Statistical Machine Learning

Fabian Peri

October 10, 2018

```r
# packages needed for chapter 6
library(dplyr)
```

```
##
## 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
```

```r
library(ggplot2)
library(FNN)
library(rpart)
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(xgboost)
```

```
##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
# Import the datasets needed for chapter 6
PSDS_PATH <- file.path('C:/Users/fabia/Desktop', 'psds_data')

## Import datasets needed for chapter 6
loan200 <- read.csv(file.path(PSDS_PATH, 'data', 'loan200.csv'))
loan200$outcome <- ordered(loan200$outcome, levels=c('paid off', 'default'))

loan3000 <- read.csv(file.path(PSDS_PATH, 'data', 'loan3000.csv'))
loan3000$outcome <- ordered(loan3000$outcome, levels=c('paid off', 'default'))
```

```
loan_data <- read.csv(file.path(PSDS_PATH, 'data', 'loan_data.csv'))
loan_data <- select(loan_data, -X, -status)
```

```
## KNN
## the first row of loan200 is the target data
newloan <- loan200[1, 2:3, drop=FALSE]
knn_pred <- knn(train=loan200[-1,2:3], test=newloan, cl=loan200[-1,1], k=20)
knn_pred == 'paid off'
```

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

```
## look at the nearest 20 records
loan200[attr(knn_pred, 'nn.index')-1, ]
```

```
##      outcome payment_inc_ratio   dti
## 34  paid off           3.84084  9.36
## 181 paid off          11.10790 15.33
## 180 paid off           5.00386  5.97
## 84  paid off           8.60830 16.17
## 8   paid off          13.85620 11.24
## 168  default          10.12410 16.67
## 20  paid off           5.92267 18.11
## 198  default           2.97641 16.41
## 76   default           3.86227 22.91
## 54  paid off           2.49545  2.40
## 140  default          10.02450 19.11
## 30   default          16.41910 26.08
## 65   default          10.85580  6.80
## 162 paid off          10.18890 25.47
## 160 paid off           1.65527 12.91
## 111 paid off           4.76040 25.95
## 77  paid off           4.39094  2.86
## 45   default           1.49472 23.79
## 40  paid off           4.27237 12.22
## 138 paid off           3.29013 24.39
```

```
dist <- attr(knn_pred, 'nn.dist')

circleFun <- function(center = c(0,0), r = 1, npoints = 100){
  tt <- seq(0, 2*pi, length.out = npoints-1)
  xx <- center[1] + r * cos(tt)
  yy <- center[2] + r * sin(tt)
  return(data.frame(x = c(xx, xx[1]), y = c(yy, yy[1])))
}

circle_df <- circleFun(center=unlist(newloan), r=max(dist), npoints=201)
loan200_df <- bind_cols(loan200, circle_df)

## Code for figure 6-2: small KNN example
png(filename=file.path(PSDS_PATH, 'figures', 'psds_0602.png'), width = 5.5, height=4, units='in', res=30

ggplot(data=loan200_df, aes(x=payment_inc_ratio, dti, color=outcome, shape=outcome)) +
  geom_point(size=2) +
  scale_shape_manual(values = c(1, 4, 15)) +
  geom_path(aes(x=x, y=y), color='black') +
```

```r
  xlim(3, 15) +
  ylim(17, 29) +
  theme_bw()
```

```
## Warning: Removed 126 rows containing missing values (geom_point).
```

```r
dev.off()
```

```
## pdf
##   2
```

```r
## Standardization
loan_df <- model.matrix(~ -1 + payment_inc_ratio + dti + revol_bal + revol_util, data=loan_data)
newloan = loan_df[1,, drop=FALSE]
loan_df = loan_df[-1,]
outcome <- loan_data[-1,1]
knn_pred <- knn(train=loan_df, test=newloan, cl=outcome, k=5)
knn_pred
```

```
## [1] 4000
## attr(,"nn.index")
##       [,1]  [,2]  [,3]  [,4]  [,5]
## [1,] 35536 33651 25863 42953 43599
## attr(,"nn.dist")
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 1.555631 5.640407 7.138838 8.842243 8.972774
## Levels: 4000
```

```r
loan_df[attr(knn_pred,"nn.index"),]
```

```
##       payment_inc_ratio  dti revol_bal revol_util
## 35537           1.47212 1.46      1686       10.0
## 33652           3.38178 6.37      1688        8.4
## 25864           2.36303 1.39      1691        3.5
## 42954           1.28160 7.14      1684        3.9
## 43600           4.12244 8.98      1684        7.2
```

```r
loan_df <- model.matrix(~ -1 + payment_inc_ratio + dti + revol_bal + revol_util, data=loan_data)
loan_std <- scale(loan_df)
target_std = loan_std[1,, drop=FALSE]
loan_std = loan_std[-1,]
outcome <- loan_data[-1,1]
knn_pred <- knn(train=loan_std, test=target_std, cl=outcome, k=5)
knn_pred
```

```
## [1] 2000
## attr(,"nn.index")
##      [,1] [,2]  [,3]  [,4]  [,5]
## [1,] 2080 1438 30215 28542 44737
## attr(,"nn.dist")
##           [,1]       [,2]       [,3]      [,4]     [,5]
## [1,] 0.0575066 0.09801921 0.09886893 0.1054015 0.116448
## Levels: 2000
```

```r
loan_df[attr(knn_pred,"nn.index"),]
```

```
##      payment_inc_ratio   dti revol_bal revol_util
## 2080          10.04400 19.89      9179       51.5
## 1438            3.87890  5.31      1687       51.1
## 30215           6.71820 15.44      4295       26.0
## 28542           6.93816 20.31     11182       76.1
## 44737           8.20170 16.65      5244       73.9
```

```
## Create a feature for borrowers
borrow_df <- model.matrix(~ -1 + dti + revol_bal + revol_util + open_acc +
                            delinq_2yrs_zero + pub_rec_zero, data=loan_data)
borrow_knn <- knn(borrow_df, test=borrow_df, cl=loan_data[, 'outcome'], prob=TRUE, k=20)
prob <- attr(borrow_knn, "prob")
borrow_feature <- ifelse(borrow_knn=='default', 1-prob, prob)
summary(borrow_feature)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.050   0.400   0.500   0.499   0.600   1.000
```

```
loan_data$borrower_score <- borrow_feature
```

```
# Decision trees

loan_tree <- rpart(outcome ~ borrower_score + payment_inc_ratio,
                   data=loan3000,
                   control = rpart.control(cp=.005))

## Figure 6-3: Rules for simple tree model (not same as in book)
png(filename=file.path(PSDS_PATH, 'figures', 'psds_rpart_tree.png'),  width = 6, height=4, units='in', 

par(mar=c(0,0,0,0)+.1)
plot(loan_tree, uniform=TRUE, margin=.05)
text(loan_tree, cex=.75)

dev.off()
```

```
## pdf
##   2
```

```
## Figure 6-4: View of partition rules
r_tree <- data_frame(x1 = c(0.575, 0.375, 0.375, 0.375, 0.475),
                     x2 = c(0.575, 0.375, 0.575, 0.575, 0.475),
                     y1 = c(0,         0, 10.42, 4.426, 4.426),
                     y2 = c(25,       25, 10.42, 4.426, 10.42),
                     rule_number = factor(c(1, 2, 3, 4, 5)))
r_tree <- as.data.frame(r_tree)

labs <- data.frame(x=c(.575 + (1-.575)/2,
                       .375/2,
                       (.375 + .575)/2,
                       (.375 + .575)/2,
                       (.475 + .575)/2,
                       (.375 + .475)/2
                       ),
                   y=c(12.5,
                       12.5,
                       10.42 + (25-10.42)/2,
                       4.426/2,
                       4.426 + (10.42-4.426)/2,
                       4.426 + (10.42-4.426)/2
                       ),
                   decision = factor(c('paid off', 'default', 'default', 'paid off', 'paid off', 'defaul


png(filename=file.path(PSDS_PATH, 'figures', 'psds_0604.png'), width = 6, height=4, units='in', res=300

ggplot(data=loan3000, aes(x=borrower_score, y=payment_inc_ratio)) +
  geom_point( aes(color=outcome, shape=outcome), alpha=.5) +
  scale_color_manual(values=c('blue', 'red')) +
  scale_shape_manual(values = c(1, 46)) +
  # scale_shape_discrete(solid=FALSE) +
  geom_segment(data=r_tree, aes(x=x1, y=y1, xend=x2, yend=y2, linetype=rule_number), size=1.5, alpha=.7
  guides(colour = guide_legend(override.aes = list(size=1.5)),
         linetype = guide_legend(keywidth=3, override.aes = list(size=1))) +
  scale_x_continuous(expand=c(0,0)) +
  scale_y_continuous(expand=c(0,0), limits=c(0, 25)) +
  geom_label(data=labs, aes(x=x, y=y, label=decision)) +
  #theme(legend.position='bottom') +
  theme_bw()
```

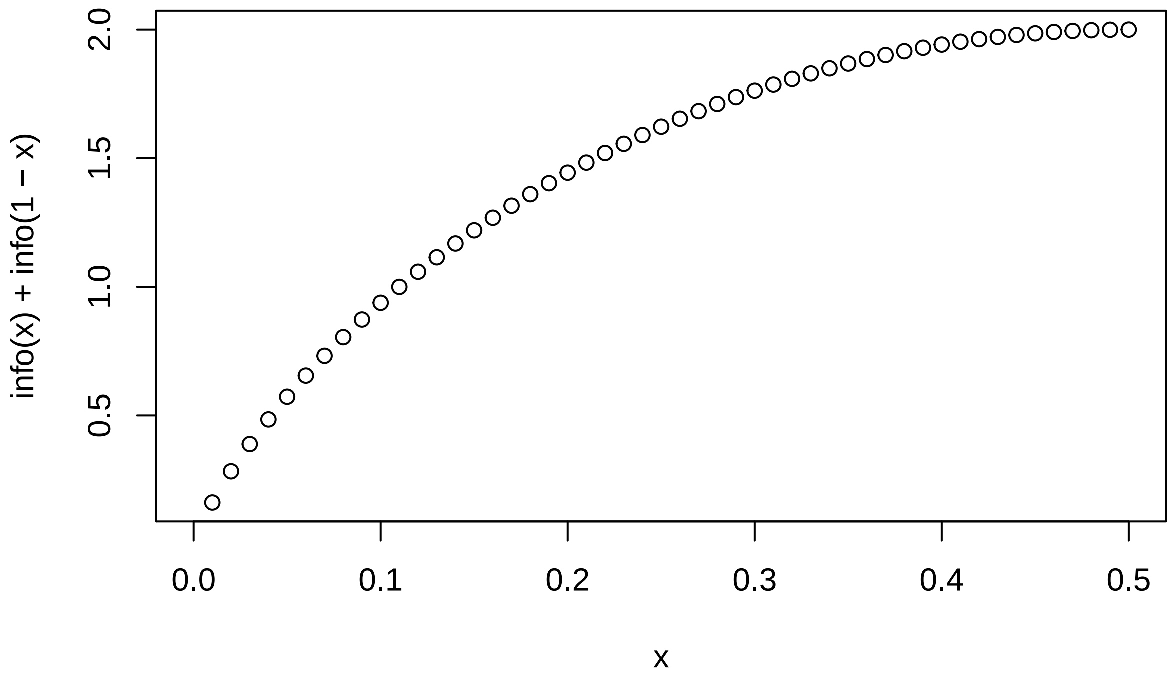
```
## Warning: Removed 2 rows containing missing values (geom_point).
```

```
dev.off()
```

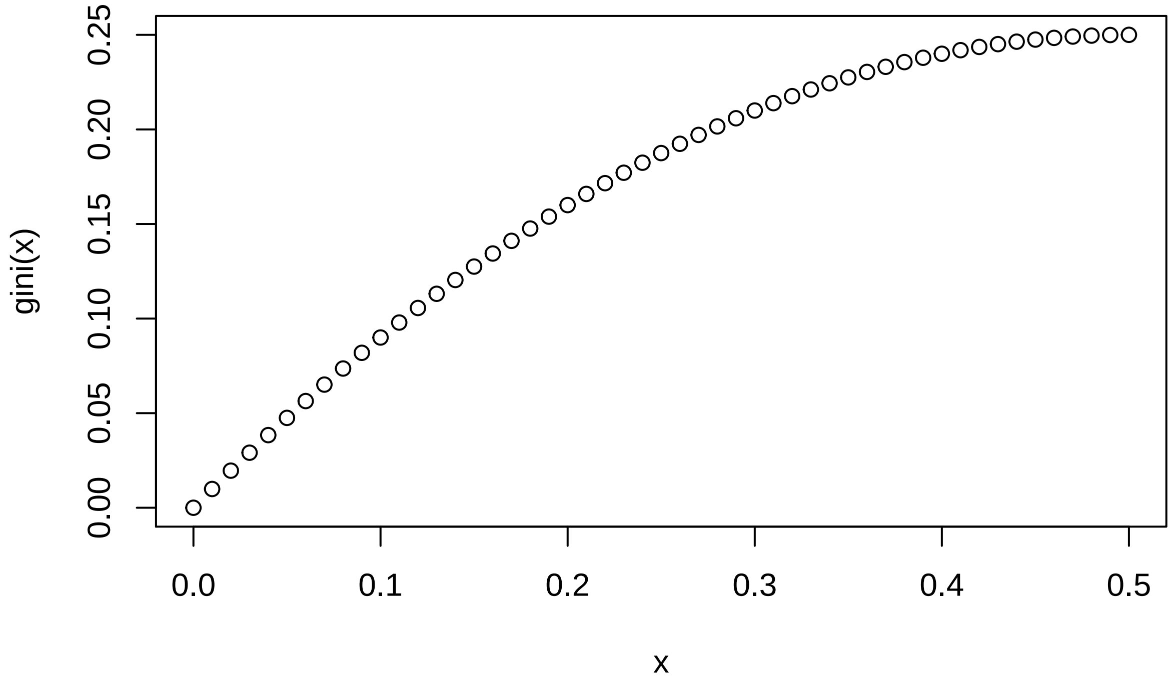
```
## pdf
##   2
```

```
## Gini coefficient and impurity

info <- function(x){
  info <- ifelse(x==0, 0, -x * log2(x) - (1-x) * log2(1-x))
  return(info)
}
x <- 0:50/100
plot(x, info(x) + info(1-x))
```



```
gini <- function(x){
  return(x * (1-x))
}
plot(x, gini(x))
```



```
impure <- data.frame(p = rep(x, 3),
                     impurity = c(2*x,
                                  gini(x)/gini(.5)*info(.5),
                                  info(x)),
                     type = rep(c('Accuracy', 'Gini', 'Entropy'), rep(51,3)))

## Figure 06-05: comparison of impurity measures
png(filename=file.path(PSDS_PATH, 'figures', 'psds_0605.png'), width = 5, height=4, units='in', res=300

ggplot(data=impure, aes(x=p, y=impurity, linetype=type, color=type)) +
  geom_line(size=1.5) +
  guides( linetype = guide_legend( keywidth=3, override.aes = list(size=1))) +
  scale_x_continuous(expand=c(0,0.01)) +
  scale_y_continuous(expand=c(0,0.01)) +
  theme_bw() +
  theme( legend.title=element_blank())

dev.off()
```

```
## pdf
##   2
```

```
# ensemble models: random forest

rf <- randomForest(outcome ~ borrower_score + payment_inc_ratio,
                   data=loan3000)
rf
```

```
##
## Call:
##  randomForest(formula = outcome ~ borrower_score + payment_inc_ratio,      data = loan3000)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 38.63%
## Confusion matrix:
##          paid off default class.error
## paid off      970     585   0.3762058
## default       574     871   0.3972318
```

```
## Figure 6-6: error rate of random forest
png(filename=file.path(PSDS_PATH, 'figures', 'psds_0606.png'), width = 5, height=4, units='in', res=300

error_df = data.frame(error_rate = rf$err.rate[,'OOB'],
                      num_trees = 1:rf$ntree)
ggplot(error_df, aes(x=num_trees, y=error_rate)) +
  geom_line()  +
  theme_bw()

dev.off()
```

```
## pdf
##   2
```

```
## Figure 6-7: plot of random forest predictions
png(filename=file.path(PSDS_PATH, 'figures', 'psds_0607.png'),  width = 5, height=4, units='in', res=30

pred <- predict(rf, prob=TRUE)
rf_df <- cbind(loan3000, pred = pred)

ggplot(data=rf_df, aes(x=borrower_score, y=payment_inc_ratio,
                       shape=pred, color=pred)) +
  geom_point(alpha=.6, size=2) +
  scale_shape_manual( values=c( 46, 4)) +
  scale_x_continuous(expand=c(0,0)) +
  scale_y_continuous(expand=c(0,0), lim=c(0, 20)) +
  theme_bw()
```

```
## Warning: Removed 18 rows containing missing values (geom_point).
```

```
dev.off()
```

```
## pdf
##   2
```

```
# ensemble models: xgboost

predictors <- data.matrix(loan3000[, c('borrower_score', 'payment_inc_ratio')])
label <- as.numeric(loan3000[,'outcome'])-1
xgb <- xgboost(data=predictors, label=label, objective = "binary:logistic",
               params=list(subsample=.63, eta=0.1), nrounds=100)
```

```
## [1]  train-error:0.362000
## [2]  train-error:0.351667
## [3]  train-error:0.346667
## [4]  train-error:0.344000
## [5]  train-error:0.342667
## [6]  train-error:0.338000
## [7]  train-error:0.338667
## [8]  train-error:0.338000
## [9]  train-error:0.331667
## [10] train-error:0.329333
## [11] train-error:0.323333
## [12] train-error:0.325333
## [13] train-error:0.324000
## [14] train-error:0.322667
## [15] train-error:0.325667
## [16] train-error:0.323667
## [17] train-error:0.317667
## [18] train-error:0.317333
## [19] train-error:0.314333
## [20] train-error:0.314000
## [21] train-error:0.310333
## [22] train-error:0.308667
## [23] train-error:0.310333
## [24] train-error:0.309333
## [25] train-error:0.310000
## [26] train-error:0.311000
## [27] train-error:0.312667
## [28] train-error:0.308667
## [29] train-error:0.308667
## [30] train-error:0.303000
## [31] train-error:0.302333
## [32] train-error:0.299000
## [33] train-error:0.298000
## [34] train-error:0.296667
## [35] train-error:0.295000
## [36] train-error:0.295000
## [37] train-error:0.294000
## [38] train-error:0.291333
## [39] train-error:0.292667
## [40] train-error:0.292000
## [41] train-error:0.289667
## [42] train-error:0.292000
## [43] train-error:0.288667
## [44] train-error:0.289000
## [45] train-error:0.288000
## [46] train-error:0.286333
## [47] train-error:0.287000
## [48] train-error:0.284000
## [49] train-error:0.283000
## [50] train-error:0.282667
## [51] train-error:0.279667
## [52] train-error:0.279667
## [53] train-error:0.280333
## [54] train-error:0.278000
## [55] train-error:0.277333
## [56] train-error:0.276000
## [57] train-error:0.276000
## [58] train-error:0.275000
## [59] train-error:0.275000
## [60] train-error:0.274000
## [61] train-error:0.274667
## [62] train-error:0.274667
## [63] train-error:0.274667
## [64] train-error:0.273667
## [65] train-error:0.273667
## [66] train-error:0.274333
## [67] train-error:0.273000
## [68] train-error:0.270333
## [69] train-error:0.269000
## [70] train-error:0.266000
## [71] train-error:0.265667
## [72] train-error:0.263333
## [73] train-error:0.261667
## [74] train-error:0.261000
## [75] train-error:0.259000
## [76] train-error:0.258333
## [77] train-error:0.256333
## [78] train-error:0.254000
## [79] train-error:0.254333
## [80] train-error:0.251667
## [81] train-error:0.251667
## [82] train-error:0.250667
## [83] train-error:0.253333
## [84] train-error:0.252000
## [85] train-error:0.251667
## [86] train-error:0.249000
## [87] train-error:0.248667
## [88] train-error:0.247000
## [89] train-error:0.246333
## [90] train-error:0.247000
## [91] train-error:0.245667
## [92] train-error:0.248000
## [93] train-error:0.247667
## [94] train-error:0.245333
## [95] train-error:0.246667
## [96] train-error:0.245667
## [97] train-error:0.245333
## [98] train-error:0.246667
## [99] train-error:0.245000
## [100]    train-error:0.244000
```

```
pred <- predict(xgb, newdata=predictors)
xgb_df <- cbind(loan3000, pred_default=pred>.5, prob_default=pred)

## Figure 6-9: prediction from xgboost
png(filename=file.path(PSDS_PATH, 'figures', 'psds_0609.png'), width = 5, height=4, units='in', res=300

ggplot(data=xgb_df, aes(x=borrower_score, y=payment_inc_ratio,
                        color=pred_default, shape=pred_default)) +
  geom_point(alpha=.6, size=2) +
  scale_shape_manual( values=c( 46, 4)) +
  scale_x_continuous(expand=c(.03, 0)) +
  scale_y_continuous(expand=c(0,0), lim=c(0, 20)) +
  theme_bw()
```

```
## Warning: Removed 18 rows containing missing values (geom_point).
```

```
dev.off()
```

```
## pdf
##   2
```

```
## Create a test and training set and compare the learning rates under different hyperparameter choices
seed <- 400820
predictors <- data.matrix(loan_data[,-which(names(loan_data) %in% 'outcome')])
label <- as.numeric(loan_data$outcome)-1
test_idx <- sample(nrow(loan_data), 10000)

xgb_default <- xgboost(data=predictors[-test_idx,], label=label[-test_idx],
                       objective = "binary:logistic", nrounds=250, verbose=0)
pred_default <- predict(xgb_default, predictors[test_idx,])
error_default <- abs(label[test_idx] - pred_default) > 0.5
xgb_default$evaluation_log[250,]
```

```
##    iter train_error
## 1:  250    0.130015
```

```
mean(error_default)
```

```
## [1] 0.3521
```

```
xgb_penalty <- xgboost(data=predictors[-test_idx,],
                       label=label[-test_idx],
                       params=list(eta=.1, subsample=.63, lambda=1000),
                       objective = "binary:logistic", nrounds=250, verbose=0)
pred_penalty <- predict(xgb_penalty, predictors[test_idx,])
error_penalty <- abs(label[test_idx] - pred_penalty) > 0.5
xgb_penalty$evaluation_log[250,]
```

```
##    iter train_error
## 1:  250    0.310367
```

```
mean(error_penalty)
```

```
## [1] 0.3317
```

```
error_default <- rep(0, 250)
error_penalty <- rep(0, 250)
for(i in 1:250)
{
  pred_default <- predict(xgb_default, predictors[test_idx,], ntreelimit = i)
  error_default[i] <- mean(abs(label[test_idx] - pred_default) > 0.5)
  pred_penalty <- predict(xgb_penalty, predictors[test_idx,], ntreelimit = i)
  error_penalty[i] <- mean(abs(label[test_idx] - pred_penalty) > 0.5)
}

errors <- rbind(xgb_default$evaluation_log,
                xgb_penalty$evaluation_log,
                data.frame(iter=1:250, train_error=error_default),
                data.frame(iter=1:250, train_error=error_penalty))
errors$type <- rep(c('default train', 'penalty train',
                     'default test', 'penalty test'), rep(250, 4))

## Figure 6-10: learning rates for different choices of hyperparameters
png(filename=file.path(PSDS_PATH, 'figures', 'psds_0610.png'), width = 6, height=4, units='in', res=300)

ggplot(errors, aes(x=iter, y=train_error, group=type)) +
  geom_line(aes(linetype=type, color=type), size=1) +
  scale_linetype_manual(values=c('solid', 'dashed', 'dotted', 'longdash')) +
  theme_bw() +
  theme(legend.key.width = unit(1.5,"cm")) +
  labs(x="Iterations", y="Error") +
  guides(colour = guide_legend(override.aes = list(size=1)))

dev.off()
```

```
## pdf
##   2
```

```
## Cross validation
N <- nrow(loan_data)
fold_number <- sample(1:5, N, replace = TRUE)
params <- data.frame(eta = rep(c(.1, .5, .9), 3),
                     max_depth = rep(c(3, 6, 12), rep(3,3)))
rf_list <- vector('list', 9)
error <- matrix(0, nrow=9, ncol=5)
for(i in 1:nrow(params)){
  for(k in 1:5){
    cat('Fold', k, 'for model', i, '\n')
    fold_idx <- (1:N)[fold_number == k]
    xgb <- xgboost(data=predictors[-fold_idx,], label=label[-fold_idx],
                   params = list(eta = params[i, 'eta'],
                                 max_depth = params[i, 'max_depth']),
                   objective = "binary:logistic", nrounds=100, verbose=0)
    pred <- predict(xgb, predictors[fold_idx,])
    error[i, k] <- mean(abs(label[fold_idx] - pred) >= 0.5)
  }
}
```

```
## Fold 1 for model 1
## Fold 2 for model 1
## Fold 3 for model 1
## Fold 4 for model 1
## Fold 5 for model 1
## Fold 1 for model 2
## Fold 2 for model 2
## Fold 3 for model 2
## Fold 4 for model 2
## Fold 5 for model 2
## Fold 1 for model 3
## Fold 2 for model 3
## Fold 3 for model 3
## Fold 4 for model 3
## Fold 5 for model 3
## Fold 1 for model 4
## Fold 2 for model 4
## Fold 3 for model 4
## Fold 4 for model 4
## Fold 5 for model 4
## Fold 1 for model 5
## Fold 2 for model 5
## Fold 3 for model 5
## Fold 4 for model 5
## Fold 5 for model 5
## Fold 1 for model 6
## Fold 2 for model 6
## Fold 3 for model 6
## Fold 4 for model 6
## Fold 5 for model 6
## Fold 1 for model 7
## Fold 2 for model 7
## Fold 3 for model 7
## Fold 4 for model 7
## Fold 5 for model 7
## Fold 1 for model 8
## Fold 2 for model 8
## Fold 3 for model 8
## Fold 4 for model 8
## Fold 5 for model 8
## Fold 1 for model 9
## Fold 2 for model 9
## Fold 3 for model 9
## Fold 4 for model 9
## Fold 5 for model 9
```

```
avg_error <- 100 * round(rowMeans(error), 4)
cbind(params, avg_error)
```

```
##   eta max_depth avg_error
## 1 0.1         3     32.87
## 2 0.5         3     33.61
## 3 0.9         3     34.34
## 4 0.1         6     33.17
## 5 0.5         6     35.58
## 6 0.9         6     38.00
## 7 0.1        12     34.78
## 8 0.5        12     36.91
## 9 0.9        12     37.99
```