Lecture 17: Machine learning: classification and regression tree & K-nearest neighbor

IST5573

統計方法 Statistical methods

2017/1/4

Machine learning

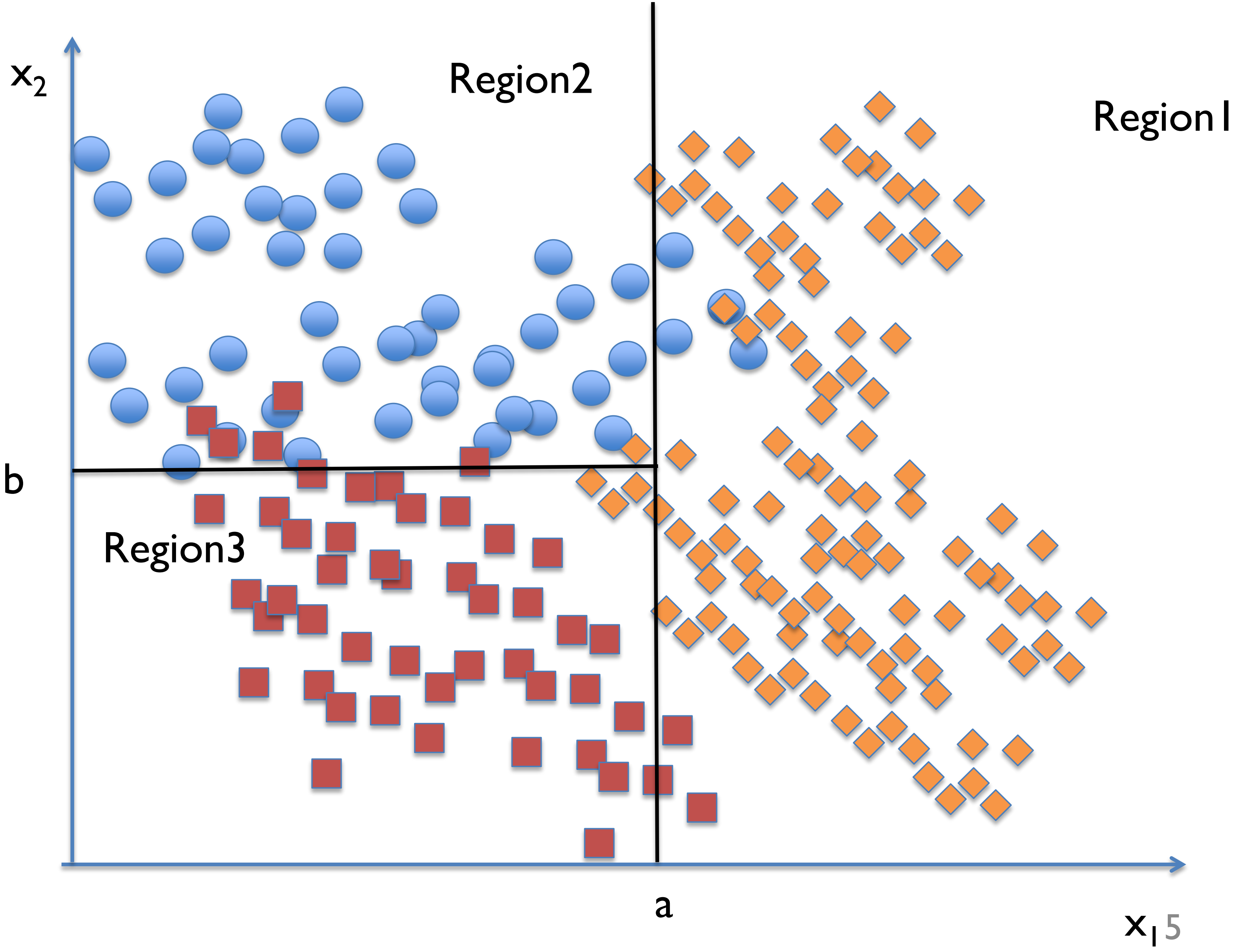
- Closely related to
  - Computational statistics
  - Mathematical optimization
  - Data mining
  - Supervised / unsupervised learning
- We will be studying
  - Support vector machine
  - Neural networks
  - Classification and regression tree
  - K-nearest neighbor

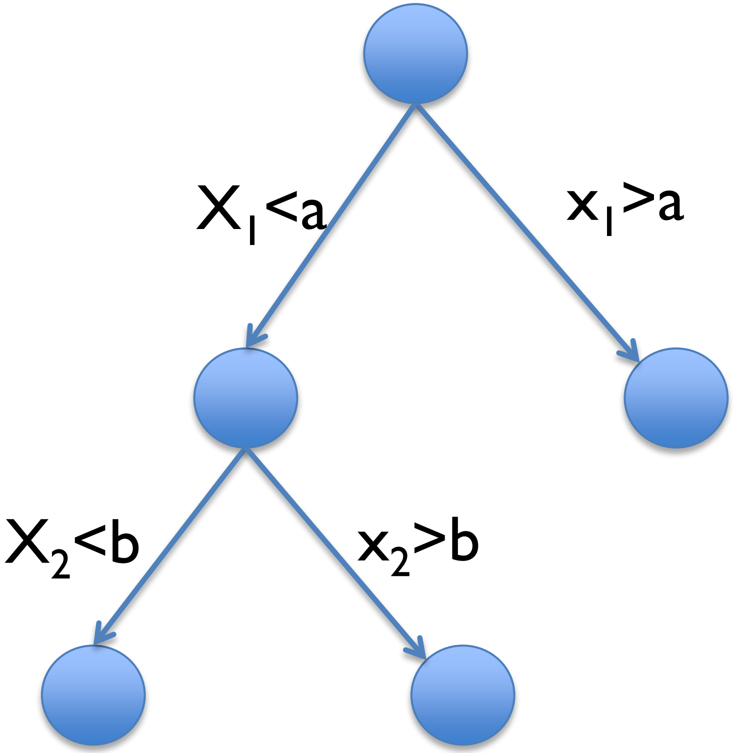
Classification and regression tree (CART)

These slides are courtesy of CS 109A/AC 209A/STAT 121A Data Science: Harvard University, Fall 2016

https://canvas.harvard.edu/courses/12656/files/3076446/download?verifier=JUFsbW6kMay2QYaopJzThbVoK7b3oEpzMIoOyuCI&wrap=1

Tree-based methods


x_2
Region2
Region1
b
Region3
a
x_1

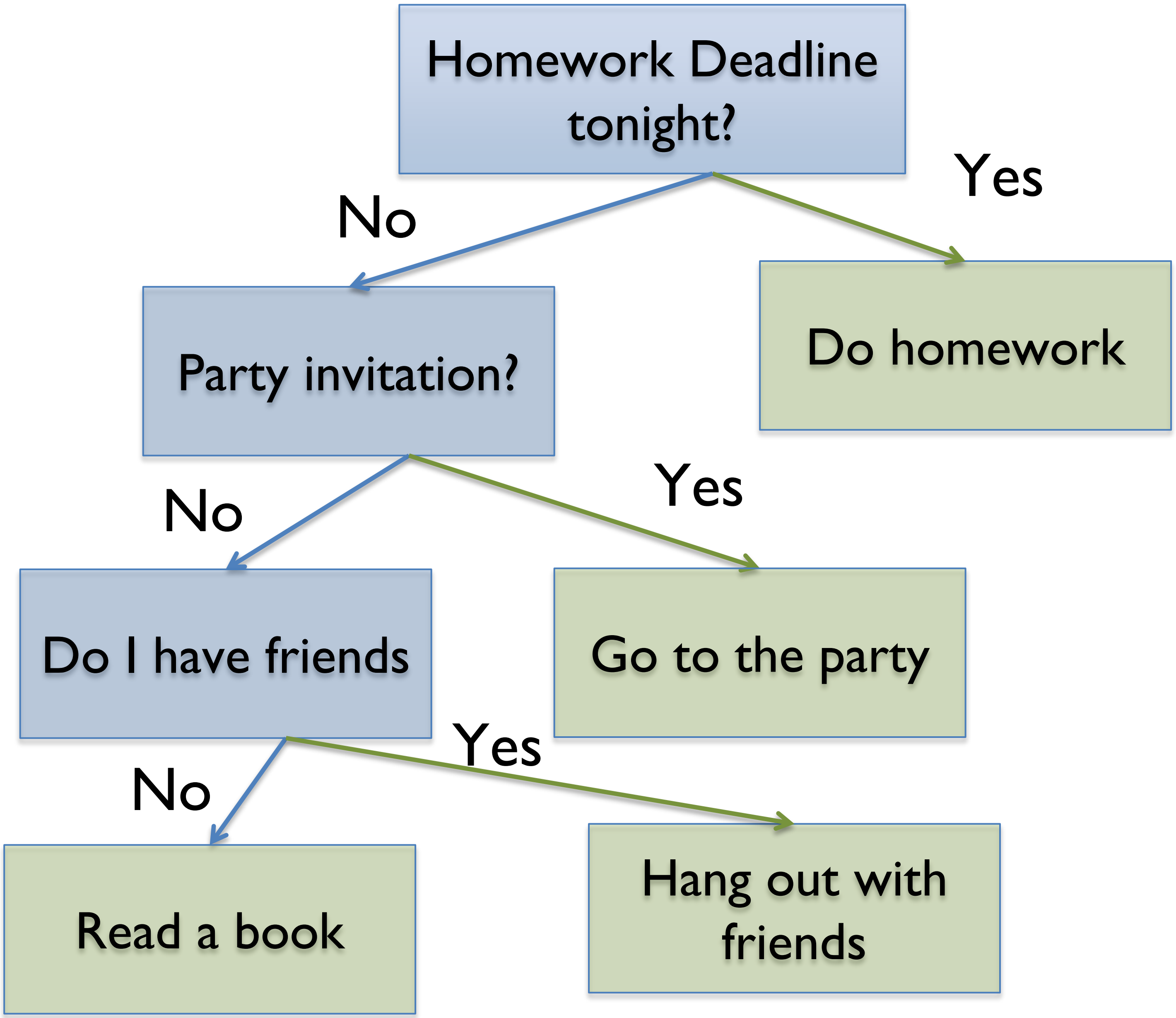

$X_1<a$
$x_1>a$
$X_2<b$
$x_2>b$

Basic idea

- Segment the predictor space into sub-regions and we learn from the training set the value to predict as the mean or mode or median of the respond variable of the training examples that are in that segment.

Why trees?

- What would you do tonight? Decide amongst the following:
  - Finish homework
  - Go to a party
  - Read a book
  - Hang out with friends



Homework Deadline tonight?

No

Yes

Do homework

Party invitation?

No

Yes

Go to the party

Do I have friends

No

Yes

Read a book

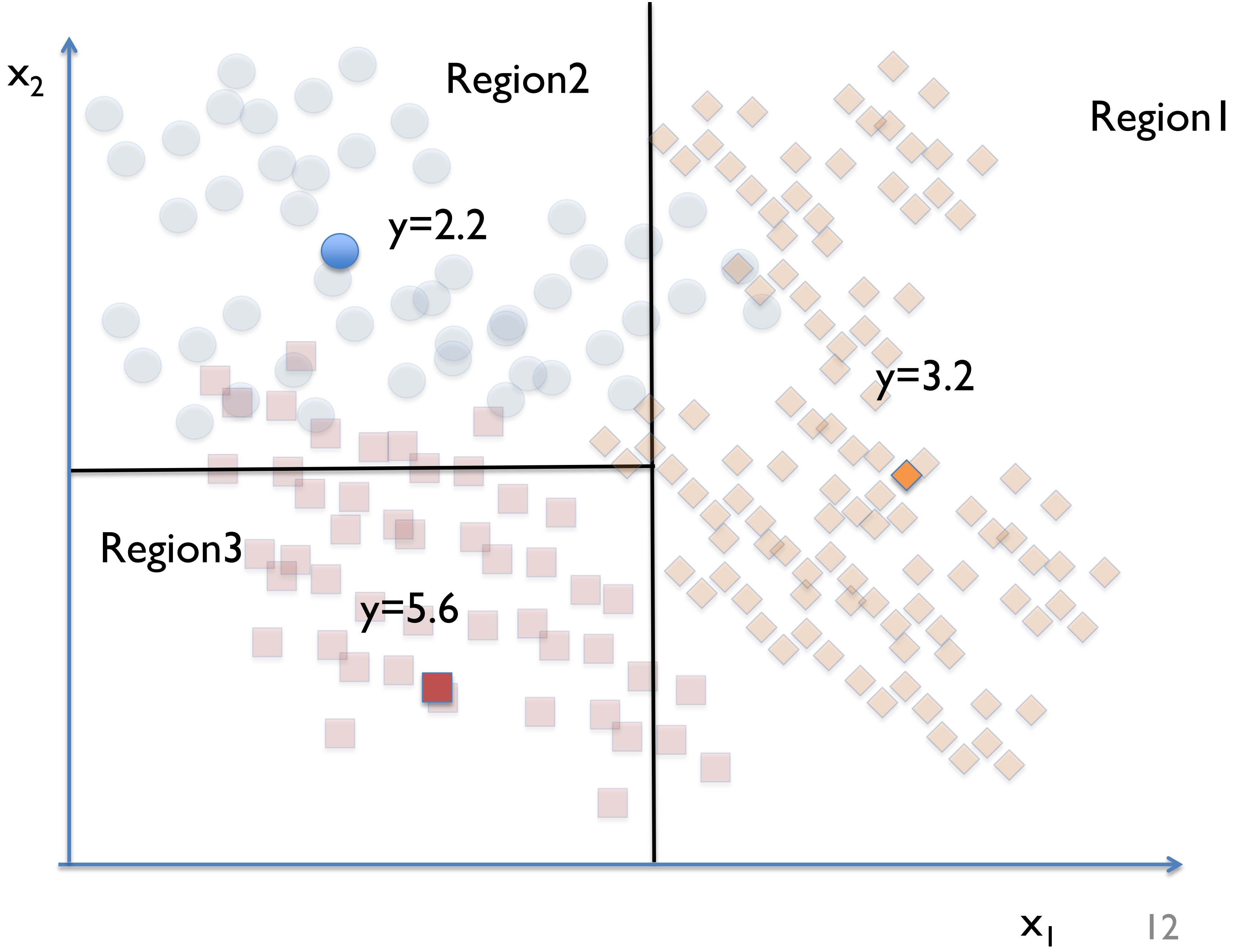
Hang out with friends

Why trees?

- We split the predictor space as brunches of a tree and therefore these methods are called decision tree methods

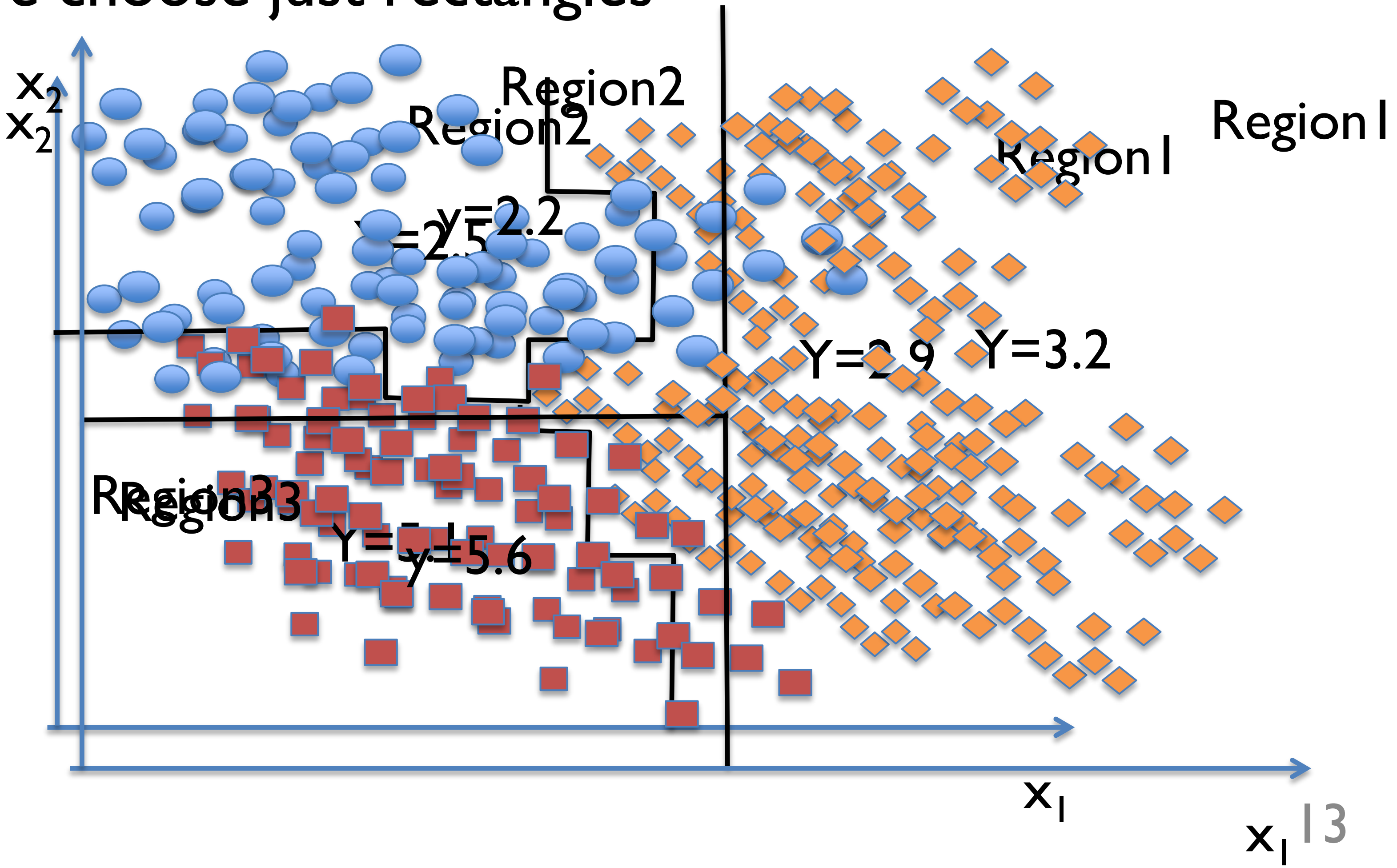
Regression trees

- Build a regression tree:
    - Divide the predictor space into J distinct not overlapping regions $[R_1, R_2, \cdots, R_J]$
    - We make the same prediction for all observations in the same region; use the mean of responses for all training observations that are in the region


x2
Region2
Region1
y=2.2
y=3.2
Region3
y=5.6
x1

Finding the sub-regions

The regions could have any shape.
But we choose just rectangles



- Our data $(\boldsymbol{x}_i, y_i), \boldsymbol{x}_i = (x_{i1}, \cdots, x_{ip}), i = 1, \cdots, N$
- Find boxes $R_1, \cdots, R_J$ that minimize the RSS

$$\mathrm{RSS} = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

where $\hat{y}_{R_j}$ is the mean response value of all training observations in the R_j region

- This computationally very expensive!
- **Solution:** Top down approach, greedy approach

recursive binary splitting

Recursive binary splitting

1. Consider all predictor X_j and all the all possible values of the cutpoints s for each of the predictors. Choose the predictor and cutpoint s.t. it minimizes the RSS

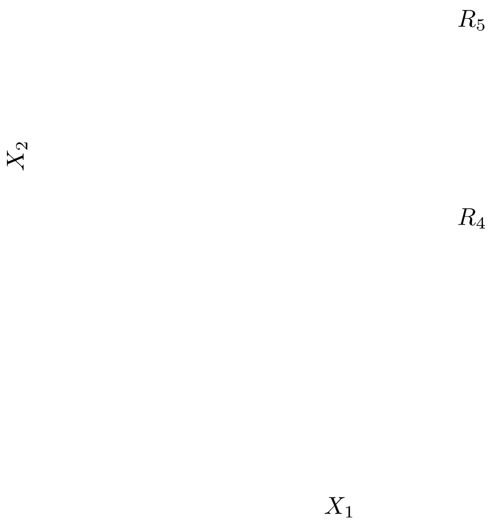
$$\sum_{i:\,\boldsymbol{x}_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\,\boldsymbol{x}_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

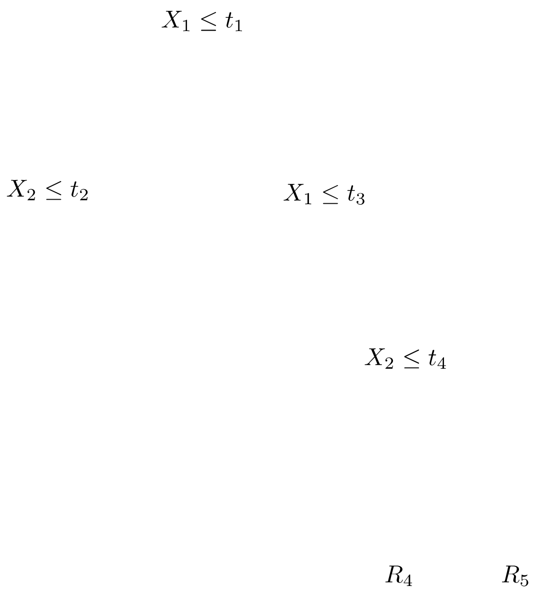
where $R_1(j,s) = \{\boldsymbol{X} | X_j \leq s\}$ and $R_2(j,s) = \{\boldsymbol{X} | X_j > s\}$.

This can be done quickly, assuming number of predictors is not very large

Recursive binary splitting (cont'd)

2. Repeat #1 but only consider the sub-regions
3. Stop: node contains only one class or node contains less than n data points or max depth is reached

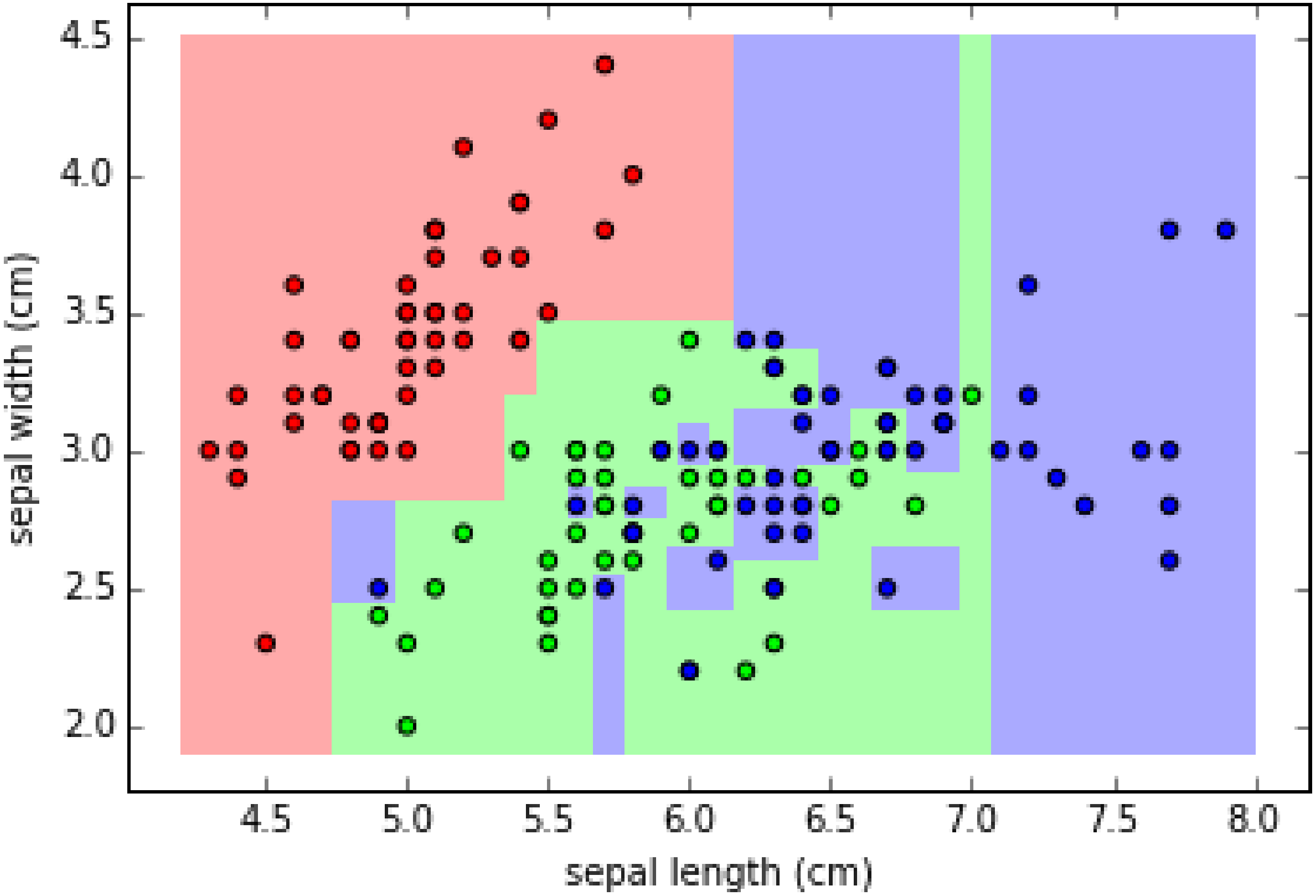

R_5
X_2
R_4
X_1


$X_1 \leq t_1$
$X_2 \leq t_2$
$X_1 \leq t_3$
$X_2 \leq t_4$
R_4
R_5

X_2
X_1

Overfitting

If we keep splitting we will be reducing RSS



Pruning

- Fewer splits or fewer regions lower variance better interpretation at cost of little more bias
- Ideas?
- Stop splitting when RSS improvement is lower than a threshold
    - Smaller trees but not effective (short sighted)
    - A split early on in the tree might be followed by a very good split; a split that leads to a large reduction in RSS later on

Pruning

- Better is to grow a large tree and then look subtrees that minimize the test error
- How?
- Cross-validation of all possible subtrees?
- This is too expensive
- Cost complexity pruning—also known as weakest link pruning

Cost complexity pruning

- Consider a tuning parameter α that for each value of α there is a subtree that minimizes

$$\sum_{m=1}^{|T|} \sum_{i:\, x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

  - a subtree is any tree that can be obtained by collapsing any number of its internal (non-terminal) nodes,
  - $|T|$ is the number of terminal nodes,
  - α controls the complexity of the tree similarly we saw with other regularizations (e.g. LASSO).

Cost complexity pruning (cont'd)

- It turns out that as we increase α from zero in, branches get pruned from the tree in a nested and predictable fashion, so obtaining the whole sequence of subtrees as a function of α is easy.

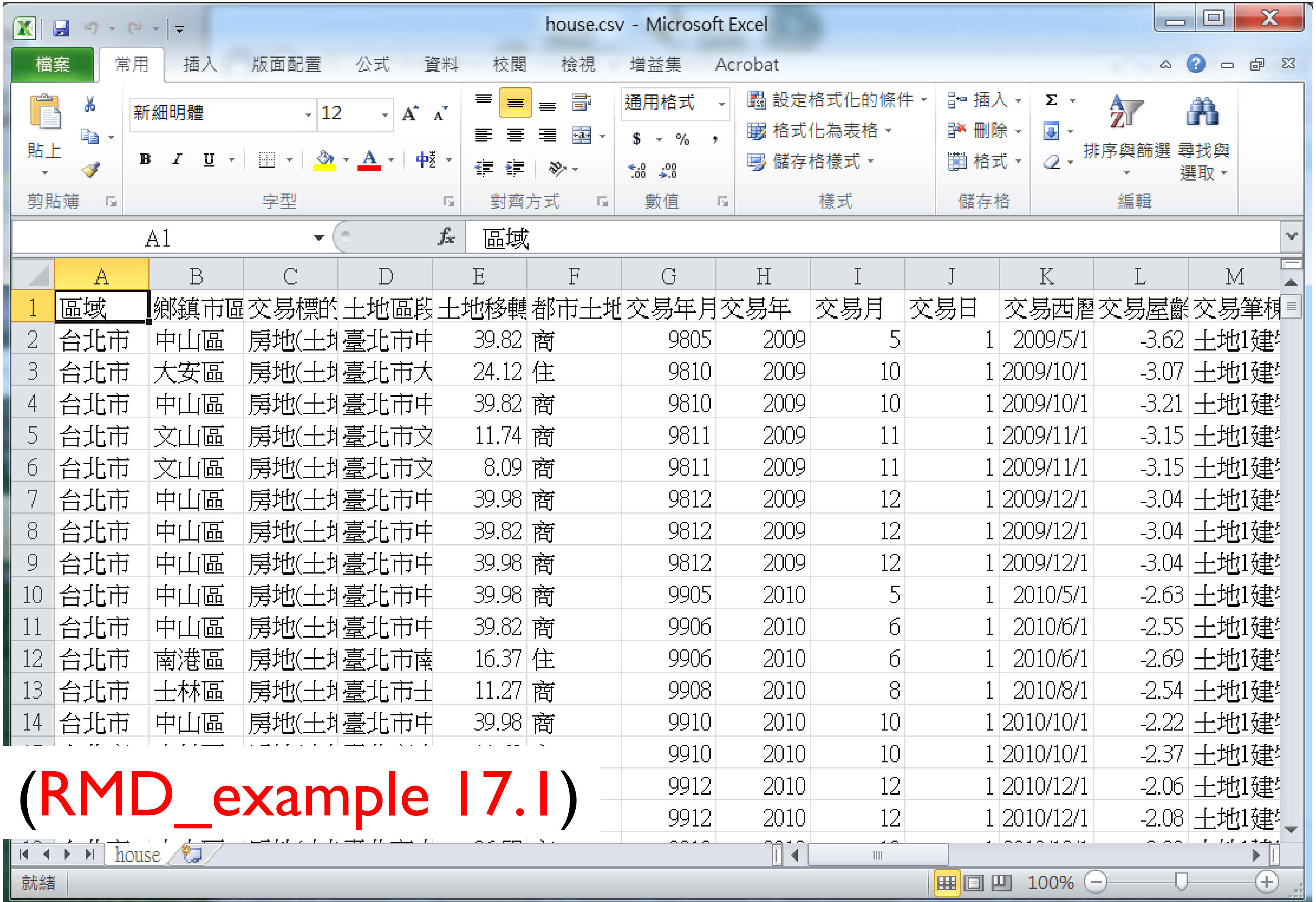
Algorithm for pruning

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α

Algorithm for pruning (cont'd)

3. Use *K*-fold cross-validation to choose α: for each α value
   - Repeat #1 and #2 on the data except the *k*-th fold
   - Estimate the MSE on the *k*-th fold
   - Average MSE over all folds
   - Pick α with the smallest average MSE
4. Return the subtree from Step 2 that corresponds to the chosen value of α

六都房地產實價登錄資料



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	區域	鄉鎮市區	交易標的	土地區段	土地移轉	都市土地	交易年月	交易年	交易月	交易日	交易西曆	交易屋齡	交易筆棟
2	台北市	中山區	房地(土地	臺北市中	39.82	商	9805	2009	5	1	2009/5/1	-3.62	土地1建
3	台北市	大安區	房地(土地	臺北市大	24.12	住	9810	2009	10	1	2009/10/1	-3.07	土地1建
4	台北市	中山區	房地(土地	臺北市中	39.82	商	9810	2009	10	1	2009/10/1	-3.21	土地1建
5	台北市	文山區	房地(土地	臺北市文	11.74	商	9811	2009	11	1	2009/11/1	-3.15	土地1建
6	台北市	文山區	房地(土地	臺北市文	8.09	商	9811	2009	11	1	2009/11/1	-3.15	土地1建
7	台北市	中山區	房地(土地	臺北市中	39.98	商	9812	2009	12	1	2009/12/1	-3.04	土地1建
8	台北市	中山區	房地(土地	臺北市中	39.82	商	9812	2009	12	1	2009/12/1	-3.04	土地1建
9	台北市	中山區	房地(土地	臺北市中	39.98	商	9812	2009	12	1	2009/12/1	-3.04	土地1建
10	台北市	中山區	房地(土地	臺北市中	39.98	商	9905	2010	5	1	2010/5/1	-2.63	土地1建
11	台北市	中山區	房地(土地	臺北市中	39.82	商	9906	2010	6	1	2010/6/1	-2.55	土地1建
12	台北市	南港區	房地(土地	臺北市南	16.37	住	9906	2010	6	1	2010/6/1	-2.69	土地1建
13	台北市	士林區	房地(土地	臺北市士	11.27	商	9908	2010	8	1	2010/8/1	-2.54	土地1建
14	台北市	中山區	房地(土地	臺北市中	39.98	商	9910	2010	10	1	2010/10/1	-2.22	土地1建
							9910	2010	10	1	2010/10/1	-2.37	土地1建
							9912	2010	12	1	2010/12/1	-2.06	土地1建
							9912	2010	12	1	2010/12/1	-2.08	土地1建

(RMD_example 17.1)

Variable	Description
每平方公尺單價	元
豪宅	0=每平方公尺單價 ≤ 20萬 1=每平方公尺單價 > 20萬
區域	台北市、新北市、桃園市、台中市、台南市、高雄市
車位	0=無, 1=有
屋齡	建築完成到2015/9/18 (年)
主要用途	工業用、住家用、住商用、商業用、國民住宅
建物型態	公寓(5樓含以下無電梯)、住宅大樓(11層含以上有電梯)、店面(店鋪)、套房(1房1廳1衛)、透天厝、華廈(10層含以下有電梯)、廠辦、辦公商業大樓
有無管理組織	0=無, 1=有

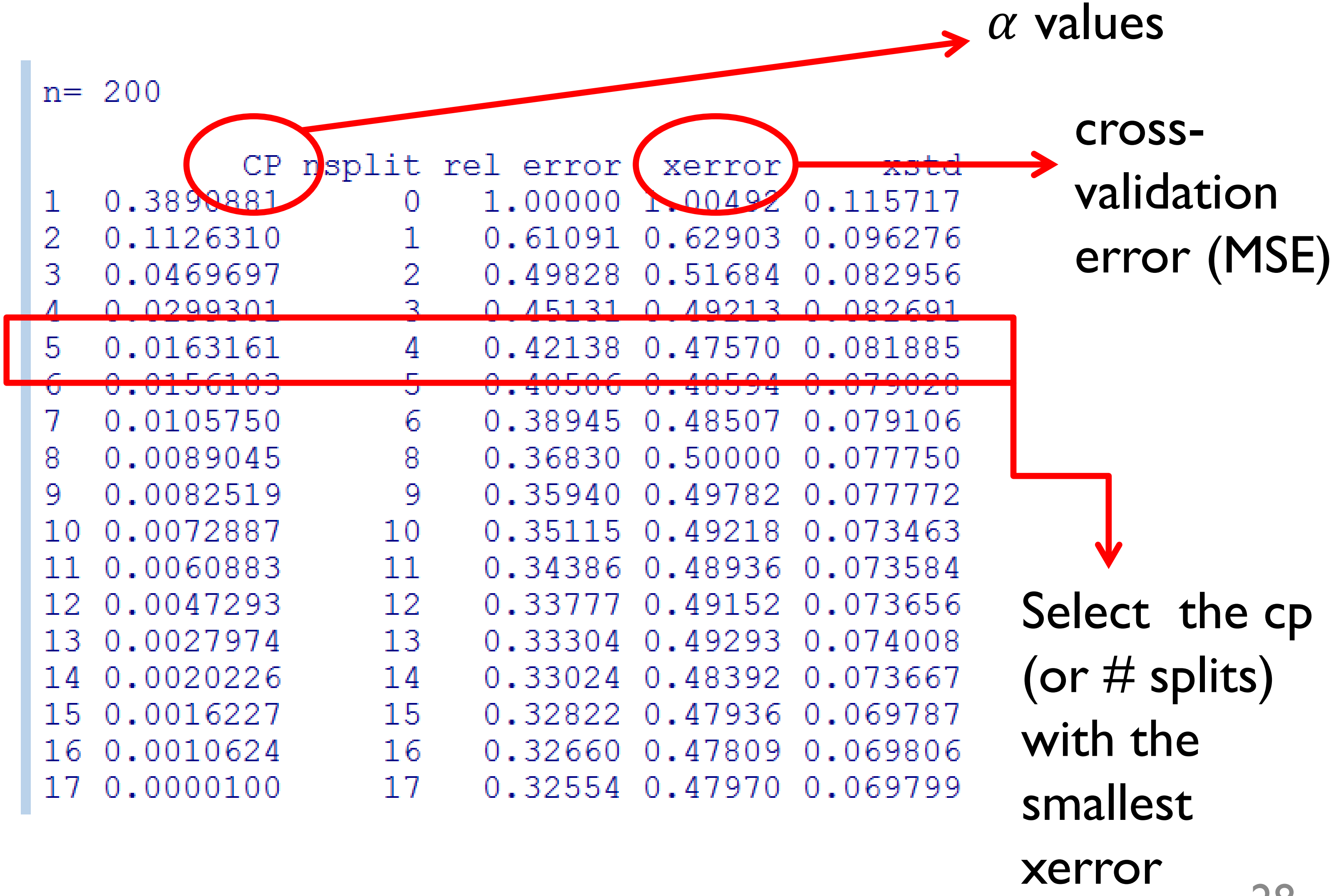
六都房地產實價登錄資料-regression tree

- Response variable: 每平方公尺平均單價
- Predictors:
  - 區域
  - 車位
  - 屋齡
  - 主要用途
  - 建物型態
  - 有無管理組織

(RMD_example 17.2)

- Note: log-transform每平方公尺平均單價so that its distribution has more of a typical bell-shape.

Cross-validation results for selecting complexity tuning parameter α



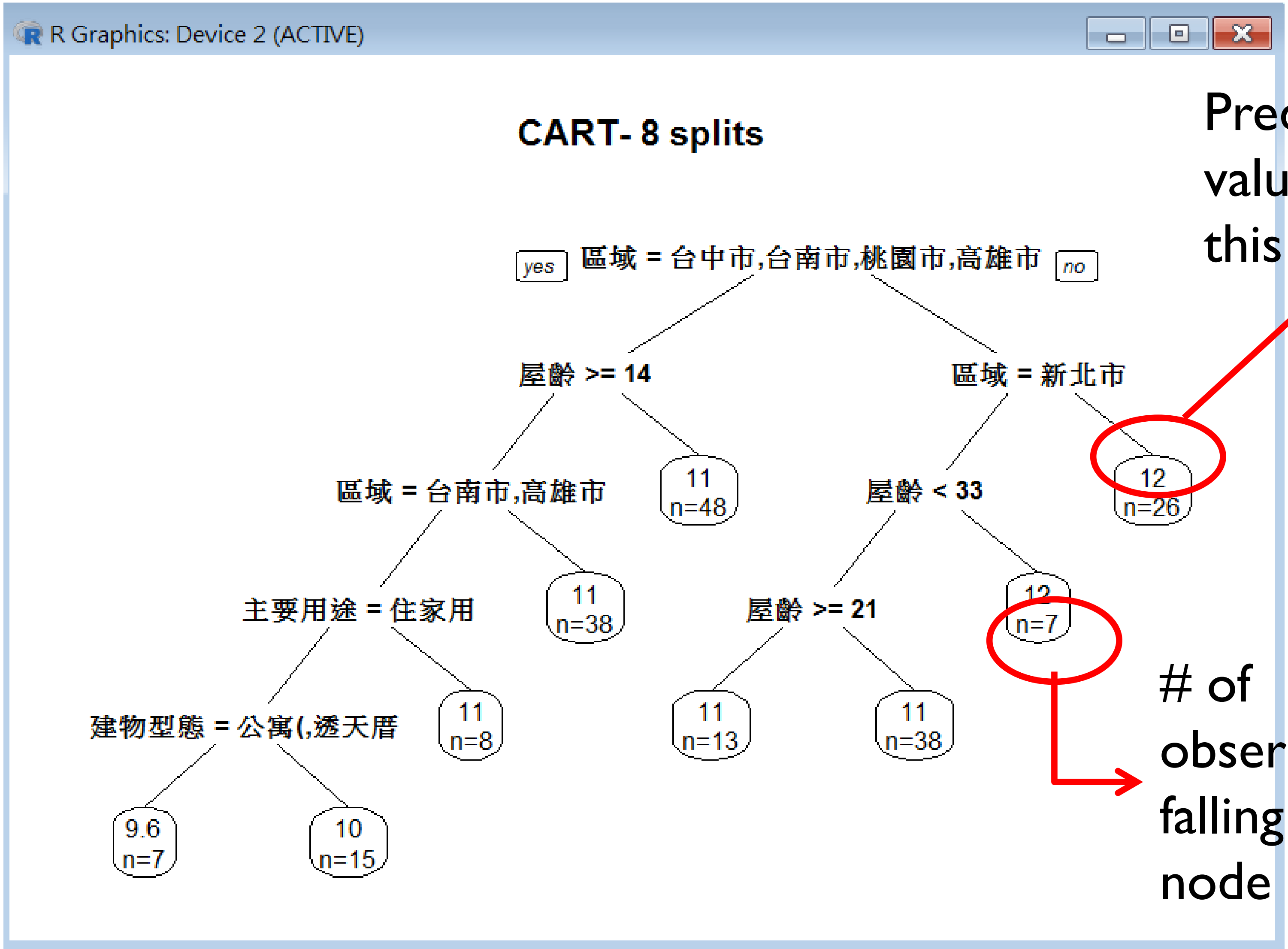
```
n= 200

         CP nsplit rel error  xerror     xstd
1  0.3890881      0   1.00000 1.00492 0.115717
2  0.1126310      1   0.61091 0.62903 0.096276
3  0.0469697      2   0.49828 0.51684 0.082956
4  0.0299301      3   0.45131 0.49213 0.082691
5  0.0163161      4   0.42138 0.47570 0.081885
6  0.0156103      5   0.40506 0.48594 0.079020
7  0.0105750      6   0.38945 0.48507 0.079106
8  0.0089045      8   0.36830 0.50000 0.077750
9  0.0082519      9   0.35940 0.49782 0.077772
10 0.0072887     10   0.35115 0.49218 0.073463
11 0.0060883     11   0.34386 0.48936 0.073584
12 0.0047293     12   0.33777 0.49152 0.073656
13 0.0027974     13   0.33304 0.49293 0.074008
14 0.0020226     14   0.33024 0.48392 0.073667
15 0.0016227     15   0.32822 0.47936 0.069787
16 0.0010624     16   0.32660 0.47809 0.069806
17 0.0000100     17   0.32554 0.47970 0.069799
```

α values

cross-validation error (MSE)

Select the cp (or # splits) with the smallest xerror

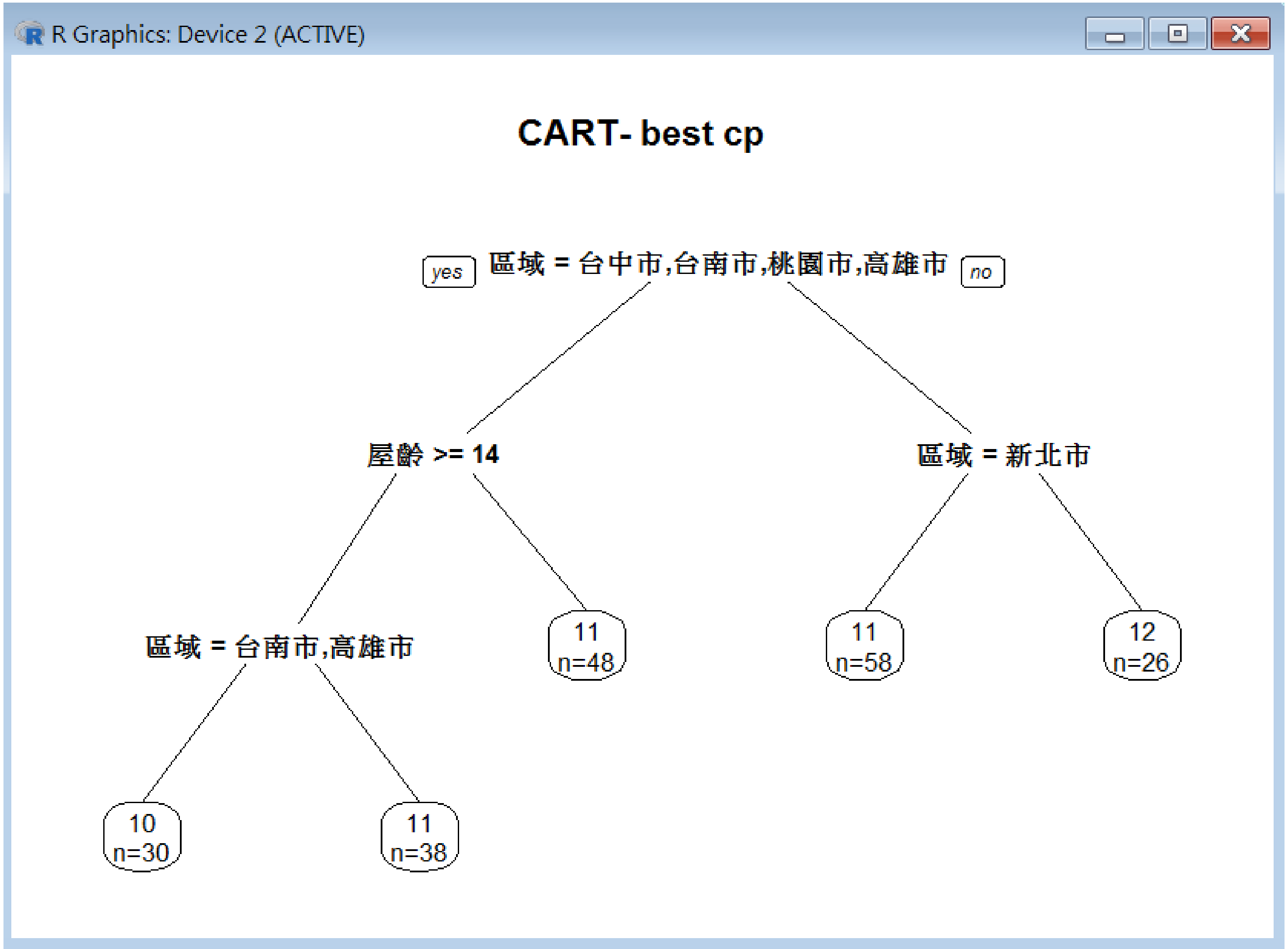
Prune the tree to 8 splits



Predicted value of this node

# of observations falling in this node

Prune the tree using the best cp



Classification trees

- Very similar to regression except that it is used to predict a qualitative response rather than a quantitative one
- In regression trees, we use the mean response of the training observations. For classification trees, we use most commonly occurring class.
- Interested in the class proportions of each region

Classification trees

We learn the model using recursive binary splitting as with the regression trees except …

- RSS cannot be used as a criterion for making the binary splits.
- Classification error rate:

$$E = 1 - \max_k \hat{p}_{mk}$$

  - $\hat{p}_{mk}$ represents the proportion of training observations in the *m*-th region that are from the *k*-th class
  - We classify the observations in the *m*-th region to the class with the biggest $\hat{p}_{mk}$

Gini index

- Classification error is not differentiable or sensitive enough for tree growing.
- Purity of the nodes, Gini index

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

- G takes small values when p_{mk} is small or close to 1, therefore is a measure of purity of the nodes.

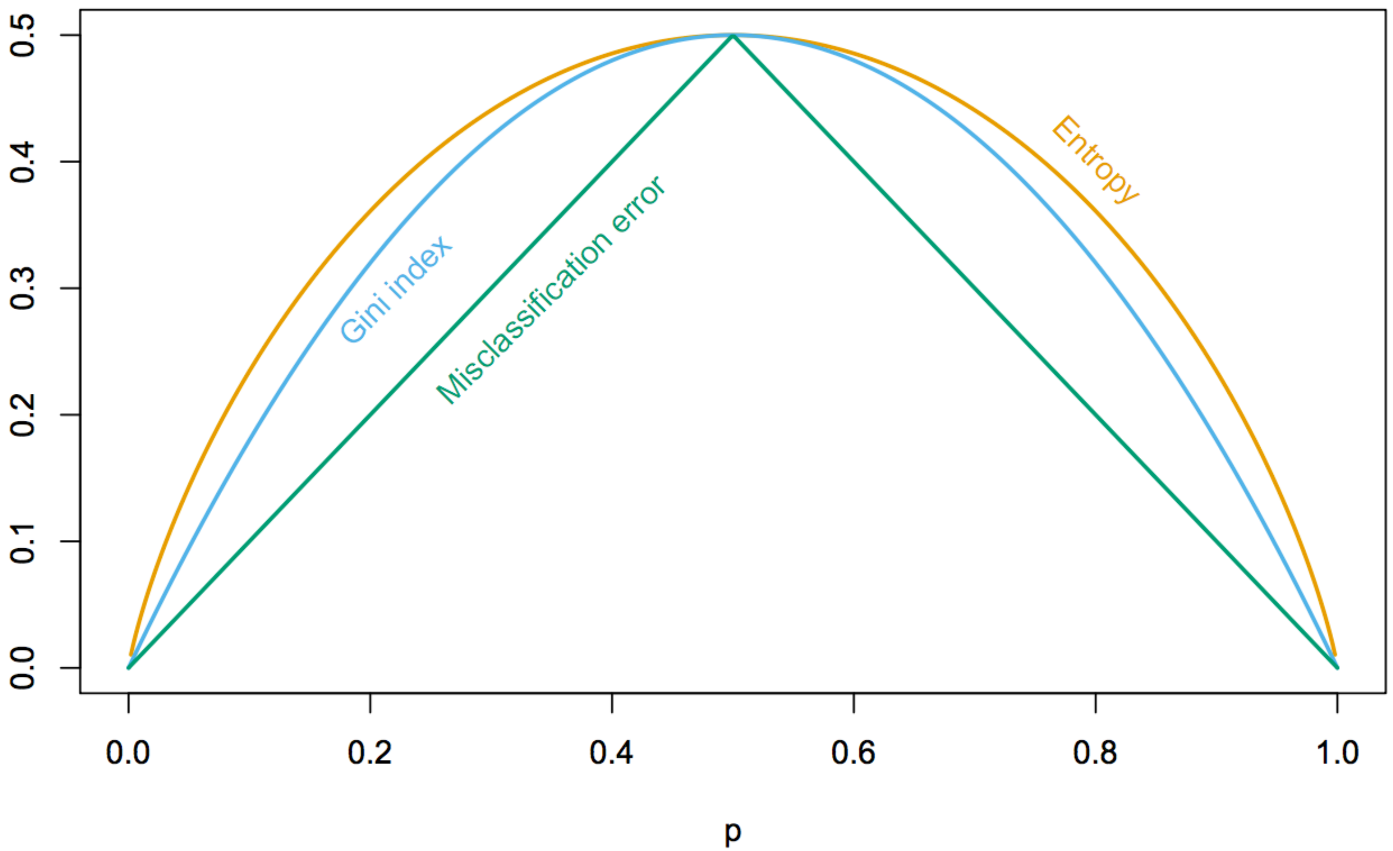
Cross entropy

- Alternative to the Gini index is cross entropy

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk})$$

- $D > 0$ and will take value near zero when p_{mk} is either near zero or one

Node impurity for two class problem



Hastie et al.,"The Elements of Statistical Learning: Data Mining, Inference, and Prediction", Springer (2009)

Pruning classification tree

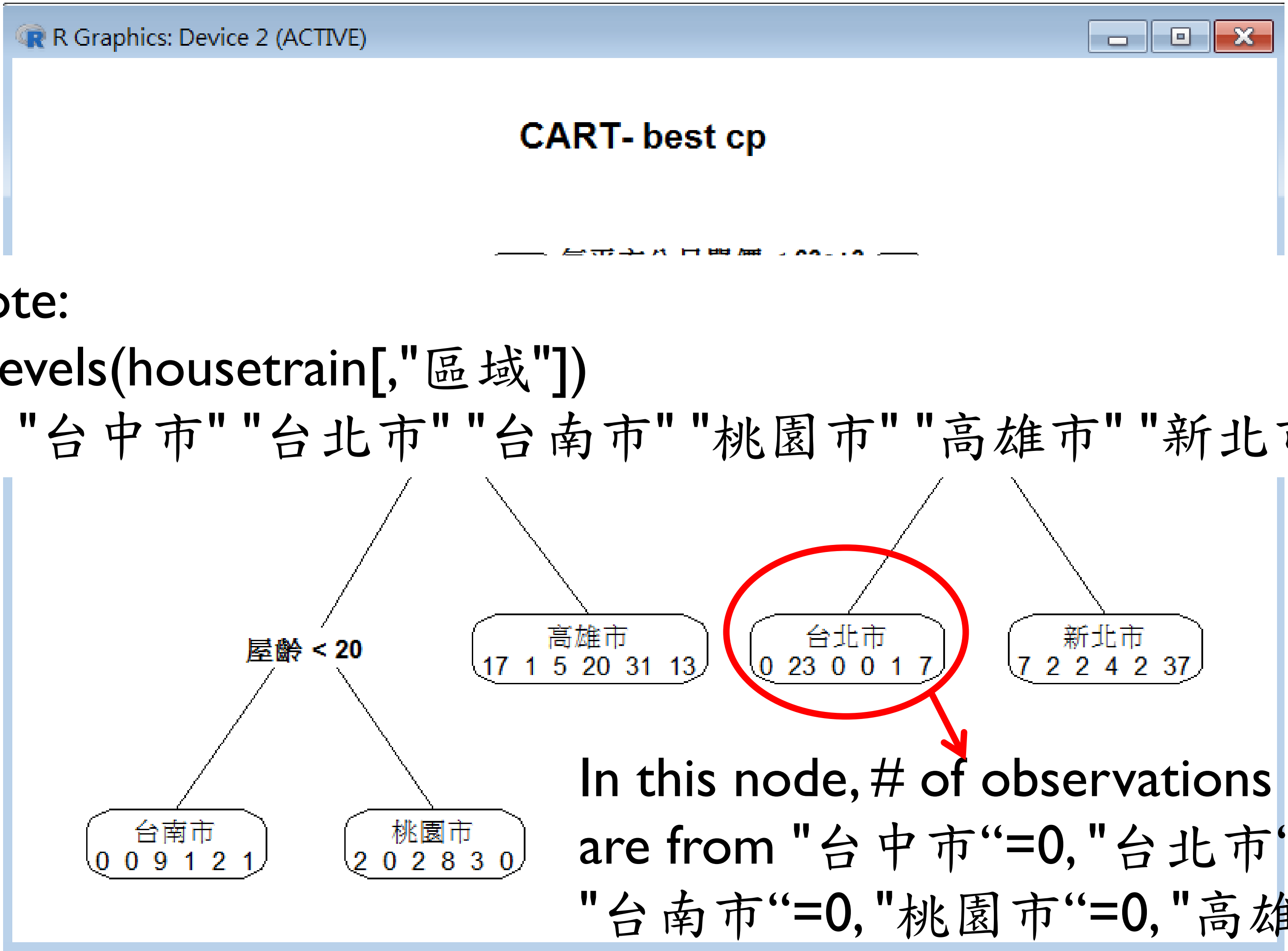
- Use the same algorithm as for regression tree but instead of RSS use Gini index or Entropy
- HOWEVER: classification error rate is preferable for the final pruned tree

六都房地產實價登錄資料-classification tree

- Response variable: 區域
- Predictors:
  - 每平方公尺平均單價
  - 車位
  - 屋齡
  - 主要用途
  - 建物型態
  - 有無管理組織

(RMD_example 17.3)

Prune the tree using the best cp



Note:

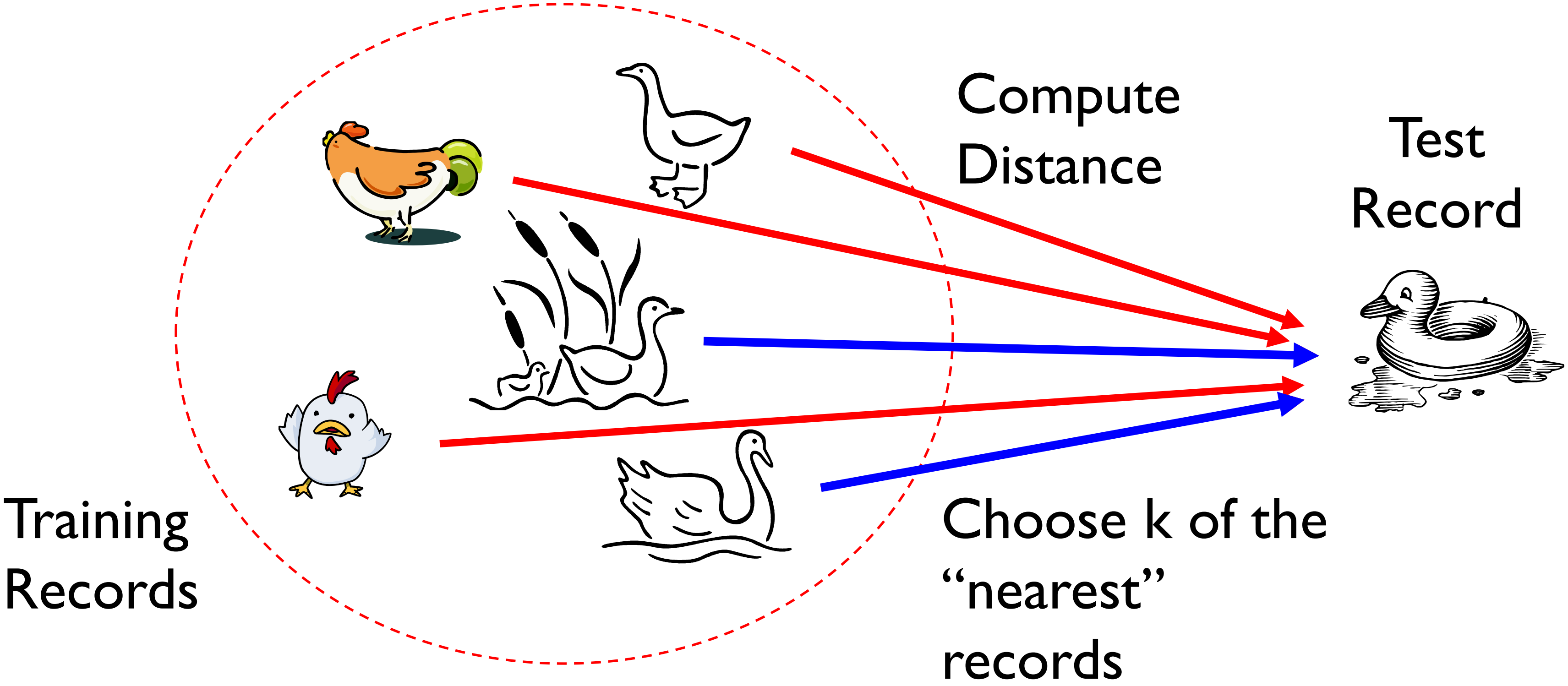
```
> levels(housetrain[,"區域"])
[1] "台中市" "台北市" "台南市" "桃園市" "高雄市" "新北市"
```

In this node, # of observations that are from "台中市“=0, "台北市“=23, "台南市“=0, "桃園市“=0, "高雄市“=1, "新北市“=7

K-nearest neighbor classifier

Nearest neighbor classifiers

- Basic idea: If it walks like a duck, quacks like a duck, then it's probably a duck



Compute Distance

Test Record

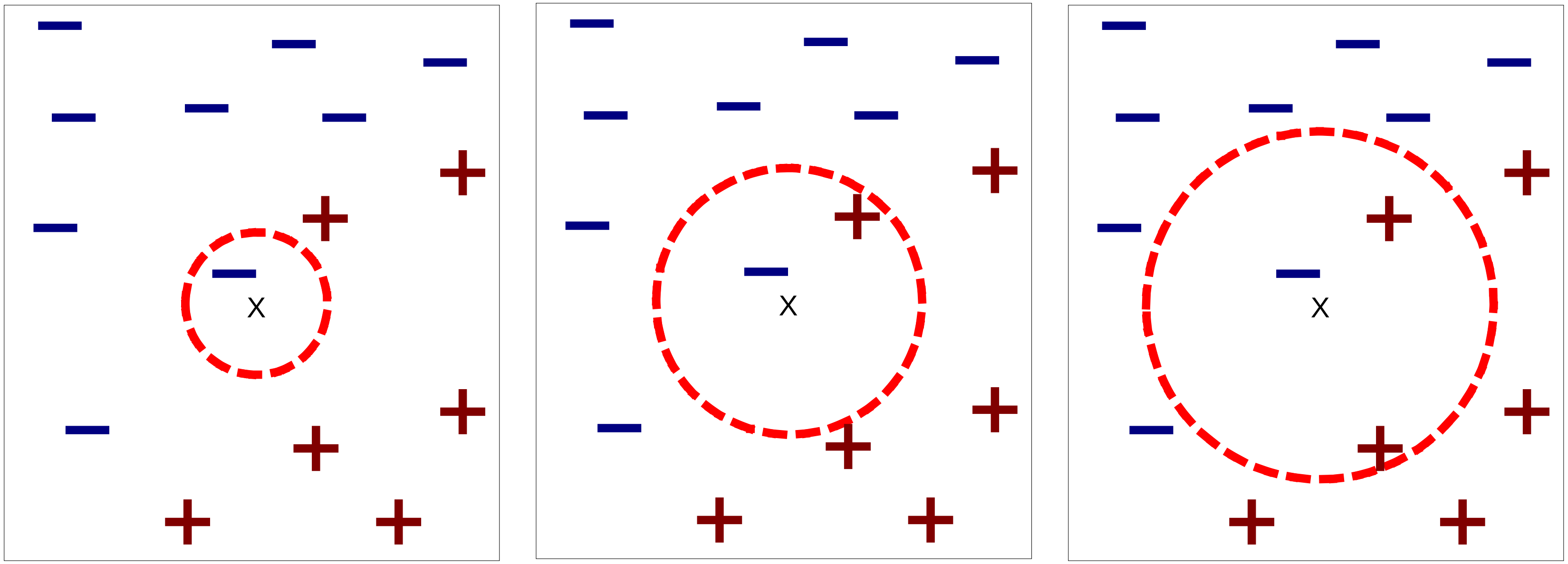
Training Records

Choose k of the "nearest" records

Nearest neighbor classifiers

- Requires three things
  - The set of stored records
  - Distance metric to compute distance between records
  - The value of *K*, the number of nearest neighbors to retrieve
- To classify an unknown record:
  - Compute distance to other training records
  - Identify *K* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)
- The number of neighbors *K* can be chosen by cross-validation.

Definition of nearest neighbor



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record ***x*** are data points that have the *K* smallest distance to x

Missing values

- Some clustering and class prediction methods require complete data, i.e., measures for all variables in all samples.
- However, missing data are a common problem in many experiments.
- A simple, intuitive imputation approach is *K*-nearest neighbor imputation.

K-nearest neighbor imputation

- First computes a matrix of pairwise distances between variables, by ignoring the missing values.
- Then, for imputing the measure X(i, g) of variable g in sample i, one looks for the *K* nearest neighbors of variable g having data for sample i.
- One can impute X(i, g) by the average of the measures of these k neighboring variables in sample i.

六都房地產實價登錄資料- K-nearest neighbor classifier

- Response variable: 區域
- Predictors:
  - 每平方公尺平均單價
  - 屋齡

(RMD_example 17.4)