Tree Classification Example

A young bank wants to explore ways of converting its liability (deposit) customers to personal loan customers. A campaign the bank ran for liability customers showed a healthy conversion rate of over 9% successes. This has encouraged the retail marketing department to devise smarter campaigns with better target marketing. The goal of our analysis is to model the previous campaign's customer behavior to analyze what combination of factors make a customer more likely to accept a personal loan. This will serve as the basis for the design of a new campaign.

Column Name	Description
ID	Customer ID
Age	Customer's age in completed years
Experience	#years of professional experience
Income	Annual income of the customer ($000)
ZIPCode	Home Address ZIP code.
Family	Family size of the customer
CCAvg	Avg. spending on credit cards per month ($000)
Education	Education Level. 1: Undergrad; 2: Graduate; 3: Advanced/Professional
Mortgage	Value of house mortgage if any. ($000)
Personal Loan	Did this customer accept the personal loan offered in the last campaign?
Securities Account	Does the customer have a securities account with the bank?
CD Account	Does the customer have a certificate of deposit (CD) account with the bank?
Online	Does the customer use internet banking facilities?
CreditCard	Does the customer use a credit card issued by UniversalBank?

Analysis

Loading libraries

```
library(rpart)
library(rpart.plot)
library(caret)    # used for confusionMatrix
```

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

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

Loading data

```
bank.df <- read.csv("UniversalBank.csv")
dim(bank.df)
```

```
## [1] 5000   14
```

Drop ID and zip code columns.

```
bank.df <- bank.df[ , -c(1, 5)]
bank.df$Personal.Loan <- as.factor(bank.df$Personal.Loan)
```

Partition 5000 observations randomly into training (3000 records) and validation (2000 records).

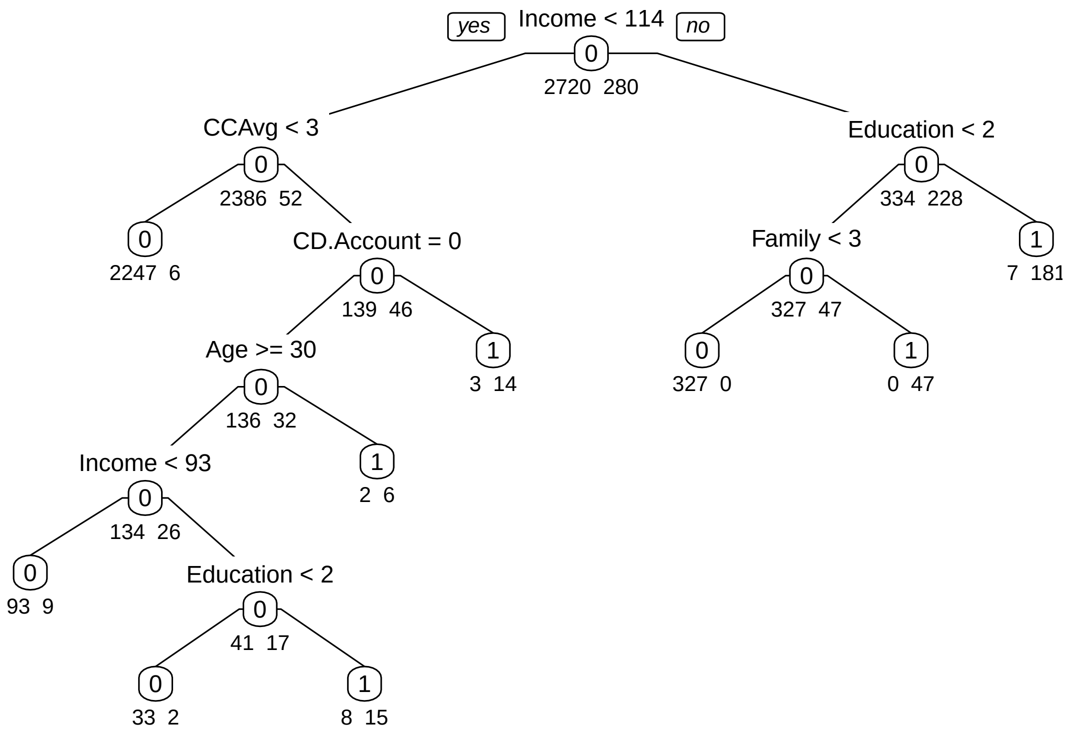
```r
set.seed(131)
train.index <- sample(c(1:dim(bank.df)[1]), dim(bank.df)[1]*0.6)
train.df <- bank.df[train.index, ]
valid.df <- bank.df[-train.index, ]
```

Build a classification tree

```r
default.ct <- rpart(Personal.Loan ~ ., data = train.df, method = "class")
```

Plot tree

```r
prp(default.ct, type = 1, extra = 1, under = TRUE, split.font = 1, varlen = -10)
```



Let us investigate a larger tree

```r
deeper.ct <- rpart(Personal.Loan ~ ., data = train.df, method = "class", cp = 0, minsplit = 1)
```

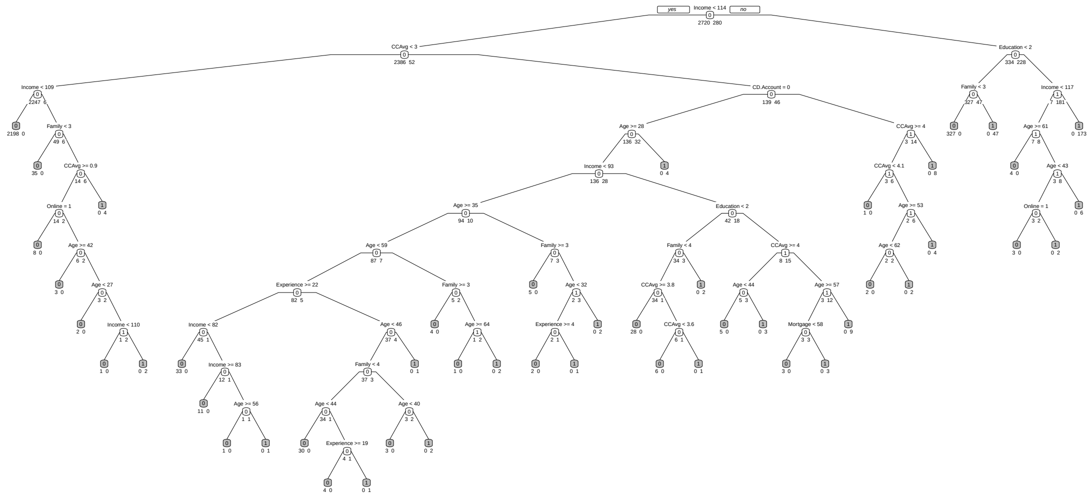
Count number of leaves

```r
length(deeper.ct$frame$var[deeper.ct$frame$var == "<leaf>"])
```

```
## [1] 47
```

Plot tree

```r
prp(deeper.ct, type = 1, extra = 1, under = TRUE, split.font = 1, varlen = -10,
      box.col=ifelse(deeper.ct$frame$var == "<leaf>", 'gray', 'white'))
```



Compare predictions from both trees on both the training and validation data. We need to set argument `type = "class"` in `predict()` to generate predicted class membership.

Generate confusion matrix for training data

```
default.ct.point.pred.train <- predict(default.ct, train.df, type = "class")
deeper.ct.point.pred.train <- predict(deeper.ct, train.df, type = "class")
cm.default.train <- confusionMatrix(default.ct.point.pred.train, train.df$Personal.Loan)
cm.deeper.train <- confusionMatrix(deeper.ct.point.pred.train, train.df$Personal.Loan)
print(cm.default.train)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2700   17
##          1   20  263
##
##                Accuracy : 0.9877
##                  95% CI : (0.983, 0.9913)
##     No Information Rate : 0.9067
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9275
##
##  Mcnemar's Test P-Value : 0.7423
##
##             Sensitivity : 0.9926
##             Specificity : 0.9393
##          Pos Pred Value : 0.9937
##          Neg Pred Value : 0.9293
##              Prevalence : 0.9067
##          Detection Rate : 0.9000
##    Detection Prevalence : 0.9057
##       Balanced Accuracy : 0.9660
##
##        'Positive' Class : 0
##
```

```r
print(cm.deeper.train)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2720    0
##          1    0  280
##
##                Accuracy : 1
##                  95% CI : (0.9988, 1)
##     No Information Rate : 0.9067
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.9067
##          Detection Rate : 0.9067
##    Detection Prevalence : 0.9067
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 0
##
```

Generate confusion matrix for validation data

```r
default.ct.point.pred.valid <- predict(default.ct, valid.df, type = "class")
deeper.ct.point.pred.valid <- predict(deeper.ct, valid.df, type = "class")
cm.default.valid <- confusionMatrix(default.ct.point.pred.valid, valid.df$Personal.Loan)
cm.deeper.valid <- confusionMatrix(deeper.ct.point.pred.valid, valid.df$Personal.Loan)
print(cm.default.valid)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1778   14
##          1   22  186
##
##                Accuracy : 0.982
##                  95% CI : (0.9752, 0.9874)
##     No Information Rate : 0.9
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9017
##
##  Mcnemar's Test P-Value : 0.2433
##
##             Sensitivity : 0.9878
```

```
##             Specificity : 0.9300
##          Pos Pred Value : 0.9922
##          Neg Pred Value : 0.8942
##              Prevalence : 0.9000
##          Detection Rate : 0.8890
##    Detection Prevalence : 0.8960
##       Balanced Accuracy : 0.9589
##
##        'Positive' Class : 0
##
```

```
print(cm.deeper.valid)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1785   17
##          1   15  183
##
##                Accuracy : 0.984
##                  95% CI : (0.9775, 0.989)
##     No Information Rate : 0.9
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9107
##
##  Mcnemar's Test P-Value : 0.8597
##
##             Sensitivity : 0.9917
##             Specificity : 0.9150
##          Pos Pred Value : 0.9906
##          Neg Pred Value : 0.9242
##              Prevalence : 0.9000
##          Detection Rate : 0.8925
##    Detection Prevalence : 0.9010
##       Balanced Accuracy : 0.9533
##
##        'Positive' Class : 0
##
```

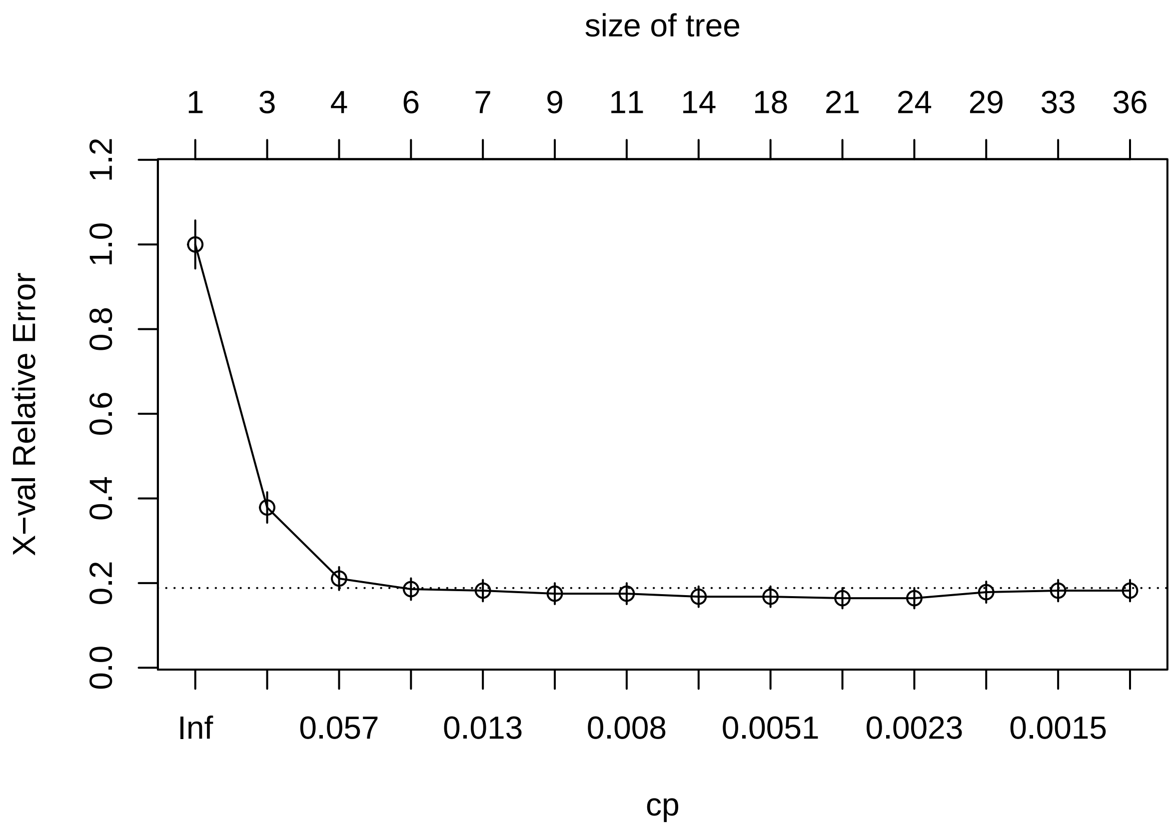
Argument `xval` refers to the number of folds to use in rpart's built-in cross-validation procedure. Argument `cp` sets the smallest value for the complexity parameter.

```
cv.ct <- rpart(Personal.Loan ~ ., data = train.df, method = "class",
            cp = 0.00001, minsplit = 5, xval = 5)
printcp(cv.ct)
```

```
##
## Classification tree:
## rpart(formula = Personal.Loan ~ ., data = train.df, method = "class",
##     cp = 1e-05, minsplit = 5, xval = 5)
##
## Variables actually used in tree construction:
## [1] Age        CCAvg      CD.Account Education  Experience Family     Income
## [8] Mortgage   Online
```

```
##
## Root node error: 280/3000 = 0.093333
##
## n= 3000
##
##           CP nsplit rel error  xerror     xstd
## 1  0.3107143      0  1.000000 1.00000 0.056904
## 2  0.1678571      2  0.378571 0.37857 0.036115
## 3  0.0196429      3  0.210714 0.21071 0.027162
## 4  0.0142857      5  0.171429 0.18571 0.025530
## 5  0.0125000      6  0.157143 0.18214 0.025287
## 6  0.0089286      8  0.132143 0.17500 0.024795
## 7  0.0071429     10  0.114286 0.17500 0.024795
## 8  0.0053571     13  0.092857 0.16786 0.024292
## 9  0.0047619     17  0.071429 0.16786 0.024292
## 10 0.0023810     20  0.057143 0.16429 0.024036
## 11 0.0021429     23  0.050000 0.16429 0.024036
## 12 0.0017857     28  0.039286 0.17857 0.025042
## 13 0.0011905     32  0.032143 0.18214 0.025287
## 14 0.0000100     35  0.028571 0.18214 0.025287
```

```r
plotcp(cv.ct)
```

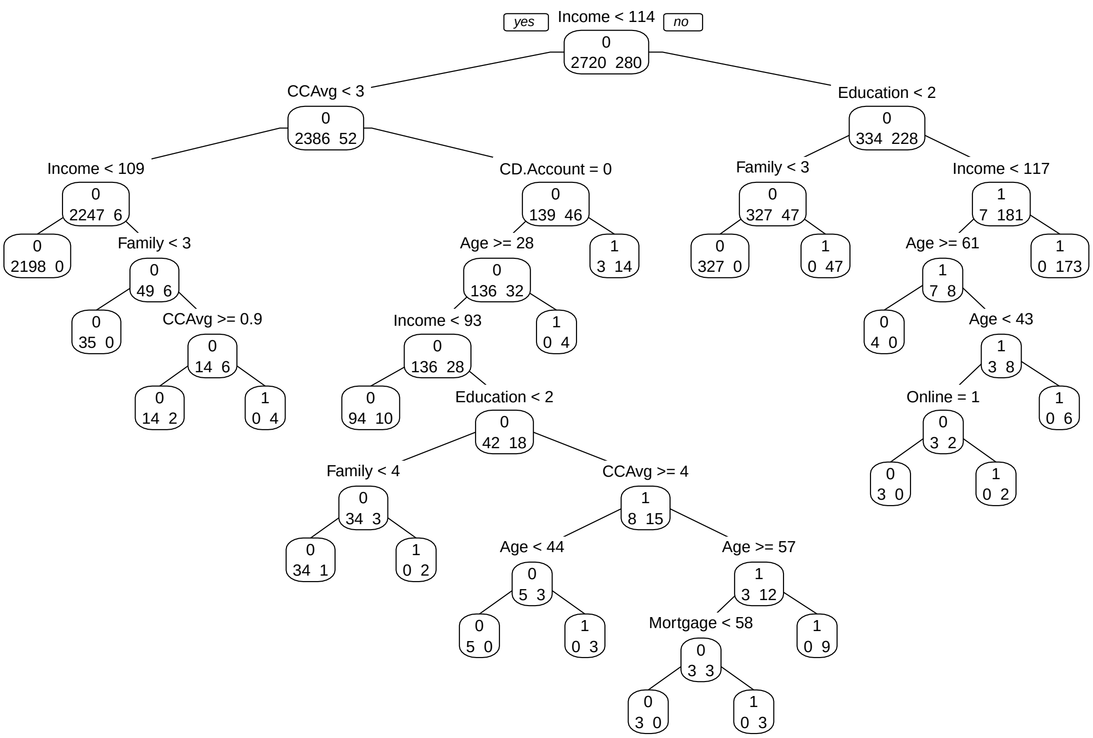


Prune by lower cp

```r
pruned.ct <- prune(cv.ct, cp = cv.ct$cptable[which.min(cv.ct$cptable[,"xerror"]),"CP"])
length(pruned.ct$frame$var[pruned.ct$frame$var == "<leaf>"])
```

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

```r
prp(pruned.ct, type = 1, extra = 1, split.font = 1, varlen = -10)
```



Improved version of pruning

```r
# this is the cp parameter with smallest cv-error
index_cp_min = which.min(cv.ct$cptable[,"xerror"])

# one standard deviation rule
# need to find first cp value for which the xerror is below horizontal line on the plot
(val_h = cv.ct$cptable[index_cp_min, "xerror"] + cv.ct$cptable[index_cp_min, "xstd"])
```

```
## [1] 0.1883219
```

```r
(index_cp_std = Position(function(x) x < val_h, cv.ct$cptable[, "xerror"]))
```

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

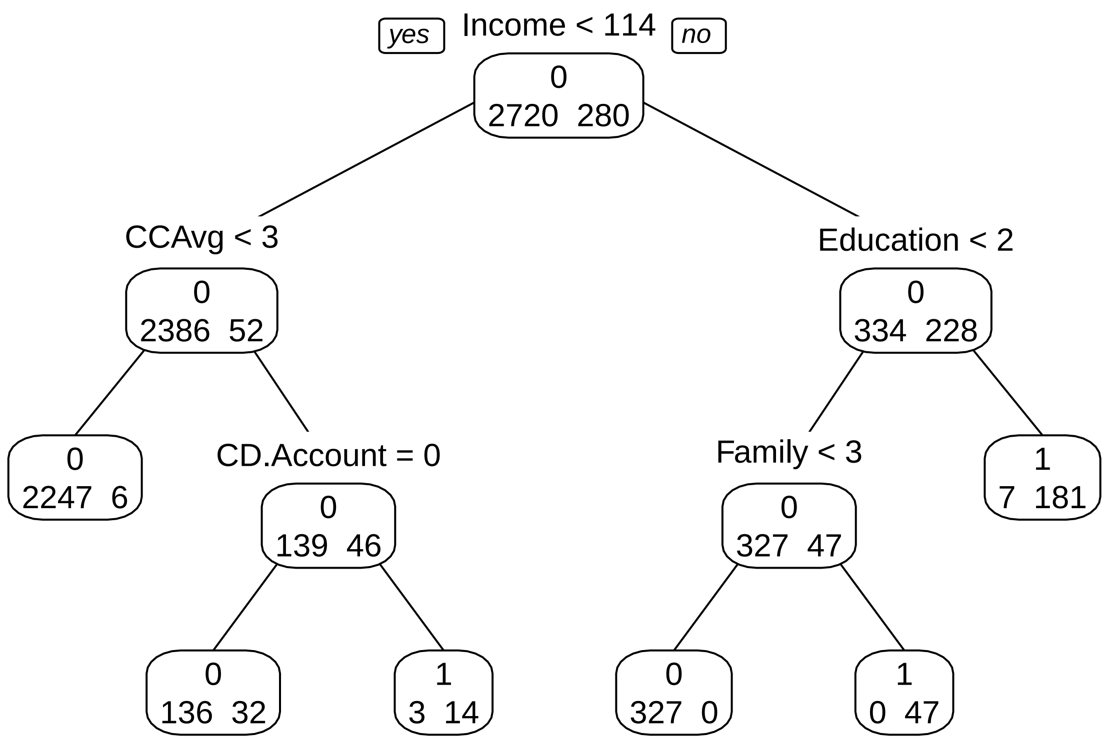
```r
(cp_std = cv.ct$cptable[ index_cp_std, "CP" ])
```

```
## [1] 0.01428571
```

```r
pruned.ct <- prune(cv.ct, cp = cp_std)
length(pruned.ct$frame$var[pruned.ct$frame$var == "<leaf>"])
```

```
## [1] 6
```

```r
prp(pruned.ct, type = 1, extra = 1, split.font = 1, varlen = -10)
```


yes
Income < 114
no
0
2720 280
CCAvg < 3
0
2386 52
Education < 2
0
334 228
0
2247 6
CD.Account = 0
0
139 46
Family < 3
0
327 47
1
7 181
0
136 32
1
3 14
0
327 0
1
0 47