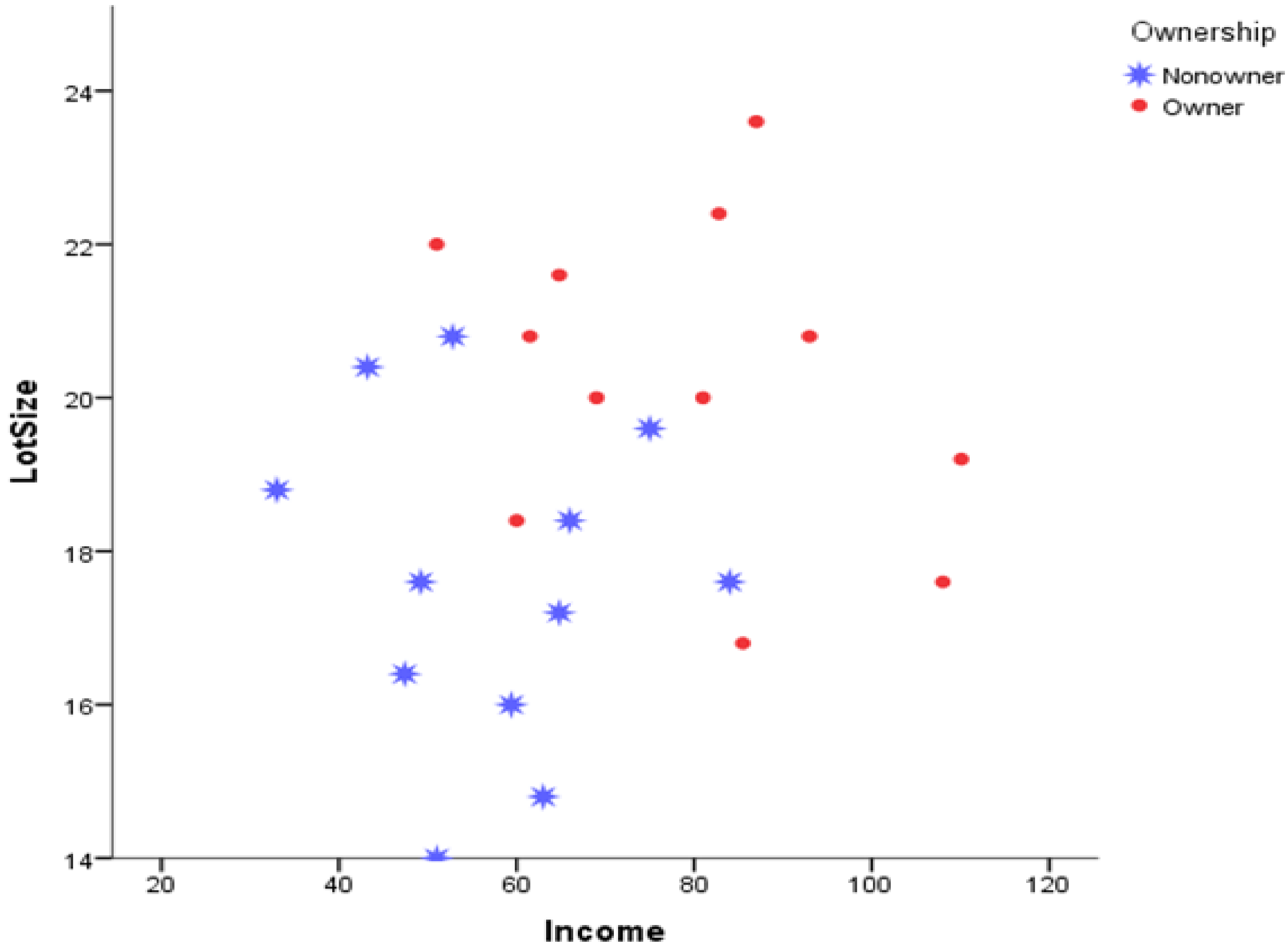
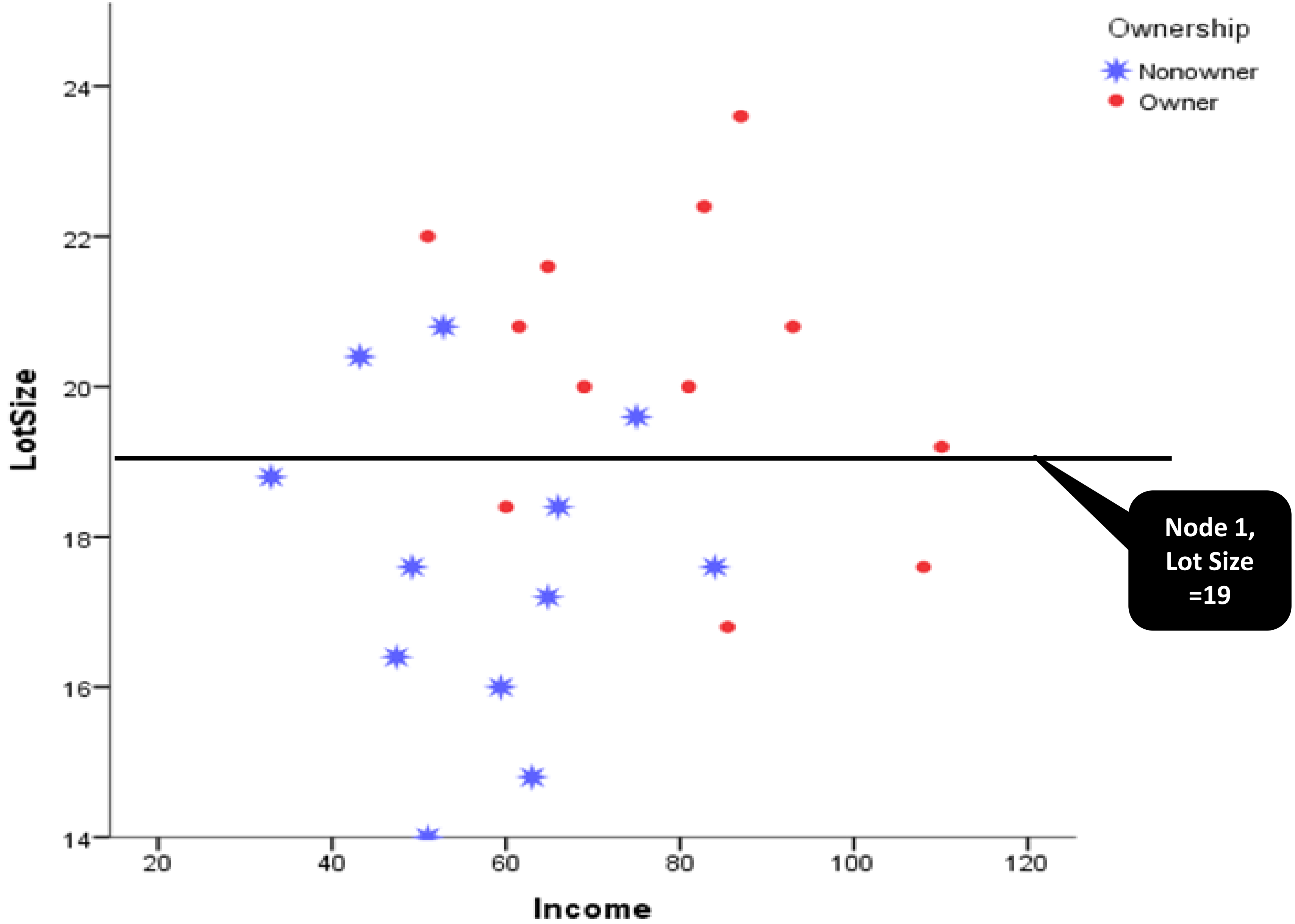
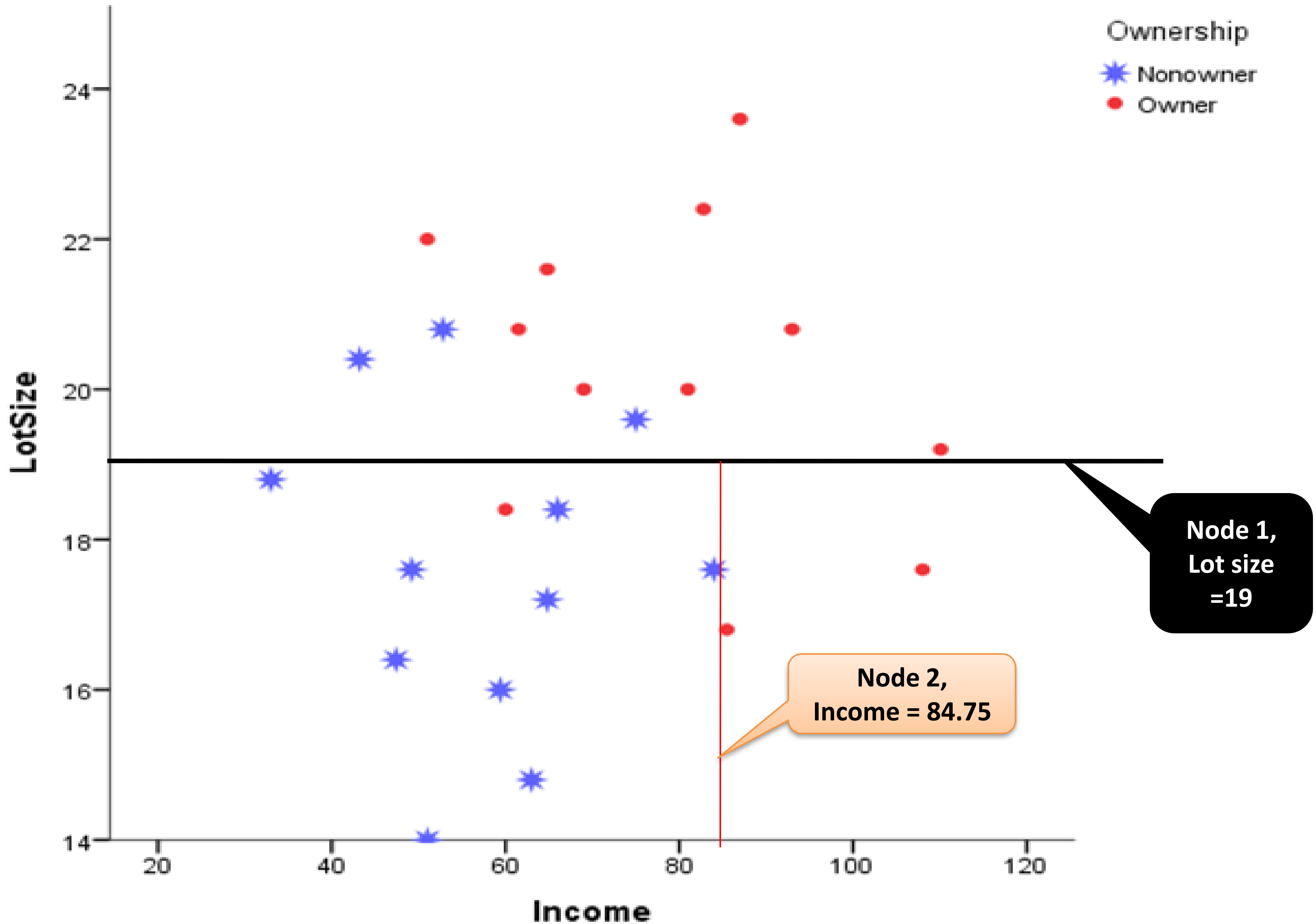
Decision Trees

Basic Concepts

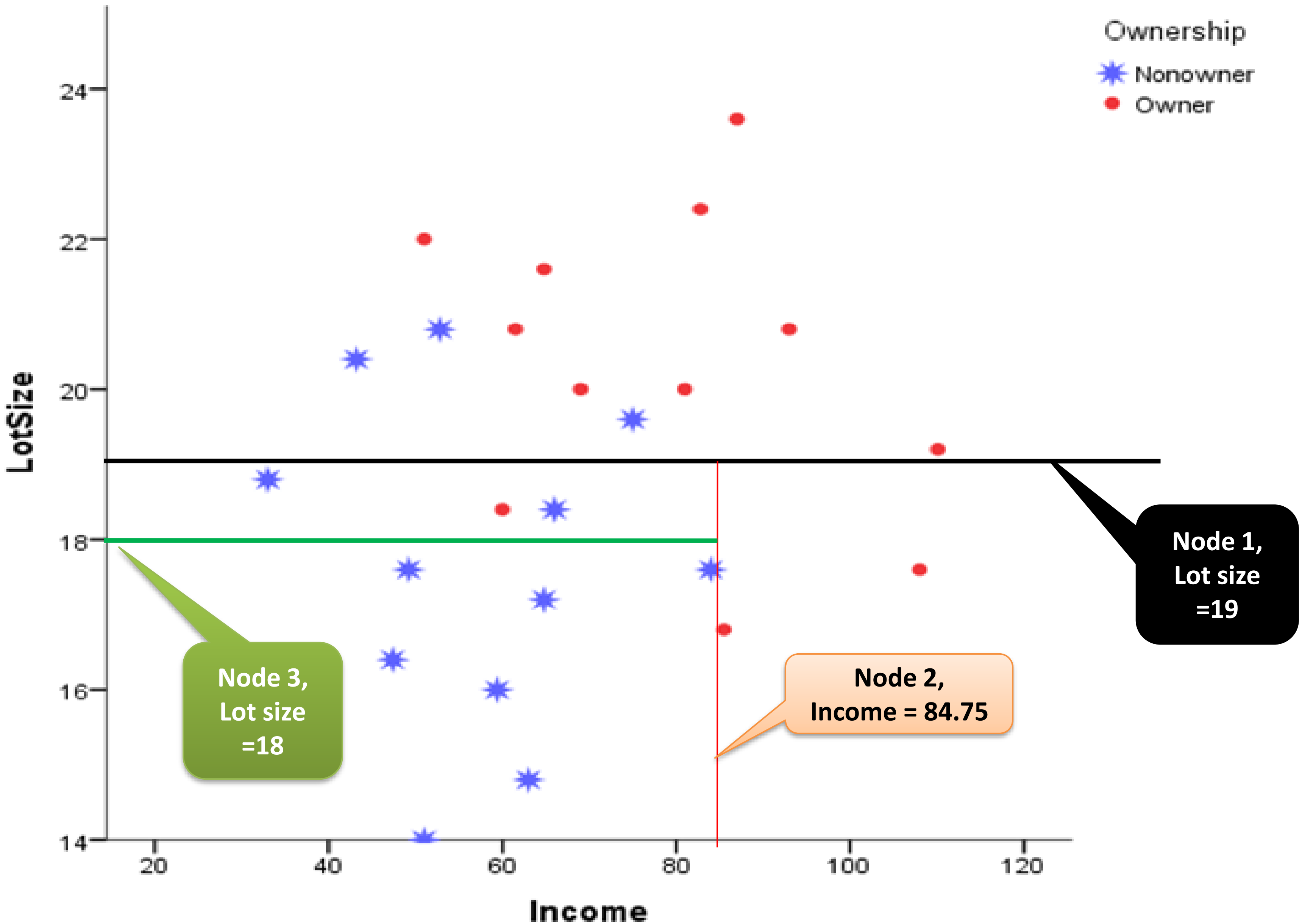
Household Number	Income	Lot Size	Ownership
1	60	18.4	Owner
2	85.5	16.8	Owner
3	64.8	21.6	Owner
4	61.5	20.8	Owner
5	87	23.6	Owner
6	110.1	19.2	Owner
7	108	17.6	Owner
8	82.8	22.4	Owner
9	69	20	Owner
10	93	20.8	Owner
11	51	22	Owner
12	81	20	Owner
13	75	19.6	Nonowner
14	52.8	20.8	Nonowner
15	64.8	17.2	Nonowner
16	43.2	20.4	Nonowner
17	84	17.6	Nonowner
18	49.2	17.6	Nonowner
19	59.4	16	Nonowner
20	66	18.4	Nonowner
21	47.4	16.4	Nonowner
22	33	18.8	Nonowner
23	51	14	Nonowner
24	63	14.8	Nonowner


Ownership
Nonowner
Owner
LotSize
24
22
20
18
16
14
20
40
60
80
100
120
Income

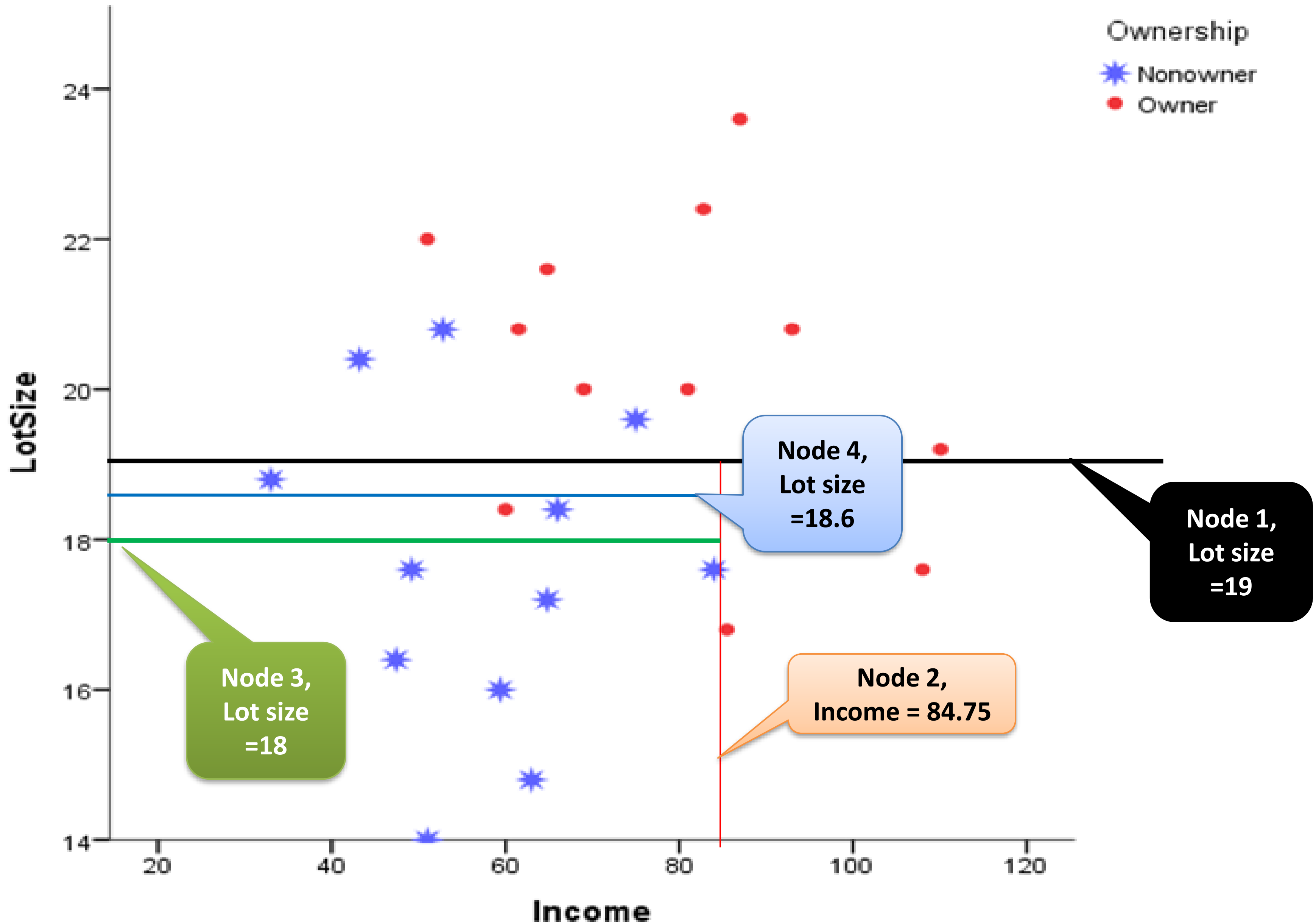

Ownership
Nonowner
Owner
LotSize
Income
24
22
20
18
16
14
20
40
60
80
100
120
Node 1,
Lot Size
=19

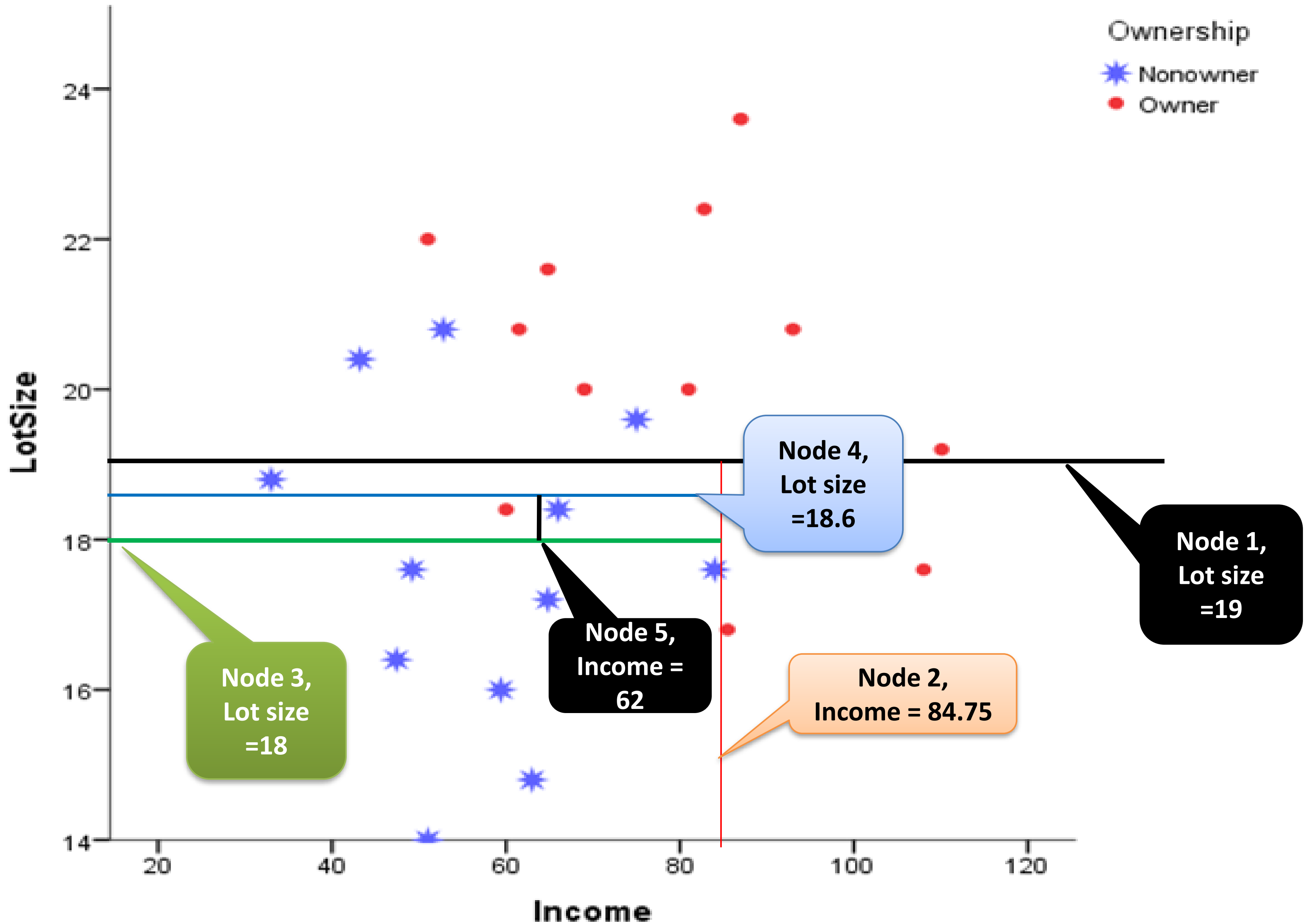


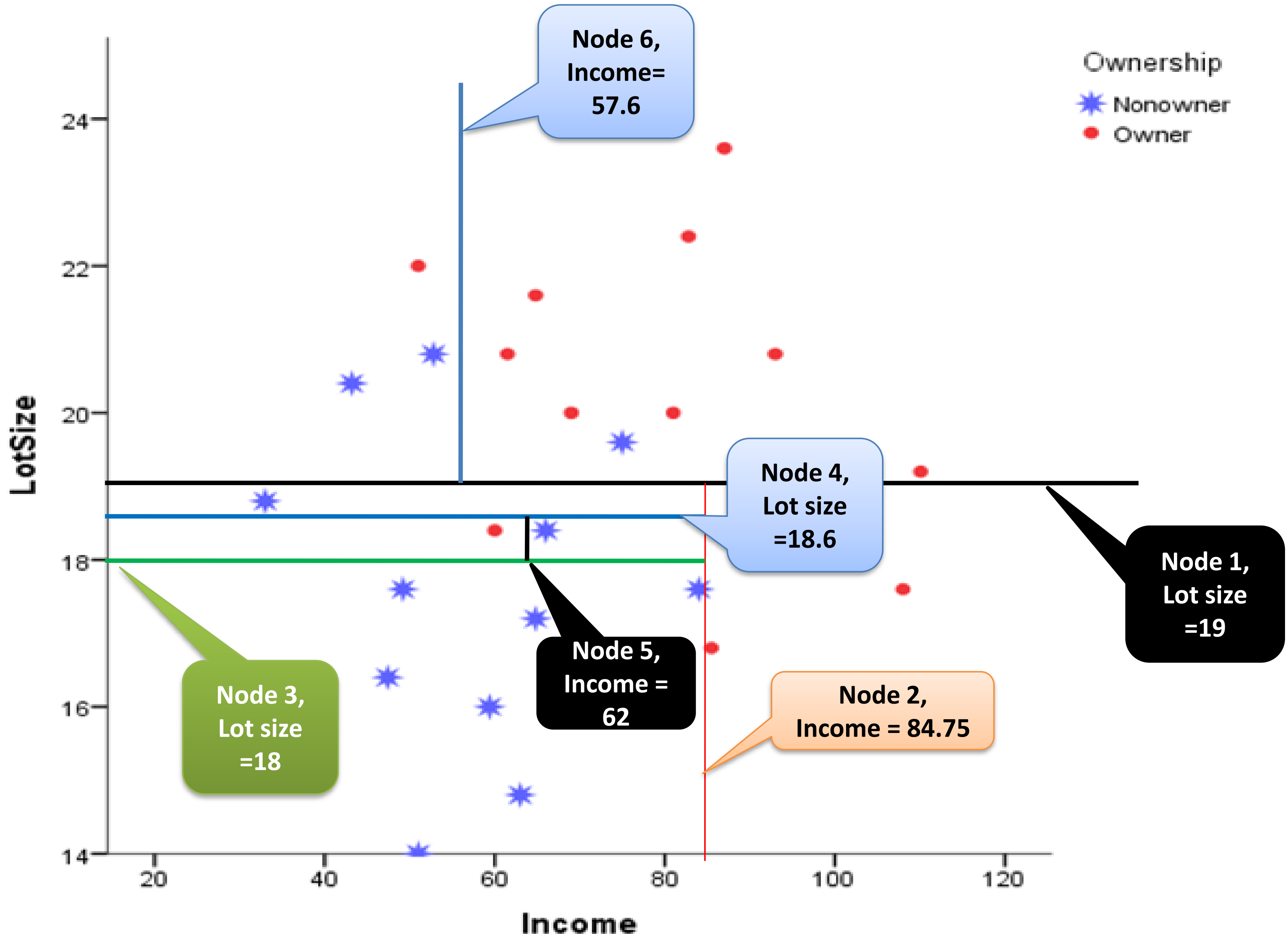
Ownership
Nonowner
Owner
LotSize
24
22
20
18
16
14
20
40
60
80
100
120
Income
Node 1,
Lot size
=19
Node 2,
Income = 84.75



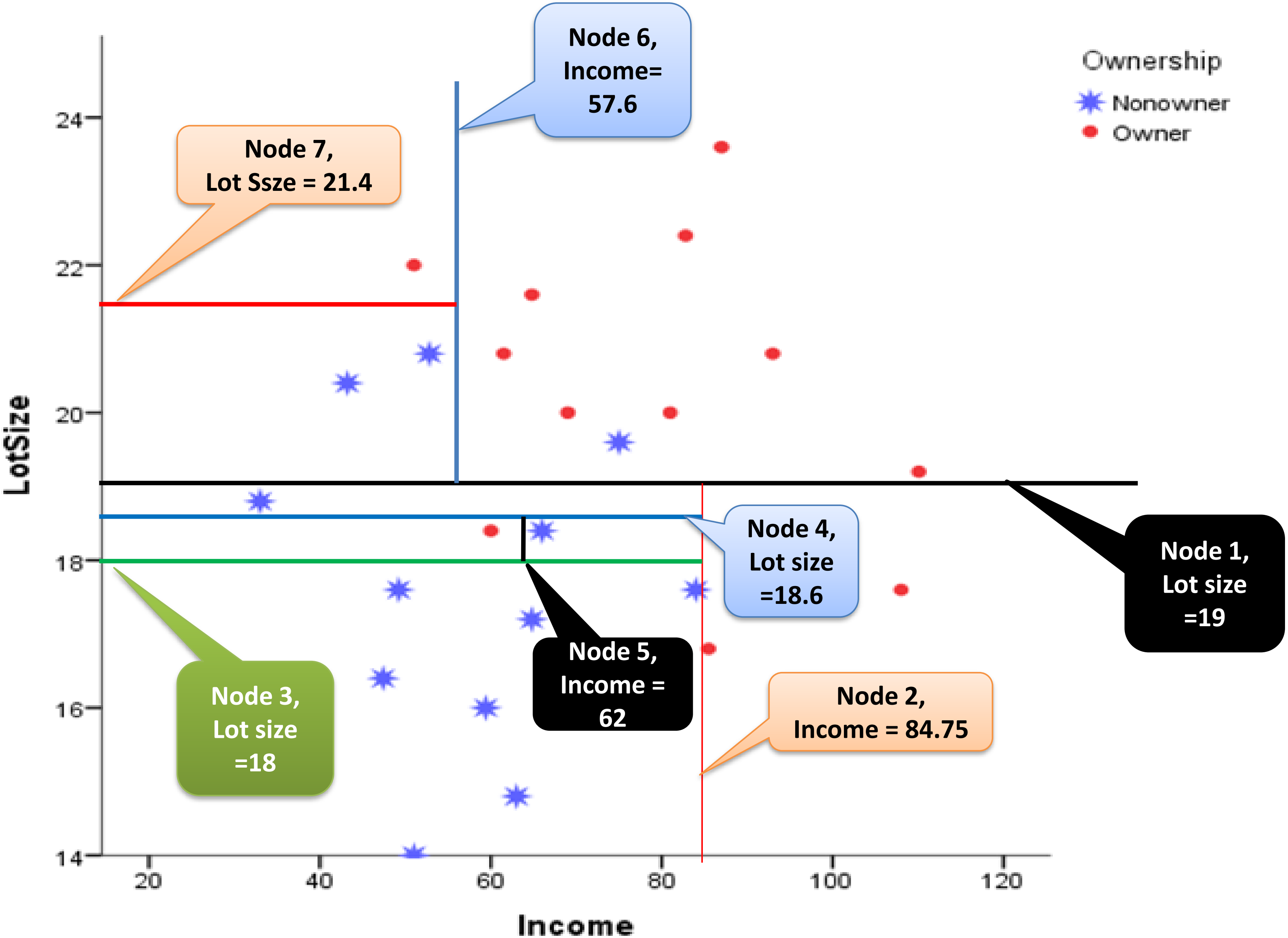
Ownership
Nonowner
Owner
LotSize
24
22
20
18
16
14
20
40
60
80
100
120
Income
Node 1,
Lot size
=19
Node 2,
Income = 84.75
Node 3,
Lot size
=18


Ownership
Nonowner
Owner
LotSize
24
22
20
18
16
14
20
40
60
80
100
120
Income
Node 4,
Lot size
=18.6
Node 1,
Lot size
=19
Node 3,
Lot size
=18
Node 2,
Income = 84.75

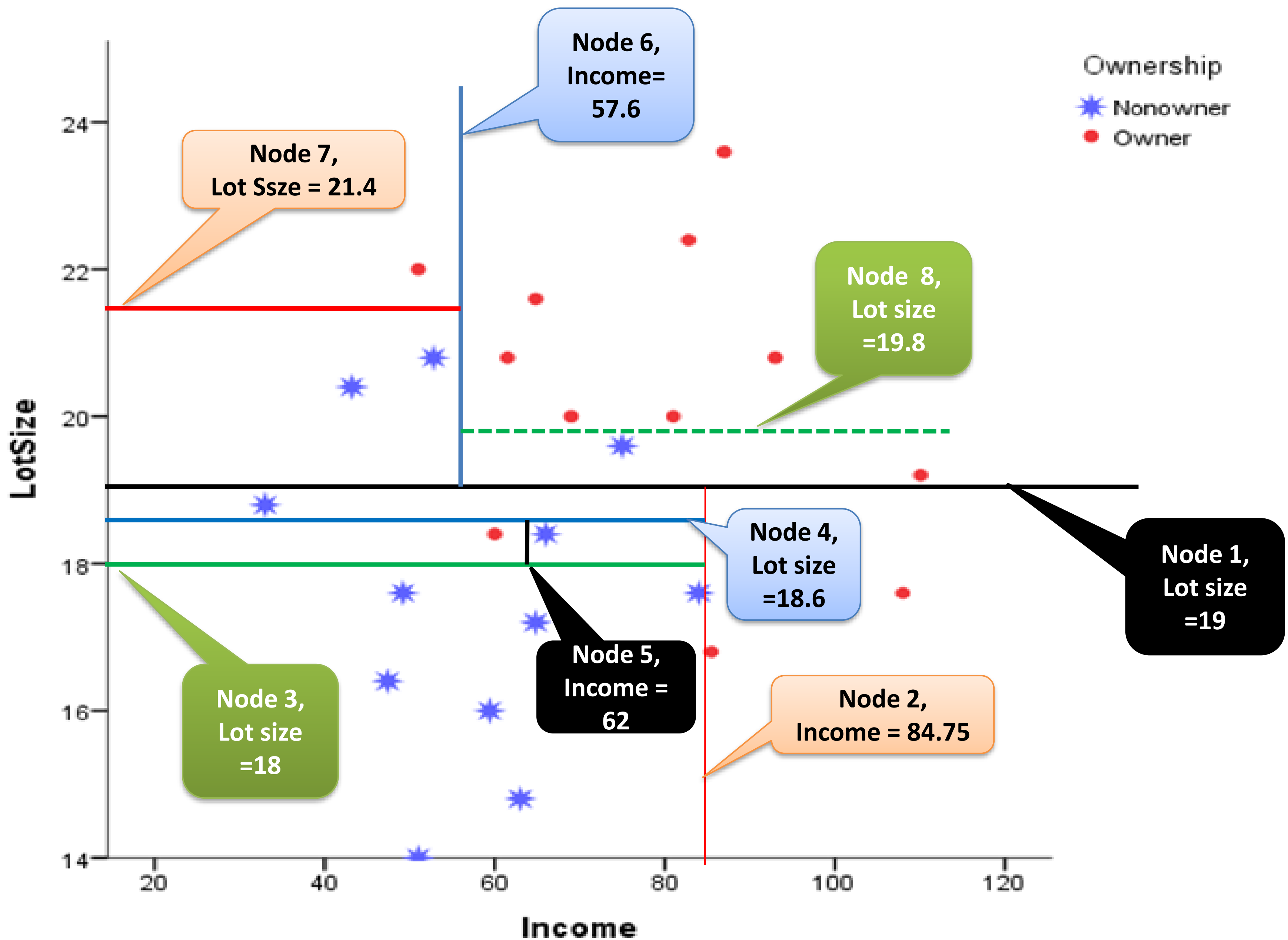

Ownership
Nonowner
Owner
LotSize
24
22
20
18
16
14
20
40
60
80
100
120
Income
Node 4, Lot size =18.6
Node 1, Lot size =19
Node 5, Income = 62
Node 3, Lot size =18
Node 2, Income = 84.75



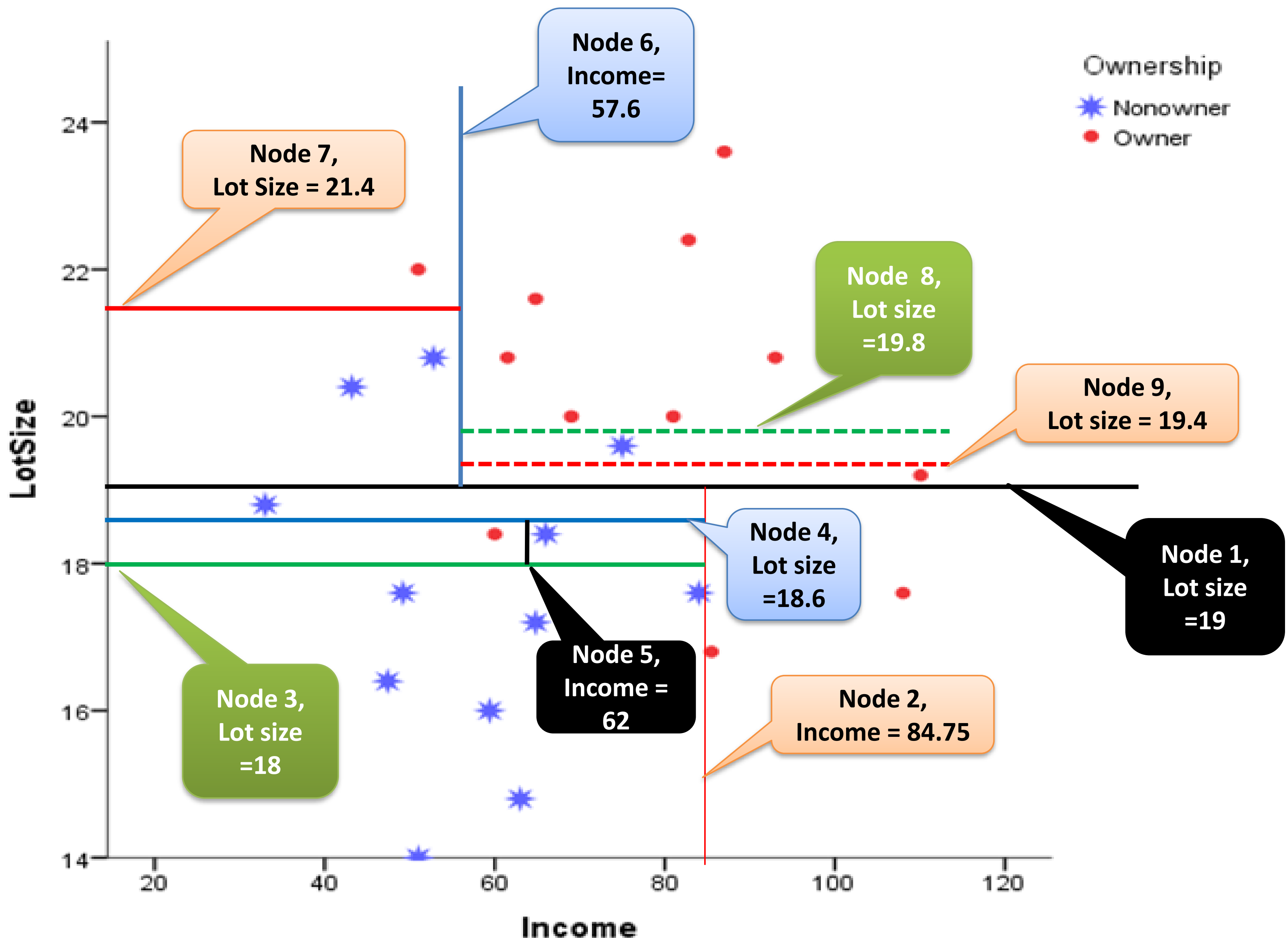
Node 6,
Income=
57.6
Ownership
Nonowner
Owner
LotSize
24
22
20
18
16
14
20
40
60
80
100
120
Income
Node 4,
Lot size
=18.6
Node 1,
Lot size
=19
Node 3,
Lot size
=18
Node 5,
Income =
62
Node 2,
Income = 84.75



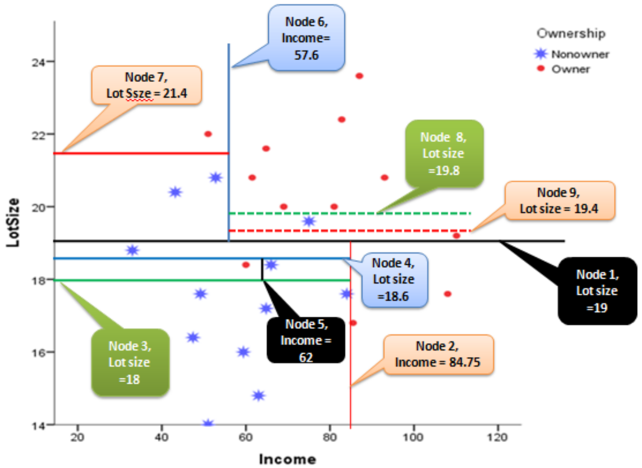
Node 6, Income= 57.6
Node 7, Lot Ssze = 21.4
Ownership
Nonowner
Owner
24
22
20
18
16
14
LotSize
Node 4, Lot size =18.6
Node 1, Lot size =19
Node 5, Income = 62
Node 3, Lot size =18
Node 2, Income = 84.75
20
40
60
80
100
120
Income

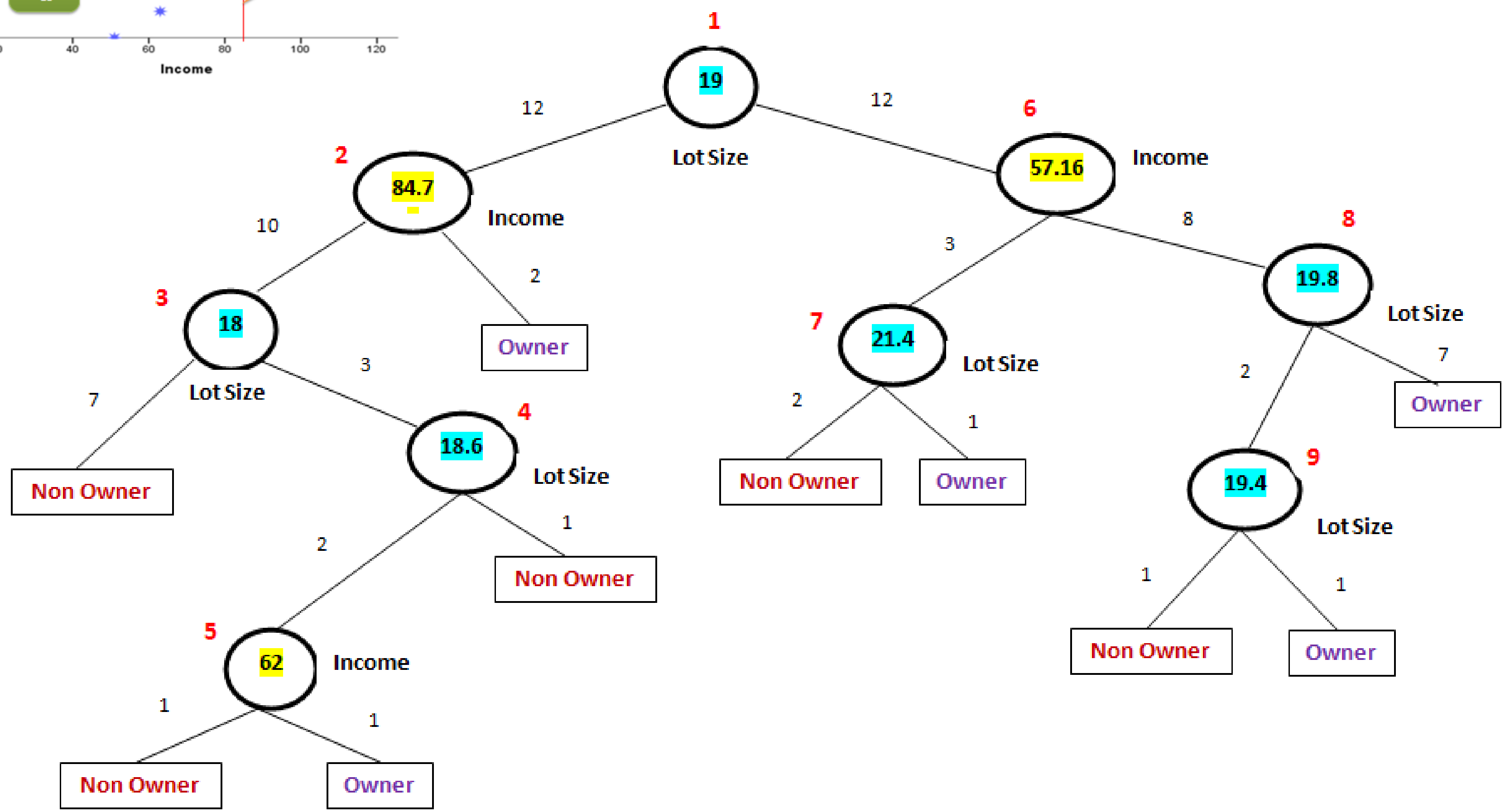


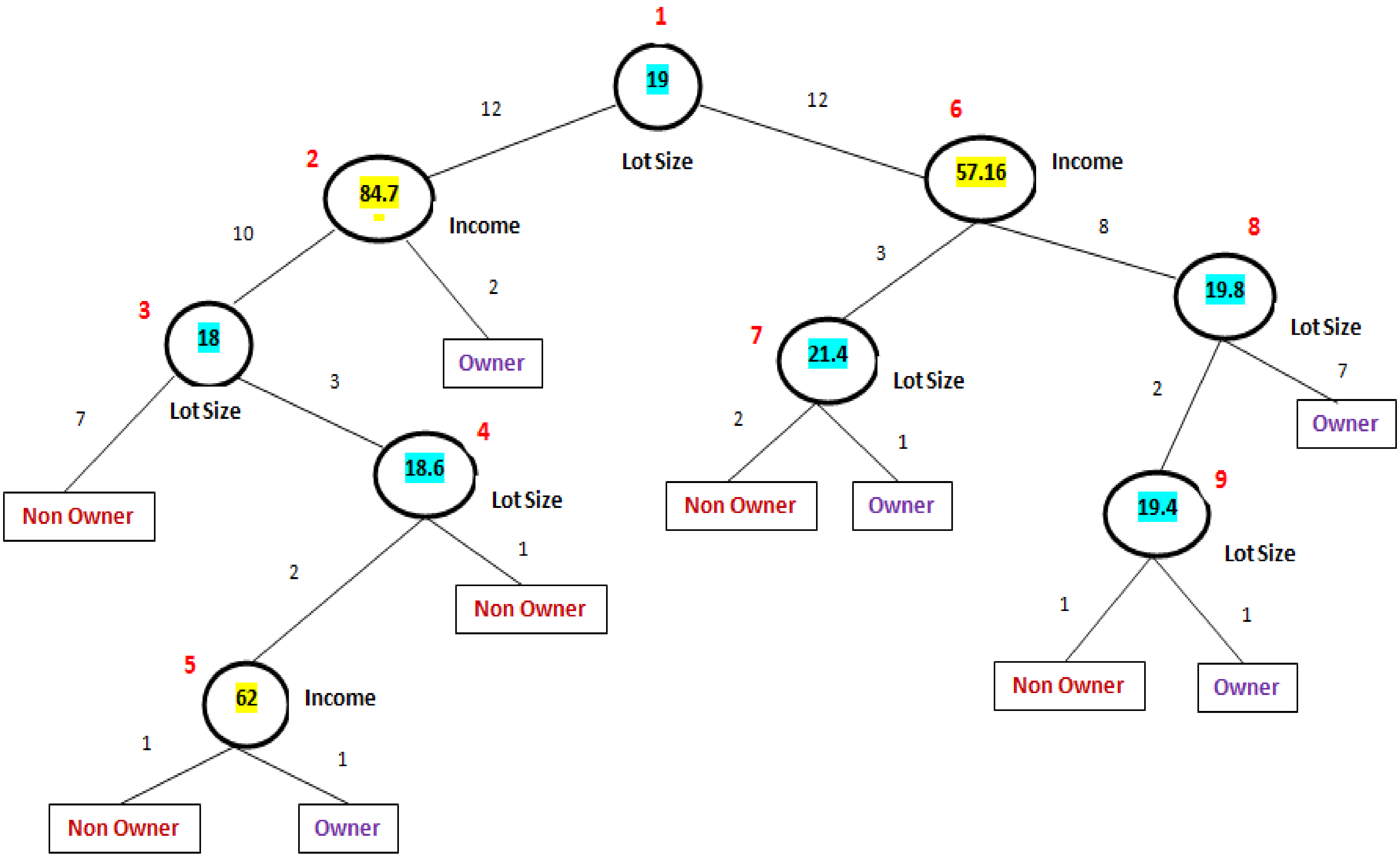
Node 6,
Income=
57.6
Ownership
Nonowner
Owner
Node 7,
Lot Ssze = 21.4
Node 8,
Lot size
=19.8
Node 4,
Lot size
=18.6
Node 1,
Lot size
=19
Node 5,
Income =
62
Node 3,
Lot size
=18
Node 2,
Income = 84.75
LotSize
Income
24
22
20
18
16
14
20
40
60
80
100
120



Node 6,
Income=
57.6
Node 7,
Lot Size = 21.4
Ownership
Nonowner
Owner
Node 8,
Lot size
=19.8
Node 9,
Lot size = 19.4
Node 4,
Lot size
=18.6
Node 1,
Lot size
=19
Node 5,
Income =
62
Node 3,
Lot size
=18
Node 2,
Income = 84.75
LotSize
24
22
20
18
16
14
20
40
60
80
100
120
Income


Ownership
Nonowner
Owner
LotSize
Income
Node 1, Lot size =19
Node 2, Income = 84.75
Node 3, Lot size =18
Node 4, Lot size =18.6
Node 5, Income = 62
Node 6, Income= 57.6
Node 7, Lot Ssze = 21.4
Node 8, Lot size =19.8
Node 9, Lot size = 19.4


1
19
Lot Size
12
12
2
84.7
Income
6
57.16
Income
10
2
Owner
3
8
3
18
Lot Size
3
7
Non Owner
4
18.6
Lot Size
2
1
Non Owner
5
62
Income
1
1
Non Owner
Owner
7
21.4
Lot Size
2
1
Non Owner
Owner
8
19.8
Lot Size
2
7
Owner
9
19.4
Lot Size
1
1
Non Owner
Owner



1
19
Lot Size
12
12
2
84.7
Income
6
57.16
Income
10
2
3
8
3
18
Lot Size
Owner
7
21.4
Lot Size
8
19.8
Lot Size
7
3
4
18.6
Lot Size
2
1
2
7
Owner
Non Owner
Non Owner
Owner
2
1
Non Owner
9
19.4
Lot Size
5
62
Income
1
1
1
1
Non Owner
Owner
Non Owner
Owner

Impurity
Gini Index

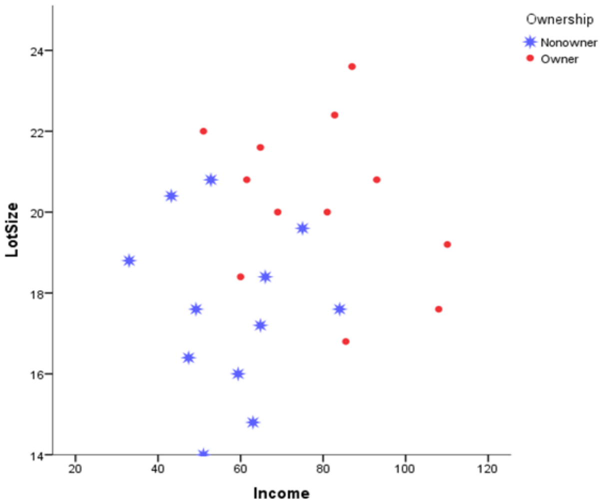
$$Gini\ Index\ for\ rectangle\ A, I(A) = 1 - \sum_{k=1}^{m} p_k^2$$

$$where, k is\ particular\ class\ (owner, non-owner); m\ is\ number\ of\ classes$$

$$Gini\ Index\ for\ rectangle\ A, I(A) = 1 - \left\{\left(\frac{12}{24}\right)^2 + \left(\frac{12}{24}\right)^2\right\}$$

$$Gini\ Index\ for\ rectangle\ A, I(A) = 1 - \{(0.5)^2 + (0.5)^2\}$$

$$Gini\ Index\ for\ rectangle\ A, I(A) = 1 - \{0.25 + 0.25\} = \boxed{0.50}$$



Gini Index before split

For Upper Rectangle, there were 9 owners and 3 non-owners

$$Gini\ Index\ for\ rectangle\ A, I(A) = 1 - \left\{\left(\frac{9}{12}\right)^2 + \left(\frac{3}{12}\right)^2\right\}$$

$$Gini\ Index\ for\ rectangle\ A, I(A) = 1 - \{(0.75)^2 + (0.25)^2\}$$

$$Gini\ Index\ for\ rectangle\ A, I(A) = 1 - \{0.5625 + 0.0625\} = \mathbf{0.375}$$

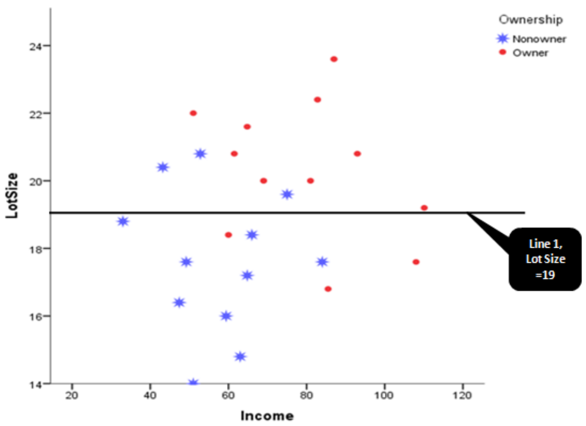
Total Impurity

For Lower Rectangle, there were 3 owners and 9 non-owners

$$Gini\ Index\ for\ rectangle\ A, I(A) = 1 - \left\{\left(\frac{3}{12}\right)^2 + \left(\frac{9}{12}\right)^2\right\}$$

$$Gini\ Index\ for\ rectangle\ A, I(A) = 1 - \{(0.25)^2 + (0.75)^2\}$$

$$Gini\ Index\ for\ rectangle\ A, I(A) = 1 - \{0.0625 + 0.5625\} = \mathbf{0.375}$$



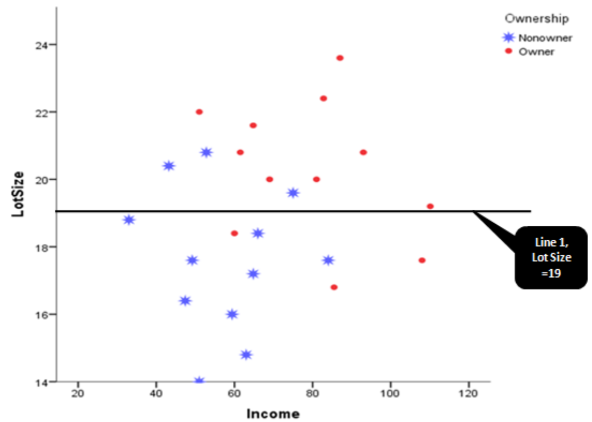
Total Impurity

$$TI = w_{Upper\ Rectangle} \times GI_{Upper\ Rectangle} + w_{Lower\ Rectangle} \times GI_{Lower\ Rectangle}$$

$$TI = 0.50 \times 0.375 + 0.5 \times 0.375$$

$$TI = \mathbf{0.375}$$

Entropy Measure



Entropy measure before split

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

$$entopy\,(A) = \; -0.5 \times log_2(0.5) + \; -0.5 \times log_2(0.5)$$

$$entopy\,(A) = \; -0.5 \times (-1) + \; -0.5 \times (-1)$$

$$entopy\,(A) = \; 0.5 + 0.5 = \mathbf{1}$$

TOP RECTANGLE	Proportion of Owner in Top Rectangle	Proportion of non-Owner in Top Rectangle
p_k	-0.75	-0.25
$log_2(p_k)$	-0.415037499	-2
$p_k*log_2(p_k)$	0.311278124	0.5
Total =		0.811278124

BOTTOM RECTANGLE	Proportion of Owner in Bottom Rectangle	Proportion of non-Owner in Bottom Rectangle
p_k	-0.25	-0.75
$log_2(p_k)$	-2	-0.415037499
$p_k*log_2(p_k)$	0.5	0.311278124
Total =		0.811278124

Weighted Sum = 0.5*0.811 + 0.5*0.811 = **0.811**

Techniques to improve Classification Accuracies

Ensemble Models

a) Bagging
b) Boosting
c) Random Forest

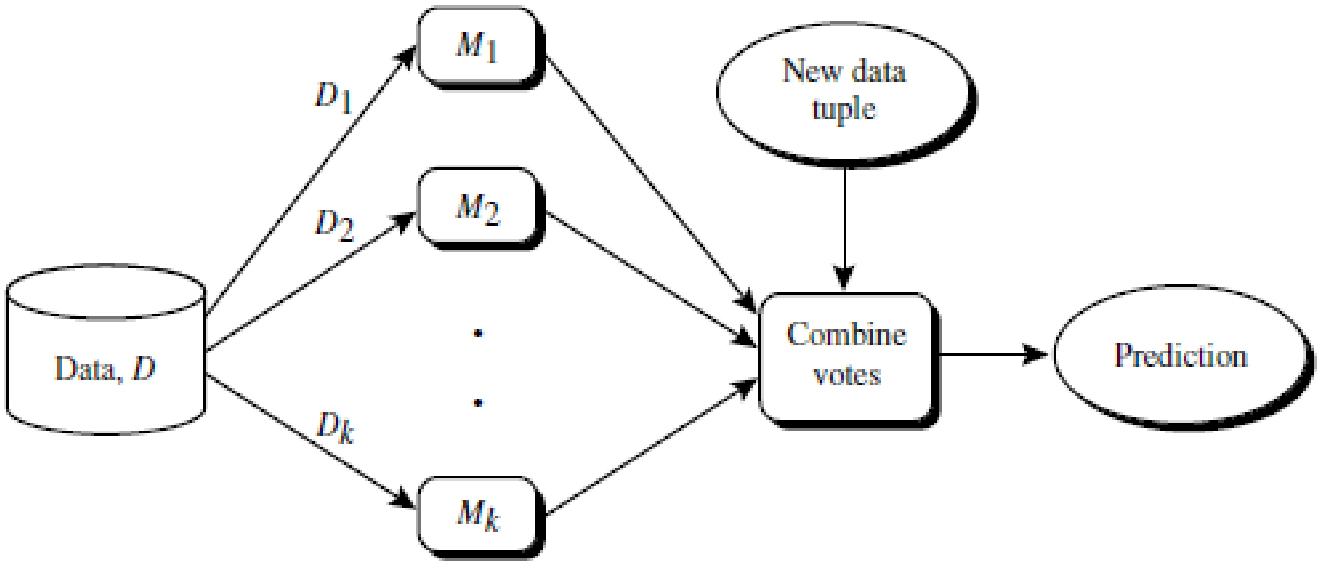


Figure 8.21 Increasing classifier accuracy: Ensemble methods generate a set of classification models, $M_1, M_2, \ldots, M_k$. Given a new data tuple to classify, each classifier "votes" for the class label of that tuple. The ensemble combines the votes to return a class prediction.

Bagging

1. Apply ***n*** classifiers to training sets
2. Each training set is a Bootstrap sample
3. Each classifier has **one** vote
4. So, for deciding about the fate (group membership) of row X, votes are counted

Majority vote is the final verdict for X

Algorithm - **Bagging**

Algorithm: Bagging. The bagging algorithm—create an ensemble of classification models for a learning scheme where each model gives an equally weighted prediction.

Input:

- D, a set of d training tuples;
- k, the number of models in the ensemble;
- a classification learning scheme (decision tree algorithm, naïve Bayesian, etc.).

Output: The ensemble—a composite model, $M*$.

Method:

(1) **for** $i = 1$ to k **do** // create k models:
(2) create bootstrap sample, D_i, by sampling D with replacement;
(3) use D_i and the learning scheme to derive a model, M_i;
(4) **endfor**

To use the ensemble to classify a tuple, X:

let each of the k models classify $\boldsymbol{X}$ and return the majority vote;

Figure 8.23 Bagging.

Random Forest

Decorrelation among trees is incorporated

Random Selection of features

Adaptive Boosting

1. A sample (training set) is generated from D {with replacement}
2. Initially AdaBoost assigns each training row an equal weight $\boxed{{}^{1}/_{d}}$

Xi	weights
1	0.1
2	0.1
3	0.1
4	0.1
5	0.1
6	0.1
7	0.1
8	0.1
9	0.1
10	0.1

Adaptive Boosting

3. Classifier does classification on training data set

Xi	weights	result
1	0.1	cc
2	0.1	cc
3	0.1	cc
4	0.1	mc
5	0.1	cc
6	0.1	cc
7	0.1	mc
8	0.1	cc
9	0.1	mc
10	0.1	cc

Adaptive Boosting

4. Classifier does classification on training data set and error is found $error\ (M_i)$

To compute the error rate of model *Mi*, we sum the weights of each rows in *Di* that *Mi* misclassified. That is,

$$error\ (M_i) = \sum_{j=1}^{d} w_j \times err(X_j)$$

where, $err(X_j)$ is the misclassification error of row *Xj*.

If *Xj* was misclassified, then $err(X_j) = 1$, otherwise, $err(X_j) = 0$

If, error of model Mi is poor that is $error\ (M_i) > 0.5$, Model is bad and abandoned. In this case, a new model is generated with new training sample.

Adaptive Boosting

$$error\,(M_i) = \sum_{j=1}^{d} w_j \times err(X_j)$$

Xi	weights	result	err(Xi)	wt*err(Xi)
1	0.1	cc	0	0
2	0.1	cc	0	0
3	0.1	cc	0	0
4	0.1	mc	1	0.1
5	0.1	cc	0	0
6	0.1	cc	0	0
7	0.1	mc	1	0.1
8	0.1	cc	0	0
9	0.1	mc	1	0.1
10	0.1	cc	0	0
	1		error Mi =	0.3

Adaptive Boosting

5. Weights are adjusted
6. If row is misclassified, its weight is increased and vice versa

The error rate of *Mi* affects how the weights of the training sample are updated.

If a row, in round i is correctly classified, its weight is multiplied by

$$\boldsymbol{error(M_i)} / \boldsymbol{(1 - error(M_i))}$$

Adaptive Boosting

The error rate of *Mi* affects how the weights of the training sample are updated.

If a row, in round i is correctly classified, its weight is multiplied by $\mathit{error}(M_i) / (1 - \mathit{error}(M_i))$

Wt = 0.1 multiply by 0.4285714

Xi	weights	result	err(Xi)	wt*err(Xi)	updated wt for correctly classified
1	0.1	cc	0	0	0.042857143
2	0.1	cc	0	0	0.042857143
3	0.1	cc	0	0	0.042857143
4	0.1	mc	1	0.1	0.1
5	0.1	cc	0	0	0.042857143
6	0.1	cc	0	0	0.042857143
7	0.1	mc	1	0.1	0.1
8	0.1	cc	0	0	0.042857143
9	0.1	mc	1	0.1	0.1
10	0.1	cc	0	0	0.042857143
	1		error Mi =	0.3	0.6
			error Mi/(1-errorMi) =	0.4285714	sum of new wts

Misclassified wts are untouched

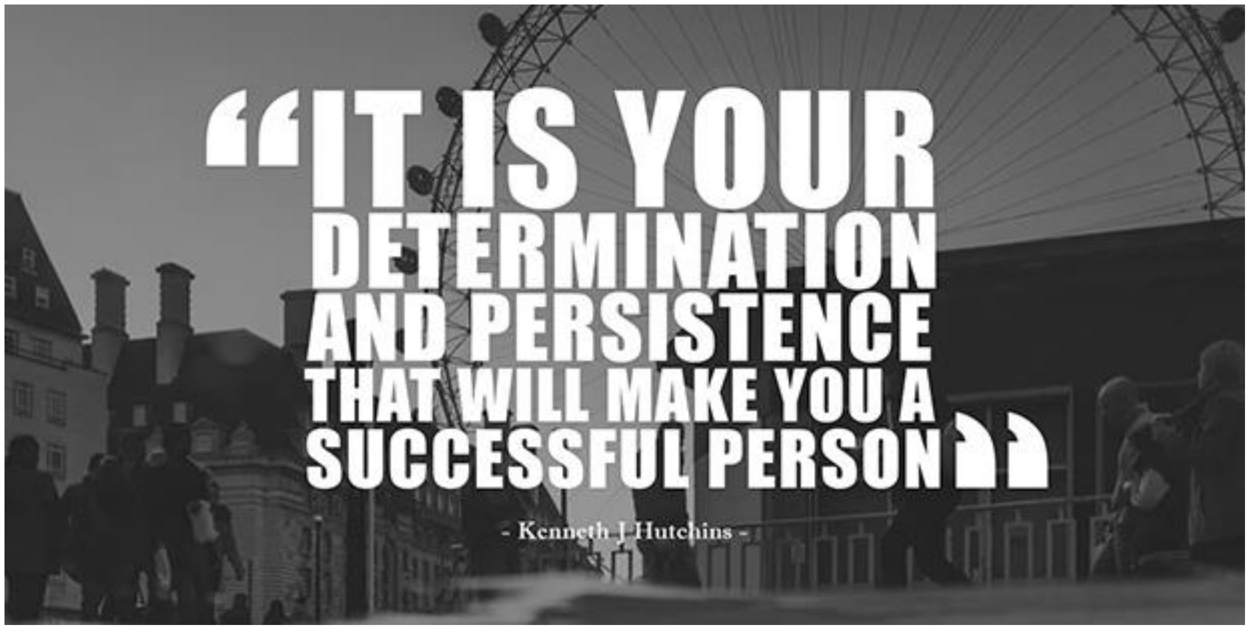
If
0.6 is for 0.042857143
Then what will be equivalent for 1?
= 0.04/0.6

Adaptive Boosting

Divide by 0.6 to get normalized wt

7. New weights are normalized by

Xi	weights	result	err(Xi)	wt*err(Xi)	updated wt for correctly classified	normalized wts
1	0.1	cc	0	0	0.042857143	0.071428571
2	0.1	cc	0	0	0.042857143	0.071428571
3	0.1	cc	0	0	0.042857143	0.071428571
4	0.1	mc	1	0.1	0.1	0.166666667
5	0.1	cc	0	0	0.042857143	0.071428571
6	0.1	cc	0	0	0.042857143	0.071428571
7	0.1	mc	1	0.1	0.1	0.166666667
8	0.1	cc	0	0	0.042857143	0.071428571
9	0.1	mc	1	0.1	0.1	0.166666667
10	0.1	cc	0	0	0.042857143	0.071428571
	1		error Mi =	0.3	0.6	1
			error Mi/(1-errorMi) =	0.4285714	sum of new wts	


"IT IS YOUR
DETERMINATION
AND PERSISTENCE
THAT WILL MAKE YOU A
SUCCESSFUL PERSON"
- Kenneth J Hutchins -