Liver Disease Analysis

Sean Cerulean Johnson

2021-11-13

Library

```
library(dplyr)
```

```
## Warning: package ’dplyr’ was built under R version 4.1.3

##
## Attaching package: ’dplyr’

## The following objects are masked from ’package:stats’:
##
##     filter, lag

## The following objects are masked from ’package:base’:
##
##     intersect, setdiff, setequal, union
```

Data

```
#data was downloaded from the IC repository
hcv = readr::read_csv(here::here("data/hcvdat0.csv"))
```

```
## New names:
## Rows: 615 Columns: 14
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (2): Category, Sex dbl (12): ...1, Age, ALB, ALP, ALT, AST, BIL, CHE, CHOL,
## CREA, GGT, PROT
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## * ‘‘ -> ‘...1‘
```

Quick Overview

```r
summary(hcv)
```

```
##       ...1          Category              Age            Sex           
##  Min.   :  1.0   Length:615         Min.   :19.00   Length:615        
##  1st Qu.:154.5   Class :character   1st Qu.:39.00   Class :character  
##  Median :308.0   Mode  :character   Median :47.00   Mode  :character  
##  Mean   :308.0                      Mean   :47.41                     
##  3rd Qu.:461.5                      3rd Qu.:54.00                     
##  Max.   :615.0                      Max.   :77.00                     
##                                                                       
##       ALB             ALP              ALT              AST        
##  Min.   :14.90   Min.   : 11.30   Min.   :  0.90   Min.   : 10.60  
##  1st Qu.:38.80   1st Qu.: 52.50   1st Qu.: 16.40   1st Qu.: 21.60  
##  Median :41.95   Median : 66.20   Median : 23.00   Median : 25.90  
##  Mean   :41.62   Mean   : 68.28   Mean   : 28.45   Mean   : 34.79  
##  3rd Qu.:45.20   3rd Qu.: 80.10   3rd Qu.: 33.08   3rd Qu.: 32.90  
##  Max.   :82.20   Max.   :416.60   Max.   :325.30   Max.   :324.00  
##  NA's   :1       NA's   :18       NA's   :1                        
##       BIL             CHE              CHOL            CREA        
##  Min.   :  0.8   Min.   : 1.420   Min.   :1.430   Min.   :   8.00  
##  1st Qu.:  5.3   1st Qu.: 6.935   1st Qu.:4.610   1st Qu.:  67.00  
##  Median :  7.3   Median : 8.260   Median :5.300   Median :  77.00  
##  Mean   : 11.4   Mean   : 8.197   Mean   :5.368   Mean   :  81.29  
##  3rd Qu.: 11.2   3rd Qu.: 9.590   3rd Qu.:6.060   3rd Qu.:  88.00  
##  Max.   :254.0   Max.   :16.410   Max.   :9.670   Max.   :1079.10  
##                                   NA's   :10                       
##       GGT              PROT      
##  Min.   :  4.50   Min.   :44.80  
##  1st Qu.: 15.70   1st Qu.:69.30  
##  Median : 23.30   Median :72.20  
##  Mean   : 39.53   Mean   :72.04  
##  3rd Qu.: 40.20   3rd Qu.:75.40  
##  Max.   :650.90   Max.   :90.00  
##                   NA's   :1
```

Data Wrangling

Removing NAs

```r
df = as.data.frame(na.omit(hcv))
1 - nrow(df)/nrow(hcv)
```

```
## [1] 0.04227642
```

Renaming

```r
#cleaning
df<-df%>%
  mutate(Category = case_when(Category == "0=Blood Donor" ~0,
                              Category == "0s=suspect Blood Donor" ~1,
                              Category == "1=Hepatitis" ~2,
                              Category == "2=Fibrosis" ~3,
                              Category == "3=Cirrhosis" ~4),
         Sex = case_when(Sex=="m" ~0,
                         Sex=="f" ~1))
```

Drop ID Col

```r
df <-df%>%
  select(-...1)
```

Formatting

```r
df[,1] = as.numeric(df[,1])
df[,3] = as.numeric(df[,3])
```

```r
ind0 = (df$Category==0)
ind1 = (df$Category==1)
ind = ind0 | ind1
self = df[ind,]
bad = df[ind1,]
allBad = df[!ind0,]
#checks
nrow(allBad)/nrow(df)
```

```
## [1] 0.106961
```

```r
nrow(bad)/nrow(df)
```

```
## [1] 0.01188455
```

Subset of DF

```r
x = df
x<-x%>%
  mutate(Category = case_when(Category == 2 ~1,
                              Category == 3 ~1,
                              Category == 4 ~1,
                              TRUE ~0))
```

Table for Predictive Modeling

```
# table
table_accuracy = matrix(nrow=6,ncol=3)
colnames(table_accuracy) = c('Accuracy','Precision','Recall')
rownames(table_accuracy) = c('DTree','NB','SVM-Linerar','SVM-Polynomial','ANN','KNN')
table_accuracy
```

```
##                Accuracy Precision Recall
## DTree                NA        NA     NA
## NB                   NA        NA     NA
## SVM-Linerar          NA        NA     NA
## SVM-Polynomial       NA        NA     NA
## ANN                  NA        NA     NA
## KNN                  NA        NA     NA
```

K-fold Cross-Validation

```
n = nrow(x)
k = 10
tail = n%/%k

set.seed(2)

rnd = runif(n)
rank = rank(rnd)
blk = (rank-1)%/%tail+1
blk = as.factor(blk)

print(summary(blk))
```

```
##  1  2  3  4  5  6  7  8  9 10 11
## 58 58 58 58 58 58 58 58 58 58  9
```

Predictive Models

Decision Tree

```
set.seed(2)

all.acc = numeric(0)
all.pre = numeric(0)
all.rec = numeric(0)
for(i in 1:k){
  tree = rpart::rpart(Category~.,x[blk != i,],method="class")
  pred = predict(tree,x[blk==i,],type="class")
  confMat = table(pred,x$Category[blk==i])
  acc = (confMat[1,1]+confMat[2,2])/sum(confMat)
  pre = (confMat[1,1])/sum(confMat[1,])
  rec = (confMat[1,1])/sum(confMat[,1])
  all.acc = rbind(all.acc,acc)
  all.pre = rbind(all.pre,pre)
  all.rec = rbind(all.rec,rec)
}
j=1
print(mean(all.acc))
```

```
## [1] 0.9465517
```

```
print(mean(all.pre))
```

```
## [1] 0.9658513
```

```
print(mean(all.rec))
```

```
## [1] 0.9750259
```

```
table_accuracy[j,1] = mean(all.acc)
table_accuracy[j,2] = mean(all.pre)
table_accuracy[j,3] = mean(all.rec)
```

Naive Bayes

```
set.seed(2)

all.acc = numeric(0)
all.pre = numeric(0)
all.rec = numeric(0)
for(i in 1:k){
  model = e1071::naiveBayes(Category~.,x[blk != i,],method="class")
  pred = predict(model,x[blk==i,],type="class")
  confMat = table(pred,x$Category[blk==i])
  acc = (confMat[1,1]+confMat[2,2])/sum(confMat)
  pre = (confMat[1,1])/sum(confMat[1,])
  rec = (confMat[1,1])/sum(confMat[,1])
  all.acc = rbind(all.acc,acc)
  all.pre = rbind(all.pre,pre)
  all.rec = rbind(all.rec,rec)
}
j=2
print(mean(all.acc))
```

```
## [1] 0.9431034
```

```
print(mean(all.pre))
```

```
## [1] 0.9653461
```

```
print(mean(all.rec))
```

```
## [1] 0.9716209
```

```
table_accuracy[j,1] = mean(all.acc)
table_accuracy[j,2] = mean(all.pre)
table_accuracy[j,3] = mean(all.rec)
```

SVM Linear

```
set.seed(2)

all.acc = numeric(0)
all.pre = numeric(0)
all.rec = numeric(0)
for(i in 1:k){
  model = e1071::svm(Category~.,x[blk != i,],kernel="linear",type="C")
  pred = predict(model,x[blk==i,],type="class")
  confMat = table(pred,x$Category[blk==i])
  acc = (confMat[1,1]+confMat[2,2])/sum(confMat)
  pre = (confMat[1,1])/sum(confMat[1,])
  rec = (confMat[1,1])/sum(confMat[,1])
  all.acc = rbind(all.acc,acc)
  all.pre = rbind(all.pre,pre)
  all.rec = rbind(all.rec,rec)
}
j=3
print(mean(all.acc))
```

```
## [1] 0.987931
```

```
print(mean(all.pre))
```

```
## [1] 0.9903394
```

```
print(mean(all.rec))
```

```
## [1] 0.9961874
```

```
table_accuracy[j,1] = mean(all.acc)
table_accuracy[j,2] = mean(all.pre)
table_accuracy[j,3] = mean(all.rec)
```

SVM Polynomial

```
set.seed(2)

all.acc = numeric(0)
all.pre = numeric(0)
all.rec = numeric(0)
for(i in 1:k){
  model = e1071::svm(Category~.,x[blk != i,],kernel="polynomial",type="C")
  pred = predict(model,x[blk==i,],type="class")
  confMat = table(pred,x$Category[blk==i])
  acc = (confMat[1,1]+confMat[2,2])/sum(confMat)
  pre = (confMat[1,1])/sum(confMat[1,])
  rec = (confMat[1,1])/sum(confMat[,1])
  all.acc = rbind(all.acc,acc)
  all.pre = rbind(all.pre,pre)
  all.rec = rbind(all.rec,rec)
}
j=4
print(mean(all.acc))
```

```
## [1] 0.9534483
```

```
print(mean(all.pre))
```

```
## [1] 0.9554892
```

```
print(mean(all.rec))
```

```
## [1] 0.994104
```

```
table_accuracy[j,1] = mean(all.acc)
table_accuracy[j,2] = mean(all.pre)
table_accuracy[j,3] = mean(all.rec)
```

Nerual Net

```
set.seed(2)

all.acc = numeric(0)
all.pre = numeric(0)
all.rec = numeric(0)
for(i in 1:k){
  model = nnet::nnet(Category~.,x[blk != i,], size = 7, trace=FALSE, wgts=.1)
  pred = as.integer(predict(model, x[blk==i,]))
  confMat = table(pred,x$Category[blk==i])
  acc = (confMat[1,1])/sum(confMat)
  pre = (confMat[1,1])/sum(confMat[1,])
  rec = (confMat[1,1])/sum(confMat[,1])
  all.acc = rbind(all.acc,acc)
  all.pre = rbind(all.pre,pre)
  all.rec = rbind(all.rec,rec)
}
j=5
print(mean(all.acc))
```

```
## [1] 0.9034483
```

```
print(mean(all.pre))
```

```
## [1] 0.9034483
```

```
print(mean(all.rec))
```

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

```
table_accuracy[j,1] = mean(all.acc)
table_accuracy[j,2] = mean(all.pre)
table_accuracy[j,3] = mean(all.rec)
```

K-Nearest Neighbors

```
set.seed(2)
n=5

trControl = caret::trainControl(method="cv",number=n)
x1 = x[,]
x1$Category = as.factor(x1$Category)
model = caret::train(Category ~ ., method = "knn", tuneGrid = expand.grid(k = 1:10), trControl = trCont
```

```
## Warning: package 'caret' was built under R version 4.1.3
```

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

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```

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

```
model
```

```
## k-Nearest Neighbors
##
## 589 samples
##  12 predictor
##   2 classes: '0', '1'
```

```
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 471, 471, 472, 471, 471
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9778792  0.8653782
##    2  0.9710850  0.8130933
##    3  0.9745038  0.8315294
##    4  0.9711140  0.8078734
##    5  0.9694191  0.7917161
##    6  0.9711140  0.8039485
##    7  0.9660293  0.7684672
##    8  0.9643343  0.7573608
##    9  0.9660293  0.7636440
##   10  0.9643199  0.7469666
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 1.
```

```
set.seed(2)

aall.acc = numeric(0)
all.pre = numeric(0)
all.rec = numeric(0)
for(i in 1:k){
  tr = x1[blk != i,]
  te = x1[blk == i,]
  pred = class::knn(train = tr, test = te, cl = tr[,1], k=8)
  confMat = table(pred,x1$Category[blk==i])
  acc = (confMat[1,1]+confMat[2,2])/sum(confMat)
  pre = (confMat[1,1])/sum(confMat[1,])
  rec = (confMat[1,1])/sum(confMat[,1])
  all.acc = rbind(all.acc,acc)
  all.pre = rbind(all.pre,pre)
  all.rec = rbind(all.rec,rec)
}
j=6
print(mean(all.acc))
```

```
## [1] 0.9353448
```

```
print(mean(all.pre))
```

```
## [1] 0.968173
```

```
print(mean(all.rec))
```

```
## [1] 0.9961874
```

```
table_accuracy[j,1] = mean(all.acc)
table_accuracy[j,2] = mean(all.pre)
table_accuracy[j,3] = mean(all.rec)
```

Table of Models

```
tab = round(table_accuracy,4)
tab
```

```
##                 Accuracy Precision Recall
## DTree             0.9466    0.9659 0.9750
## NB                0.9431    0.9653 0.9716
## SVM-Linerar       0.9879    0.9903 0.9962
## SVM-Polynomial    0.9534    0.9555 0.9941
## ANN               0.9034    0.9034 1.0000
## KNN               0.9353    0.9682 0.9962
```

Write Out Info for further use

```
write.table(tab, file = 'data/accuracy.txt', sep =' ', row.names = TRUE, col.names = TRUE)
write.table(x, file = 'data/main_df.txt', sep =' ', row.names = TRUE, col.names = TRUE)
```

~~~~~~~~~~

Analysis Liver Disease Stages

```
x = allBad
```

Table

```
table_accuracy = matrix(nrow=6,ncol=1)
colnames(table_accuracy) = c('Accuracy')
rownames(table_accuracy) = c('DTree','NB','SVM-Linerar','SVM-Polynomial','ANN','KNN')
table_accuracy
```

```
##                 Accuracy
## DTree                 NA
## NB                    NA
## SVM-Linerar           NA
## SVM-Polynomial        NA
## ANN                   NA
## KNN                   NA
```
~~~~~~~~~~

K-Fold CV

```
n = nrow(x)
k = 5
tail = n%/%k

set.seed(2)

rnd = runif(n)
rank = rank(rnd)
blk = (rank-1)%/%tail+1
blk = as.factor(blk)

print(summary(blk))
```

```
##  1  2  3  4  5  6
## 12 12 12 12 12  3
```

```
#cannot have there be a categoryof 1 in holdout.
print(x$Category[blk==6])
```

```
## [1] 2 4 4
```

Predictive Models

Decision Tree

```
set.seed(2)
all.acc = numeric(0)
for(i in 1:k){
  tree = rpart::rpart(Category~.,x[blk != i,],method="class")
  pred = predict(tree,x[blk==i,],type="class")
  confMat = table(pred,x$Category[blk==i])
  acc = (confMat[1,1]+confMat[2,2]+confMat[3,3]+confMat[4,4])/sum(confMat)
  all.acc = rbind(all.acc,acc)
}
j=1
print(mean(all.acc))
```

```
## [1] 0.45
```

```
table_accuracy[j,1] = mean(all.acc)
```

Naive Bayes

```r
set.seed(2)
all.acc = numeric(0)
for(i in 1:k){
  model = e1071::naiveBayes(Category~.,x[blk != i,],method="class")
  pred = predict(model,x[blk==i,],type="class")
  confMat = table(pred,x$Category[blk==i])
  acc = (confMat[1,1]+confMat[2,2]+confMat[3,3]+confMat[4,4])/sum(confMat)
  all.acc = rbind(all.acc,acc)
}
j=2
print(mean(all.acc))
```

```
## [1] 0.5666667
```

```r
table_accuracy[j,1] = mean(all.acc)
```

SVM Linear

```r
set.seed(2)
all.acc = numeric(0)
for(i in 1:k){
  model = e1071::svm(Category~.,x[blk != i,],kernel="linear",type="C")
  pred = predict(model,x[blk==i,],type="class")
  confMat = table(pred,x$Category[blk==i])
  acc = (confMat[1,1]+confMat[2,2]+confMat[3,3]+confMat[4,4])/sum(confMat)
  all.acc = rbind(all.acc,acc)
}
j=3
print(mean(all.acc))
```

```
## [1] 0.6833333
```

```r
table_accuracy[j,1] = mean(all.acc)
```

SVM Polynomial

```r
set.seed(2)
all.acc = numeric(0)
for(i in 1:k){
  model = e1071::svm(Category~.,x[blk != i,],kernel="polynomial",type="C")
  pred = predict(model,x[blk==i,],type="class")
  confMat = table(pred,x$Category[blk==i])
  acc = (confMat[1,1]+confMat[2,2]+confMat[3,3]+confMat[4,4])/sum(confMat)
  all.acc = rbind(all.acc,acc)
}
j=4
print(mean(all.acc))
```

```
## [1] 0.6333333
```

```
table_accuracy[j,1] = mean(all.acc)
```

Neural Network

```
set.seed(2)
all.acc = numeric(0)
for(i in 1:k){
  model = nnet::nnet(Category~.,x[blk != i,], size = 7, trace=FALSE, wgts=.1)
  pred = as.integer(predict(model, x[blk==i,]))
  confMat = table(pred,x$Category[blk==i])
  acc = (confMat[1,1])/sum(confMat)
  all.acc = rbind(all.acc,acc)
}
j=5
print(mean(all.acc))
```

```
## [1] 0.1166667
```

```
table_accuracy[j,1] = mean(all.acc)
```

K Nearest Neighbors

```
set.seed(2)
n=5
trControl = caret::trainControl(method="cv",number=n)
x1 = x[,]
x1$Category = as.factor(x1$Category)
model = train(Category ~ ., method = "knn", tuneGrid = expand.grid(k = 1:10), trControl = trControl, da
model
```

```
## k-Nearest Neighbors
##
## 63 samples
## 12 predictors
##  4 classes: '1', '2', '3', '4'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 51, 50, 50, 50, 51
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.5705128  0.3851058
##    2  0.5410256  0.3324962
##    3  0.4897436  0.2712879
##    4  0.4897436  0.2703523
##     5  0.4910256  0.2755556
##     6  0.4935897  0.2794914
##     7  0.4948718  0.2824444
##     8  0.4602564  0.2363541
##     9  0.4602564  0.2267973
##    10  0.4923077  0.2710222
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 1.
```

```
set.seed(2)
aall.acc = numeric(0)
for(i in 1:k){
  tr = x1[blk != i,]
  te = x1[blk == i,]
  pred = class::knn(train = tr, test = te, cl = tr[,1], k=10)
  confMat = table(pred,x1$Category[blk==i])
  acc = (confMat[1,1]+confMat[2,2]+confMat[3,3]+confMat[4,4])/sum(confMat)
  all.acc = rbind(all.acc,acc)
}
j=6
print(mean(all.acc))
```

```
## [1] 0.3166667
```

```
table_accuracy[j,1] = mean(all.acc)
```

Table

```
tab = round(table_accuracy,4)
tab
```

```
##                 Accuracy
## DTree             0.4500
## NB                0.5667
## SVM-Linerar       0.6833
## SVM-Polynomial    0.6333
## ANN               0.1167
## KNN               0.3167
```

Write Out

```
write.table(tab, file = 'data/accuracy_allBad.txt', sep =' ', row.names = TRUE, col.names = TRUE)
write.table(x, file = 'data/allBad_df.txt', sep =' ', row.names = TRUE, col.names = TRUE)
```