HW7

A. Import Data

Delete the variable which has only one class. And turn all variables into the type of factor.

```
setwd("/Users/cindychen/Desktop/   /HW/")
data <- read_csv("mushrooms.csv")
# str(data)
data[] <- lapply(data, factor)
# str(data)
data$`veil-type` <- NULL
data$bruises <- NULL
```

B. Split the dataset for training and testing

Using package **caret** to split the dataset for training and testing base on the class of the mushrooms. There are total of 4208 edible mushrooms and 3916 poisson mushrooms. And the training set has a total of 2806 edible mushrooms and 2611 poisson mushrooms.

```
set.seed(20211123)
train.id <- createDataPartition(data$class , p = 2/3 , list = F)
train <- data[train.id,]
test <- data[-train.id,]
dim(train);dim(test)
```

```
## [1] 5417   21
```

```
## [1] 2707   21
```

```
table(data$class)
```

```
##
##    e    p
## 4208 3916
```

```
table(train$class)
```

```
##
##    e    p
## 2806 2611
```

```
table(test$class)
```

```
##
##    e    p
## 1402 1305
```

C. Naive Bayes Test

Using Naive Bayes to train the data and make prediction on the test data. Compare the real data and the predicted data which shows 94.7% of accuracy on this model.

```
model.nb <- naiveBayes(class~. , train)
# model.nb
# predict(model.nb,test)
tab = table("predict" = predict(model.nb,test) , 'real' = test$class)
tab
```

```
##         real
## predict    e    p
##       e 1402  144
##       p    0 1161
```

```
sum(diag(tab))/sum(tab)
```

```
## [1] 0.9468046
```

D. Logistic Regression Test

a-1. Use cap-color to predict

Fit the logistic model with one variable **cap-color**. And make prediction for testing data.

```
model.lr.capcolor <- glm(class~`cap-color`,data = train,family = "binomial")
summary(model.lr.capcolor)
```

```
##
## Call:
## glm(formula = class ~ `cap-color`, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5801  -1.0942  -0.8559   1.2630   1.5928
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    0.9102     0.2062   4.415 1.01e-05 ***
## `cap-color`c  -1.8485     0.4439  -4.164 3.13e-05 ***
## `cap-color`e  -0.6278     0.2158  -2.909  0.00362 **
## `cap-color`g  -1.1585     0.2139  -5.417 6.06e-08 ***
## `cap-color`n  -1.1091     0.2127  -5.216 1.83e-07 ***
## `cap-color`p  -0.4369     0.2919  -1.497  0.13444
## `cap-color`r -15.4763   294.2479  -0.053  0.95805
## `cap-color`u -15.4763   254.8261  -0.061  0.95157
## `cap-color`w  -1.7258     0.2218  -7.780 7.26e-15 ***
## `cap-color`y  -0.3277     0.2206  -1.486  0.13738
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7502.5  on 5416  degrees of freedom
## Residual deviance: 7228.5  on 5407  degrees of freedom
## AIC: 7248.5
##
## Number of Fisher Scoring iterations: 13
```

```
predict(model.lr.capcolor,test) %>% head()
```

```
##          1          2          3          4          5          6
## -0.1988999 -0.8155706  0.5825026 -0.8155706  0.5825026  0.5825026
```

```
predict(model.lr.capcolor,test,type = "response") %>% head()
```

```
##         1         2         3         4         5         6
## 0.4504383 0.3067047 0.6416431 0.3067047 0.6416431 0.6416431
```

```
p <- predict(model.lr.capcolor,test,type = "response")
```

a-2. Set 0.5 as the threshold of the prediction data.

The result shows that the model has 59.73% accuracy. From the confusion table below, there are 718 mushrooms predicted edible which is poisson and 372 ones predicted poisson which actually are edible.

```
labels <- ifelse(p > 0.5, "p" , "e" )
tab2 = table("predict" = labels, "real" = test$class)
tab2
```

```
##        real
## predict    e    p
```

```
##        e 1030  718
##        p  372  587
```

```
sum(diag(tab2))/sum(tab2)
```

```
## [1] 0.5973402
```

b-1. Use cap-surface to predict

Next we use one variable **cap-surface** to fit the logistic regression. And made the prediction on the testing dataset.

```
model.lr.capsuface <- glm(class~`cap-surface`, data = train , family = "binomial")
model.lr.capsuface
```

```
##
## Call:  glm(formula = class ~ `cap-surface`, family = "binomial", data = train)
##
## Coefficients:
##    (Intercept)  `cap-surface`g  `cap-surface`s  `cap-surface`y
##        -0.7263         13.2923          0.9396          0.8745
##
## Degrees of Freedom: 5416 Total (i.e. Null);  5413 Residual
## Null Deviance:       7503
## Residual Deviance: 7283  AIC: 7291
```

```
summary(model.lr.capsuface)
```

```
##
## Call:
## glm(formula = class ~ `cap-surface`, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2693  -1.2410  -0.8883   1.1151   1.4972
```

```
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -0.72626    0.05420 -13.399   <2e-16 ***
## `cap-surface`g  13.29232  187.49087   0.071    0.943
## `cap-surface`s   0.93963    0.07290  12.890   <2e-16 ***
## `cap-surface`y   0.87454    0.06927  12.625   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7502.5  on 5416  degrees of freedom
## Residual deviance: 7282.7  on 5413  degrees of freedom
## AIC: 7290.7
##
## Number of Fisher Scoring iterations: 11
```

```
predict(model.lr.capsuface,test) %>% head()
```

```
##         1         2         3         4         5         6
## 0.2133719 0.1482822 0.1482822 0.1482822 0.2133719 0.1482822
```

```
predict(model.lr.capsuface,test,type = "response") %>% head()
```

```
##         1         2         3         4         5         6
## 0.5531415 0.5370028 0.5370028 0.5370028 0.5531415 0.5370028
```

```
p2 <- predict(model.lr.capsuface,test,type = "response")
```

b-2. Set 0.5 as the threshold of the prediction data

The result shows lower accuracy than the one we tested above which has only 57.85%. From the confusion table it displays, there are a lot of mis-predicted mushrooms.

```
labels.capsurface <- ifelse(p2 > 0.5,"p","e")
tab.capsurface = table("predict" = labels.capsurface , "real" = test$class) ; tab.caps
```

```
##         real
## predict    e    p
##       e  516  255
##       p  886 1050
```

```
sum(diag(tab.capsurface)) / sum(tab.capsurface)
```

```
## [1] 0.5785002
```

c-1. Use cap-shape to predict

And we use another variable **cap-shape** to fit the logistic regression model.

```
(model.lr.capshape <- glm(class ~`cap-shape` , data = train , family = "binomial"))
```

```
##
## Call:  glm(formula = class ~ `cap-shape`, family = "binomial", data = train)
##
## Coefficients:
##  (Intercept)  `cap-shape`c  `cap-shape`f  `cap-shape`k  `cap-shape`s
##       -2.065        16.631         2.052         2.980       -12.501
## `cap-shape`x
##        1.927
##
## Degrees of Freedom: 5416 Total (i.e. Null);  5411 Residual
## Null Deviance:       7503
## Residual Deviance: 7145  AIC: 7157
```

```
summary(model.lr.capshape)
```

```
##
## Call:
```

```
## glm(formula = class ~ `cap-shape`, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5823  -1.1196  -0.4888   1.1827   2.0900
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -2.0646     0.1821 -11.341   <2e-16 ***
## `cap-shape`c  16.6307   441.3717   0.038    0.970
## `cap-shape`f   2.0521     0.1873  10.958   <2e-16 ***
## `cap-shape`k   2.9797     0.2046  14.567   <2e-16 ***
## `cap-shape`s -12.5014   176.5488  -0.071    0.944
## `cap-shape`x   1.9272     0.1865  10.333   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7502.5  on 5416  degrees of freedom
## Residual deviance: 7145.0  on 5411  degrees of freedom
## AIC: 7157
##
## Number of Fisher Scoring iterations: 13
```

```
predict(model.lr.capshape,test) %>% head()
```

```
##         1         2         3         4         5         6
## -0.137471 -0.137471 -0.137471 -2.064626 -2.064626 -0.137471
```

```
predict(model.lr.capshape,test,type = "response") %>% head()
```

```
##         1         2         3         4         5         6
## 0.4656863 0.4656863 0.4656863 0.1125828 0.1125828 0.4656863
```

```
p3 <- predict(model.lr.capshape , test , type = "response")
```

c-2. Set 0.5 as the threshold of the prediction data

The accuracy on this model is even lower than the above two variables. The confusion table shows that there are many false prediction on the poisson mushrooms.

```
labels.capshape <- ifelse(p3 > 0.5 , "p" , "e")
tab.capshape <- table("predict" = labels.capshape , "real" = test$class)
tab.capshape
```

```
##         real
## predict    e    p
##       e 1335 1107
##       p   67  198
```

```
sum(diag(tab.capshape)) / sum(tab.capshape)
```

```
## [1] 0.5663096
```

d-1. Use habitat to predict

Next we use **habitat** to fit the logistic model because usually the habitat of poisson mushrooms and edible mushrooms are different. The dataset classify mushrooms into seven categories (grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d). And we use the model to predict on the testing dataset.

```
(model.lr.habitat <- glm(class~habitat, data = train , family = "binomial"))
```

```
##
## Call:  glm(formula = class ~ habitat, family = "binomial", data = train)
##
```

```
## Coefficients:
## (Intercept)     habitatg     habitatl     habitatm     habitatp     habitatu
##     -0.3904      -0.1961       1.2531      -1.5555       2.3332       1.4120
##    habitatw
##    -16.1757
##
## Degrees of Freedom: 5416 Total (i.e. Null);  5410 Residual
## Null Deviance:       7503
## Residual Deviance: 6383  AIC: 6397
```

```
summary(model.lr.habitat)
```

```
##
## Call:
## glm(formula = class ~ habitat, family = "binomial", data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.03803  -1.01675  -0.00036   1.34705   2.03933
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.39038    0.04452  -8.769  < 2e-16 ***
## habitatg     -0.19612    0.07057  -2.779  0.00545 **
## habitatl      1.25314    0.10296  12.171  < 2e-16 ***
## habitatm     -1.55553    0.22271  -6.984 2.86e-12 ***
## habitatp      2.33325    0.11893  19.618  < 2e-16 ***
## habitatu      1.41203    0.15352   9.198  < 2e-16 ***
## habitatw    -16.17569  208.06692  -0.078  0.93803
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7502.5  on 5416  degrees of freedom
## Residual deviance: 6382.6  on 5410  degrees of freedom
## AIC: 6396.6
##
## Number of Fisher Scoring iterations: 15
```

```
predict(model.lr.habitat,test) %>% head()
```

```
##          1          2          3          4          5          6
##  1.0216512  1.0216512 -0.5865013 -1.9459101 -1.9459101 -1.9459101
```

```
p.hab <- predict(model.lr.habitat, test, type = "response")
```

d-2. Set 0.5 as the threshold of the prediction data

The accuracy of this model is much higher than the above models which shows 70.26% of precision. The false predictions are mostly on the poisson mushrooms.

```
labels.hab <- ifelse(p.hab > 0.5 , "p" , "e")
tab.hab <- table("predict" = labels.hab , "real" = test$class);tab.hab
```

```
##        real
## predict    e    p
##       e 1252  655
##       p  150  650
```

```
sum(diag(tab.hab)) / sum(tab.hab)
```

```
## [1] 0.7026228
```

e-1. Use the four variables above to predict

Last we use all the variables above to fit the logistic regression. And make the prediction on the testing data.

```
(model.lr.all <- glm(class~`cap-shape` + `cap-surface` + `cap-color` + habitat , data
```

```
##
## Call:  glm(formula = class ~ `cap-shape` + `cap-surface` + `cap-color` +
##     habitat, family = "binomial", data = train)
##
## Coefficients:
##    (Intercept)    `cap-shape`c    `cap-shape`f    `cap-shape`k    `cap-shape`s
##        16.1262         19.0052          2.6457          3.7021        -16.5254
##   `cap-shape`x  `cap-surface`g  `cap-surface`s  `cap-surface`y    `cap-color`c
##         2.5448         17.5571          1.1495          0.9185        -21.9517
##   `cap-color`e    `cap-color`g    `cap-color`n    `cap-color`p    `cap-color`r
##       -19.8543        -19.6573        -20.4676        -15.4110        -38.2009
##   `cap-color`u    `cap-color`w    `cap-color`y       habitatg       habitatl
##       -38.1896        -20.0833        -18.4134         -0.3223          1.3333
##       habitatm        habitatp        habitatu       habitatw
##        -2.6270          2.0988          1.9013        -36.7961
##
## Degrees of Freedom: 5416 Total (i.e. Null);  5393 Residual
## Null Deviance:       7503
## Residual Deviance: 5376  AIC: 5424
```

```
summary(model.lr.all)
```

```
##
## Call:
## glm(formula = class ~ `cap-shape` + `cap-surface` + `cap-color` +
##     habitat, family = "binomial", data = train)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.6361  -0.8312   0.0000   0.7539   3.3672
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)      16.12620  347.68810   0.046 0.963006
## `cap-shape`c     19.00517 2951.66220   0.006 0.994863
## `cap-shape`f      2.64573    0.25839  10.239  < 2e-16 ***
## `cap-shape`k      3.70210    0.27906  13.266  < 2e-16 ***
## `cap-shape`s    -16.52538 1279.70305  -0.013 0.989697
## `cap-shape`x      2.54484    0.25662   9.917  < 2e-16 ***
## `cap-surface`g   17.55705 3429.78595   0.005 0.995916
## `cap-surface`s    1.14953    0.09729  11.816  < 2e-16 ***
## `cap-surface`y    0.91850    0.08390  10.948  < 2e-16 ***
## `cap-color`c    -21.95168  347.68843  -0.063 0.949658
## `cap-color`e    -19.85435  347.68818  -0.057 0.954462
## `cap-color`g    -19.65728  347.68817  -0.057 0.954914
## `cap-color`n    -20.46758  347.68817  -0.059 0.953058
## `cap-color`p    -15.41104  347.68850  -0.044 0.964646
## `cap-color`r    -38.20089 2200.95440  -0.017 0.986152
## `cap-color`u    -38.18964 1914.06737  -0.020 0.984082
## `cap-color`w    -20.08334  347.68817  -0.058 0.953938
## `cap-color`y    -18.41341  347.68815  -0.053 0.957764
## habitatg         -0.32233    0.09192  -3.507 0.000454 ***
## habitatl          1.33328    0.12588  10.592  < 2e-16 ***
## habitatm         -2.62701    0.32480  -8.088 6.06e-16 ***
## habitatp          2.09884    0.12502  16.788  < 2e-16 ***
## habitatu          1.90127    0.20074   9.471  < 2e-16 ***
```

```
## habitatw        -36.79605  522.90657  -0.070 0.943901
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7502.5  on 5416  degrees of freedom
## Residual deviance: 5375.5  on 5393  degrees of freedom
## AIC: 5423.5
##
## Number of Fisher Scoring iterations: 17
```

```
predict(model.lr.all , test) %>% head()
```

```
##          1          2          3          4          5          6
##  1.2542638  1.4074708  0.8538065 -5.6656483 -3.7646852 -1.4508714
```

```
predict(model.lr.all, test, type = "response") %>% head()
```

```
##           1           2           3           4           5           6
## 0.778037075 0.803366720 0.701365043 0.003450952 0.022649995 0.189867487
```

```
p.all <- predict(model.lr.all, test, type = "response")
```

e-2. Set 0.5 as the threshold of the prediction data

With these four variables we have accuracy up to 77.09% which is higher than any of the accuracy of prediction above.

```
labels.all <- ifelse(p.all > 0.5 , "p", "e")
(tab.all <- table("predict" = labels.all , "real" = test$class))
```

```
##        real
## predict    e    p
##       e 1151  369
```

```
##       p  251  936
```

```
sum(diag(tab.all)) / sum(tab.all)
```

```
## [1] 0.7709642
```

E. Multiple Logistic Regression

Predict habitat

Try to use Multiple Logistic Regression to predict habitat of these mushrooms. Because the dataset only offer categorical-type variables, we turn the some variables into numeric form to predict habitat. And check the coefficient of the model we create.

```
train.c <- train
train.c$`cap-shape` <- as.numeric(train.c$`cap-shape`)
train.c$`cap-surface` <- as.numeric(train.c$`cap-surface`)
train.c$`cap-color` <- as.numeric(train.c$`cap-color`)
xtrain <- as.matrix(train.c[,2:4])
ytrain <- as.matrix(train[,21])
model.mlr <- glmnet(xtrain , ytrain , family = "multinomial" ,lambda = 0)
coef(model.mlr)
```

```
## $d
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
##              2.7083446
## cap-shape    0.1141624
## cap-surface -0.3752082
## cap-color   -0.1265491
##
## $g
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
```

```
##              1.45267667
## cap-shape   -0.01547577
## cap-surface -0.46706688
## cap-color    0.17431690
##
## $l
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
##              1.2904877
## cap-shape   -0.1081806
## cap-surface  0.1316920
## cap-color   -0.2346767
##
## $m
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
##             -5.4010681
## cap-shape   -0.2375503
## cap-surface  0.3490588
## cap-color    0.5702815
##
## $p
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
##             0.14709248
## cap-shape   0.06524749
## cap-surface .
## cap-color   .
##
## $u
```

```
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
##             -0.004265329
## cap-shape    0.101947184
## cap-surface -0.406266768
## cap-color    0.005013482
##
## $w
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
##             -0.1932680
## cap-shape    .
## cap-surface  0.3136678
## cap-color   -0.4908828
```

The prediction of the model has only 46.17% accuracy, we can perceive from the table below. The prediction only make three categories but it actually has seven categories.

```
test.c <- test
test.c$`cap-shape` <- as.numeric(test.c$`cap-shape`)
test.c$`cap-surface` <- as.numeric(test.c$`cap-surface`)
test.c$`cap-color` <- as.numeric(test.c$`cap-color`)


xtest <- as.matrix(test.c[,2:4])
ytest <- as.matrix(test[,21])
p.mlr <- predict(model.mlr , xtest , type = "response")
labels.mlr <- predict(model.mlr, xtest, type = "class")
(tab.mlr <- table("predict" = labels.mlr , "real" = ytest))
```

```
##        real
## predict   d   g   l   m   p   u   w
##       d 906 343 273   6 307  95  59
```

```
##       g 146 343   2  79  87  35   0
##       m   0  10   1  15   0   0   0
```

```
sum(diag(tab.mlr)) / sum(tab.mlr)
```

```
## [1] 0.4617658
```

F. Support Vector Machine

a. Use all variable to predict

Using all variable to fit the SVM model. The result shows that it can be perfectly separated by lines.

```
(model.svm <- svm(class~. , data = train, kernel = "linear"))
```

```
##
## Call:
## svm(formula = class ~ ., data = train, kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  210
```

```
predict(model.svm , test) %>% head()
```

```
## 1 2 3 4 5 6
## p p e e e e
## Levels: e p
```

```
tab.svm <- table("predict" = predict(model.svm , test) , "real" = test$class)
tab.svm
```

```
##         real
## predict    e    p
##       e 1402    0
##       p    0 1305
```

```
sum(diag(tab.svm)) / sum(tab.svm)
```

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

b. Use cap-shape, cap-surface, cap-color, habitat to predict

Then we try to use the variables we used to fit logistic regression to fit SVM model. The result gives a 76.99% of accuracy on this model.

```
(model.svm.cap <- svm(class~`cap-color`+`cap-surface`+`cap-shape`+habitat , data = tra
```

```
##
## Call:
## svm(formula = class ~ `cap-color` + `cap-surface` + `cap-shape` +
##     habitat, data = train, kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  2843
```

```
(tab.svm.cap <- table("predict" = predict(model.svm.cap , test) , "real" = test$class)
```

```
##         real
```

```
## predict    e    p
##       e 1182  403
##       p  220  902
```

```
sum(diag(tab.svm.cap))/sum(tab.svm.cap)
```

```
## [1] 0.7698559
```

c. changing kernel type to radial

Trying to improve the accuracy of the model, we use different type of kernel to separate the data. The accuracy of the model using radial kernel type is higher than the model using linear kernel type which is 83.82%.

```
(model.svm.cap.k <- svm(class~`cap-color`+`cap-surface`+`cap-shape`+habitat , data = t
```

```
##
## Call:
## svm(formula = class ~ `cap-color` + `cap-surface` + `cap-shape` +
##     habitat, data = train, kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  2740
```

```
(tab.svm.cap.k <- table("predict" = predict(model.svm.cap.k , test) , "real" = test$cl
```

```
##        real
## predict    e    p
##       e 1257  293
##       p  145 1012
```

```
sum(diag(tab.svm.cap.k))/sum(tab.svm.cap.k)
```

```
## [1] 0.8381973
```

G. Nonparametric Classification

a. Use cap-shape, cap-surface, cap-color, habitat to predict

Use the four variables above to fit the model of nonparametric classification. The accuracy using the unweighted NN algorithms is 78.79%.

```
(model.kknn <- kknn(class~`cap-color`+`cap-surface`+`cap-surface`+habitat, train , tes
```

```
##
## Call:
## kknn(formula = class ~ `cap-color` + `cap-surface` + `cap-surface` +     habitat, tra
##
## Response: "nominal"
```

```
model.kknn$fitted.values %>% head()
```

```
## [1] p p e e e e
## Levels: e p
```

```
(tab.kknn <- table("predict" = model.kknn$fitted.values , "real" = test$class))
```

```
##         real
## predict    e    p
##       e 1329  501
##       p   73  804
```

```
sum(diag(tab.kknn)) / sum(tab.kknn)
```

```
## [1] 0.7879571
```

b. try k=4

Then we try to change the k value to see if it can predict more precisely. The result comes out with an accuracy of 79.31% which is slightly higher than the one with k equal 7.

```
(model.kknn5 <- kknn(class~`cap-color`+`cap-surface`+`cap-surface`+habitat, train , te
```

```
##
## Call:
## kknn(formula = class ~ `cap-color` + `cap-surface` + `cap-surface` +     habitat, tra
##
## Response: "nominal"
```

```
model.kknn5$fitted.values %>% head()
```

```
## [1] p p e e e e
## Levels: e p
```

```
(tab.kknn5 <- table("predict" = model.kknn5$fitted.values , "real" = test$class))
```

```
##         real
## predict    e    p
##       e 1333  491
##       p   69  814
```

```
sum(diag(tab.kknn5)) / sum(tab.kknn5)
```

```
## [1] 0.7931289
```

c. experience with another kernel type

Try different kernel type to separate the data. The result shows that the accuracy of this model is 79.31%.

```
(model.gaussian <- kknn(class~`cap-color`+`cap-surface`+`cap-surface`+habitat, train ,
```

```
##
## Call:
```

```
## kknn(formula = class ~ `cap-color` + `cap-surface` + `cap-surface` +     habitat, tra
##
## Response: "nominal"
```

```
(tab.gaussian <- table("predict" = model.gaussian$fitted.values , "real" = test$class)
```

```
##         real
## predict    e    p
##       e 1333  491
##       p   69  814
```

```
sum(diag(tab.gaussian))/sum(tab.gaussian)
```

```
## [1] 0.7931289
```

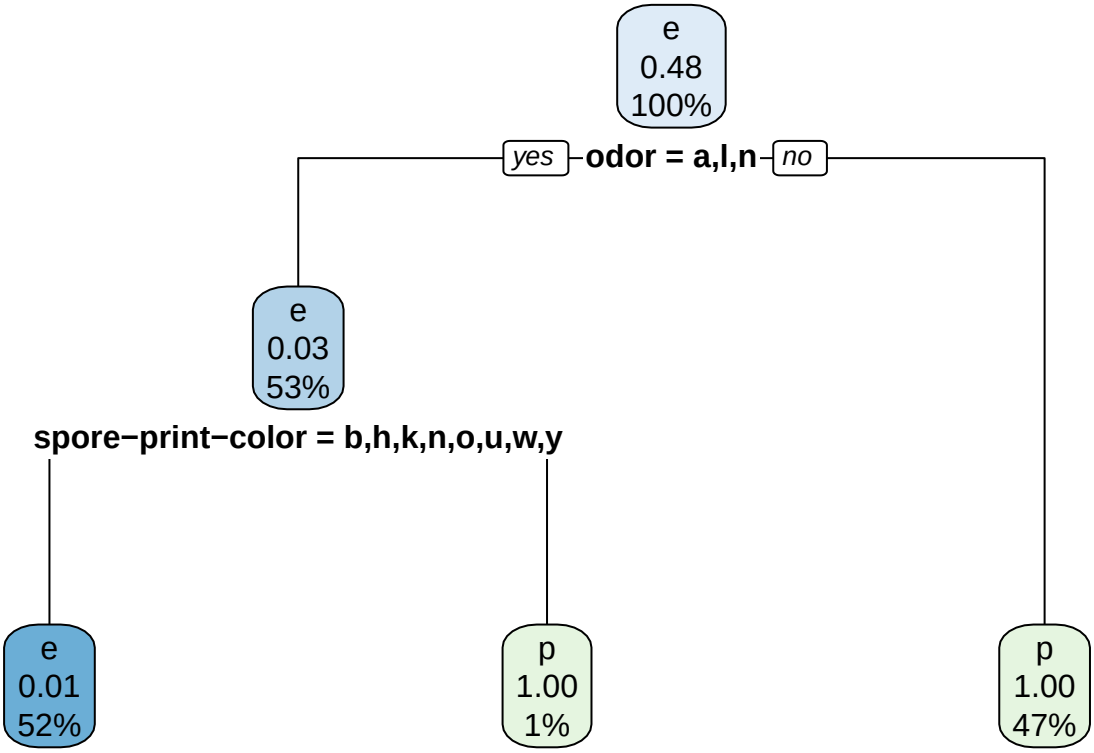
H. Decision Tree

Make decision tree on this data set and predict on the test data set. The accuracy is 99.48%.

```
model.decision <- rpart(class~., data = train)
model.decision
```

```
## n= 5417
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 5417 2611 e (0.51799889 0.48200111)
##   2) odor=a,l,n 2888   82 e (0.97160665 0.02839335)
##     4) spore-print-color=b,h,k,n,o,u,w,y 2840   34 e (0.98802817 0.01197183) *
##     5) spore-print-color=r 48    0 p (0.00000000 1.00000000) *
##   3) odor=c,f,m,p,s,y 2529    0 p (0.00000000 1.00000000) *
```

```
rpart.plot(model.decision)
```




```
tab.decision <- table("predict" = predict(model.decision, test , type = "class") , "re
tab.decision
```

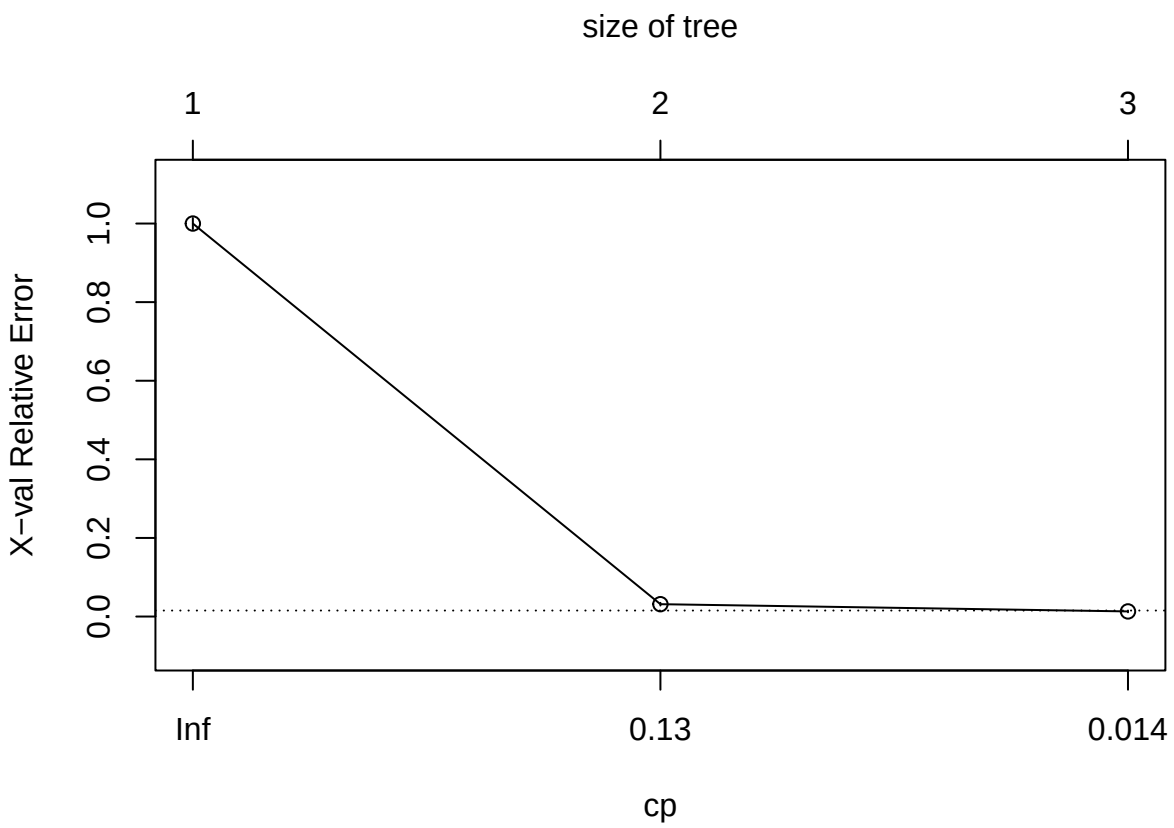
```
##         real
## predict    e    p
##       e 1402   14
##       p    0 1291
```

```
sum(diag(tab.decision))/sum(tab.decision)
```

```
## [1] 0.9948282
```

Prune Tree

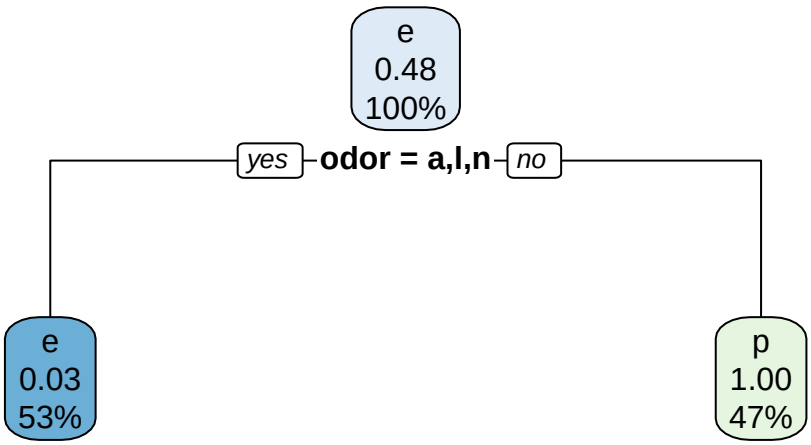
```
plotcp(model.decision)
```




```
model.cp <- rpart(class~. , data = train , cp = 0.13)
model.cp
```

```
## n= 5417
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 5417 2611 e (0.51799889 0.48200111)
##   2) odor=a,l,n 2888   82 e (0.97160665 0.02839335) *
##   3) odor=c,f,m,p,s,y 2529    0 p (0.00000000 1.00000000) *
```

```
rpart.plot(model.cp)
```




```
tab.cp <- table("predict" = predict(model.cp , test , type = "class") , "real" = test$
tab.cp
```

```
##         real
## predict    e    p
##       e 1402   38
##       p    0 1267
```

```
sum(diag(tab.cp))/sum(tab.cp)
```

```
## [1] 0.9859623
```