PeerReview_3_MinxinCheng

0. load packages

```
#install.packages("class")
#install.packages("gmodels")
#install.packages("caret")

library(class)
library(gmodels)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following object is masked from 'package:gmodels':
##
##     ci
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

0. read in dataset

```
pscData <- read.csv("prostate_cancer.csv",
                    stringsAsFactors = FALSE)
```

1. preparing the dataset 1.1 check the dataset

```
# check if the data is structured or not
str(pscData)
```

```
## 'data.frame':    100 obs. of  10 variables:
##  $ id                : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ diagnosis_result : chr  "M" "B" "M" "M" ...
##  $ radius            : int  23 9 21 14 9 25 16 15 19 25 ...
##  $ texture           : int  12 13 27 16 19 25 26 18 24 11 ...
##  $ perimeter         : int  151 133 130 78 135 83 120 90 88 84 ...
##  $ area              : int  954 1326 1203 386 1297 477 1040 578 520 476 ...
##  $ smoothness        : num  0.143 0.143 0.125 0.07 0.141 0.128 0.095 0.119 0.127 0.119 ...
##  $ compactness       : num  0.278 0.079 0.16 0.284 0.133 0.17 0.109 0.165 0.193 0.24 ...
##  $ symmetry          : num  0.242 0.181 0.207 0.26 0.181 0.209 0.179 0.22 0.235 0.203 ...
##  $ fractal_dimension: num  0.079 0.057 0.06 0.097 0.059 0.076 0.057 0.075 0.074 0.082 ...
```

```r
# remove the id column
pscData <- pscData[-1]
```

1.2 get the information of the number of patients

```r
# add a column of diagnosis with the same information in the diagnosis_result column
pscData$diagnosis <- factor(pscData$diagnosis_result,
                            levels = c("B", "M"),
                            labels = c("Benign", "Malignant"))

# get the number of patients
table(pscData$diagnosis_result)
```

```
##
##  B  M
## 38 62
```

```r
# round the result in percentage form to 1 decimal place
round(prop.table(table(pscData$diagnosis)) * 100,
      digits = 1)
```

```
##
##    Benign Malignant
##        38        62
```

1.3 normalize the dataset

```r
# function for normalization
normalize <- function(x){
  return((x - min(x)) / (max(x) - min(x)))
}

# normalize the dataset expect the diagnosis
pscData_n <- as.data.frame(lapply(pscData[,2:9], normalize))
summary(pscData_n)
```

```
##      radius          texture         perimeter           area
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.1875   1st Qu.:0.1875   1st Qu.:0.2542   1st Qu.:0.1639
```

```
##  Median :0.5000   Median :0.4062   Median :0.3500   Median :0.2637
##  Mean   :0.4906   Mean   :0.4519   Mean   :0.3732   Mean   :0.2989
##  3rd Qu.:0.7500   3rd Qu.:0.7031   3rd Qu.:0.5188   3rd Qu.:0.4266
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##    smoothness       compactness        symmetry      fractal_dimension
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.3219   1st Qu.:0.1384   1st Qu.:0.2189   1st Qu.:0.1364
##  Median :0.4384   Median :0.2622   Median :0.3254   Median :0.2273
##  Mean   :0.4484   Mean   :0.2889   Mean   :0.3442   Mean   :0.2657
##  3rd Qu.:0.5753   3rd Qu.:0.3876   3rd Qu.:0.4379   3rd Qu.:0.3636
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

2. split the data

```
pscData_train <- pscData_n[1:65,]
pscData_test <- pscData_n[66:100,]

pscData_trail_labels <- pscData[1:65, 1]
pscData_test_label <- pscData[66:100, 1]
```

3. train a model on data

```
set.seed(123)
pscData_test_pred <- knn(train = pscData_train,
                         test = pscData_test,
                         cl = pscData_trail_labels,
                         k = 10)
```

4. Evaluate the model's performance

```
CrossTable(x = pscData_test_label,
           y = pscData_test_pred,
           prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  35
##
##
##                    | pscData_test_pred
## pscData_test_label |         B |         M | Row Total |
## -------------------|-----------|-----------|-----------|
```

```
##                   B |         7 |        12 |        19 |
##                     |     0.368 |     0.632 |     0.543 |
##                     |     0.875 |     0.444 |           |
##                     |     0.200 |     0.343 |           |
## -------------------|-----------|-----------|-----------|
##                   M |         1 |        15 |        16 |
##                     |     0.062 |     0.938 |     0.457 |
##                     |     0.125 |     0.556 |           |
##                     |     0.029 |     0.429 |           |
## -------------------|-----------|-----------|-----------|
##       Column Total |         8 |        27 |        35 |
##                     |     0.229 |     0.771 |           |
## -------------------|-----------|-----------|-----------|
##
##
```

The test data consisted of 35 observations. 7 predictions are true negatives 15 are true positive. 1 of them was false negatives meaning it was actually malignant in nature but got predicted as benign. 13 were false positive meaning they were benign in nature but got predicted as malignant.

The overall accuracy of the model is 62.85% (7 + 15) / 35.

5. try different k values see if there is any better resultss

```
set.seed(123)
pscData_test_pred_2 <- knn(train = pscData_train,
                           test = pscData_test,
                           cl = pscData_trail_labels,
                           k = 9)
CrossTable(x = pscData_test_label,
           y = pscData_test_pred_2,
           prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  35
##
##
##                     | pscData_test_pred_2
## pscData_test_label |         B |         M | Row Total |
## -------------------|-----------|-----------|-----------|
##                   B |         8 |        11 |        19 |
##                     |     0.421 |     0.579 |     0.543 |
##                     |     1.000 |     0.407 |           |
```

```
##                    |     0.229 |     0.314 |           |
## -------------------|-----------|-----------|-----------|
##                  M |         0 |        16 |        16 |
##                    |     0.000 |     1.000 |     0.457 |
##                    |     0.000 |     0.593 |           |
##                    |     0.000 |     0.457 |           |
## -------------------|-----------|-----------|-----------|
##       Column Total |         8 |        27 |        35 |
##                    |     0.229 |     0.771 |           |
## -------------------|-----------|-----------|-----------|
##
##
```

I tried few different k value around 10 (8, 9, 11, and 12) and found k = 9 can improve the accuracy slightly (8 + 16) / 35 = 68.6%

6. using creat package to do a knn training 6.1 prepare the dataset

```
# extract all diagnose information
labels <- pscData[1:100, 1]
# combine it with the normalized data created before
pscDataNorm <- cbind(pscData_n, labels)
```

6.2 split the dataset

```
set.seed(123)
inTraining <- createDataPartition(pscDataNorm$labels,
                                  p = 0.65,
                                  list = FALSE)

pscDataTraining <- pscDataNorm[inTraining, ]
pscDataTesting <- pscDataNorm[-inTraining, ]

# check distribution in original data, training data, and testing data
prop.table(table(pscDataTraining$labels)) * 100
```

```
##
##        B        M
## 37.87879 62.12121
```

```
prop.table(table(pscDataTesting$labels)) * 100
```

```
##
##        B        M
## 38.23529 61.76471
```

```
prop.table(table(pscDataNorm$labels)) * 100
```

```
##
##  B  M
## 38 62
```

6.3 split training data to training data and training control data

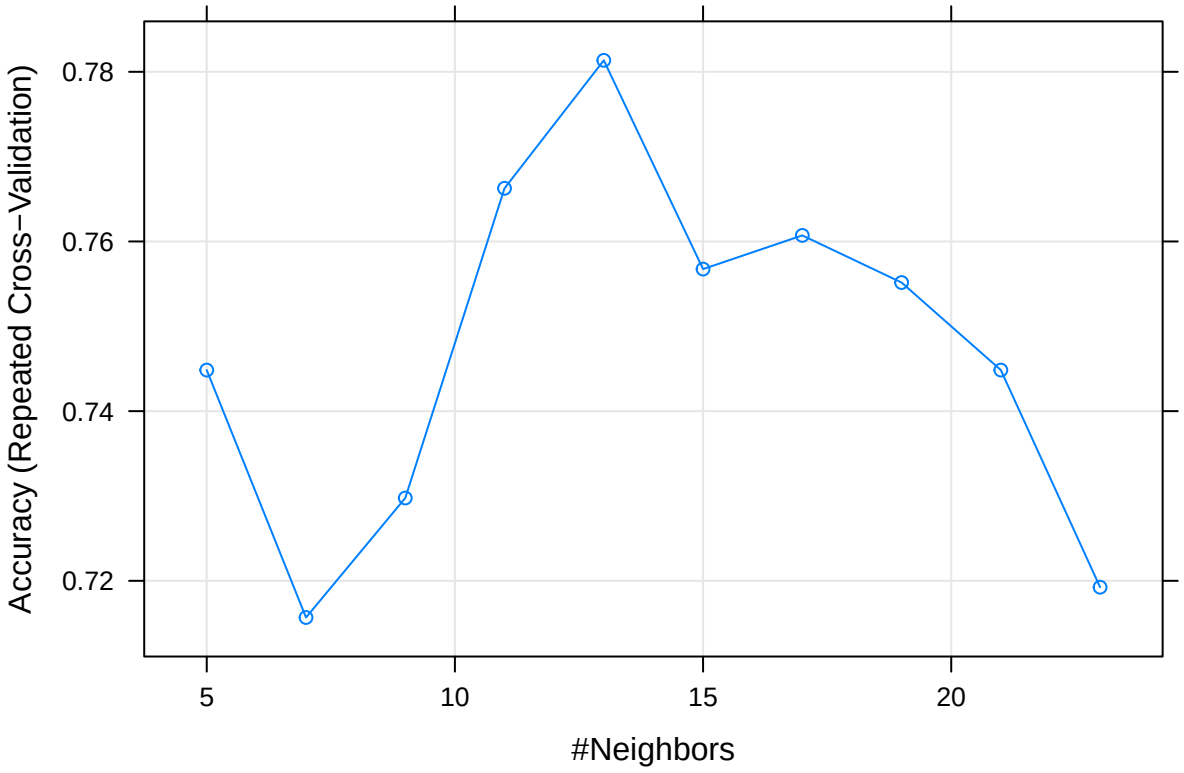
```
set.seed(123)

# specify that it is a 3-fold cross-validation and there will be 20 possible k values to evaluate
pscDataCtrl_1 <- trainControl(method = "repeatedcv",
                              repeats = 3)
knnFit_1 <- train(labels ~ .,
                  data = pscDataTraining,
                  method = "knn",
                  trControl = pscDataCtrl_1,
                  tuneLength = 10)
knnFit_1
```

```
## k-Nearest Neighbors
##
## 66 samples
##  8 predictor
##  2 classes: 'B', 'M'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 59, 59, 60, 60, 59, 60, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.7448413  0.4291317
##    7  0.7156746  0.3716280
##    9  0.7297619  0.3801376
##   11  0.7662698  0.4539472
##   13  0.7813492  0.4824838
##   15  0.7567460  0.4230290
##   17  0.7607143  0.4379736
##   19  0.7551587  0.4218954
##   21  0.7448413  0.3936840
##   23  0.7192460  0.3285265
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 13.
```

```
knnFit_1$bestTune
```

```
##    k
## 5 13
```

```
# plotting yields Number of Neighbor vs Accuracy based on repeated cross validation
plot(knnFit_1)
```



As the result, the best tuning parameter k that maximizes model accuracy is 5

6.4 Make prediction on the test data

```
knnPredict_1 <- predict(knnFit_1,
                        newdata = pscDataTesting)
head(knnPredict_1)
```

```
## [1] M M M M M M
## Levels: B M
```

```
# compute the accuracy rate
mean(knnPredict_1 == pscDataTesting$labels)
```

```
## [1] 0.8823529
```

The overall prediction accuracy of our model is 85.3%.

6.5 Get the confusion matrix to see accuracy value and other parameter values

```
# confusion matrix for caret package result
confusionMatrix(table(knnPredict_1,
                      pscDataTesting$labels))
```

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

```
##
## knnPredict_1  B  M
##            B  9  0
##            M  4 21
##
##                Accuracy : 0.8824
##                  95% CI : (0.7255, 0.967)
##     No Information Rate : 0.6176
##     P-Value [Acc > NIR] : 0.00065
##
##                   Kappa : 0.7354
##
##  Mcnemar's Test P-Value : 0.13361
##
##             Sensitivity : 0.6923
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.8400
##              Prevalence : 0.3824
##          Detection Rate : 0.2647
##    Detection Prevalence : 0.2647
##       Balanced Accuracy : 0.8462
##
##        'Positive' Class : B
##
```

```
# also plotting confusion matrix for class package
confusionMatrix(table(pscData_test_pred,
                      pscData_test_label))
```

```
## Confusion Matrix and Statistics
##
##                  pscData_test_label
## pscData_test_pred  B  M
##                 B  7  1
##                 M 12 15
##
##                Accuracy : 0.6286
##                  95% CI : (0.4492, 0.7853)
##     No Information Rate : 0.5429
##     P-Value [Acc > NIR] : 0.198713
##
##                   Kappa : 0.2902
##
##  Mcnemar's Test P-Value : 0.005546
##
##             Sensitivity : 0.3684
##             Specificity : 0.9375
##          Pos Pred Value : 0.8750
##          Neg Pred Value : 0.5556
##              Prevalence : 0.5429
##          Detection Rate : 0.2000
##    Detection Prevalence : 0.2286
##       Balanced Accuracy : 0.6530
```

```
##
##        'Positive' Class : B
##
```

The test data consisted of 34 observations. 20 predictions are true negatives 9 are true positive. 1 of them was false negatives meaning it was actually malignant in nature but got predicted as benign. 4 were false positive meaning they were benign in nature but got predicted as malignant. The overall accuracy is 85.3%, it is better than class package 62.86%.

6.6 Applying random forest to see if the performance improves

```
set.seed(123)
pscDataCtrl_2 <- trainControl(method = "repeatedcv",
                              repeats = 3)

# random forest
pscDataRfFit <- train(labels ~ ., data = pscDataTraining,
                      method = "rf",
                      trControl = pscDataCtrl_2,
                      tuneLength = 10)
```

```
## note: only 7 unique complexity parameters in default grid. Truncating the grid to 7 .
```

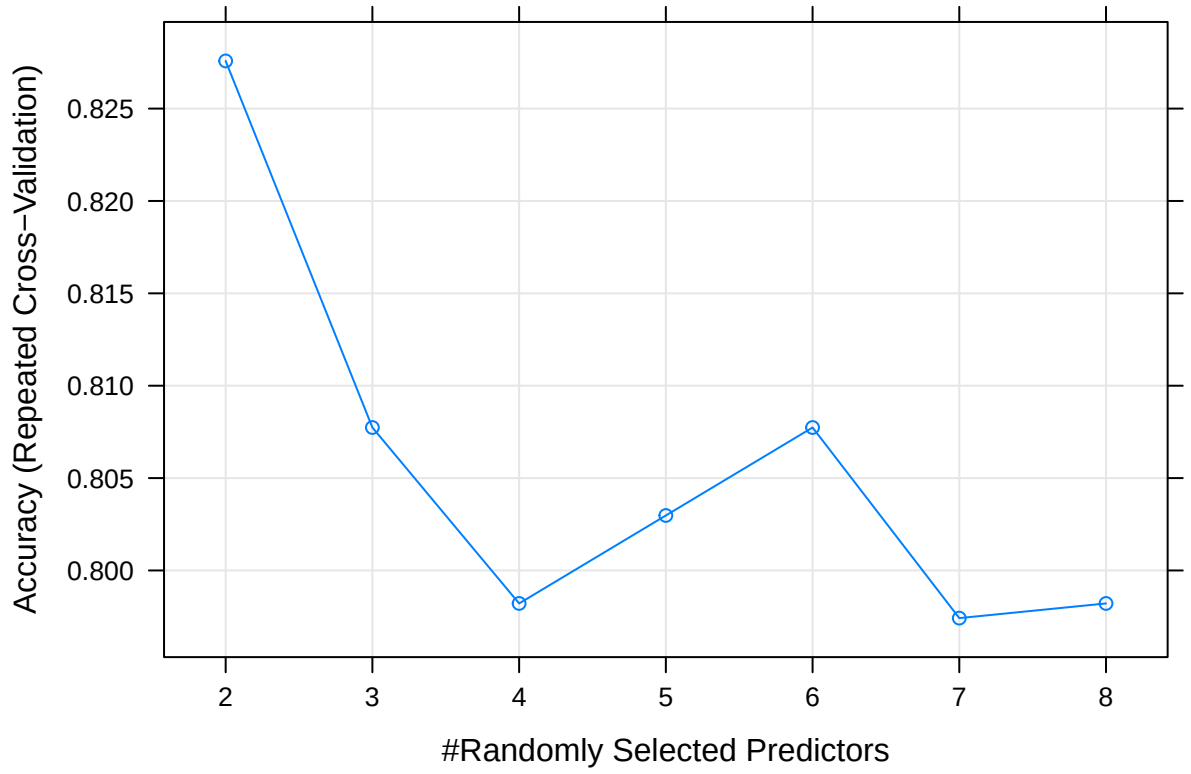
```
pscDataRfFit
```

```
## Random Forest
##
## 66 samples
##  8 predictor
##  2 classes: 'B', 'M'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 59, 59, 60, 60, 59, 60, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.8275794  0.6300743
##   3     0.8077381  0.5902732
##   4     0.7982143  0.5692654
##   5     0.8029762  0.5784881
##   6     0.8077381  0.5895553
##   7     0.7974206  0.5594405
##   8     0.7982143  0.5692654
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
pscDataRfFit$bestTune
```

```
##   mtry
## 1    2
```

```
plot(pscDataRfFit)
```



6.7 Make prediction on the test data

```
pscDataRfPredict <- predict(pscDataRfFit,
                            newdata = pscDataTesting)
```

6.8 evaluate the model performance

```
mean(pscDataRfPredict == pscDataTesting$labels)
```

```
## [1] 0.9117647
```

```
confusionMatrix(table(pscDataRfPredict,
                      pscDataTesting$labels))
```

```
## Confusion Matrix and Statistics
##
##
## pscDataRfPredict  B  M
##                B 12  2
##                M  1 19
##
##                Accuracy : 0.9118
```

```
##                  95% CI : (0.7632, 0.9814)
##     No Information Rate : 0.6176
##     P-Value [Acc > NIR] : 0.0001272
##
##                   Kappa : 0.8159
##
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.9231
##             Specificity : 0.9048
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 0.9500
##              Prevalence : 0.3824
##          Detection Rate : 0.3529
##    Detection Prevalence : 0.4118
##       Balanced Accuracy : 0.9139
##
##        'Positive' Class : B
##
```

As the result, the accuracy slightly improved to 88.24%.