

# projectfinal tree classification

ThuanBui

2025-04-29

Load data

```
# Read the data
df <- read.csv("CharlesBookClub.csv", header = TRUE)

# Drop columns 1 and 2 (Seq# and ID#), and also the last five columns (yes and noflorence)
df <- df[, -c(1:2, (ncol(df)-4):ncol(df))]

# Convert Florence to factor for classification
df$Florence <- as.factor(df$Florence)

# Build and plot the tree
library(rpart)
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 4.4.3

library(caret)

## Warning: package 'caret' was built under R version 4.4.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.4.2

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 4.4.3

# View first few rows
head(df)

##   Gender   M   R F FirstPurch ChildBks YouthBks CookBks DoItYBks RefBks ArtBks
## 1     1 297 14  2        22      0      1      1      0      0      0
## 2     0 128  8  2        10      0      0      0      0      0      0
## 3     1 138 22  7        56      2      1      2      0      1      0
## 4     1 228  2  1         2      0      0      0      0      0      0
## 5     1 257 10  1        10      0      0      0      0      0      0
## 6     1 145  6  2        12      0      0      0      0      0      0
```

```

##   GeogBks ItalCook ItalAtlas ItalArt Florence Related.Purchase
## 1      0      0      0      0      0          0
## 2      0      0      0      0      0          0
## 3      1      1      0      0      0          2
## 4      0      0      0      0      0          0
## 5      0      0      0      0      0          0
## 6      0      0      0      0      0          0

```

Data patrition

```

set.seed(6210)
train.index <- sample(rownames(df), 0.6*nrow(df))
valid.index <- setdiff(rownames(df), train.index)

train.df <- df[train.index, ]
valid.df <- df[valid.index, ]

```

Default classification tree and its confusion matrix (without adjustment)

```

# Train the tree
default.ct <- rpart(Florence ~ ., data = train.df, method = "class")

# Predict on validation set (NOT training set)
default.ct.point.pred.valid <- predict(default.ct, valid.df, type = "class")

# Compare prediction to true validation labels
confusionMatrix(default.ct.point.pred.valid, as.factor(valid.df$Florence), positive = "1")

## Confusion Matrix and Statistics
##
##             Reference
## Prediction    0     1
##           0 1468  132
##           1     0     0
##
##             Accuracy : 0.9175
##                 95% CI : (0.9029, 0.9305)
##     No Information Rate : 0.9175
##     P-Value [Acc > NIR] : 0.5231
##
##             Kappa : 0
##
## McNemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##     Pos Pred Value :     NaN
##     Neg Pred Value : 0.9175
##             Prevalence : 0.0825
##     Detection Rate : 0.0000
## Detection Prevalence : 0.0000
##     Balanced Accuracy : 0.5000
##

```

```

##      'Positive' Class : 1
##  

# Comment: this one fail because of sensitivity is 0

Classification tree with cutoff and its confusion matrix

# Train the tree
default.ct <- rpart(Florence ~ ., data = train.df, method = "class")  

# Get predicted probabilities (for class "1")
default.ct.probs.valid <- predict(default.ct, valid.df, type = "prob")[,2]  

# Apply custom cutoff = 0.2
default.ct.pred.valid <- ifelse(default.ct.probs.valid > 0.2, "1", "0")  

# Convert to factor to match true labels
default.ct.pred.valid <- factor(default.ct.pred.valid, levels = c("0", "1"))  

# Compare prediction to true validation labels
confusionMatrix(default.ct.pred.valid, as.factor(valid.df$Florence), positive = "1")  

## Confusion Matrix and Statistics
##  

##      Reference
## Prediction 0 1
##          0 1468 132
##          1 0 0
##  

##      Accuracy : 0.9175
##      95% CI : (0.9029, 0.9305)
##      No Information Rate : 0.9175
##      P-Value [Acc > NIR] : 0.5231
##  

##      Kappa : 0
##  

##  Mcnemar's Test P-Value : <2e-16
##  

##      Sensitivity : 0.0000
##      Specificity : 1.0000
##      Pos Pred Value :     NaN
##      Neg Pred Value : 0.9175
##      Prevalence : 0.0825
##      Detection Rate : 0.0000
##      Detection Prevalence : 0.0000
##      Balanced Accuracy : 0.5000
##  

##      'Positive' Class : 1
##  

# Comment: this one fail because of sensitivity is 0

```

best prune tree with cutoff and its confusion matrix

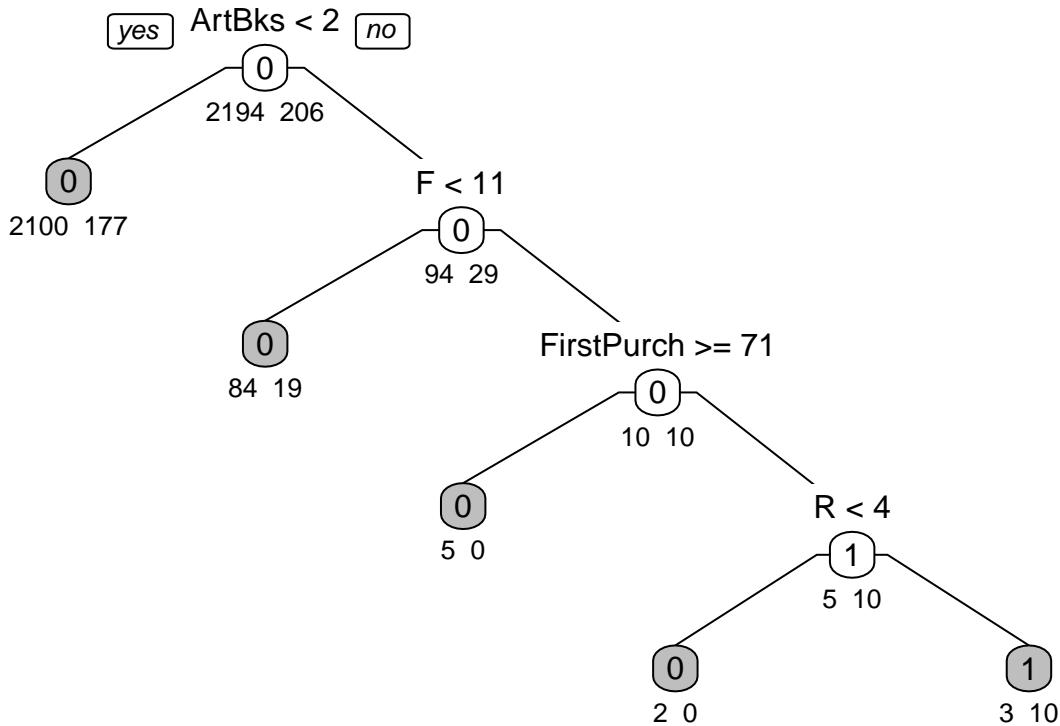
```

set.seed(1)
cv.ct <- rpart(Florence ~ ., data = train.df, method = "class", cp = 0.00001, minsplit = 1, xval = 5)
printcp(cv.ct)

##
## Classification tree:
## rpart(formula = Florence ~ ., data = train.df, method = "class",
##       cp = 1e-05, minsplit = 1, xval = 5)
##
## Variables actually used in tree construction:
## [1] ArtBks          ChildBks        CookBks         DoItYBks
## [5] F               FirstPurch      Gender          GeogBks
## [9] ItalArt         ItalAtlas       ItalCook        M
## [13] R              RefBks         Related.Purchase YouthBks
##
## Root node error: 206/2400 = 0.085833
##
## n= 2400
##
##           CP nsplit rel error xerror     xstd
## 1  0.0080906      0 1.000000 1.0000 0.066616
## 2  0.0072816      4 0.966019 1.0728 0.068763
## 3  0.0048544      7 0.941748 1.1214 0.070140
## 4  0.0036408     21 0.873786 1.2427 0.073411
## 5  0.0035305     47 0.752427 1.2961 0.074779
## 6  0.0033981     58 0.713592 1.3204 0.075387
## 7  0.0032362     71 0.665049 1.3301 0.075628
## 8  0.0029126     77 0.645631 1.3689 0.076580
## 9  0.0027739     92 0.592233 1.5631 0.081055
## 10 0.0024272     99 0.572816 1.6214 0.082312
## 11 0.0022065    159 0.407767 1.7039 0.084033
## 12 0.0019417    178 0.364078 1.7039 0.084033
## 13 0.0018204    238 0.228155 1.8058 0.086066
## 14 0.0016181    254 0.199029 1.8155 0.086255
## 15 0.0013870    329 0.053398 1.8398 0.086722
## 16 0.0012136    336 0.043689 1.8398 0.086722
## 17 0.0000100    355 0.014563 1.8398 0.086722

pruned.ct <- prune(cv.ct, cp = 0.0080906)
prp(pruned.ct, type = 1, extra = 1, under = TRUE, split.font = 1, varlen = -10,
  box.col=ifelse(pruned.ct$frame$var == "<leaf>", 'gray', 'white'))

```



```

library(caret)
# Predict probabilities on validation set
default.ct.probs.valid <- predict(pruned.ct, valid.df, type = "prob")[,2] # Probability for class "1"

# Apply custom threshold
default.ct.point.pred.valid <- ifelse(default.ct.probs.valid > 0.2, "1", "0")
default.ct.point.pred.valid <- factor(default.ct.point.pred.valid, levels = c("0", "1"))

# Evaluate
conf_matrix <- confusionMatrix(default.ct.point.pred.valid, as.factor(valid.df$Florence), positive = "1")
print(conf_matrix)

## Confusion Matrix and Statistics
##
##          Reference
## Prediction      0      1
##           0 1459   128
##           1     9     4
##
##          Accuracy : 0.9144
## 95% CI : (0.8996, 0.9276)
## No Information Rate : 0.9175
## P-Value [Acc > NIR] : 0.6947
##
##          Kappa : 0.041

```

```

## 
##   Mcnemar's Test P-Value : <2e-16
##
##           Sensitivity : 0.030303
##           Specificity : 0.993869
##           Pos Pred Value : 0.307692
##           Neg Pred Value : 0.919345
##           Prevalence : 0.082500
##           Detection Rate : 0.002500
##           Detection Prevalence : 0.008125
##           Balanced Accuracy : 0.512086
##
##           'Positive' Class : 1
## 

# Predict probabilities on training set
default.ct.probs.train <- predict(pruned.ct, train.df, type = "prob")[,2]

# Apply custom threshold
default.ct.point.pred.train <- ifelse(default.ct.probs.train > 0.2, "1", "0")
default.ct.point.pred.train <- factor(default.ct.point.pred.train, levels = c("0", "1"))

# Evaluate
conf_matrix <- confusionMatrix(default.ct.point.pred.train, as.factor(train.df$Florence), positive = "1")
print(conf_matrix)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0     1
##           0 2191  196
##           1     3    10
##
##           Accuracy : 0.9171
##           95% CI : (0.9053, 0.9278)
##           No Information Rate : 0.9142
##           P-Value [Acc > NIR] : 0.3207
##
##           Kappa : 0.082
##
##   Mcnemar's Test P-Value : <2e-16
##
##           Sensitivity : 0.048544
##           Specificity : 0.998633
##           Pos Pred Value : 0.769231
##           Neg Pred Value : 0.917889
##           Prevalence : 0.085833
##           Detection Rate : 0.004167
##           Detection Prevalence : 0.005417
##           Balanced Accuracy : 0.523588
##
##           'Positive' Class : 1
## 
```

Lift chart

```
library(gains)

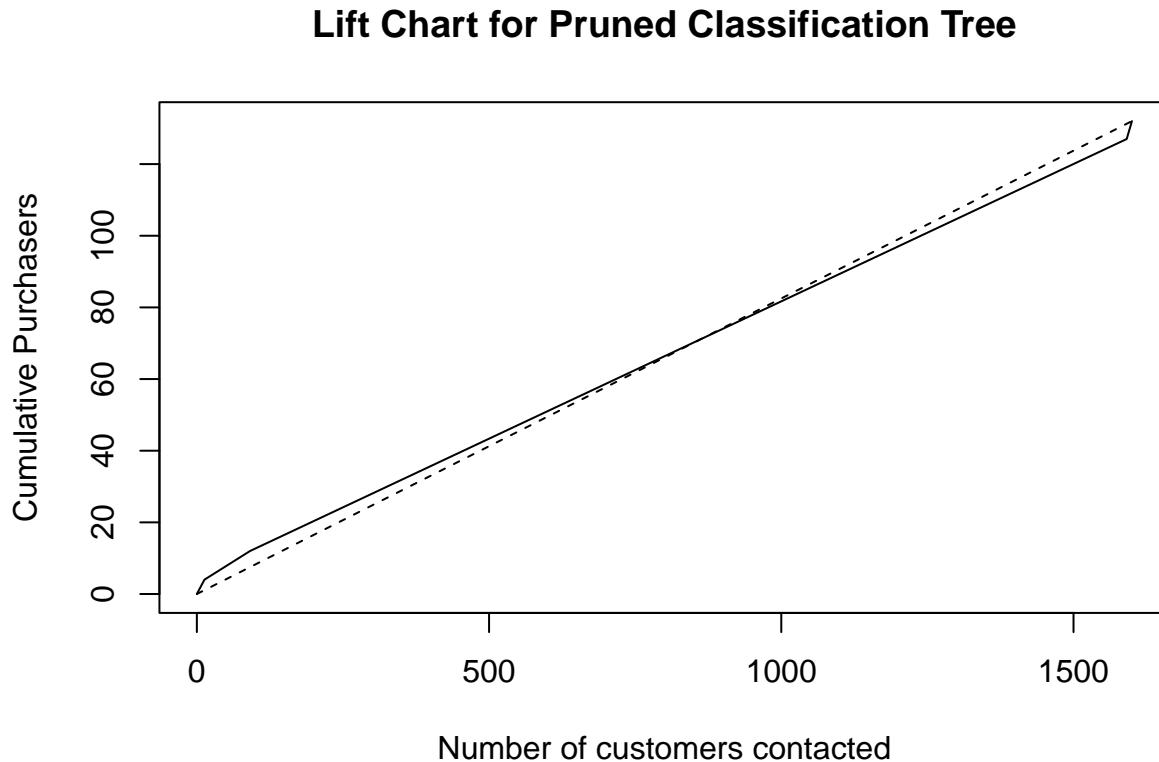
# Convert Florence to numeric (0 = no, 1 = yes)
florence.numeric <- as.numeric(as.character(valid.df$Florence))

# Use probabilities from pruned classification tree
rf.gains <- gains(florence.numeric, default.ct.probs.valid)

## Warning in gains(florence.numeric, default.ct.probs.valid): Warning: Fewer
## distinct predicted values than groups requested

# Plot lift chart
plot(c(0, rf.gains$cume.pct.of.total*sum(florence.numeric)) ~ c(0, rf.gains$cume.obs),
     xlab = "Number of customers contacted",
     ylab = "Cumulative Purchasers",
     main = "Lift Chart for Pruned Classification Tree",
     type = "l")

# Add baseline (random model)
lines(c(0, sum(florence.numeric)) ~ c(0, nrow(valid.df)), lty = 2)
```



Profit chart

```

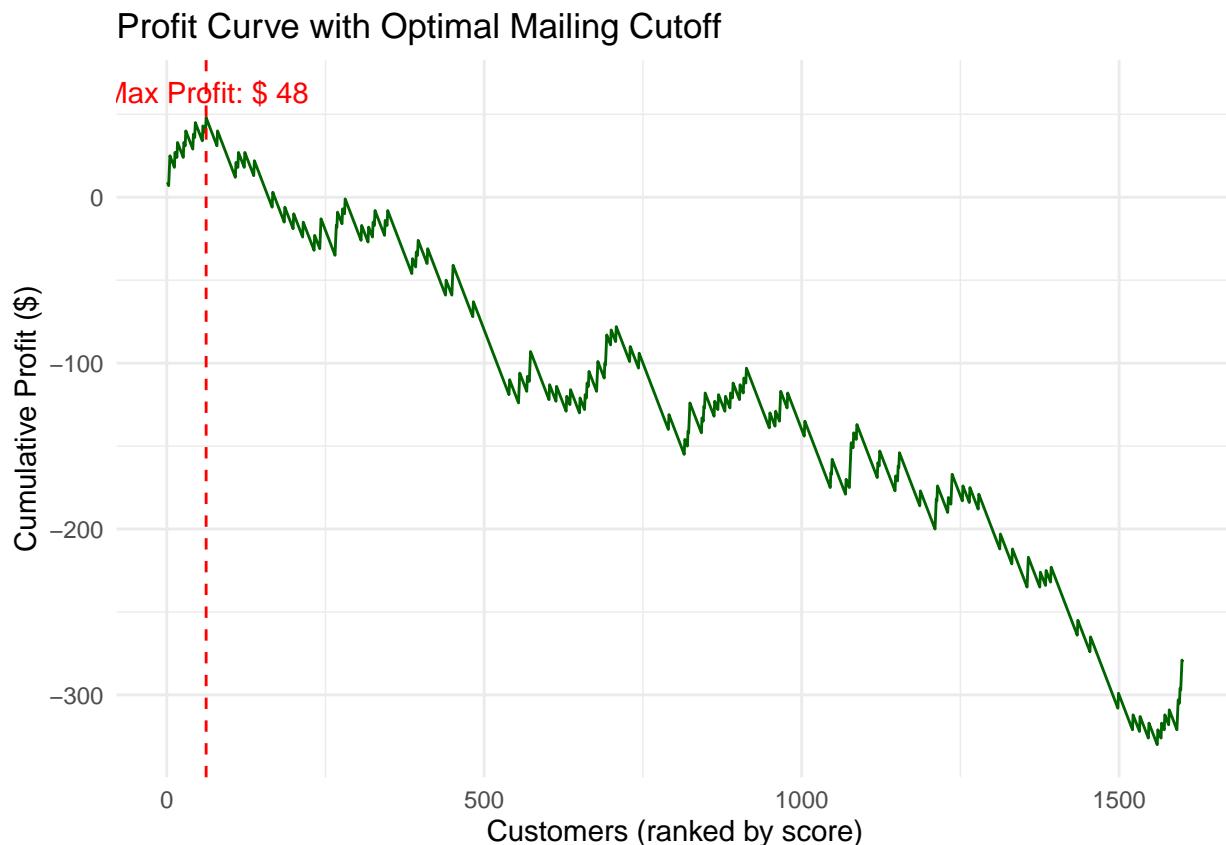
df <- data.frame(prob = default.ct.probs.valid,
                  actual = as.numeric(as.character(valid.df$Florence)))
df <- df[order(-df$prob), ]

df$profit <- ifelse(df$actual == 1, 10 - 1, -1) # +9 if buyer, -1 if not
df$cum_profit <- cumsum(df$profit)

max_profit <- max(df$cum_profit)
max_index <- which.max(df$cum_profit)

library(ggplot2)
ggplot(df, aes(x = 1:nrow(df), y = cum_profit)) +
  geom_line(color = "darkgreen") +
  geom_vline(xintercept = max_index, linetype = "dashed", color = "red") +
  annotate("text", x = max_index, y = max_profit + 15,
           label = paste("Max Profit: $", max_profit), color = "red") +
  labs(title = "Profit Curve with Optimal Mailing Cutoff",
       x = "Customers (ranked by score)",
       y = "Cumulative Profit ($)") +
  theme_minimal()

```



```

max_index <- which.max(df$cum_profit)
best_cutoff <- df$prob[max_index]

```

```

print(best_cutoff)

## [1] 0.184466

Pune + weight

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

# Set custom class weights
classwt <- c("0" = 1, "1" = 5.9)

# Train a classification tree with custom weights
ct <- rpart(Florence ~ ., data = train.df,
             method = "class",
             control = rpart.control(cp = 0.001), # grow full tree
             weights = ifelse(train.df$Florence == 1, classwt["1"], classwt["0"]))

# Find best cp from cross-validation
cp_table <- printcp(ct)

## 
## Classification tree:
## rpart(formula = Florence ~ ., data = train.df, weights = ifelse(train.df$Florence ==
##       1, classwt["1"], classwt["0"])), method = "class", control = rpart.control(cp = 0.001))
##
## Variables actually used in tree construction:
## [1] ArtBks          ChildBks        CookBks         F
## [5] FirstPurch     Gender          GeogBks        ItalCook
## [9] M               R              RefBks         Related.Purchase
##
## Root node error: 1215.4/2400 = 0.50642
##
## n= 2400
##
##           CP nsplit rel error  xerror      xstd
## 1  0.0362021      0  1.00000 1.00000 0.023010
## 2  0.0234491      2  0.92760 1.01325 0.023077
## 3  0.0132467      3  0.90415 0.99251 0.022971
## 4  0.0106961      4  0.89090 1.02221 0.023121
## 5  0.0085980      5  0.88020 1.01078 0.023065
## 6  0.0082277      7  0.86301 1.01917 0.023106
## 7  0.0075284     10  0.83668 1.02082 0.023114
## 8  0.0066233     12  0.82162 1.00708 0.023046
## 9  0.0065822     15  0.79752 1.00708 0.023046
## 10 0.0058417     16  0.79093 1.01275 0.023074
## 11 0.0055949     17  0.78509 1.01530 0.023087
## 12 0.0052932     19  0.77390 1.03250 0.023170
## 13 0.0051423     25  0.72914 1.03497 0.023181
## 14 0.0050189     28  0.71302 1.03340 0.023174
## 15 0.0044018     29  0.70800 1.02789 0.023148

```

```

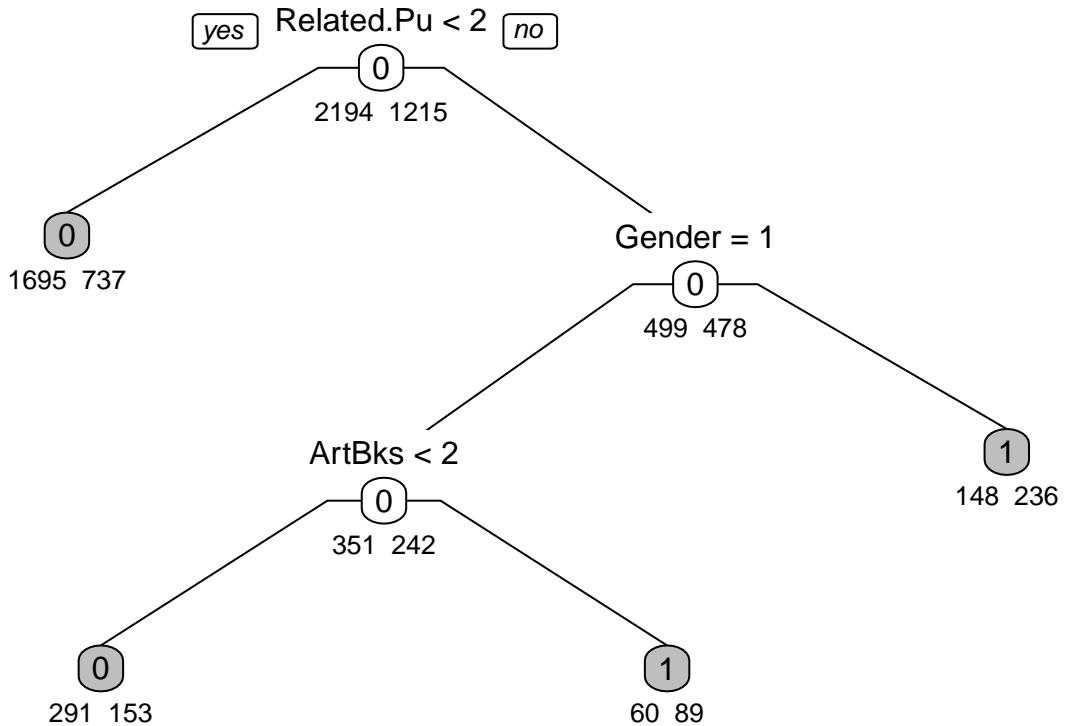
## 16 0.0043881      40  0.65271 1.03201 0.023167
## 17 0.0041139      43  0.63954 1.04443 0.023225
## 18 0.0041002      46  0.62473 1.04854 0.023243
## 19 0.0036751      53  0.59470 1.06393 0.023310
## 20 0.0035379      56  0.58368 1.08565 0.023400
## 21 0.0033734      58  0.57660 1.08976 0.023416
## 22 0.0033734      59  0.57323 1.09149 0.023423
## 23 0.0033734      60  0.56985 1.09149 0.023423
## 24 0.0031265      62  0.56311 1.09470 0.023435
## 25 0.0029826      64  0.55685 1.12588 0.023549
## 26 0.0029071      68  0.54492 1.13592 0.023583
## 27 0.0025506      71  0.53620 1.13518 0.023580
## 28 0.0025506      72  0.53365 1.13074 0.023565
## 29 0.0025506      73  0.53110 1.13074 0.023565
## 30 0.0023038      74  0.52855 1.13238 0.023571
## 31 0.0022215      75  0.52625 1.14308 0.023606
## 32 0.0021118      76  0.52403 1.14316 0.023606
## 33 0.0019747      82  0.50806 1.14078 0.023599
## 34 0.0019253      84  0.50411 1.14152 0.023601
## 35 0.0013987      90  0.48889 1.13675 0.023586
## 36 0.0010000      91  0.48749 1.15024 0.023629

best_cp <- cp_table[which.min(cp_table[, "xerror"]), "CP"]

# Prune the tree
pruned_ct <- prune(ct, cp = best_cp)

# Plot the pruned tree
prp(pruned_ct, type = 1, extra = 1, under = TRUE, split.font = 1, varlen = -10,
    box.col = ifelse(pruned_ct$frame$var == "<leaf>", 'gray', 'white'))

```



```

# Predict on validation set (probability of class "1")
probs <- predict(pruned_ct, valid.df, type = "prob")[,2]

# Apply cutoff of 0.2
preds <- ifelse(probs > 0.2, "1", "0")
preds <- factor(preds, levels = c("0", "1"))

# Evaluate with confusion matrix
conf_matrix <- confusionMatrix(preds, as.factor(valid.df$Florence), positive = "1")
print(conf_matrix)

```

```

## Confusion Matrix and Statistics
##
##             Reference
## Prediction      0      1
##           0     0     0
##           1 1468 132
##
##             Accuracy : 0.0825
##                 95% CI : (0.0695, 0.0971)
##   No Information Rate : 0.9175
##   P-Value [Acc > NIR] : 1
##
##             Kappa : 0
## 
```

```

##  McNemar's Test P-Value : <2e-16
##
##          Sensitivity : 1.0000
##          Specificity : 0.0000
##          Pos Pred Value : 0.0825
##          Neg Pred Value :     NaN
##          Prevalence : 0.0825
##          Detection Rate : 0.0825
##          Detection Prevalence : 1.0000
##          Balanced Accuracy : 0.5000
##
##          'Positive' Class : 1
##

```

## CASE WITH SELECTED PREDICTORS

```

cv.ct <- rpart(Florence ~ .,
                 data = train.df[, c(2:5, which(names(train.df) == "Florence"))],
                 method = "class",
                 cp = 0.00001,
                 minsplit = 1,
                 xval = 5)
printcp(cv.ct)

##
## Classification tree:
## rpart(formula = Florence ~ ., data = train.df[, c(2:5, which(names(train.df) ==
##           "Florence"))], method = "class", cp = 1e-05, minsplit = 1,
##           xval = 5)
##
## Variables actually used in tree construction:
## [1] F           FirstPurch M           R
##
## Root node error: 206/2400 = 0.085833
##
## n= 2400
##
##          CP nsplits rel_error xerror      xstd
## 1  0.00295483      0  1.000000 1.0000 0.066616
## 2  0.00291262     26  0.912621 1.4515 0.078537
## 3  0.00242718     39  0.873786 1.5583 0.080948
## 4  0.00226537     97  0.679612 1.6990 0.083934
## 5  0.00208044    161  0.500000 1.7039 0.084033
## 6  0.00194175    181  0.456311 1.8301 0.086536
## 7  0.00161812    219  0.373786 1.8301 0.086536
## 8  0.00121359    363  0.092233 1.9223 0.088272
## 9  0.00097087    367  0.087379 1.9223 0.088272
## 10 0.00001000   386  0.058252 1.9223 0.088272

default.ct.point.pred.valid <- predict(cv.ct, valid.df, type = "class")
conf_matrix <- confusionMatrix(default.ct.point.pred.valid, as.factor(valid.df$Florence), positive = "1")
conf_matrix

```

```

## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0     1
##           0 1326  116
##           1 142   16
##
##                   Accuracy : 0.8388
##                   95% CI : (0.8198, 0.8564)
##       No Information Rate : 0.9175
##       P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : 0.0225
##
## Mcnemar's Test P-Value : 0.1196
##
##             Sensitivity : 0.12121
##             Specificity  : 0.90327
## Pos Pred Value : 0.10127
## Neg Pred Value : 0.91956
##      Prevalence  : 0.08250
## Detection Rate : 0.01000
## Detection Prevalence : 0.09875
## Balanced Accuracy : 0.51224
##
## 'Positive' Class : 1
##

```

```
# Comment: this one fail because of overfitting
```

Check overfitting

```

default.ct.point.pred.train <- predict(cv.ct, train.df, type = "class")
conf_matrix <- confusionMatrix(default.ct.point.pred.train, as.factor(train.df$Florence), positive = "1")
conf_matrix

```

```

## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0     1
##           0 2193  11
##           1    1  195
##
##                   Accuracy : 0.995
##                   95% CI : (0.9913, 0.9974)
##       No Information Rate : 0.9142
##       P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9674
##
## Mcnemar's Test P-Value : 0.009375
##
##             Sensitivity : 0.94660

```

```

##          Specificity : 0.99954
##      Pos Pred Value : 0.99490
##      Neg Pred Value : 0.99501
##          Prevalence : 0.08583
##      Detection Rate : 0.08125
## Detection Prevalence : 0.08167
##      Balanced Accuracy : 0.97307
##
##      'Positive' Class : 1
##
```

Random forest

Randomn forest with 4 predictor cutoff = 0.2, weight 2:8

```

library(randomForest)

## Warning: package 'randomForest' was built under R version 4.4.3

## randomForest 4.7-1.2

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

##
## Attaching package: 'randomForest'

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

rf <- randomForest(Florence ~ ., data = train.df,
                     ntree = 500,
                     mtry = 4,
                     classwt = c('0' = 0.2, '1' = 0.8),
                     nodesize = 10) # More weight to buyers
rf.probs <- predict(rf, valid.df, type = "prob")[,2]
rf.pred <- ifelse(rf.probs > 0.2, "1", "0")
rf.pred <- factor(rf.pred, levels = c("0", "1"))
confusionMatrix(rf.pred, valid.df$Florence, positive = "1")

## Confusion Matrix and Statistics
##
##          Reference
## Prediction 0 1
##           0 533 37
##           1 935 95
##
##          Accuracy : 0.3925
##             95% CI : (0.3685, 0.4169)
## No Information Rate : 0.9175
## P-Value [Acc > NIR] : 1
##
```

```

##          Kappa : 0.0202
##
##  Mcnemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.71970
##          Specificity : 0.36308
##          Pos Pred Value : 0.09223
##          Neg Pred Value : 0.93509
##          Prevalence : 0.08250
##          Detection Rate : 0.05937
##          Detection Prevalence : 0.64375
##          Balanced Accuracy : 0.54139
##
##          'Positive' Class : 1
##
```

*# Comment: very low accuracy that lower and 50% and overfit*

check overfit

```

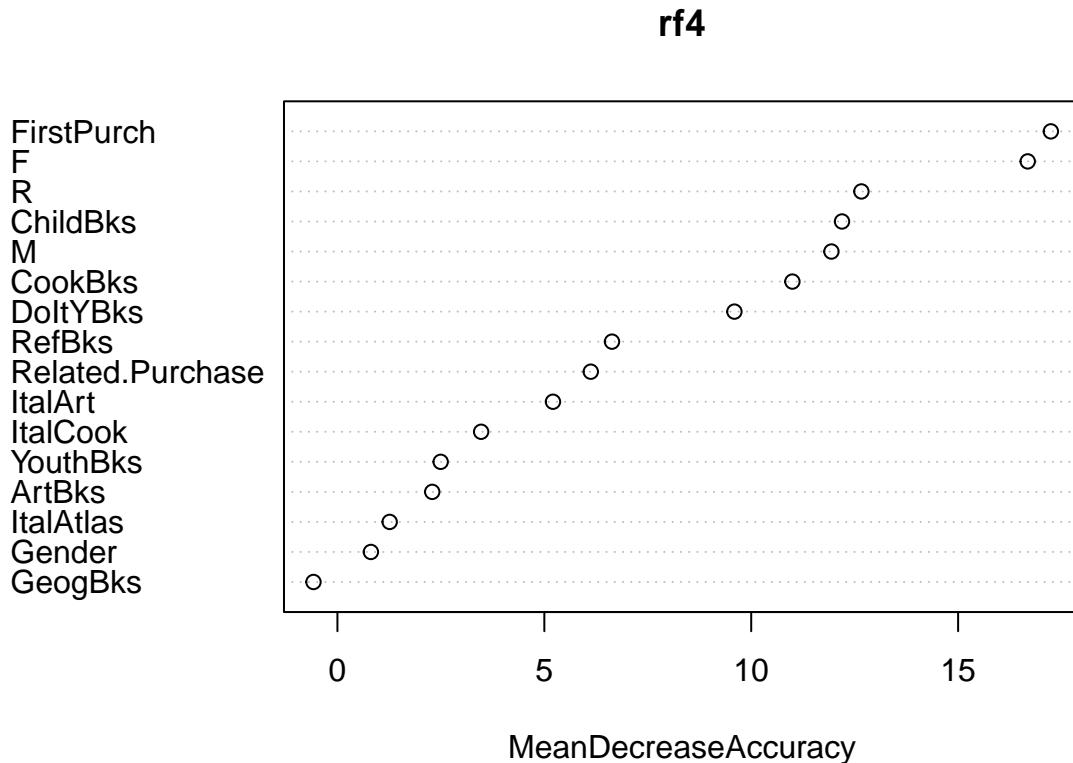
library(randomForest)
rf.probs <- predict(rf, train.df, type = "prob")[,2]
rf.pred <- ifelse(rf.probs > 0.2, "1", "0")
rf.pred <- factor(rf.pred, levels = c("0", "1"))
confusionMatrix(rf.pred, train.df$Florence, positive = "1")
```

```

## Confusion Matrix and Statistics
##
##          Reference
## Prediction   0     1
##           0 1005    0
##           1 1189   206
##
##          Accuracy : 0.5046
##          95% CI : (0.4844, 0.5248)
##          No Information Rate : 0.9142
##          P-Value [Acc > NIR] : 1
##
##          Kappa : 0.1267
##
##  Mcnemar's Test P-Value : <2e-16
##
##          Sensitivity : 1.00000
##          Specificity : 0.45807
##          Pos Pred Value : 0.14767
##          Neg Pred Value : 1.00000
##          Prevalence : 0.08583
##          Detection Rate : 0.08583
##          Detection Prevalence : 0.58125
##          Balanced Accuracy : 0.72903
##
##          'Positive' Class : 1
##
```

Random forest with 4 but no cutoff

```
library(randomForest)
rf4 <- randomForest(as.factor(Florence) ~ ., data = train.df, ntree = 500,
                     mtry = 4, nodesize = 10, importance = TRUE)
varImpPlot(rf4, type = 1)
```



```
rf.probs <- predict(rf4, valid.df, type = "prob")[,2]
rf.pred <- ifelse(rf.probs > 0.5, "1", "0")
rf.pred <- factor(rf.pred, levels = c("0", "1"))
confusionMatrix(rf.pred, valid.df$Florence, positive = "1")

## Confusion Matrix and Statistics
##
##             Reference
## Prediction      0      1
##           0 1467   132
##           1     1     0
##
##                   Accuracy : 0.9169
##                   95% CI : (0.9023, 0.9299)
##       No Information Rate : 0.9175
##       P-Value [Acc > NIR] : 0.5591
##
##                   Kappa : -0.0012
```

```

## 
##   McNemar's Test P-Value : <2e-16
## 
##           Sensitivity : 0.000000
##           Specificity : 0.999319
##           Pos Pred Value : 0.000000
##           Neg Pred Value : 0.917448
##           Prevalence : 0.082500
##           Detection Rate : 0.000000
##           Detection Prevalence : 0.000625
##           Balanced Accuracy : 0.499659
## 
##           'Positive' Class : 1
## 

# Comment: this one fail because of sensitivity is 0

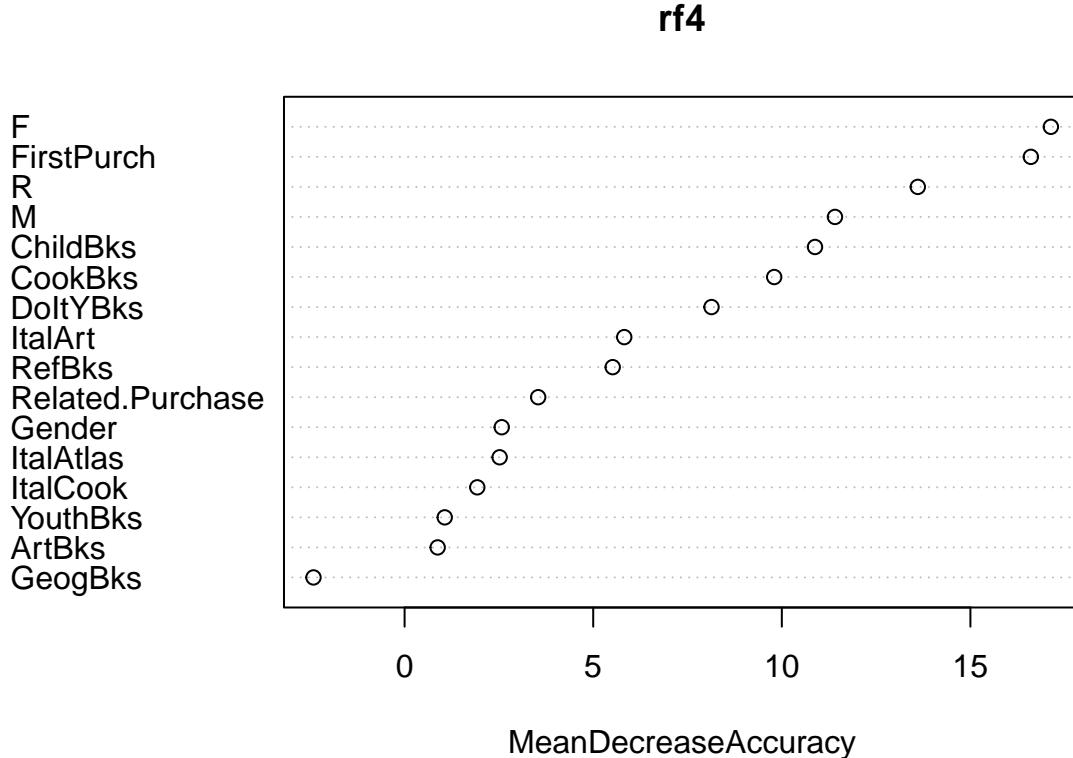
```

Random forest with 4 predictors cutoff

```

library(randomForest)
rf4 <- randomForest(as.factor(Florence) ~ ., data = train.df, ntree = 500,
                     mtry = 4, nodesize = 10, importance = TRUE)
varImpPlot(rf4, type = 1)

```



```

rf.probs <- predict(rf4, valid.df, type = "prob")[,2]
rf.pred <- ifelse(rf.probs > 0.2, "1", "0")
rf.pred <- factor(rf.pred, levels = c("0", "1"))
confusionMatrix(rf.pred, valid.df$Florence, positive = "1")

## Confusion Matrix and Statistics
##
##             Reference
## Prediction      0      1
##           0 1424   116
##           1    44    16
##
##                   Accuracy : 0.9
##                   95% CI : (0.8842, 0.9143)
##       No Information Rate : 0.9175
##       P-Value [Acc > NIR] : 0.9942
##
##                   Kappa : 0.1214
##
## Mcnemar's Test P-Value : 1.988e-08
##
##                   Sensitivity : 0.1212
##                   Specificity : 0.9700
##       Pos Pred Value : 0.2667
##       Neg Pred Value : 0.9247
##                   Prevalence : 0.0825
##       Detection Rate : 0.0100
## Detection Prevalence : 0.0375
##       Balanced Accuracy : 0.5456
##
##       'Positive' Class : 1
##
# Comment: this one fail because of overfitting

```

Check overfit

```

library(randomForest)

rf.probs <- predict(rf4, train.df, type = "prob")[,2]
rf.pred <- ifelse(rf.probs > 0.2, "1", "0")
rf.pred <- factor(rf.pred, levels = c("0", "1"))
confusionMatrix(rf.pred, train.df$Florence, positive = "1")

## Confusion Matrix and Statistics
##
##             Reference
## Prediction      0      1
##           0 2180   94
##           1    14   112
##
##                   Accuracy : 0.955

```

```

##                               95% CI : (0.9459, 0.9629)
##      No Information Rate : 0.9142
##      P-Value [Acc > NIR] : 4.815e-15
##
##                           Kappa : 0.652
##
## McNemar's Test P-Value : 2.921e-14
##
##                           Sensitivity : 0.54369
##                           Specificity : 0.99362
##      Pos Pred Value : 0.88889
##      Neg Pred Value : 0.95866
##                           Prevalence : 0.08583
##                           Detection Rate : 0.04667
##      Detection Prevalence : 0.05250
##                           Balanced Accuracy : 0.76865
##
##      'Positive' Class : 1
##

```

Random forest with 5 predictors

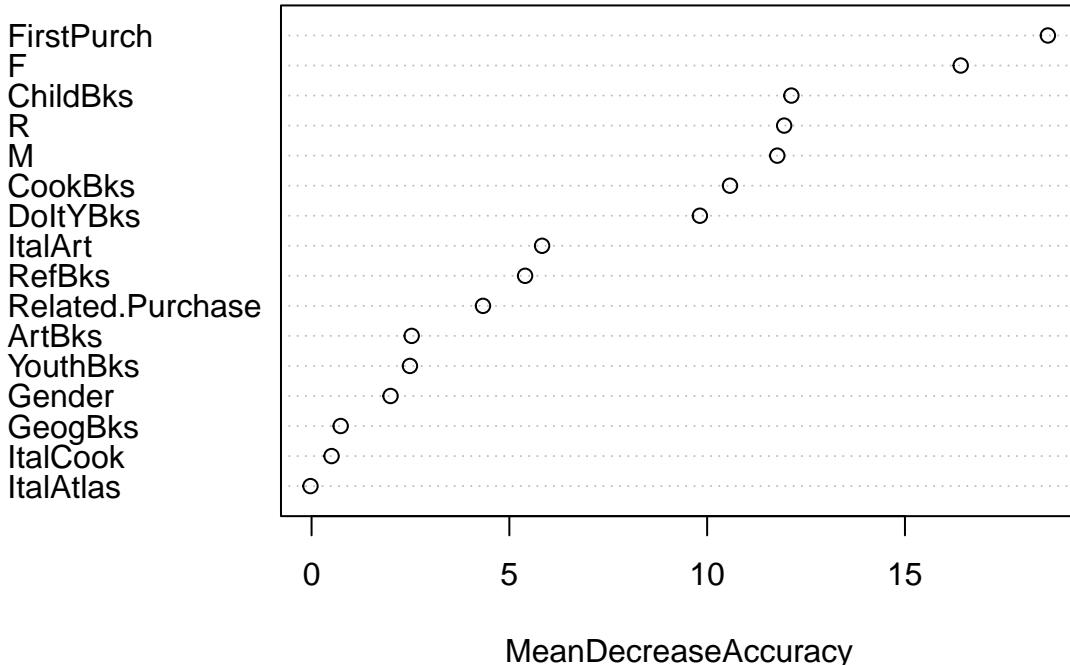
```

library(randomForest)
rf5 <- randomForest(as.factor(Florence) ~ ., data = train.df, ntree = 500,
                     mtry = 5, nodesize = 10, importance = TRUE)

varImpPlot(rf5, type = 1)

```

**rf5**



```

rf.probs <- predict(rf5, valid.df, type = "prob")[,2]
rf.pred <- ifelse(rf.probs > 0.2, "1", "0")
rf.pred <- factor(rf.pred, levels = c("0", "1"))
confusionMatrix(rf.pred, valid.df$Florence, positive = "1")

```

```

## Confusion Matrix and Statistics
##
##             Reference
## Prediction      0      1
##           0 1421   115
##           1    47    17
##
##                  Accuracy : 0.8988
##                  95% CI : (0.8829, 0.9131)
##      No Information Rate : 0.9175
##      P-Value [Acc > NIR] : 0.9965
##
##                  Kappa : 0.1264
##
##  Mcnemar's Test P-Value : 1.409e-07
##
##                  Sensitivity : 0.12879
##                  Specificity  : 0.96798
##      Pos Pred Value : 0.26562
##      Neg Pred Value : 0.92513

```

```

##          Prevalence : 0.08250
##          Detection Rate : 0.01063
##  Detection Prevalence : 0.04000
##          Balanced Accuracy : 0.54839
##
##          'Positive' Class : 1
##
```

*# Comment: this one fail because of overfitting*

Check overfit

```

library(randomForest)

rf.probs <- predict(rf5, train.df, type = "prob")[,2]
rf.pred <- ifelse(rf.probs > 0.2, "1", "0")
rf.pred <- factor(rf.pred, levels = c("0", "1"))
confusionMatrix(rf.pred, train.df$Florence, positive = "1")

## Confusion Matrix and Statistics
##
##          Reference
## Prediction    0     1
##          0 2180    74
##          1    14   132
##
##          Accuracy : 0.9633
##          95% CI : (0.955, 0.9705)
##  No Information Rate : 0.9142
##  P-Value [Acc > NIR] : < 2.2e-16
##
##          Kappa : 0.7308
##
##  Mcnemar's Test P-Value : 3.187e-10
##
##          Sensitivity : 0.64078
##          Specificity : 0.99362
##  Pos Pred Value : 0.90411
##  Neg Pred Value : 0.96717
##          Prevalence : 0.08583
##          Detection Rate : 0.05500
##  Detection Prevalence : 0.06083
##          Balanced Accuracy : 0.81720
##
##          'Positive' Class : 1
##
```

With 5 predictor classweight

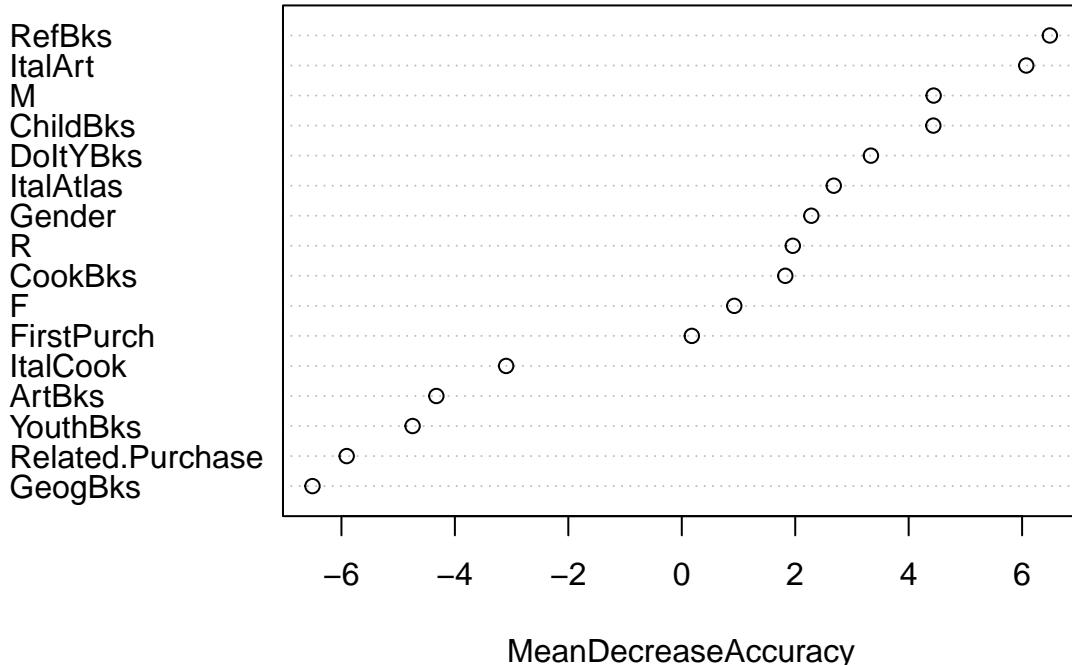
```

library(randomForest)
rf5 <- randomForest(as.factor(Florence) ~ ., data = train.df, ntree = 500,
                     mtry = 5, classwt = c('0' = 0.4, '1' = 0.6), nodesize = 10, importance = TRUE)

varImpPlot(rf5, type = 1)

```

**rf5**



```

rf.probs <- predict(rf5, valid.df, type = "prob")[,2]
rf.pred <- ifelse(rf.probs > 0.2, "1", "0")
rf.pred <- factor(rf.pred, levels = c("0", "1"))
confusionMatrix(rf.pred, valid.df$Florence, positive = "1")

```

```

## Confusion Matrix and Statistics
##
##          Reference
## Prediction 0 1
##           0 726 44
##           1 742 88
##
##          Accuracy : 0.5088
## 95% CI : (0.4839, 0.5335)
## No Information Rate : 0.9175
## P-Value [Acc > NIR] : 1
##
##          Kappa : 0.0473
##
## McNemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.6667
##          Specificity : 0.4946
## Pos Pred Value : 0.1060
## Neg Pred Value : 0.9429

```

```

##                  Prevalence : 0.0825
##                  Detection Rate : 0.0550
##      Detection Prevalence : 0.5188
##                  Balanced Accuracy : 0.5806
##
##      'Positive' Class : 1
##

rf.probs <- predict(rf5, train.df, type = "prob")[,2]
rf.pred <- ifelse(rf.probs > 0.2, "1", "0")
rf.pred <- factor(rf.pred, levels = c("0", "1"))
confusionMatrix(rf.pred, train.df$Florence, positive = "1")

## Confusion Matrix and Statistics
##
##      Reference
## Prediction    0     1
##      0 1307     0
##      1   887   206
##
##                  Accuracy : 0.6304
##                  95% CI : (0.6107, 0.6498)
##      No Information Rate : 0.9142
##      P-Value [Acc > NIR] : 1
##
##                  Kappa : 0.2019
##
##      Mcnemar's Test P-Value : <2e-16
##
##                  Sensitivity : 1.00000
##                  Specificity : 0.59572
##      Pos Pred Value : 0.18847
##      Neg Pred Value : 1.00000
##                  Prevalence : 0.08583
##                  Detection Rate : 0.08583
##      Detection Prevalence : 0.45542
##      Balanced Accuracy : 0.79786
##
##      'Positive' Class : 1
##
# Comment: this one fail because of overfitting

```

Random forest with 6 predictors

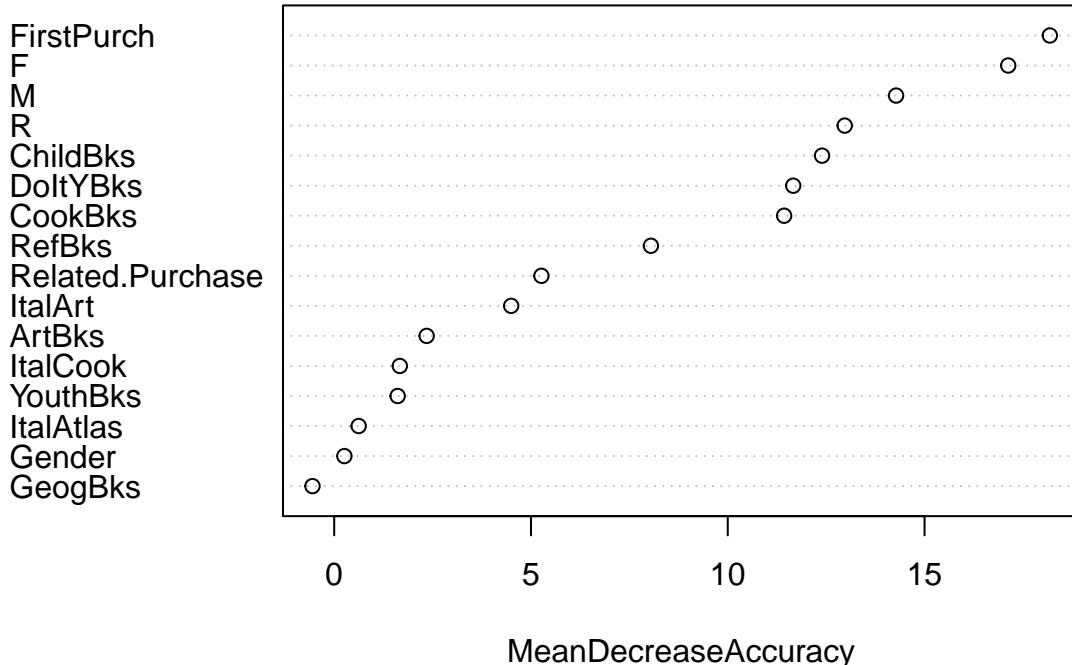
```

library(randomForest)
rf6 <- randomForest(as.factor(Florence) ~ ., data = train.df, ntree = 500,
                     mtry = 6, nodesize = 10, importance = TRUE)

varImpPlot(rf6, type = 1)

```

## rf6



```

rf.probs <- predict(rf6, valid.df, type = "prob")[,2]
rf.pred <- ifelse(rf.probs > 0.2, "1", "0")
rf.pred <- factor(rf.pred, levels = c("0", "1"))
confusionMatrix(rf.pred, valid.df$Florence, positive = "1")

```

```

## Confusion Matrix and Statistics
##
##             Reference
## Prediction      0      1
##           0 1407   115
##           1    61    17
##
##                  Accuracy : 0.89
##                  95% CI : (0.8736, 0.9049)
##      No Information Rate : 0.9175
##      P-Value [Acc > NIR] : 0.9999
##
##                  Kappa : 0.1072
##
##  Mcnemar's Test P-Value : 6.469e-05
##
##                  Sensitivity : 0.12879
##                  Specificity  : 0.95845
##      Pos Pred Value : 0.21795
##      Neg Pred Value : 0.92444

```

```

##           Prevalence : 0.08250
##           Detection Rate : 0.01063
##   Detection Prevalence : 0.04875
##           Balanced Accuracy : 0.54362
##
##           'Positive' Class : 1
##
# Comment: this one fail because of overfitting

```

check overfitting

```

rf.probs <- predict(rf6, train.df, type = "prob")[,2]
rf.pred <- ifelse(rf.probs > 0.2, "1", "0")
rf.pred <- factor(rf.pred, levels = c("0", "1"))
confusionMatrix(rf.pred, train.df$Florence, positive = "1")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0     1
##       0 2176    61
##       1    18   145
##
##           Accuracy : 0.9671
##           95% CI : (0.9591, 0.9739)
##   No Information Rate : 0.9142
##   P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.7683
##
##   Mcnemar's Test P-Value : 2.297e-06
##
##           Sensitivity : 0.70388
##           Specificity : 0.99180
##           Pos Pred Value : 0.88957
##           Neg Pred Value : 0.97273
##           Prevalence : 0.08583
##           Detection Rate : 0.06042
##   Detection Prevalence : 0.06792
##           Balanced Accuracy : 0.84784
##
##           'Positive' Class : 1
##

```