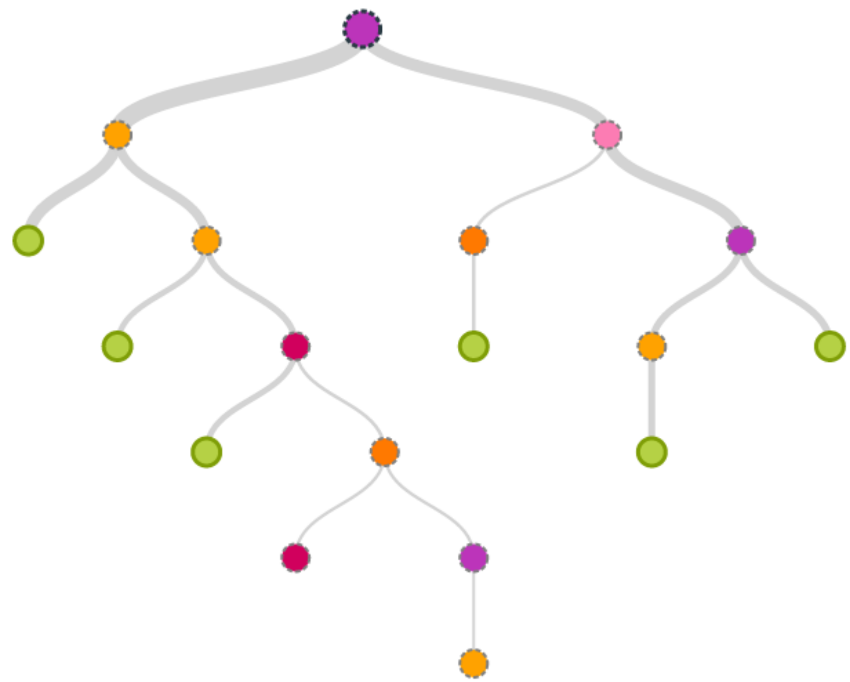


Decision Tree R Exercise: Classification Tree

Pilsung Kang

School of Industrial Management Engineering

Korea University

R Exercise: Data Set

- Personal Loan
  - ✓ Purpose: identify future customer who will use the personal loan service based on his/her demographic information and banking service history

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	ID	Age	Experienc	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal L	Securities	CD Accou	Online	CreditCard
2	1	25	1	49	91107	4	1.6	1	0	0	1	0	0	0
3	2	45	19	34	90089	3	1.5	1	0	0	1	0	0	0
4	3	39	15	11	94720	1	1	1	0	0	0	0	0	0
5	4	35	9	100	94112	1	2.7	2	0	0	0	0	0	0
6	5	35	8	45	91330	4	1	2	0	0	0	0	0	1
7	6	37	13	29	92121	4	0.4	2	155	0	0	0	1	0
8	7	53	27	72	91711	2	1.5	2	0	0	0	0	1	0
9	8	50	24	22	93943	1	0.3	3	0	0	0	0	0	1
10	9	35	10	81	90089	3	0.6	2	104	0	0	0	1	0

  - A total of 14 variables (columns)
  - ID, ZIP Code: irrelevant column (remove)
  - Personal loan: target variable

R Exercise: Preprocessing (Post-Pruning)

- Data: Personal loan prediction
  - ✓ Write a performance evaluation function
  - ✓ Load the data
  - ✓ Use the "tree" package
  - ✓ Transform the target variable as "factor" type
  - ✓ Divide the dataset into the training (1,500) and validation (1,000)

R Exercise: Preprocessing (Post-Pruning)

- Install packages & write a performance evaluation function

```
# Performance Evaluation Function -----------------------------------------
perf_eval <- function(cm){
     # True positive rate: TPR (Recall)
     TPR <- cm[2,2]/sum(cm[2,])
     # Precision
     PRE <- cm[2,2]/sum(cm[,2])
     # True negative rate: TNR
     TNR <- cm[1,1]/sum(cm[1,])
     # Simple Accuracy
     ACC <- (cm[1,1]+cm[2,2])/sum(cm)
     # Balanced Correction Rate
     BCR <- sqrt(TPR*TNR)
     # F1-Measure
     F1 <- 2*TPR*PRE/(TPR+PRE)
     return(c(TPR, PRE, TNR, ACC, BCR, F1))
}
Perf_Table <- matrix(0, nrow = 2, ncol = 6)
rownames(Perf_Table) <- c("Post-Pruning" , "Pre-Pruning")
colnames(Perf_Table) <- c("TPR", "Precision", "TNR", "Accuracy", "BCR",
                            "F1-Measure")
Perf_Table
```

R Exercise: Preprocessing (Post-Pruning)

- Load the dataset and set the input/target indices

```
# Load the data & Preprocessing
Ploan <- read.csv("Personal Loan.csv")
input_idx <- c(2,3,4,6,7,8,9,11,12,13,14)
target_idx <- 10

Ploan_input <- Ploan[,input_idx]
Ploan_target <- as.factor(Ploan[,target_idx])

trn.idx <- 1:1500
tst.idx <- 1501:2500
```

✓ [ID], [ZIP Code], [Personal Loan] are excluded from the input variable set

✓ [Personal Loan] is set to the target variable

✓ Convert the variable type of [Personal Loan] from binary(0/1) to factor for building a classification model

✓ Use the first 1,500 customers to train the model and use the remaining 1,000 customers to validate the model

R Exercise: Training and Evaluation (Post-Pruning)

- Training and evaluating CART

```
# Classification and Regression Tree (CART) -------------------------------------
install.packages("tree")
library(tree)

CART_trn <- data.frame(Ploan_input[trn_idx,], PloanYN = Ploan_target[trn_idx])
CART_tst <- data.frame(Ploan_input[tst_idx,], PloanYN = Ploan_target[tst_idx])

# Training the tree
CART_post <- tree(PloanYN ~ ., CART_trn)
summary(CART_post)
```

✓ tree( ) function

- Formula: the left side of (~) is target and the right side of (~) is input variables
- Y ~ X1: Set X1 as the input variable and Y as the target variable
- Y ~ X1+X2: Set X1 and X2 as the input variables and Y as the target variable
- Y ~ .: Set Y as the target variable and all the remaining variables as the input variables

R Exercise: Training and Evaluation (Post-Pruning)

- Training and evaluating CART

```
> summary(CART_post)

Classification tree:
tree(formula = PloanYN ~ ., data = CART_trn)
Variables actually used in tree construction:
[1] "Income"     "CCAvg"      "CD.Account" "Education"  "Family"
Number of terminal nodes:  10
Residual mean deviance:  0.06996 = 104.2 / 1490
Misclassification error rate: 0.01267 = 19 / 1500
```

✓ A total of 5 variables are used at least once as a split variable during the tree construction

- [Income], [CCAvg], [CD.Account], [Education], [Family]

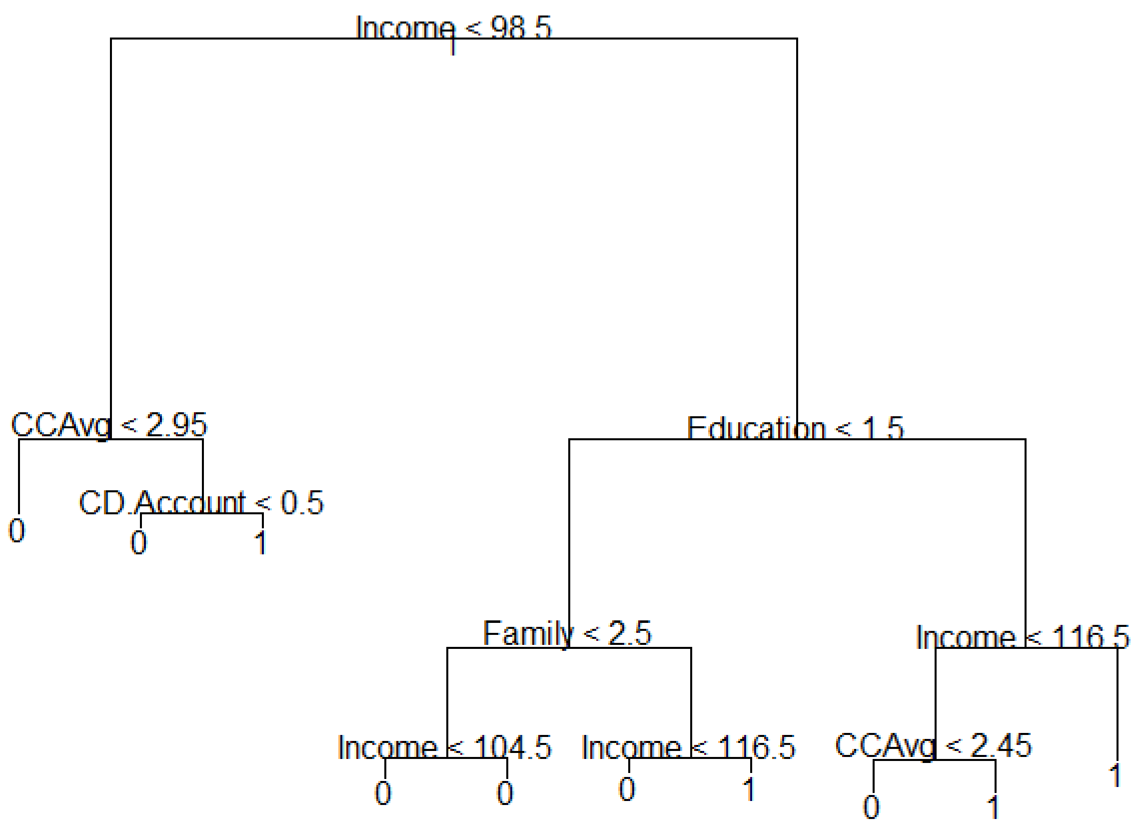
✓ The number of terminal/leaf nodes = 10

✓ Training error: 1.267% (19 out of 1,500 observations)

R Exercise: Training and Evaluation (Post-Pruning)

- Training and evaluating CART

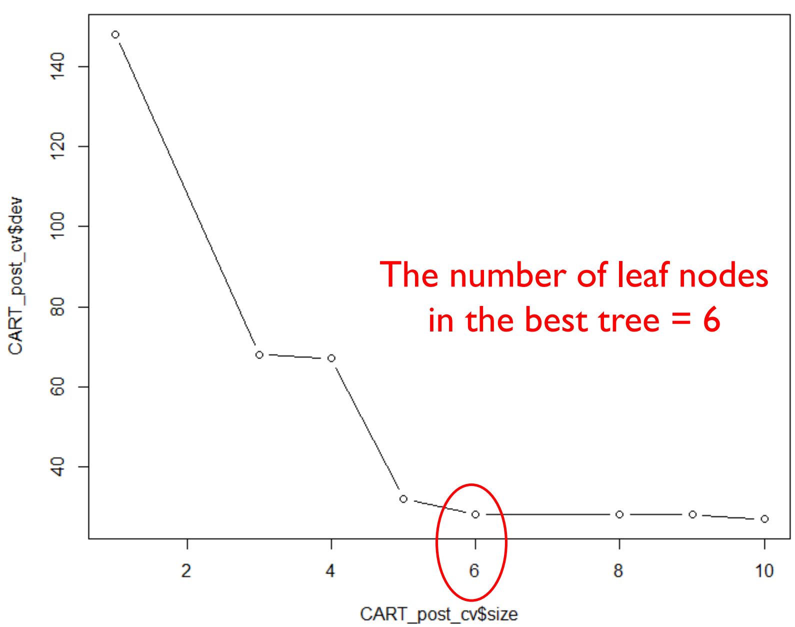
```
# Plot the tree
plot(CART_model)
text(CART_model, pretty = 1)
```



R Exercise: Training and Evaluation (Post-Pruning)

- Find the best tree based on cross-validation

```r
# Find the best tree
set.seed(12345)
CART_post_cv <- cv.tree(CART_post, FUN = prune.misclass)

# Plot the pruning result
plot(CART_post_cv$size, CART_post_cv$dev, type = "b")
CART_post_cv
```

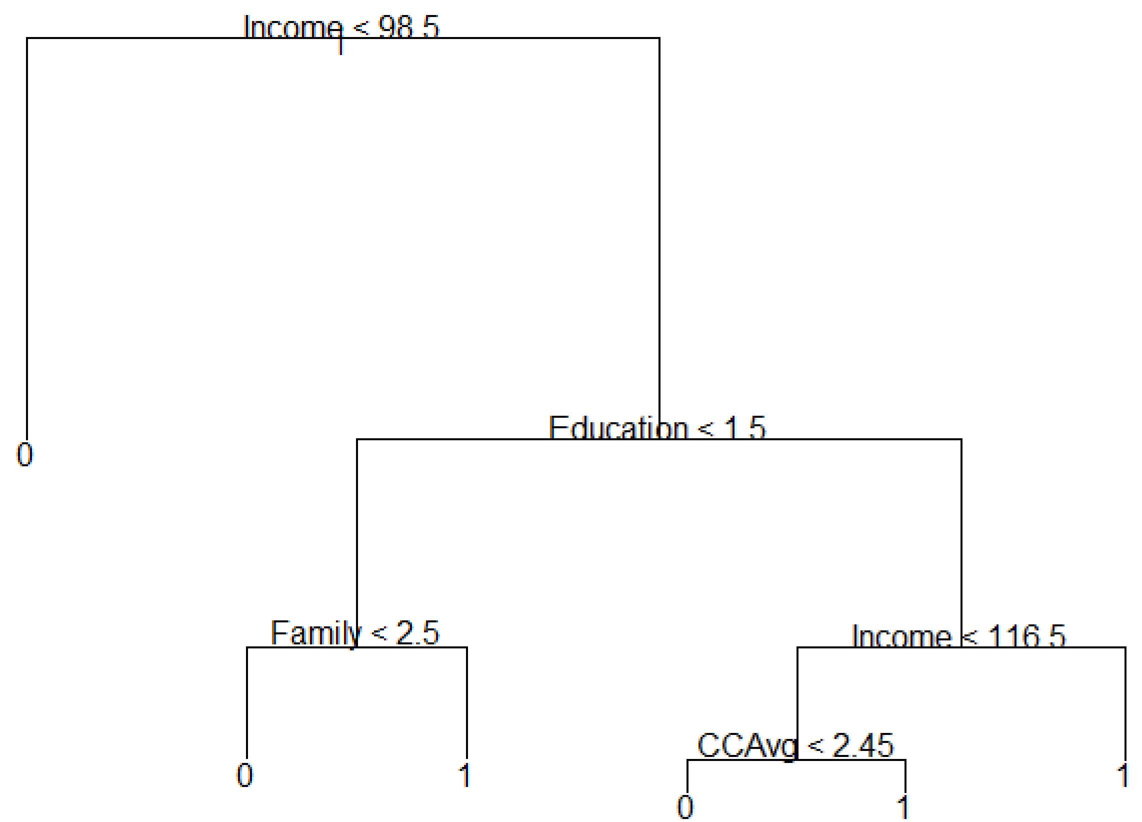



```
> CART_post_cv
$size
[1] 10  9  8  6  5  4  3  1

$dev
[1]  27  28  28  28  32  67  68 148

$k
[1] -Inf  0.0  1.0  1.5  9.0 17.0 19.0 42.0

$method
[1] "misclass"

attr(,"class")
[1] "prune"         "tree.sequence"
```

R Exercise: Training and Evaluation (Post-Pruning)

- Find the best tree based on cross-validation

```
# Select the final model
CART_post_pruned <- prune.misclass(CART_post, best = 6)
plot(CART_post_pruned)
text(CART_post_pruned, pretty = 1)
```



R Exercise: Training and Evaluation (Post-Pruning)

- Prediction performance with the best tree

```
# Prediction
CART_post_prey <- predict(CART_post_pruned, CART_tst, type = "class")
CART_post_cm <- table(CART_tst$PloanYN, CART_post_cm)
CART_post_cm

Perf_Table[1,] <- perf_eval(CART_post_cm)
Perf_Table
```

Confusion matrix		Predicted	
		No (0)	Yes (1)
Actual	No (0)	888	8
	Yes (1)	11	93

	TPR	Precision	TNR	Accuracy	BCR	F1-Measure
Post-Pruning	0.8942	0.9208	0.9911	0.9810	0.9414	0.9073
Pre-Pruning	0	0	0	0	0	0

R Exercise: Preprocessing (Pre-Pruning)

- Install necessary packages

```r
# CART with Post-Pruning -------------------------------
# For CART
install.packages("party")
library(party)

# For AUROC
install.packages("ROCR")
library(ROCR)
```

- Divide the dataset into training/validation/test datasets (4:2:4)

```r
# Divide the dataset into training/validation/test datasets
trn_idx <- 1:1000
val_idx <- 1001:1500
tst_idx <- 1501:2500

CART_trn <- data.frame(Ploan_input[trn_idx,], PloanYN = Ploan_target[trn_idx])
CART_val <- data.frame(Ploan_input[val_idx,], PloanYN = Ploan_target[val_idx])
CART_tst <- data.frame(Ploan_input[tst_idx,], PloanYN = Ploan_target[tst_idx])
```

R Exercise: Training and Evaluation (Pre-Pruning)

- Define parameter search space for pre-pruning

```
# Construct single tree and evaluation # tree parameter settings
min_criterion = c(0.9, 0.95, 0.99)
min_split = c(10, 30, 50, 100)
max_depth = c(0, 10, 5)

CART_pre_search_result =
matrix(0,length(min_criterion)*length(min_split)*length(max_depth),11)

colnames(CART_pre_search_result) <- c("min_criterion", "min_split", "max_depth",
"TPR", "Precision", "TNR", "ACC", "BCR", "F1", "AUROC", "N_leaves")
```

✓ min_criterion: minimum statistical significance to split the current node

✓ min_split: minimum number of observations to consider splitting the current node

✓ max_depth: maximun depth of the entire tree

R Exercise: Training and Evaluation (Pre-Pruning)

- Find the optimal parameters

```
iter_cnt = 1

for (i in 1:length(min_criterion)) {
     for ( j in 1:length(min_split)) {
          for ( k in 1:length(max_depth)) {

               cat("CART Min criterion:", min_criterion[i], ", Min split:",
                    min_split[j], ", Max depth:", max_depth[k], "\n")
```

✓ Run for loop for three difference model parameters

✓ cat( ): print the strings in the console

R Exercise: Training and Evaluation (Pre-Pruning)

- Find the optimal parameters

```
tmp_control = ctree_control(mincriterion = min_criterion[i],
                            minsplit = min_split[j], maxdepth = max_depth[k])

tmp_tree <- ctree(PloanYN ~ ., data = CART_trn, controls = tmp_control)

tmp_tree_val_prediction <- predict(tmp_tree, newdata = CART_val)
```

✓ ctree( ): training a classification tree

- Arg 1: Formula
- Arg 2: Dataset for training
- Arg 3: Parameter values

✓ predict( ): make predictions

- Arg 1: Trained model
- Arg 2: Dataset to predict

R Exercise: Training and Evaluation (Pre-Pruning)

- Find the optimal parameters

```
tmp_tree_val_prediction <- predict(tmp_tree, newdata = CART_val)

tmp_tree_val_response <- treeresponse(tmp_tree, newdata = CART_val)

tmp_tree_val_prob <- 1-unlist(tmp_tree_val_response,
                              use.names=F)[seq(1,nrow(CART_val)*2,2)]

tmp_tree_val_rocr <- prediction(tmp_tree_val_prob, CART_val$PloanYN)

# Confusion matrix for the validation dataset
tmp_tree_val_cm <- table(CART_val$PloanYN, tmp_tree_val_prediction)
```

- ✓ tmp_tree_val_prediction: binary outcome
- ✓ tmp_tree_val_response: predicted probability for the two classes
- ✓ tmp_tree_val_prob: predicted probability for "1" class (for AUROC computation)
- ✓ tmp_tree_val_rocr: summary data to draw ROC curve

R Exercise: Training and Evaluation (Pre-Pruning)

- Find the optimal parameters

```
# parameters
CART_pre_search_result[iter_cnt,1] = min_criterion[i]
CART_pre_search_result[iter_cnt,2] = min_split[j]
CART_pre_search_result[iter_cnt,3] = max_depth[k]

# Performances from the confusion matrix
CART_pre_search_result[iter_cnt,4:9] = perf_eval(tmp_tree_val_cm)
# AUROC
CART_pre_search_result[iter_cnt,10] = unlist(performance(tmp_tree_val_rocr,
"auc")@y.values)
# Number of leaf nodes
CART_pre_search_result[iter_cnt,11] = length(nodes(tmp_tree,
unique(where(tmp_tree))))
iter_cnt = iter_cnt + 1
```

✓ Compute seven performance metrics and stores them with the corresponding parameters

R Exercise: Training and Evaluation (Pre-Pruning)

- Find the optimal parameters

```
# Find the best set of parameters
CART_pre_search_result <- CART_pre_search_result[order(CART_pre_search_result[,10],
                                                 decreasing = T),]
CART_pre_search_result

best_criterion <- CART_pre_search_result[1,1]
best_split <- CART_pre_search_result[1,2]
best_depth <- CART_pre_search_result[1,3]
```

  ✓ Sort the performance matrix in terms of AUROC

  ✓ Find the best parameter values

R Exercise: Training and Evaluation (Pre-Pruning)

- Find the optimal parameters

```
> CART_pre_search_result
```

	min_criterion	min_split	max_depth	TPR	Precision	TNR	ACC	BCR	F1	AUROC	N_leaves
[1,]	0.90	10	0	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[2,]	0.90	10	10	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[3,]	0.90	10	5	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[4,]	0.90	30	0	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[5,]	0.90	30	10	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[6,]	0.90	30	5	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[7,]	0.90	50	0	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[8,]	0.90	50	10	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[9,]	0.90	50	5	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[10,]	0.90	100	0	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[11,]	0.90	100	10	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[12,]	0.90	100	5	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[13,]	0.95	10	0	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[14,]	0.95	10	10	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[15,]	0.95	10	5	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[16,]	0.95	30	0	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[17,]	0.95	30	10	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[18,]	0.95	30	5	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[19,]	0.95	50	0	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[20,]	0.95	50	10	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[21,]	0.95	50	5	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[22,]	0.95	100	0	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[23,]	0.95	100	10	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[24,]	0.95	100	5	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9920568	6
[25,]	0.99	10	0	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9777389	5
[26,]	0.99	10	10	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9777389	5
[27,]	0.99	10	5	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9777389	5
[28,]	0.99	30	0	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9777389	5
[29,]	0.99	30	10	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9777389	5
[30,]	0.99	30	5	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9777389	5
[31,]	0.99	50	0	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9777389	5
[32,]	0.99	50	10	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9777389	5
[33,]	0.99	50	5	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9777389	5
[34,]	0.99	100	0	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9777389	5
[35,]	0.99	100	10	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9777389	5
[36,]	0.99	100	5	0.7678571	1	1	0.974	0.8762746	0.8686869	0.9777389	5

R Exercise: Training and Evaluation (Pre-Pruning)

- Training the tree with the best parameters

```
# Construct the best tree
tree_control = ctree_control(mincriterion = best_criterion, minsplit = best_split,
                             maxdepth = best_depth)

# Use the training and validation dataset to train the best tree
CART_trn <- rbind(CART_trn, CART_val)
CART_pre <- ctree(PloanYN ~ ., data = CART_trn, controls = tree_control)
CART_pre_prediction <- predict(CART_pre, newdata = CART_tst)
CART_pre_response <- treeresponse(CART_pre, newdata = CART_tst)

# Performance of the best tree
CART_pre_cm <- table(CART_tst$PloanYN, CART_pre_prediction)
CART_pre_cm Perf_Table[2,] <- perf_eval(CART_pre_cm)
Perf_Table
```

R Exercise: Training and Evaluation (Pre-Pruning)

- Performance comparison

✓ Post-pruning

Confusion matrix		Predicted	
		No (0)	Yes (1)
Actual	No (0)	888	8
	Yes (1)	11	93

✓ Pre-pruning

Confusion matrix		Predicted	
		No (0)	Yes (1)
Actual	No (0)	891	5
	Yes (1)	19	85

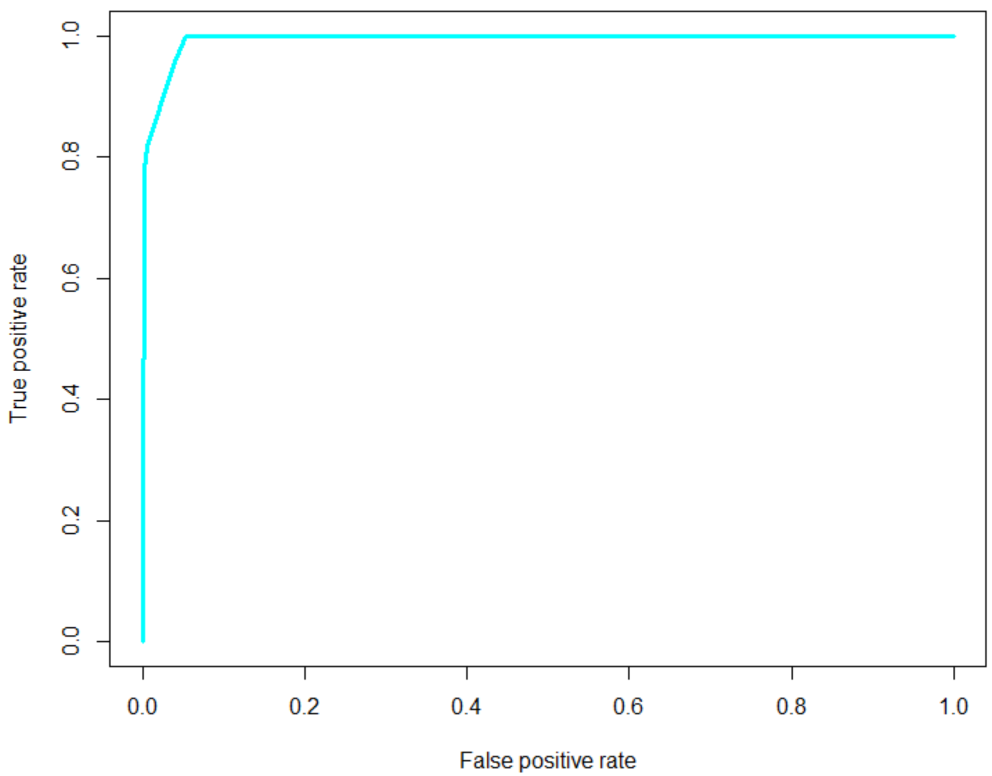
✓ Post-pruning vs. Pre-pruning

	TPR	Precision	TNR	Accuracy	BCR	F1-Measure
Post-Pruning	0.8942	0.9208	0.9911	0.9810	0.9414	0.9073
Pre-Pruning	0.8173	0.9444	0.9944	0.9760	0.9015	0.8762

R Exercise: Training and Evaluation (Pre-Pruning)

- Plot the ROC curve

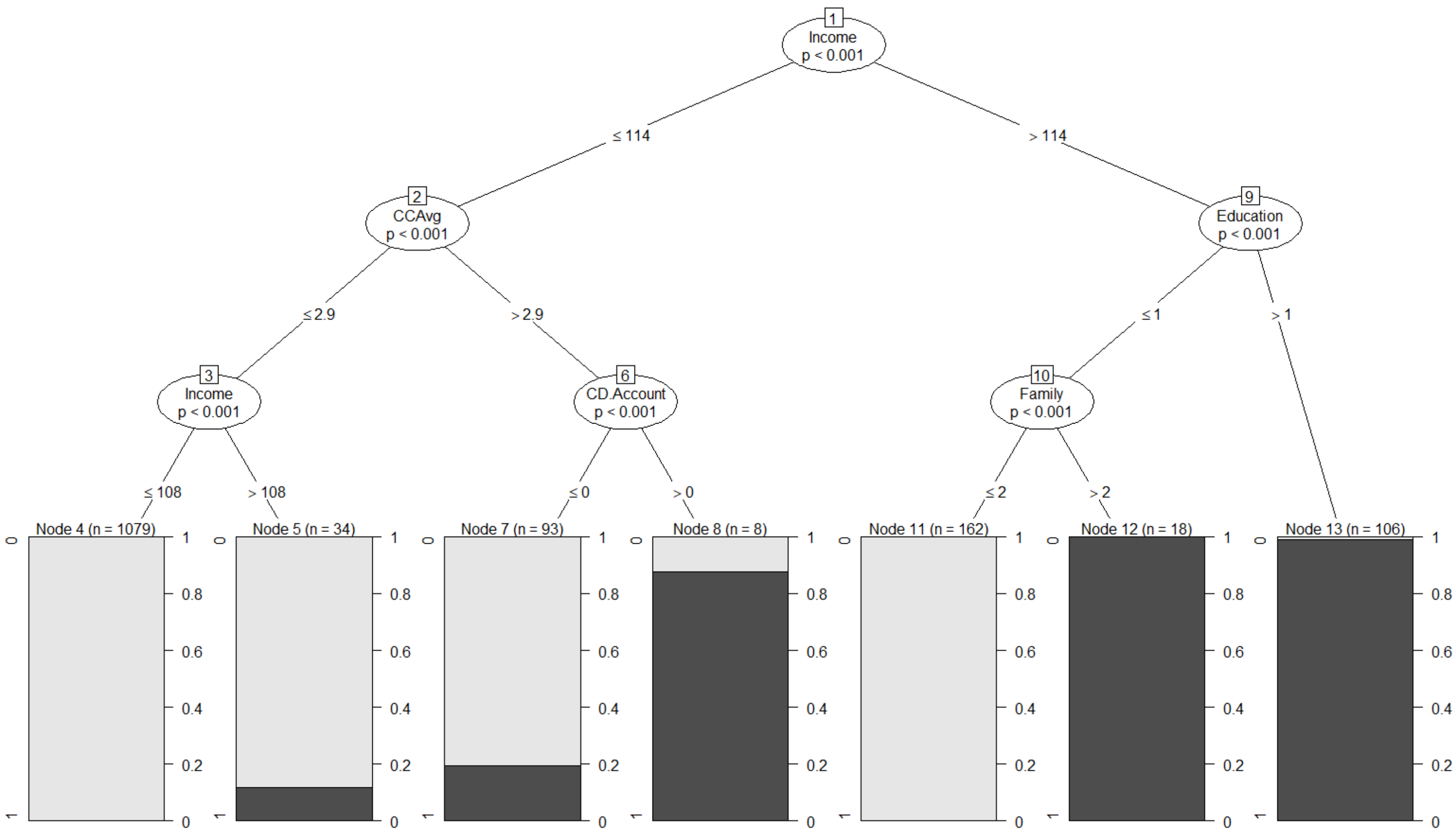
```
# Plot the ROC
CART_pre_prob <- 1-unlist(CART_pre_response,use.names=F)[seq(1,nrow(CART_tst)*2,2)]
CART_pre_rocr <- prediction(CART_pre_prob, CART_tst$PloanYN)
CART_pre_perf <- performance(CART_pre_rocr, "tpr","fpr")
plot(CART_pre_perf, col=5, lwd = 3)
```



R Exercise: Training and Evaluation (Pre-Pruning)

- Plot the best tree

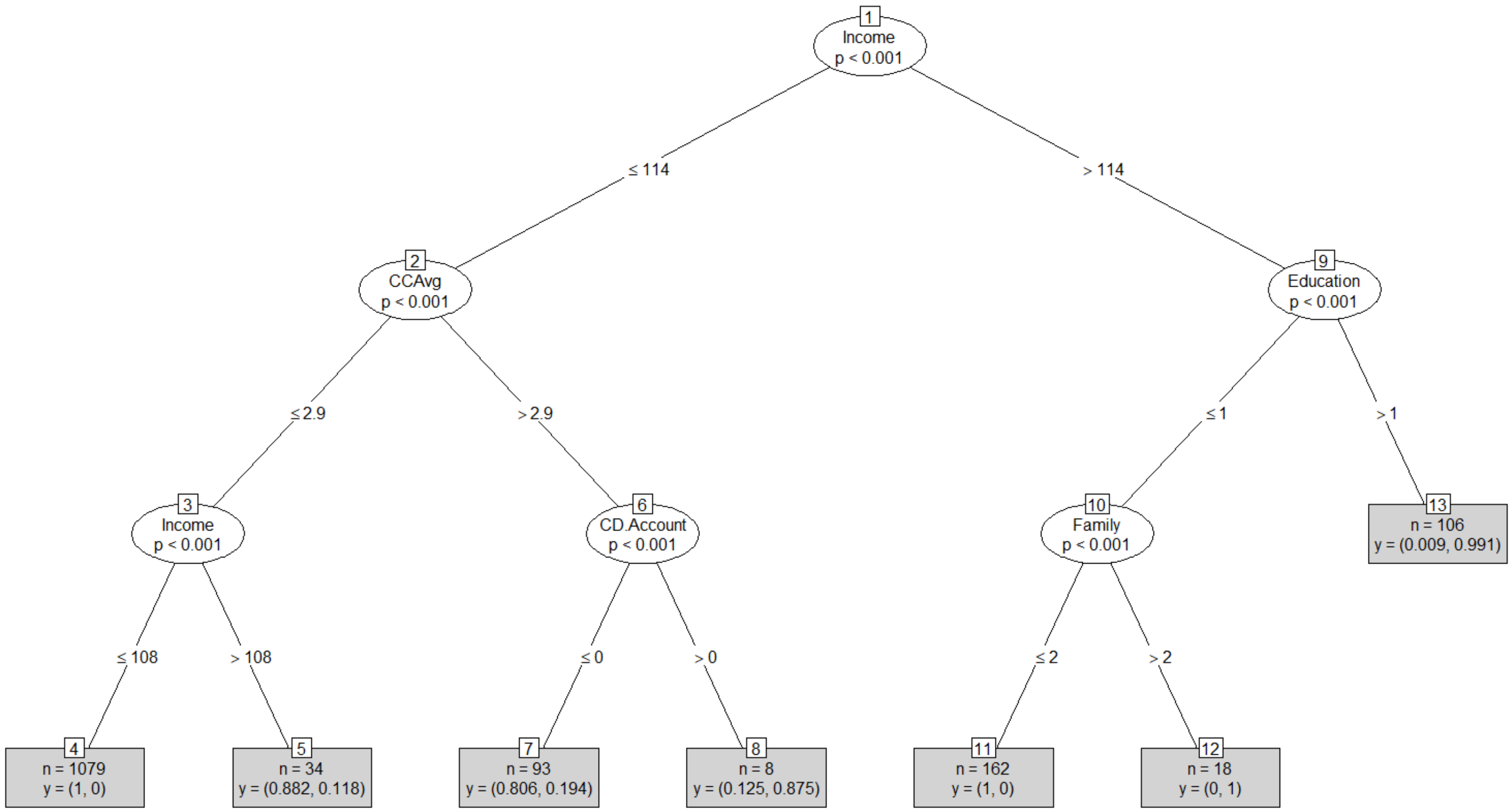
```
# Plot the best tree
plot(CART_pre)
plot(CART_pre, type="simple")
```



R Exercise: Training and Evaluation (Pre-Pruning)

- Plot the best tree

```
# Plot the best tree
plot(CART_pre)
plot(CART_pre, type="simple")
```




ANY
questions?