Classification and Regression Trees

What is Decision tree?

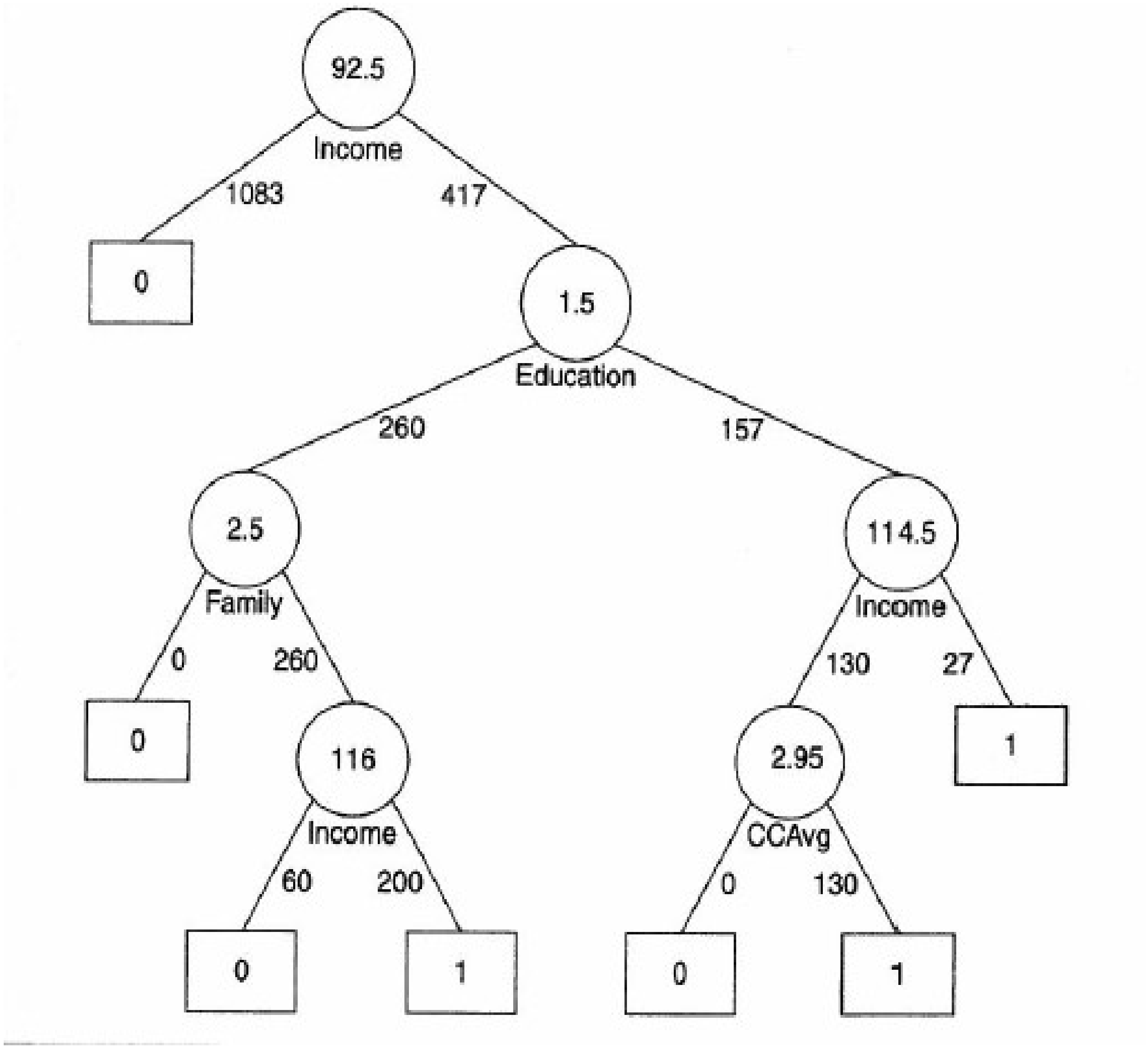
The data-driven method that can be used for both classification (called classification tree) and prediction (call regression tree).

Nodes of Decision Trees

- The squared terminal nodes indicate the predicted classes. They are called the leaves of the tree.

- The circles represent decision nodes.

Classification Tree

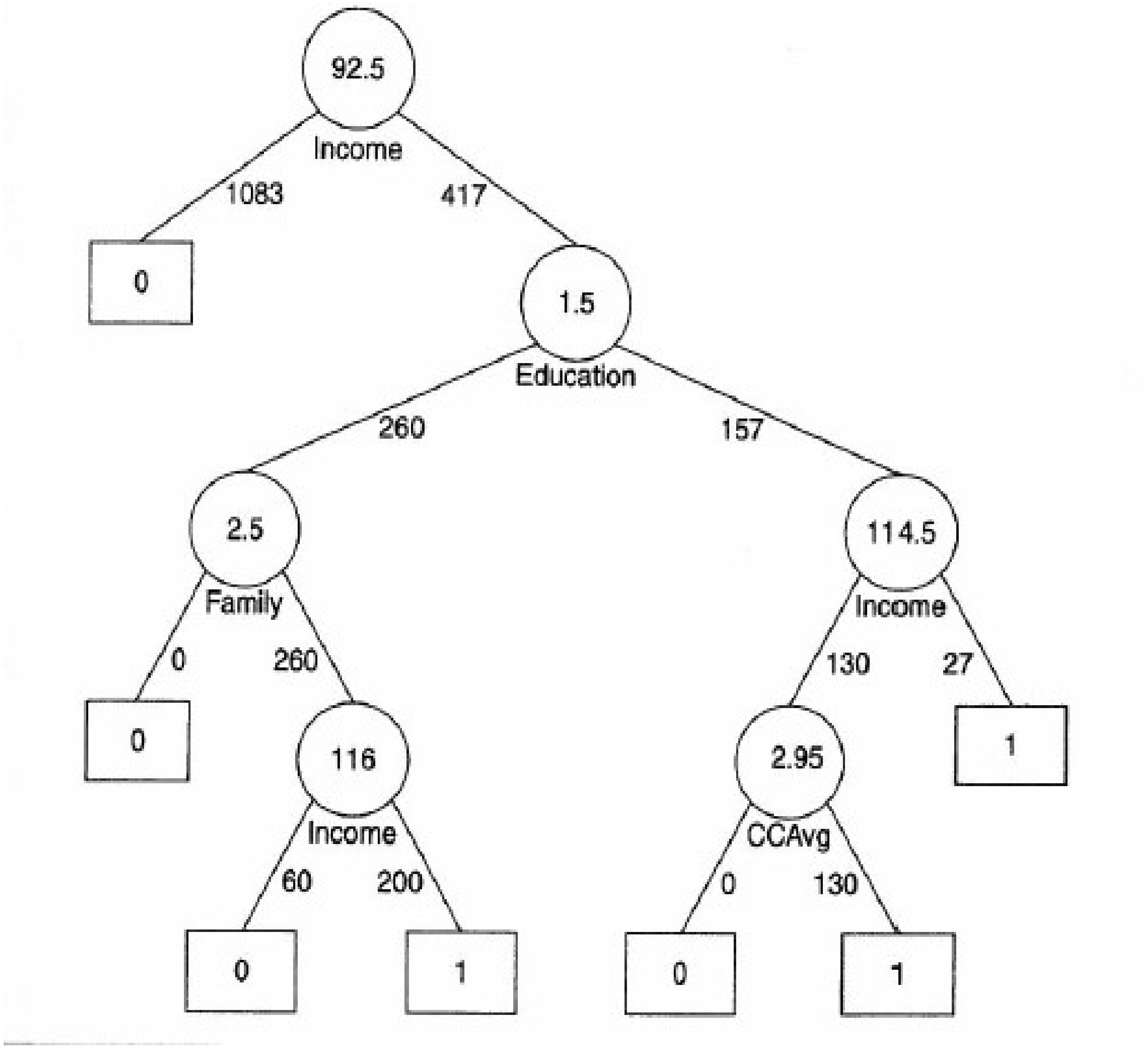


Summary of the Tree Structure

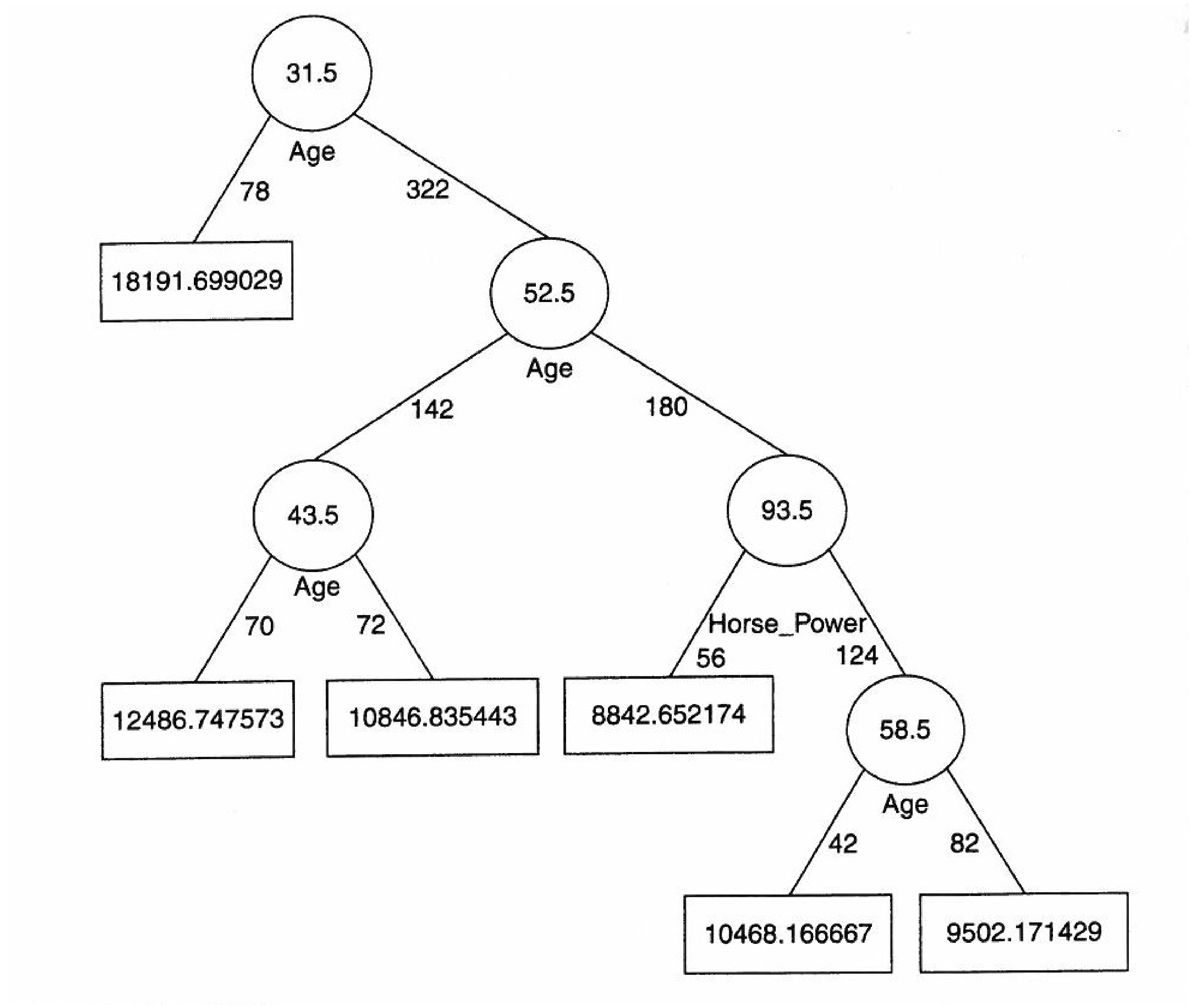
Based on the classification tree generated from XLMiner

- The numbers inside the circle are the splitting values.
- The name of the variable chosen for splitting at that node is shown below the node.
- The numbers on the left fork at a decision node are the number of records in the decision node that had values less than or equal to the splitting value.
- The numbers on the right fork are the number of that had a greater value.

Classification Tree



Regression Tree



Classification Tree

Example: Riding Mowers

A riding-mower manufacturer would like to find a way of classifying families in a city into those likely to purchase a riding mower and those not likely to buy one.

y = ownership of riding mower (owner, nonowner)

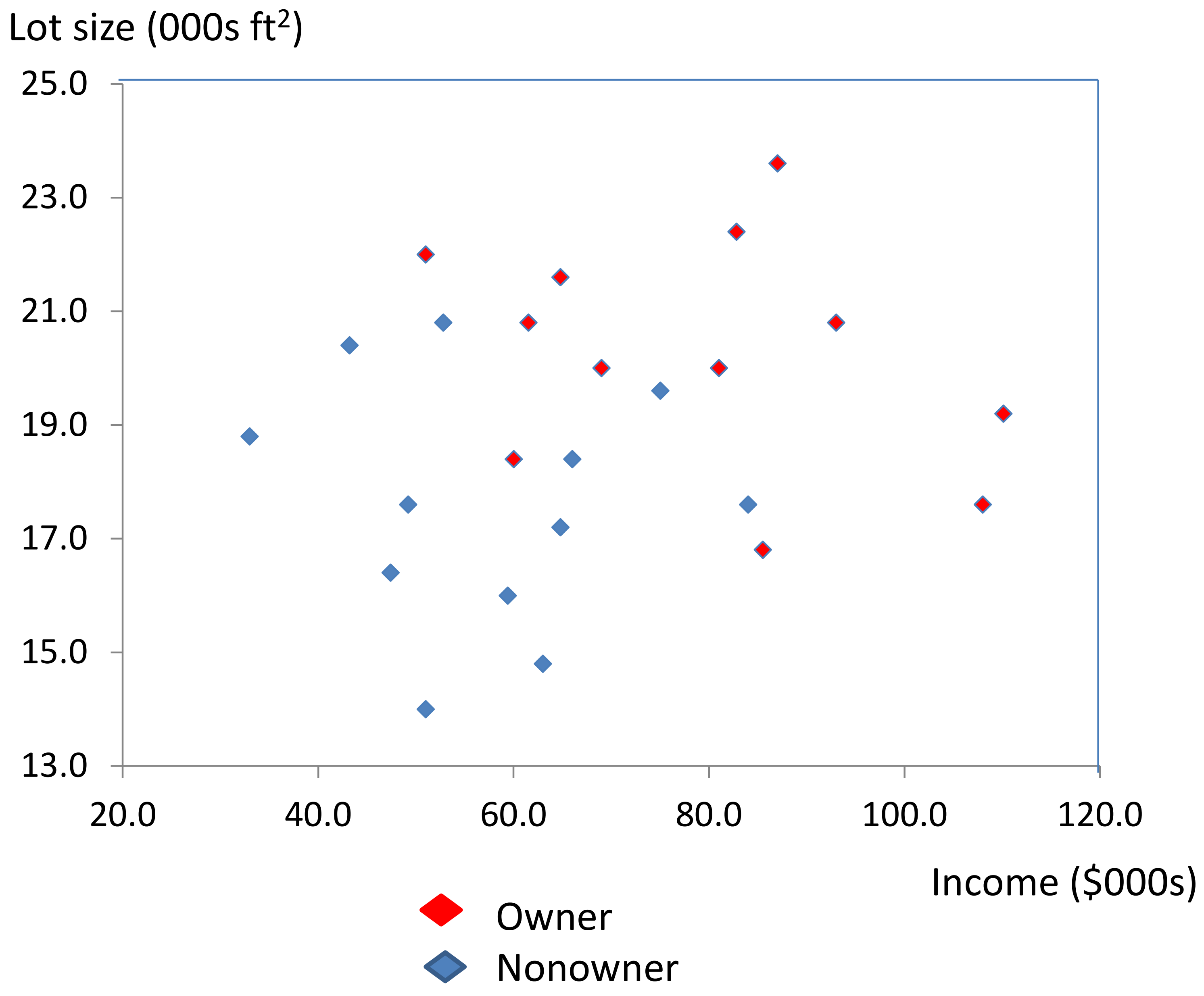
x1 = income ($1000)

x2 = lot size (1000 ft^2)

Example: Riding Mowers

Household number	Income ($000s)	Lot Size (000s ft2)	Ownership of Riding Mower
1	60.0	18.4	Owner
2	85.5	16.8	Owner
3	64.8	21.6	Owner
4	61.5	20.8	Owner
5	87.0	23.6	Owner
6	110.1	19.2	Owner
7	108.0	17.6	Owner
8	82.8	22.4	Owner
9	69.0	20.0	Owner
10	93.0	20.8	Owner
11	51.0	22.0	Owner
12	81.0	20.0	Owner
13	75.0	19.6	Nonowner
14	52.8	20.8	Nonowner
15	64.8	17.2	Nonowner
16	43.2	20.4	Nonowner
17	84.0	17.6	Nonowner
18	49.2	17.6	Nonowner
19	59.4	16.0	Nonowner
20	66.0	18.4	Nonowner
21	47.4	16.4	Nonowner
22	33.0	18.8	Nonowner
23	51.0	14.0	Nonowner
24	63.0	14.8	Nonowner

Scatter Plot



Recursive Partitioning

- Response y is a categorical variable
- Suppose that we have p independent variables, x_1, x_2, …, x_p. They can be continuous or categorical variables.
- Idea: Recursive partitioning divides up the p-dimensional space of the x variables into nonoverlapping multidimensional rectangles.

Recursive Partitioning

- First, one of the variables is selected, say x_i, and the value of x_i, say s_i, is chosen to split the space into two parts: a part that contains all the points with $xi \le s_i$, and a part that contain $xi > s_i$.
- Keep doing the partitioning in a similar manner.

Recursive Partitioning

- The idea is to divide the entire x space up into rectangles such that each rectangle is as homogeneous or “pure” as possible.
- By “pure” we mean containing points that belong to just one class.

How to Choose the Variable and the Value

Select the variable and value that optimize the selected criterion, e.g. the gini index, entropy, and etc.

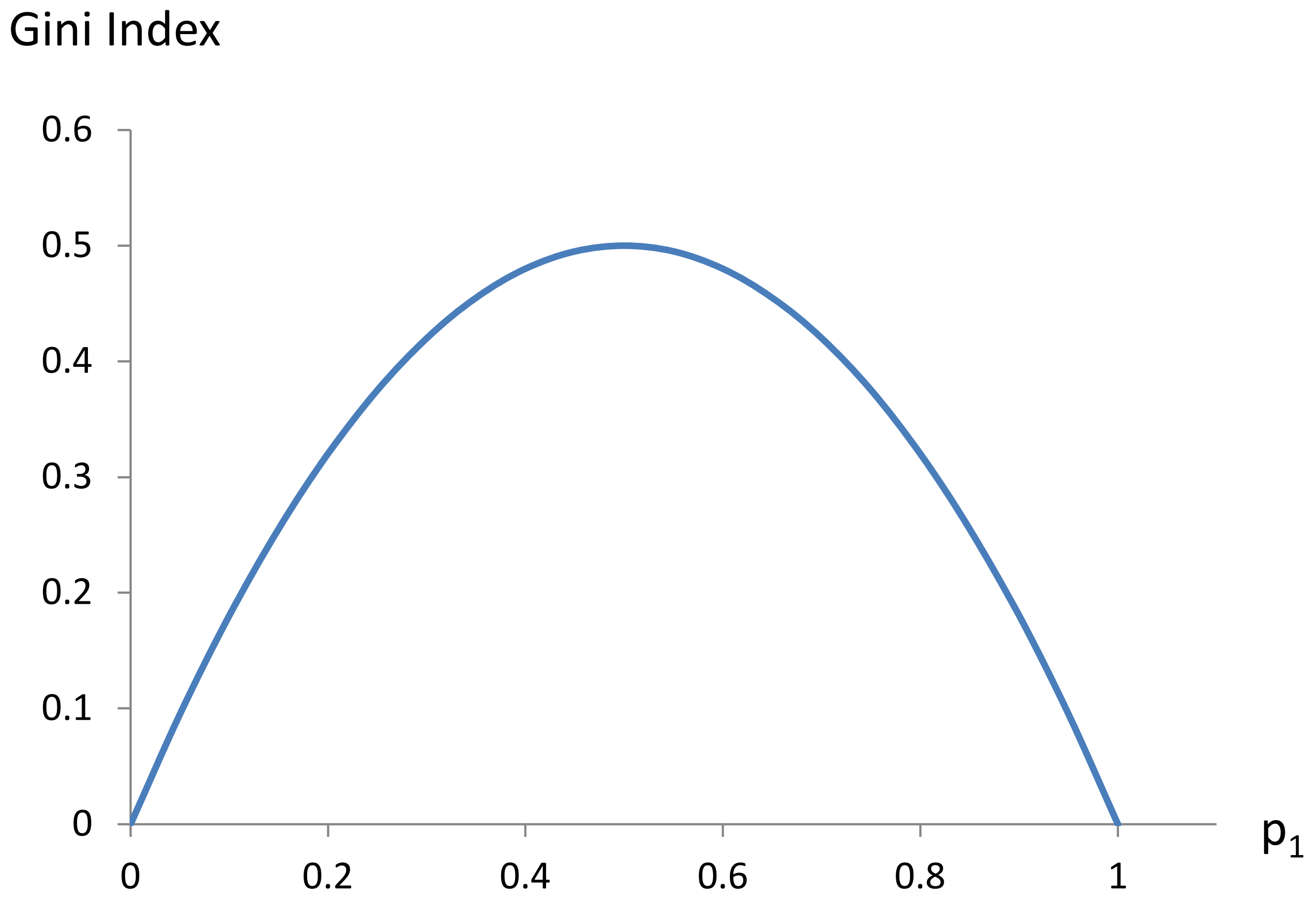
Measure of Impurity

- The Gini impurity index for a rectangle A is defined by

$$I(A) = 1 - \sum_{k=1}^{m} p_k^2$$

where p_k is the proportion of observations in rectangle A that belong to class k.

Gini Index for Two Classes



Measure of Impurity (2)

- The entropy for a rectangle A is defined by

$$Entropy(A) = -\sum_{k=1}^{m} p_k \log_2(p_k)$$

Splitting values

The possible splitting values are the midpoints between pairs of consecutive values for the variable.

The Splitting Value

- Choose the splitting value that reduces impurity the most.

- The reduction in impurity is defined as overall impurity before the split minus the combined impurity of the two rectangles that result from the split.

Note: The combined impurity of the two rectangles is a weighted average of the two impurity measures

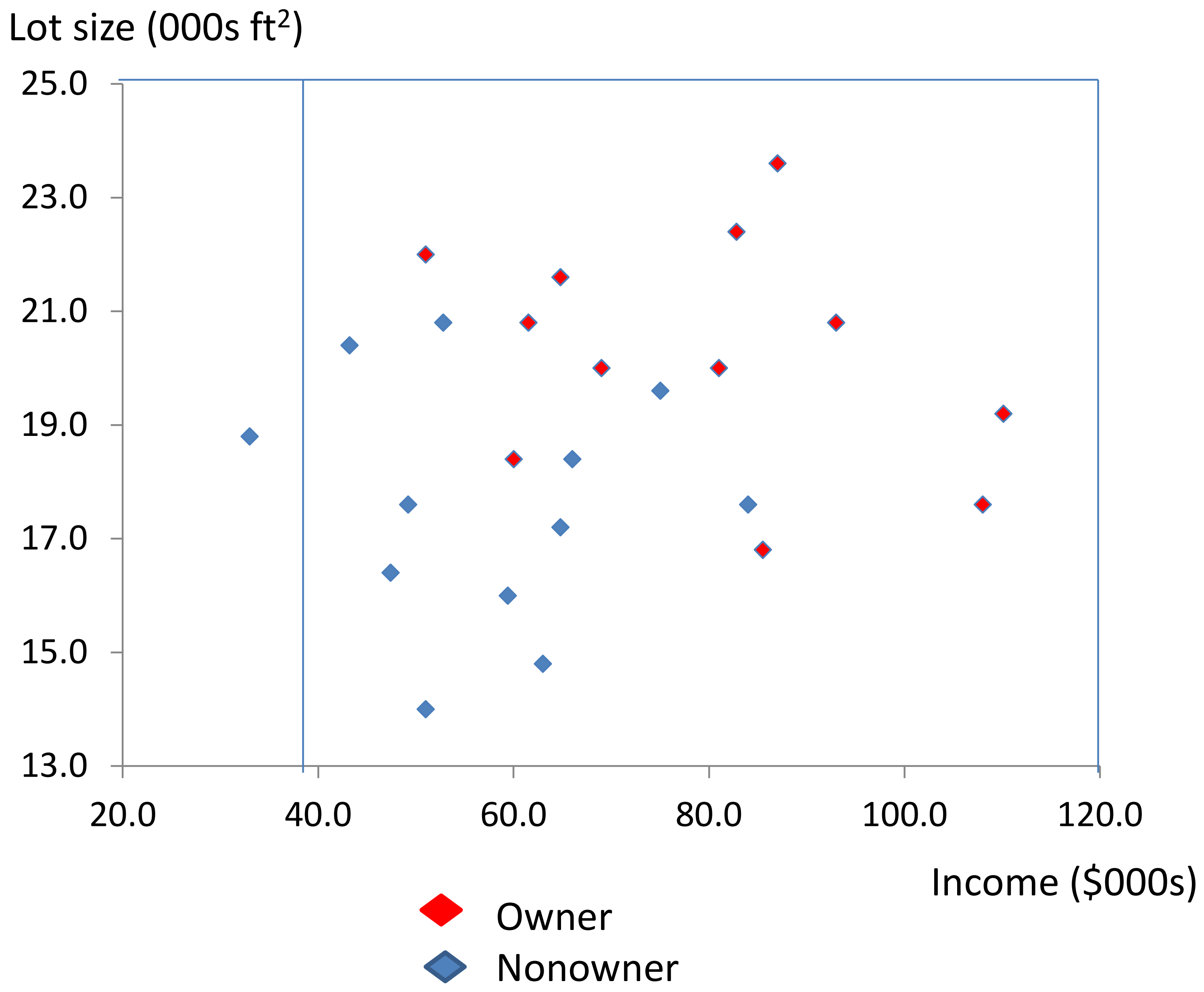
Ordered the Values

Ordered Income ($000s)	Ordered Lot Size (000s ft2)
33.0	14.0
43.2	14.8
47.4	16.0
49.2	16.4
51.0	16.8
51.0	17.2
52.8	17.6
59.4	17.6
60.0	17.6
61.5	18.4
63.0	18.4
64.8	18.8
64.8	19.2
66.0	19.6
69.0	20.0
75.0	20.0
81.0	20.4
82.8	20.8
84.0	20.8
85.5	20.8
87.0	21.6
93.0	22.0
108.0	22.4
110.1	23.6

All Possible Splitting Values

Ordered Income	Possible Splitting value for Income	Ordered Lot Size	Possible Splitting value for Lot Size
33		14	
	38.1		14.4
43.2		14.8	
	45.3		15.4
47.4		16	
	48.3		16.2
49.2		16.4	
.	.	.	.
.	.	.	.
59.4		18.8	
	59.7		19
60.0		19.2	
.	.	.	.
.	.	.	.
108		22.4	
	109.05		23
110.1		23.6	

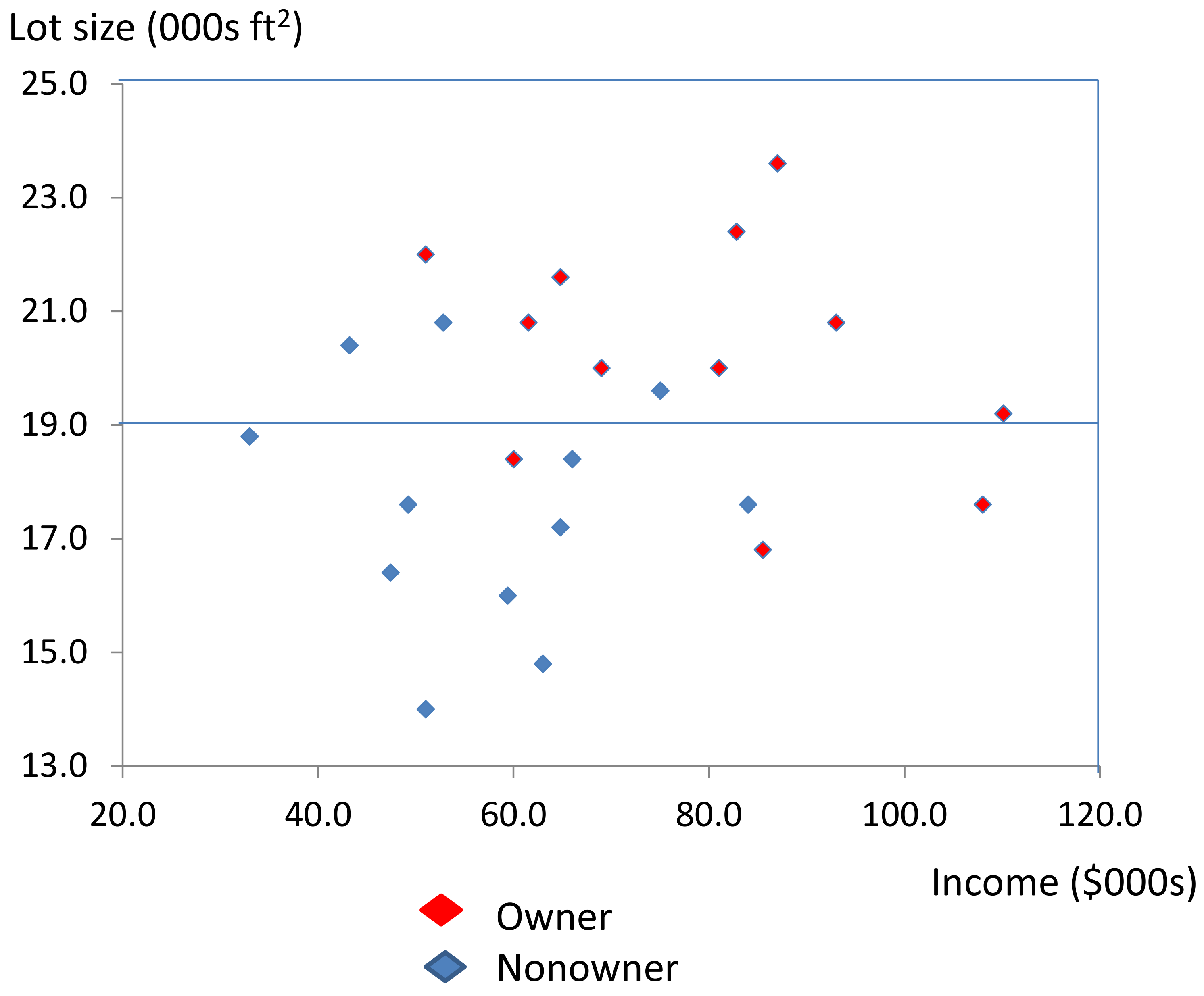
Some Result


Lot size (000s ft²)
25.0
23.0
21.0
19.0
17.0
15.0
13.0
20.0
40.0
60.0
80.0
100.0
120.0
Income ($000s)
Owner
Nonowner

All Possible Splitting Values

Ordered Income	Possible Splitting value for Income	Ordered Lot Size	Possible Splitting value for Lot Size
33		14	
	38.1		14.4
43.2		14.8	
	45.3		15.4
47.4		16	
	48.3		16.2
49.2		16.4	
.	.	.	.
.	.	.	.
59.4		18.8	
	59.7		19
60.0		19.2	
.	.	.	.
.	.	.	.
108		22.4	
	109.05		23
110.1		23.6	

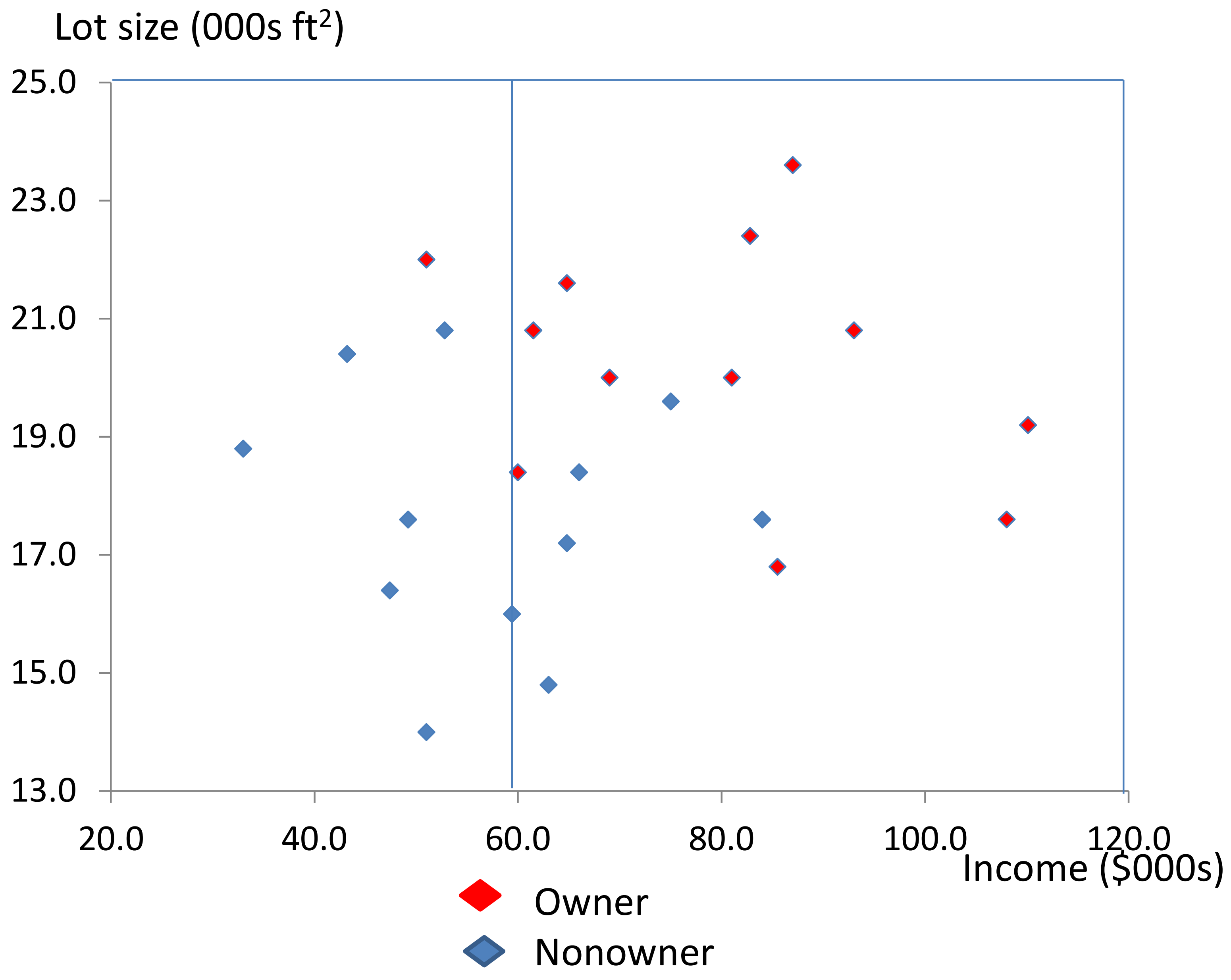
Some Result


Lot size (000s ft²)
25.0
23.0
21.0
19.0
17.0
15.0
13.0
20.0
40.0
60.0
80.0
100.0
120.0
Income ($000s)
Owner
Nonowner

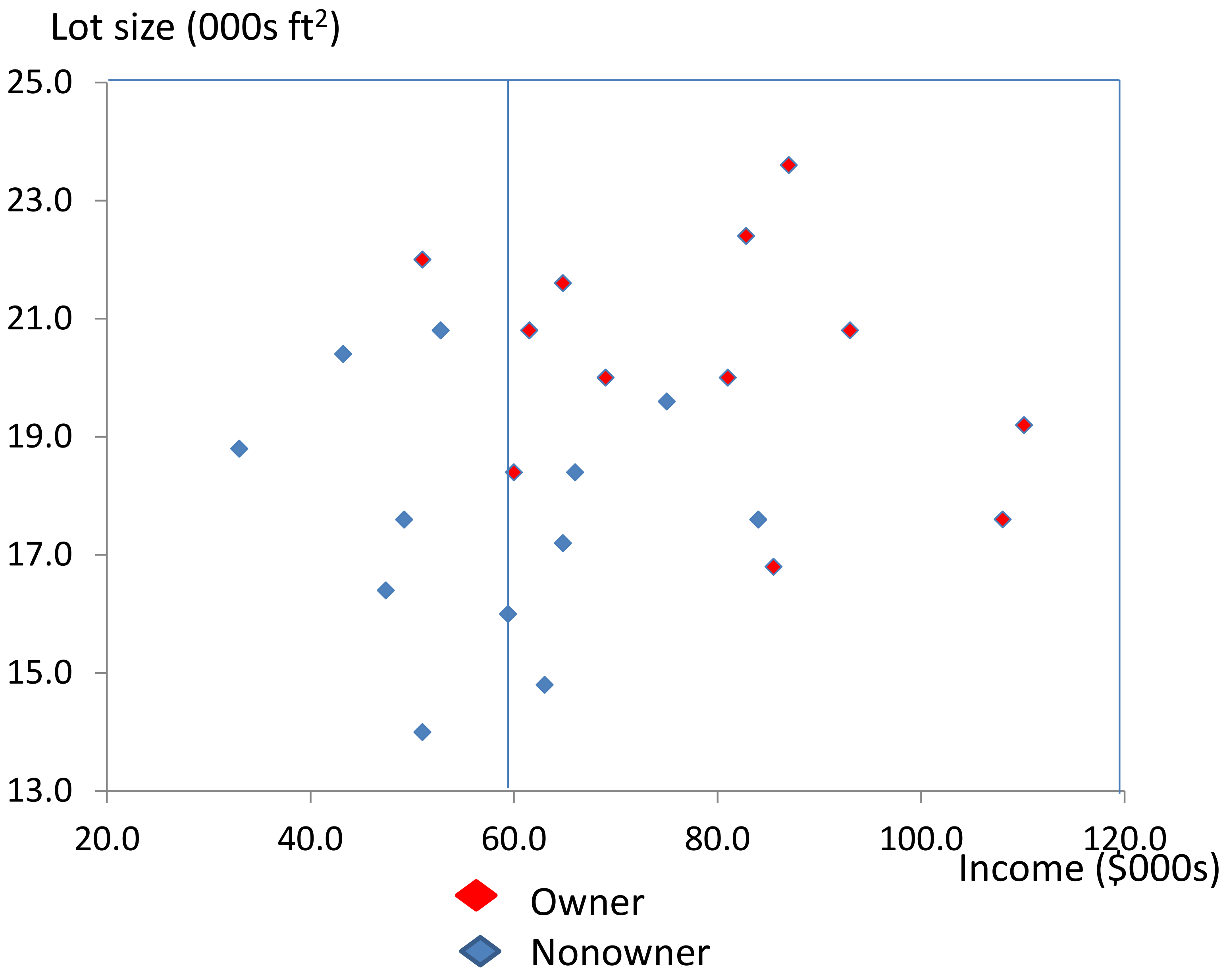
All Possible Splitting Values

Ordered Income	Possible Splitting value for Income	Ordered Lot Size	Possible Splitting value for Lot Size
33		14	
	38.1		14.4
43.2		14.8	
	45.3		15.4
47.4		16	
	48.3		16.2
49.2		16.4	
.	.	.	.
.	.	.	.
59.4		18.8	
	59.7		19
60.0		19.2	
.	.	.	.
.	.	.	.
108		22.4	
	109.05		23
110.1		23.6	

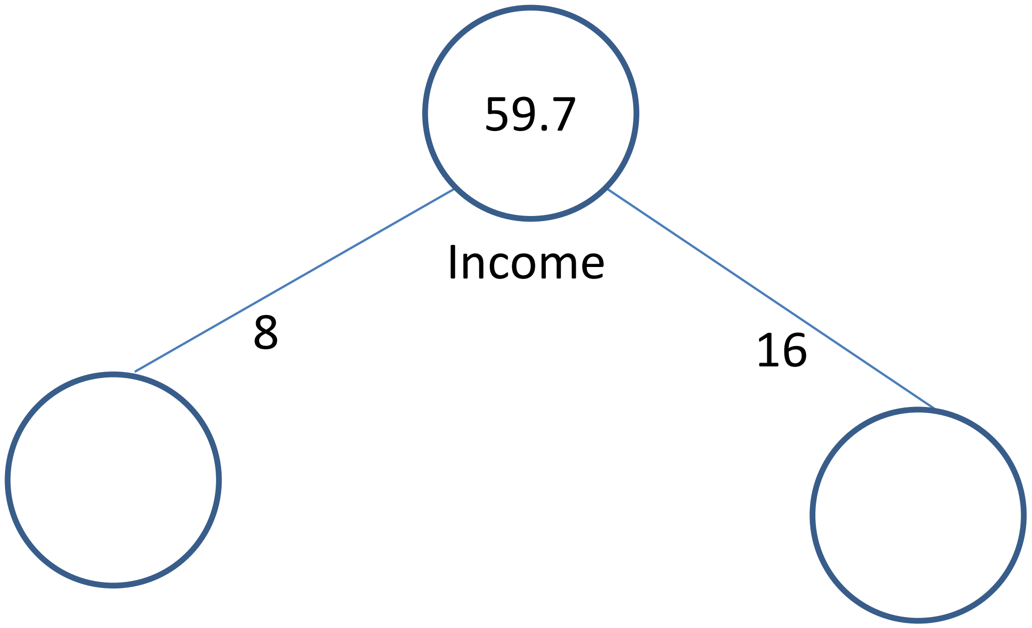
Scatter Plot



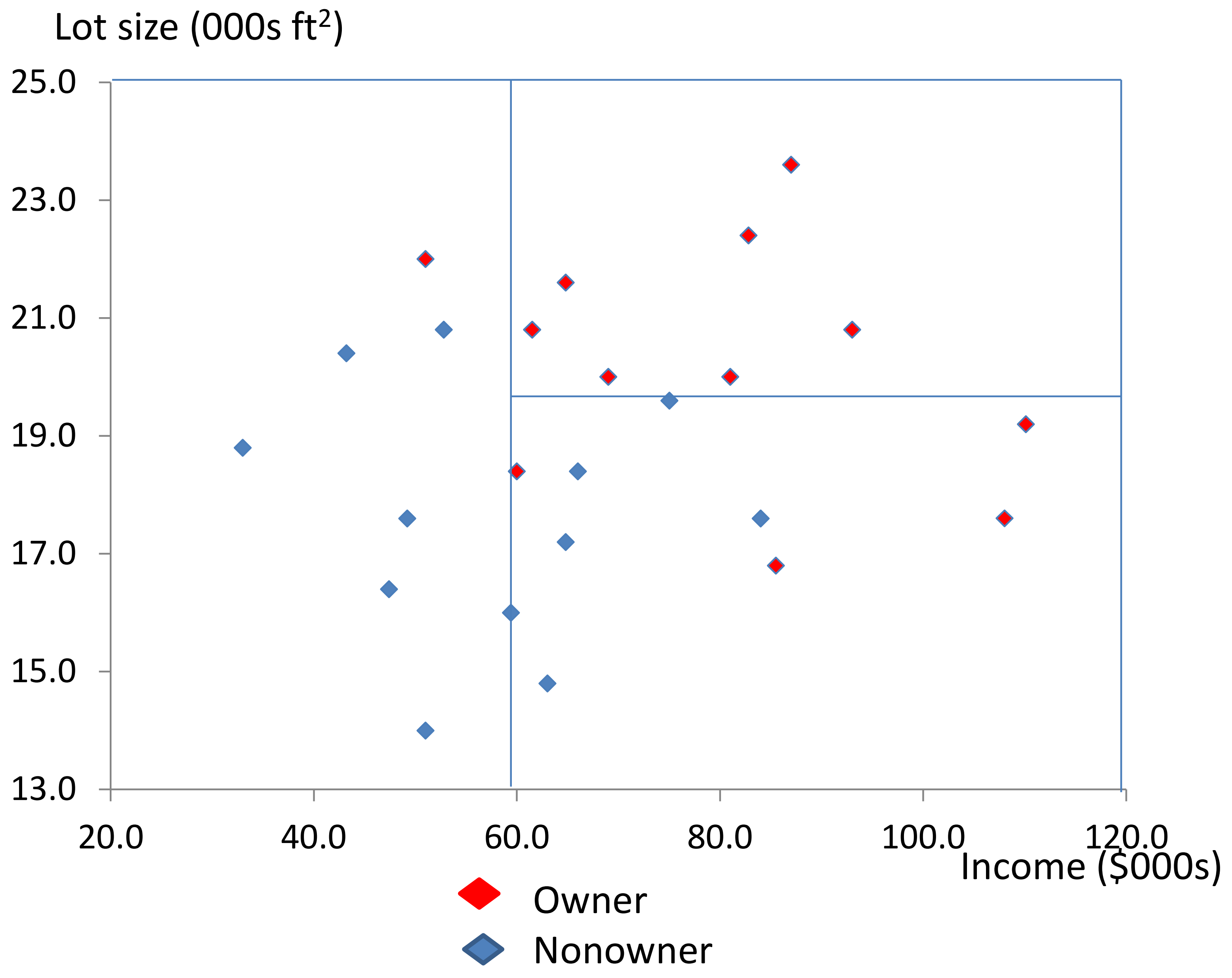
First stage of Recursive Partitioning



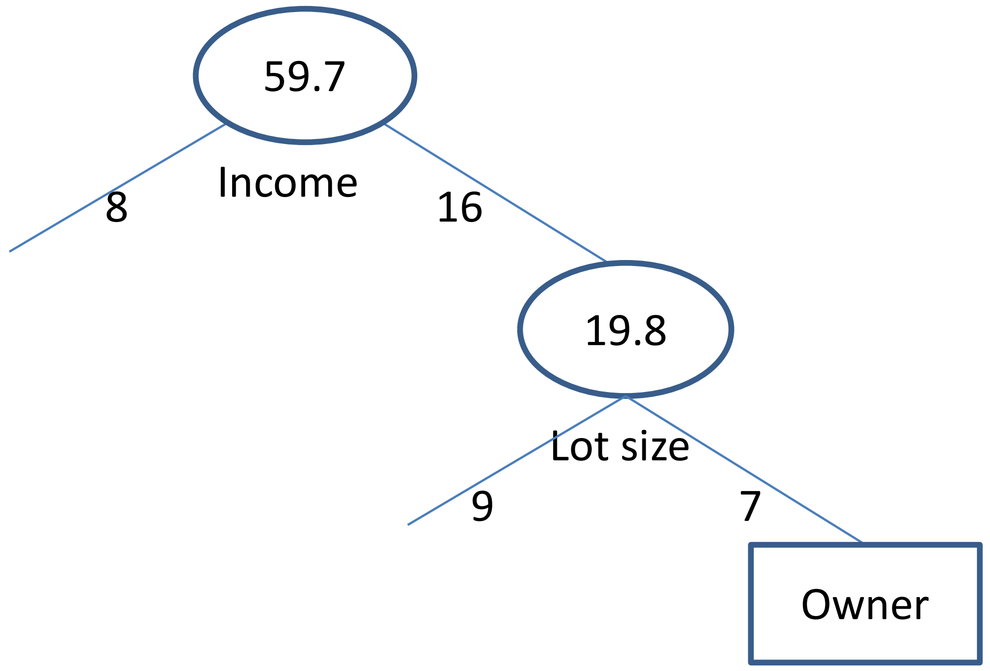
First Split



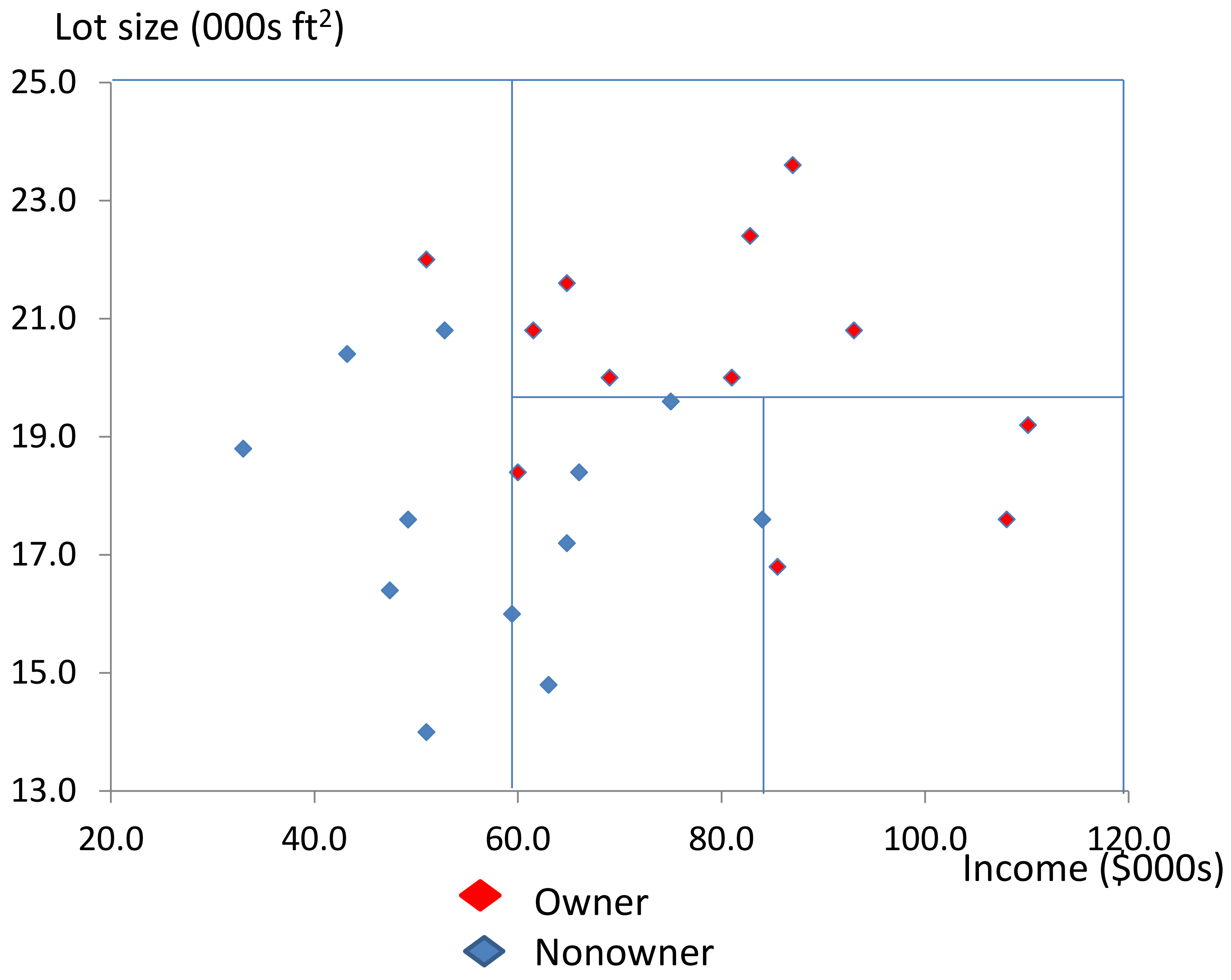
Second stage of Recursive Partitioning



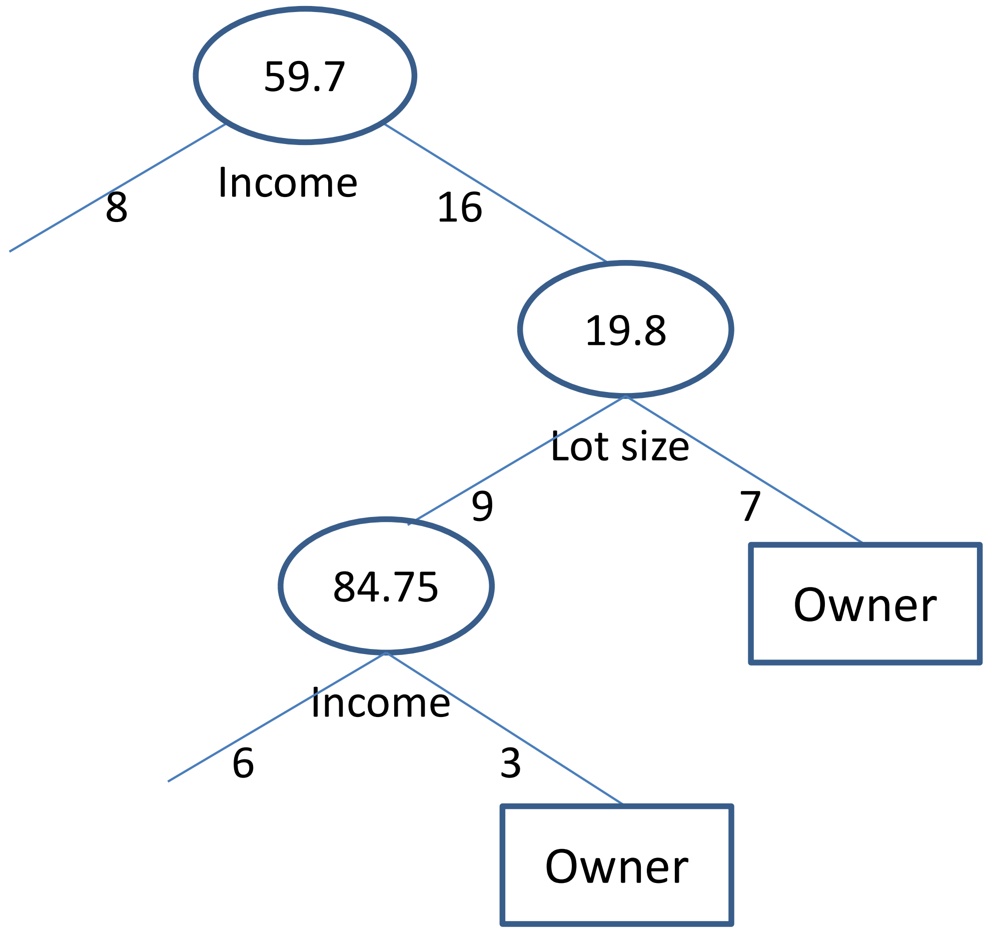
Second Split



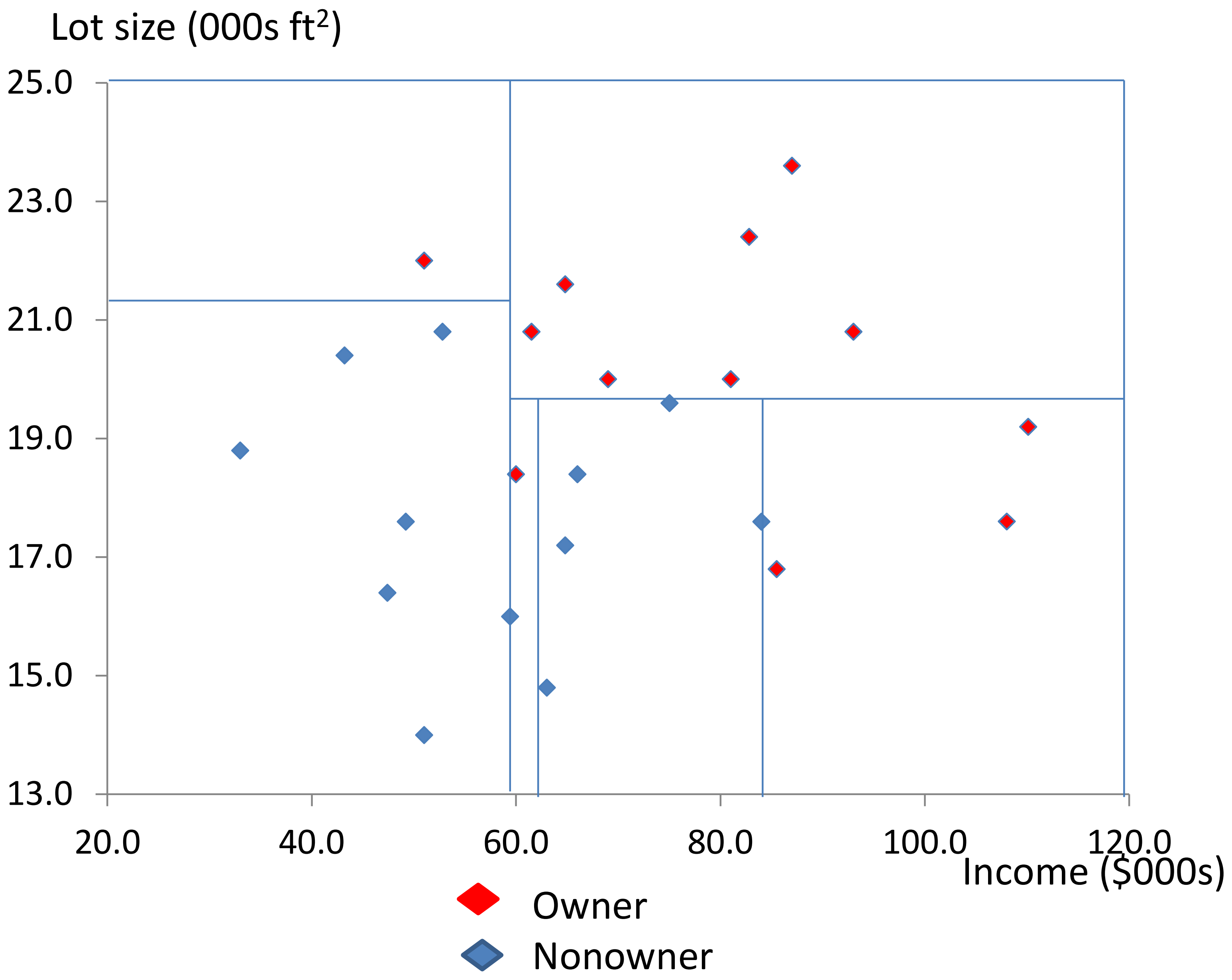
Third stage of Recursive Partitioning



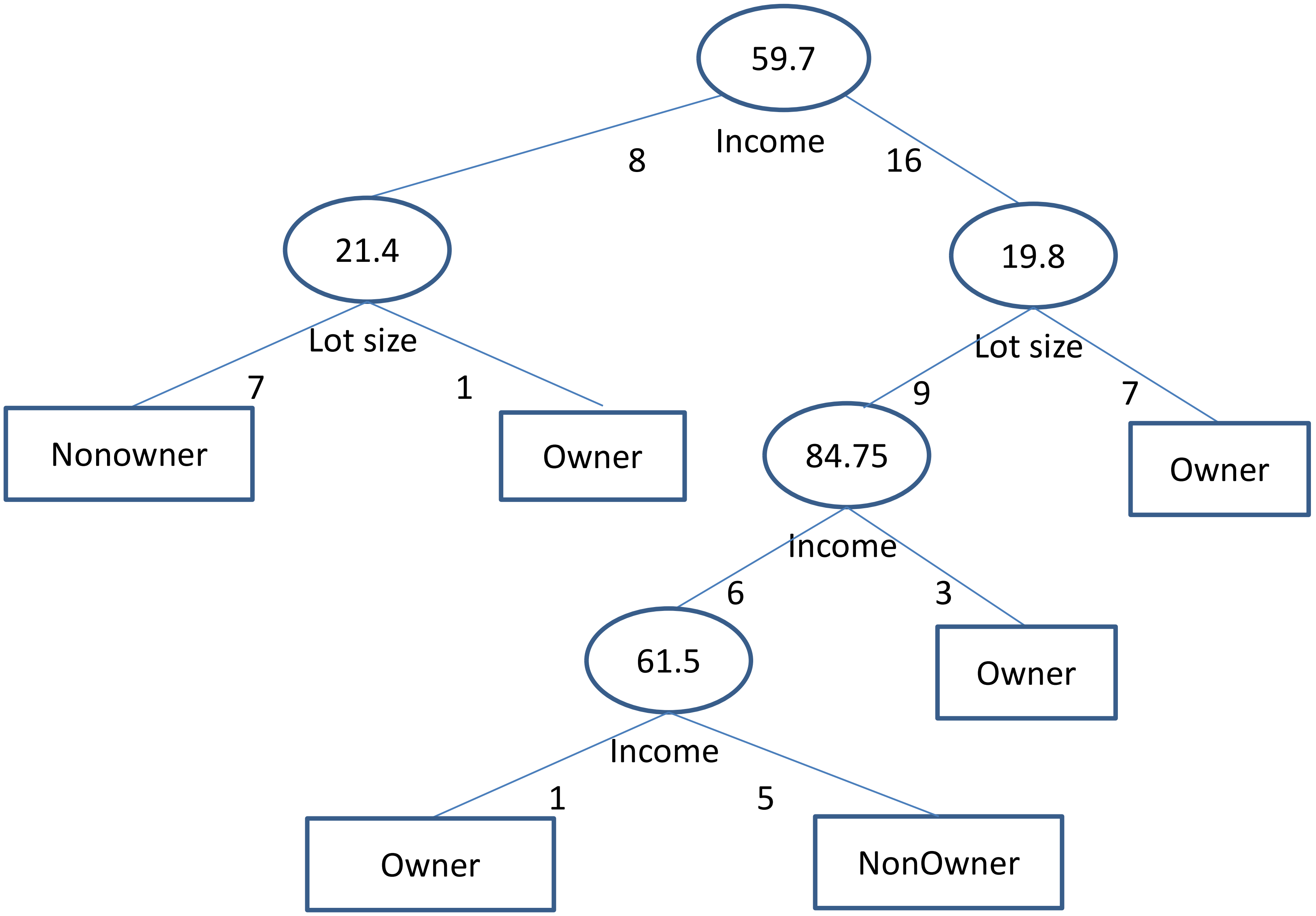
Third Split



Fourth stage and Fifth stage of Recursive Partitioning



Full Grown Tree



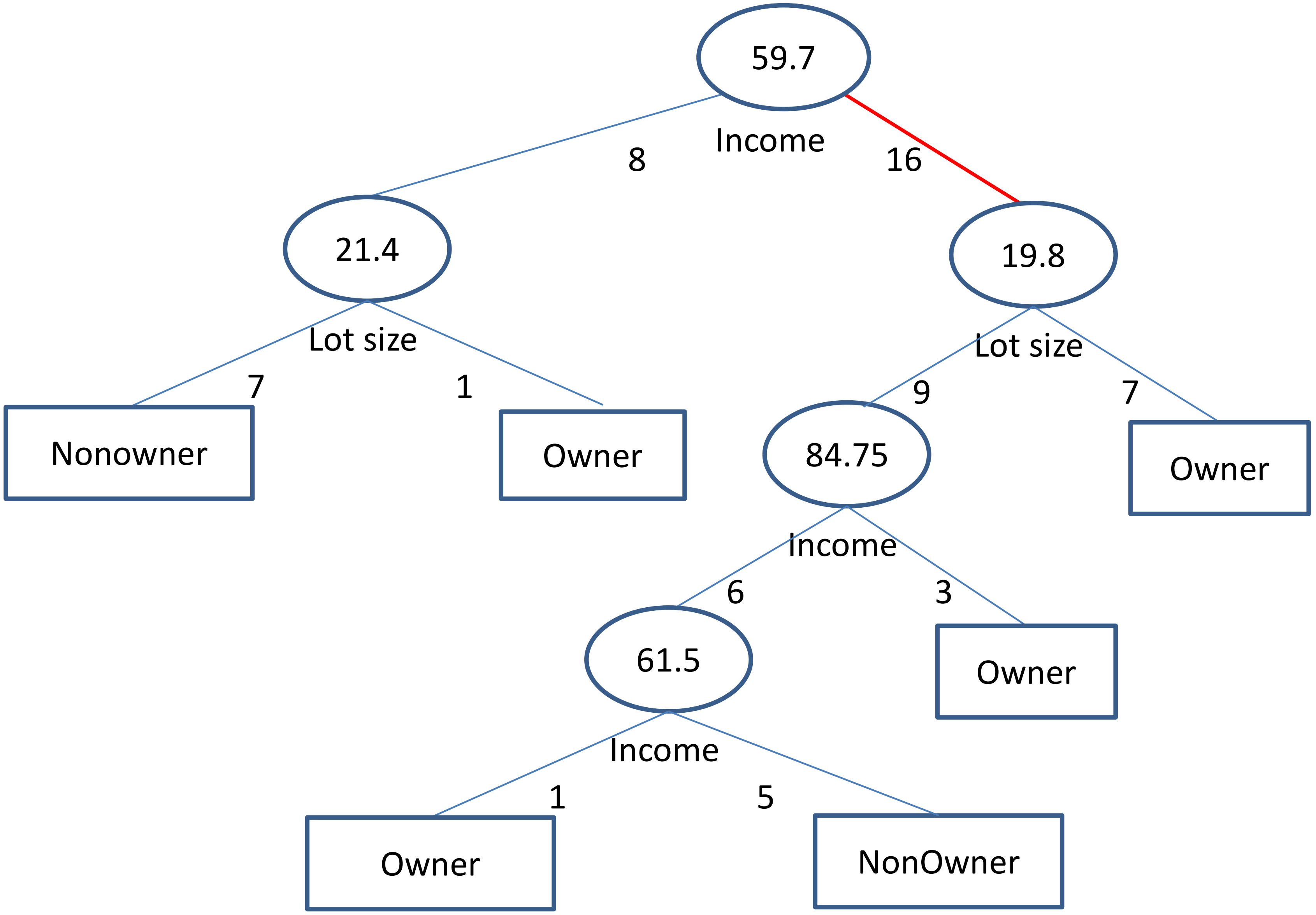
Classifying a New Observation

- If the new observation has a lot size = 18,000 ft2 and income = $85,000, what class should we assign for this new observation?

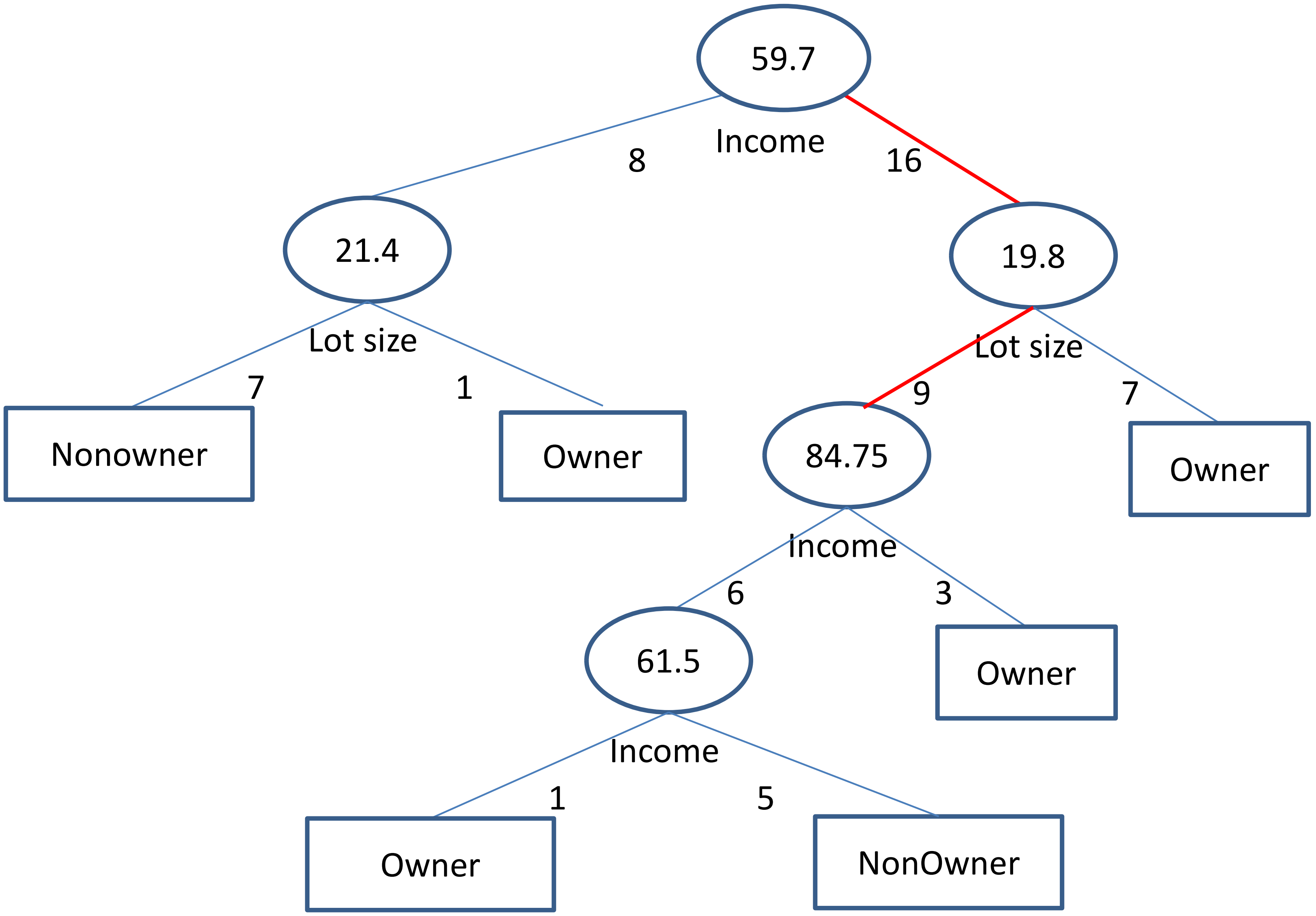
- If the new observation has a lot size = 19,500 ft2 and income = $60,000, what class should we assign for this new observation?

- If the new observation has a lot size = 21,500 ft2 and income = $55,000, what class should we assign for this new observation?

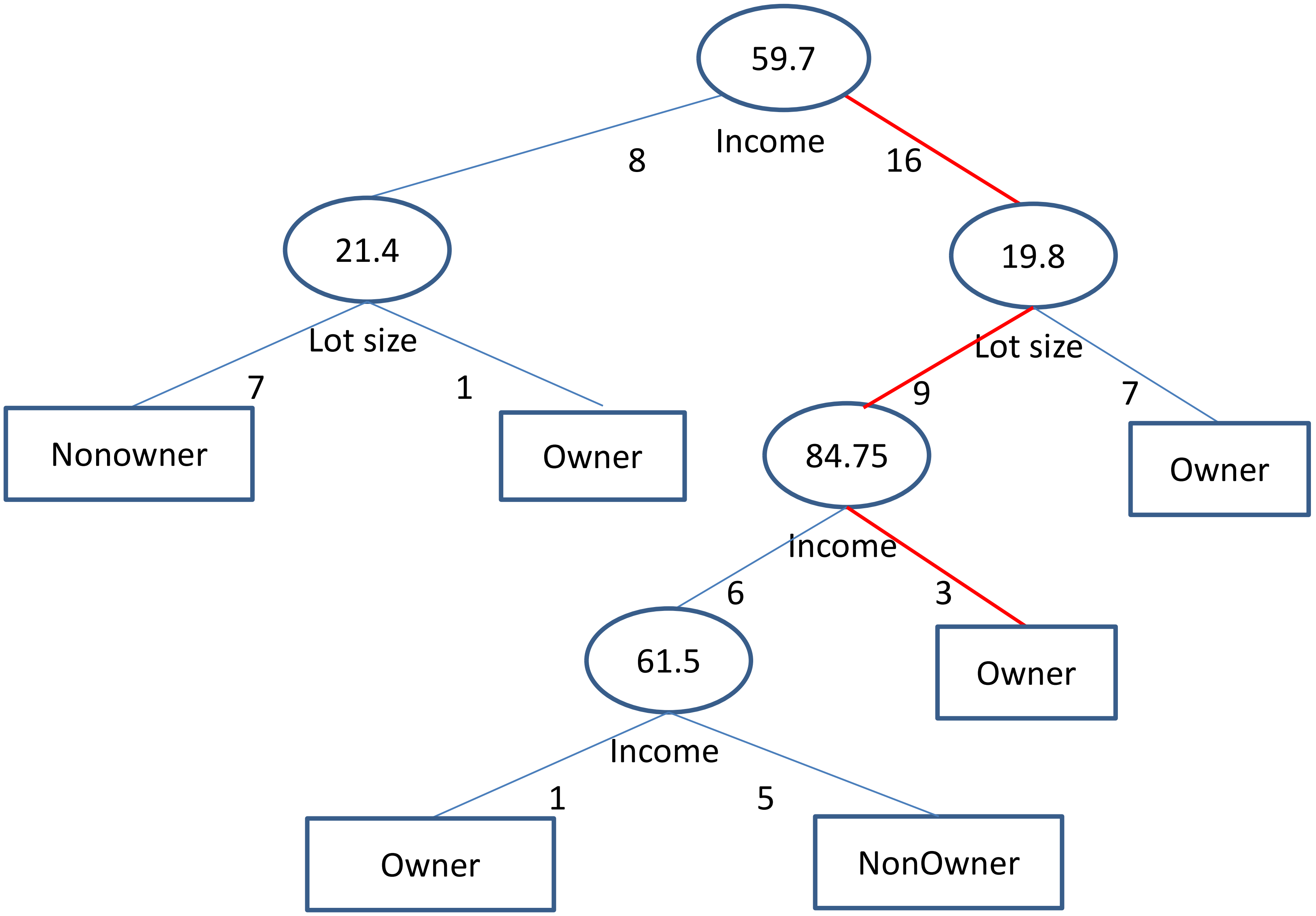
Full Grown Tree



Full Grown Tree



Full Grown Tree



Evaluation the Performance of a Classification Tree

- We use cross validation method to assess the accuracy of the tree.

- In general, the training set is used to grow the tree, and the validation set is used to assess its performance.

Avoiding Overfitting

- Overfitting will lead to poor performance on the new data.

- The overfitting at the high levels of the tree is likely to happen since these splits are based on very small number of observations ➔ class difference is likely to be attributed to noise rather than predictor information.

Avoiding Overfitting

Two ways to avoid overfitting

1. Stop tree growth by using stopping criterion, e.g.

(i) the number of splits

(ii) minimum number of observations in a node

(iii) minimum reduction in impurity.

Avoiding Overfitting

Two ways to avoid overfitting

2. Pruning the tree

- An alternative solution that has proven to be more successful than stopping tree growth is pruning the full-grown tree.

- This is the basis of method such as CART which is implemented in multiple data mining software packages.

Pruning the tree

The idea behind pruning is to recognize that a very large tree is likely to overfitting the training data and that the weakest branches, which hardly reduce the error rate, should be removed.

Pruning the tree

Step 1: Find the best subtree of each size (1,2,3, …).

Step 2: Pick the tree in the sequence that gives the smallest misclassification error in the validation set.

Classification Tree from R

There are many packages to build a tree, such as

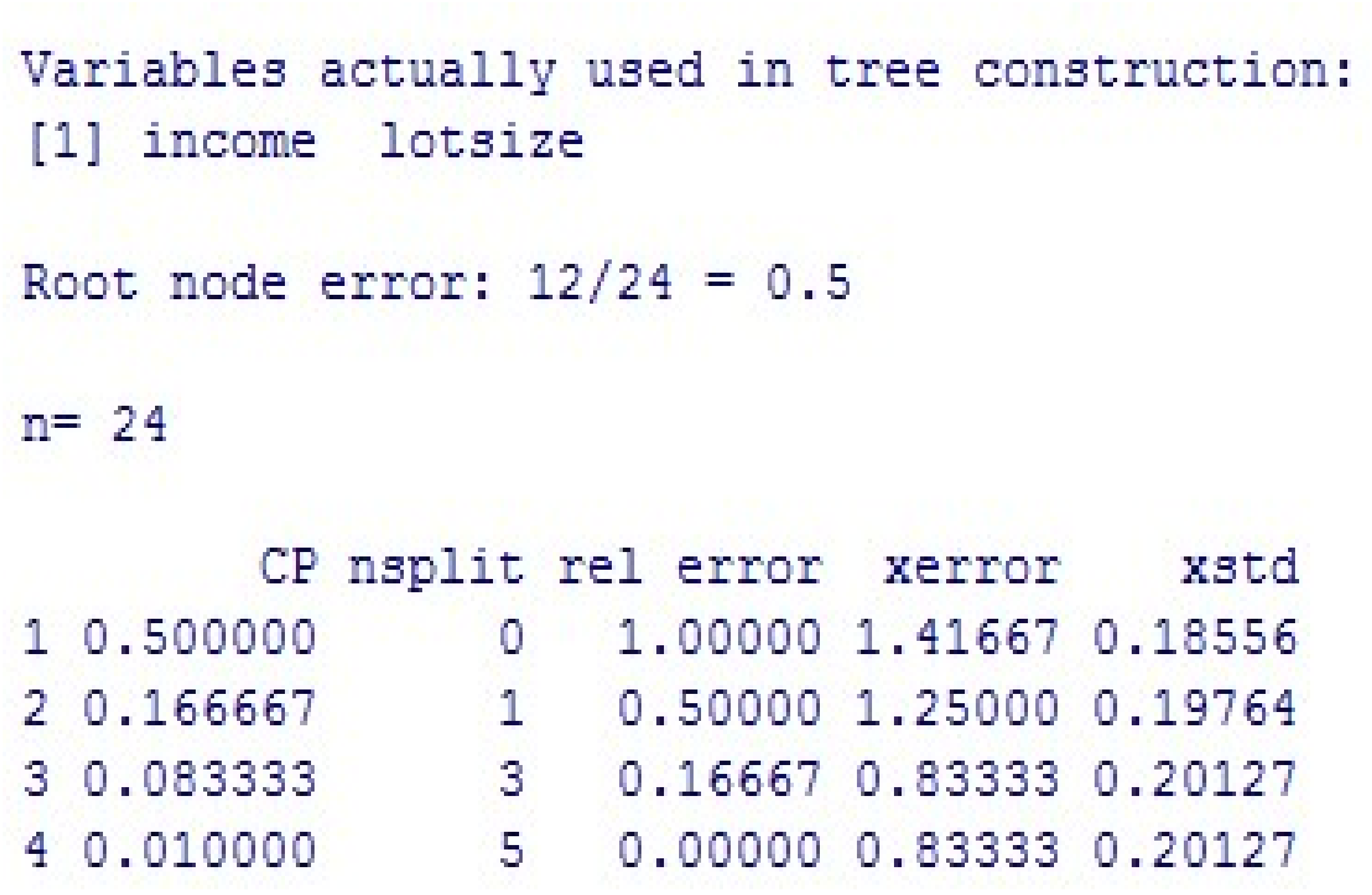
- rpart
- tree
- party
- evtree.

Note: Different packages may generate different trees.

rpart

```
## Classification Tree with rpart (mower)
# Classification Tree with rpart
> install.packages("rpart")
> library(rpart)

# read data
> mower24 <- read.csv("C:/MA 299/R/mower24.csv")

# grow tree (default minsplit = 20 and xval = 10 fold cross validation)
> fit <- rpart(ownership ~ income + lotsize,
   method="class", data=mower24, control=rpart.control(minsplit = 1, xval = 10))

# show result
> printcp(fit) # display the results
```

rpart

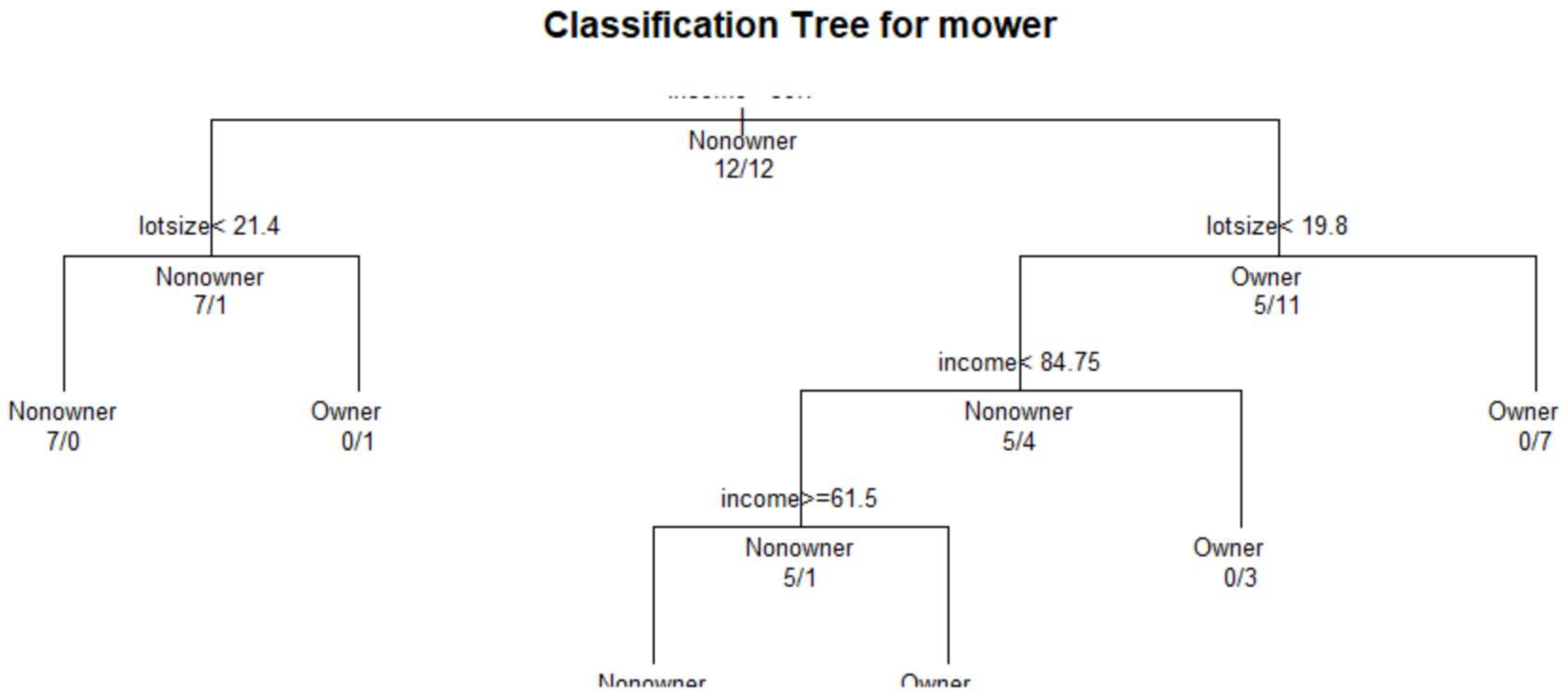


```
Variables actually used in tree construction:
[1] income  lotsize

Root node error: 12/24 = 0.5

n= 24

        CP nsplit rel error  xerror    xstd
1 0.500000      0   1.00000 1.41667 0.18556
2 0.166667      1   0.50000 1.25000 0.19764
3 0.083333      3   0.16667 0.83333 0.20127
4 0.010000      5   0.00000 0.83333 0.20127
```

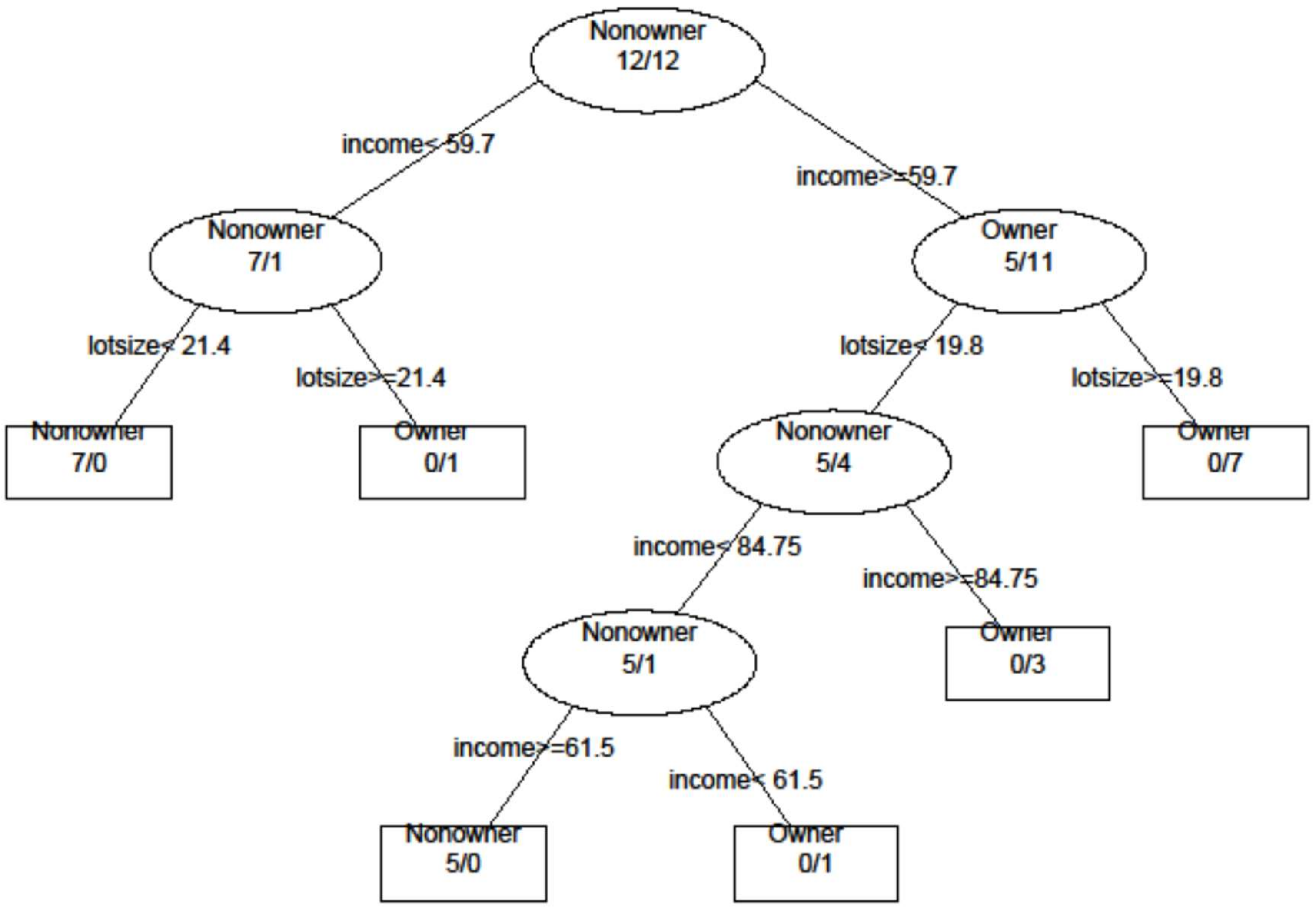
rpart

```
# plot tree
> plot(fit, uniform=TRUE, main="Classification Tree for mower")
> text(fit, use.n=TRUE, all=TRUE, cex=.8)

# create a postscript plot of tree
# You need to install postscript viewer.
# Go to http://download.cnet.com/Postscript-Viewer/3000-2094_4-10845650.html.
> post(fit, file = "c:/MA 299/R/mowertree.ps", title = "Classification Tree for Mower")
```

Plot from R


Classification Tree for mower
Nonowner
12/12
lotsize< 21.4
Nonowner
7/1
lotsize< 19.8
Owner
5/11
Nonowner
7/0
Owner
0/1
income< 84.75
Nonowner
5/4
Owner
0/7
income>=61.5
Nonowner
5/1
Owner
0/3
Nonowner
Owner

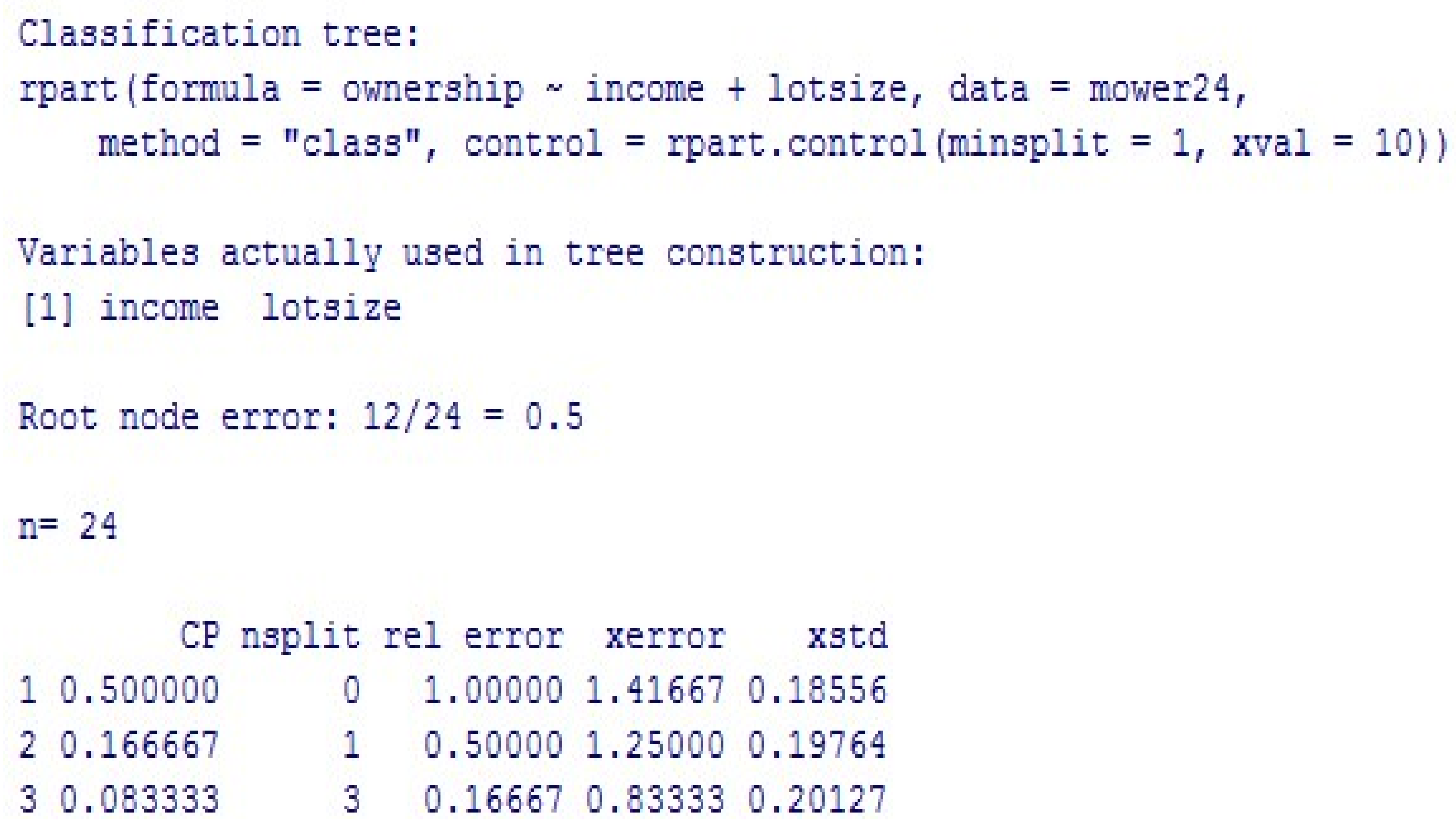
Classification Tree for Mower



rpart (prune tree)

```
# Prune the tree
> pfit<- prune(fit, cp=   fit$cptable[which.min(fit$cptable[,"xerror"]),"CP"])
# show result
> printcp(pfit) # display the results
```

rpart (prune tree)

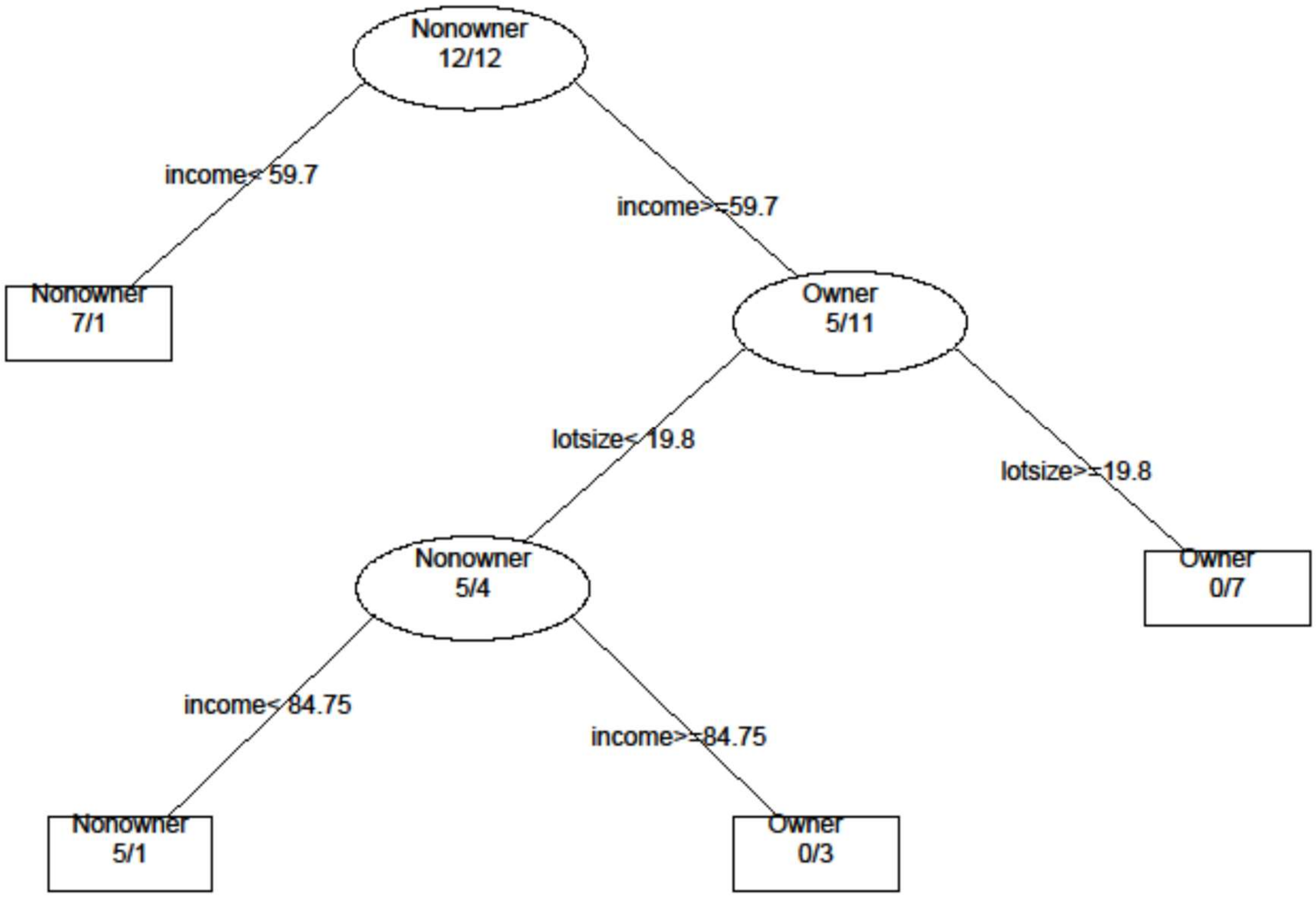


```
Classification tree:
rpart(formula = ownership ~ income + lotsize, data = mower24,
    method = "class", control = rpart.control(minsplit = 1, xval = 10))

Variables actually used in tree construction:
[1] income  lotsize

Root node error: 12/24 = 0.5

n= 24

        CP nsplit rel error  xerror    xstd
1 0.500000      0   1.00000 1.41667 0.18556
2 0.166667      1   0.50000 1.25000 0.19764
3 0.083333      3   0.16667 0.83333 0.20127
```

rpart

```
# plot the pruned tree
> plot(pfit, uniform=TRUE, main="Pruned Classification Tree for Mower")
> text(pfit, use.n=TRUE, all=TRUE, cex=.8)
# create a postscript plot of tree
> post(pfit, file = "c:/MA 299/R/prunemowertree.ps",
  title = "Pruned Classification Tree for Mower")
```

Pruned Classification Tree for Mower

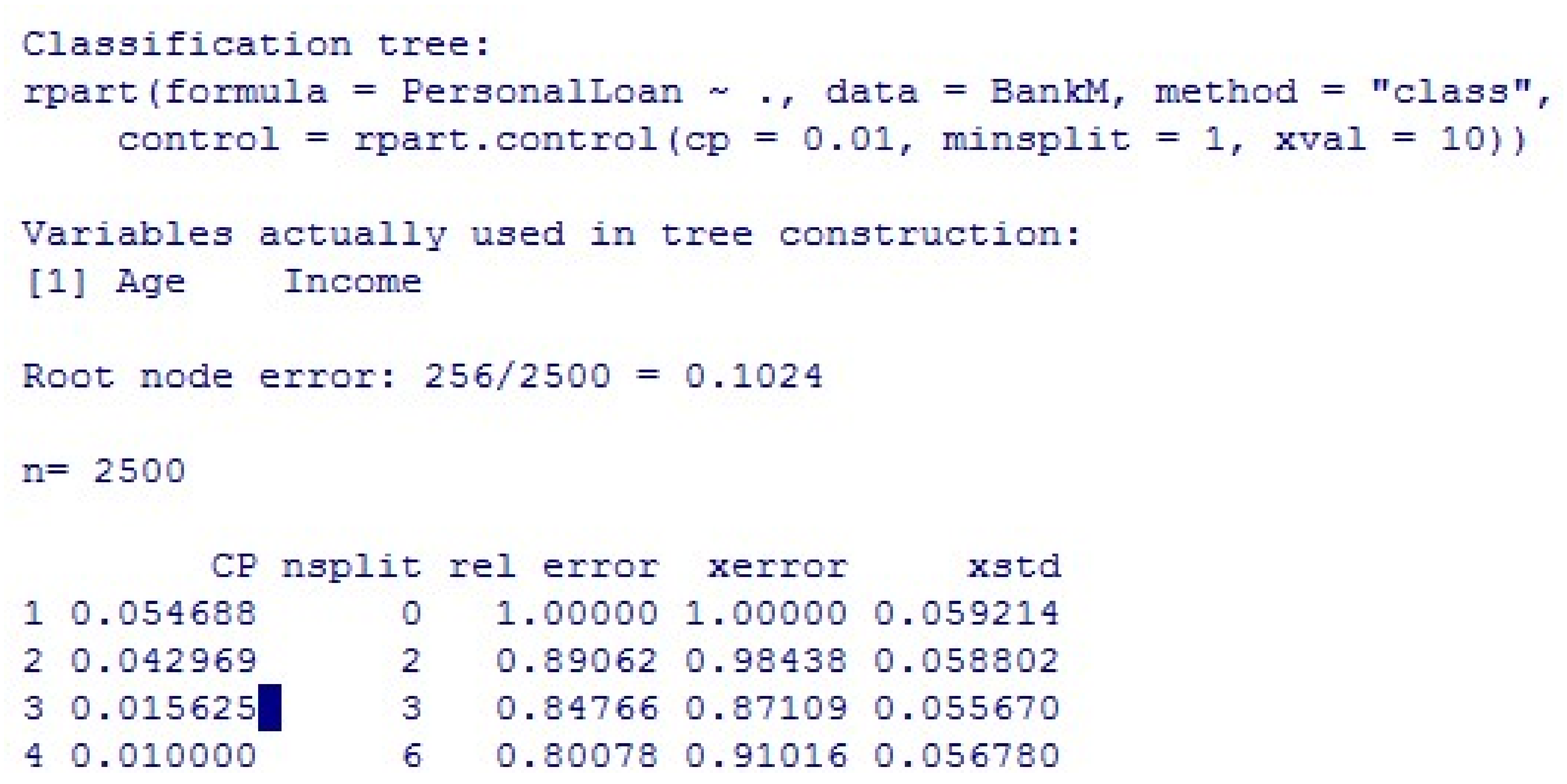


Example: UniversalBankM

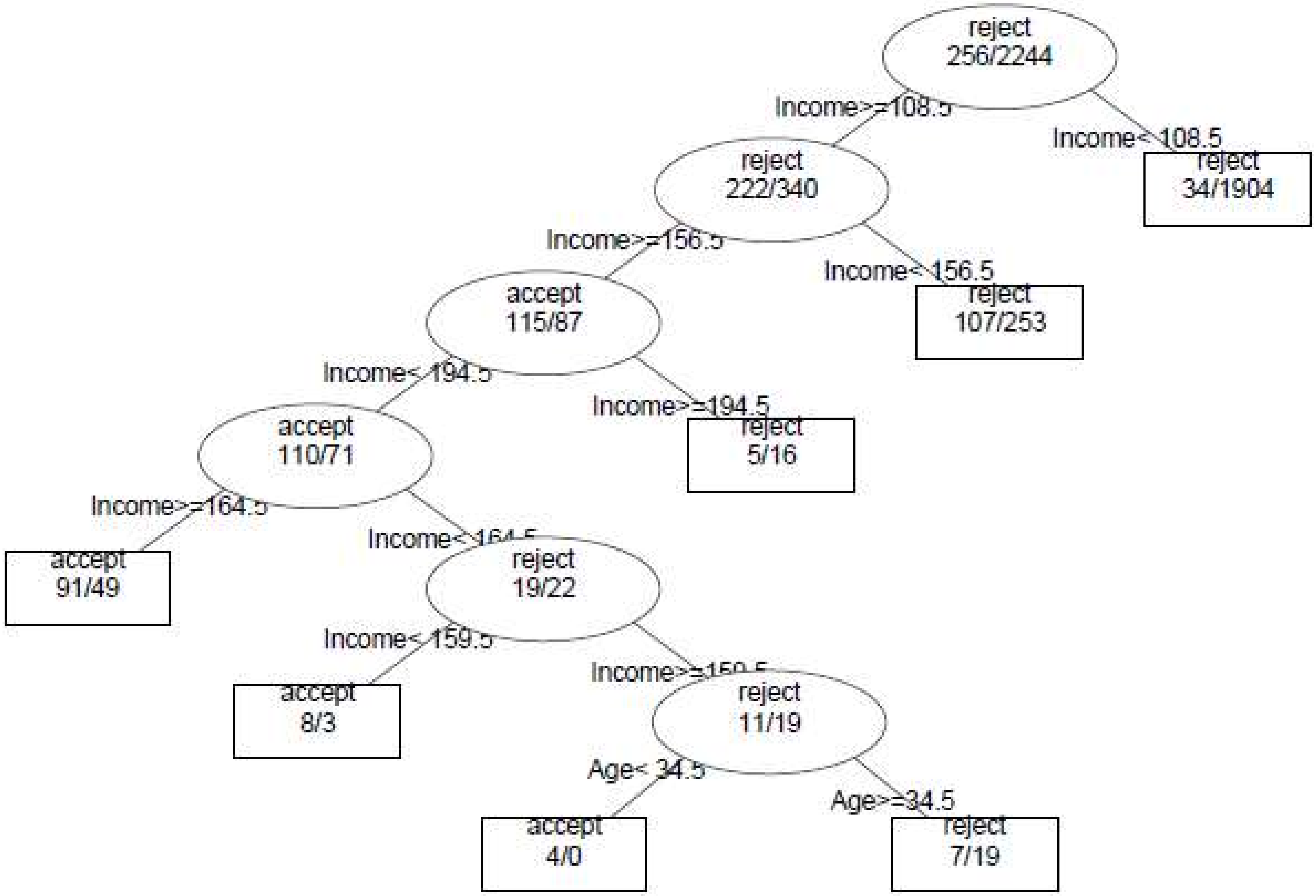
The details of 5 predictors and the response (PersonalLoan) are as follows.

Online	Whether or not the customer is an active user of online banking services? Online = 1 (yes), Online = 0 (no)
CC	Does the customer hold a credit card issued by the bank? CC = 1 (yes), CC = 0 (no)
Income	Annual income of the customer ($000)
Experience	# years of professional experience
Age	Customer’s age in years
PersonalLoan	Did this customer accept the personal loan offered in the last campaign? There are two classes: accept (1) and reject (0).

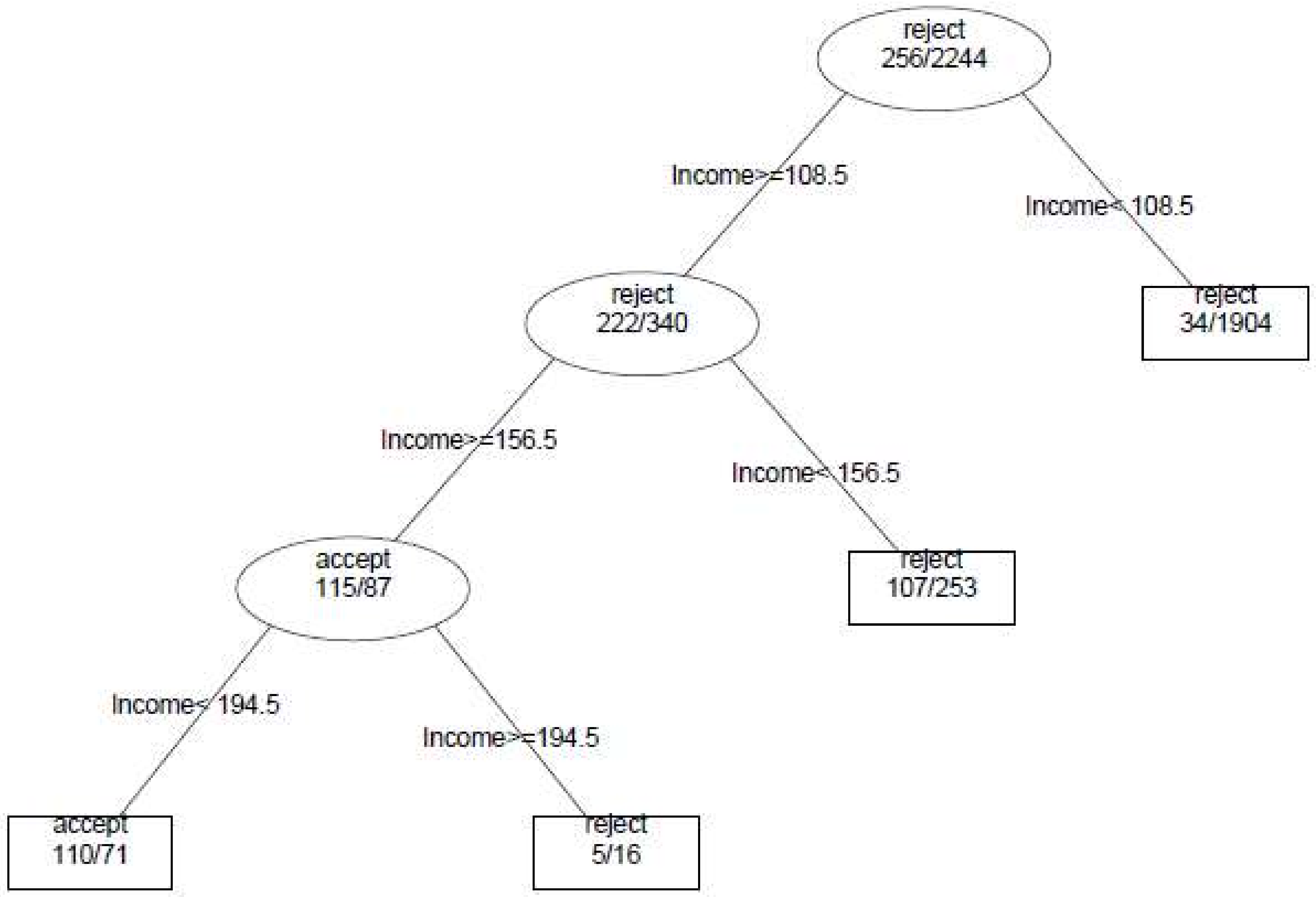
```
library(rpart)

# read data
BankM <- read.csv("C:/MA 299/R/UniversalBankM.csv")

# grow tree (default minsplit = 1 and xval = 10 fold cross validation)
fit <- rpart(PersonalLoan ~ .,
  method="class", data=BankM, control=rpart.control(minsplit = 1, xval = 10))

# show result
printcp(fit) # display the results

# plot tree
plot(fit, uniform=TRUE,
  main="Classification Tree for UniversalankM")
text(fit, use.n=TRUE, all=TRUE, cex=.8)

# create a postscript plot of tree
# You need to install postscript viewer.
# Go to http://download.cnet.com/Postscript-Viewer/3000-2094_4-10845650.html.
post(fit, file = "c:/MA 299/R/UniverslBankM.ps",
  title = "Classification Tree for UniversalBankM")
```



```
Classification tree:
rpart(formula = PersonalLoan ~ ., data = BankM, method = "class",
    control = rpart.control(cp = 0.01, minsplit = 1, xval = 10))

Variables actually used in tree construction:
[1] Age    Income

Root node error: 256/2500 = 0.1024

n= 2500

        CP nsplit rel error  xerror     xstd
1 0.054688      0   1.00000 1.00000 0.059214
2 0.042969      2   0.89062 0.98438 0.058802
3 0.015625      3   0.84766 0.87109 0.055670
4 0.010000      6   0.80078 0.91016 0.056780
```


reject
256/2244
Income>=108.5
Income< 108.5
reject
222/340
reject
34/1904
Income>=156.5
Income< 156.5
accept
115/87
reject
107/253
Income< 194.5
Income>=194.5
accept
110/71
reject
5/16
Income>=164.5
Income< 164.5
accept
91/49
reject
19/22
Income< 159.5
Income>=159.5
accept
8/3
reject
11/19
Age< 34.5
Age>=34.5
accept
4/0
reject
7/19

```
## Prune Tree
# prune the tree
pfit<- prune(fit, cp=   fit$cptable[which.min(fit$cptable[,"xerror"]),"CP"])
# show result
printcp(pfit) # display the results

# plot the pruned tree
plot(pfit, uniform=TRUE,
  main="Pruned Classification Tree for pruneUniversalBankM")
text(pfit, use.n=TRUE, all=TRUE, cex=.8)

# create a postscript plot of tree
post(pfit, file = "c:/MA 299/R/pruneUniversalBankM.ps",
  title = "Pruned Classification Tree for UniverslaBankM")
```


reject
256/2244
Income>=108.5
Income< 108.5
reject
222/340
reject
34/1904
Income>=156.5
Income< 156.5
accept
115/87
reject
107/253
Income< 194.5
Income>=194.5
accept
110/71
reject
5/16

Classification Trees for More Than Two Classes

- Classification trees can be used with an outcome that has more than two classes.

- The tree method can also be used for numerical response variables ➔ regression tree.

  - Both the principle and the procedure are the same.

  - There are three details that are different from the classification tree.

    (i) Prediction

    (ii) Impurity measures

    (iii) Evaluating performance

Classification Trees for More Than Two Classes

- Classification trees can be used with an outcome that has more than two classes.

- The tree method can also be used for numerical response variables ➔ regression trees.

Regression Trees

- Both the principle and the procedure are the same.
- There are three details that are different from the classification tree.

    (i) Prediction

    (ii) Impurity measures

    (iii) Evaluating performance

Differences between classification trees and regression trees

Prediction

- The value of the leaf node is the predicted value which is determined by the average of the data in that leaf.

Measuring Impurity

- The typical impurity measure is the squared errors.

Evaluating Performance

- RMSE (root mean squared error)

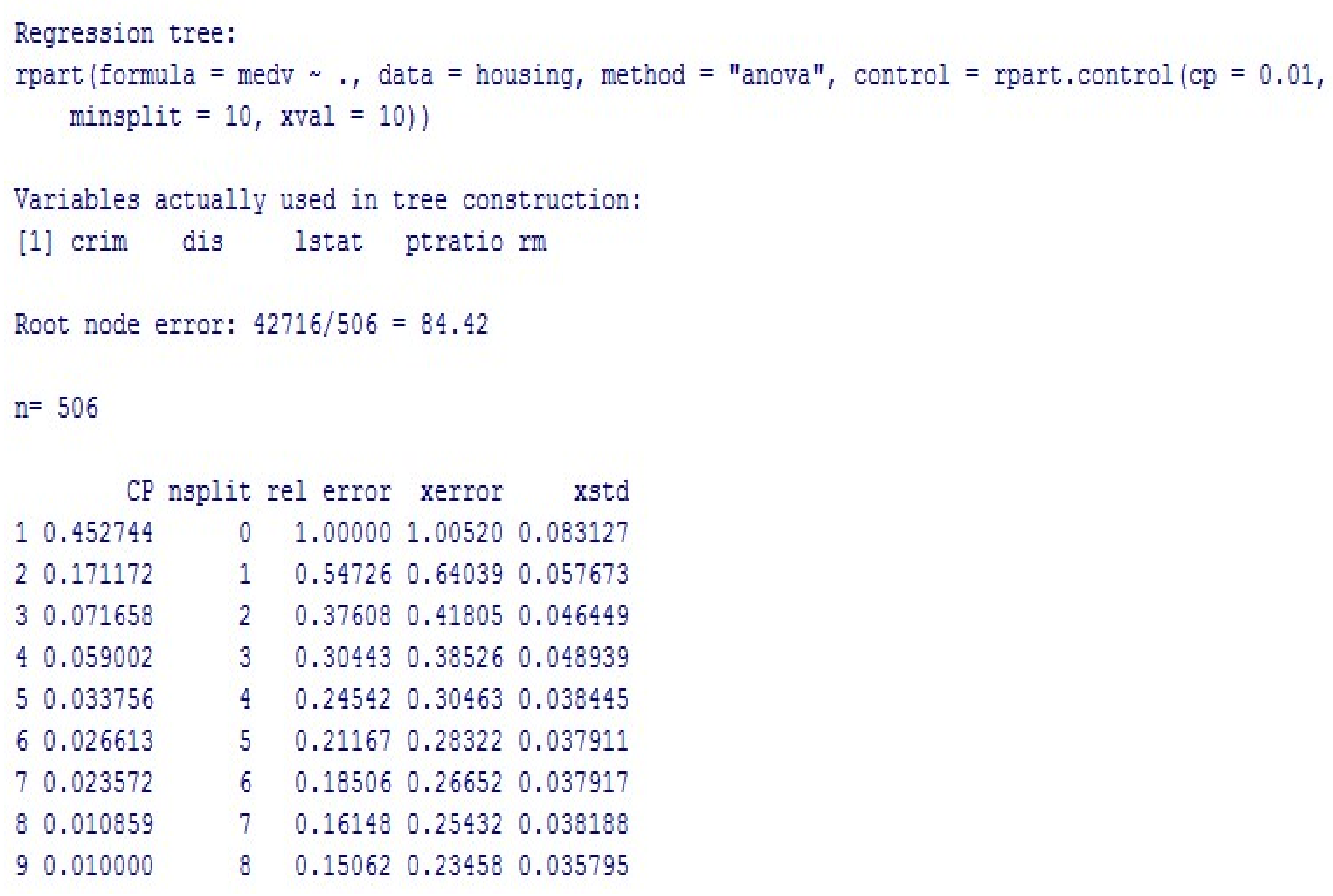
Example: Boston.housing.csv

Predicting Boston House Prices. The dataset contains 14 predictors, and the predictor is the median house price (MEDV).

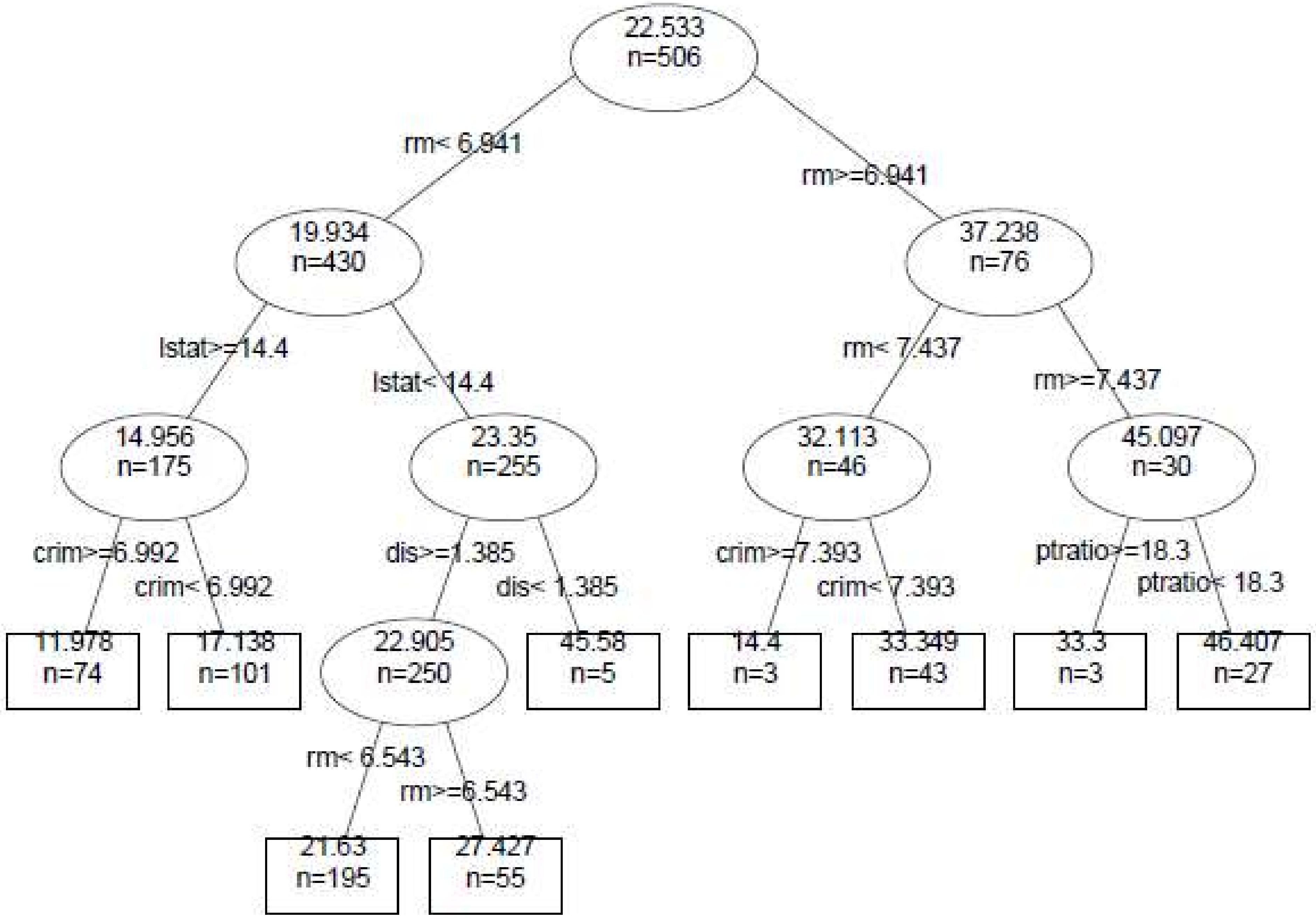
Variable Information:

1. CRIM: per capita crime rate by town
2. ZN: proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS: proportion of non-retail business acres per town
4. CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
5. NOX: nitric oxides concentration (parts per 10 million)
6. RM: average number of rooms per dwelling
7. AGE: proportion of owner-occupied units built prior to 1940
8. DIS: weighted distances to five Boston employment centres
9. RAD: index of accessibility to radial highways
10. TAX: full-value property-tax rate per $10,000
11. PTRATIO: pupil-teacher ratio by town
12. B: 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
13. LSTAT: % lower status of the population
14. MEDV: Median value of owner-occupied homes in $1000's

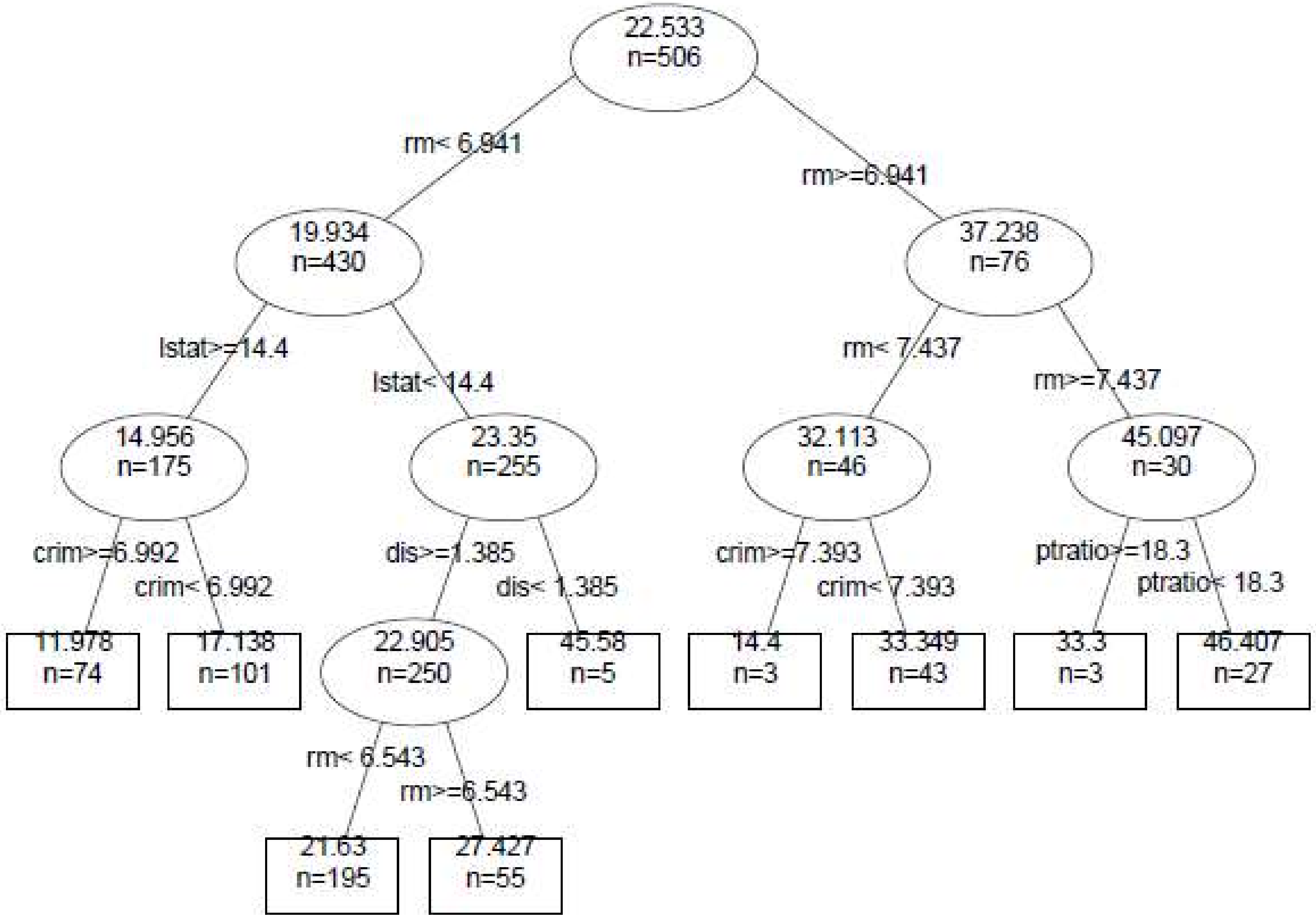
```
library(rpart)

# read data
housing <- read.csv("C:/MA 299/R/housing.csv")

# grow tree (default minsplit = 0 and xval = 10 fold cross validation)
fit <- rpart(medv ~ .,
  method="anova", data=housing, control=rpart.control(cp = 0.01, minsplit = 10,
xval = 10))

# show result
printcp(fit) # display the results

# plot tree
plot(fit, uniform=TRUE,
  main="Regression Tree for Housing")
text(fit, use.n=TRUE, all=TRUE, cex=.8)

# create a postscript plot of tree
post(fit, file = "c:/MA 299/R/housing.ps",
  title = "Regression Tree for Housing")
```



```
Regression tree:
rpart(formula = medv ~ ., data = housing, method = "anova", control = rpart.control(cp = 0.01,
    minsplit = 10, xval = 10))

Variables actually used in tree construction:
[1] crim    dis     lstat   ptratio rm

Root node error: 42716/506 = 84.42

n= 506

        CP nsplit rel error  xerror     xstd
1 0.452744      0   1.00000 1.00520 0.083127
2 0.171172      1   0.54726 0.64039 0.057673
3 0.071658      2   0.37608 0.41805 0.046449
4 0.059002      3   0.30443 0.38526 0.048939
5 0.033756      4   0.24542 0.30463 0.038445
6 0.026613      5   0.21167 0.28322 0.037911
7 0.023572      6   0.18506 0.26652 0.037917
8 0.010859      7   0.16148 0.25432 0.038188
9 0.010000      8   0.15062 0.23458 0.035795
```


22.533
n=506
rm< 6.941
rm>=6.941
19.934
n=430
37.238
n=76
lstat>=14.4
lstat< 14.4
rm< 7.437
rm>=7.437
14.956
n=175
23.35
n=255
32.113
n=46
45.097
n=30
crim>=6.992
crim< 6.992
dis>=1.385
dis< 1.385
crim>=7.393
crim< 7.393
ptratio>=18.3
ptratio< 18.3
11.978
n=74
17.138
n=101
22.905
n=250
45.58
n=5
14.4
n=3
33.349
n=43
33.3
n=3
46.407
n=27
rm< 6.543
rm>=6.543
21.63
n=195
27.427
n=55

```
## Prune Tree
# prune the tree
pfit<- prune(fit, cp=fit$cptable[which.min(fit$cptable[,"xerror"]),"CP"])
# show result
printcp(pfit) # display the results

# plot the pruned tree
plot(pfit, uniform=TRUE, main="Pruned Regression Tree for Housing")
text(pfit, use.n=TRUE, all=TRUE, cex=.8)
# create a postscript plot of tree
post(pfit, file = "c:/MA 299/R/pruneHousing.ps",
  title = "Pruned Classification Tree for Housing")
```



22.533
n=506
rm< 6.941
rm>=6.941
19.934
n=430
37.238
n=76
lstat>=14.4
lstat< 14.4
rm< 7.437
rm>=7.437
14.956
n=175
23.35
n=255
32.113
n=46
45.097
n=30
crim>=6.992
crim< 6.992
dis>=1.385
dis< 1.385
crim>=7.393
crim< 7.393
ptratio>=18.3
ptratio< 18.3
11.978
n=74
17.138
n=101
22.905
n=250
45.58
n=5
14.4
n=3
33.349
n=43
33.3
n=3
46.407
n=27
rm< 6.543
rm>=6.543
21.63
n=195
27.427
n=55

Advantages of Trees

- Trees generate transparent rules.
- Trees are nonlinear and non-parametric ➔ allow a wide range of relationships between the predictors and response.
- Variable selection is automatic since it is a part of the split selection.
- Trees are robust for outliers.
- They can handle missing values without having to impute values or delete observations.

Disadvantages of Trees

- Trees are sensitive to changes in the data, even a slight can cause very different splits.
- The splits are done on single predictors rather than on combinations of predictors ➔ miss interaction effects among predictors.
- They require a large dataset to construct a good classifier.
- Trees can be relatively expensive (from a computational aspect) to grow and to prune.

Extension

<u>Random Forest</u> is an extension to classification trees to improve predictive ability.

The basic idea is to create multiple classification trees from the data and combine their output to obtain better classifier.

Random Forest

The basic idea of random forest is to:

1. Draw multiple random samples, with replacement, from the data.
2. Using a random subset of predictors at each stage, fit a classification tree to each sample (and thus obtain a "forest").
3. Combine the predictions/classifications from the individual trees to combine improved predictions. Use voting for classification and averaging for prediction.

Example: UniversalBankRF

- The bank’s dataset includes data on 5000 customers. The data include customer demographic information (age, Income, etc.), customer response to the last personal loan campaign (Personal Loan), and the customer’s relationship with the bank (mortgage, securities account, etc.).
- Among there 5000 customers, only 480 (=9.6%) accepted the personal loan that was offered to them in a previous campaign.
- The goal is to find characteristics of customers who are most likely to accept the loan offer in future mailings.

Predictors and Response

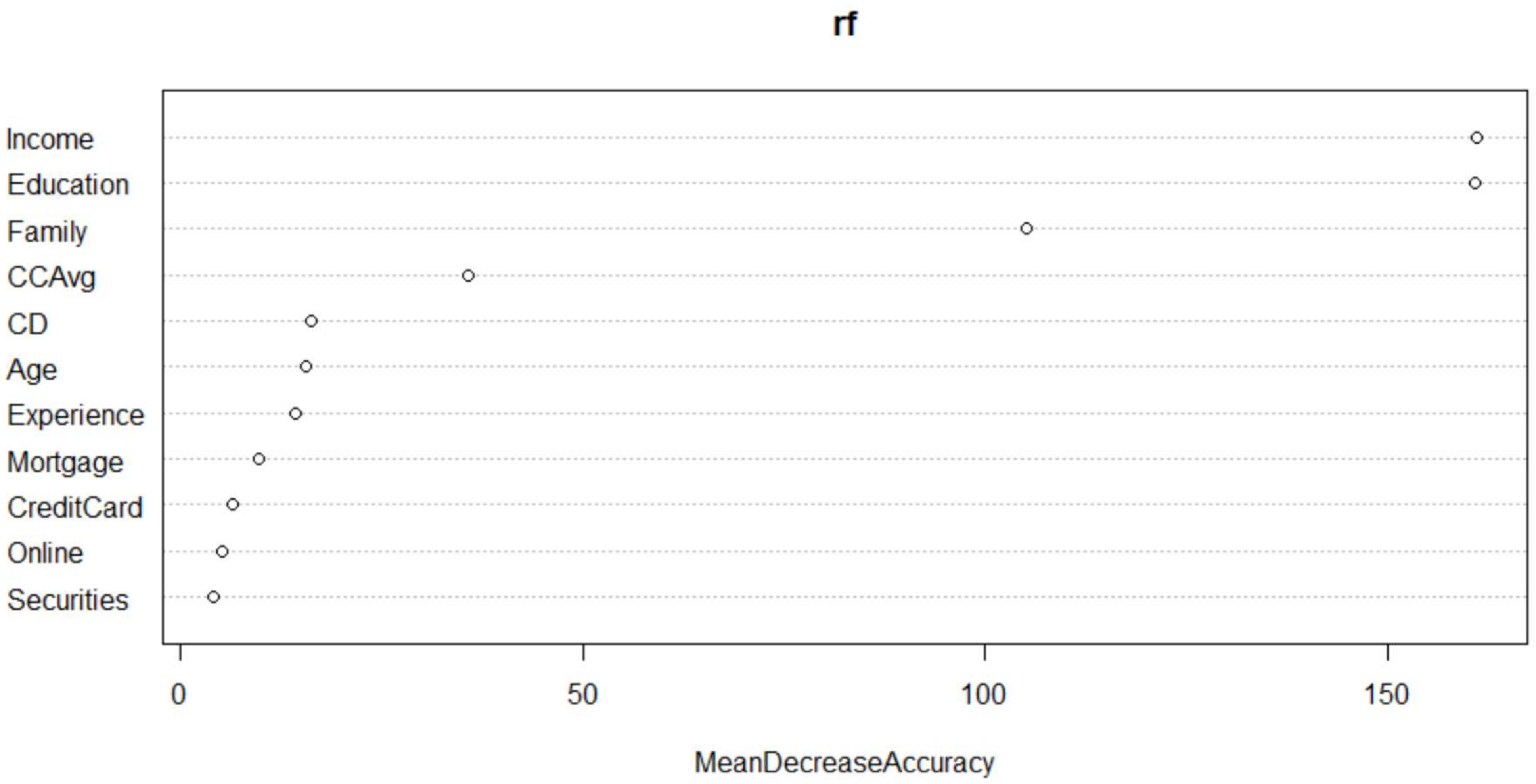
Age	Customer's age in completed years
Experience	#years of professional experience
Income	Annual income of the customer ($000)
Family	Family size of the customer
CCAvg	Avg. spending on credit cards per month ($000)
Education	Education Level. Undergrad (UG); Graduate (Gr); Advanced/Professional (Prof)
Mortgage	Value of house mortgage if any. ($000)
Securities	Does the customer have a securities account with the bank? (yes, no)
CD	Does the customer have a certificate of deposit (CD) account with the bank? (yes, no)
Online	Does the customer use internet banking facilities? (yes, no)
CreditCard	Does the customer use a credit card issued by UniversalBank? (yes, no)
Personal Loan	Did this customer accept the personal loan offered in the last campaign? (yes,no)

```
library(randomForest)
library(caret)
BankRFfull <- read.csv("C:/MA 299/R/UniversalBankRF.csv")
```

Age	Experience	Income	Family	CCAvg	Mortgage	Education	Securities	CD	Online	CreditCard	PersonalLoan
25	1	49	4	1.6	0	UG	yes	no	no	no	no
45	19	34	3	1.5	0	UG	yes	no	no	no	no
39	15	11	1	1	0	UG	no	no	no	no	no
35	9	100	1	2.7	0	Prof	no	no	no	no	no
35	8	45	4	1	0	Prof	no	no	no	yes	no
37	13	29	4	0.4	155	Prof	no	no	yes	no	no
53	27	72	2	1.5	0	Prof	no	no	yes	no	no
50	24	22	1	0.3	0	Gr	no	no	no	yes	no
35	10	81	3	0.6	104	Prof	no	no	yes	no	no
34	9	180	1	8.9	0	Gr	no	no	no	no	yes
65	39	105	4	2.4	0	Gr	no	no	no	no	no
29	5	45	3	0.1	0	Prof	no	no	yes	no	no
48	23	114	2	3.8	0	Gr	yes	no	no	no	no
59	32	40	4	2.5	0	Prof	no	no	yes	no	no
…											
30	10	50	4	0.5	0	Gr	no	no	yes	no	

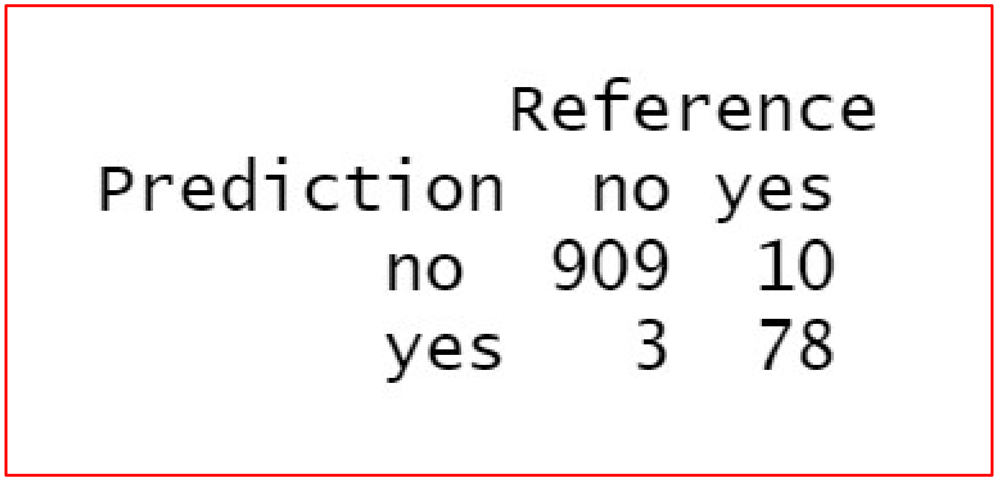
```
#Separate the data for prediction from the rest
#The data point for prediction is called "new" and the rest is called "BankRF"
new <- BankRFfull[5001,]
BankRF <- BankRFfull[1:5000,]

#Separate BankRF into the training set and the test set
idxs = sample(1:nrow(BankRF), as.integer(0.80*nrow(BankRF)))
trainBankRF = BankRF[idxs,]
testBankRF = BankRF[-idxs,]

#Apply Random Forest with the training set
rf <- randomForest(as.factor(PersonalLoan) ~ ., data = trainBankRF, ntree =
500, mtry = 4, nodesize = 5, importance = TRUE)

#Construct variable importance plot
varImpPlot(rf, type = 1)
```

Variable Importance Plot



Classification result from the test set

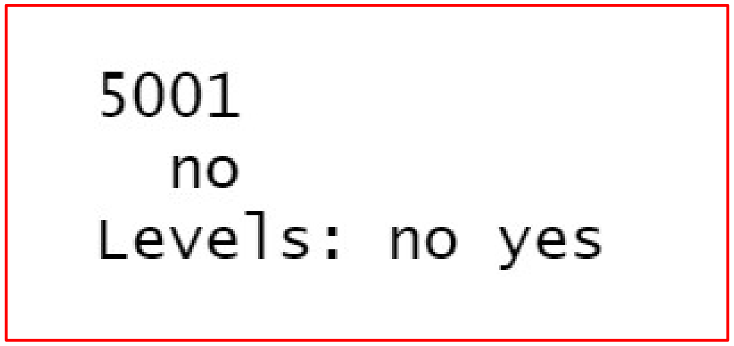
```
#Create confusion matrix for the test set
rf.pred <- predict(rf, testBankRF)
confusionMatrix(rf.pred, as.factor(testBankRF$PersonalLoan))
```



```
          Reference
Prediction  no yes
       no  909  10
       yes   3  78
```

Assign class for the new data point

```
#Predict class for the data point called "new"
result=predict(rf,new,"class")
result
```



```
5001 
  no 
Levels: no yes
```