes is a variation of K-Means clustering algorithm designed for clustering with categorical covariates. It allows the user to specify cluster number.

Reasoning: I chose to use K-Modes clustering in order to cluster categorical covariates and specify two clusters with the hypothesis that I may find two clusters representing the mushroom Families, Agaricus and Lepiota. It is also possible and potentially likely that these clusters would have a stronger correlation to class level (poisonous, edible).

Conclusions

After analysis and comparison, the best model was chosen to be the Random Forest model, with the runner-up being the Logistic Regression Small Model, which had a high accuracy of class prediction and AUC value with only the covariates odor and spore.print.color. Through the use of EDA and Random Forest method, there was some very unique and deep meaning considering the nature of fungi elaborated upon in the data. Odor is virtually a perfect predictor of edibility, suggesting that our brains and sense organs are virtually perfect predictors of mushroom class. Odor and spore.print.color together are able to classify all mushrooms except for odorless white mushrooms–making this category the most difficult mushrooms to classify and the ones to be most careful when ingesting. These discoveries illustrate the reasons behind the perfect success rates for classification. I was unable to find a third predictor to rule out all ambiguities surrounding white spore print odorless mushrooms. The study also suggest that mushroom colonies have some type of hive/swarm intelligence. What do you think?

References

UC Irvine Machine Learning Repository: http://archive.ics.uci.edu/ml/index.php

TIDYVERSE (On ggplot2 geom_jitter plotting): http://ggplot2.tidyverse.org/reference/geom_jitter.html

Caret Package Documentation: https://cran.r-project.org/web/packages/caret/caret.pdf