mushrooms

Luqi Guan

Mingjie Ye

2018 年 12 月 16 日

Business Understanding

Mushrooms4all is an organization that aims to promote the collection and use of mushrooms for feeding purposes. However, this organization is aware that there are many mushrooms that are not suitable for human consumption, being dangerous to eat them since they can cause poisoning. Although they have a quite complete database of dangerous/non-dangerous mushrooms, they aim to create a machine learning model that, based on the mushroom characteristics, allows to classify a not known mushroom species as safe or not for human consumption.

```
To distinguish between edible mushrooms and poisonous ones by how they
look.
To know whether we can eat mushroom, to survive in the wild.
```

Data Understanding

The dataset includes descriptions of hypothetical samples corresponding to 23 species of mushrooms. Each species is identified as definitely edible, definitely poisonous.

```
mushrooms <- read.csv("mushrooms_v2 (prob 0.05).csv",header=TRUE)
dim(mushrooms)

## [1] 8124   23

str(mushrooms)

## 'data.frame':    8124 obs. of  23 variables:
##  $ class                   : Factor w/ 2 levels "e","p": 2 1 1 2 1 1
 1 1 2 1 ...
##  $ cap.shape               : Factor w/ 6 levels "b","c","f","k",..:
5 6 4 1 1 4 3 1 5 1 ...
##  $ cap.surface             : Factor w/ 4 levels "f","g","s","y": 1 2
 3 4 3 3 1 4 4 3 ...
##  $ cap.color               : Factor w/ 10 levels "b","c","e","g",..:
 4 10 5 9 4 10 9 9 10 10 ...
##  $ bruises                 : Factor w/ 2 levels "f","t": 2 2 2 2 1 1
 2 2 1 2 ...
##  $ odor                    : Factor w/ 9 levels "a","c","f","l",..:
7 1 4 9 6 1 1 4 7 1 ...
```

```
##  $ gill.attachment         : Factor w/ 4 levels "a","d","f","n": 3 2
 3 3 4 3 1 3 3 2 ...
##  $ gill.spacing            : Factor w/ 3 levels "c","d","w": 2 1 3 1
 1 3 1 1 2 1 ...
##  $ gill.size               : Factor w/ 2 levels "b","n": 2 1 1 2 1 2
 1 1 2 1 ...
##  $ gill.color              : Factor w/ 12 levels "b","e","g","h",..:
 5 5 6 2 7 6 3 9 1 3 ...
##  $ stalk.shape             : Factor w/ 2 levels "e","t": 1 1 2 2 2 2
 1 2 2 1 ...
##  $ stalk.root              : Factor w/ 7 levels "?","b","c","e",..:
4 3 3 4 2 7 3 3 6 3 ...
##  $ stalk.surface.above.ring: Factor w/ 4 levels "f","k","s","y": 3 4
 3 3 3 3 3 3 3 4 ...
##  $ stalk.surface.below.ring: Factor w/ 4 levels "f","k","s","y": 3 4
 2 2 3 3 1 3 3 4 ...
##  $ stalk.color.above.ring  : Factor w/ 9 levels "b","c","e","g",..:
2 3 1 8 8 1 8 9 8 8 ...
##  $ stalk.color.below.ring  : Factor w/ 9 levels "b","c","e","g",..:
2 3 1 4 3 1 8 9 8 6 ...
##  $ veil.type               : Factor w/ 2 levels "p","u": 1 1 1 1 1 1
 1 2 1 1 ...
##  $ veil.color              : Factor w/ 4 levels "n","o","w","y": 3 3
 3 3 3 3 3 3 4 3 ...
##  $ ring.number             : Factor w/ 4 levels "b","n","o","t": 4 3
 3 1 1 3 3 4 4 3 ...
##  $ ring.type               : Factor w/ 8 levels "c","e","f","l",..:
6 6 6 5 8 6 6 6 6 6 ...
##  $ spore.print.color       : Factor w/ 9 levels "b","h","k","n",..:
3 7 4 3 8 3 8 9 1 3 ...
##  $ population              : Factor w/ 6 levels "a","c","n","s",..:
3 1 5 4 1 3 3 6 5 4 ...
##  $ habitat                 : Factor w/ 7 levels "d","g","l","m",..:
6 5 2 6 2 5 4 7 7 2 ...
```

```
head(mushrooms)
```

```
##   class cap.shape cap.surface cap.color bruises odor gill.attachment
## 1     p         s           f         g       t    p               f
## 2     e         x           g         y       t    a               d
## 3     e         k           s         n       t    l               f
## 4     p         b           y         w       t    y               f
## 5     e         b           s         g       f    n               n
## 6     e         k           s         y       f    a               f
##   gill.spacing gill.size gill.color stalk.shape stalk.root
## 1            d         n          k           e          e
## 2            c         b          k           e          c
## 3            w         b          n           t          c
## 4            c         n          e           t          e
## 5            c         b          o           t          b
```

```
## 6            w         n          n           t          z
##   stalk.surface.above.ring stalk.surface.below.ring stalk.color.abov
e.ring
## 1                        s                        s
     c
## 2                        y                        y
     e
## 3                        s                        k
     b
## 4                        s                        k
     w
## 5                        s                        s
     w
## 6                        s                        s
     b
##   stalk.color.below.ring veil.type veil.color ring.number ring.type
## 1                      c         p          w           t         p
## 2                      e         p          w           o         p
## 3                      b         p          w           o         p
## 4                      g         p          w           b         n
## 5                      e         p          w           b         z
## 6                      b         p          w           o         p
##   spore.print.color population habitat
## 1                 k          n       u
## 2                 u          a       p
## 3                 n          v       g
## 4                 k          s       u
## 5                 w          a       g
## 6                 k          n       p
```

```
summary(mushrooms)
```

```
##  class    cap.shape cap.surface   cap.color    bruises       odor

##  e:4208   b: 968    f:2182      n      :1488   f:4329   n      :2098

##  p:3916   c: 715    g:1141      g      :1245   t:3795   f      :1468

##           f:2195    s:2227      e      :1154            y      : 768

##           k:1075    y:2574      w      : 943            s      : 729

##           s: 742                y      : 914            a      : 689

##           x:2429                p      : 536            l      : 666

##                                 (Other):1844            (Other):1706

##  gill.attachment gill.spacing gill.size   gill.color   stalk.shape
##  a:1259          c:4626       b:4843    b      :1161   e:3873
```

```
##  d:1126          d:1481       n:3281    p      :1042   t:4251
##  f:4644          w:2017                 w      : 915
##  n:1095                                 n      : 813
##                                         h      : 735
##                                         g      : 697
##                                         (Other):2761
##  stalk.root stalk.surface.above.ring stalk.surface.below.ring
##  ?:1121     f:1343                   f:1387
##  b:2513     k:2178                   k:2160
##  c: 963     s:3458                   s:3363
##  e:1177     y:1145                   y:1214
##  r: 828
##  u: 772
##  z: 750
##  stalk.color.above.ring stalk.color.below.ring veil.type veil.color
##  w      :2501           w      :2446           p:5968    n:1135
##  p      :1341           p      :1411           u:2156    o:1182
##  g      : 769           g      : 786                     w:4703
##  n      : 709           n      : 737                     y:1104
##  b      : 683           b      : 665
##  o      : 592           o      : 549
##  (Other):1529           (Other):1530
##  ring.number   ring.type    spore.print.color population habitat
##  b:1509      p      :2358   w      :1538      a: 964     d:2072
##  n:  11      e      :1842   n      :1388      c: 899     g:1599
##  o:4835      l      :1128   k      :1367      n: 893     l:1005
##  t:1769      c      : 588   h      :1241      s:1313     m: 711
##              n      : 567   u      : 551      v:2591     p:1180
##              s      : 561   r      : 527      y:1464     u: 845
##              (Other):1080   (Other):1512                 w: 712
```

Contingence tables are useful for revealing how edible/poisonous mushrooms are segmented across their dataset features.

```
mush_features <- colnames(mushrooms)[-1]
tables <- lapply(mush_features, function(x) {table(mushrooms$class, mus
hrooms[,x])})
names(tables) <- mush_features
print(tables)
```

```
## $cap.shape
##
##        b    c    f    k    s    x
##   e  590  379 1152  450  376 1261
##   p  378  336 1043  625  366 1168
##
## $cap.surface
##
##        f    g    s    y
##   e 1253  602 1102 1251
```

```
##   p  929  539 1125 1323
##
## $cap.color
##
##       b   c   e   g   n   p   r   u   w   y
##   e 259 235 508 679 823 260 236 227 569 412
##   p 255 234 646 566 665 276 186 212 374 502
##
## $bruises
##
##       f    t
##   e 1792 2416
##   p 2537 1379
##
## $odor
##
##       a    c    f    l    m    n    p    s    y
##   e  448  255  259  438  251 1829  266  223  239
##   p  241  321 1209  228  253  269  360  506  529
##
## $gill.attachment
##
##       a    d    f    n
##   e  666  592 2385  565
##   p  593  534 2259  530
##
## $gill.spacing
##
##       c    d    w
##   e 2188  732 1288
##   p 2438  749  729
##
## $gill.size
##
##       b    n
##   e 2976 1232
##   p 1867 2049
##
## $gill.color
##
##       b   e   g   h   k   n   o   p   r   u   w   y
##   e 196 213 313 313 330 610 241 594 184 390 618 206
##   p 965 171 384 422 240 203 180 448 207 195 297 204
##
## $stalk.shape
##
##       e    t
##   e 1919 2289
##   p 1954 1962
##
```

```
## $stalk.root
##
##        ?    b    c    e    r    u    z
##   e  320 1302  602  750  458  397  379
##   p  801 1211  361  427  370  375  371
##
## $stalk.surface.above.ring
##
##        f    k    s    y
##   e  741  627 2236  604
##   p  602 1551 1222  541
##
## $stalk.surface.below.ring
##
##        f    k    s    y
##   e  788  634 2122  664
##   p  599 1526 1241  550
##
## $stalk.color.above.ring
##
##        b    c    e    g    n    o    p    w    y
##   e  236  268  313  502  265  351  523 1514  236
##   p  447  250  212  267  444  241  818  987  250
##
## $stalk.color.below.ring
##
##        b    c    e    g    n    o    p    w    y
##   e  247  263  326  542  285  314  539 1477  215
##   p  418  271  207  244  452  235  872  969  248
##
## $veil.type
##
##        p    u
##   e 3064 1144
##   p 2904 1012
##
## $veil.color
##
##        n    o    w    y
##   e  600  644 2402  562
##   p  535  538 2301  542
##
## $ring.number
##
##        b    n    o    t
##   e  815    0 2394  999
##   p  694   11 2441  770
##
## $ring.type
##
```

```
##        c    e    f    l    n    p    s    z
##   e  304  744  295  298  281 1721  291  274
##   p  284 1098  244  830  286  637  270  267
##
## $spore.print.color
##
##        b    h    k    n    o    r    u    w    y
##   e  279  269 1026 1053  267  248  322  516  228
##   p  245  972  341  335  243  279  229 1022  250
##
## $population
##
##        a    c    n    s    v    y
##   e  590  531  546  793  927  821
##   p  374  368  347  520 1664  643
##
## $habitat
##
##        d    g    l    m    p    u    w
##   e 1195  963  447  400  408  395  400
##   p  877  636  558  311  772  450  312
```

From the tables, we can know that when ring number= none, all the mushrooms are poisonous.

Here is the correlation analysis between mushroom features and class.

```
chisq_test_res = list()
for (i in 2:length(colnames(mushrooms))) {
  fname = colnames(mushrooms)[i]
  res = chisq.test(mushrooms[,i], mushrooms[,"class"], simulate.p.value
 = TRUE)
  res$data.name = paste(fname, "class", sep= " and ")
  chisq_test_res[[fname]] = res
}
chisq_test_res
```

```
## $cap.shape
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  cap.shape and class
## X-squared = 76.216, df = NA, p-value = 0.0004998
##
##
## $cap.surface
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
```

```
## data:  cap.surface and class
## X-squared = 43.401, df = NA, p-value = 0.0004998
##
##
## $cap.color
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  cap.color and class
## X-squared = 89.289, df = NA, p-value = 0.0004998
##
##
## $bruises
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  bruises and class
## X-squared = 401.6, df = NA, p-value = 0.0004998
##
##
## $odor
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  odor and class
## X-squared = 2136.5, df = NA, p-value = 0.0004998
##
##
## $gill.attachment
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  gill.attachment and class
## X-squared = 1.2639, df = NA, p-value = 0.7446
##
##
## $gill.spacing
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  gill.spacing and class
## X-squared = 158.34, df = NA, p-value = 0.0004998
##
##
## $gill.size
```

```
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  gill.size and class
## X-squared = 447.47, df = NA, p-value = 0.0004998
##
##
## $gill.color
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  gill.color and class
## X-squared = 954.32, df = NA, p-value = 0.0004998
##
##
## $stalk.shape
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  stalk.shape and class
## X-squared = 14.994, df = NA, p-value = 0.0004998
##
##
## $stalk.root
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  stalk.root and class
## X-squared = 358.67, df = NA, p-value = 0.0004998
##
##
## $stalk.surface.above.ring
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  stalk.surface.above.ring and class
## X-squared = 697.6, df = NA, p-value = 0.0004998
##
##
## $stalk.surface.below.ring
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  stalk.surface.below.ring and class
```

```
## X-squared = 625.93, df = NA, p-value = 0.0004998
##
##
## $stalk.color.above.ring
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  stalk.color.above.ring and class
## X-squared = 389.04, df = NA, p-value = 0.0004998
##
##
## $stalk.color.below.ring
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  stalk.color.below.ring and class
## X-squared = 409.33, df = NA, p-value = 0.0004998
##
##
## $veil.type
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  veil.type and class
## X-squared = 1.8783, df = NA, p-value = 0.1814
##
##
## $veil.color
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  veil.color and class
## X-squared = 5.2712, df = NA, p-value = 0.1719
##
##
## $ring.number
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  ring.number and class
## X-squared = 40.361, df = NA, p-value = 0.0004998
##
##
## $ring.type
##
```

```
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  ring.type and class
## X-squared = 814.25, df = NA, p-value = 0.0004998
##
##
## $spore.print.color
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  spore.print.color and class
## X-squared = 1292.4, df = NA, p-value = 0.0004998
##
##
## $population
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  population and class
## X-squared = 400.36, df = NA, p-value = 0.0004998
##
##
## $habitat
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  habitat and class
## X-squared = 255.65, df = NA, p-value = 0.0004998
```

What we can know from the results is that there are three features: veil color, veil type, gill attachment with high p value, which means that these three features don't have significant relationship with the class.

We also can perform query on the mushroom database to analyse specific subset of the overall available information. We are going to show some example using facilities within the sqldf R package.

```
query_1 <- sqldf("select class,population from mushrooms where habitat
=='d'")
table(query_1)
```

```
##      population
## class   a   c   n   s   v   y
##     e 139 117 113 143 355 328
##     p  79  93  78 107 366 154
```

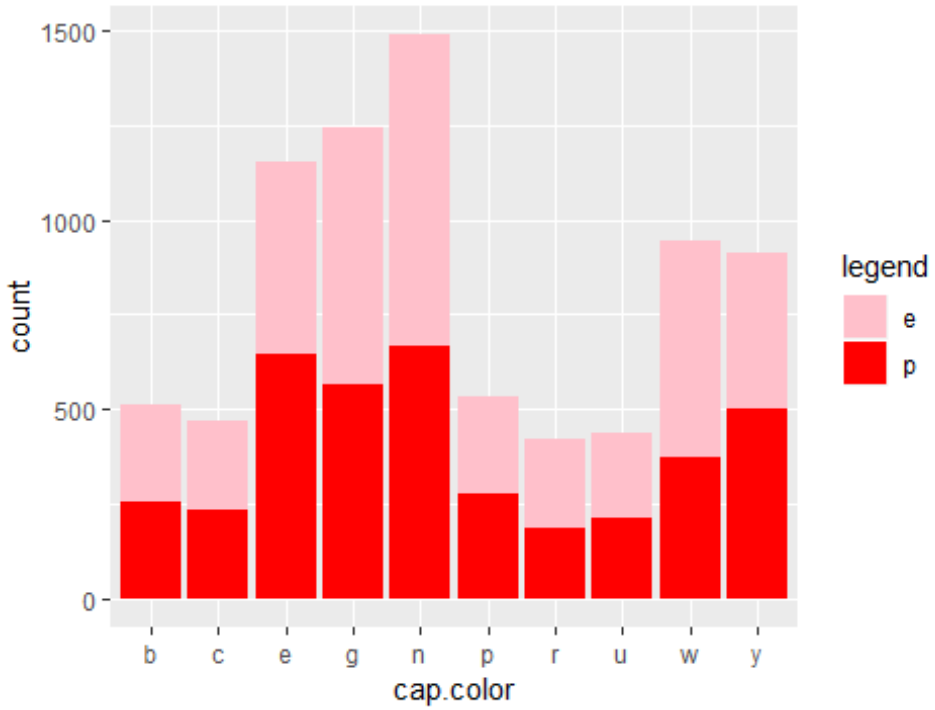
```
query_2 <- sqldf("select class,odor from mushrooms where bruises =='t'")
table(query_2)
```

```
##       odor
## class    a    c    f    l    m    n    p    s    y
##     e  286  139  140  286  141 1000  147  132  145
##     p   78  106  447   81   81   91  174  157  164
```

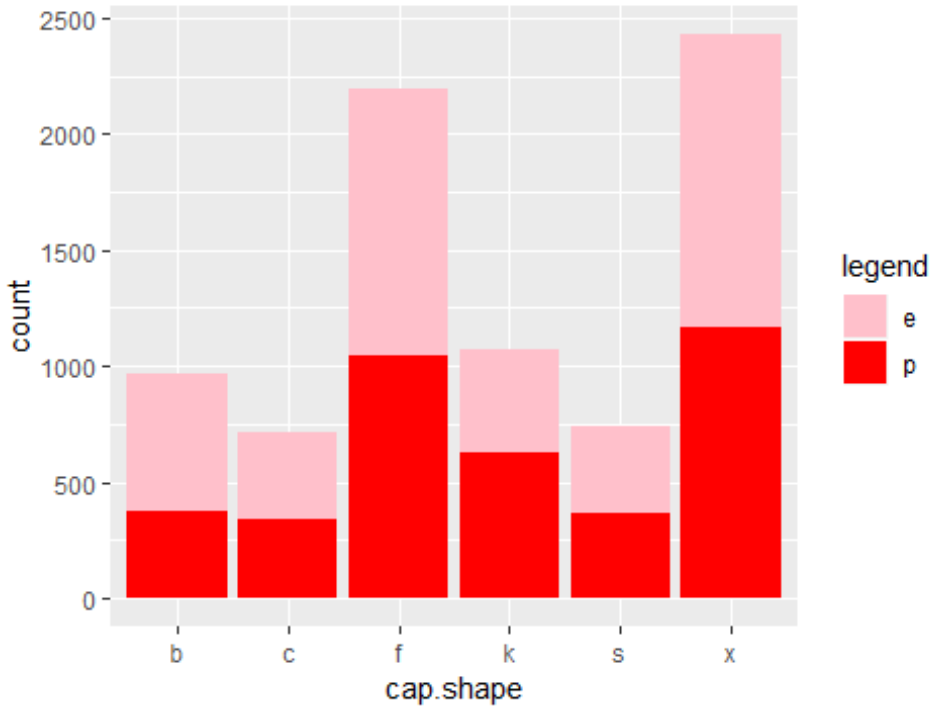
Data Distribution Visualization

Here we are going to show the distribution in barplot and pie of cap shape and cap color features, and all the features we can see in the app.

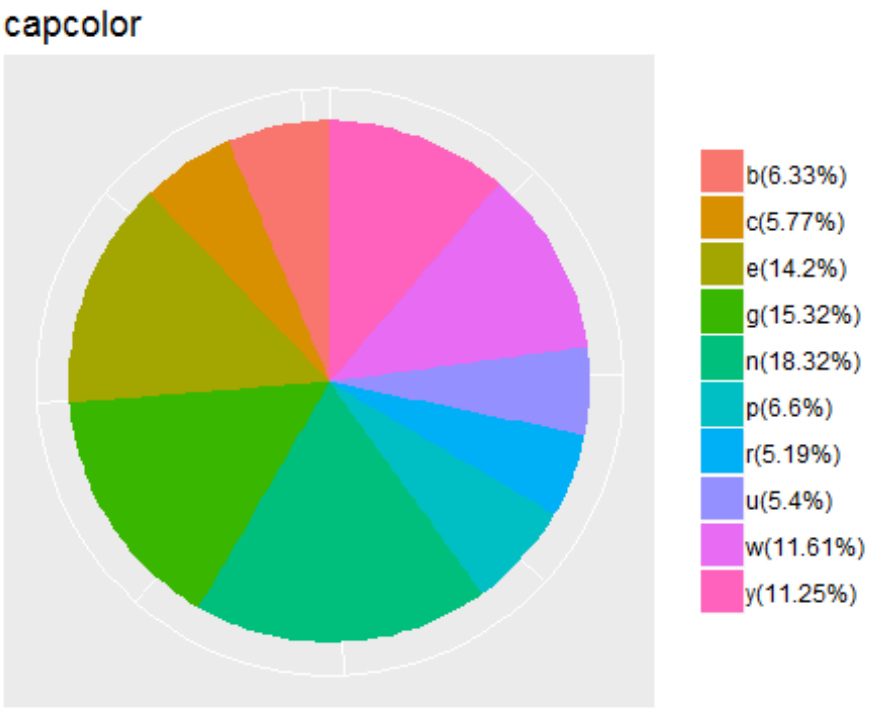
```
ggplot(data=mushrooms, aes(x = cap.color, fill = class)) + geom_bar()
+ scale_fill_manual("legend", values = c("e" = "pink", "p" = "red")) +
ggtitle("")
```



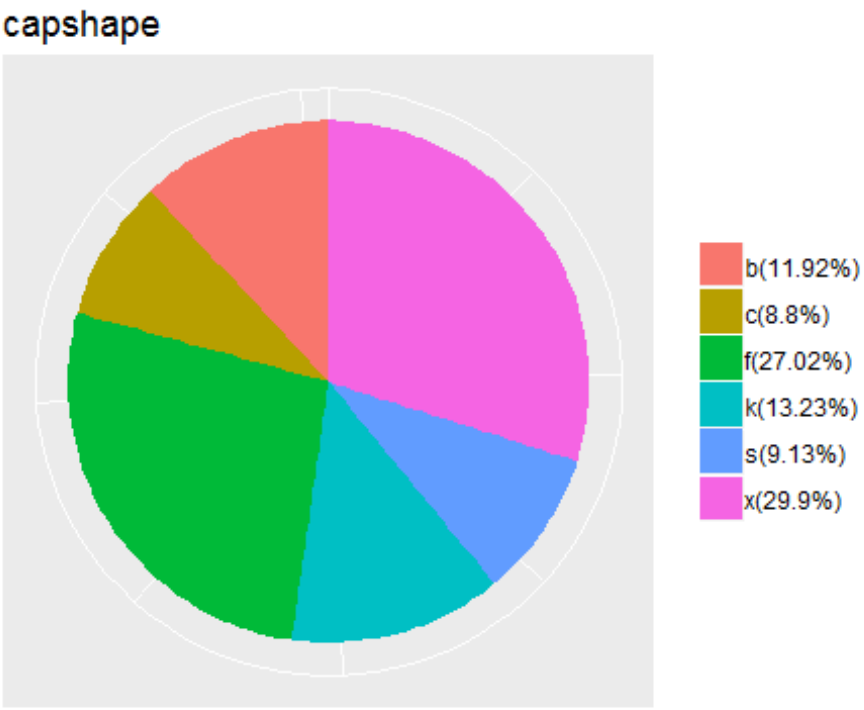

```
ggplot(data=mushrooms, aes(x = cap.shape, fill = class)) + geom_bar()
+ scale_fill_manual("legend", values = c("e" = "pink", "p" = "red")) +
ggtitle("")
```




```
data1<-data.frame(table(mushrooms$cap.color))
myLabel = as.vector(data1$Var1)
myLabel = paste(myLabel, "(", round(data1$Freq / sum(data1$Freq) * 100,
 2), "%)", sep = "")
ggplot(data1, aes(x = "", y = Freq, fill = Var1)) + geom_bar(stat = "id
entity", width = 1) +scale_fill_discrete(breaks = data1$Var1, labels =
myLabel)+coord_polar(theta = "y") + labs(x = "", y = "", title = "") +
theme(axis.ticks = element_blank()) + theme(axis.text = element_blank())
+ theme(legend.title = element_blank())+ ggtitle("capcolor")
```




```
data2<-data.frame(table(mushrooms$cap.shape))
myLabel2 = as.vector(data2$Var1)
myLabel2 = paste(myLabel2, "(", round(data2$Freq / sum(data2$Freq) * 10
0, 2), "%)", sep = "")
ggplot(data2, aes(x = "", y = Freq, fill = Var1)) + geom_bar(stat = "id
entity", width = 1) +scale_fill_discrete(breaks = data2$Var1, labels =
myLabel2)+coord_polar(theta = "y") + labs(x = "", y = "", title = "") +
 theme(axis.ticks = element_blank()) + theme(axis.text = element_blank
())+ theme(legend.title = element_blank()) + ggtitle("capshape")
```



Data Preparation

After checking the NA and INF values in the dataset, we know the dataset is well-structured and clean. Only some "?" values in the stalk.root feature and we consider it as a new seperate category.

In order to use machine learning algorithms, we create 70:30 stratified split using caret between Train and Test. Therefore, we get x_train, x_test, y_train and y_test.

Because the model will be evaluated, we create a train control object for 10 fold CV repeated 2 times. This is important to make sure we do not over-fit the training data. The process of 10-fold cross-validation is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:

- Take the group as a hold out or test data set
- Take the remaining groups as a training data set
- Fit a model on the training set and evaluate it on the test set
- Retain the evaluation score and discard the model

4. Summarize the skill of the model using the sample of model evaluation scores

Build Models

We already know it is a binary classification problem with all categorical variables. In this section, we built five different models to train, test and predict. After evaluation, we compared these models and chose the best model: Random Forest.

Random Forest

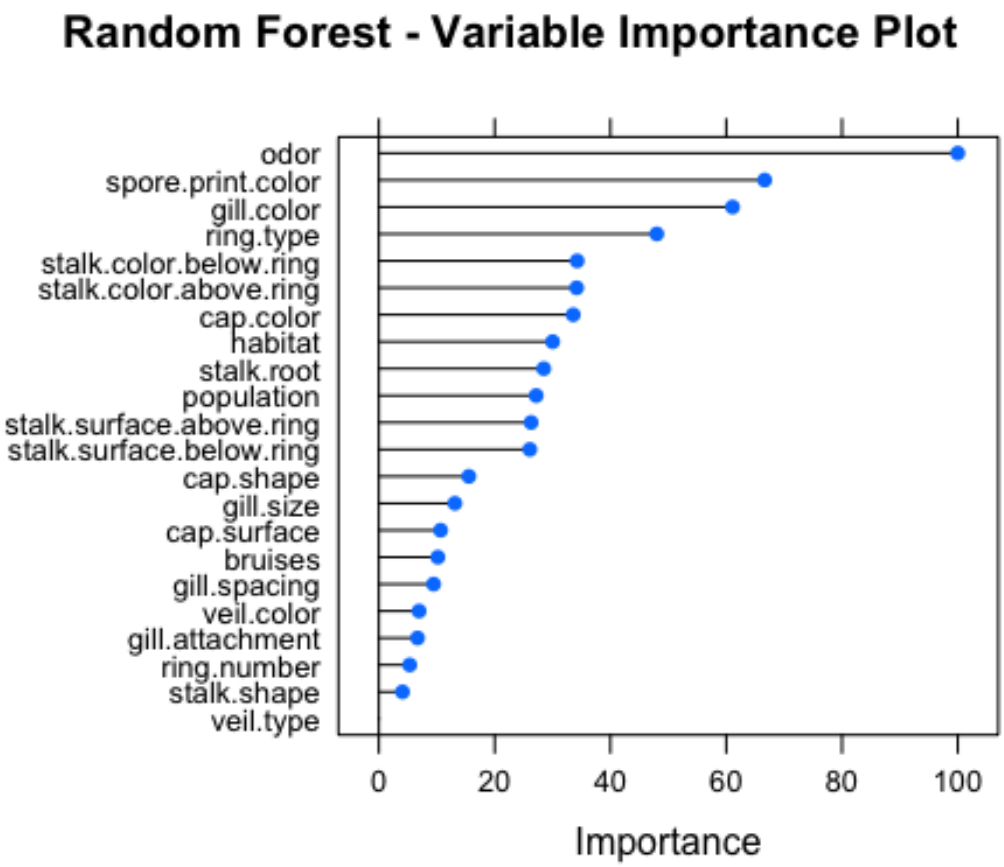
First, we choose random forest model with the help of caret package. Random Forest model can deal with large numbers of predictor variables even in the presence of complex interactions. The "tuneLength" parameter defines the total number of parameter combinations that will be evaluated. In this case, we defined as 3.

```
# Model 1: random forest
rf.1.cv<-train(x_train,y_train,method="rf",trControl=ctrl.1,tuneLength=
3)
# predict on test set and see the confusion matrix
y_predicted<-predict(rf.1.cv,x_test)
df1<-data.frame(Orig=y_test,Pred=y_predicted)
confusionMatrix(table(df1$Orig,df1$Pred))
```

```
## Confusion Matrix and Statistics
##
##
##        e    p
##   e 1168   94
##   p  150 1025
##
##                Accuracy : 0.8999
##                  95% CI : (0.8873, 0.9115)
##     No Information Rate : 0.5408
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7992
##  Mcnemar's Test P-Value : 0.0004299
##
##             Sensitivity : 0.8862
##             Specificity : 0.9160
##          Pos Pred Value : 0.9255
##          Neg Pred Value : 0.8723
##              Prevalence : 0.5408
##          Detection Rate : 0.4793
##    Detection Prevalence : 0.5178
##       Balanced Accuracy : 0.9011
##
##        'Positive' Class : e
##
```

From the confusion matrix, we could find the accuracy is 0.8966. By using `varImp()`, we could also see the variable importance in this model.

```
plot(varImp(rf.1.cv),main="Random Forest - Variable Importance Plot")
```



RPART

The second model is Recursive Partitioning and Regression Trees (RPART) model. Recursive partitioning methods have become popular and widely used tools for non-parametric regression and classification in many scientific fields. Especially random forests, have been applied successfully in genetics, clinical medicine and bioinformatics within the past few years.

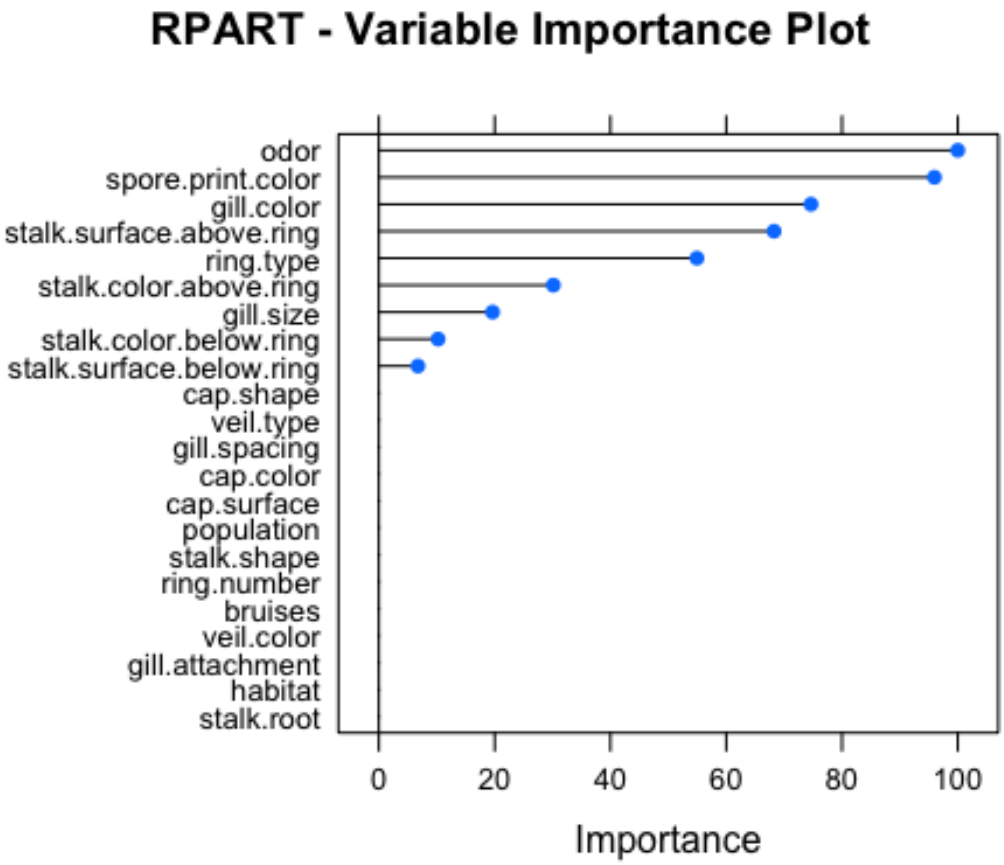
```
# Model 2: RPART model
rpart <-train(x=x_train,y=y_train,method="rpart",trControl=ctrl.1,tuneL
ength=3)
y2_predicted<-predict(rpart,x_test)
df2<-data.frame(Orig=y_test,Pred=y2_predicted)
confusionMatrix(table(df2$Orig,df2$Pred))  #<-100% accuracy

## Confusion Matrix and Statistics
##
##
```

```
##       e   p
##   e 979 283
##   p 278 897
##
##                Accuracy : 0.7698
##                  95% CI : (0.7526, 0.7864)
##     No Information Rate : 0.5158
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5391
##  Mcnemar's Test P-Value : 0.8659
##
##             Sensitivity : 0.7788
##             Specificity : 0.7602
##          Pos Pred Value : 0.7758
##          Neg Pred Value : 0.7634
##              Prevalence : 0.5158
##          Detection Rate : 0.4017
##    Detection Prevalence : 0.5178
##       Balanced Accuracy : 0.7695
##
##        'Positive' Class : e
##
```

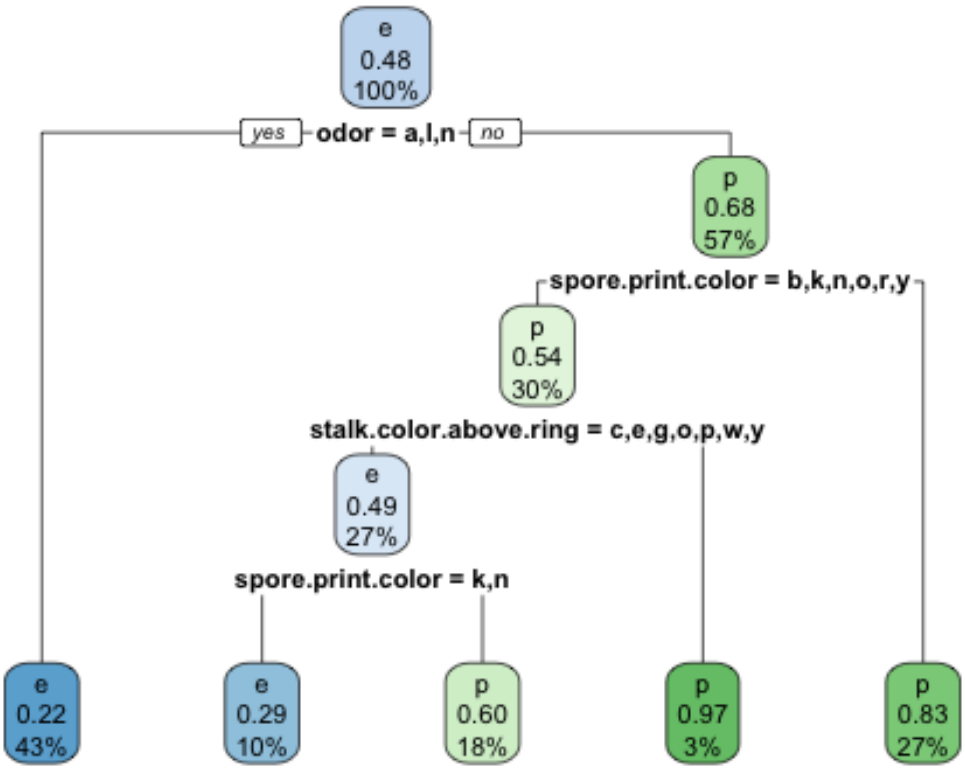
The accuarcy is 0.7698. Only 9 varibles have importance in this model according to the RPART - Variable Importance Plot.

```
plot(varImp(rpart),main="RPART - Variable Importance Plot")
```



The resulting models can also be represented as binary trees showed below.

```
rpart.plot(rpart$finalModel) #<- creates the decision tree with better
formatting
```



Bayesian Generalized Linear Model

The third model is Bayesian Generalized Linear Model.

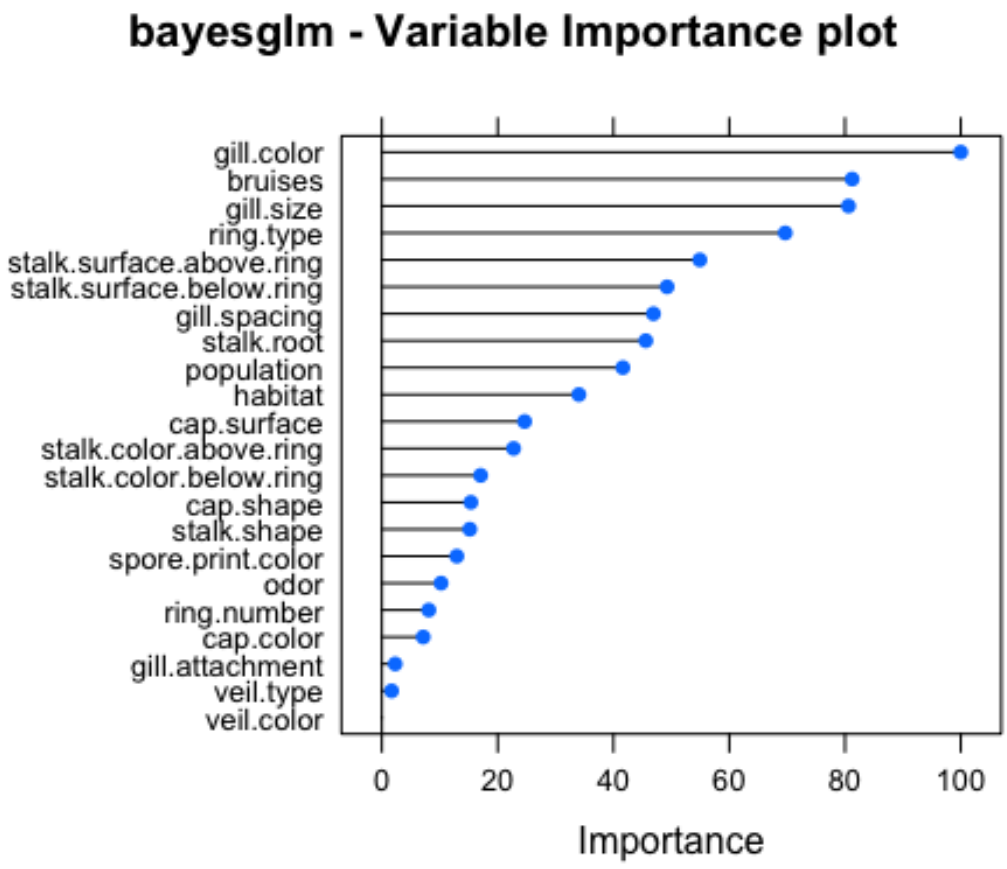
```
# Model 3: bayesglm model
bayesglm <- train(x_train, y_train, method = "bayesglm", trControl = ct
rl.1, tuneLength=3)
y3_predicted <- predict(bayesglm, x_test)
df3 <- data.frame (Original = y_test, Predicted = y3_predicted)
confusionMatrix(table(df3$Original, df3$Predicted))

## Confusion Matrix and Statistics
##
##
##        e    p
##   e 1099  163
##   p  160 1015
##
##                Accuracy : 0.8675
##                  95% CI : (0.8534, 0.8807)
##     No Information Rate : 0.5166
##     P-Value [Acc > NIR] : <2e-16
##
```

```
##                   Kappa : 0.7346
##  Mcnemar's Test P-Value : 0.9114
##
##             Sensitivity : 0.8729
##             Specificity : 0.8616
##          Pos Pred Value : 0.8708
##          Neg Pred Value : 0.8638
##              Prevalence : 0.5166
##          Detection Rate : 0.4510
##    Detection Prevalence : 0.5178
##       Balanced Accuracy : 0.8673
##
##        'Positive' Class : e
##
```

The accuracy is 0.8675 and the variables importance plot is showed.

```
plot(varImp(bayesglm), main = "bayesglm - Variable Importance plot")
```



C5.0 Classification Model

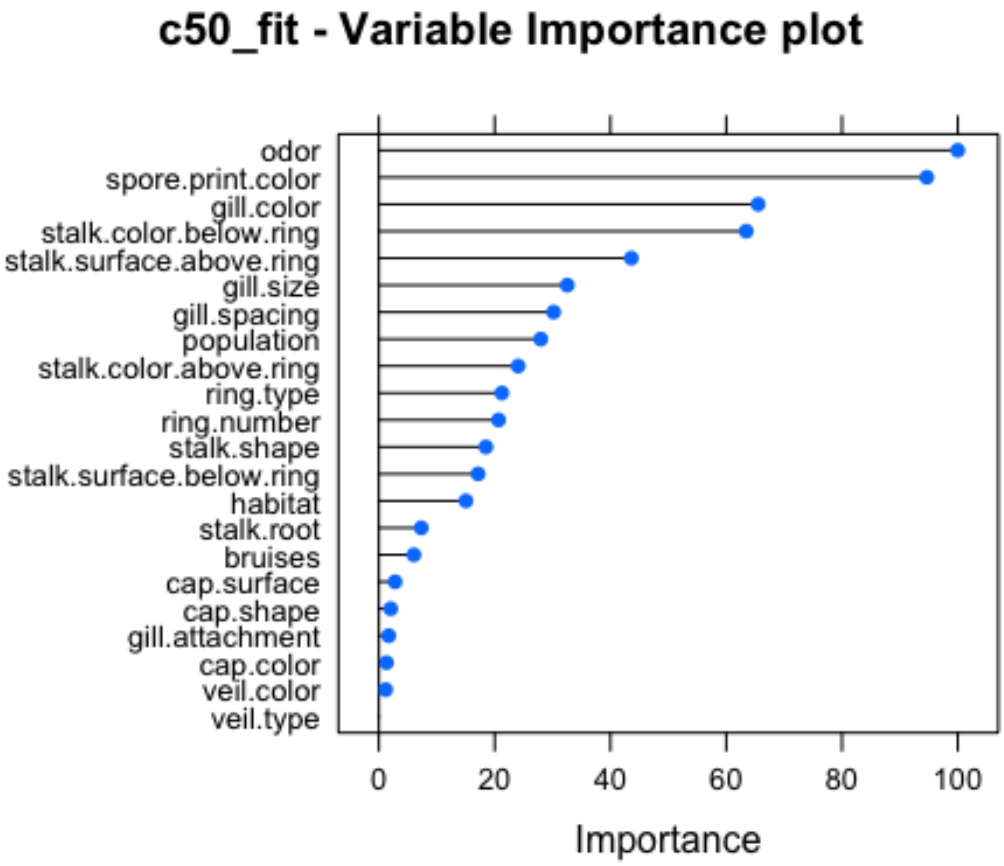
Then we used another tree-based model: C5.0 Classification Model. It is a rule-based model and accords to our practical prediction.

```r
# Model 4: C5.0
c50_fit <- train(x_train, y_train, method ="C5.0Rules", trControl = ctr
l.1, tuneLength=3)
y4_predicted <- predict(c50_fit, x_test)
df4 <- data.frame (Original = y_test, Predicted = y4_predicted)
confusionMatrix(table(df4$Original, df4$Predicted))
```

```
## Confusion Matrix and Statistics
##
##
##        e    p
##   e 1120  142
##   p  185  990
##
##                Accuracy : 0.8658
##                  95% CI : (0.8516, 0.8791)
##     No Information Rate : 0.5355
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.731
##  Mcnemar's Test P-Value : 0.0202
##
##             Sensitivity : 0.8582
##             Specificity : 0.8746
##          Pos Pred Value : 0.8875
##          Neg Pred Value : 0.8426
##              Prevalence : 0.5355
##          Detection Rate : 0.4596
##    Detection Prevalence : 0.5178
##       Balanced Accuracy : 0.8664
##
##        'Positive' Class : e
##
```

The accuracy is 0.8658.

```r
plot(varImp(c50_fit), main = "c50_fit - Variable Importance plot")
```



Conditional Tree

The traditional recursive partitioning algorithms like CART has two fundamental problems: overfitting and a selection bias towards covariates with many possible splits. However, Conditional trees estimate a regression relationship by binary recursive partitioning in a conditional inference framework.

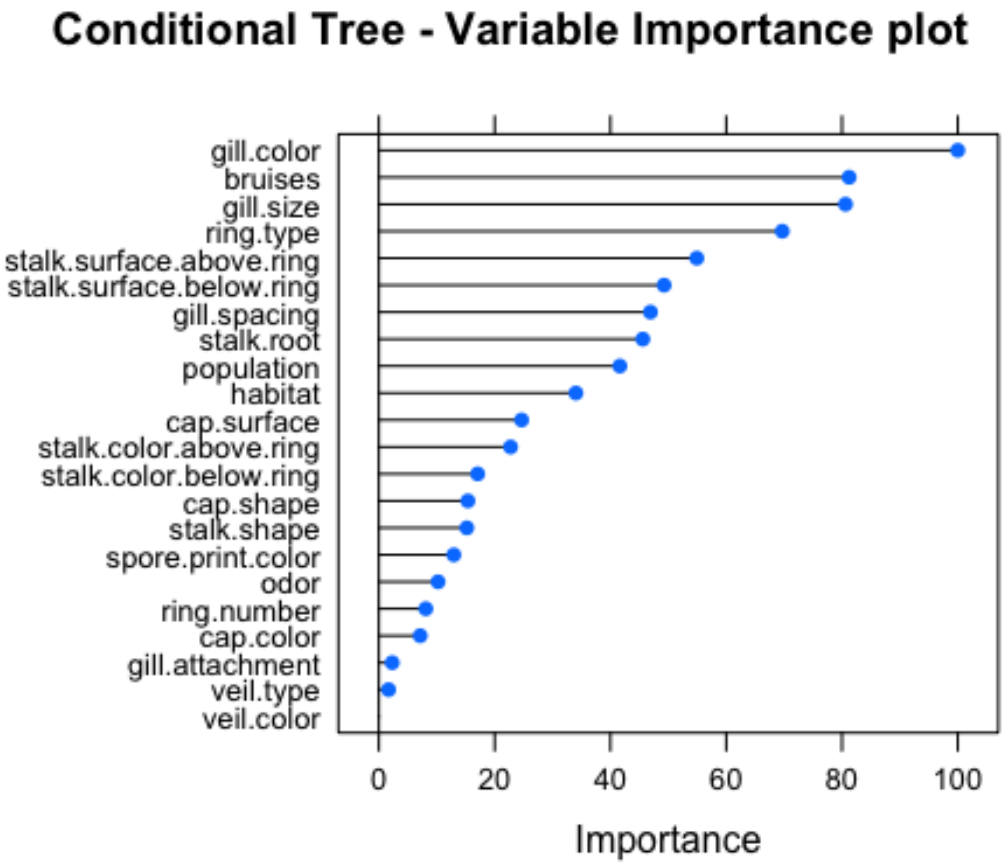
Roughly, the algorithm works as follows according to rdocumentation.com: 1) Test the global null hypothesis of independence between any of the input variables and the response (which may be multivariate as well). Stop if this hypothesis cannot be rejected. Otherwise select the input variable with strongest association to the resonse. This association is measured by a p-value corresponding to a test for the partial null hypothesis of a single input variable and the response. 2) Implement a binary split in the selected input variable. 3) Recursively repeate steps 1) and 2).

```
# Model 5: Conditional Tree
ctree <- train(x_train, y_train, method ="ctree", trControl = ctrl.1, t
uneLength=3)
y5_predicted <- predict(ctree, x_test)
df5 <- data.frame (Original = y_test, Predicted = y5_predicted)
confusionMatrix(table(df5$Original, df5$Predicted))
```

```
## Confusion Matrix and Statistics
##
##
##        e    p
##   e 1072  190
##   p  208  967
##
##                Accuracy : 0.8367
##                  95% CI : (0.8214, 0.8512)
##     No Information Rate : 0.5252
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6728
##  Mcnemar's Test P-Value : 0.3941
##
##             Sensitivity : 0.8375
##             Specificity : 0.8358
##          Pos Pred Value : 0.8494
##          Neg Pred Value : 0.8230
##              Prevalence : 0.5252
##          Detection Rate : 0.4399
##    Detection Prevalence : 0.5178
##       Balanced Accuracy : 0.8366
##
##        'Positive' Class : e
##
```

The accuracy is 0.8367, which is better than RPART.

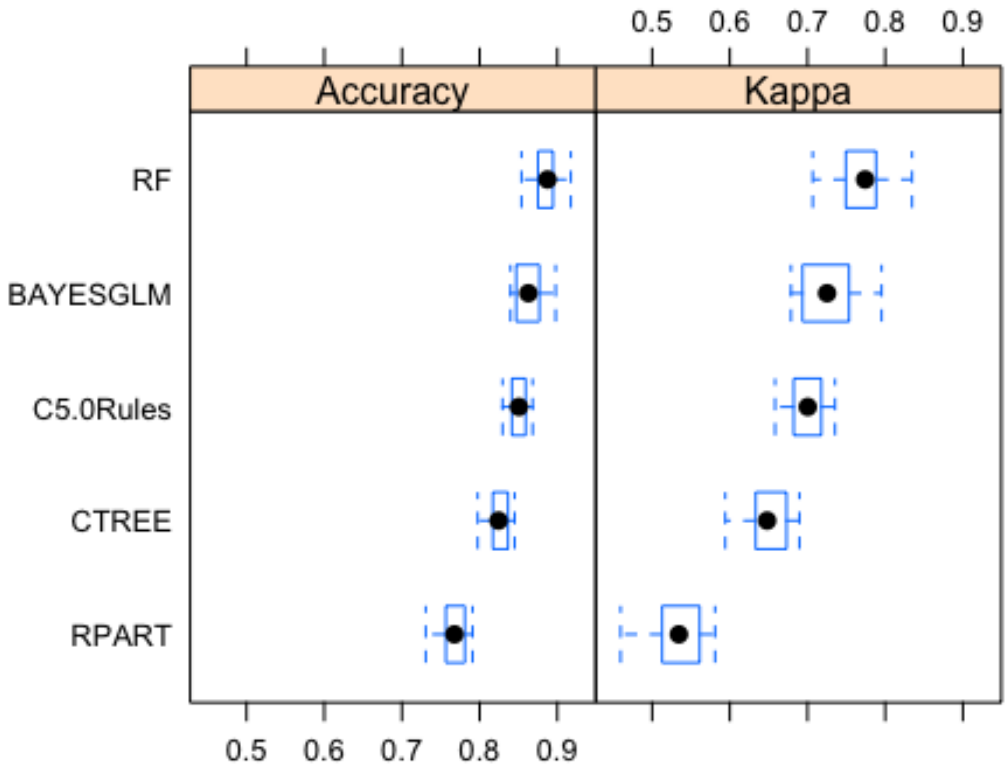
```
plot(varImp(ctree), main = "Conditional Tree - Variable Importance plot
")
```



Models Evaluation

Finally, we compared these five models in accuary and kappa. According to the figure, random forest is the best among our models no matter in accuarcy or kappa. Therefore, we chose random forest as our final model and predict users' input data.

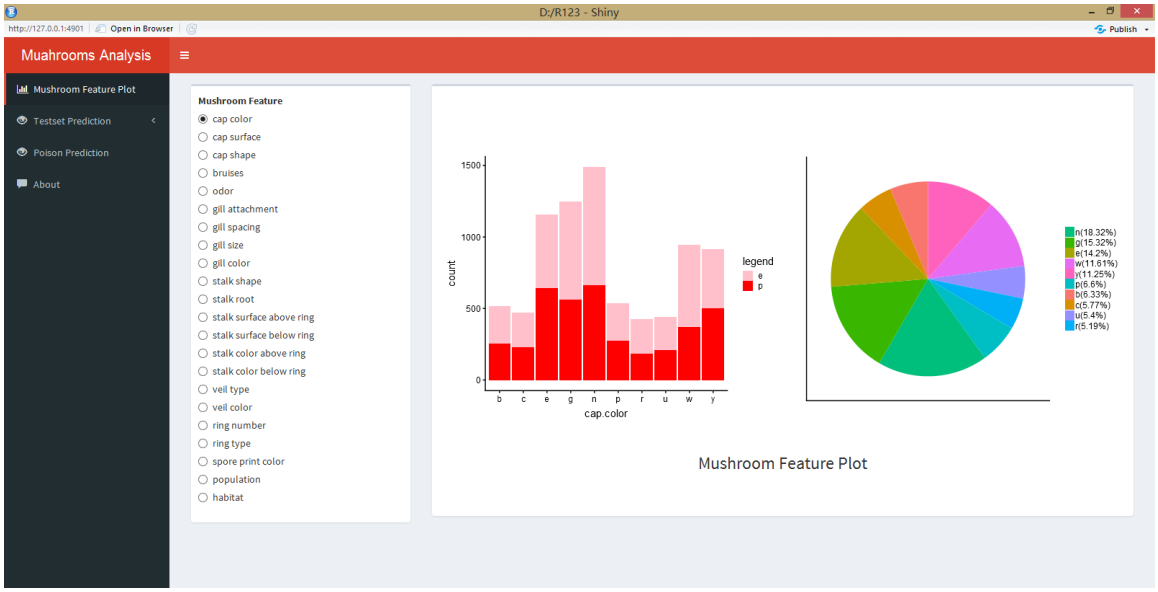
```
# Comparing Models
results <- resamples(list(RF=rf.1.cv, RPART=rpart, BAYESGLM=bayesglm, C
5.0Rules=c50_fit, CTREE=ctree))
bwplot(results)
```



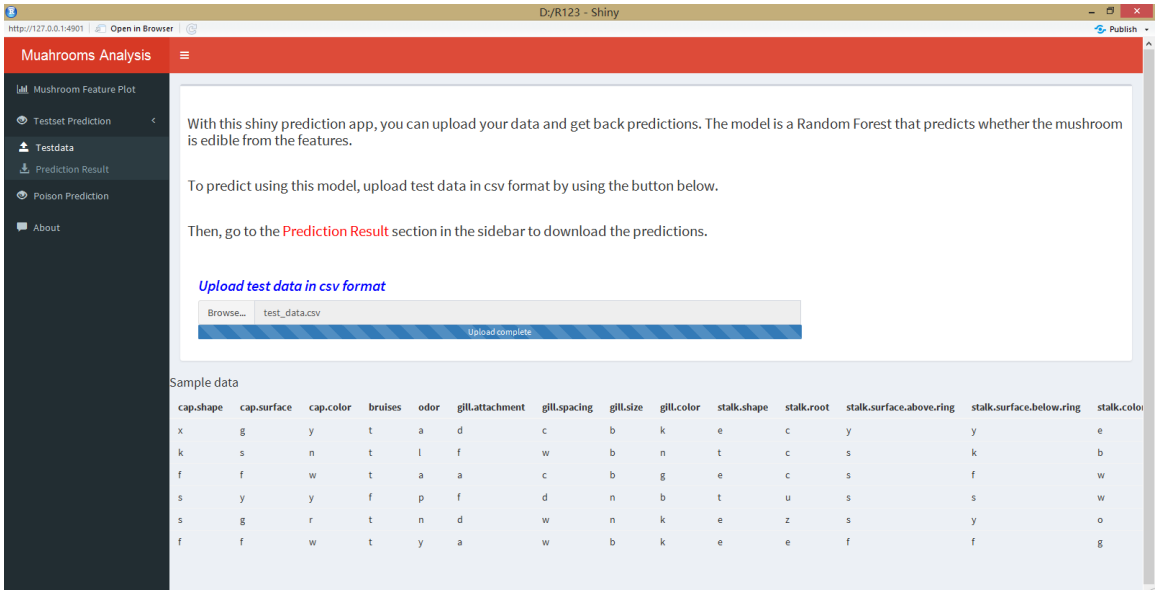
Deployment

We use package "shiny" to make our app. The app can be divided into three parts.

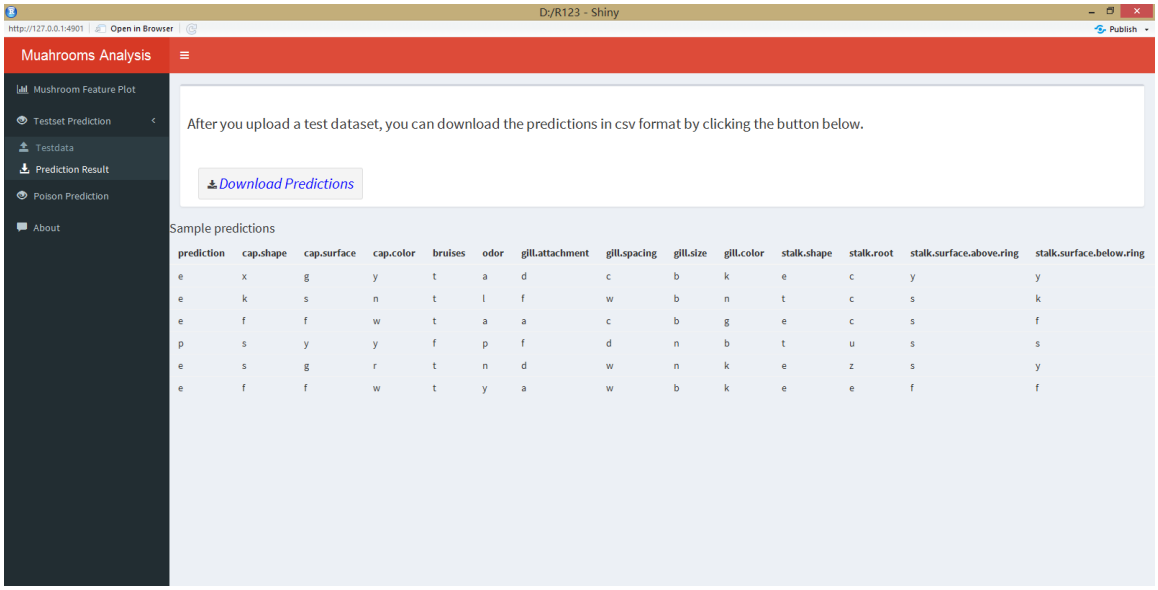
The first part is draw the plot of the distribution of each mushroom featrues, here we draw the barplot and pie in order to see the distribution intuitively.



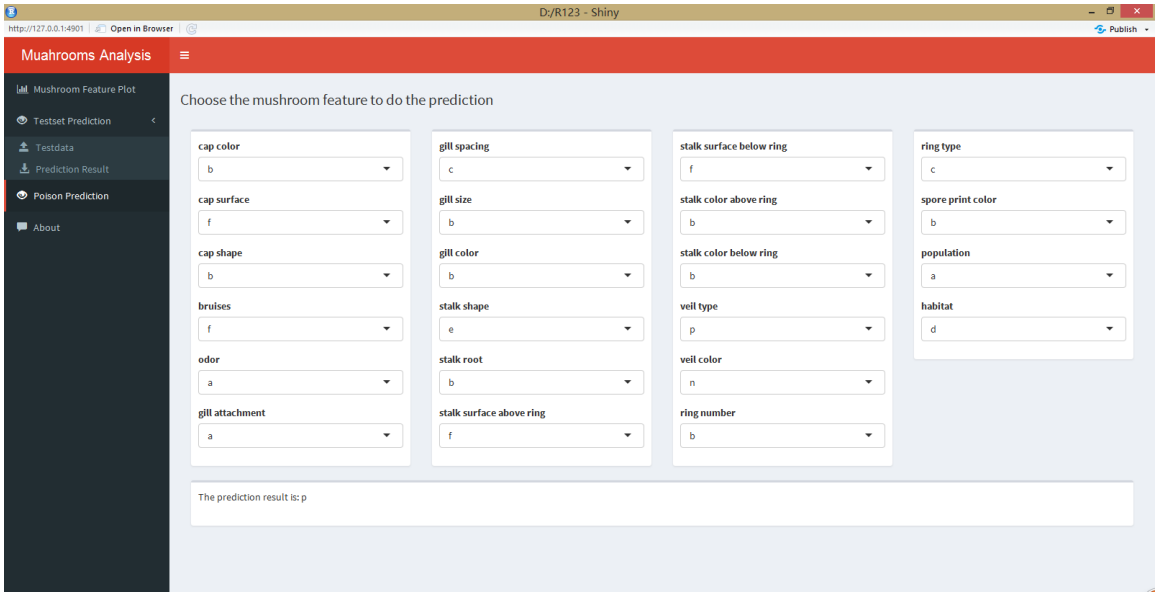
The second part is the display of test set prediction results. We could upload the csv file to see the result of the prediction. And we can down load the file which contains the result of prediction and the text set.



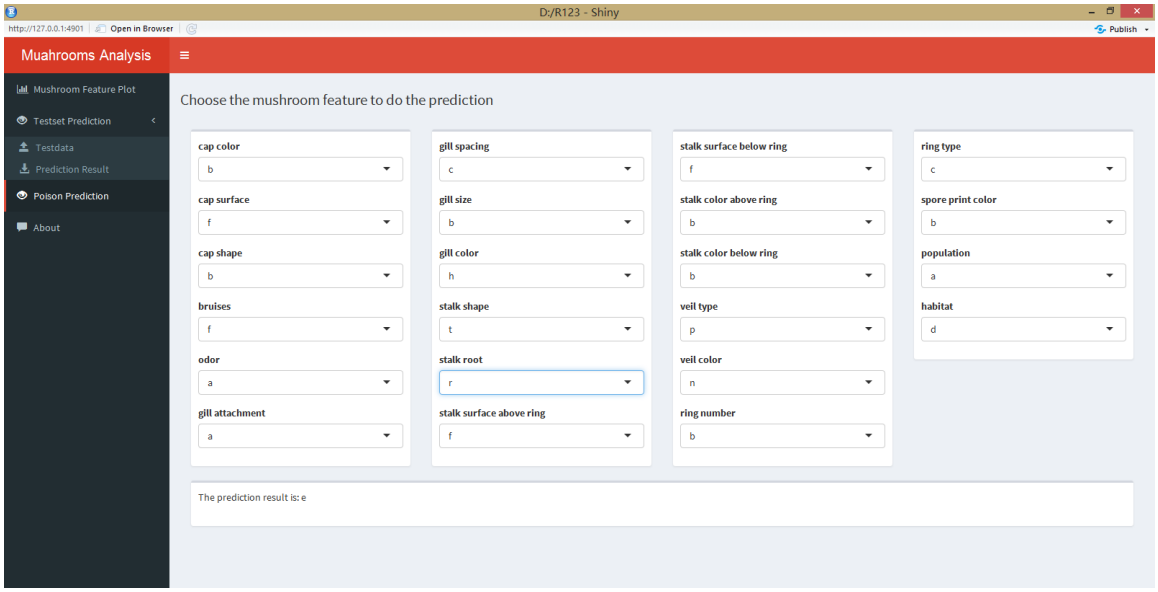
cap.shape	cap.surface	cap.color	bruises	odor	gill.attachment	gill.spacing	gill.size	gill.color	stalk.shape	stalk.root	stalk.surface.above.ring	stalk.surface.below.ring	stalk.color
x	g	y	t	a	d	c	b	k	e	c	y	y	e
k	s	n	t	l	f	w	b	n	t	c	s	k	b
f	f	w	t	a	a	c	b	g	e	c	s	f	w
s	y	y	f	p	f	d	n	b	t	u	s	s	w
s	g	r	t	n	d	w	n	k	e	z	s	y	o
f	f	w	t	y	a	w	b	k	e	e	f	f	g



The third part is the prediction part. We can input the features of mushrooms, and it will show the result due to the random forest model.



Change the input, we can get different result.


D:/R123 - Shiny
http://127.0.0.1:4901
Open in Browser
Publish
Muahrooms Analysis
Mushroom Feature Plot
Testset Prediction
Testdata
Prediction Result
Poison Prediction
About
Choose the mushroom feature to do the prediction
cap color
b
cap surface
f
cap shape
b
bruises
f
odor
a
gill attachment
a
gill spacing
c
gill size
b
gill color
h
stalk shape
t
stalk root
r
stalk surface above ring
f
stalk surface below ring
f
stalk color above ring
b
stalk color below ring
b
veil type
p
veil color
n
ring number
b
ring type
c
spore print color
b
population
a
habitat
d
The prediction result is: e