Mushroom Classification

Vivek Pruthi, Rajesh Grandhi and Jyothi Pulimamidi

July 12, 2017

Figure 1:

```
library(ggplot2)
library(caret)
library(ggthemes)
library(ipred)
library(ranger)
```

Importing the data

```
mushrooms_data<-read.csv("C:\\vik\\2017\\personal\\DSLA\\course material\\project 1 files\\mushrooms.cs
```

Exploring the data

Dimensions of the mushroom datasets are :

```
dim(mushrooms_data)
```

```
## [1] 8124   23
```

Fields in the dataset are:

```
names(mushrooms_data)
```

```
##  [1] "class"                    "cap.shape"
##  [3] "cap.surface"              "cap.color"
##  [5] "bruises"                  "odor"
```

```
##  [7] "gill.attachment"          "gill.spacing"
##  [9] "gill.size"                "gill.color"
## [11] "stalk.shape"              "stalk.root"
## [13] "stalk.surface.above.ring" "stalk.surface.below.ring"
## [15] "stalk.color.above.ring"   "stalk.color.below.ring"
## [17] "veil.type"                "veil.color"
## [19] "ring.number"              "ring.type"
## [21] "spore.print.color"        "population"
## [23] "habitat"
```

Following are the definitions of these fields:

- Fields/Attributes/features of the dataframe are
  - classes: edible=e, poisonous=p
  - cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
  - cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
  - cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r,pink=p,purple=u,red=e,white=w,yellow=y
  - bruises: bruises=t,no=f
  - odor: almond=a,anise=l,creosote=c,fishy=y,foul=f,musty=m,none=n,pungent=p,spicy=s
  - gill-attachment: attached=a,descending=d,free=f,notched=n
  - gill-spacing: close=c,crowded=w,distant=d
  - gill-size: broad=b,narrow=n
  - gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g,green=r,orange=o,pink=p,purple=u,red=e,white=w ,yellow=y
  - stalk-shape: enlarging=e,tapering=t
  - stalk-root: bulbous=b,club=c,cup=u,equal=e,rhizomorphs=z,rooted=r,missing=?
  - stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s
  - stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s
  - stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y
  - stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y
  - veil-type: partial=p,universal=u
  - veil-color: brown=n,orange=o,white=w,yellow=y
  - ring-number: none=n,one=o,two=t
  - ring-type: cobwebby=c,evanescent=e,flaring=f,large=l,none=n,pendant=p,sheathing=s,zone=z
  - spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r,orange=o,purple=u,white=w,yellow=y
  - population: abundant=a,clustered=c,numerous=n,scattered=s,several=v,solitary=y
  - habitat: grasses=g,leaves=l,meadows=m,paths=p,urban=u,waste=w,woods=d

Let's have a look at the structure of the dataset :

```r
str(mushrooms_data)
```

```
## '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",..: 6 6 1 6 6 6 1 1 6 1 ...
##  $ cap.surface             : Factor w/ 4 levels "f","g","s","y": 3 3 3 4 3 4 3 4 4 3 ...
##  $ cap.color               : Factor w/ 10 levels "b","c","e","g",..: 5 10 9 9 4 10 9 9 9 10 ...
##  $ bruises                 : Factor w/ 2 levels "f","t": 2 2 2 2 1 2 2 2 2 2 ...
##  $ odor                    : Factor w/ 9 levels "a","c","f","l",..: 7 1 4 7 6 1 1 4 7 1 ...
##  $ gill.attachment         : Factor w/ 2 levels "a","f": 2 2 2 2 2 2 2 2 2 2 ...
##  $ gill.spacing            : Factor w/ 2 levels "c","w": 1 1 1 1 2 1 1 1 1 1 ...
##  $ gill.size               : Factor w/ 2 levels "b","n": 2 1 1 2 1 1 1 1 2 1 ...
##  $ gill.color              : Factor w/ 12 levels "b","e","g","h",..: 5 5 6 6 5 6 3 6 8 3 ...
##  $ stalk.shape             : Factor w/ 2 levels "e","t": 1 1 1 1 2 1 1 1 1 1 ...
##  $ stalk.root              : Factor w/ 5 levels "?","b","c","e",..: 4 3 3 4 4 3 3 3 4 3 ...
##  $ stalk.surface.above.ring: Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.surface.below.ring: Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.color.above.ring  : Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ stalk.color.below.ring  : Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ veil.type               : Factor w/ 1 level "p": 1 1 1 1 1 1 1 1 1 1 ...
##  $ veil.color              : Factor w/ 4 levels "n","o","w","y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ ring.number             : Factor w/ 3 levels "n","o","t": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ring.type               : Factor w/ 5 levels "e","f","l","n",..: 5 5 5 5 1 5 5 5 5 5 ...
##  $ spore.print.color       : Factor w/ 9 levels "b","h","k","n",..: 3 4 4 3 4 3 3 4 3 3 ...
##  $ population              : Factor w/ 6 levels "a","c","n","s",..: 4 3 3 4 1 3 3 4 5 4 ...
##  $ habitat                 : Factor w/ 7 levels "d","g","l","m",..: 6 2 4 6 2 2 4 4 2 4 ...
```

It is good to have a little peek at a slice of data.

```r
head(mushrooms_data)
```

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

```
## 1                      w         p          w          o        p
## 2                      w         p          w          o        p
## 3                      w         p          w          o        p
## 4                      w         p          w          o        p
## 5                      w         p          w          o        e
## 6                      w         p          w          o        p
##   spore.print.color population habitat
## 1                 k          s       u
## 2                 n          n       g
## 3                 n          n       m
## 4                 k          s       u
## 5                 n          a       g
## 6                 k          n       g
```

```r
tail(mushrooms_data)
```

```
##      class cap.shape cap.surface cap.color bruises odor gill.attachment
## 8119     p         k           y         n       f    f               f
## 8120     e         k           s         n       f    n               a
## 8121     e         x           s         n       f    n               a
## 8122     e         f           s         n       f    n               a
## 8123     p         k           y         n       f    y               f
## 8124     e         x           s         n       f    n               a
##      gill.spacing gill.size gill.color stalk.shape stalk.root
## 8119            c         n          b           t          ?
## 8120            c         b          y           e          ?
## 8121            c         b          y           e          ?
## 8122            c         b          n           e          ?
## 8123            c         n          b           t          ?
## 8124            c         b          y           e          ?
##      stalk.surface.above.ring stalk.surface.below.ring
## 8119                        k                        s
## 8120                        s                        s
## 8121                        s                        s
## 8122                        s                        s
## 8123                        s                        k
## 8124                        s                        s
##      stalk.color.above.ring stalk.color.below.ring veil.type veil.color
## 8119                      p                      w         p          w
## 8120                      o                      o         p          o
## 8121                      o                      o         p          n
## 8122                      o                      o         p          o
## 8123                      w                      w         p          w
## 8124                      o                      o         p          o
##      ring.number ring.type spore.print.color population habitat
## 8119           o         e                 w          v       d
## 8120           o         p                 b          c       l
## 8121           o         p                 b          v       l
## 8122           o         p                 b          c       l
## 8123           o         e                 w          v       l
## 8124           o         p                 o          c       l
```

It is pertinent from the data that the fields in the dataset are of type factor i.e. these are catgorical variables with different levels. it is better to visualize this data . We will first check the summary and then explore the data visually :

```r
summary(mushrooms_data)
```

```
##  class    cap.shape cap.surface   cap.color    bruises       odor     
##  e:4208   b: 452    f:2320      n      :2284   f:4748   n      :3528  
##  p:3916   c:   4    g:   4      g      :1840   t:3376   f      :2160  
##           f:3152    s:2556      e      :1500            s      : 576  
##           k: 828    y:3244      y      :1072            y      : 576  
##           s:  32                w      :1040            a      : 400  
##           x:3656                b      : 168            l      : 400  
##                                 (Other): 220            (Other): 484  
##  gill.attachment gill.spacing gill.size   gill.color   stalk.shape
##  a: 210          c:6812       b:5612    b      :1728   e:3516     
##  f:7914          w:1312       n:2512    p      :1492   t:4608     
##                                         w      :1202              
##                                         n      :1048              
##                                         g      : 752              
##                                         h      : 732              
##                                         (Other):1170              
##  stalk.root stalk.surface.above.ring stalk.surface.below.ring
##  ?:2480     f: 552                   f: 600                  
##  b:3776     k:2372                   k:2304                  
##  c: 556     s:5176                   s:4936                  
##  e:1120     y:  24                   y: 284                  
##  r: 192                                                      
##                                                              
##                                                              
##  stalk.color.above.ring stalk.color.below.ring veil.type veil.color
##  w      :4464           w      :4384           p:8124    n:  96    
##  p      :1872           p      :1872                     o:  96    
##  g      : 576           g      : 576                     w:7924    
##  n      : 448           n      : 512                     y:   8    
##  b      : 432           b      : 432                              
##  o      : 192           o      : 192                              
##  (Other): 140           (Other): 156                              
##  ring.number ring.type spore.print.color population habitat 
##  n:  36      e:2776    w      :2388      a: 384     d:3148  
##  o:7488      f:  48    n      :1968      c: 340     g:2148  
##  t: 600      l:1296    k      :1872      n: 400     l: 832  
##              n:  36    h      :1632      s:1248     m: 292  
##              p:3968    r      :  72      v:4040     p:1144  
##                        b      :  48      y:1712     u: 368  
##                        (Other): 144                 w: 192  
```

As few of the levels are shown in summary as (others), let's check what the complete levels are of all the categorical variables in this dataset :

```r
for(i in 1:23){
  print(names(mushrooms_data[i]))
  print(levels(mushrooms_data[,i]))
}
```

```
## [1] "class"
## [1] "e" "p"
## [1] "cap.shape"
## [1] "b" "c" "f" "k" "s" "x"
```

```
## [1] "cap.surface"
## [1] "f" "g" "s" "y"
## [1] "cap.color"
##  [1] "b" "c" "e" "g" "n" "p" "r" "u" "w" "y"
## [1] "bruises"
## [1] "f" "t"
## [1] "odor"
## [1] "a" "c" "f" "l" "m" "n" "p" "s" "y"
## [1] "gill.attachment"
## [1] "a" "f"
## [1] "gill.spacing"
## [1] "c" "w"
## [1] "gill.size"
## [1] "b" "n"
## [1] "gill.color"
##  [1] "b" "e" "g" "h" "k" "n" "o" "p" "r" "u" "w" "y"
## [1] "stalk.shape"
## [1] "e" "t"
## [1] "stalk.root"
## [1] "?" "b" "c" "e" "r"
## [1] "stalk.surface.above.ring"
## [1] "f" "k" "s" "y"
## [1] "stalk.surface.below.ring"
## [1] "f" "k" "s" "y"
## [1] "stalk.color.above.ring"
## [1] "b" "c" "e" "g" "n" "o" "p" "w" "y"
## [1] "stalk.color.below.ring"
## [1] "b" "c" "e" "g" "n" "o" "p" "w" "y"
## [1] "veil.type"
## [1] "p"
## [1] "veil.color"
## [1] "n" "o" "w" "y"
## [1] "ring.number"
## [1] "n" "o" "t"
## [1] "ring.type"
## [1] "e" "f" "l" "n" "p"
## [1] "spore.print.color"
## [1] "b" "h" "k" "n" "o" "r" "u" "w" "y"
## [1] "population"
## [1] "a" "c" "n" "s" "v" "y"
## [1] "habitat"
## [1] "d" "g" "l" "m" "p" "u" "w"
```

We can check their proportionate distribution too:

```
for(i in 1:23){
  print(names(mushrooms_data[i]))
 print(prop.table((table(mushrooms_data[,i])))*100)
}
```

```
## [1] "class"
##
##        e        p
## 51.79714 48.20286
## [1] "cap.shape"
```

```
##
##           b           c           f           k           s           x
##  5.56376169  0.04923683 38.79862137 10.19202363  0.39389463 45.00246184
## [1] "cap.surface"
##
##           f           g           s           y
## 28.55736091  0.04923683 31.46233383 39.93106844
## [1] "cap.color"
##
##          b          c          e          g          n          p
##  2.0679468  0.5416051 18.4638109 22.6489414 28.1142294  1.7725258
##          r          u          w          y
##  0.1969473  0.1969473 12.8015756 13.1954702
## [1] "bruises"
##
##        f        t
## 58.44412 41.55588
## [1] "odor"
##
##          a          c          f          l          m          n
##  4.9236829  2.3633678 26.5878877  4.9236829  0.4431315 43.4268833
##          p          s          y
##  3.1511571  7.0901034  7.0901034
## [1] "gill.attachment"
##
##         a         f
##  2.584934 97.415066
## [1] "gill.spacing"
##
##        c        w
## 83.85032 16.14968
## [1] "gill.size"
##
##        b        n
## 69.07927 30.92073
## [1] "gill.color"
##
##          b          e          g          h          k          n
## 21.2703102  1.1816839  9.2565239  9.0103397  5.0221566 12.9000492
##          o          p          r          u          w          y
##  0.7877893 18.3653373  0.2954210  6.0561300 14.7956672  1.0585918
## [1] "stalk.shape"
##
##        e        t
## 43.27917 56.72083
## [1] "stalk.root"
##
##         ?         b         c         e         r
## 30.526834 46.479567  6.843919 13.786312  2.363368
## [1] "stalk.surface.above.ring"
##
##         f         k         s         y
##  6.794682 29.197440 63.712457  0.295421
## [1] "stalk.surface.below.ring"
```

```
##
##         f         k         s         y
##  7.385524 28.360414 60.758247  3.495815
## [1] "stalk.color.above.ring"
##
##           b           c           e           g           n           o
##  5.31757755  0.44313146  1.18168390  7.09010340  5.51452486  2.36336780
##           p           w           y
## 23.04283604 54.94830133  0.09847366
## [1] "stalk.color.below.ring"
##
##          b          c          e          g          n          o
##  5.3175775  0.4431315  1.1816839  7.0901034  6.3023141  2.3633678
##          p          w          y
## 23.0428360 53.9635647  0.2954210
## [1] "veil.type"
##
##   p
## 100
## [1] "veil.color"
##
##           n           o           w           y
##  1.18168390  1.18168390 97.53815854  0.09847366
## [1] "ring.number"
##
##          n          o          t
##  0.4431315 92.1713442  7.3855244
## [1] "ring.type"
##
##          e          f          l          n          p
## 34.1703594  0.5908419 15.9527326  0.4431315 48.8429345
## [1] "spore.print.color"
##
##          b          h          k          n          o          r
##  0.5908419 20.0886263 23.0428360 24.2245199  0.5908419  0.8862629
##          u          w          y
##  0.5908419 29.3943870  0.5908419
## [1] "population"
##
##         a         c         n         s         v         y
##  4.726736  4.185130  4.923683 15.361891 49.729197 21.073363
## [1] "habitat"
##
##         d         g         l         m         p         u         w
## 38.749385 26.440177 10.241260  3.594289 14.081733  4.529788  2.363368
```

```r
library(ggplot2)
library(gridExtra)
 p1<-ggplot(mushrooms_data,aes(x=class))+geom_histogram(stat="count",fill="blue")+ggtitle(label="Poisono
 p2<-ggplot(mushrooms_data,aes(x=cap.shape))+geom_histogram(stat="count",aes(fill=class))+ggtitle(label=
 p3<-ggplot(mushrooms_data,aes(x=cap.surface))+geom_histogram(stat="count",aes(fill=class))+ggtitle(labe
 p4<-ggplot(mushrooms_data,aes(x=cap.color))+geom_histogram(stat="count",aes(fill=class))+ggtitle(label=
 p5<-ggplot(mushrooms_data,aes(x=bruises))+geom_histogram(stat="count",aes(fill=class))+ggtitle(label="
 p6<-ggplot(mushrooms_data,aes(x=odor))+geom_histogram(stat="count",aes(fill=class))+ggtitle(label="odo
```

```
p7<-ggplot(mushrooms_data,aes(x=gill.attachment))+geom_histogram(stat="count",aes(fill=class))+ggtitle
p8<-ggplot(mushrooms_data,aes(x=gill.spacing))+geom_histogram(stat="count",aes(fill=class))+ggtitle(lab
p9<-ggplot(mushrooms_data,aes(x=gill.size))+geom_histogram(stat="count",aes(fill=class))+ggtitle(label=
p10<-ggplot(mushrooms_data,aes(x=gill.color))+geom_histogram(stat="count",aes(fill=class))+ggtitle(labe
p11<-ggplot(mushrooms_data,aes(x=stalk.shape))+geom_histogram(stat="count",aes(fill=class))+ggtitle(lab
p12<-ggplot(mushrooms_data,aes(x=stalk.root))+geom_histogram(stat="count",aes(fill=class))+ggtitle(labe
p13<-ggplot(mushrooms_data,aes(x=stalk.surface.above.ring))+geom_histogram(stat="count",aes(fill=class)
p14<-ggplot(mushrooms_data,aes(x=stalk.surface.below.ring))+geom_histogram(stat="count",aes(fill=class)
p15<-ggplot(mushrooms_data,aes(x=stalk.color.above.ring))+geom_histogram(stat="count",aes(fill=class))+
p16<-ggplot(mushrooms_data,aes(x=stalk.color.below.ring))+geom_histogram(stat="count",aes(fill=class))+
p17<-ggplot(mushrooms_data,aes(x=veil.type))+geom_histogram(stat="count",aes(fill=class))+ggtitle(label
p18<-ggplot(mushrooms_data,aes(x=veil.color))+geom_histogram(stat="count",aes(fill=class))+ggtitle(labe
p19<-ggplot(mushrooms_data,aes(x=ring.number))+geom_histogram(stat="count",aes(fill=class))+ggtitle(lab
p20<-ggplot(mushrooms_data,aes(x=ring.type))+geom_histogram(stat="count",aes(fill=class))+ggtitle(label
p21<-ggplot(mushrooms_data,aes(x=spore.print.color))+geom_histogram(stat="count",aes(fill=class))+ggtit
p22<-ggplot(mushrooms_data,aes(x=population))+geom_histogram(stat="count",aes(fill=class))+ggtitle(labe
p23<-ggplot(mushrooms_data,aes(x=habitat))+geom_histogram(stat="count",aes(fill=class))+ggtitle(label=
grid.arrange(p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14,p15,p16,p17,p18,p19,p20,p21,p22,p23,ncol=2)
```


Poisonous or Edible
count
class
cap shape
cap.shape
cap surface
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.type
veil.color
ring.number
ring.type
spore.print.color
population
habitat
class
e
p

We can make the hunches based on the exploratory analysis , but will confirm the huncheas based on the model that we select for machine learning.

Machine Learning

We will follow following steps to decide about the classification model:

1. split the data in train set and test set
2. train the model on the train set
3. check the efficiency of the model on the train set
4. predict the classification of the test set data
5. check the efficiency of the model on the test set

We will iterate these steps for different models and then compare the efficiencies of different models to choose the best model.

Defining split factor

First of all we will define a splitting factor which will be used to split data between train and test set . As it is better to train the model on bigger data set and test on small dataset, we will use a variable to accomodate that thought. Thought behind defining the split factor is to check the effect of the size of training set on the efficiency of the model.

```
mushroom_split_factor<-0.8
```

We will now define the train and test sets:

```
set.seed(1)
mushrooms_split_index<-createDataPartition(mushrooms_data$class,p = mushroom_split_factor,list = FALSE)
mushrooms_trainset<-mushrooms_data[mushrooms_split_index,]
mushrooms_testset<-mushrooms_data[-mushrooms_split_index,]
```

We will now check the dimensions of mashromm dataset, mushroom_trainset and mushroom_testset to make sure that split is fine.

```
dim(mushrooms_data)
```

```
## [1] 8124   23
```

```
dim(mushrooms_testset)
```

```
## [1] 1624   23
```

```
dim(mushrooms_trainset)
```

```
## [1] 6500   23
```

As this is a classification problem. I intend to use rpart,Classification decision trees, bagging , Random Forest and boosting models and then compare the results.We will load the requisite packages here:

```
library(rpart)
library(rpart.plot)
library(caret)
```

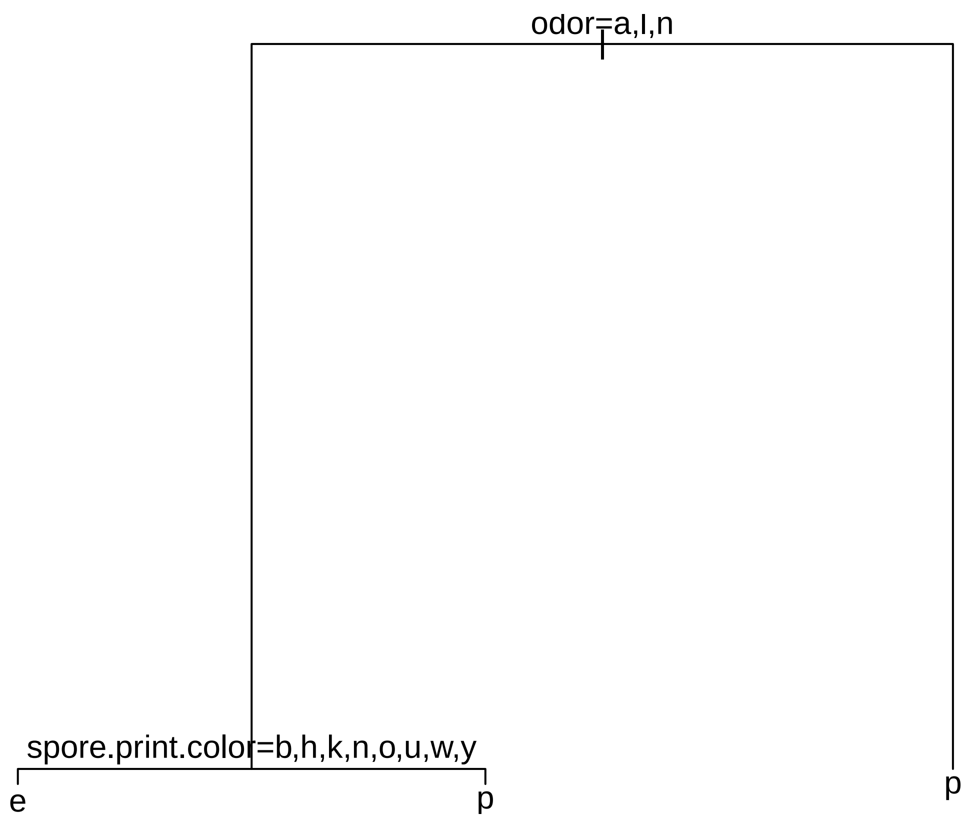
1. **Model I : rpart**

We will use the same trainset and testset defined earlier for different models . we will train the model on trainset,plot the model, predict for trainset ,calculate the efficiency of model on trainset ,predict for testset , calculate the efficiency for testset and then compare the change in efficiency from train to testset , which will give us an idea about underfitting or overfitting .

```
mushrooms_mdl_rpart<-rpart(class~.,mushrooms_trainset,method = "class")
```

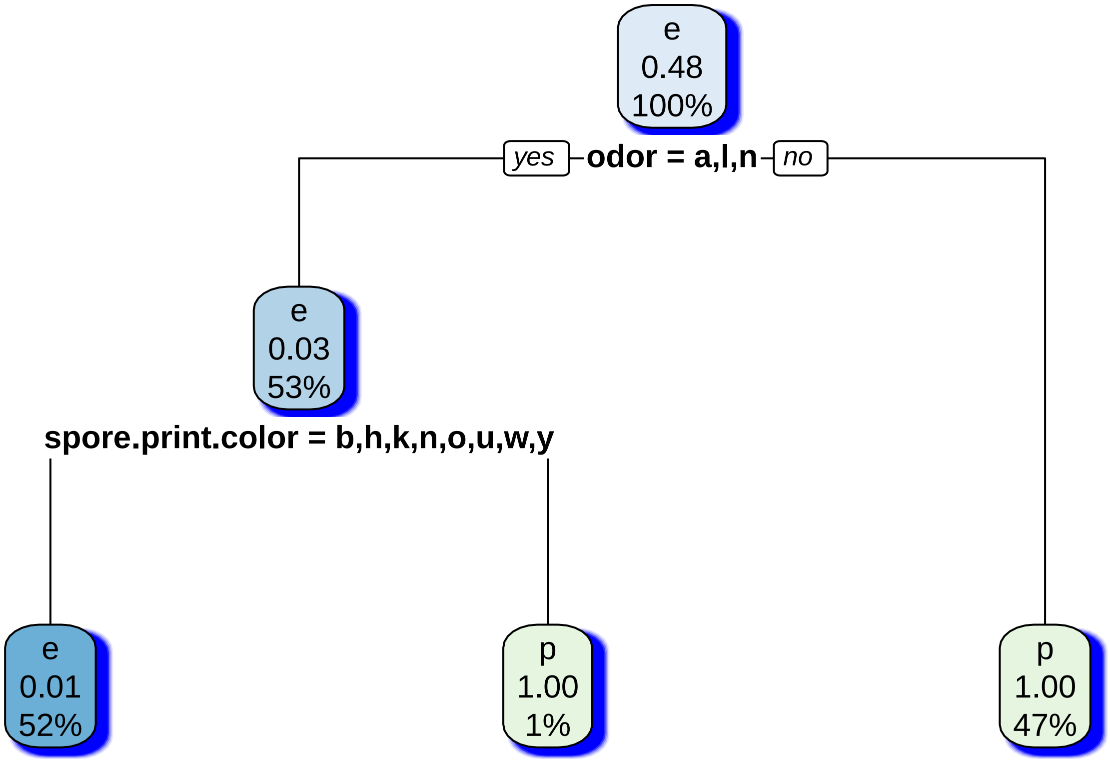
We will plot this model to get an insight now:

```
plot(mushrooms_mdl_rpart)
text(mushrooms_mdl_rpart,pretty = 0)
```



Let's look into a little better version of it :

```
rpart.plot(mushrooms_mdl_rpart,shadow.col = "blue")
```



Predictions for trainset :

```
mushroom_pred_rpart_train<-predict(mushrooms_mdl_rpart,mushrooms_trainset,type = "class")
```

let's look at the consolidated predictions:

```
table(mushroom_pred_rpart_train)
```

```
## mushroom_pred_rpart_train
##    e    p
## 3407 3093
```

To check for the accuracy for trainset :

```
confusionMatrix(mushroom_pred_rpart_train,mushrooms_trainset$class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    e    p
##          e 3367   40
##          p    0 3093
##
##                Accuracy : 0.9938
##                  95% CI : (0.9916, 0.9956)
##     No Information Rate : 0.518
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9877
```

```
##  Mcnemar's Test P-Value : 6.984e-10
##
##             Sensitivity : 1.0000
##             Specificity : 0.9872
##          Pos Pred Value : 0.9883
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5180
##          Detection Rate : 0.5180
##    Detection Prevalence : 0.5242
##       Balanced Accuracy : 0.9936
##
##        'Positive' Class : e
##
```

Let's look at the predictions on the test set and check the accuracy there :

```r
mushroom_pred_rpart_test<-predict(mushrooms_mdl_rpart,mushrooms_testset,type="class")
table(mushroom_pred_rpart_test)
```

```
## mushroom_pred_rpart_test
##   e   p
## 849 775
```

```r
confusionMatrix(mushroom_pred_rpart_test,mushrooms_testset$class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   e   p
##          e 841   8
##          p   0 775
##
##                Accuracy : 0.9951
##                  95% CI : (0.9903, 0.9979)
##     No Information Rate : 0.5179
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9901
##  Mcnemar's Test P-Value : 0.01333
##
##             Sensitivity : 1.0000
##             Specificity : 0.9898
##          Pos Pred Value : 0.9906
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5179
##          Detection Rate : 0.5179
##    Detection Prevalence : 0.5228
##       Balanced Accuracy : 0.9949
##
##        'Positive' Class : e
##
```

As we see that the accuracy has increased from 99.38% to 99.51% from trainset to testset, which means our model has performed better for unseen data , but still the acceptance of the model depends upon what is the threshold above which, you will accept.

---

2. Model II : Decision Trees and Pruning :

```r
library(tree)
```

Model:

```r
mushroom_mdl_tree<-tree(class~.,mushrooms_testset)
```

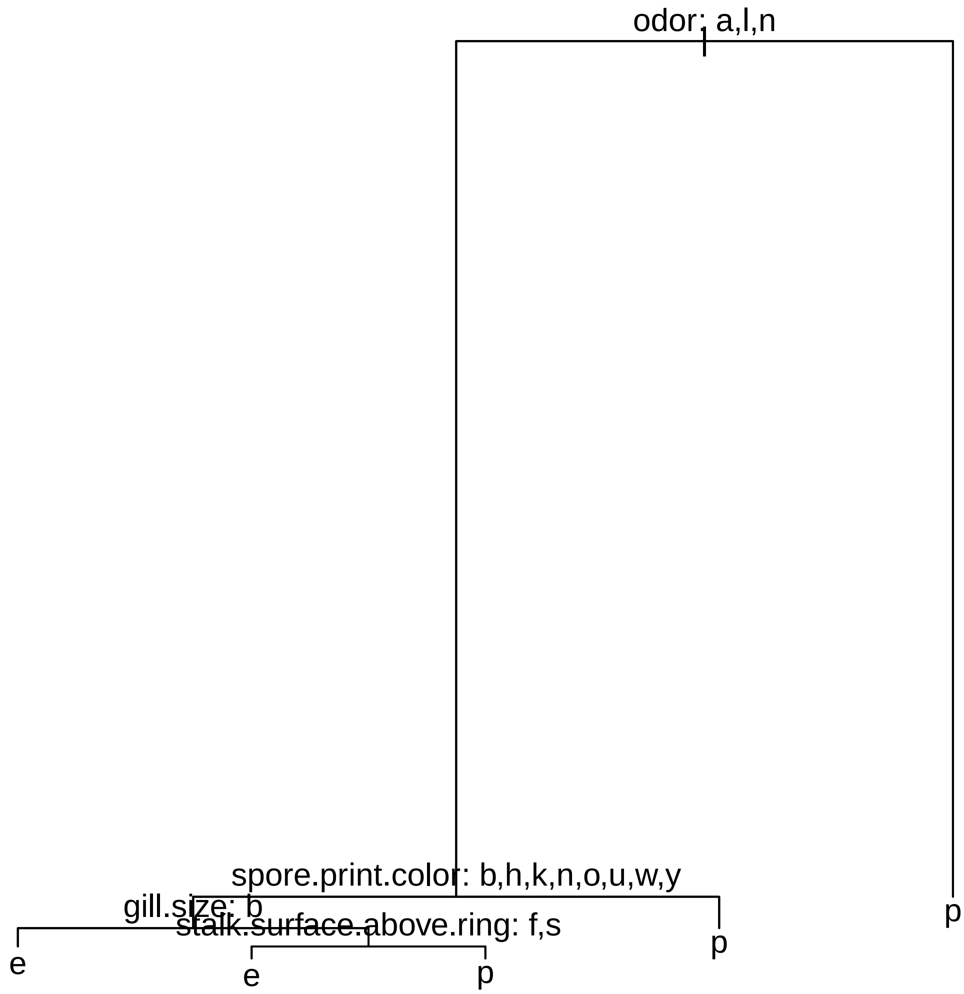
summary of the model :

```r
summary(mushroom_mdl_tree)
```

```
##
## Classification tree:
## tree(formula = class ~ ., data = mushrooms_testset)
## Variables actually used in tree construction:
## [1] "odor"                      "spore.print.color"
## [3] "gill.size"                 "stalk.surface.above.ring"
## Number of terminal nodes:  5
## Residual mean deviance:  0.01037 = 16.79 / 1619
## Misclassification error rate: 0.001232 = 2 / 1624
```

Plotting the decision Tree:

```r
plot(mushroom_mdl_tree)
text(mushroom_mdl_tree,pretty=0)
```



A look at the tree in text :

```
mushroom_mdl_tree
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 1624 2249.00 e ( 0.517857 0.482143 )
##    2) odor: a,l,n 858  167.00 e ( 0.980186 0.019814 )
##      4) spore.print.color: b,h,k,n,o,u,w,y 849   90.56 e ( 0.990577 0.009423 )
##        8) gill.size: b 793    0.00 e ( 1.000000 0.000000 ) *
##        9) gill.size: n 56   45.93 e ( 0.857143 0.142857 )
```

```
##         18) stalk.surface.above.ring: f,s 50   16.79 e ( 0.960000 0.040000 ) *
##         19) stalk.surface.above.ring: k 6    0.00 p ( 0.000000 1.000000 ) *
##      5) spore.print.color: r 9    0.00 p ( 0.000000 1.000000 ) *
##    3) odor: c,f,m,p,s,y 766    0.00 p ( 0.000000 1.000000 ) *
```

Prediction for training set and the evaluation of efficiency of model on training set :

```
mushroom_pred_tree_train<-predict(mushroom_mdl_tree,mushrooms_trainset,type="class")
mushroom_tree_train_perf<-table(mushroom_pred_tree_train,mushrooms_trainset$class)
mushroom_tree_train_perf
```

```
##
## mushroom_pred_tree_train    e    p
##                        e 3367   14
##                        p    0 3119
```

```
sum(diag(mushroom_tree_train_perf))/sum(mushroom_tree_train_perf)
```

```
## [1] 0.9978462
```

Prediction for test set and the evaluation of efficiency of model on test set :

```
mushroom_pred_tree_test<-predict(mushroom_mdl_tree,mushrooms_testset,type="class")
mushroom_tree_test_perf<-table(mushroom_pred_tree_test,mushrooms_testset$class)
mushroom_tree_test_perf
```

```
##
## mushroom_pred_tree_test   e   p
##                       e 841   2
##                       p   0 781
```

```
sum(diag(mushroom_tree_test_perf))/sum(mushroom_tree_test_perf)
```

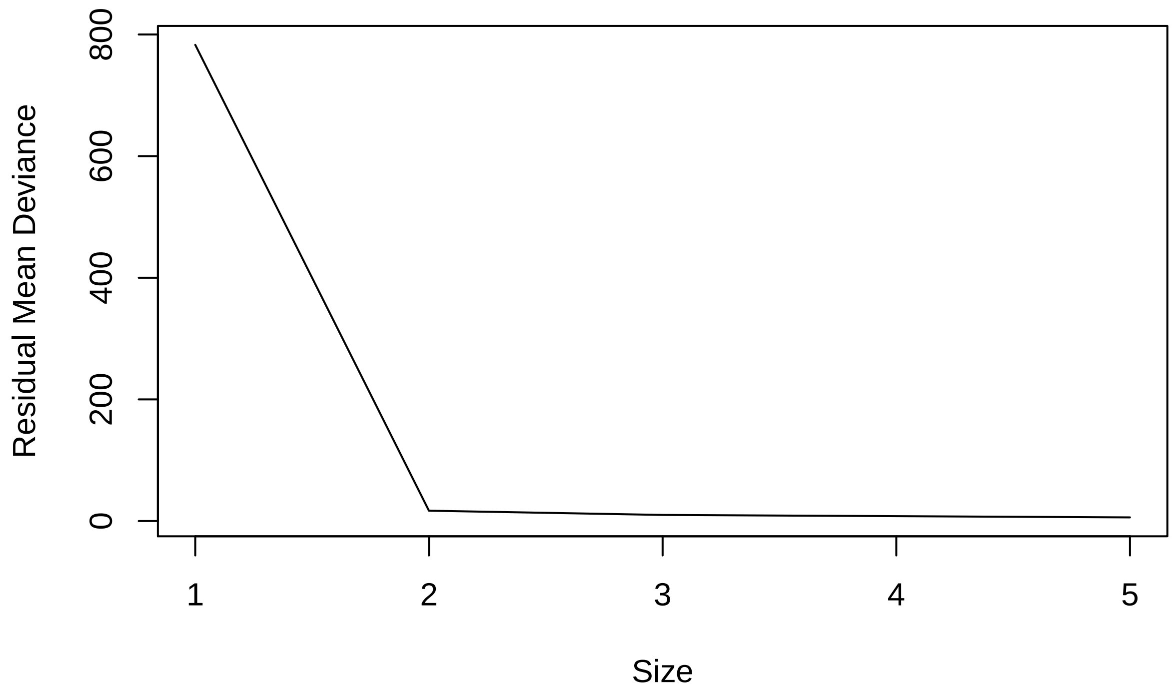
```
## [1] 0.9987685
```

In this model also , model performed better with test data than the training data.To find the optimal level of tree complexity, we can use cost complexity pruning in order to select sequence of trees. We do this by using cross validation. It will help us identify the size of tree that will have minimum residual mean davience.

```
set.seed(1)
mushroom_mdl_tree_cv<-cv.tree(mushroom_mdl_tree,FUN = prune.misclass)
mushroom_mdl_tree_cv
```

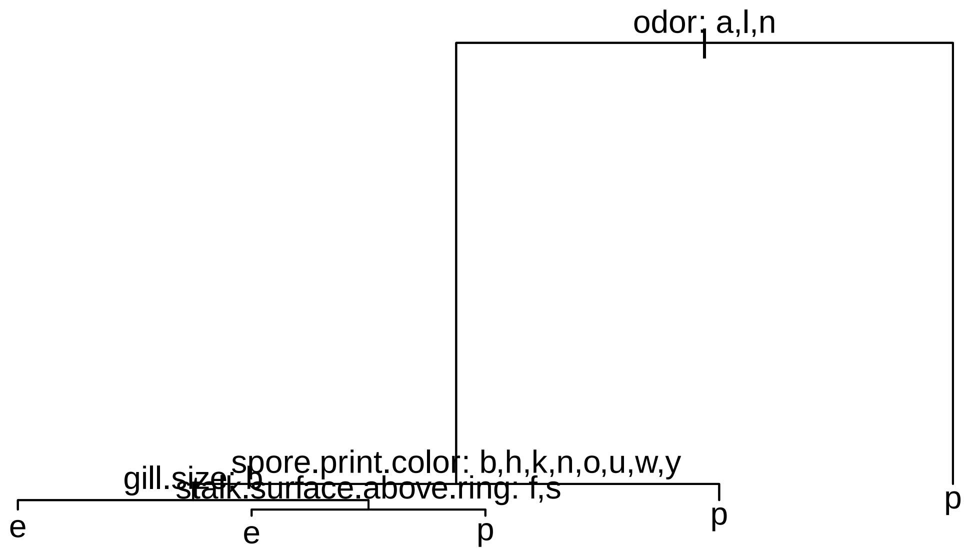
```
## $size
## [1] 5 3 2 1
##
## $dev
## [1]   6  10  17 783
##
## $k
## [1] -Inf    3    9  766
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
plot(mushroom_mdl_tree_cv$size,mushroom_mdl_tree_cv$dev,type = "l",xlab = "Size",ylab="Residual Mean De
```



we can create a pruned tree for the optimum size 5 as:

```
mushroom_mdl_tree_prune<-prune.misclass(mushroom_mdl_tree,best=5)
plot(mushroom_mdl_tree_prune)
text(mushroom_mdl_tree_prune,pretty=0)
```




```
mushroom_pred_tree_train_prune<-predict(mushroom_mdl_tree_prune,mushrooms_trainset,type="class")
mushroom_pred_tree_test_prune<- predict(mushroom_mdl_tree_prune,mushrooms_testset,type="class")
mush_prn_train_perftab<-table(mushroom_pred_tree_train_prune,mushrooms_trainset$class)
mush_prn_test_perftab<-table(mushroom_pred_tree_test_prune,mushrooms_testset$class)
```

Performance of the pruned tree on trainset :

```
sum(diag(mush_prn_train_perftab))/sum(mush_prn_train_perftab)
```

```
## [1] 0.9978462
```

Performance of the pruned tree on testset :

```
sum(diag(mush_prn_test_perftab))/sum(mush_prn_test_perftab)
```

```
## [1] 0.9987685
```

In fact the tree that we created before pruning was optimum already as it had the 5 terminal nodes as were concluded from cross validation.

---

3. Model III : Bagging

Next Model that we will consider .Here we would try to create trees taking all variables into account while creating multiple trees and then using their average as the final result.first we will load the requisite package :

we will now create the model bagging the trees taking into account all the variables i.e. all the predictors should be considered for each split of the tree(minus the dependent variable):

```r
set.seed(1)
mushroom_mdl_bagging<-randomForest(class~.,data=mushrooms_trainset,mtry=22,importance=TRUE)
```

Let's take a look at the bagged tree model:

```r
mushroom_mdl_bagging
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = mushrooms_trainset,      mtry = 22, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 22
##
##         OOB estimate of  error rate: 0.02%
## Confusion matrix:
##      e    p  class.error
## e 3367    0 0.0000000000
## p    1 3132 0.0003191829
```

We would do the predictions for train and test dataset now and check the performance accuracy of the model.We could have used MSE , if the data would have been numeric to test the accuracy of model , but in this case we will use the confusionMatrix to check the efficiency of the model:

```r
mushroom_pred_bag_train<-predict(mushroom_mdl_bagging,mushrooms_trainset)
mushroom_pred_bag_train_tbl<-table(mushroom_pred_bag_train,mushrooms_trainset$class)
mushroom_pred_bag_train_tbl
```

```
##
## mushroom_pred_bag_train    e    p
##                       e 3367    0
##                       p    0 3133
```

so the accuracy of the model for the training set is :

```r
(sum(diag(mushroom_pred_bag_train_tbl))/sum(mushroom_pred_bag_train_tbl))*100
```

```
## [1] 100
```

As the model on the trainset may be overfitted to give 100% accuract, let's try this on testset:

```r
mushroom_pred_bag_test<-predict(mushroom_mdl_bagging,mushrooms_testset)
mushroom_pred_bag_test_tbl<-table(mushroom_pred_bag_test,mushrooms_testset$class)
mushroom_pred_bag_test_tbl
```

```
##
## mushroom_pred_bag_test   e   p
##                      e 841   0
##                      p   0 783
```

so the accuracy of the model for the test set is :

```r
(sum(diag(mushroom_pred_bag_test_tbl))/sum(mushroom_pred_bag_test_tbl))*100
```

```
## [1] 100
```

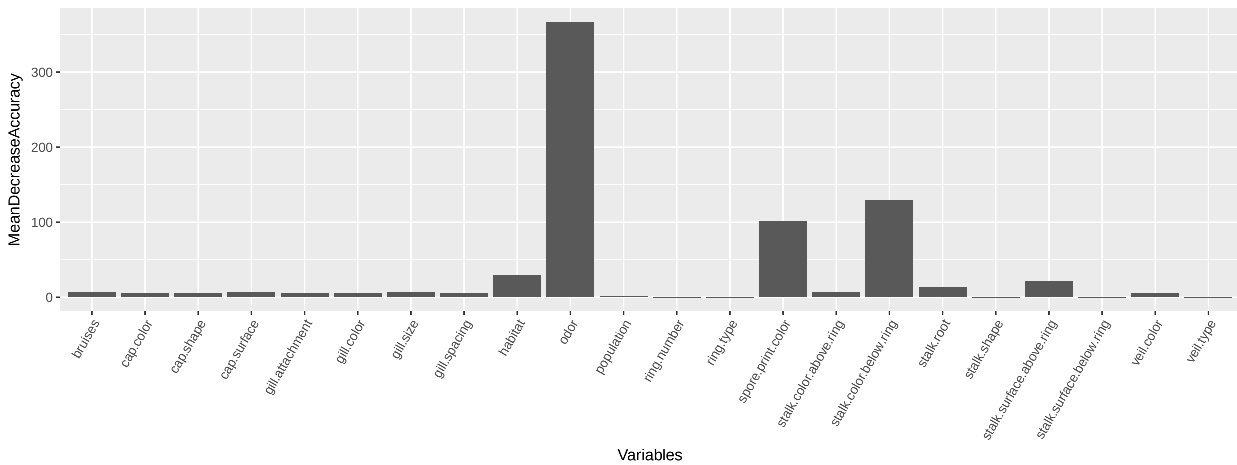
so , we can clearly see that bagging has improved the accuracy of the model.

Let us check at the importance of these variables in this model

```r
mushroom_imp_bagging<-importance(mushroom_mdl_bagging)
mushroom_imp_bagging
```

```
##                                  e           p MeanDecreaseAccuracy
## cap.shape                 4.896537  -0.5887669             4.909002
## cap.surface               6.664476   7.9949604             6.901242
## cap.color                 5.906325   3.1514921             5.904314
## bruises                   6.468706   3.6529900             6.475842
## odor                    817.423374 164.0282639           366.850539
## gill.attachment           5.919623   0.0000000             5.917669
## gill.spacing              5.578414   3.7414657             5.636164
## gill.size                 7.171052   4.9527073             7.209767
## gill.color                5.907561   2.0077569             5.908320
## stalk.shape               0.000000   0.0000000             0.000000
## stalk.root               11.655025  18.9036291            13.857407
## stalk.surface.above.ring 19.566633  19.5185920            21.089021
## stalk.surface.below.ring  0.000000   0.0000000             0.000000
## stalk.color.above.ring    6.106820   1.4169494             6.105548
## stalk.color.below.ring  118.554154  77.5745890           129.701302
## veil.type                 0.000000   0.0000000             0.000000
## veil.color                5.521455   0.0000000             5.519290
## ring.number               0.000000   0.0000000             0.000000
## ring.type                 0.000000   0.0000000             0.000000
## spore.print.color        51.222850 140.7635297           101.475489
## population                1.001002   1.0010015             1.001002
## habitat                  29.438197  19.0325768            29.593294
##                         MeanDecreaseGini
## cap.shape                   2.677097e+00
## cap.surface                 5.811078e+00
## cap.color                   3.103987e-01
## bruises                     3.691480e-01
## odor                        3.044693e+03
## gill.attachment             3.061402e-01
## gill.spacing                2.600468e-01
## gill.size                   3.597446e-01
## gill.color                  2.943350e-01
## stalk.shape                 0.000000e+00
## stalk.root                  1.158332e+01
## stalk.surface.above.ring    1.085451e+01
## stalk.surface.below.ring    0.000000e+00
## stalk.color.above.ring      2.758827e-01
## stalk.color.below.ring      4.505889e+01
## veil.type                   0.000000e+00
## veil.color                  2.593538e-01
## ring.number                 0.000000e+00
## ring.type                   0.000000e+00
## spore.print.color           1.220579e+02
## population                  1.566138e-02
## habitat                     1.116141e-01
```

```r
mushroom_imp_baggingdf<-as.data.frame(unlist(mushroom_imp_bagging))
ggplot(mushroom_imp_baggingdf,aes(x=row.names(mushroom_imp_baggingdf),y=MeanDecreaseAccuracy))+geom_bar
```



We can clearly see that the odor,stalk.colorbelow.ring and sport.printcolor are the top 3 variables in the bagged model.

---

4. Model IV : randomForest

This model allows random number of variables to be considered at each split unlike the bagging. By default in classification , number of variables considered are sqrt(total no. of variables) i.e for us , it is roundup(sqrt(23))=5

```
mushroom_mdl_ranforest<-randomForest(class~.,data=mushrooms_trainset,mtry=5,importance=TRUE,ntree=500)

mushroom_mdl_ranforest
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = mushrooms_trainset,      mtry = 5, importance = TRUE, ntre
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 0%
## Confusion matrix:
##      e    p class.error
## e 3367    0           0
## p    0 3133           0
```

let's check the predictions and accuracy on testset:

```
mushroom_pred_ranforest_train<-predict(mushroom_mdl_ranforest,mushrooms_trainset)
mushroom_pred_ranforest_traintbl<-table(mushroom_pred_ranforest_train,mushrooms_trainset$class)
```

Accuracy of the model is :

```
(sum(diag(mushroom_pred_ranforest_traintbl))/sum(mushroom_pred_ranforest_traintbl))*100
```

```
## [1] 100
```

Let's do the predictions for the testset and find the accuracy:

```r
mushroom_pred_ranforest_test<-predict(mushroom_mdl_ranforest,mushrooms_testset)
mushroom_pred_ranforest_testtbl<-table(mushroom_pred_ranforest_test,mushrooms_testset$class)
mushroom_pred_ranforest_testtbl
```

```
##
## mushroom_pred_ranforest_test   e   p
##                            e 841   0
##                            p   0 783
```

Accuracy of the testset is :

```r
(sum(diag(mushroom_pred_ranforest_testtbl))/sum(mushroom_pred_ranforest_testtbl))*100
```

```
## [1] 100
```

Let's check this model give what importance to which variable:

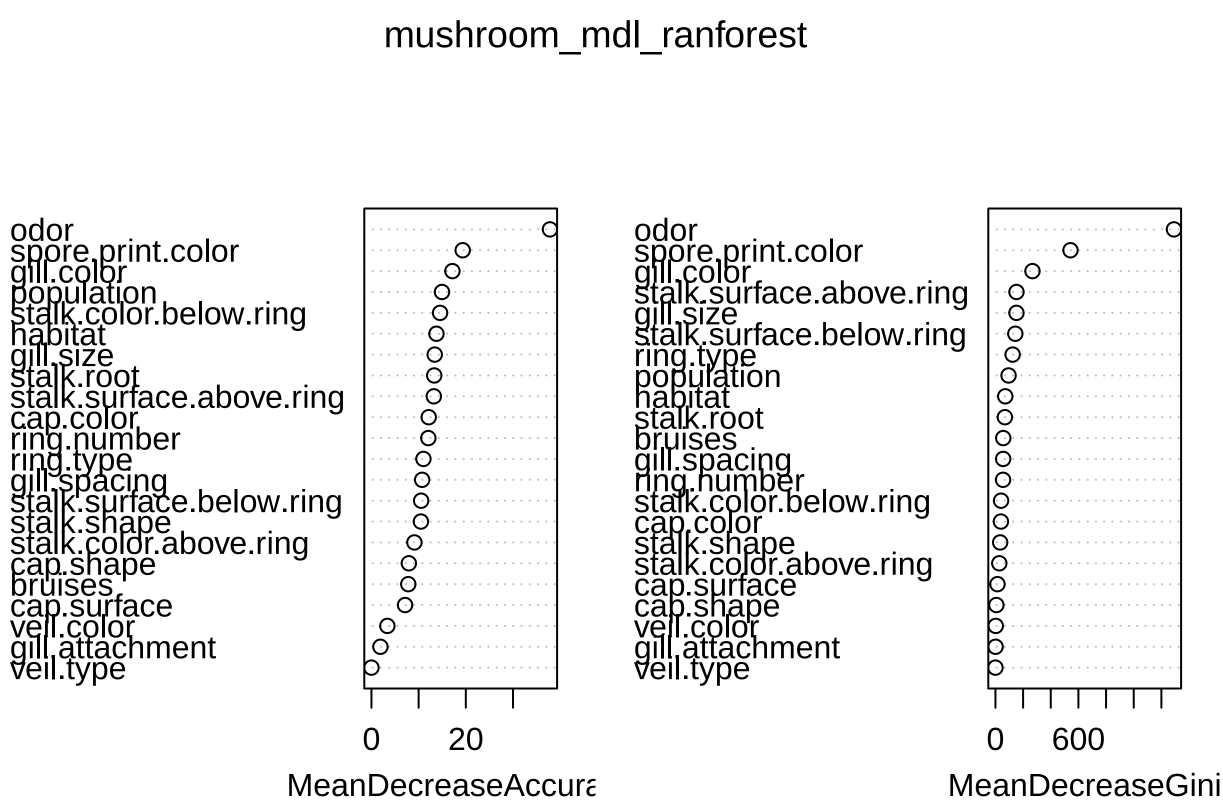
```r
importance(mushroom_mdl_ranforest)
```

```
##                                  e         p MeanDecreaseAccuracy
## cap.shape                 5.080529  6.619204             7.929082
## cap.surface               5.642044  6.372138             7.126618
## cap.color                12.275185  8.072518            12.139288
## bruises                   6.828824  6.736306             7.809696
## odor                     33.943566 30.613914            37.829428
## gill.attachment           1.951394  1.430975             1.900784
## gill.spacing              8.597167  9.644807            10.742491
## gill.size                14.472528 10.666829            13.408822
## gill.color               16.767960 10.086676            17.156519
## stalk.shape               7.604760 10.540555            10.481279
## stalk.root               12.480795  9.880242            13.304965
## stalk.surface.above.ring 12.699791  8.347161            13.223349
## stalk.surface.below.ring  9.468360  8.260831            10.531920
## stalk.color.above.ring    9.277861  6.081152             9.102598
## stalk.color.below.ring   14.895532  6.342392            14.545454
## veil.type                 0.000000  0.000000             0.000000
## veil.color                2.785361  3.733575             3.371209
## ring.number              11.137914 10.555566            12.054208
## ring.type                 8.686938  9.563414            11.015712
## spore.print.color        18.569862 15.223876            19.336402
## population               12.455042 12.080278            14.955008
## habitat                  13.178816  9.052500            13.770274
##                          MeanDecreaseGini
## cap.shape                       6.8611205
## cap.surface                    14.7355960
## cap.color                      39.5676751
## bruises                        56.3030363
## odor                         1290.9265078
## gill.attachment                 0.9371213
## gill.spacing                   55.8342115
## gill.size                     151.9342793
## gill.color                    267.9314166
## stalk.shape                    34.0853235
## stalk.root                     68.3581452
## stalk.surface.above.ring      152.1710823
## stalk.surface.below.ring      143.6505514
```

```
## stalk.color.above.ring         27.9892543
## stalk.color.below.ring         40.6361788
## veil.type                       0.0000000
## veil.color                      2.1918414
## ring.number                    54.9147640
## ring.type                     124.2530803
## spore.print.color             542.6689806
## population                     94.9234245
## habitat                        70.9116371
```

Graphically we can see the importance as:

```
varImpPlot(mushroom_mdl_ranforest)
```



We see that odor, spore.print.color and gill.color are top three variables to affect the accuracy of this model.

---

5. Model V : Boosting

In boosting, trees are grown sequentially. each tree is grown using the information from previously grown trees.
We first load the requisite package-gbm:

```
library(gbm)
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.3
```

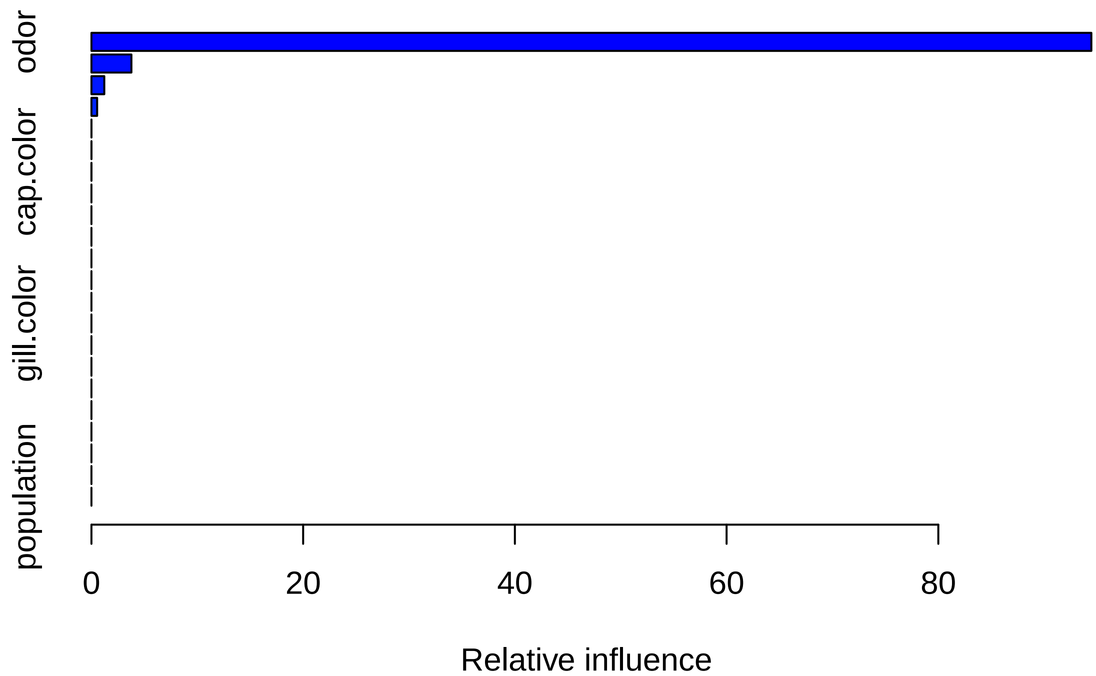
As this is a classification problem, we will use distribution="bernoulli" as one of the options of gbm() to cretae model. we have kept no. of trees as 500 just to keep it same with the bagging model to facilitate easy comparison. In this model the expectation from dependent variable is be in the form of 0 or 1 , so we change the data for class=p as 0 and class=e as 1. we will call these newsets as testset1 and trainset1

```
set.seed(1)
mushrooms_trainset1<-mushrooms_trainset
mushrooms_testset1<-mushrooms_testset
mushrooms_trainset1$class<-ifelse(mushrooms_trainset1$class=="e",1,0)
mushrooms_testset1$class<-ifelse(mushrooms_testset1$class=="e",1,0)
mushroom_mdl_boost<-gbm(class~.-class,mushrooms_trainset1,distribution="bernoulli",n.trees = 500,interac
```

```
## Warning in gbm.fit(x, y, offset = offset, distribution = distribution, w =
## w, : variable 16: veil.type has no variation.
```

here is the summary of Model:

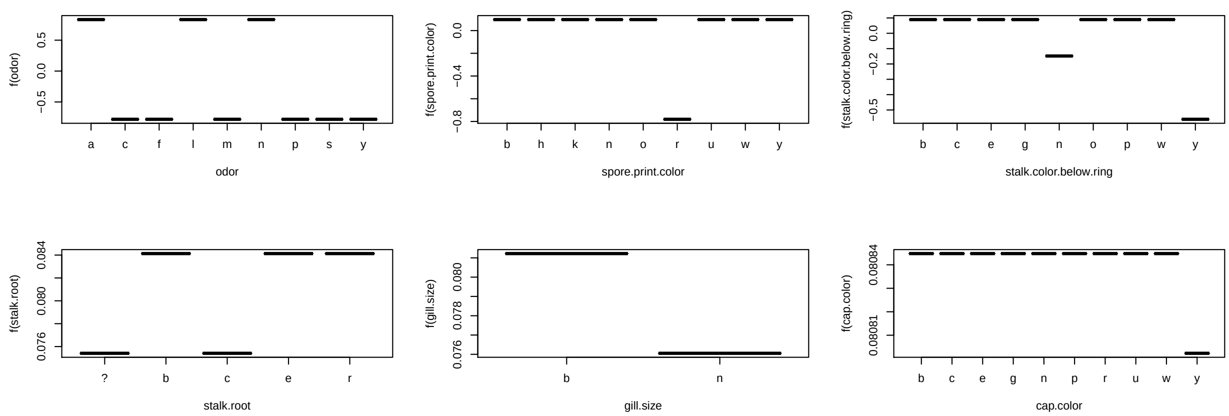
```
summary(mushroom_mdl_boost)
```




```
##                                               var      rel.inf
## odor                                         odor 9.445683e+01
## spore.print.color           spore.print.color 3.775892e+00
## stalk.color.below.ring     stalk.color.below.ring 1.214827e+00
## stalk.root                             stalk.root 5.396358e-01
## gill.size                               gill.size 6.078012e-03
## stalk.surface.above.ring stalk.surface.above.ring 4.532729e-03
## cap.color                               cap.color 2.173386e-03
## habitat                                   habitat 3.154492e-05
## cap.shape                               cap.shape 0.000000e+00
## cap.surface                           cap.surface 0.000000e+00
## bruises                                   bruises 0.000000e+00
## gill.attachment                   gill.attachment 0.000000e+00
## gill.spacing                         gill.spacing 0.000000e+00
## gill.color                             gill.color 0.000000e+00
## stalk.shape                           stalk.shape 0.000000e+00
## stalk.surface.below.ring stalk.surface.below.ring 0.000000e+00
## stalk.color.above.ring     stalk.color.above.ring 0.000000e+00
## veil.type                               veil.type 0.000000e+00
## veil.color                             veil.color 0.000000e+00
## ring.number                           ring.number 0.000000e+00
## ring.type                               ring.type 0.000000e+00
## population                             population 0.000000e+00
```

let's check the partial dependence plots of top 6 variables:

```
par(mfrow=c(2,3))
plot(mushroom_mdl_boost,i="odor")
plot(mushroom_mdl_boost,i="spore.print.color")
plot(mushroom_mdl_boost,i="stalk.color.below.ring")
plot(mushroom_mdl_boost,i="stalk.root")
plot(mushroom_mdl_boost,i="gill.size")
plot(mushroom_mdl_boost,i="cap.color")
```



we will now use this model to do the prediction for testset:

```
mushroom_pred_boost_test<-predict(mushroom_mdl_boost,mushrooms_testset1,n.trees=500)
head(mushroom_pred_boost_test)
```

```
## [1] -0.7808930  0.8812237  0.8812237  0.8812237  0.8812237  0.8712815
```

we can use the confusionmatrix for accuracy after using ifelse as the predictions don't come in 0 or 1 format.

Summary of Models used:

Based on the analysis and checking the accuracy of the above models with test data, we will go for Bagging or Random Tree as they have already reached perfect accuracy. Increasing the complexity further with boosting and decreasing the explainabilty would not be appropriate.

Updates on 07.22.2017 :

Following updates are intended : 1. create all the above model in using caret package 2. compare the performance of the models using caret's functions 3. explore more into data and try neuralnet and h2o

Model I : rpart:

Training the model on the trainset , we will also do the 10 fold cross validation and repeat 4 times:

```r
mushroom_mdl_crt_rpart<-train(x = mushrooms_trainset[,-1],y=mushrooms_trainset[,1],method="rpart",trCon

mushroom_mdl_crt_rpart
```

```
## CART
##
## 6500 samples
##   22 predictor
##    2 classes: 'e', 'p'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 5850, 5851, 5849, 5849, 5850, 5850, ...
## Resampling results across tuning parameters:
##
##   cp           Accuracy   Kappa
##   0.006383658  0.9944618  0.9889034
##   0.020108522  0.9875363  0.9750142
##   0.967124162  0.7035992  0.3854847
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.006383658.
```

let's do the predictions using this model now :

```r
mushroom_pred_crt_rpart_test<-predict(mushroom_mdl_crt_rpart,mushrooms_testset)
```

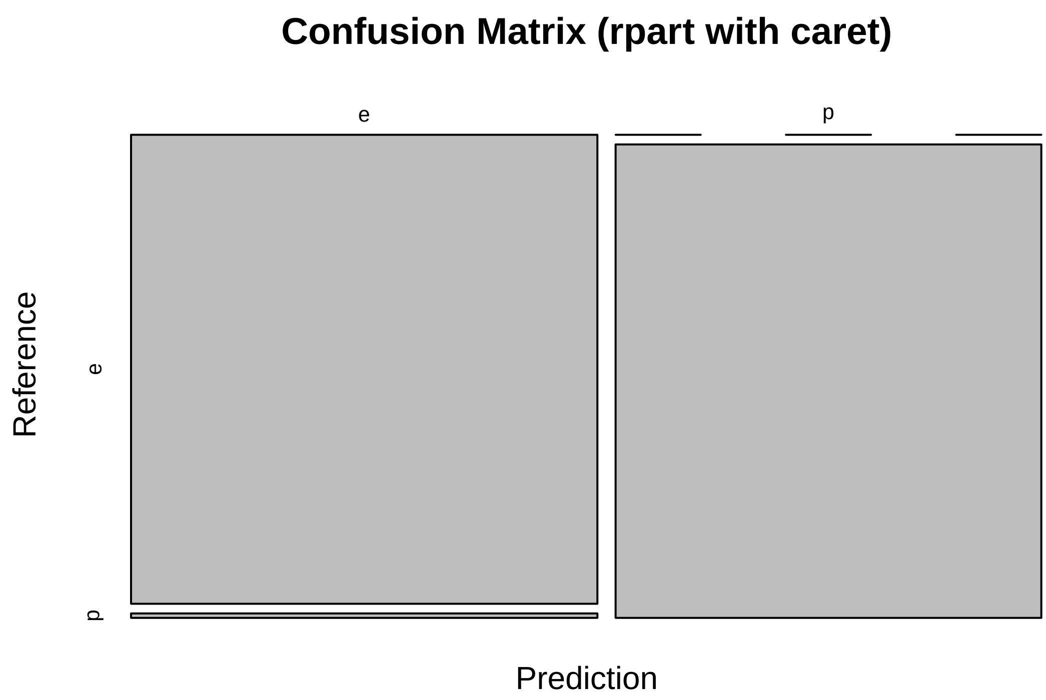
Let's check the confusion Matrix for this model:

```r
mushroom_tbl_crt_rpart_test<-confusionMatrix(mushroom_pred_crt_rpart_test,mushrooms_testset$class)
mushroom_tbl_crt_rpart_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   e   p
##          e 841   8
##          p   0 775
##
##                Accuracy : 0.9951
##                  95% CI : (0.9903, 0.9979)
##     No Information Rate : 0.5179
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9901
##  Mcnemar's Test P-Value : 0.01333
##
##             Sensitivity : 1.0000
##             Specificity : 0.9898
##          Pos Pred Value : 0.9906
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5179
##          Detection Rate : 0.5179
##    Detection Prevalence : 0.5228
##       Balanced Accuracy : 0.9949
##
```

```
##        'Positive' Class : e
##
```

```
plot(mushroom_tbl_crt_rpart_test$table,main="Confusion Matrix (rpart with caret)")
```



##Model II : Decision Tree and Bagging:

training the model with the same cross validation options :

```
mushroom_mdl_crt_bag<-train(x=mushrooms_trainset[,-1],y=mushrooms_trainset[,1],method="treebag",trContr
```

```
## Loading required package: plyr
```

```
## Loading required package: e1071
```

```
mushroom_mdl_crt_bag
```

```
## Bagged CART
##
## 6500 samples
##   22 predictor
##    2 classes: 'e', 'p'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 5850, 5850, 5850, 5851, 5851, 5849, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9998462  0.9996919
```

let's do the predictions using Bagged tree now :

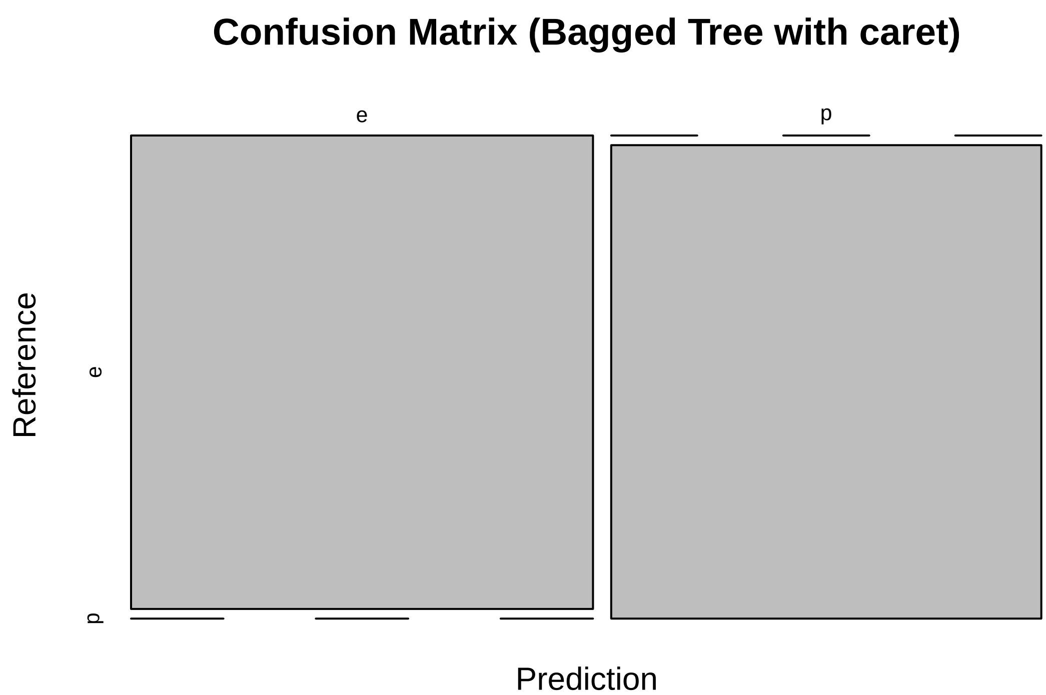
```
mushroom_pred_crt_bag_test<-predict(mushroom_mdl_crt_bag,mushrooms_testset)
```

Let's check the confusion Matrix for Bagged Tree:

```
mushroom_tbl_crt_bag_test<-confusionMatrix(mushroom_pred_crt_bag_test,mushrooms_testset$class)
mushroom_tbl_crt_bag_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   e   p
##          e 841   0
##          p   0 783
##
##                Accuracy : 1
##                  95% CI : (0.9977, 1)
##     No Information Rate : 0.5179
##     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 : 0.5179
##          Detection Rate : 0.5179
##    Detection Prevalence : 0.5179
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : e
##
```

```
plot(mushroom_tbl_crt_bag_test$table,main="Confusion Matrix (Bagged Tree with caret)")
```



Model III : RandomForest :

Training the model with the same cross validation options :

```
mushroom_mdl_crt_rf<-train(x=mushrooms_trainset[,-1],y=mushrooms_trainset[,1],method="ranger",trControl=

mushroom_mdl_crt_rf
```

```
## Random Forest
##
## 6500 samples
##   22 predictor
##    2 classes: 'e', 'p'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 5850, 5851, 5850, 5850, 5849, 5849, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    1         1
##   12    1         1
##   22    1         1
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

let's do the predictions using Bagged tree now :

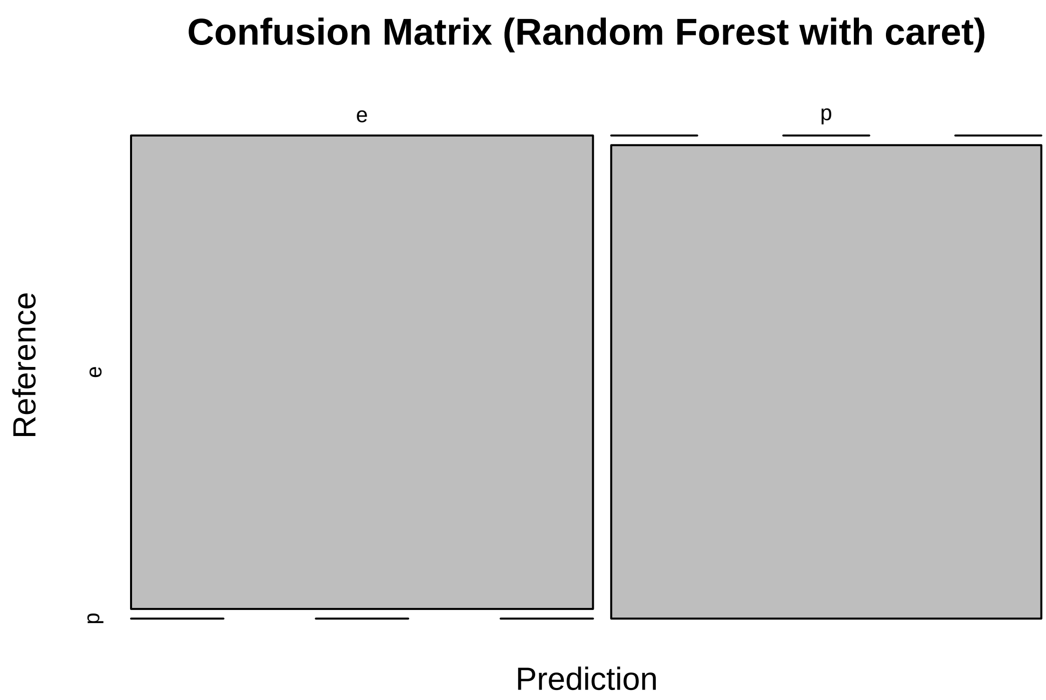
```
mushroom_pred_crt_rf_test<-predict(mushroom_mdl_crt_rf,mushrooms_testset)
```

Let's check the confusion Matrix for Bagged Tree:

```
mushroom_tbl_crt_rf_test<-confusionMatrix(mushroom_pred_crt_rf_test,mushrooms_testset$class)
mushroom_tbl_crt_rf_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   e   p
##          e 841   0
##          p   0 783
##
##                Accuracy : 1
##                  95% CI : (0.9977, 1)
##     No Information Rate : 0.5179
##     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 : 0.5179
##          Detection Rate : 0.5179
##    Detection Prevalence : 0.5179
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : e
##
```

```
plot(mushroom_tbl_crt_rf_test$table,main="Confusion Matrix (Random Forest with caret)")
```



Model III : Boosting :

Training the model with the same cross validation options :

```
mushroom_mdl_crt_boost<-train(x=mushrooms_trainset[,-1],y=mushrooms_trainset[,1],method="gbm",trControl=
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.

## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2070             nan     0.1000    0.0889
##      2        1.0620             nan     0.1000    0.0725
##      3        0.9409             nan     0.1000    0.0604
##      4        0.8391             nan     0.1000    0.0511
##      5        0.7526             nan     0.1000    0.0435
##      6        0.6781             nan     0.1000    0.0373
##      7        0.6139             nan     0.1000    0.0321
##      8        0.5579             nan     0.1000    0.0278
##      9        0.5096             nan     0.1000    0.0243
##     10        0.4673             nan     0.1000    0.0213
##     20        0.2404             nan     0.1000    0.0072
##     40        0.1276             nan     0.1000    0.0035
##     60        0.0842             nan     0.1000    0.0015
##     80        0.0580             nan     0.1000    0.0002
##    100        0.0407             nan     0.1000    0.0002
##    120        0.0293             nan     0.1000    0.0003
```

```
##    140        0.0228             nan     0.1000   -0.0000
##    150        0.0207             nan     0.1000    0.0000
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2002             nan     0.1000    0.0926
##      2        1.0485             nan     0.1000    0.0756
##      3        0.9222             nan     0.1000    0.0632
##      4        0.8160             nan     0.1000    0.0533
##      5        0.7255             nan     0.1000    0.0454
##      6        0.6475             nan     0.1000    0.0390
##      7        0.5795             nan     0.1000    0.0341
##      8        0.5201             nan     0.1000    0.0296
##      9        0.4686             nan     0.1000    0.0259
##     10        0.4231             nan     0.1000    0.0228
##     20        0.1700             nan     0.1000    0.0076
##     40        0.0498             nan     0.1000    0.0013
##     60        0.0259             nan     0.1000    0.0002
##     80        0.0140             nan     0.1000    0.0000
##    100        0.0080             nan     0.1000    0.0001
##    120        0.0049             nan     0.1000    0.0000
##    140        0.0031             nan     0.1000    0.0000
##    150        0.0024             nan     0.1000    0.0000
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.1976             nan     0.1000    0.0938
##      2        1.0438             nan     0.1000    0.0767
##      3        0.9156             nan     0.1000    0.0639
##      4        0.8078             nan     0.1000    0.0538
##      5        0.7157             nan     0.1000    0.0459
##      6        0.6364             nan     0.1000    0.0395
##      7        0.5676             nan     0.1000    0.0343
##      8        0.5076             nan     0.1000    0.0301
##      9        0.4551             nan     0.1000    0.0263
##     10        0.4088             nan     0.1000    0.0231
##     20        0.1524             nan     0.1000    0.0077
##     40        0.0320             nan     0.1000    0.0007
##     60        0.0111             nan     0.1000    0.0003
##     80        0.0045             nan     0.1000    0.0000
##    100        0.0021             nan     0.1000    0.0000
##    120        0.0010             nan     0.1000    0.0000
##    140        0.0005             nan     0.1000    0.0000
##    150        0.0004             nan     0.1000    0.0000
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2064             nan     0.1000    0.0894
##      2        1.0604             nan     0.1000    0.0732
##      3        0.9390             nan     0.1000    0.0609
##      4        0.8363             nan     0.1000    0.0514
```

```
##      5        0.7478             nan     0.1000    0.0437
##      6        0.6732             nan     0.1000    0.0374
##      7        0.6079             nan     0.1000    0.0324
##      8        0.5519             nan     0.1000    0.0280
##      9        0.5029             nan     0.1000    0.0245
##     10        0.4604             nan     0.1000    0.0214
##     20        0.2311             nan     0.1000    0.0053
##     40        0.1174             nan     0.1000    0.0033
##     60        0.0779             nan     0.1000    0.0016
##     80        0.0524             nan     0.1000    0.0008
##    100        0.0366             nan     0.1000    0.0001
##    120        0.0277             nan     0.1000    0.0001
##    140        0.0221             nan     0.1000    0.0000
##    150        0.0195             nan     0.1000    0.0002
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.1995             nan     0.1000    0.0928
##      2        1.0473             nan     0.1000    0.0758
##      3        0.9203             nan     0.1000    0.0630
##      4        0.8140             nan     0.1000    0.0532
##      5        0.7238             nan     0.1000    0.0450
##      6        0.6451             nan     0.1000    0.0393
##      7        0.5770             nan     0.1000    0.0340
##      8        0.5178             nan     0.1000    0.0295
##      9        0.4660             nan     0.1000    0.0258
##     10        0.4200             nan     0.1000    0.0229
##     20        0.1673             nan     0.1000    0.0072
##     40        0.0493             nan     0.1000    0.0012
##     60        0.0257             nan     0.1000    0.0005
##     80        0.0135             nan     0.1000    0.0000
##    100        0.0078             nan     0.1000    0.0001
##    120        0.0051             nan     0.1000    0.0000
##    140        0.0032             nan     0.1000    0.0001
##    150        0.0027             nan     0.1000    0.0000
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.1975             nan     0.1000    0.0939
##      2        1.0445             nan     0.1000    0.0764
##      3        0.9165             nan     0.1000    0.0640
##      4        0.8087             nan     0.1000    0.0539
##      5        0.7163             nan     0.1000    0.0463
##      6        0.6370             nan     0.1000    0.0396
##      7        0.5682             nan     0.1000    0.0343
##      8        0.5079             nan     0.1000    0.0301
##      9        0.4556             nan     0.1000    0.0260
##     10        0.4091             nan     0.1000    0.0232
##     20        0.1518             nan     0.1000    0.0074
##     40        0.0315             nan     0.1000    0.0009
##     60        0.0107             nan     0.1000    0.0003
##     80        0.0047             nan     0.1000    0.0001
```

```
##    100        0.0022             nan     0.1000    0.0000
##    120        0.0010             nan     0.1000    0.0000
##    140        0.0005             nan     0.1000    0.0000
##    150        0.0004             nan     0.1000    0.0000
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2073             nan     0.1000    0.0889
##      2        1.0620             nan     0.1000    0.0727
##      3        0.9412             nan     0.1000    0.0603
##      4        0.8397             nan     0.1000    0.0510
##      5        0.7537             nan     0.1000    0.0434
##      6        0.6790             nan     0.1000    0.0373
##      7        0.6145             nan     0.1000    0.0322
##      8        0.5586             nan     0.1000    0.0278
##      9        0.5103             nan     0.1000    0.0244
##     10        0.4677             nan     0.1000    0.0212
##     20        0.2384             nan     0.1000    0.0065
##     40        0.1261             nan     0.1000    0.0007
##     60        0.0824             nan     0.1000    0.0018
##     80        0.0559             nan     0.1000    0.0004
##    100        0.0385             nan     0.1000    0.0006
##    120        0.0278             nan     0.1000    0.0001
##    140        0.0211             nan     0.1000    0.0000
##    150        0.0192             nan     0.1000    0.0000
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.1994             nan     0.1000    0.0926
##      2        1.0478             nan     0.1000    0.0757
##      3        0.9216             nan     0.1000    0.0629
##      4        0.8148             nan     0.1000    0.0532
##      5        0.7243             nan     0.1000    0.0454
##      6        0.6465             nan     0.1000    0.0388
##      7        0.5785             nan     0.1000    0.0340
##      8        0.5197             nan     0.1000    0.0293
##      9        0.4675             nan     0.1000    0.0261
##     10        0.4218             nan     0.1000    0.0228
##     20        0.1692             nan     0.1000    0.0072
##     40        0.0486             nan     0.1000    0.0009
##     60        0.0229             nan     0.1000    0.0004
##     80        0.0121             nan     0.1000    0.0000
##    100        0.0070             nan     0.1000    0.0000
##    120        0.0047             nan     0.1000    0.0000
##    140        0.0028             nan     0.1000    0.0000
##    150        0.0022             nan     0.1000    0.0000
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.1972             nan     0.1000    0.0937
##      2        1.0443             nan     0.1000    0.0764
```

```
##      3        0.9158             nan     0.1000    0.0641
##      4        0.8074             nan     0.1000    0.0541
##      5        0.7153             nan     0.1000    0.0460
##      6        0.6356             nan     0.1000    0.0398
##      7        0.5663             nan     0.1000    0.0344
##      8        0.5060             nan     0.1000    0.0300
##      9        0.4535             nan     0.1000    0.0263
##     10        0.4070             nan     0.1000    0.0231
##     20        0.1515             nan     0.1000    0.0072
##     40        0.0301             nan     0.1000    0.0009
##     60        0.0104             nan     0.1000    0.0003
##     80        0.0046             nan     0.1000    0.0001
##    100        0.0021             nan     0.1000    0.0000
##    120        0.0010             nan     0.1000    0.0000
##    140        0.0005             nan     0.1000    0.0000
##    150        0.0004             nan     0.1000    0.0000
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 6L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2058             nan     0.1000    0.0894
##      2        1.0598             nan     0.1000    0.0731
##      3        0.9374             nan     0.1000    0.0606
##      4        0.8352             nan     0.1000    0.0512
##      5        0.7478             nan     0.1000    0.0437
##      6        0.6738             nan     0.1000    0.0375
##      7        0.6093             nan     0.1000    0.0324
##      8        0.5532             nan     0.1000    0.0281
##      9        0.5045             nan     0.1000    0.0245
##     10        0.4614             nan     0.1000    0.0215
##     20        0.2305             nan     0.1000    0.0070
##     40        0.1227             nan     0.1000    0.0006
##     60        0.0807             nan     0.1000    0.0002
##     80        0.0542             nan     0.1000    0.0010
##    100        0.0383             nan     0.1000    0.0003
##    120        0.0273             nan     0.1000    0.0003
##    140        0.0220             nan     0.1000   -0.0000
##    150        0.0196             nan     0.1000    0.0003
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 6L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2000             nan     0.1000    0.0927
##      2        1.0482             nan     0.1000    0.0758
##      3        0.9219             nan     0.1000    0.0631
##      4        0.8156             nan     0.1000    0.0533
##      5        0.7245             nan     0.1000    0.0455
##      6        0.6459             nan     0.1000    0.0391
##      7        0.5782             nan     0.1000    0.0336
##      8        0.5188             nan     0.1000    0.0295
##      9        0.4660             nan     0.1000    0.0258
##     10        0.4202             nan     0.1000    0.0228
##     20        0.1680             nan     0.1000    0.0072
##     40        0.0485             nan     0.1000    0.0010
```

```
##     60        0.0251             nan     0.1000    0.0004
##     80        0.0121             nan     0.1000    0.0001
##    100        0.0068             nan     0.1000    0.0001
##    120        0.0044             nan     0.1000    0.0001
##    140        0.0028             nan     0.1000    0.0000
##    150        0.0021             nan     0.1000    0.0000
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 6L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.1974             nan     0.1000    0.0937
##      2        1.0436             nan     0.1000    0.0768
##      3        0.9158             nan     0.1000    0.0640
##      4        0.8078             nan     0.1000    0.0539
##      5        0.7152             nan     0.1000    0.0462
##      6        0.6359             nan     0.1000    0.0396
##      7        0.5668             nan     0.1000    0.0345
##      8        0.5068             nan     0.1000    0.0300
##      9        0.4543             nan     0.1000    0.0264
##     10        0.4077             nan     0.1000    0.0231
##     20        0.1508             nan     0.1000    0.0074
##     40        0.0299             nan     0.1000    0.0007
##     60        0.0096             nan     0.1000    0.0003
##     80        0.0040             nan     0.1000    0.0001
##    100        0.0019             nan     0.1000    0.0001
##    120        0.0010             nan     0.1000    0.0000
##    140        0.0005             nan     0.1000    0.0000
##    150        0.0003             nan     0.1000    0.0000
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2064             nan     0.1000    0.0895
##      2        1.0606             nan     0.1000    0.0729
##      3        0.9394             nan     0.1000    0.0607
##      4        0.8373             nan     0.1000    0.0512
##      5        0.7499             nan     0.1000    0.0436
##      6        0.6747             nan     0.1000    0.0373
##      7        0.6104             nan     0.1000    0.0324
##      8        0.5545             nan     0.1000    0.0280
##      9        0.5059             nan     0.1000    0.0245
##     10        0.4634             nan     0.1000    0.0213
##     20        0.2416             nan     0.1000    0.0023
##     40        0.1222             nan     0.1000    0.0003
##     60        0.0749             nan     0.1000    0.0002
##     80        0.0498             nan     0.1000    0.0008
##    100        0.0360             nan     0.1000    0.0001
##    120        0.0259             nan     0.1000    0.0002
##    140        0.0215             nan     0.1000    0.0003
##    150        0.0193             nan     0.1000   -0.0000
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
```

```
##      1        1.1998             nan     0.1000    0.0928
##      2        1.0477             nan     0.1000    0.0760
##      3        0.9213             nan     0.1000    0.0634
##      4        0.8145             nan     0.1000    0.0535
##      5        0.7231             nan     0.1000    0.0457
##      6        0.6446             nan     0.1000    0.0393
##      7        0.5772             nan     0.1000    0.0336
##      8        0.5176             nan     0.1000    0.0298
##      9        0.4653             nan     0.1000    0.0260
##     10        0.4196             nan     0.1000    0.0228
##     20        0.1668             nan     0.1000    0.0073
##     40        0.0480             nan     0.1000    0.0010
##     60        0.0252             nan     0.1000    0.0004
##     80        0.0128             nan     0.1000    0.0000
##    100        0.0074             nan     0.1000    0.0002
##    120        0.0048             nan     0.1000    0.0001
##    140        0.0030             nan     0.1000    0.0001
##    150        0.0025             nan     0.1000    0.0000

## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.

## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.1979             nan     0.1000    0.0936
##      2        1.0447             nan     0.1000    0.0765
##      3        0.9169             nan     0.1000    0.0639
##      4        0.8092             nan     0.1000    0.0539
##      5        0.7173             nan     0.1000    0.0459
##      6        0.6379             nan     0.1000    0.0396
##      7        0.5692             nan     0.1000    0.0345
##      8        0.5092             nan     0.1000    0.0298
##      9        0.4567             nan     0.1000    0.0262
##     10        0.4106             nan     0.1000    0.0229
##     20        0.1519             nan     0.1000    0.0076
##     40        0.0319             nan     0.1000    0.0012
##     60        0.0098             nan     0.1000    0.0002
##     80        0.0045             nan     0.1000    0.0000
##    100        0.0020             nan     0.1000    0.0000
##    120        0.0011             nan     0.1000    0.0000
##    140        0.0006             nan     0.1000    0.0000
##    150        0.0004             nan     0.1000    0.0000

## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 1L, 6L, :
## variable 16: veil.type has no variation.

## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2062             nan     0.1000    0.0895
##      2        1.0598             nan     0.1000    0.0729
##      3        0.9383             nan     0.1000    0.0606
##      4        0.8356             nan     0.1000    0.0511
##      5        0.7479             nan     0.1000    0.0435
##      6        0.6731             nan     0.1000    0.0373
##      7        0.6085             nan     0.1000    0.0323
##      8        0.5524             nan     0.1000    0.0280
##      9        0.5034             nan     0.1000    0.0243
##     10        0.4605             nan     0.1000    0.0212
```

```
##     20        0.2319             nan     0.1000    0.0073
##     40        0.1221             nan     0.1000    0.0005
##     60        0.0751             nan     0.1000    0.0014
##     80        0.0498             nan     0.1000    0.0009
##    100        0.0369             nan     0.1000    0.0001
##    120        0.0273             nan     0.1000    0.0001
##    140        0.0219             nan     0.1000    0.0002
##    150        0.0195             nan     0.1000    0.0002
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 1L, 6L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.1996             nan     0.1000    0.0928
##      2        1.0473             nan     0.1000    0.0760
##      3        0.9209             nan     0.1000    0.0633
##      4        0.8140             nan     0.1000    0.0534
##      5        0.7231             nan     0.1000    0.0455
##      6        0.6448             nan     0.1000    0.0393
##      7        0.5769             nan     0.1000    0.0339
##      8        0.5180             nan     0.1000    0.0292
##      9        0.4658             nan     0.1000    0.0258
##     10        0.4196             nan     0.1000    0.0229
##     20        0.1672             nan     0.1000    0.0072
##     40        0.0490             nan     0.1000    0.0007
##     60        0.0242             nan     0.1000    0.0003
##     80        0.0120             nan     0.1000    0.0001
##    100        0.0073             nan     0.1000    0.0001
##    120        0.0047             nan     0.1000    0.0001
##    140        0.0029             nan     0.1000    0.0000
##    150        0.0024             nan     0.1000    0.0001
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 1L, 6L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.1971             nan     0.1000    0.0938
##      2        1.0435             nan     0.1000    0.0767
##      3        0.9157             nan     0.1000    0.0640
##      4        0.8077             nan     0.1000    0.0539
##      5        0.7156             nan     0.1000    0.0460
##      6        0.6361             nan     0.1000    0.0397
##      7        0.5674             nan     0.1000    0.0344
##      8        0.5069             nan     0.1000    0.0301
##      9        0.4542             nan     0.1000    0.0263
##     10        0.4081             nan     0.1000    0.0231
##     20        0.1527             nan     0.1000    0.0071
##     40        0.0312             nan     0.1000    0.0009
##     60        0.0115             nan     0.1000    0.0003
##     80        0.0051             nan     0.1000    0.0001
##    100        0.0024             nan     0.1000    0.0001
##    120        0.0012             nan     0.1000    0.0000
##    140        0.0006             nan     0.1000    0.0000
##    150        0.0005             nan     0.1000    0.0000
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 1L, 6L, :
```

```
## variable 16: veil.type has no variation.

## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2066             nan     0.1000    0.0887
##      2        1.0615             nan     0.1000    0.0727
##      3        0.9406             nan     0.1000    0.0606
##      4        0.8393             nan     0.1000    0.0507
##      5        0.7527             nan     0.1000    0.0434
##      6        0.6781             nan     0.1000    0.0374
##      7        0.6136             nan     0.1000    0.0322
##      8        0.5577             nan     0.1000    0.0278
##      9        0.5090             nan     0.1000    0.0242
##     10        0.4663             nan     0.1000    0.0212
##     20        0.2378             nan     0.1000    0.0072
##     40        0.1204             nan     0.1000    0.0031
##     60        0.0799             nan     0.1000    0.0015
##     80        0.0561             nan     0.1000    0.0002
##    100        0.0400             nan     0.1000    0.0001
##    120        0.0292             nan     0.1000    0.0001
##    140        0.0232             nan     0.1000    0.0000
##    150        0.0205             nan     0.1000    0.0000

## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 1L, 6L, :
## variable 16: veil.type has no variation.

## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.1991             nan     0.1000    0.0927
##      2        1.0474             nan     0.1000    0.0756
##      3        0.9204             nan     0.1000    0.0631
##      4        0.8136             nan     0.1000    0.0532
##      5        0.7227             nan     0.1000    0.0454
##      6        0.6448             nan     0.1000    0.0390
##      7        0.5776             nan     0.1000    0.0336
##      8        0.5182             nan     0.1000    0.0297
##      9        0.4664             nan     0.1000    0.0259
##     10        0.4210             nan     0.1000    0.0227
##     20        0.1688             nan     0.1000    0.0072
##     40        0.0518             nan     0.1000    0.0014
##     60        0.0261             nan     0.1000    0.0003
##     80        0.0118             nan     0.1000    0.0001
##    100        0.0068             nan     0.1000    0.0000
##    120        0.0044             nan     0.1000    0.0000
##    140        0.0029             nan     0.1000    0.0000
##    150        0.0023             nan     0.1000    0.0000

## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 1L, 6L, :
## variable 16: veil.type has no variation.

## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.1976             nan     0.1000    0.0937
##      2        1.0445             nan     0.1000    0.0763
##      3        0.9169             nan     0.1000    0.0639
##      4        0.8090             nan     0.1000    0.0538
##      5        0.7168             nan     0.1000    0.0460
##      6        0.6375             nan     0.1000    0.0397
##      7        0.5682             nan     0.1000    0.0346
```

```
##      8        0.5082             nan     0.1000    0.0300
##      9        0.4558             nan     0.1000    0.0261
##     10        0.4097             nan     0.1000    0.0231
##     20        0.1548             nan     0.1000    0.0073
##     40        0.0344             nan     0.1000    0.0008
##     60        0.0111             nan     0.1000    0.0004
##     80        0.0046             nan     0.1000    0.0001
##    100        0.0022             nan     0.1000    0.0001
##    120        0.0012             nan     0.1000    0.0000
##    140        0.0007             nan     0.1000    0.0000
##    150        0.0005             nan     0.1000    0.0000
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2066             nan     0.1000    0.0892
##      2        1.0608             nan     0.1000    0.0729
##      3        0.9396             nan     0.1000    0.0607
##      4        0.8376             nan     0.1000    0.0513
##      5        0.7513             nan     0.1000    0.0436
##      6        0.6766             nan     0.1000    0.0374
##      7        0.6118             nan     0.1000    0.0322
##      8        0.5551             nan     0.1000    0.0281
##      9        0.5059             nan     0.1000    0.0243
##     10        0.4631             nan     0.1000    0.0213
##     20        0.2341             nan     0.1000    0.0072
##     40        0.1228             nan     0.1000    0.0005
##     60        0.0748             nan     0.1000    0.0002
##     80        0.0558             nan     0.1000    0.0001
##    100        0.0402             nan     0.1000    0.0001
##    120        0.0268             nan     0.1000    0.0004
##    140        0.0208             nan     0.1000   -0.0000
##    150        0.0182             nan     0.1000    0.0000
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.1998             nan     0.1000    0.0928
##      2        1.0478             nan     0.1000    0.0760
##      3        0.9207             nan     0.1000    0.0632
##      4        0.8144             nan     0.1000    0.0534
##      5        0.7237             nan     0.1000    0.0455
##      6        0.6458             nan     0.1000    0.0389
##      7        0.5784             nan     0.1000    0.0335
##      8        0.5188             nan     0.1000    0.0300
##      9        0.4666             nan     0.1000    0.0262
##     10        0.4204             nan     0.1000    0.0229
##     20        0.1669             nan     0.1000    0.0071
##     40        0.0468             nan     0.1000    0.0006
##     60        0.0210             nan     0.1000    0.0004
##     80        0.0119             nan     0.1000    0.0002
##    100        0.0062             nan     0.1000    0.0001
##    120        0.0036             nan     0.1000    0.0001
##    140        0.0024             nan     0.1000    0.0000
```

```
##    150        0.0018             nan     0.1000    0.0000

## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.

## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.1978             nan     0.1000    0.0935
##      2        1.0445             nan     0.1000    0.0764
##      3        0.9168             nan     0.1000    0.0637
##      4        0.8081             nan     0.1000    0.0543
##      5        0.7159             nan     0.1000    0.0461
##      6        0.6364             nan     0.1000    0.0397
##      7        0.5676             nan     0.1000    0.0343
##      8        0.5072             nan     0.1000    0.0302
##      9        0.4545             nan     0.1000    0.0264
##     10        0.4082             nan     0.1000    0.0232
##     20        0.1510             nan     0.1000    0.0080
##     40        0.0293             nan     0.1000    0.0011
##     60        0.0093             nan     0.1000    0.0003
##     80        0.0036             nan     0.1000    0.0001
##    100        0.0017             nan     0.1000    0.0000
##    120        0.0008             nan     0.1000    0.0000
##    140        0.0004             nan     0.1000    0.0000
##    150        0.0003             nan     0.1000    0.0000

## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.

## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2071             nan     0.1000    0.0892
##      2        1.0604             nan     0.1000    0.0729
##      3        0.9383             nan     0.1000    0.0607
##      4        0.8358             nan     0.1000    0.0514
##      5        0.7483             nan     0.1000    0.0436
##      6        0.6739             nan     0.1000    0.0375
##      7        0.6100             nan     0.1000    0.0323
##      8        0.5538             nan     0.1000    0.0280
##      9        0.5046             nan     0.1000    0.0245
##     10        0.4619             nan     0.1000    0.0214
##     20        0.2320             nan     0.1000    0.0062
##     40        0.1226             nan     0.1000    0.0035
##     60        0.0761             nan     0.1000    0.0002
##     80        0.0550             nan     0.1000    0.0001
##    100        0.0384             nan     0.1000    0.0005
##    120        0.0274             nan     0.1000    0.0003
##    140        0.0222             nan     0.1000    0.0002
##    150        0.0196             nan     0.1000    0.0000

## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.

## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.1996             nan     0.1000    0.0927
##      2        1.0471             nan     0.1000    0.0760
##      3        0.9209             nan     0.1000    0.0633
##      4        0.8141             nan     0.1000    0.0533
##      5        0.7228             nan     0.1000    0.0456
```

```
##      6        0.6445             nan     0.1000    0.0390
##      7        0.5762             nan     0.1000    0.0341
##      8        0.5172             nan     0.1000    0.0294
##      9        0.4652             nan     0.1000    0.0261
##     10        0.4202             nan     0.1000    0.0225
##     20        0.1664             nan     0.1000    0.0072
##     40        0.0476             nan     0.1000    0.0011
##     60        0.0232             nan     0.1000    0.0001
##     80        0.0113             nan     0.1000    0.0001
##    100        0.0070             nan     0.1000    0.0000
##    120        0.0044             nan     0.1000    0.0000
##    140        0.0030             nan     0.1000    0.0000
##    150        0.0022             nan     0.1000    0.0000

## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.

## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.1977             nan     0.1000    0.0935
##      2        1.0439             nan     0.1000    0.0769
##      3        0.9156             nan     0.1000    0.0641
##      4        0.8074             nan     0.1000    0.0542
##      5        0.7152             nan     0.1000    0.0461
##      6        0.6358             nan     0.1000    0.0399
##      7        0.5666             nan     0.1000    0.0346
##      8        0.5062             nan     0.1000    0.0301
##      9        0.4535             nan     0.1000    0.0263
##     10        0.4071             nan     0.1000    0.0232
##     20        0.1511             nan     0.1000    0.0072
##     40        0.0308             nan     0.1000    0.0010
##     60        0.0101             nan     0.1000    0.0001
##     80        0.0045             nan     0.1000    0.0001
##    100        0.0021             nan     0.1000    0.0000
##    120        0.0011             nan     0.1000    0.0000
##    140        0.0006             nan     0.1000    0.0000
##    150        0.0004             nan     0.1000    0.0000

## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.

## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2072             nan     0.1000    0.0889
##      2        1.0618             nan     0.1000    0.0728
##      3        0.9413             nan     0.1000    0.0605
##      4        0.8387             nan     0.1000    0.0510
##      5        0.7518             nan     0.1000    0.0434
##      6        0.6773             nan     0.1000    0.0372
##      7        0.6129             nan     0.1000    0.0322
##      8        0.5571             nan     0.1000    0.0279
##      9        0.5086             nan     0.1000    0.0243
##     10        0.4662             nan     0.1000    0.0212
##     20        0.2379             nan     0.1000    0.0072
##     40        0.1259             nan     0.1000    0.0003
##     60        0.0793             nan     0.1000    0.0002
##     80        0.0561             nan     0.1000    0.0010
##    100        0.0391             nan     0.1000    0.0001
```

```
##    120        0.0286             nan     0.1000    0.0000
##    140        0.0232             nan     0.1000    0.0000
##    150        0.0203             nan     0.1000    0.0002
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.1995             nan     0.1000    0.0924
##      2        1.0489             nan     0.1000    0.0757
##      3        0.9229             nan     0.1000    0.0632
##      4        0.8162             nan     0.1000    0.0533
##      5        0.7254             nan     0.1000    0.0455
##      6        0.6470             nan     0.1000    0.0390
##      7        0.5790             nan     0.1000    0.0339
##      8        0.5195             nan     0.1000    0.0296
##      9        0.4676             nan     0.1000    0.0259
##     10        0.4214             nan     0.1000    0.0229
##     20        0.1693             nan     0.1000    0.0070
##     40        0.0494             nan     0.1000    0.0010
##     60        0.0234             nan     0.1000    0.0003
##     80        0.0123             nan     0.1000    0.0001
##    100        0.0076             nan     0.1000    0.0002
##    120        0.0048             nan     0.1000    0.0000
##    140        0.0029             nan     0.1000    0.0000
##    150        0.0024             nan     0.1000    0.0000
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.1971             nan     0.1000    0.0939
##      2        1.0440             nan     0.1000    0.0765
##      3        0.9161             nan     0.1000    0.0640
##      4        0.8083             nan     0.1000    0.0538
##      5        0.7159             nan     0.1000    0.0463
##      6        0.6368             nan     0.1000    0.0396
##      7        0.5676             nan     0.1000    0.0346
##      8        0.5080             nan     0.1000    0.0298
##      9        0.4554             nan     0.1000    0.0264
##     10        0.4091             nan     0.1000    0.0232
##     20        0.1518             nan     0.1000    0.0075
##     40        0.0329             nan     0.1000    0.0009
##     60        0.0107             nan     0.1000    0.0003
##     80        0.0043             nan     0.1000    0.0001
##    100        0.0021             nan     0.1000    0.0000
##    120        0.0012             nan     0.1000    0.0000
##    140        0.0006             nan     0.1000    0.0000
##    150        0.0004             nan     0.1000    0.0000
```

```
## Warning in gbm.fit(x = structure(list(cap.shape = structure(c(6L, 6L, 1L, :
## variable 16: veil.type has no variation.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.1973             nan     0.1000    0.0939
##      2        1.0442             nan     0.1000    0.0764
##      3        0.9167             nan     0.1000    0.0637
```

```
##      4        0.8080             nan     0.1000    0.0542
##      5        0.7156             nan     0.1000    0.0463
##      6        0.6358             nan     0.1000    0.0398
##      7        0.5669             nan     0.1000    0.0345
##      8        0.5067             nan     0.1000    0.0300
##      9        0.4540             nan     0.1000    0.0264
##     10        0.4077             nan     0.1000    0.0232
##     20        0.1511             nan     0.1000    0.0073
##     40        0.0307             nan     0.1000    0.0012
##     60        0.0104             nan     0.1000    0.0003
##     80        0.0045             nan     0.1000    0.0000
##    100        0.0022             nan     0.1000    0.0001
```

```
mushroom_mdl_crt_boost
```

```
## Stochastic Gradient Boosting
##
## 6500 samples
##   22 predictor
##    2 classes: 'e', 'p'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 5850, 5850, 5850, 5850, 5850, 5850, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.9949231  0.9898271
##   1                  100      0.9969231  0.9938365
##   1                  150      0.9981538  0.9963017
##   2                   50      0.9970769  0.9941447
##   2                  100      0.9995385  0.9990753
##   2                  150      1.0000000  1.0000000
##   3                   50      0.9990769  0.9981510
##   3                  100      1.0000000  1.0000000
##   3                  150      1.0000000  1.0000000
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 100,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

let's do the predictions using Boosting now :

```
mushroom_pred_crt_boost_test<-predict(mushroom_mdl_crt_boost,mushrooms_testset)
```

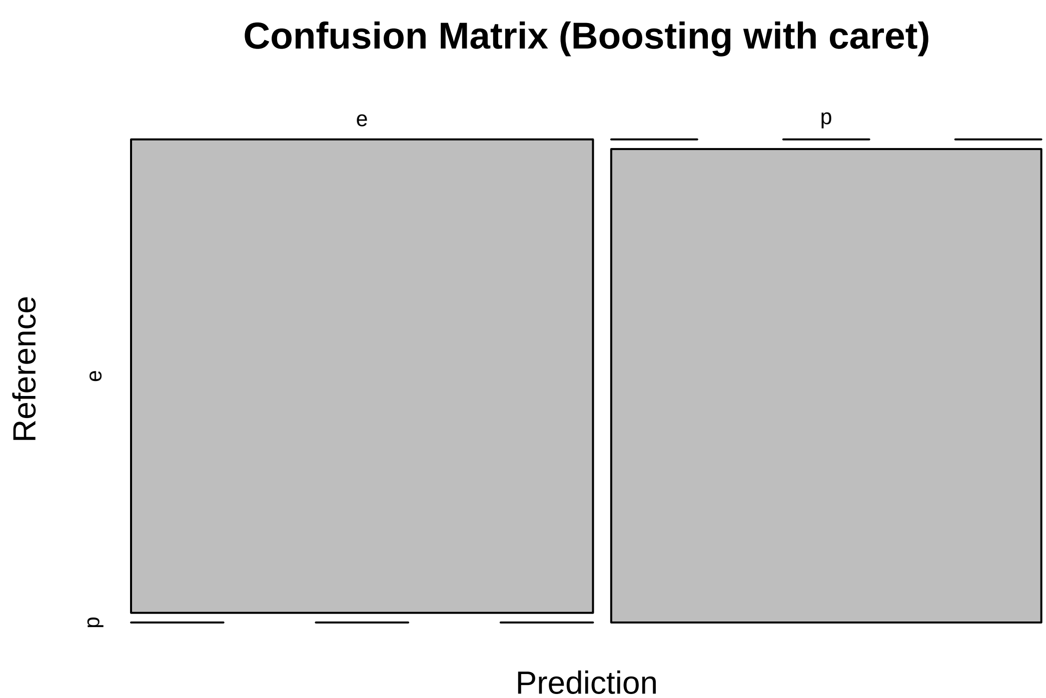
Let's check the confusion Matrix for Boosting :

```
mushroom_tbl_crt_boost_test<-confusionMatrix(mushroom_pred_crt_boost_test,mushrooms_testset$class)
mushroom_tbl_crt_boost_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   e   p
```

```
##          e 841   0
##          p   0 783
##
##                Accuracy : 1
##                  95% CI : (0.9977, 1)
##     No Information Rate : 0.5179
##     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 : 0.5179
##          Detection Rate : 0.5179
##    Detection Prevalence : 0.5179
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : e
##
```

```r
plot(mushroom_tbl_crt_boost_test$table,main="Confusion Matrix (Boosting with caret)")
```

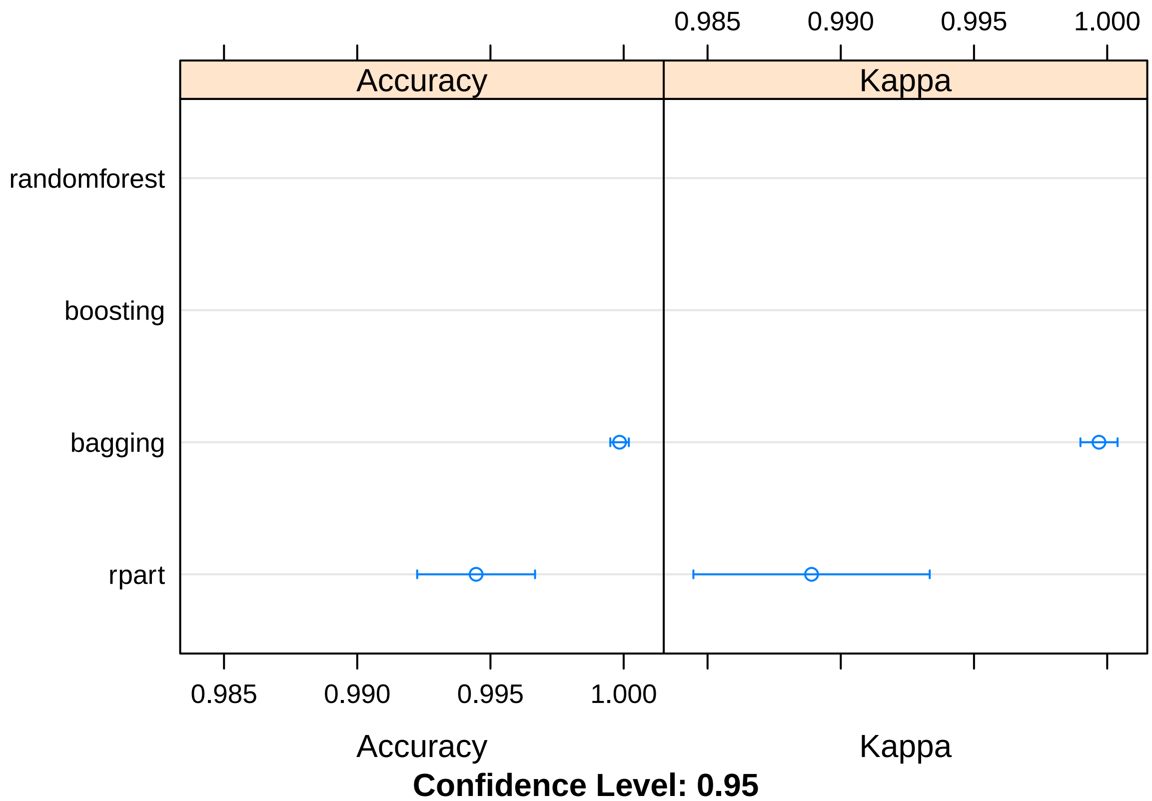


Model Comparison:

```
comparison<-resamples(list(rpart=mushroom_mdl_crt_rpart,bagging=mushroom_mdl_crt_bag,randomforest=mushro
summary(comparison)
```

```
##
## Call:
## summary.resamples(object = comparison)
##
## Models: rpart, bagging, randomforest, boosting
## Number of resamples: 10
##
## Accuracy
##                Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## rpart        0.9892  0.9927 0.9946 0.9945  0.9965 0.9985    0
## bagging      0.9985  1.0000 1.0000 0.9998  1.0000 1.0000    0
## randomforest 1.0000  1.0000 1.0000 1.0000  1.0000 1.0000    0
## boosting     1.0000  1.0000 1.0000 1.0000  1.0000 1.0000    0
##
## Kappa
##                Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## rpart        0.9784  0.9853 0.9892 0.9889  0.9931 0.9969    0
## bagging      0.9969  1.0000 1.0000 0.9997  1.0000 1.0000    0
## randomforest 1.0000  1.0000 1.0000 1.0000  1.0000 1.0000    0
## boosting     1.0000  1.0000 1.0000 1.0000  1.0000 1.0000    0
```

```
dotplot(comparison)
```


0.985
0.990
0.995
1.000
Accuracy
Kappa
randomforest
boosting
bagging
rpart
0.985
0.990
0.995
1.000
Accuracy
Kappa
Confidence Level: 0.95