Big Data Analytics

Session 6

Decision Trees

Where were we last week

- To assess the model accuracy, the measure of fit is
  - Test MSE for regression
  - Test error rate for classification
- Bias and Variance tradeoff
  - Generally,
    - The more flexible a method is, the less bias it will generally have.
    - The more flexible a method is, the more variance it has.
- Cross Validation
  - Estimate the test MSE/error rate in the absence of the designated test data
  - Compare different models and select the best one
    - Validation set approach, LOOCV and k-fold CV
  - Once the best model is selected
    - Use the whole dataset to train a model
    - Make prediction using this model

Outline

- The Basics of Decision Trees
  - Regression Trees
  - Classification Trees
  - Pruning Trees
  - Trees vs. Linear Models
  - Advantages and Disadvantages of Trees

An Example

Grade distinctions at postgraduate level

At postgraduate taught level (PGCert, PGDip and Master's degrees), you may be awarded one of the following:

- **Distinction**: You will be awarded a Distinction if you achieve an average result of **70% or above** in modules at Level 7(M) as well as a distinction mark in the dissertation.
- **Merit**: You will be awarded a Merit if you achieve an average result of between 60% and 69% in modules at credit level 7(M).
- **Pass**: You will be awarded a Pass if you achieve an average result of between 50% and 59% in modules at credit level 7(M).
- **Fail**: You will be considered to have failed if you achieve an average result of below 50% in modules at credit level 7(M).

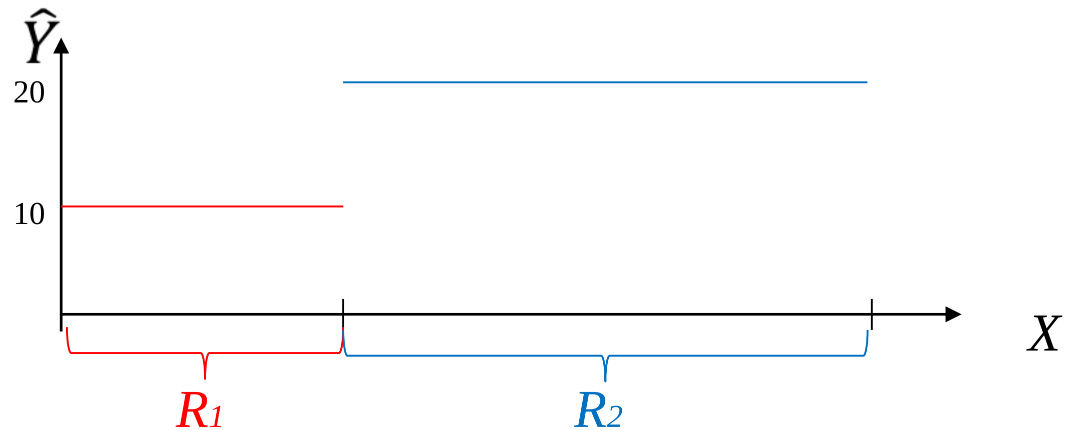
Partitioning Up the Predictor Space

- In the example:
  - X: Average marks in Level 7 modules
    - ranging from 0 to 100 (predictor space)
  - Y: Grades
    - ranging from Fail, Pass, Merit, Distinction

  - Divide [0,100] into four regions
    - R_1: [0,50) – Fail
    - R_2: [50,60) – Pass
    - R_3: [60,70) – Merit
    - R_4: [70,100] – Distinction

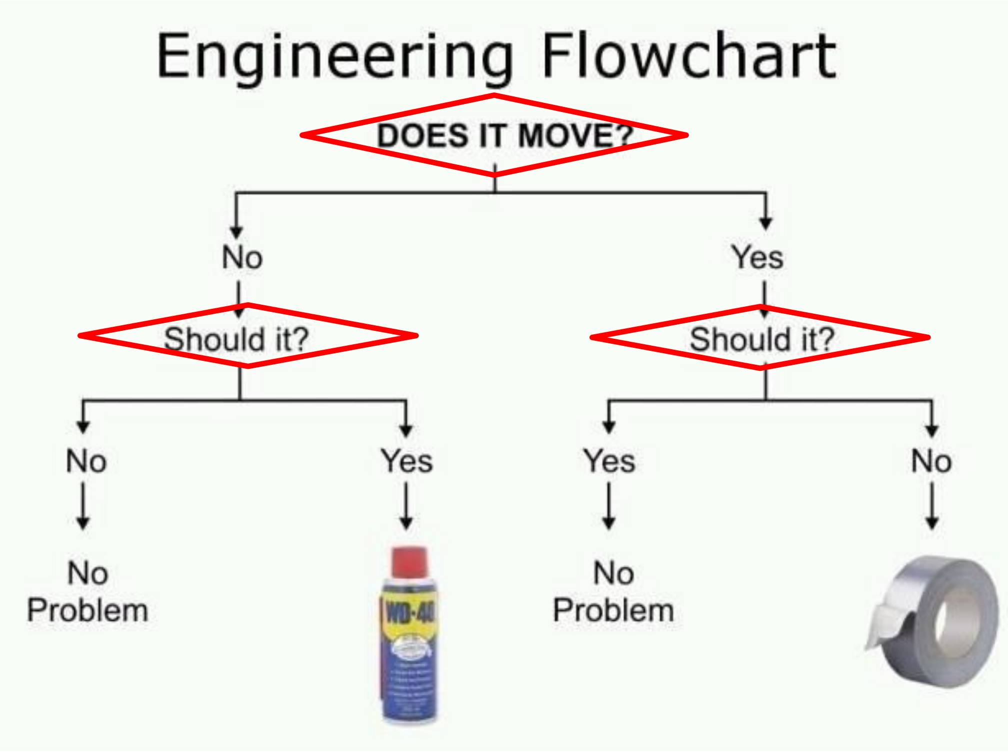
Partitioning Up the Predictor Space

- One way to make predictions in a regression problem is to divide the predictor space (i.e. all the possible values for $X_1, X_2, \ldots, X_p$) into distinct regions, say $R_1, R_2, \ldots, R_k$
  - Suppose for example we have two regions R_1 and R_2 with $\hat{Y}_1 = 10, \hat{Y}_2 = 20$
- Then for every X that falls in a particular region (say R_j) we make the same prediction.
  - Then for any value of X such that $X \in R_1$ we would predict 10, otherwise if $X \in R_2$ we would predict 20.



Decision Tree

- Decision tree
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution



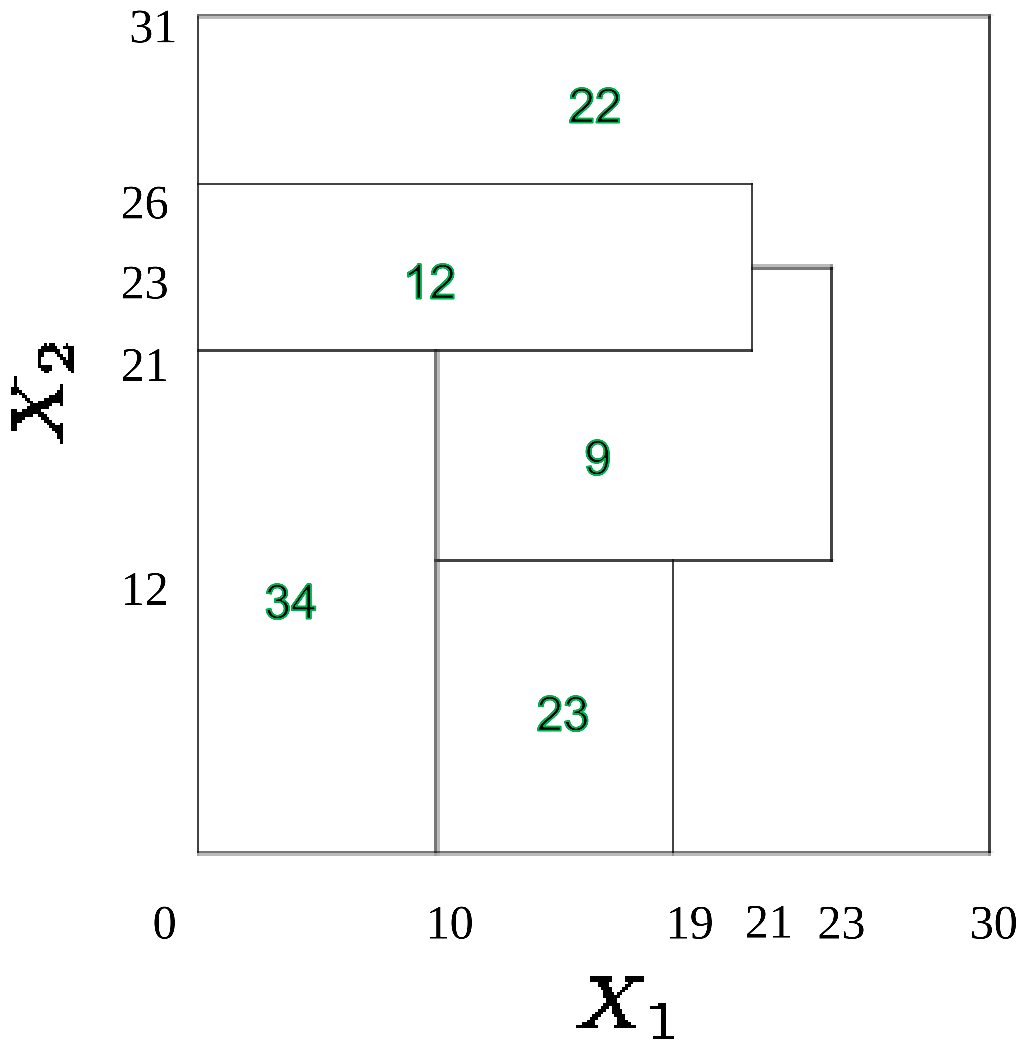
Regression Trees

Predicting a quantitative response

e.g., predicting baseball players' salary

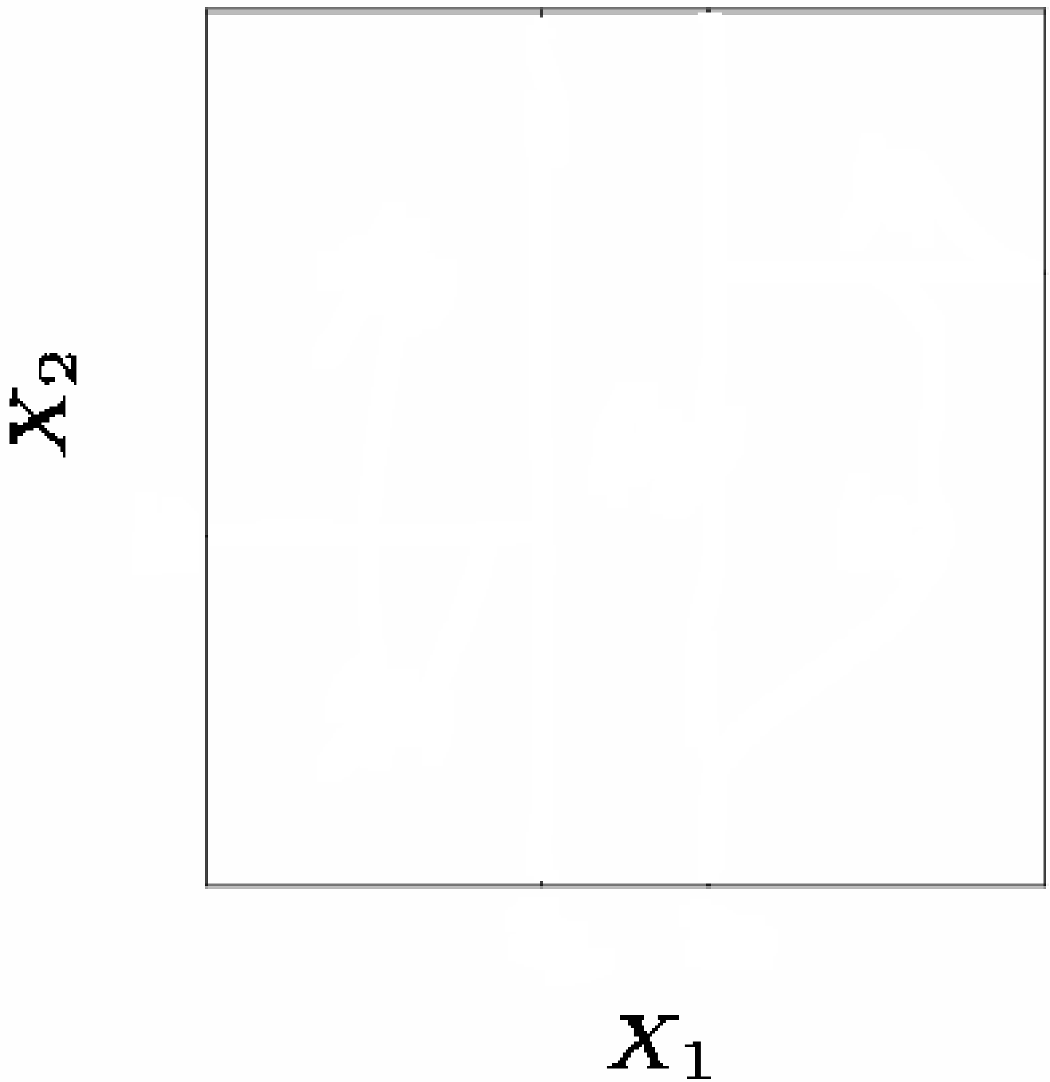
The General View

- Here we have two predictors and five distinct regions
- Depending on which region our new X comes from we would make one of five possible predictions for Y
- Predict Y based on
  - $X_1 = 15, X_2 = 15$
  - $X_1 = 28, X_2 = 24$
  - $X_1 = 5, X_2 = 29$
  - $X_1 = 22, X_2 = 25$



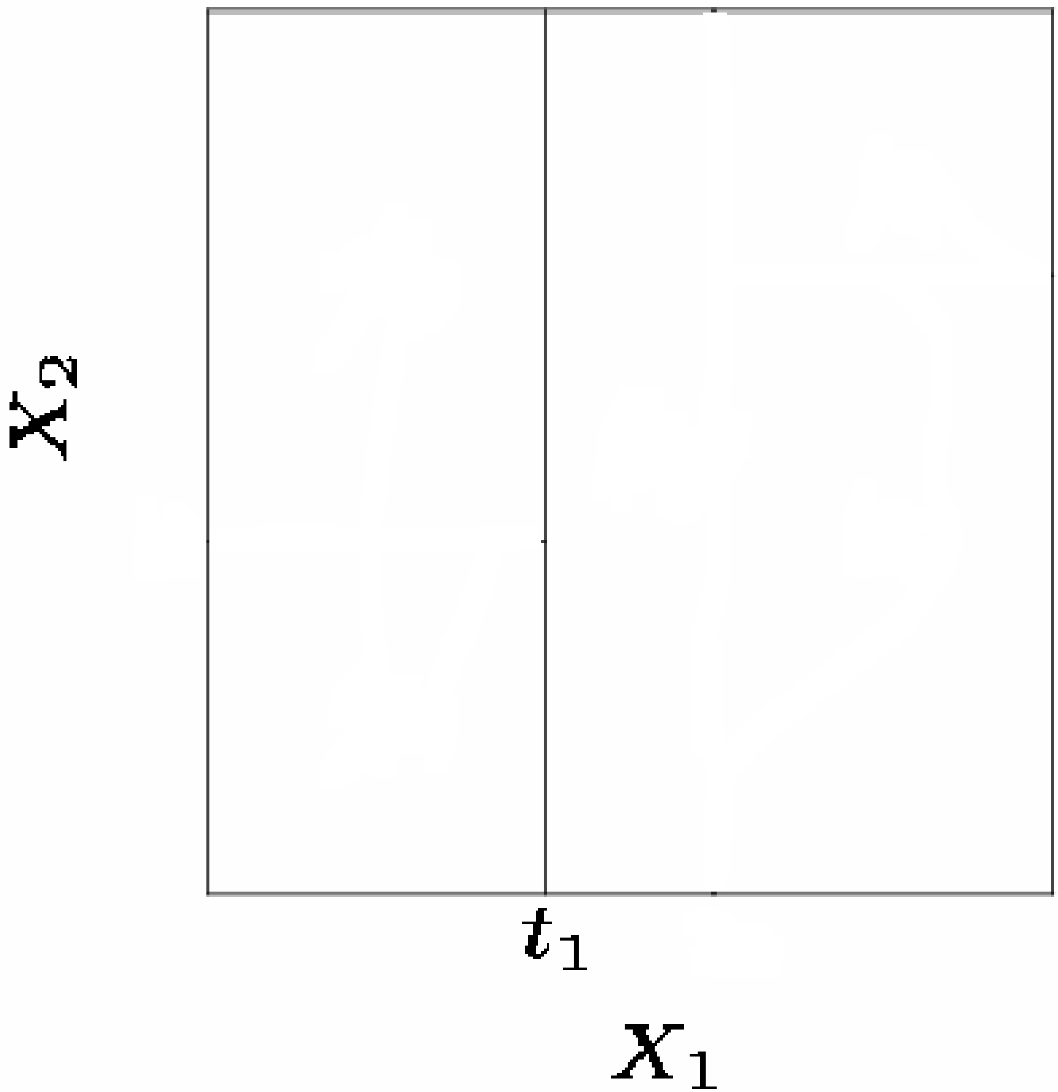
Splitting the X Variables

- Generally we create the partitions by iteratively splitting one of the X variables into two regions



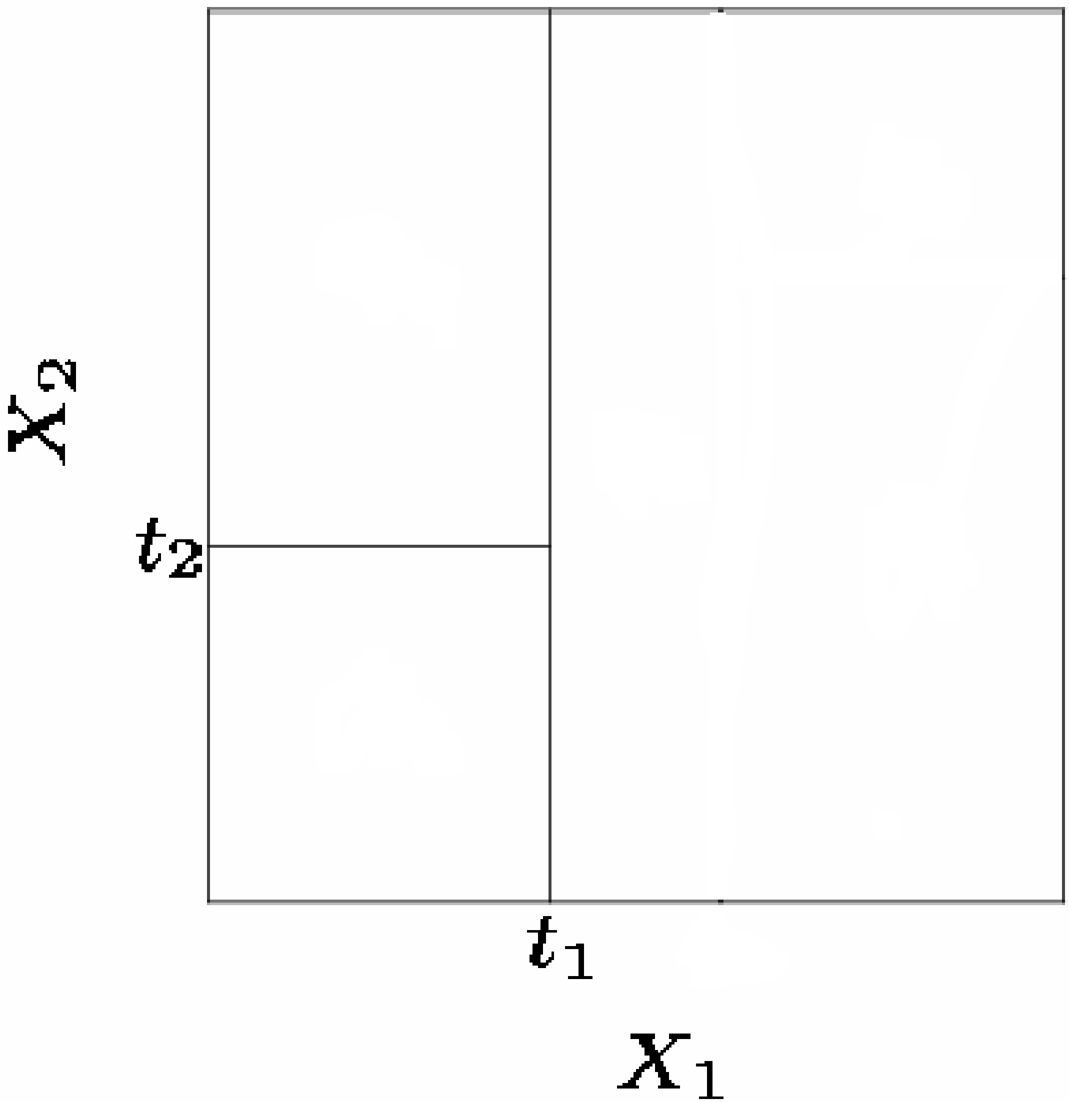
Splitting the X Variable

1. First split on $X_1 = t_1$



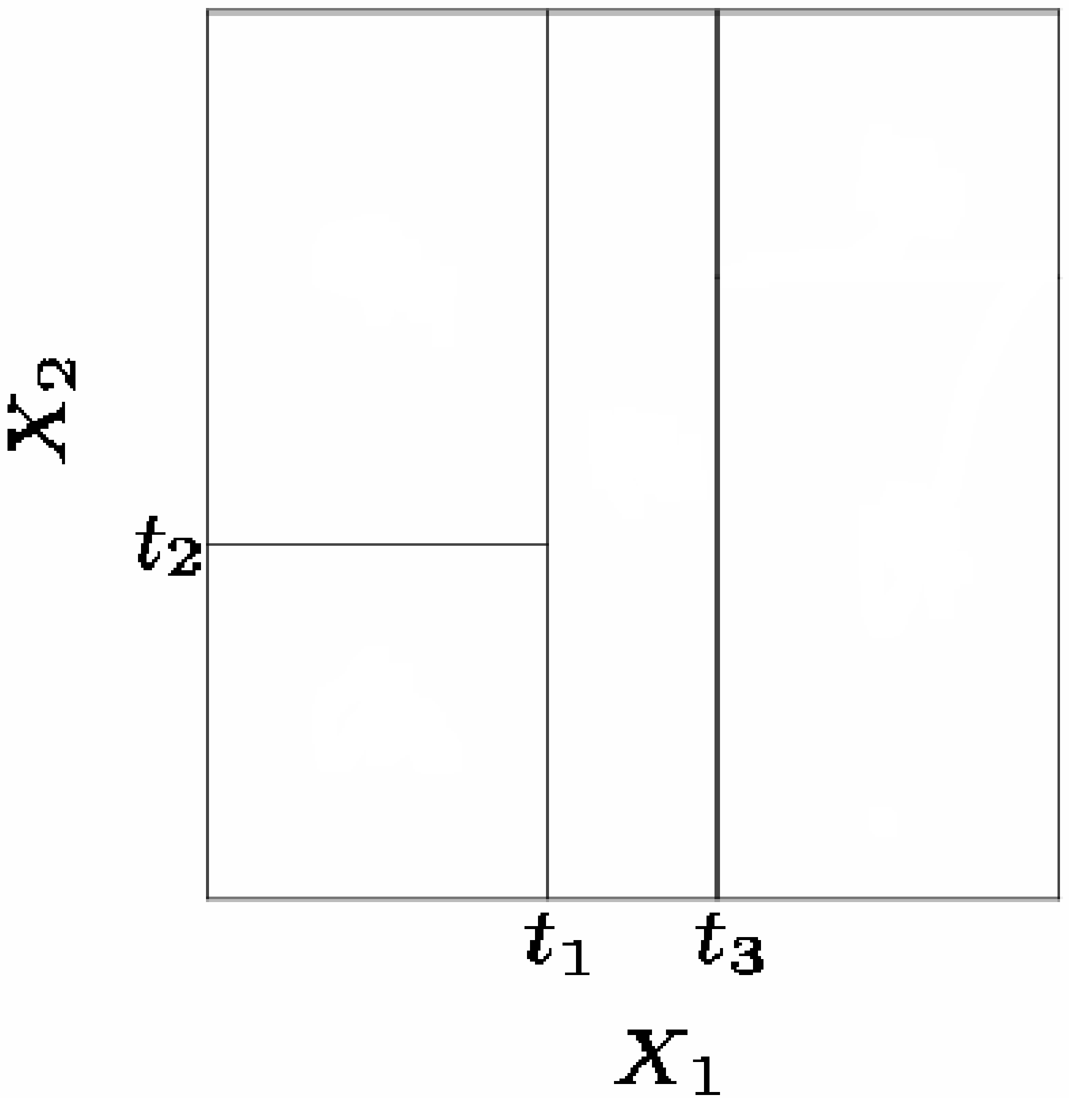
Splitting the X Variable

1. First split on $X_1 = t_1$
2. If $X_1 < t_1$, split on $X_2 = t_2$



Splitting the X Variable

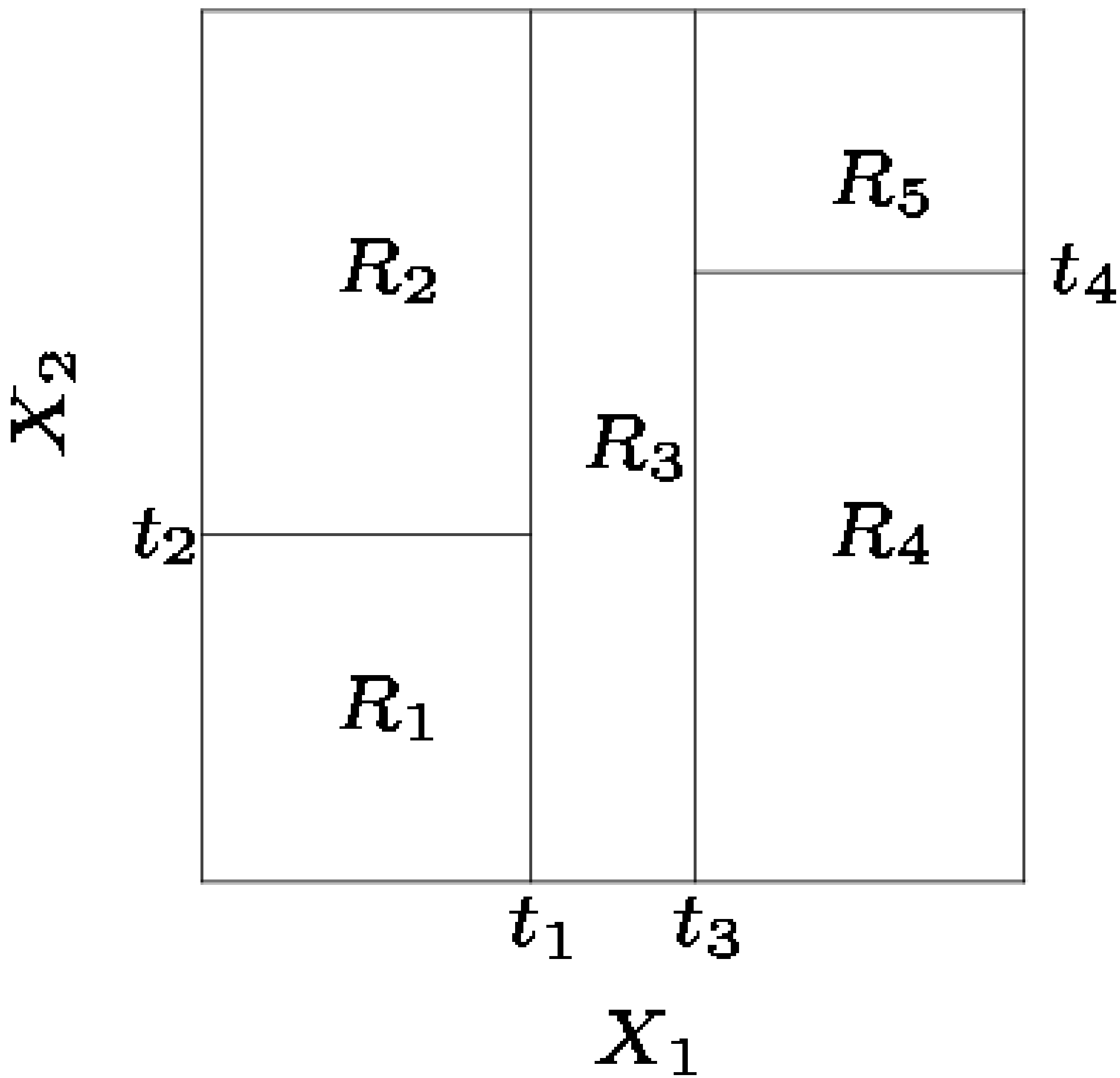
1. First split on $X_1 = t_1$
2. If $X_1 < t_1$, split on $X_2 = t_2$
3. If $X_1 > t_1$, split on $X_1 = t_3$



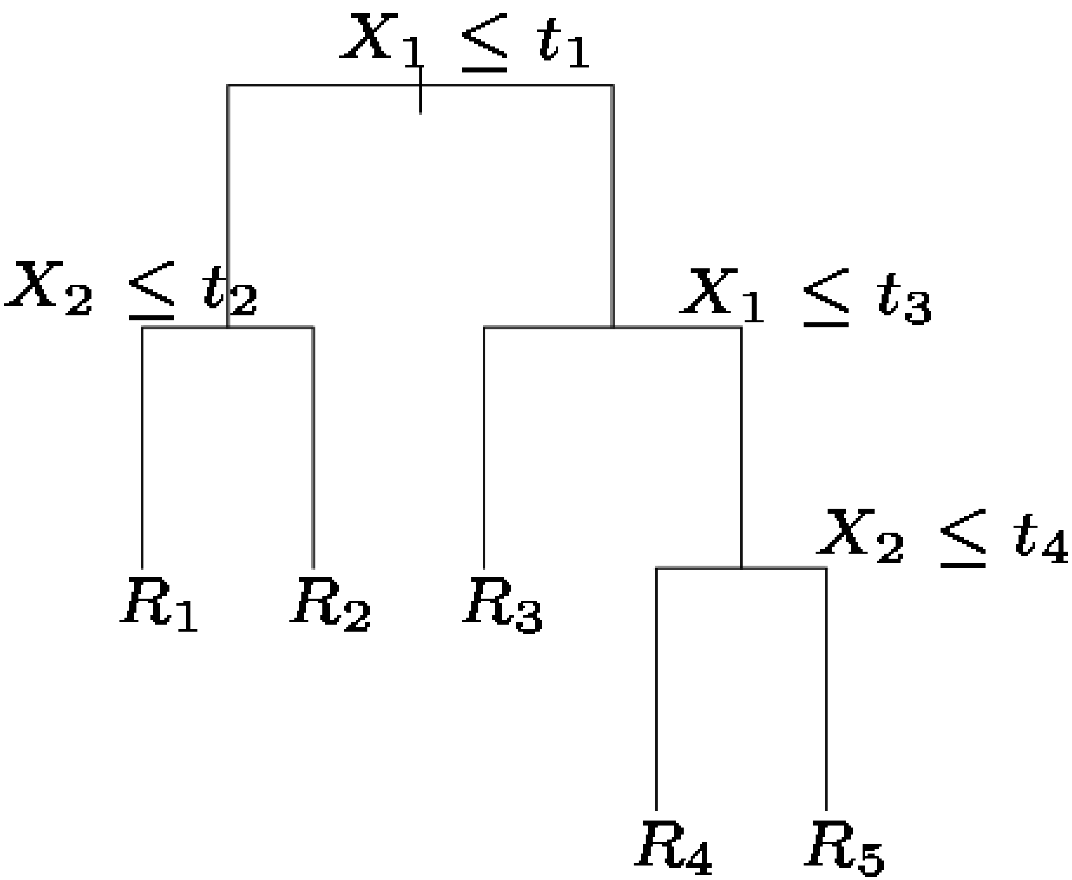
Splitting the X Variable

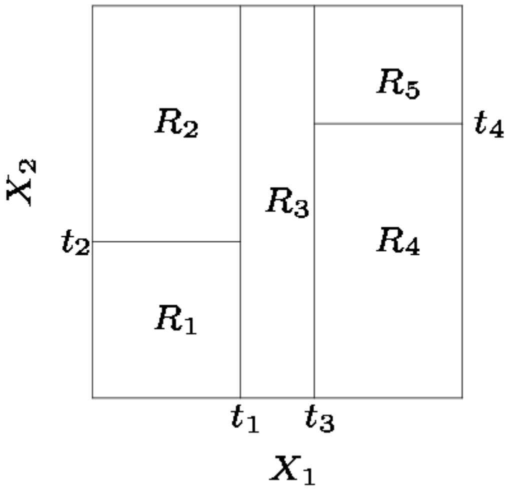
Birkbeck UNIVERSITY OF LONDON

1. First split on $X_1 = t_1$
2. If $X_1 < t_1$, split on $X_2 = t_2$
3. If $X_1 > t_1$, split on $X_1 = t_3$
4. If $X_1 > t_3$, split on $X_2 = t_4$



Splitting the X Variable





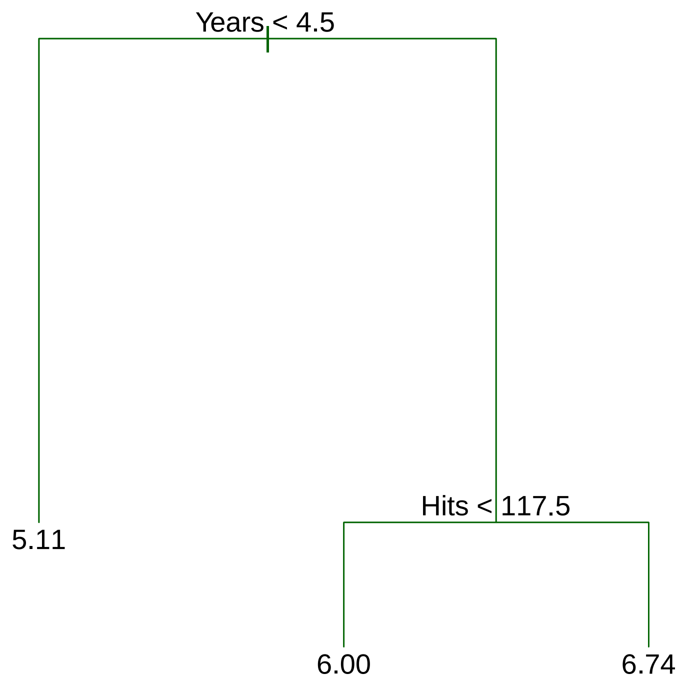
- When we create partitions this way we can always represent them using a tree structure.
- This provides a very simple way to explain the model to a non-expert e.g. your boss!

Example: Baseball Players' Salaries

- To predict baseball player's salaries by regression tree based on
  - Years
  - Hits
- Note that in the dataset, Salary is measured in 1000s. Here, Salary is then log-transformed (Why?)
- The predicted salary for a player who played in the league for more than 4.5 years and had less than 117.5 hits last year is

$$\$1000 \times e^{6.00} = \$402{,}834$$

Hitters is a dataset in the ISLR



Hitters Dataset

	AtBat	Hits	HmRun	Runs	RBI	Walks	Years	CAtBat	CHits	CHmRun	CRuns	CRBI	CWalks	League	Division	PutOuts	Assists	Errors	Salary	NewLeague
-Andy Allanson	293	66	1	30	29	14	1	293	66	1	30	29	14	A	E	446	33	20	NA	A
-Alan Ashby	315	81	7	24	38	39	14	3449	835	69	321	414	375	N	W	632	43	10	475.000	N
-Alvin Davis	479	130	18	66	72	76	3	1624	457	63	224	266	263	A	W	880	82	14	480.000	A
-Andre Dawson	496	141	20	65	78	37	11	5628	1575	225	828	838	354	N	E	200	11	3	500.000	N
-Andres Galarraga	321	87	10	39	42	30	2	396	101	12	48	46	33	N	E	805	40	4	91.500	N
-Alfredo Griffin	594	169	4	74	51	35	11	4408	1133	19	501	336	194	A	W	282	421	25	750.000	A
-Al Newman	185	37	1	23	8	21	2	214	42	1	30	9	24	N	E	76	127	7	70.000	A
-Argenis Salazar	298	73	0	24	24	7	3	509	108	0	41	37	12	A	W	121	283	9	100.000	A
-Andres Thomas	323	81	6	26	32	8	2	341	86	6	32	34	8	N	W	143	290	19	75.000	N
-Andre Thornton	401	92	17	49	66	65	13	5206	1332	253	784	890	866	A	E	0	0	0	1100.000	A
-Alan Trammell	574	159	21	107	75	59	10	4631	1300	90	702	504	488	A	E	238	445	22	517.143	A
-Alex Trevino	202	53	4	31	26	27	9	1876	467	15	192	186	161	N	W	304	45	11	512.500	N
-Andy VanSlyke	418	113	13	48	61	47	4	1512	392	41	205	204	203	N	E	211	11	7	550.000	N
-Alan Wiggins	239	60	0	30	11	22	6	1941	510	4	309	103	207	A	E	121	151	6	700.000	A
-Bill Almon	196	43	7	29	27	30	13	3231	825	36	376	290	238	N	E	80	45	8	240.000	N
-Billy Beane	183	39	3	20	15	11	3	201	42	3	20	16	11	A	W	118	0	0	NA	A
-Buddy Bell	568	158	20	89	75	73	15	8068	2273	177	1045	993	732	N	W	105	290	10	775.000	N
-Buddy Biancalana	190	46	2	24	8	15	5	479	102	5	65	23	39	A	W	102	177	16	175.000	A
-Bruce Bochte	407	104	6	57	43	65	12	5233	1478	100	643	658	653	A	W	912	88	9	NA	A

summary(Hitters$Salary)

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
67.5	190.0	425.0	535.9	750.0	2460.0	59

summary(log(Hitters$Salary))

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
4.212	5.247	6.052	5.927	6.620	7.808	59

Example: Baseball Players' Salaries

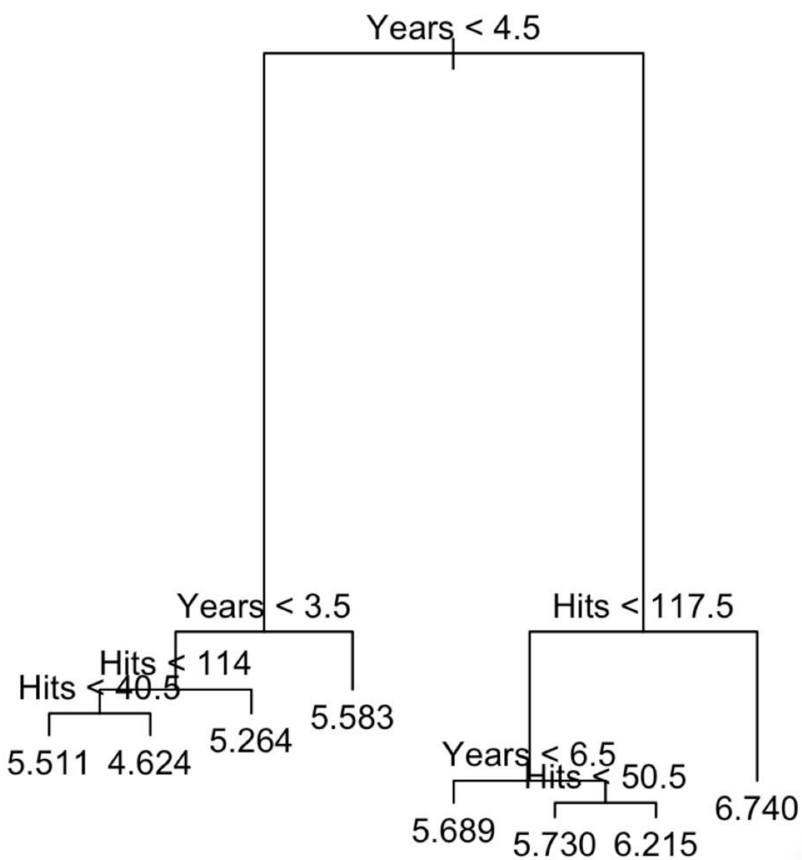
Can you 1) build a regression tree using Years and Hits and
2) make the prediction for a player with Years = 5 and Hits = 100 using R?

- Step 1: Building the tree:

```
library(tree)
nrow(Hitters)
[1] 322

Hitters <- na.omit(Hitters)   #remove rows with missing observations
nrow(Hitters)
[1] 263

tree.hitters <- tree(log(Hitters$Salary)~Years+Hits, Hitters)
# type in tree.hitters or summary(tree.hitters) to see more details

plot(tree.hitters)            #Why is this tree different from the one before?
text(tree.hitters)
```

Example: Baseball Players' Salaries

```
> tree.hitters
node), split, n, deviance, yval
      * denotes terminal node

 1) root 263 207.200 5.927
   2) Years < 4.5 90  42.350 5.107
     4) Years < 3.5 62  23.010 4.892
       8) Hits < 114 43  17.150 4.727
        16) Hits < 40.5 5  10.400 5.511 *
        17) Hits > 40.5 38   3.280 4.624 *
       9) Hits > 114 19   2.069 5.264 *
     5) Years > 3.5 28  10.130 5.583 *
   3) Years > 4.5 173  72.710 6.354
     6) Hits < 117.5 90  28.090 5.998
      12) Years < 6.5 26   7.238 5.689 *
      13) Years > 6.5 64  17.350 6.124
        26) Hits < 50.5 12   2.689 5.730 *
        27) Hits > 50.5 52  12.370 6.215 *
     7) Hits > 117.5 83  20.880 6.740 *
```



n: the number of observations in that branch
yval: the overall prediction for the branch
deviance: node residual sums of squares summed over the terminal nodes of the tree

Example: Baseball Players' Salaries

- Step 2: Making predictions

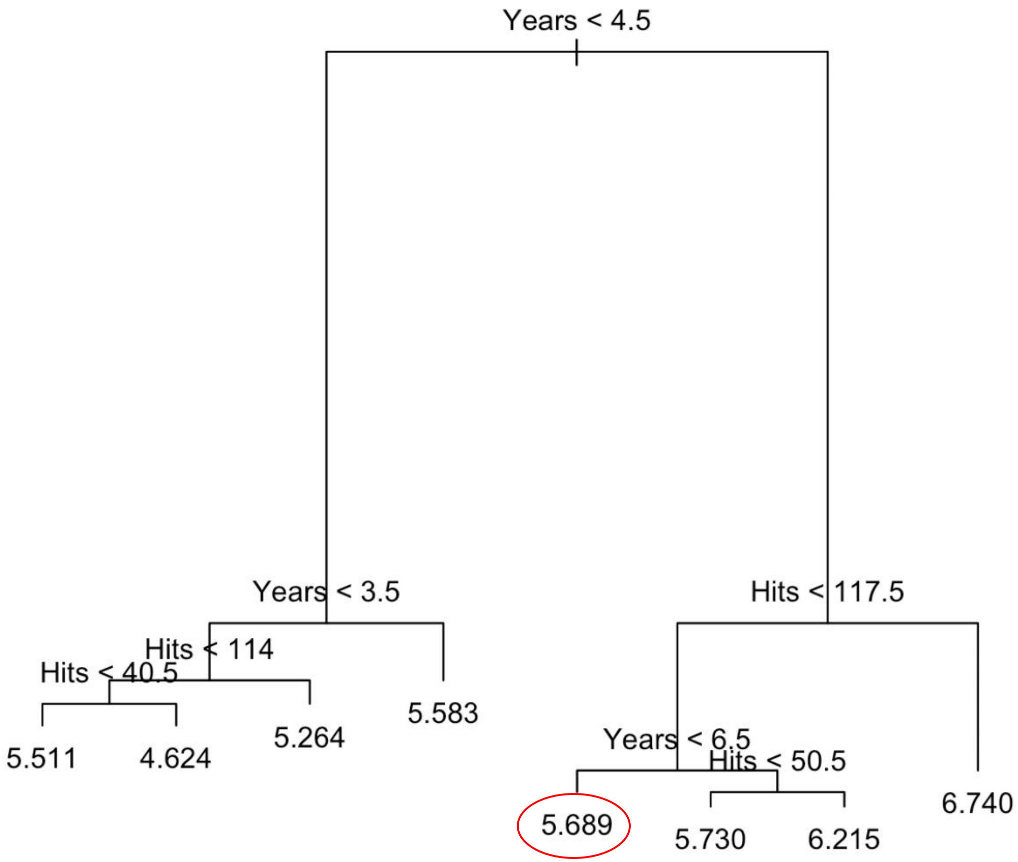
Given Years=5 and Hits=100, what is the prediction?

yhat <- predict(tree.hitters, newdata=list("Years"=5, "Hits"=100))

yhat

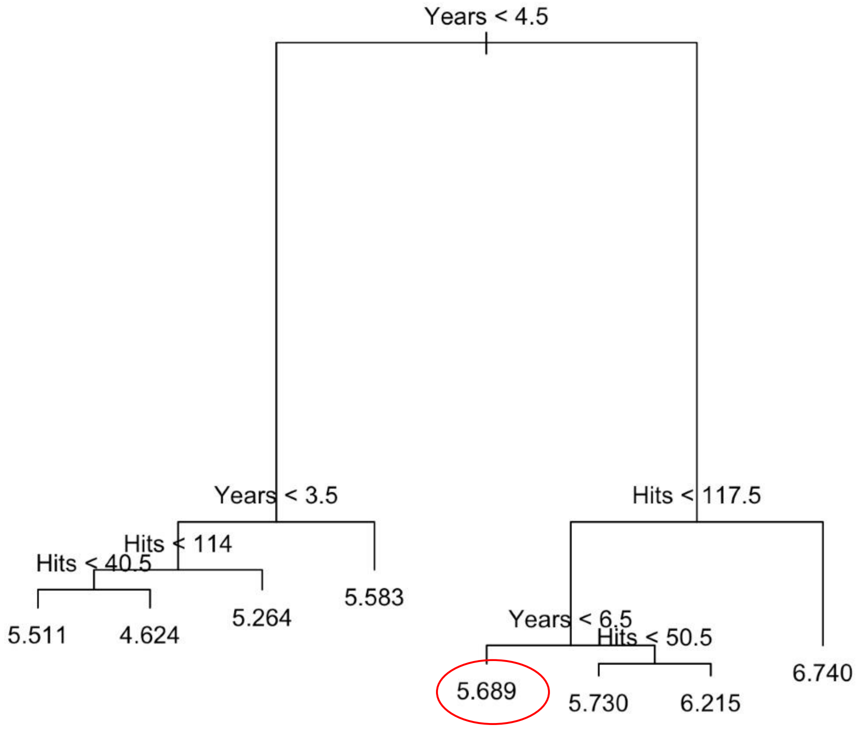
 1

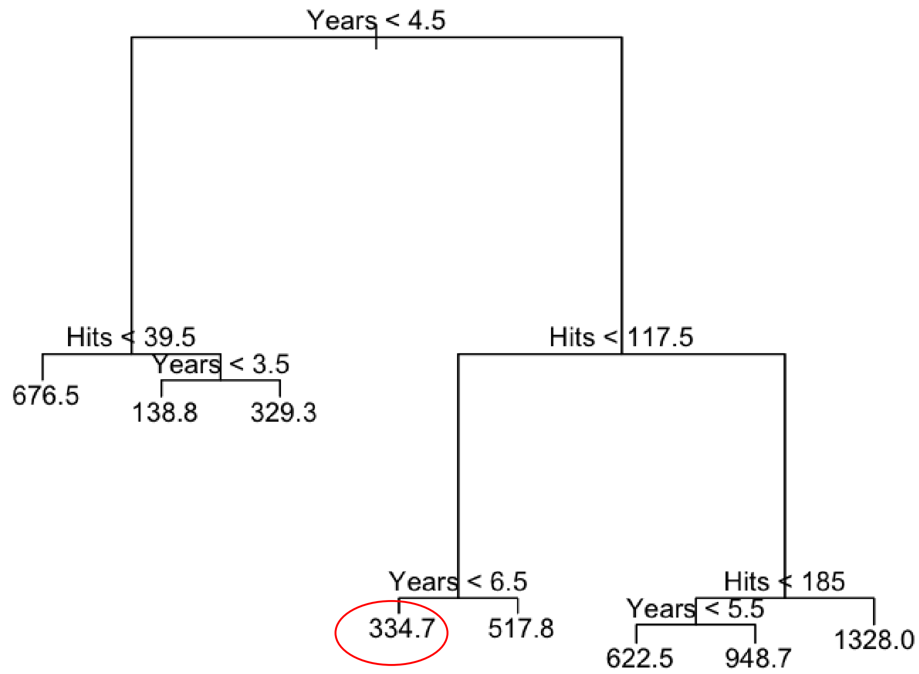
5.688925



A Comparison

Given Years=5 and Hits=100, what is the prediction?





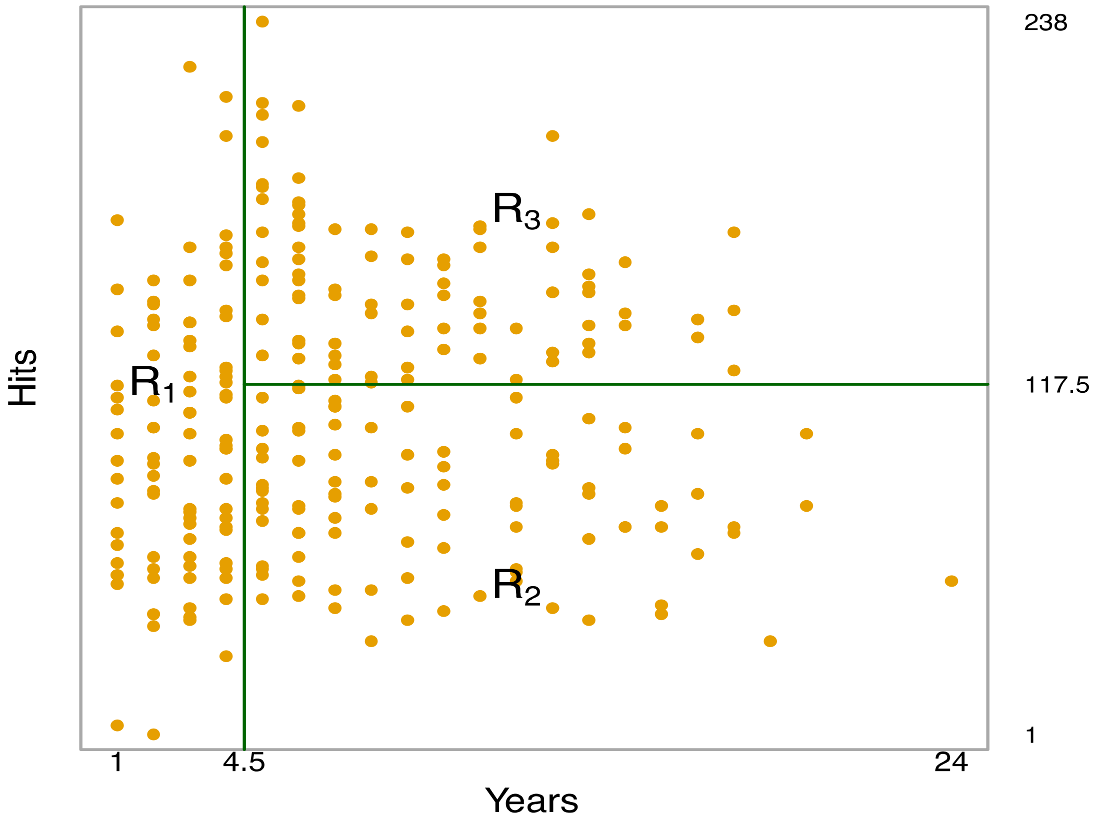
log-transformed

NOT log-transformed

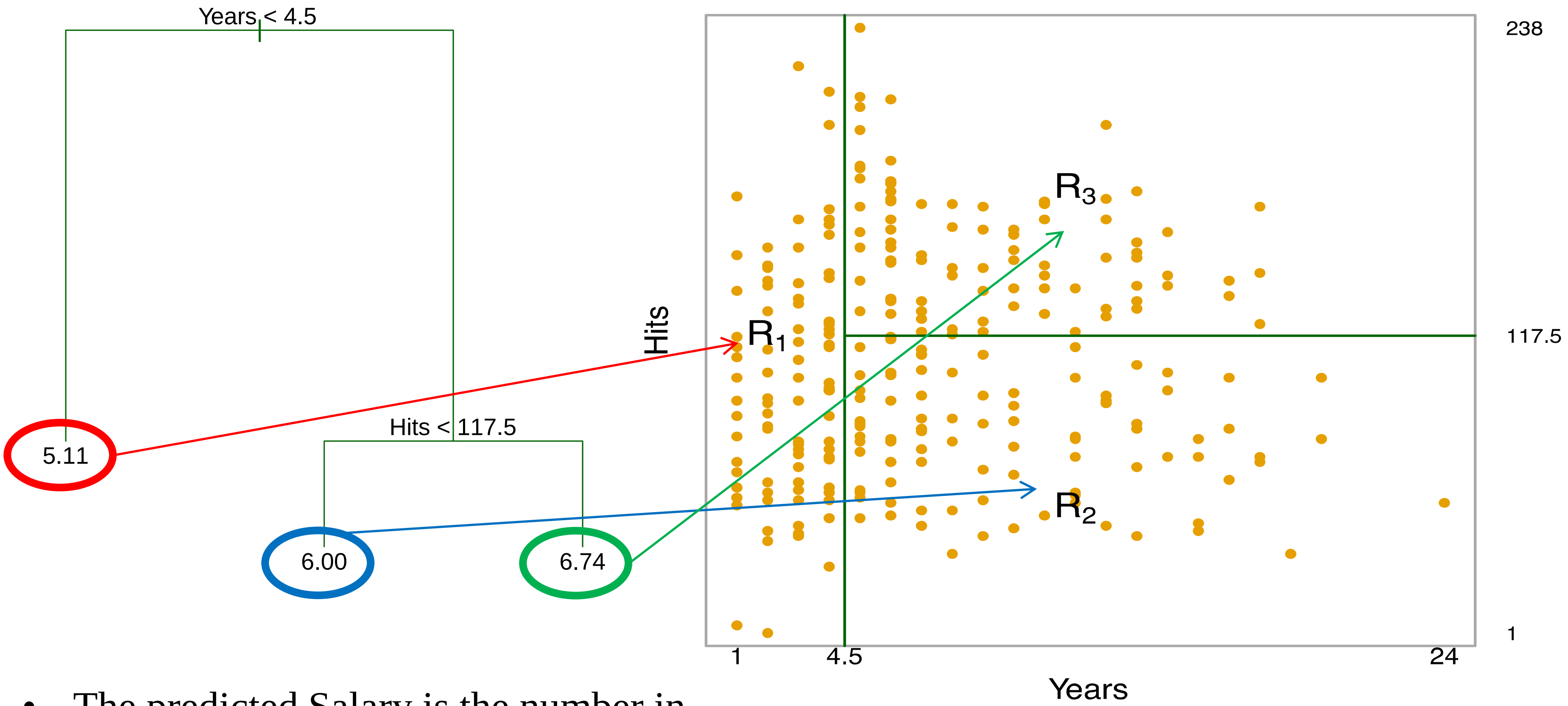
$e^{5.689} = 295.6$

334.7

Another way of visualising the decision tree



Linking two visualisations



- The predicted Salary is the number in each leaf node.
- It is the <u>mean</u> of the response for the observations that fall there

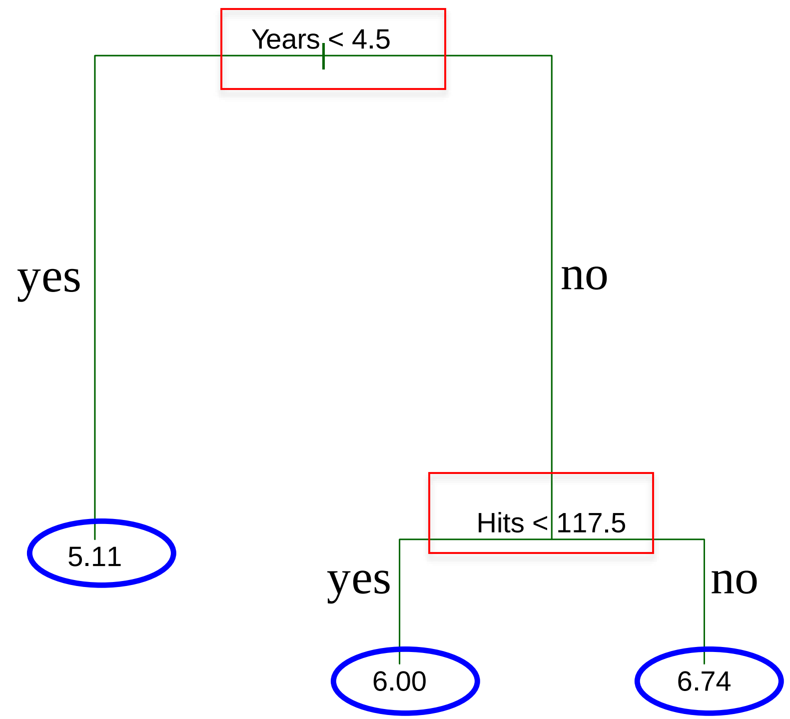
5.11 is the mean salary in region R_1

6.00 is the mean salary in region R_2

6.74 is the mean salary in region R_3

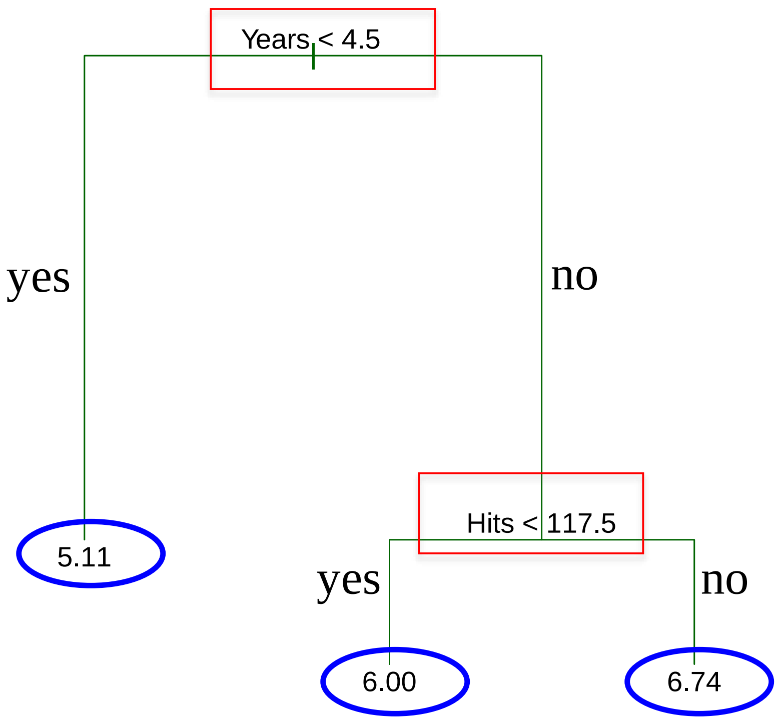
Terminology of Decision Tree

- Terminal nodes (leaves) of the tree
- Internal nodes
- Branches
  - The segments of the trees that connect the nodes
- Upside down



Interpreting the Decision Tree

- `Years` is the most important factor in determining `Salary`
  - Players with less experience tend to earn less
- For less experienced players
  - The number of hits plays little role in the salaries
- For experienced players
  - The more hits being made, the higher salary they tend to earn



Some Natural Questions

Q1. Where to split?

i.e., how do we decide on what regions to use i.e. R_1, R_2,…, R_k?

or equivalently, what tree structure should we use?

Q2. What values should we use for $\hat{Y}_1$, $\hat{Y}_2$,…, $\hat{Y}_k$?

Q2. What values should we use for $\hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}_k$?

- Simple!

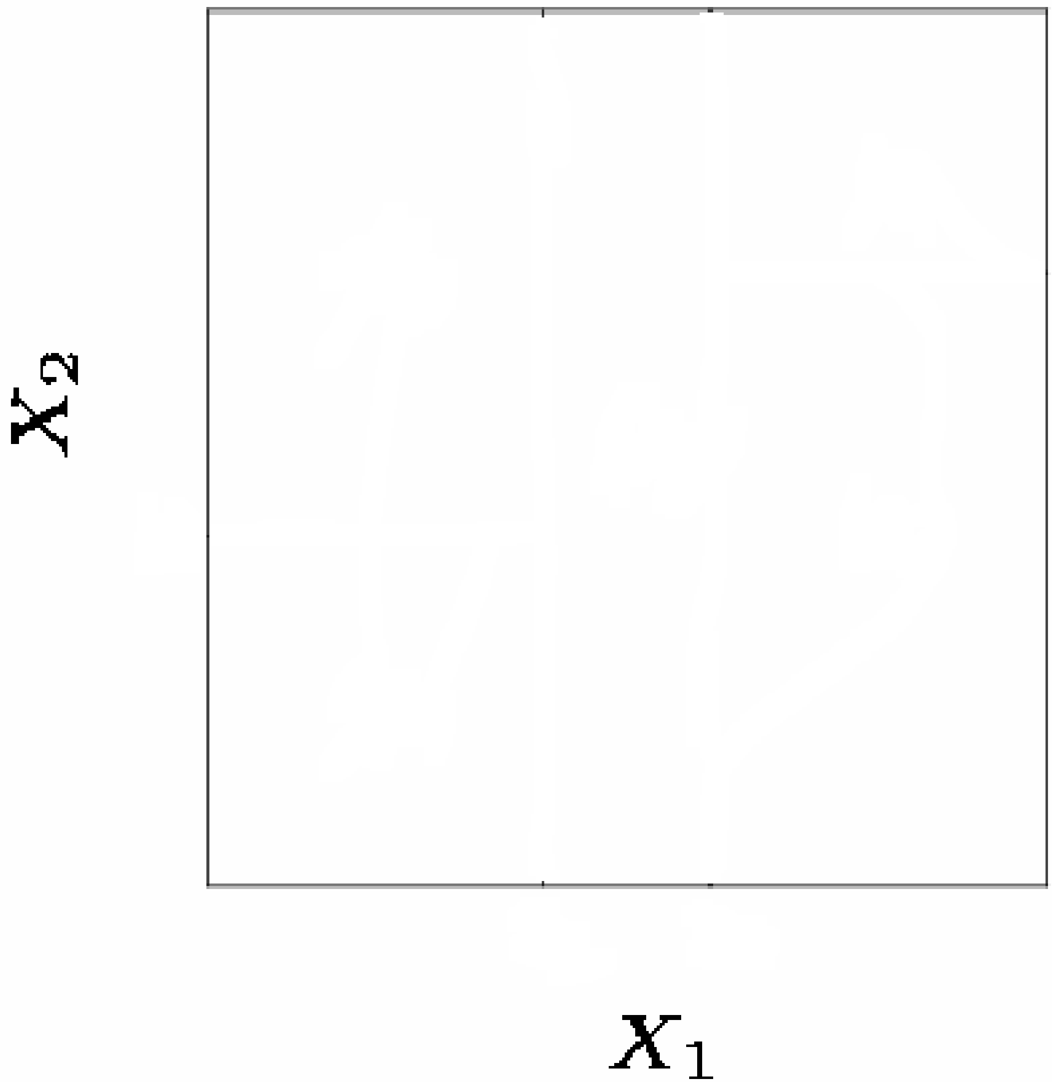
- For region R_j, the best prediction is simply the average of all the responses from our training data that fell in region R_j.

Q1. Where to Split?

- We consider splitting into two regions, $X_j > s$ and $X_j < s$ for all possible values of s and j.

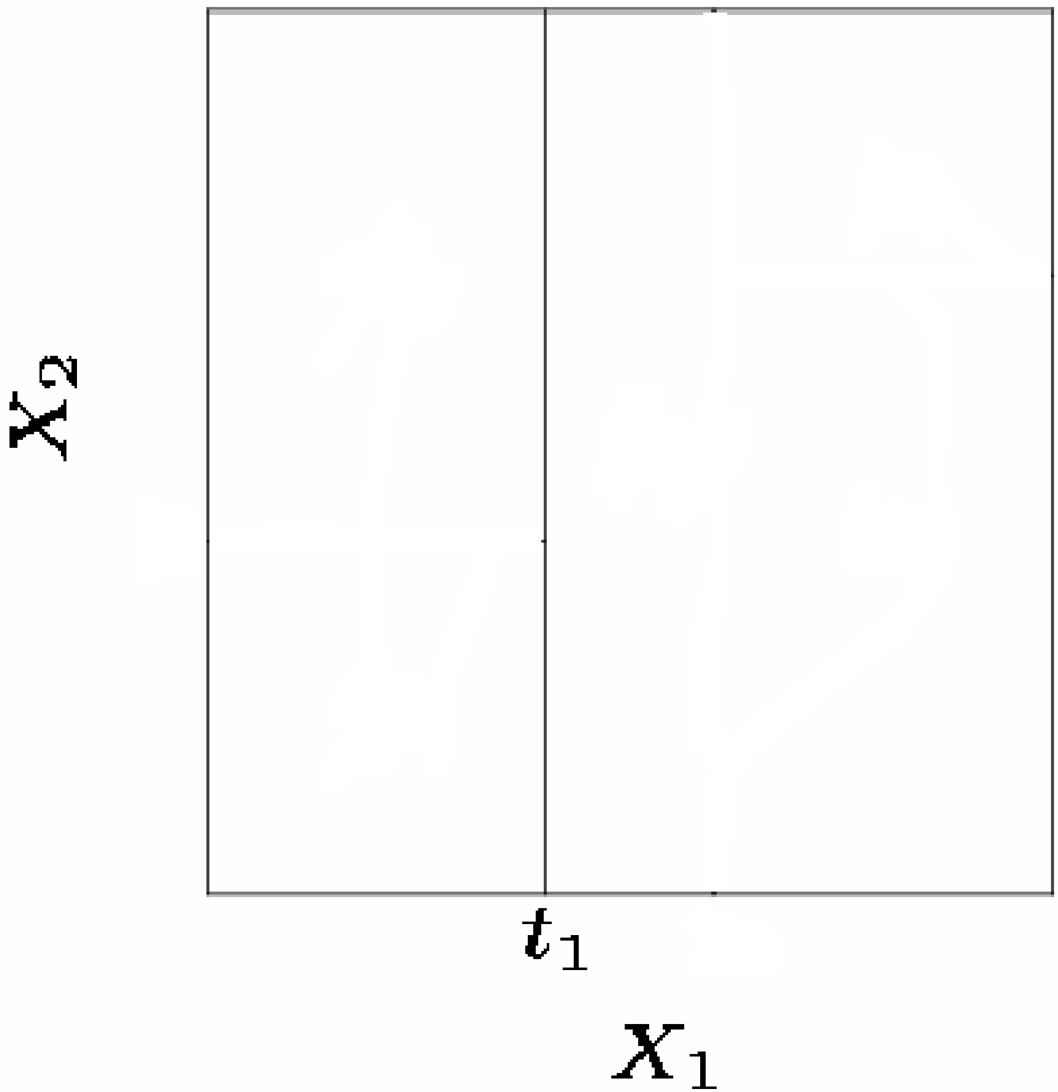
- We then choose the s and j that results in the lowest MSE on the training data.

- X_1: Years
- X_2: Hits



Q1. Where to Split?

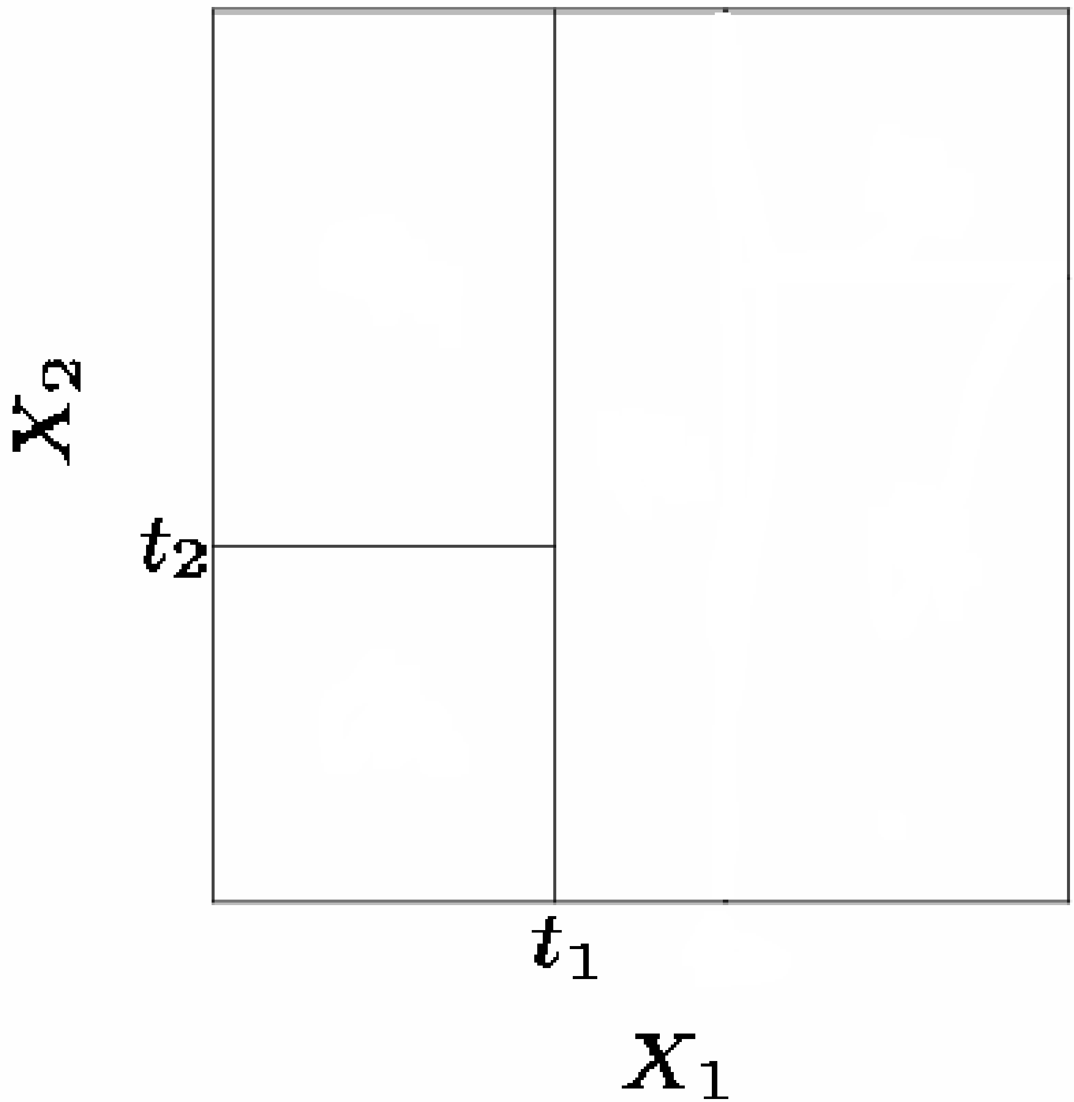
- Here the optimal split was on X_1 at point t_1.
- Now we repeat the process looking for the next best split except that we must also consider whether to split the first region or the second region up.
- Again the criteria is smallest training MSE.



Where to Split?

- Here the optimal split was the left region on X_2 at point t_2.

- [Stopping criteria] This process continues until our regions have too few observations to continue e.g. all regions have 5 or fewer points.



Tree Pruning

Improving Tree Accuracy

- A large tree (i.e. one with many terminal nodes) may tend to over fit the training data.
  - Large tree: lower bias, higher variance, worse interpretation
  - Small tree: higher bias, lower variance, better interpretation

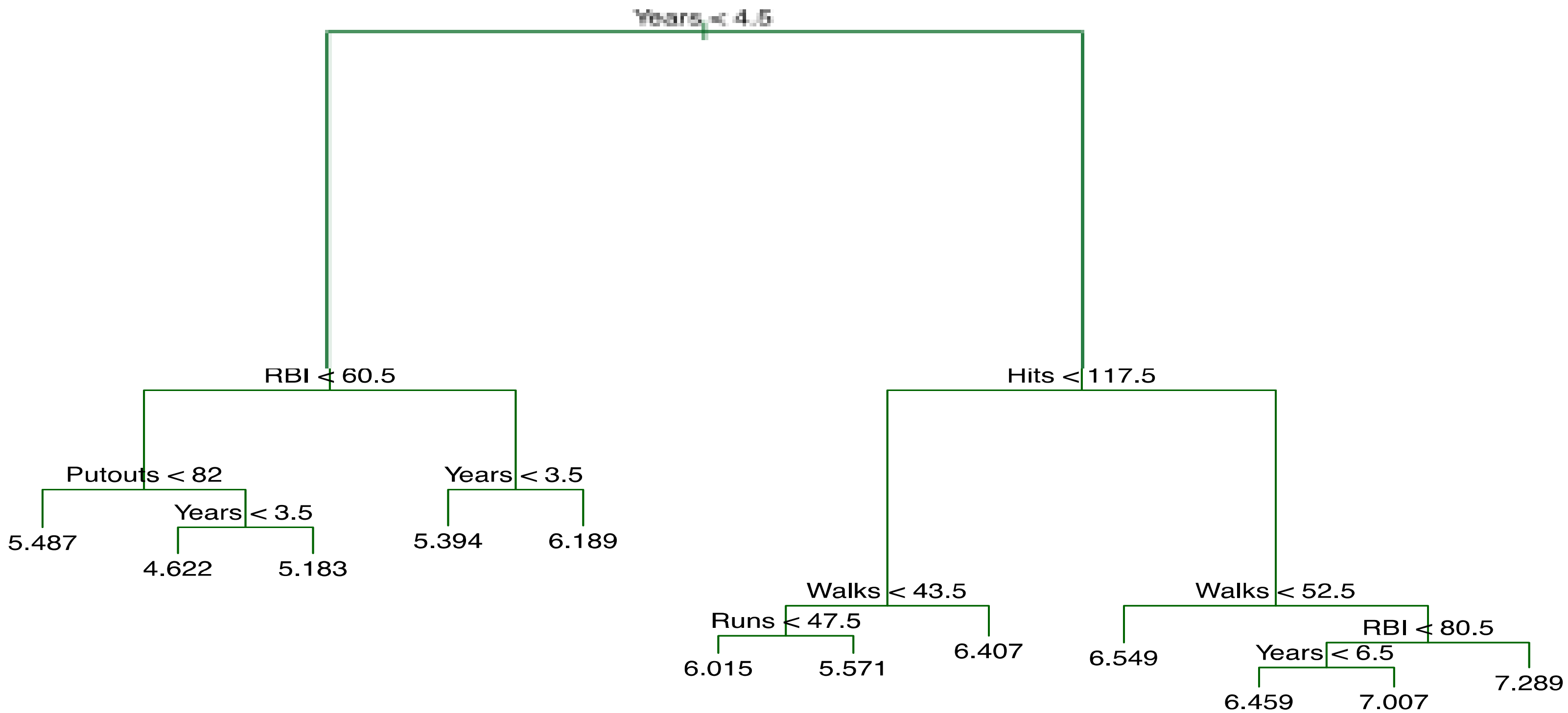
- Generally, we can improve accuracy by "pruning" the tree i.e. cutting off some of the terminal nodes.

- How do we know how far back to prune the tree?
  - We use **cross validation** to see which tree has the lowest error rate.

Outline of the Decision Tree Approach

- Split the dataset `DS` into two subsets:
  - `DS.train` and `DS.test`
- Use `DS.train` to build a decision tree `tree.train`
- Use cross validation (`cv.tree()`) to see
  - whether pruning the tree will improve performance
  - If yes, how many leaves (`w`) the best tree will have
- Use `prune.tree(tree.train,best=w)` to prune the tree if necessary
- Make predictions on the test set `DS.test` and evaluate how well the model performs
  - Calculate the test MSE or test error rate

Example: Baseball Players' Salaries



Can anyone get the same tree as this one when building a regression tree of Salary on 9 predictors?

Y: salary

X: 9 predictors

Hits+Runs+RBI+Walks+Years+PutOuts+AtBat+Assists+Errors

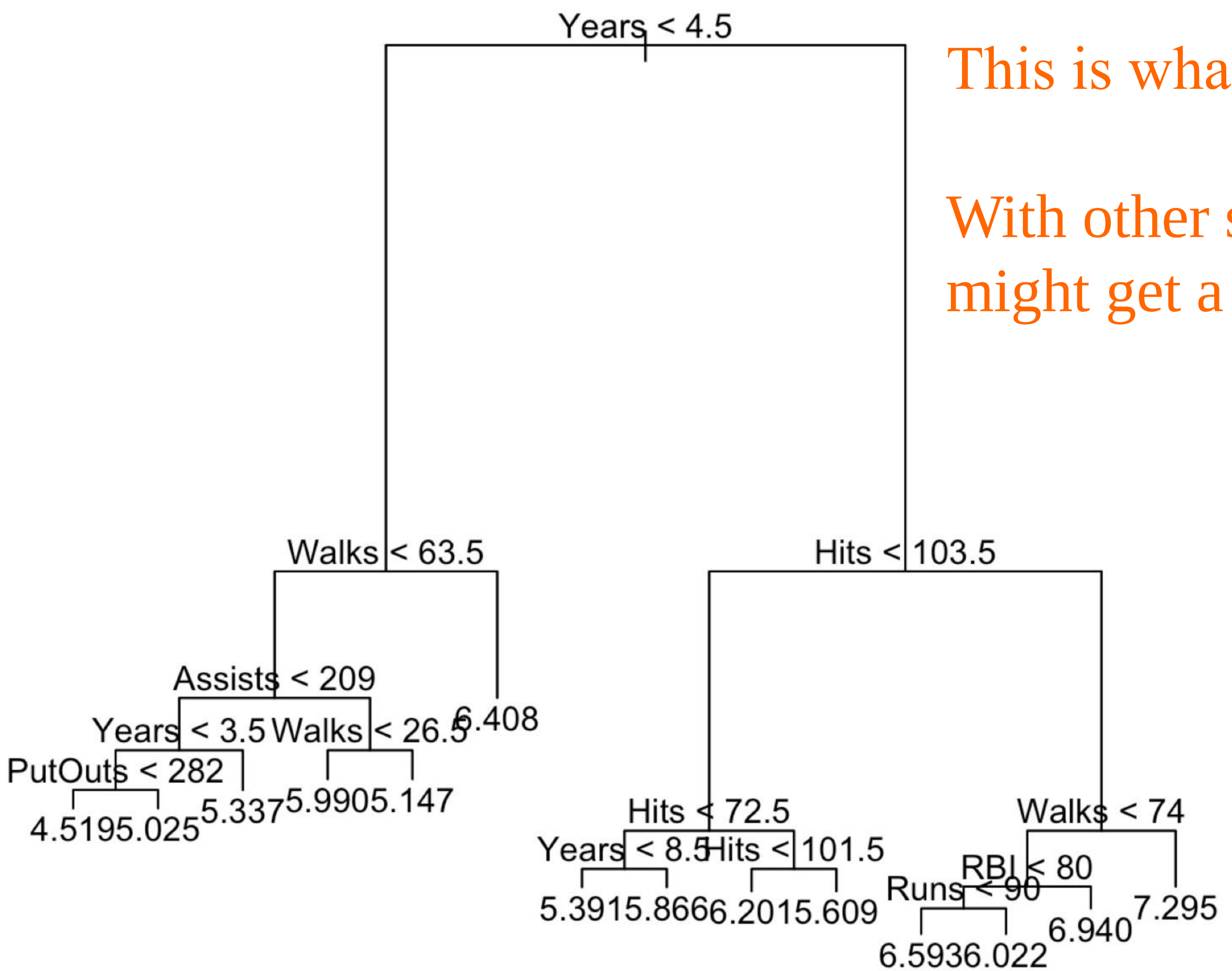
Example: Baseball Players' Salaries

```
set.seed(1)    #choosing a different seed → a different tree
train <- sample(1:nrow(Hitters),132)

tree.hitters.train <- tree(log(Hitters$Salary)~ Hits+Runs+RBI+
                      Walks+Years+PutOuts+AtBat+Assists+Errors,
                      Hitters,subset=train)
plot(tree.hitters.train)
text(tree.hitters.train)
```

Use `summary(tree.hitters.train)` to see how many predictors actually contributed to the tree

```
Variables actually used in tree construction:
[1] "Years"   "Walks"   "Assists" "PutOuts" "Hits"    "RBI"
"Runs"
Number of terminal nodes:  14
```

Example: Baseball Players' Salaries



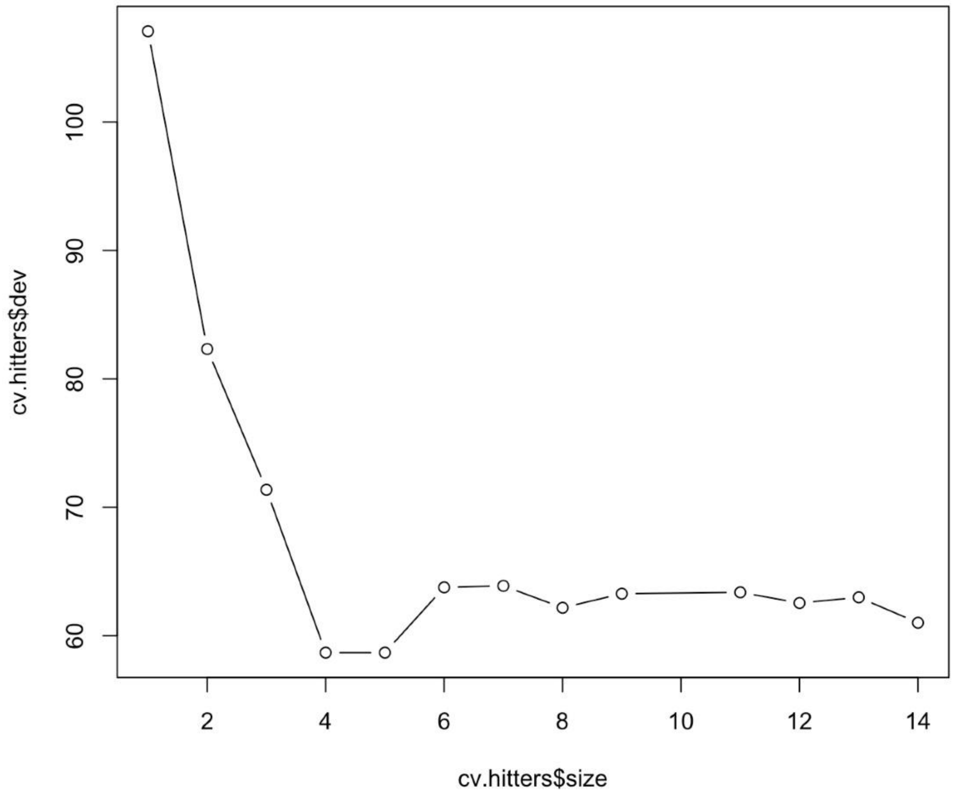
This is what I've got:

With other seeds, you might get a different tree

Example: Baseball Players' Salaries

Now we use cv.tree() function to see whether pruning the tree will improve performance.

```
cv.hitters <- cv.tree(tree.hitters.train)
plot(cv.hitters$size,cv.hitters$dev,type='b')
```

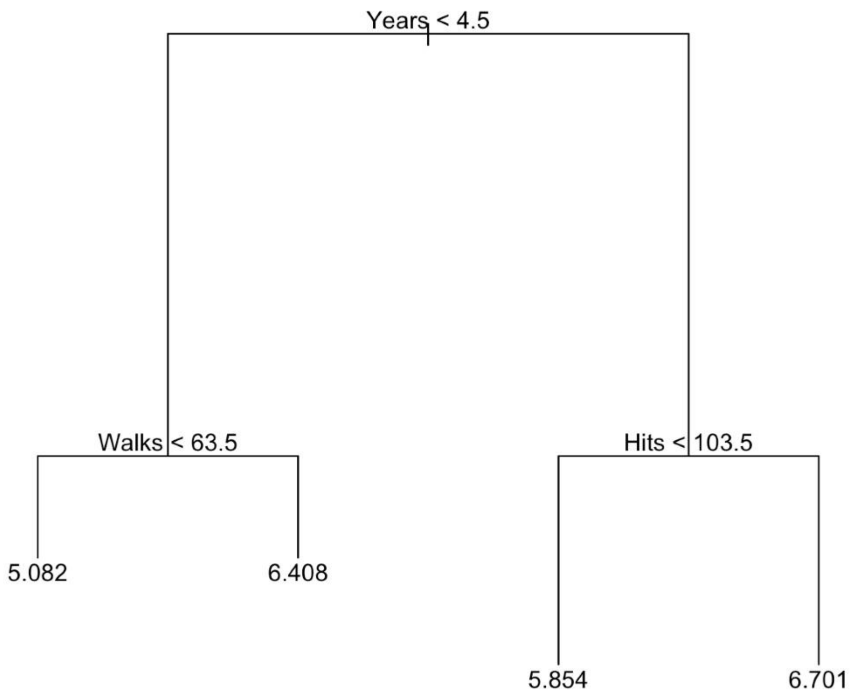


Recall: deviance is node residual sums of squares summed over the terminal nodes of the tree

Example: Baseball Players' Salaries

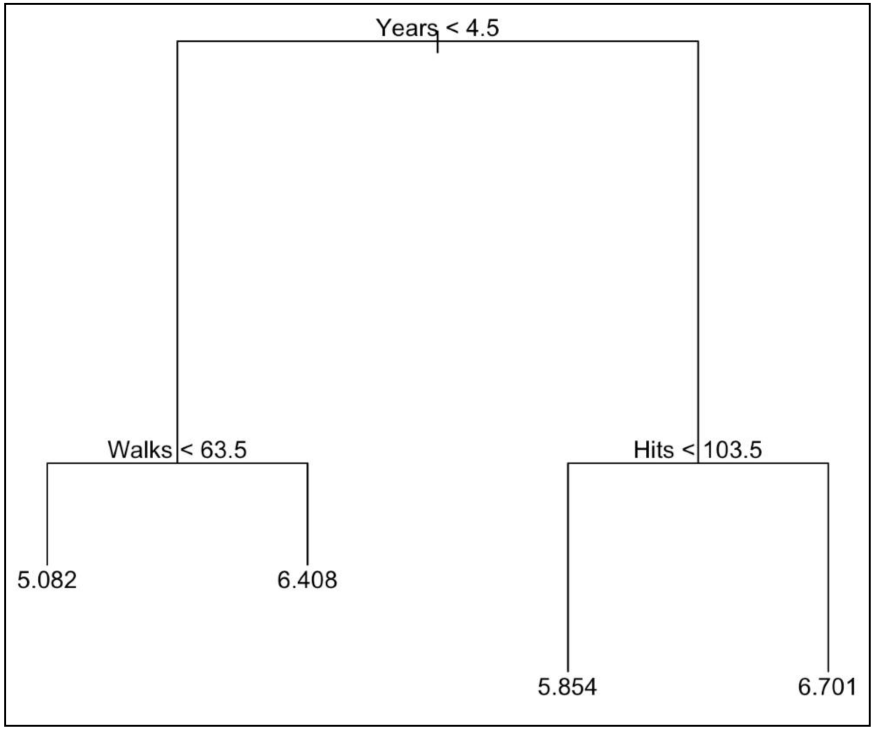
- Cross Validation indicated that the minimum MSE is when the tree size is 4 or 5 (i.e. the number of leaf nodes is 4 or 5)

- Now, we prune the tree to be of size 4:

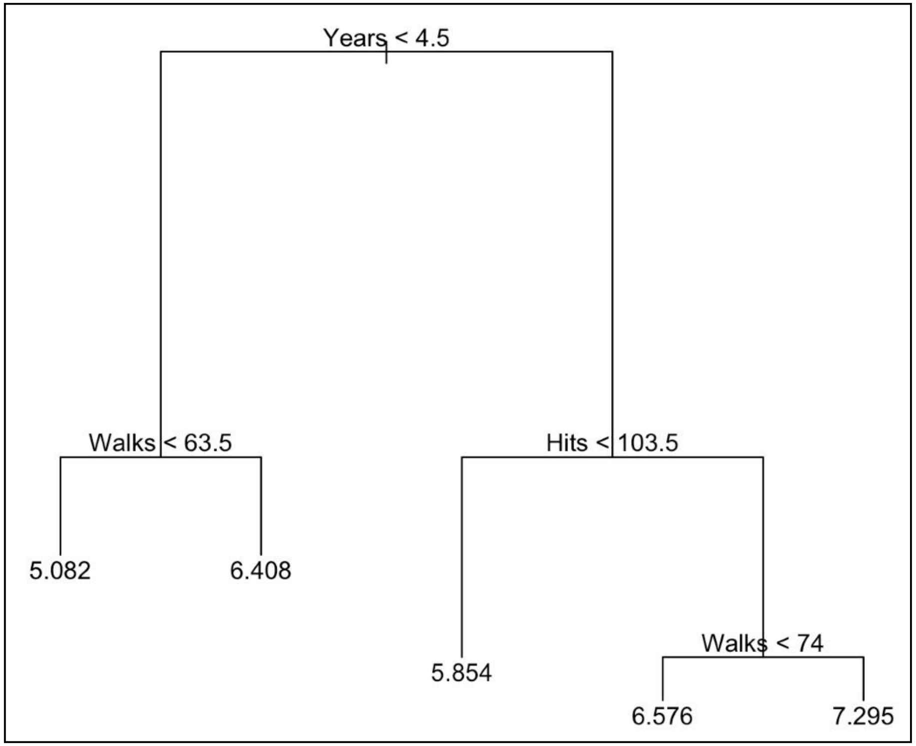



```
prune.hitters <- prune.tree(tree.hitters.train,best=4)
plot(prune.hitters)
text(prune.hitters)
```

Example: Baseball Players' Salaries



Tree pruned to be of size 4



Tree pruned to be of size 5

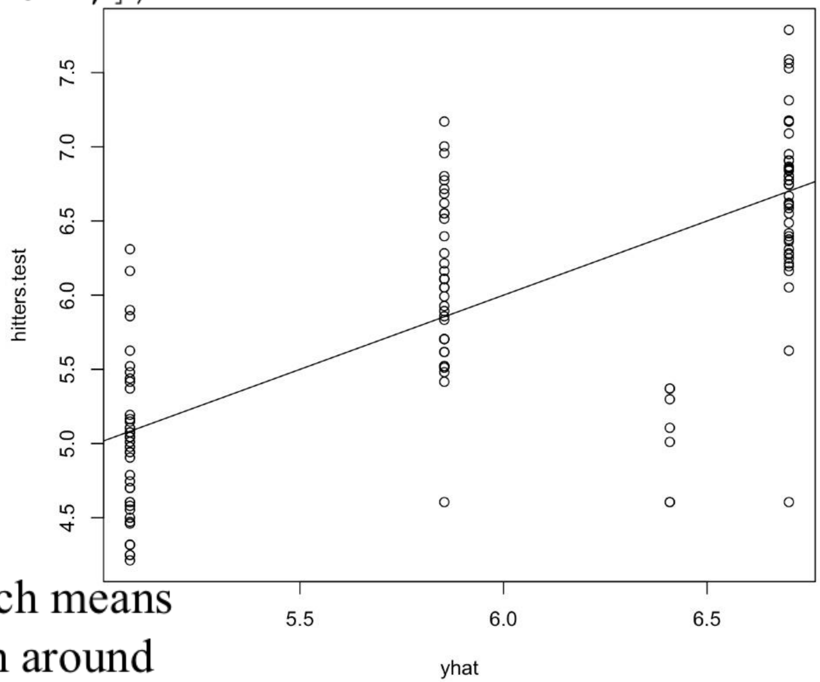
```
prune.hitters <- prune.tree(tree.hitters.train, best=5)
plot(prune.hitters)
text(prune.hitters,pretty=0)
```

Making Predictions

Use the pruned tree (size 4) to make predictions on the test set

```
yhat<-predict(prune.hitters,newdata=Hitters[-train,])
hitters.test <- log(Hitters[-train, "Salary"])

plot(yhat,hitters.test)
abline(0,1)

mean((yhat-hitters.test)^2)
[1] 0.3886456
```

← The test MSE associated with the regression tree




```
sqrt(0.3886456)
[1] 0.6234145
```

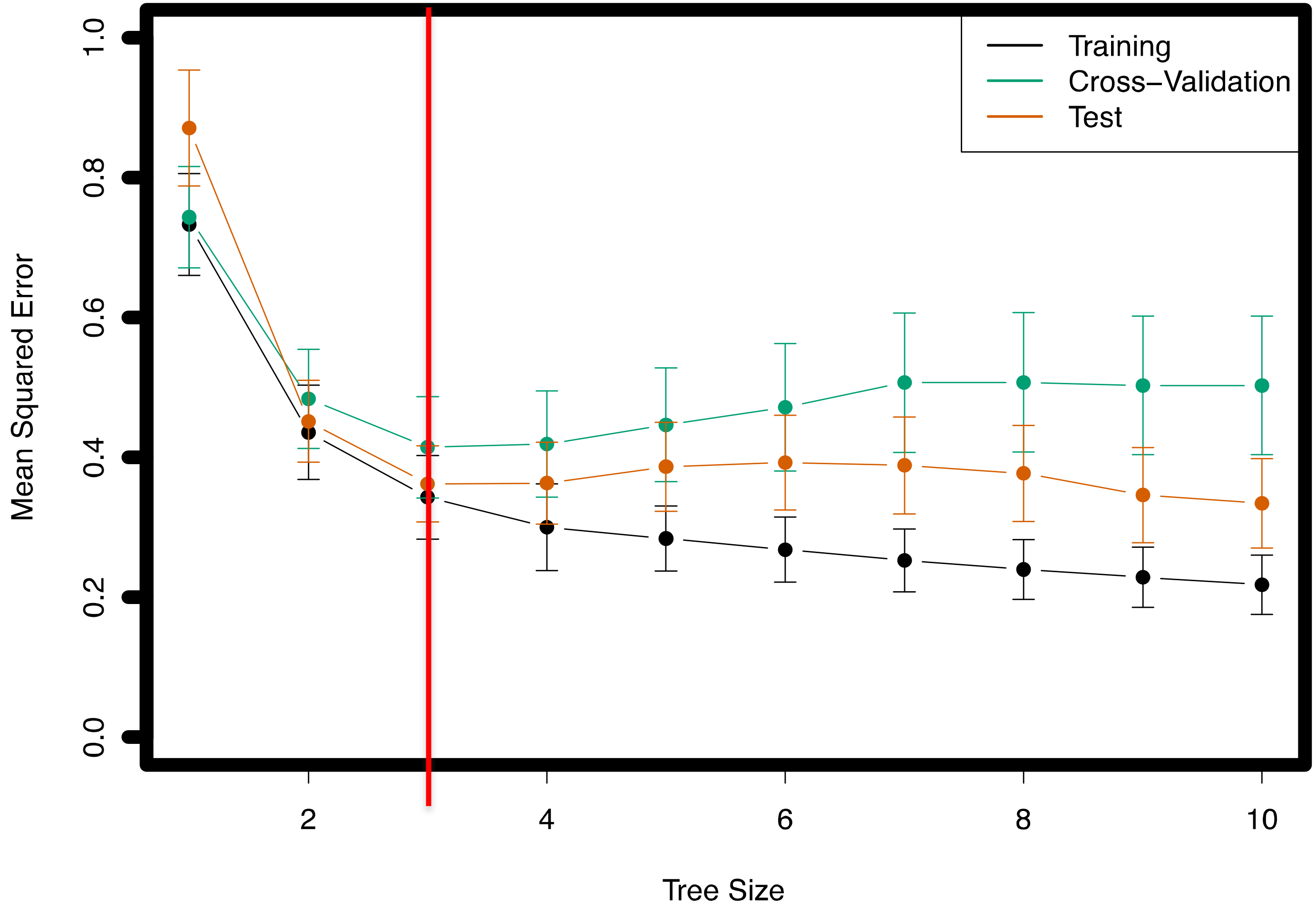
← The square root of the MSE, which means that this model leads to test predictions that are within around $\$e^{0.623}$K = \$1.865K = \$1865 of the true Salary of hitters

If using the unpruned tree to do the same, the MSE is 0.4178112.

```
yhat<-predict(tree.hitters.train, newdata=Hitters[-train,])
```

Example: Baseball Players' Salaries

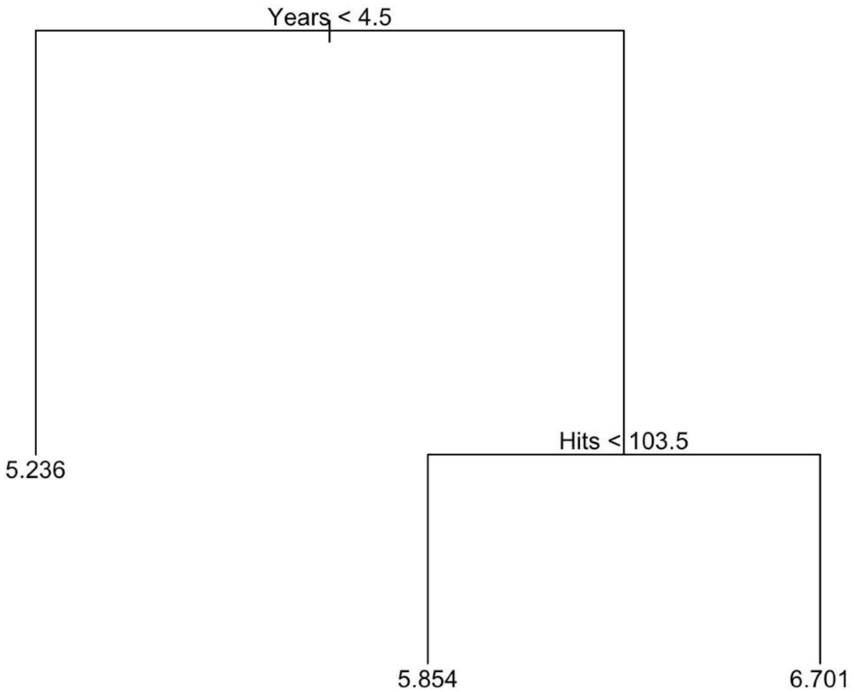
- **In the book**, with an unspecified seed to random split the data set, the minimum cross validation error occurs at a tree size of 3



Example: Baseball Players' Salaries

- Cross Validation indicated that the minimum MSE is when the tree size is three (i.e. the number of leaf nodes is 3)

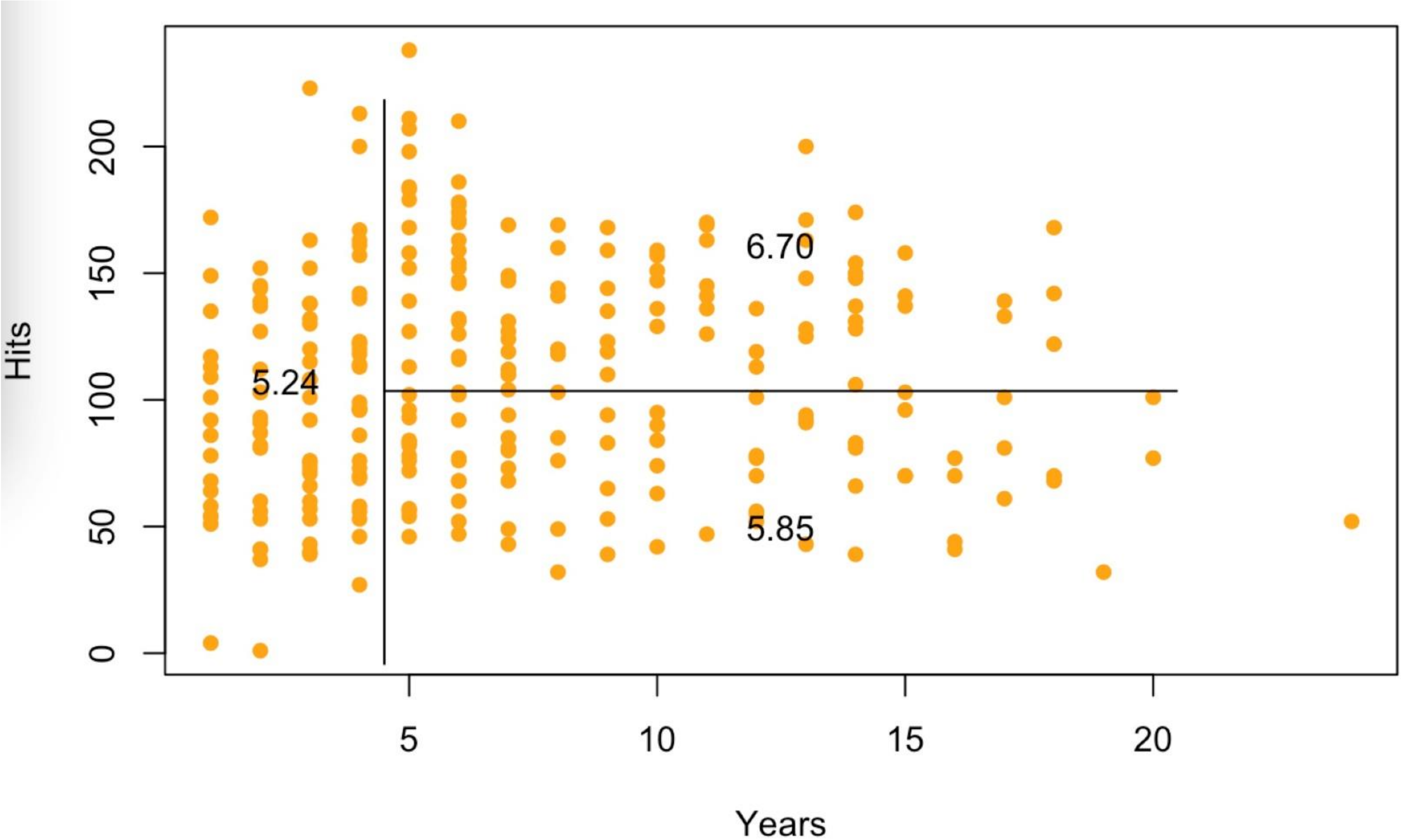
- Now, we prune the tree to be of size 3:




```
prune.hitters.3 <- prune.tree(tree.hitters.train,best=3)
plot(prune.hitters.3)
text(prune.hitters.3,pretty=0)
```

How to plot this?

```
plot(Hitters$Years, Hitters$Hits, col="orange", pch=16, xlab="Years", ylab="Hits")
partition.tree(prune.hitters, ordvars=c("Years","Hits"), add=TRUE)
```



ordvars: The ordering of the variables to be used in a 2D plot. Specify the names in a character string of length 2; the first will be used on the x axis.

Classification Trees

Predicting a qualitative response

e.g., Predicting whether a customer will default,
whether an email is a spam, etc

Growing a Classification Tree

- A classification tree is very similar to a regression tree except that we try to make a prediction for a categorical rather than continuous Y.
- For each region (or node) we predict the most common category among the training data within that region.
  - What measure can it be?
- The tree is grown (i.e. the splits are chosen) in exactly the same way as with a regression tree except that minimising MSE no longer makes sense.
- There are several possible different criteria to use such as the "gini index" and "cross-entropy" but the easiest one to think about is to minimise the error rate.

Evaluation of classification models

- Recall: Counts of test records that are correctly (or incorrectly) predicted by the classification model

- **Confusion matrix**

Actual Class \ Predicted Class	Class = 1	Class = 0
Class = 1	f_{11}	f_{10}
Class = 0	f_{01}	f_{00}

$$\text{Accuracy} = \frac{\text{\# correct predictions}}{\text{total \# of predictions}} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

$$\text{Error rate} = \frac{\text{\# wrong predictions}}{\text{total \# of predictions}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

Example: Carseats

- Goal: to analyse the `Carseats` data set

```
library(ISLR)
```

Sales, CompPrice, Income, Advertising, Population, Price, ShelveLoc, Age, Education, Urban, US

- `Sales` is a continuous variable, discretise it using `ifelse()`

```
High <- ifelse(Carseats$Sales<=8,"No","Yes")
```

- Merge `High` with the rest of the `Carseats` data

```
Carseats <- data.frame(Carseats,High)
```

- Fitting a classification tree

```
tree.carseats <- tree(High~.-Sales,Carseats)
summary(tree.carseats)        #How many predictors are used?
```

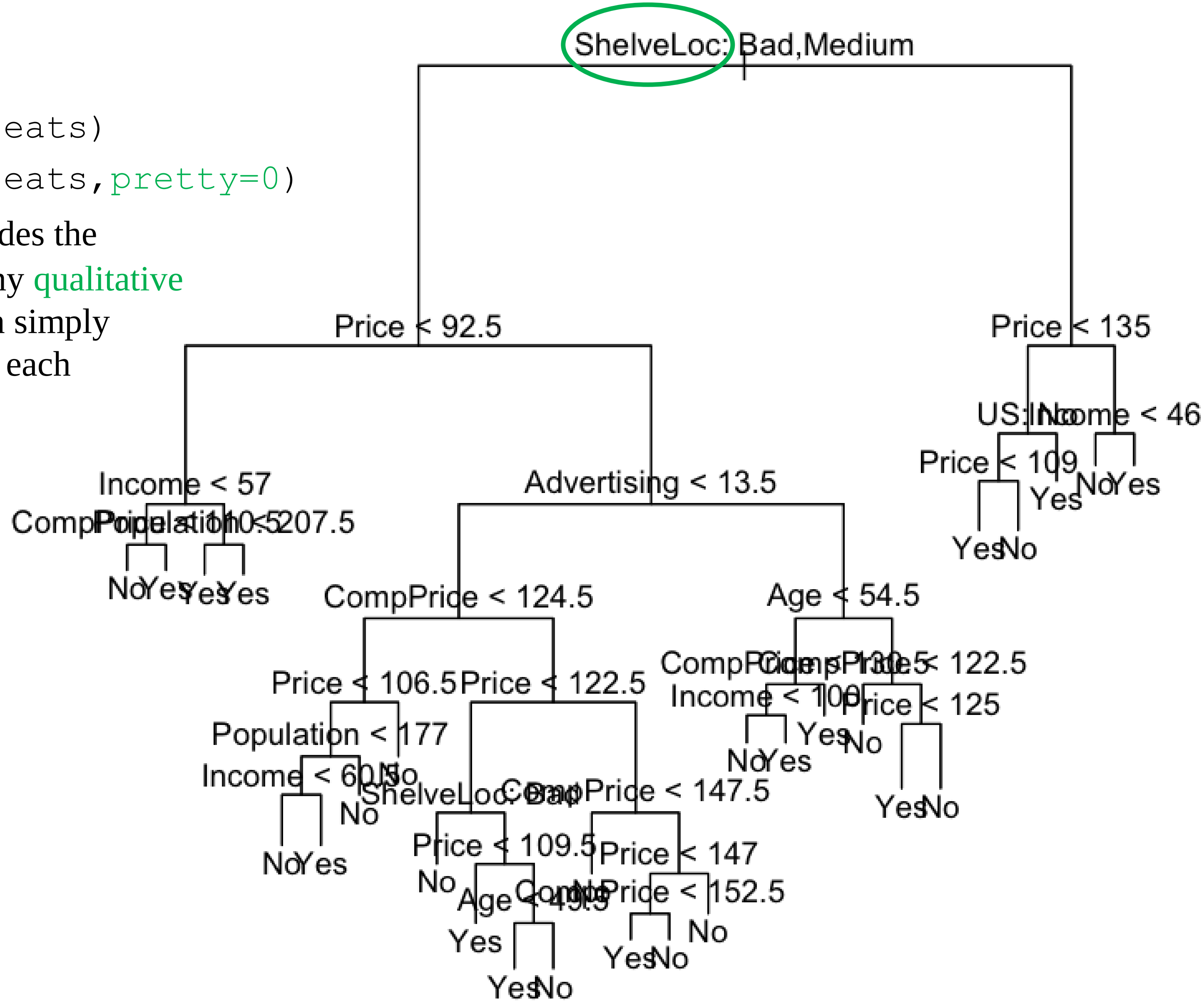
Example: Carseats

```
> summary(tree.carseats)

Classification tree:
tree(formula = High ~ . - Sales, data = Carseats)

Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"       "Income"      "CompPrice"   "Population"
[6] "Advertising" "Age"         "US"

Number of terminal nodes:  27

Residual mean deviance:  0.4575 = 170.7 / 373
Misclassification error rate: 0.09 = 36 / 400   ➔ training error rate is 9%
```

Plotting the tree

```
plot(tree.carseats)
text(tree.carseats,pretty=0)
```

`# pretty=0` includes the category names for any qualitative predictors, rather than simply displaying a letter for each category



Test Error Rate Estimation

- Estimate the test error rather than computing the training error
  - Split the observations into a training set and a test set
  - Build the tree using the training set
  - Evaluate its performance on the test data
    - By `predict()`, where `type="class"` returns the actual class prediction

```
set.seed(2)
train <- sample(1:nrow(Carseats),nrow(Carseats)/2)   # 1: can be omitted
Carseats.test <- Carseats[-train,]
High.test <- High[-train]
tree.carseats.train <- tree(High~.-Sales, Carseats, subset=train)
tree.pred.test <- predict(tree.carseats.train, Carseats.test, type="class")

table(tree.pred.test,High.test)
             High.test
tree.pred.test No Yes
           No  86  27
           Yes 30  57
```

Test Error Rate is 28.5%

What is training error rate? 9%

```
> (27+30)/200
[1] 0.285
```

Pruning a Tree

- Consider whether pruning the tree might lead to improved results

Step 1: Use `cv.tree()` to determine the optimal level of tree complexity

```
set.seed(3)
cv.carseats <- cv.tree(tree.carseats.train, FUN=prune.misclass)
cv.carseats
$size
 [1] 19 17 14 13  9  7  3  2  1
$dev
 [1] 55 55 53 52 50 56 69 65 80
```

$dev ← this is the cv error rate (here: number of misclassified)

Step 2: Use `prune.misclass()` to prune the tree

```
prune.carseats <- prune.misclass(tree.carseats.train, best=9)
```

Step 3: Performance evaluation

```
tree.pred <- predict(prune.carseats, Carseats.test, type="class")
table(tree.pred,High.test)
         High.test
tree.pred No  Yes
      No  94  24
      Yes 22  60
```

```
> (24+22)/200    test error rate
[1] 0.23
```

← better than that without pruning 28.5%

Pruning improved interpretability and classification accuracy

Pruning a Tree

- If we increase the value of best, we obtain a larger pruned tree with lower classification accuracy

```
prune.carseats <- prune.misclass(tree.carseats, best=15)

tree.pred <- predict(prune.carseats, Carseats.test, type="class")
table(tree.pred,High.test)
         High.test
tree.pred No  Yes
      No  86  22
      Yes 30  62

(30+22)/200
[1] 0.26    ➔ higher error rate than unpruned tree 0.23
```

Summary: Decision Tree Induction

- Decision tree generation consists of two phases:
  - Tree construction
    - At start, all the training examples are at the root
    - Partition examples recursively based on selected attributes
  - Tree pruning
    - Identify and remove branches that reflect noise or outliers

- Use of decision tree:
  - Regressing or classifying an unknown sample
    - Test the attribute values of the sample against the decision tree

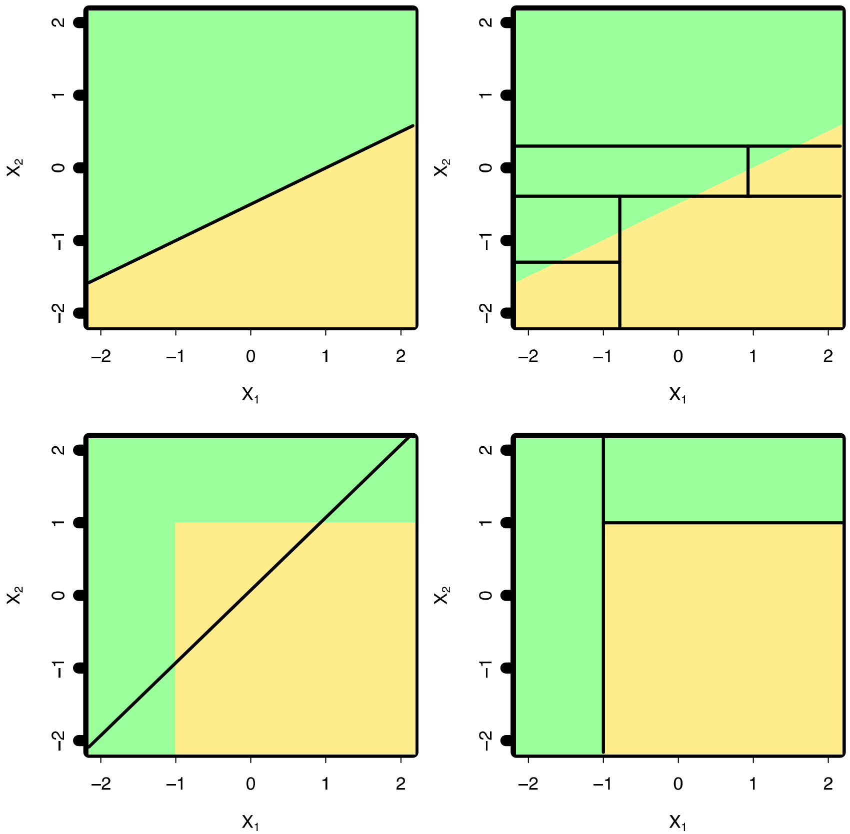
Trees vs. Linear models

Trees vs. Linear Models

- Which model is better?
  - If the relationship between the predictors and response is linear, then classical linear models such as linear regression would outperform regression trees
  - On the other hand, if the relationship between the predictors is non-linear, then decision trees would outperform classical approaches

Trees vs. Linear Model: Classification Example

- Top row: the true decision boundary is linear
  - Left: linear model (good)
  - Right: decision tree
- Bottom row: the true decision boundary is non-linear
  - Left: linear model
  - Right: decision tree (good)



Pros and Cons of Decision Trees

- Pros:
  - Trees are very easy to explain to people (probably even easier than linear regression)
  - Trees can be plotted graphically, and are easily interpreted even by non-expert
  - They work fine on both classification and regression problems
- Cons:
  - Trees don’t have the same prediction accuracy as some of the more complicated approaches that we examine in this course

Fitting Classification Trees

- The `tree` library is used to construct classification and regression trees
  - Install this library if necessary (How?)

- Several key points:
  - Fitting a tree
  - Plotting a tree
  - Pruning a tree
  - Estimating test error of the fitting

Fitting a Tree

- Goal: to analyse the `Carseats` data set

```
library(ISLR)
attach(Carseats)
```

- `Sales` is a continuous variable, discretise it using `ifelse()`

```
High <- ifelse(Sales<=8,"No","Yes")
```

- Merge `High` with the rest of the `Carseats` data

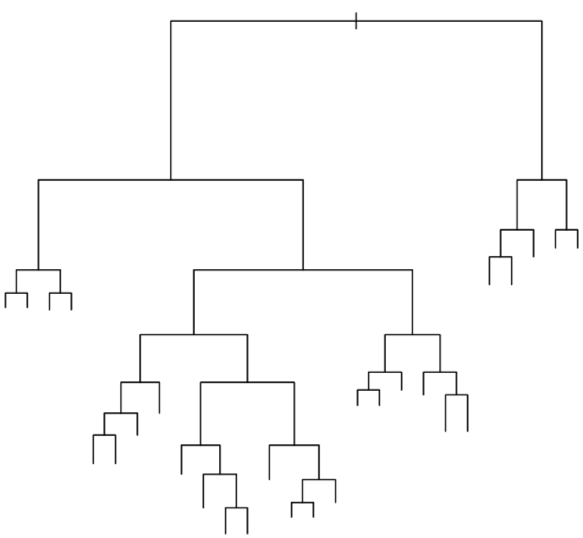
```
Carseats <- data.frame(Carseats,High)
```

- Fitting a classification tree

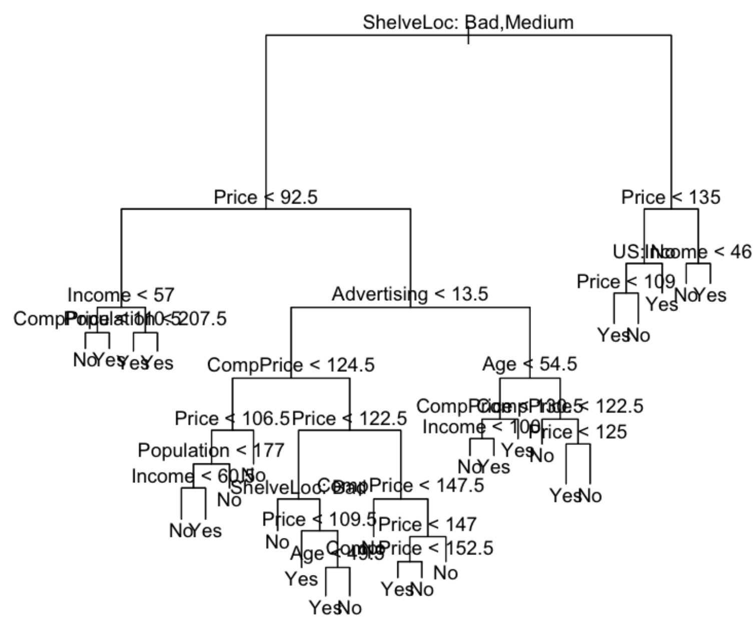
```
tree.carseats <- tree(High~.-Sales,Carseats)
summary(tree.carseats)
```

Plotting a Tree

- Trees can be naturally graphically displayed
  - `plot()` to display structure
  - `text()` to display the node labels
    - `pretty=0` includes the category names for any qualitative predictors, rather than simply displaying a letter for each category



`plot(tree.carseats)`



`text(tree.carseats,pretty=0)`

Showing More Info

```
> tree.carseats
node), split, n, deviance, yval, (yprob)      * denotes terminal node

  1) root 400 541.500 No ( 0.59000 0.41000 )
    2) ShelveLoc: Bad,Medium 315 390.600 No ( 0.68889 0.31111 )
      4) Price < 92.5 46  56.530 Yes ( 0.30435 0.69565 )
        8) Income < 57 10  12.220 No ( 0.70000 0.30000 )
         16) CompPrice < 110.5 5   0.000 No ( 1.00000 0.00000 ) *
         17) CompPrice > 110.5 5   6.730 Yes ( 0.40000 0.60000 ) *
        9) Income > 57 36  35.470 Yes ( 0.19444 0.80556 )
         18) Population < 207.5 16  21.170 Yes ( 0.37500 0.62500 ) *
         19) Population > 207.5 20   7.941 Yes ( 0.05000 0.95000 ) *
      5) Price > 92.5 269 299.800 No ( 0.75465 0.24535 )
       10) Advertising < 13.5 224 213.200 No ( 0.81696 0.18304 )
         20) CompPrice < 124.5 96  44.890 No ( 0.93750 0.06250 )
           40) Price < 106.5 38  33.150 No ( 0.84211 0.15789 )
             80) Population < 177 12  16.300 No ( 0.58333 0.41667 )
              160) Income < 60.5 6   0.000 No ( 1.00000 0.00000 ) *
              161) Income > 60.5 6   5.407 Yes ( 0.16667 0.83333 ) *
             81) Population > 177 26   8.477 No ( 0.96154 0.03846 ) *
           41) Price > 106.5 58   0.000 No ( 1.00000 0.00000 ) *
         21) CompPrice > 124.5 128 150.200 No ( 0.72656 0.27344 )
```

Test Error Rate Estimation

- Estimate the test error rather than computing the training error
  – Split the observations into a training set and a test set
  – Build the tree using the training set
  – Evaluate its performance on the test data
    - By `predict()`, where `type="class"` returns the actual class prediction

```
set.seed(2)
train <- sample(1:nrow(Carseats),200)
Carseats.test <- Carseats[-train,]
High.test <- High[-train]
tree.carseats <- tree(High~.-Sales, Carseats, subset=train)
tree.pred <- predict(tree.carseats, Carseats.test, type="class")

table(tree.pred,High.test)
         High.test
tree.pred No  Yes
      No  86  27
      Yes 30  57
(86+57)/200
[1] 0.715
```

Pruning a Tree

- Consider whether pruning the tree might lead to improved results

Step 1: Use `cv.tree()` to determine the optimal level of tree complexity

```
set.seed(3)
cv.carseats <- cv.tree(tree.carseats.train,FUN=prune.misclass)
cv.carseats
$size
[1] 19 17 14 13  9  7  3  2  1
$dev    ← this is the cv error rate
[1] 55 55 53 52 50 56 69 65 80
```

Step 2: Use `prune.misclass()` to prune the tree

```
prune.carseats <- prune.misclass(tree.carseats.train,best=9)
```

Step 3: Performance evaluation

```
tree.pred <- predict(prune.carseats,Carseats.test,type="class")
table(tree.pred,High.test)
         High.test
tree.pred No Yes
      No  94  24
      Yes 22  60
```

```
> (94+60)/200
[1] 0.77   ← better than that without pruning
```

REGRESSION TREES

Fitting Regression Trees

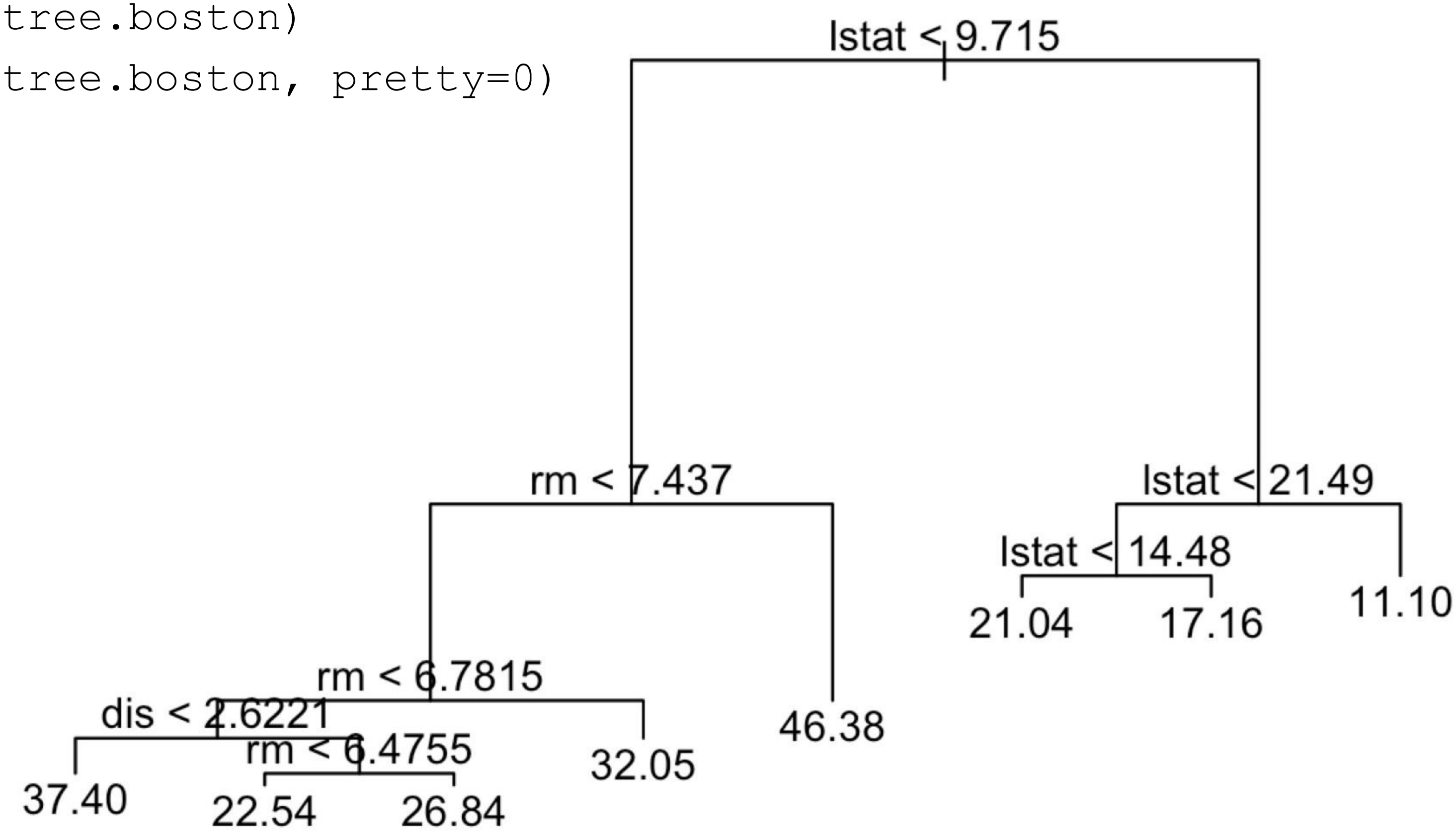
- Fit a regression tree to the Boston data
  - Create a training set and fit the tree to the training set

```
library(MASS)
set.seed(1)
train <- sample(1:nrow(Boston),nrow(Boston)/2)
tree.boston <- tree(medv~.,Boston,subset=train)

summary(tree.boston)
Regression tree:
tree(formula = medv ~ ., data = Boston, subset = train)
Variables actually used in tree construction:
[1] "lstat" "rm"    "dis"   ← only 3 variables have been used
Number of terminal nodes:  8
Residual mean deviance:  12.65 = 3099 / 245
Distribution of residuals:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-14.10000  -2.04200  -0.05357   0.00000   1.96000  12.60000
```

Fitting Regression Trees

– Plot the tree

```
plot(tree.boston)
text(tree.boston, pretty=0)
```



Decide Whether to Prune

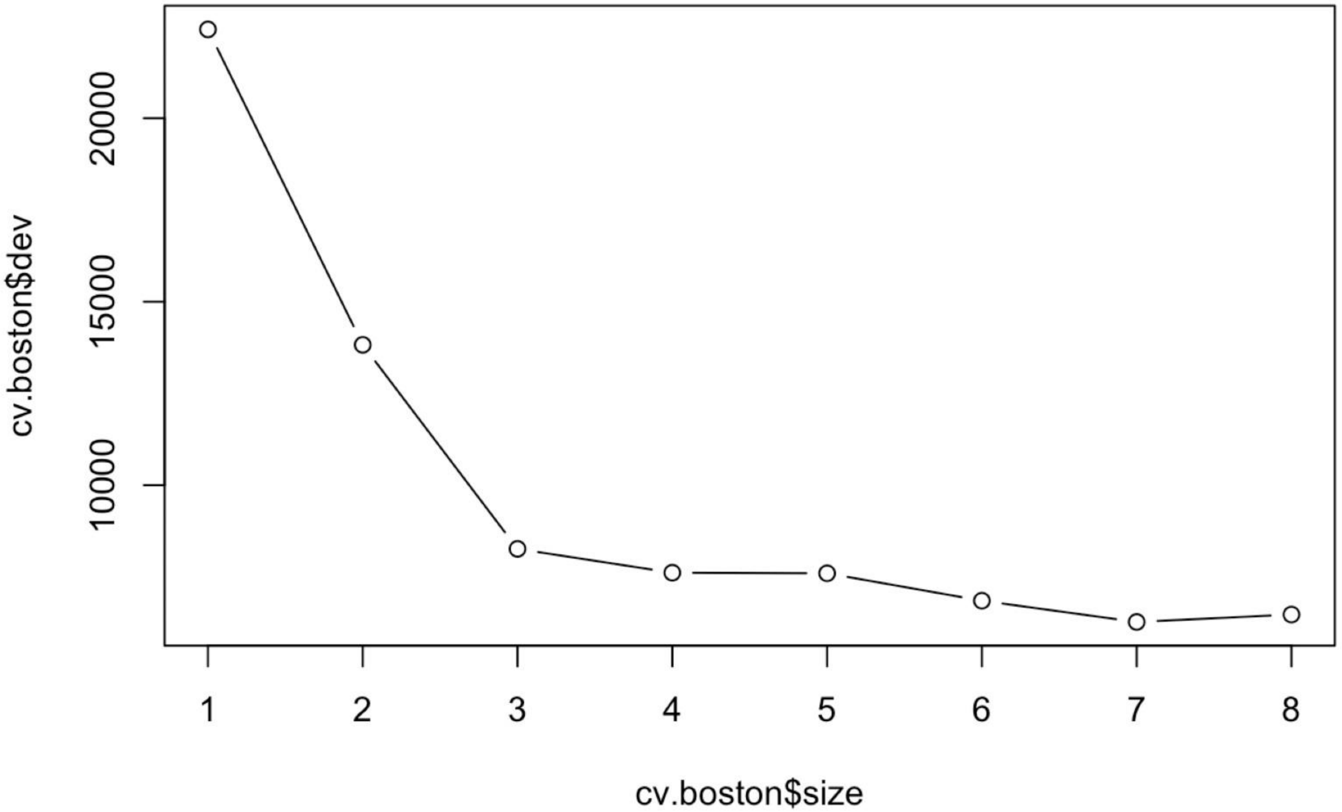
– Using `cv.tree()` function to see whether pruning the tree will improve performance

```
set.seed(1)
cv.boston <- cv.tree(tree.boston)
cv.boston
$size
[1] 8 7 6 5 4 3 2 1

$dev
[1]  6475.338  6273.382  6852.736  7599.829  7615.683  8265.076  13824.821  22419.739

$k
[1]      -Inf  255.6581  451.9272  768.5087  818.8885 1559.1264  4276.5803  9665.3582

$method
[1] "deviance"

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

The result shows that the best tree is the one with **7** terminals.

Decide Whether to Prune

```
plot(cv.boston$size, cv.boston$dev, type='b')
```



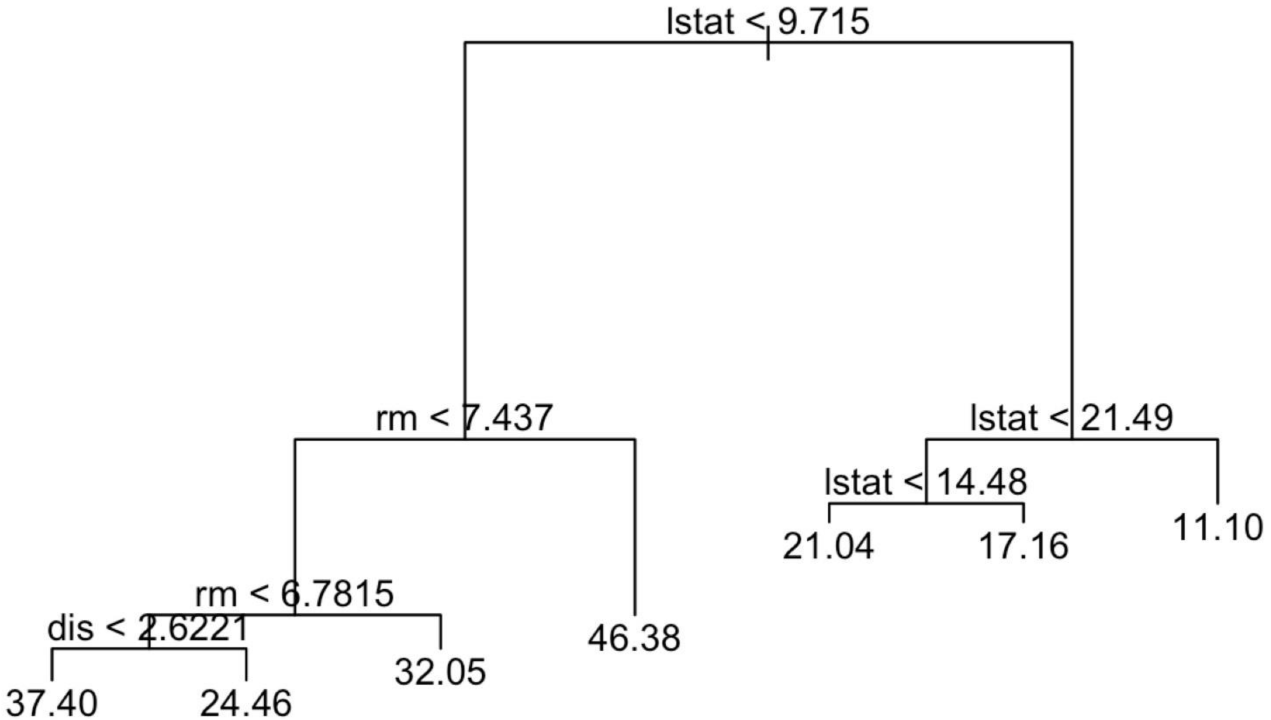
Pruning a Regression Tree

```
prune.boston <- prune.tree(tree.boston, best=7)
summary(prune.boston)
```

```
Regression tree:
snip.tree(tree = tree.boston, nodes = 17L)
Variables actually used in tree construction:
[1] "lstat" "rm"    "dis"
Number of terminal nodes:  7
Residual mean deviance:  13.64 = 3354 / 246
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-14.1000  -2.2610  -0.1033   0.0000   2.1210  12.6000
```

Pruning a Regression Tree

```
plot(prune.boston)
text(prune.boston, pretty = 0)
```



Regression Trees for Prediction

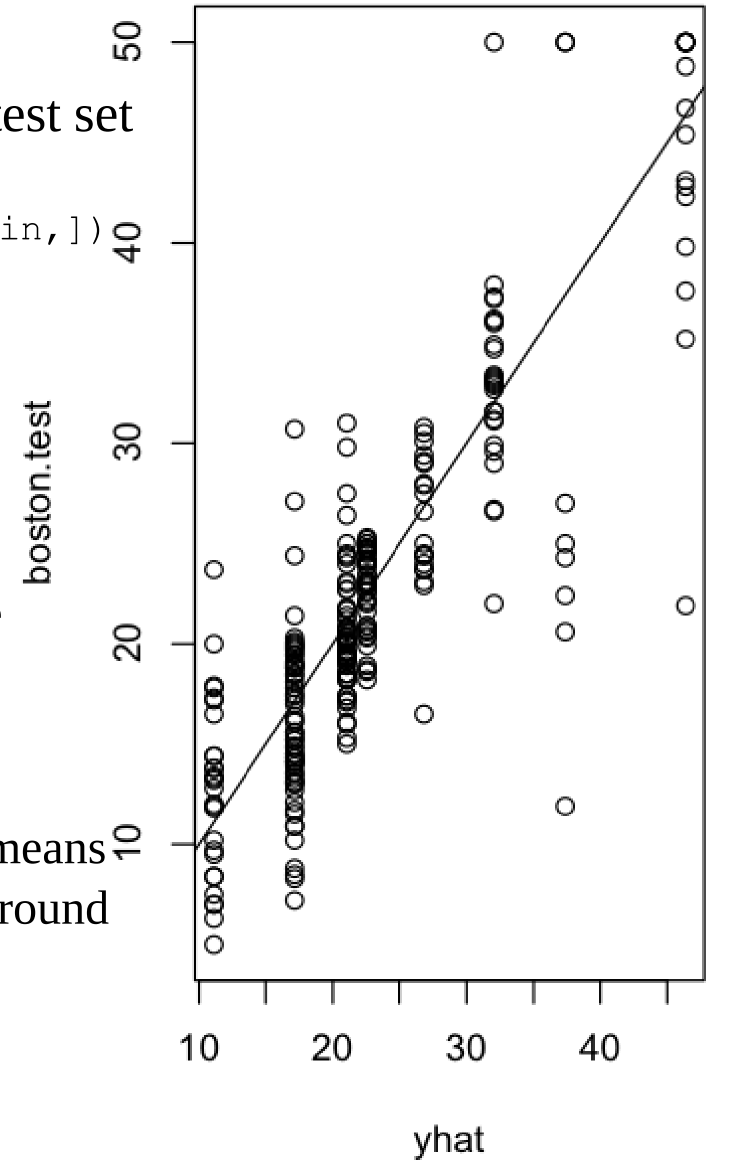
Use the unpruned tree to make predictions on the test set

```
yhat <- predict(tree.boston, newdata=Boston[-train,])
boston.test <- Boston[-train,"medv"]

plot(yhat,boston.test)
abline(0,1)

mean((yhat-boston.test)^2)
```

`[1] 25.04559` ← The test set MSE associated with the regression tree

```
sqrt(25.04559)
```

`[1] 5.004557` ← The square root of the MSE, which means that this model leads to test predictions that are within around $5,005 of the true median home value for the suburb



Regression Trees for Prediction

Use the pruned tree to make predictions on the test set

```
yhat.prune <- predict(prune.boston, newdata=Boston[-train,])
boston.test <- Boston[-train,"medv"]

mean((yhat.prune-boston.test)^2)
[1] 25.72341
```

← The estimated test MSE associated with the regression tree

```
sqrt(25.72341)
[1] 5.071825
```

← This is higher than the unpruned tree!