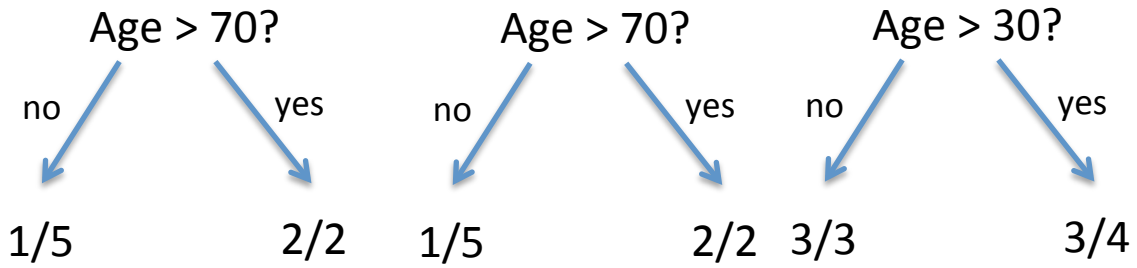
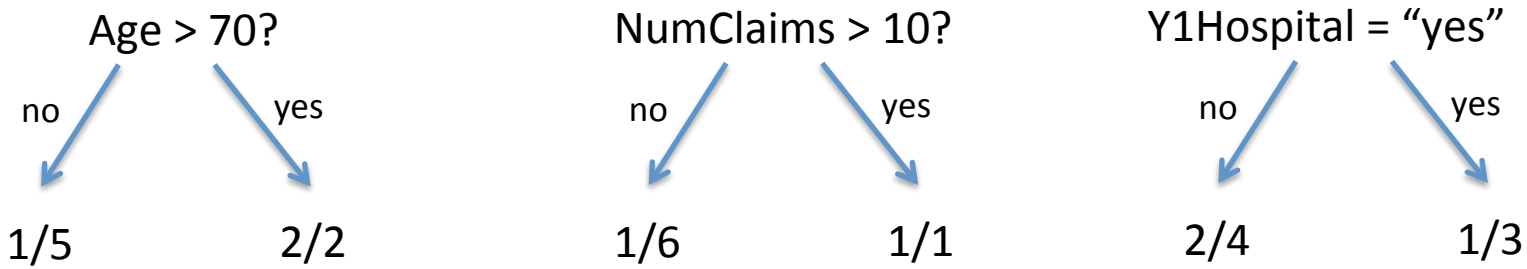
Tree Models

Age	MaxCharlson	Y1Hospital	NumClaims	Y2Hospital
75	4	Yes	4	Yes
12	1	No	1	No
35	0	No	4	Yes
38	1	Yes	1	No
15	3	Yes	5	No
23	2	No	3	No
75	4	No	11	Yes
56	5	Yes	22	?

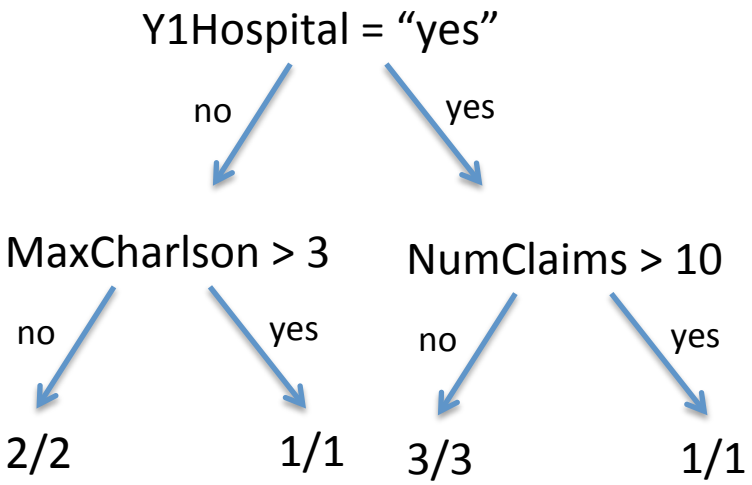


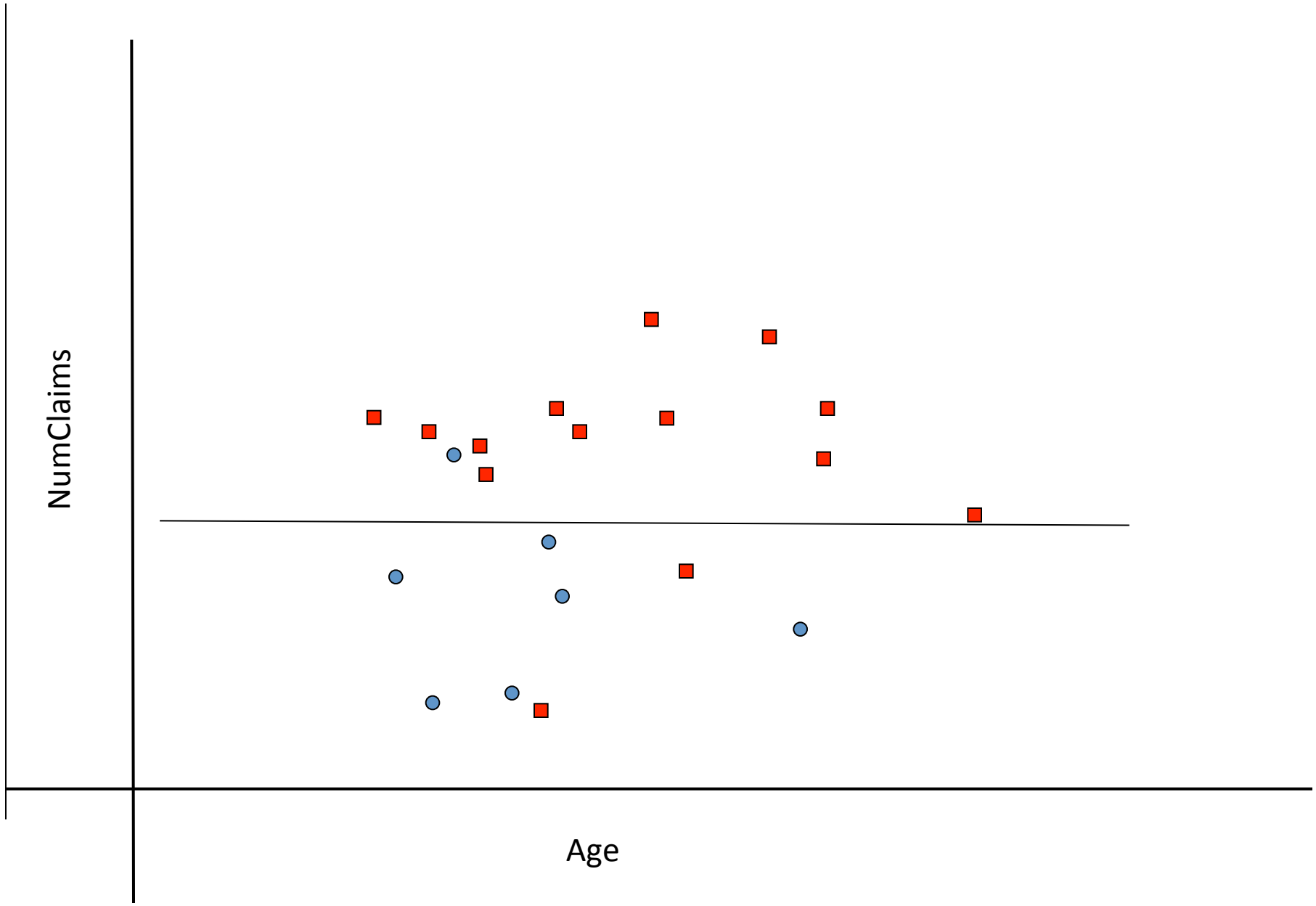
What do you predict for the 56-year-old?

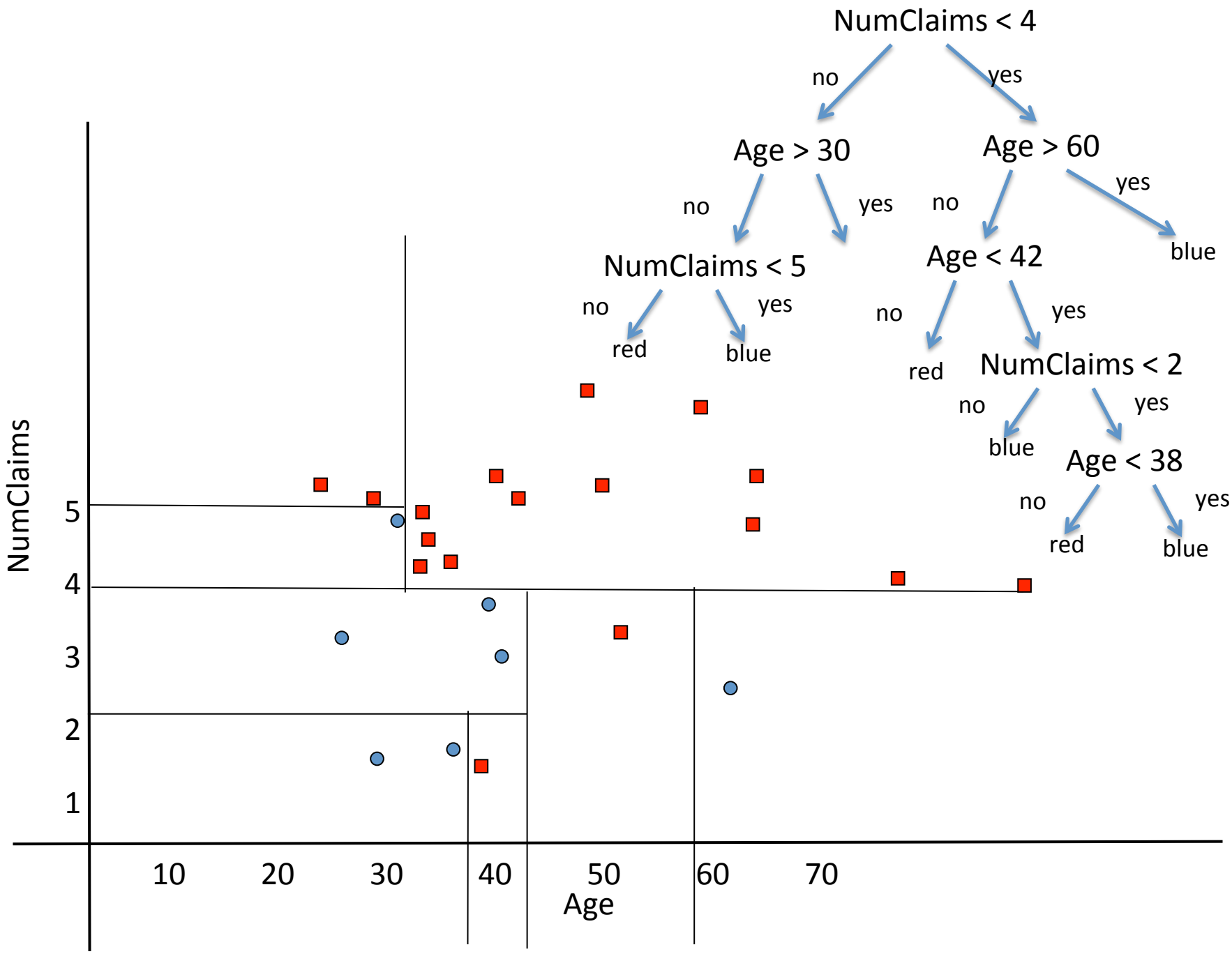
Age	MaxCharlson	Y1Hospital	NumClaims	Y2Hospital
75	4	Yes	4	Yes
12	1	No	1	No
35	0	No	4	Yes
38	1	Yes	1	No
15	3	Yes	5	No
23	2	No	3	No
75	4	No	11	Yes
56	5	Yes	22	?



Age	MaxCharlson	Y1Hospital	NumClaims	Y2Hospital
75	4	Yes	4	Yes
12	1	No	1	No
35	0	No	4	Yes
38	1	Yes	1	No
15	3	Yes	5	No
23	2	No	3	No
75	4	No	11	Yes




NumClaims
Age


NumClaims < 4
no
yes
Age > 30
Age > 60
no
yes
no
yes
NumClaims < 5
Age < 42
blue
no
yes
no
yes
red
blue
red
NumClaims < 2
no
yes
blue
Age < 38
no
yes
red
blue
NumClaims
5
4
3
2
1
10
20
30
40
50
60
70
Age

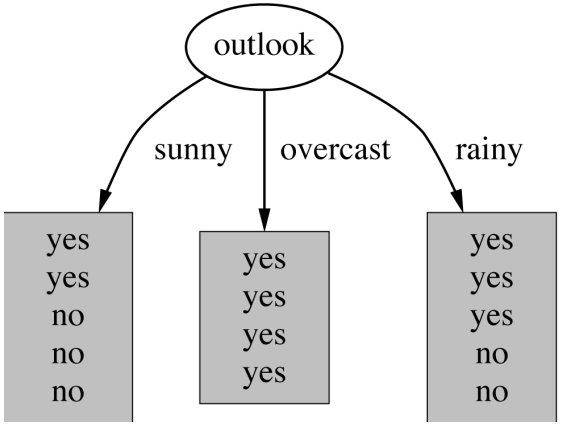
Upshot…

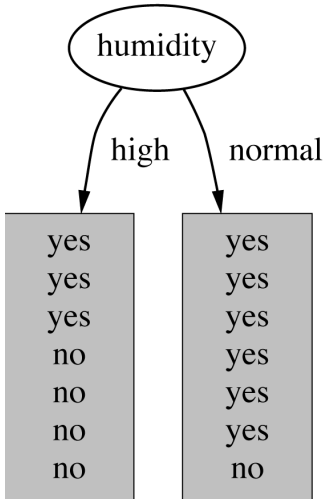
- Need a way to choose between competing splits
- Doesn’t make sense to grow a tree to the bitter end
- Need a method to stop growing the tree indefinitely

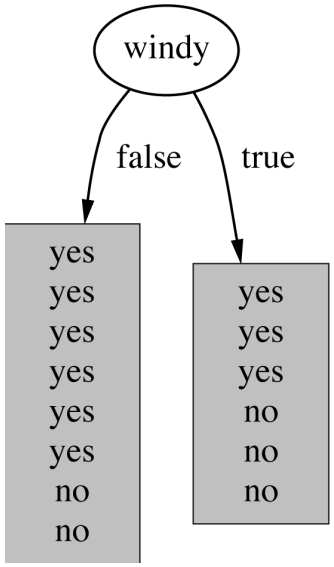
Example: play tennis?

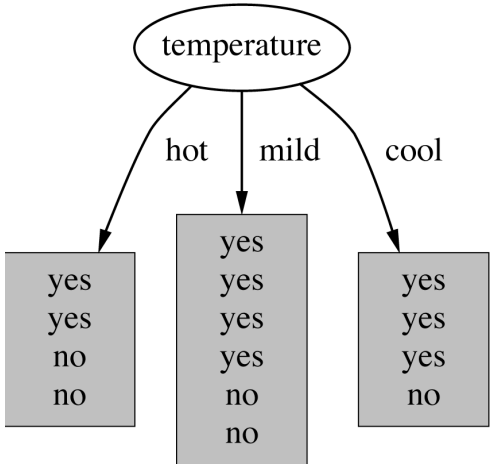
Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

Which attribute to select?









A criterion for attribute selection

- Idea: choose the attribute that produces the greatest gain in "purity"
- Popular *impurity criterion*: *information gain*
- Strategy: choose attribute that results in greatest information gain

Entropy

- Measure of purity used in information theory
- If p_1, p_2,…,p_J are non-negative numbers that add to 1, then the entropy of the numbers is:

$$-\sum_{i=1}^{J} p_i \log p_i$$

- Entropy(1/2,1/2) = -1/2log(1/2) – 1/2log(1/2)=0.69
- Entropy(9/10,1/10) = -9/10log(9/10)-1/10log(1/10)=0.33
- Entropy(1,0) = -1log(1) – 0log(0) = 0

Note: log(0) is not defined, but we evaluate 0*log(0) as zero

Example: attribute “Outlook”, 2

- “Outlook” = “Sunny”:

$$\text{info}([2,3]) = \text{entropy}(2/5,3/5) = -2/5\log(2/5) - 3/5\log(3/5) = 0.971\,\text{bits}$$

- “Outlook” = “Overcast”:

$$\text{info}([4,0]) = \text{entropy}(1,0) = -1\log(1) - 0\log(0) = 0\,\text{bits}$$

- “Outlook” = “Rainy”:

$$\text{info}([3,2]) = \text{entropy}(3/5,2/5) = -3/5\log(3/5) - 2/5\log(2/5) = 0.971\,\text{bits}$$

- Expected information for attribute:

$$\text{info}([3,2],[4,0],[3,2]) = (5/14)\times 0.971 + (4/14)\times 0 + (5/14)\times 0.971$$
$$= 0.693\,\text{bits}$$

Computing the information gain

- Information gain:

(information before split) – (information after split)

$$\text{gain("Outlook")} = \text{info}([9,5]) - \text{info}([2,3],[4,0],[3,2]) = 0.940 - 0.693$$
$$= 0.247 \text{ bits}$$

- Compute for attribute “Humidity”

Example: attribute “Humidity”

- “Humidity” = “High”:

$$\text{info}([3,4]) = \text{entropy}(3/7,4/7) = -3/7\log(3/7) - 4/7\log(4/7) = 0.985 \text{ bits}$$

- “Humidity” = “Normal”:

$$\text{info}([6,1]) = \text{entropy}(6/7,1/7) = -6/7\log(6/7) - 1/7\log(1/7) = 0.592 \text{ bits}$$

- Expected information for attribute:

$$\text{info}([3,4],[6,1]) = (7/14)\times 0.985 + (7/14)\times 0.592 \quad = 0.79 \text{ bits}$$

- Information Gain:

$$\text{info}([9,5]) - \text{info}([3,4],[6,1]) = 0.940 - 0.788 = 0.152$$

Computing the information gain

- Information gain:

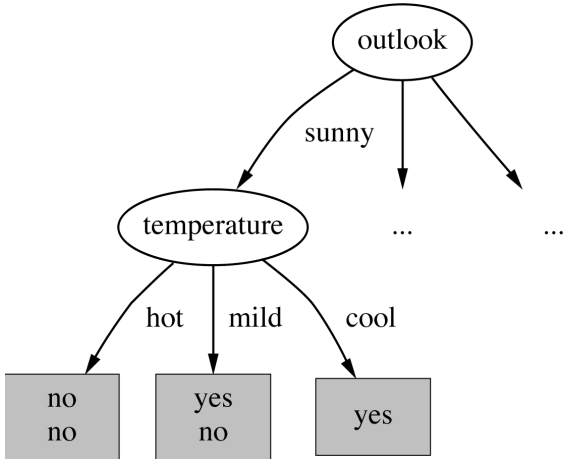
(information before split) – (information after split)

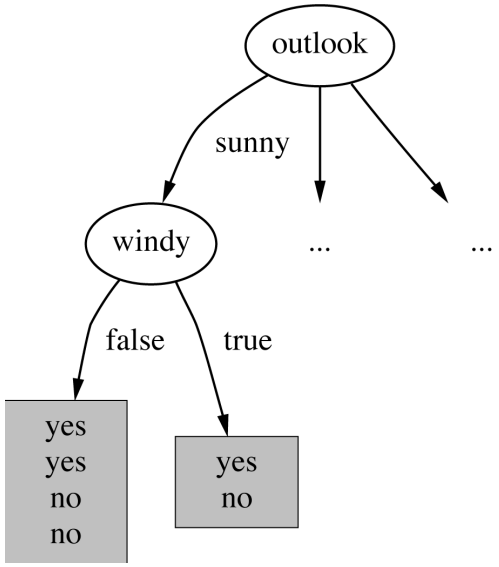
$$\text{gain("Outlook")} = \text{info}([9,5]) - \text{info}([2,3],[4,0],[3,2]) = 0.940 - 0.693$$
$$= 0.247 \text{ bits}$$

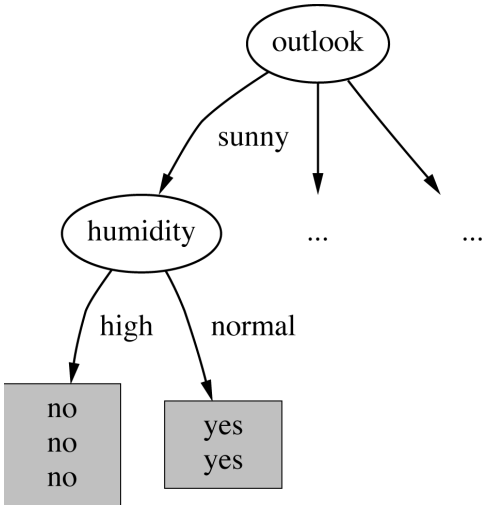
- Information gain for attributes from weather data:

$$\text{gain("Outlook")} = 0.247 \text{ bits}$$
$$\text{gain("Temperature")} = 0.029 \text{ bits}$$
$$\text{gain("Humidity")} = 0.152 \text{ bits}$$
$$\text{gain("Windy")} = 0.048 \text{ bits}$$

Continuing to split







$$\text{gain("Humidity")} = 0.971 \text{ bits}$$

$$\text{gain("Temperature")} = 0.571 \text{ bits}$$

$$\text{gain("Windy")} = 0.020 \text{ bits}$$

Stop the growth!

- "Purest" tree splits to the bitter end because every leaf is then totally pure

- Instead, we would like a tree that makes the most accurate predictions on *future* data

- Idea:
  - split the data into two parts, say 90% and 10%;
  - use 90% to grow tree
  - stop growing when predictions on the 10% stop improving
  - R does this for you

```
library(DAAG)

Scrambled <- sample(1:4601)

spam.train <- spam7[Scrambled[1:4000], ]

spam.test <- spam7[Scrambled[4001:4601], ]

boxplot(split(spam.train$crl.tot, spam.train$yesno))
```

```
> names(spam.train)
[1] "crl.tot" "dollar"  "bang"    "money"   "n000"    "make"
[7] "yesno"
```

- `crl.tot`, total length of words that are in capitals;
- `dollar`, the frequency of the $ symbol, as a percentage of all characters;
- `bang`, the frequency of the ! symbol, as a percentage of all characters;
- `money`, frequency of the word "money", as a percentage of all words;
- `n000`, frequency of the character string "000", as a percentage of all words;
- `make`, frequency of the word "make", as a percentage of all words.

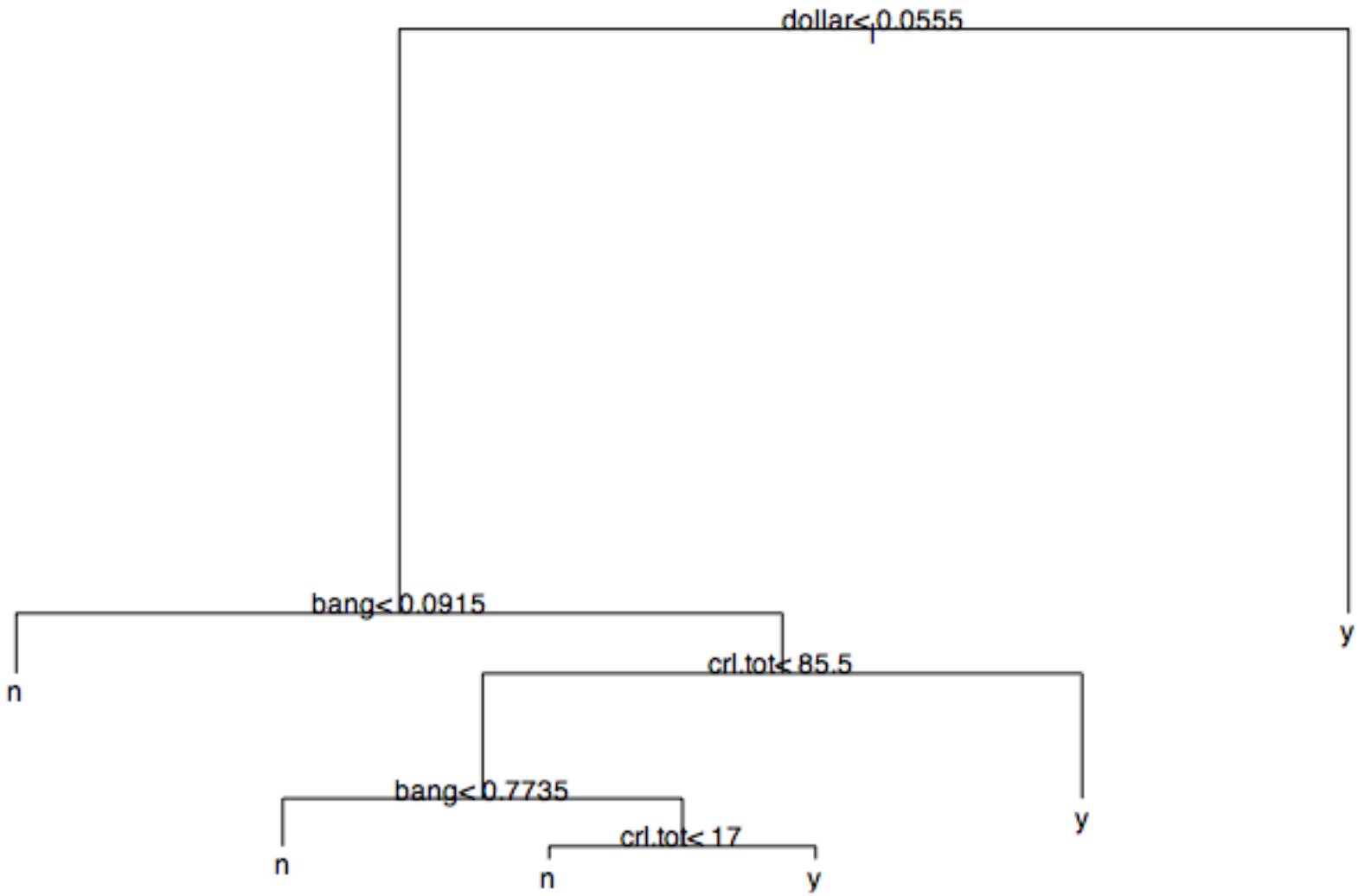
```
library(rpart)

spam.rpart <- rpart(yesno ~ crl.tot + dollar + bang
+ money + n000 + make, data=spam.train,
control=rpart.control(xval=10,cp=0.00001),
method="class")

plot(spam.rpart)

text(spam.rpart)

printcp(spam.rpart)
```


dollar< 0.0555
bang< 0.0915
y
n
crl.tot< 85.5
bang< 0.7735
y
crl.tot< 17
n
n
y

```
> printcp(spam.rpart)

Classification tree:
rpart(formula = yesno ~ crl.tot + dollar + bang + money + n000 +
    make, data = spam7, method = "class")

Variables actually used in tree construction:
[1] bang    crl.tot dollar

Root node error: 1813/4601 = 0.39404

n= 4601

        CP nsplit rel error  xerror     xstd
1 0.476558      0   1.00000 1.00000 0.018282
2 0.075565      1   0.52344 0.55764 0.015492
3 0.011583      3   0.37231 0.38224 0.013382
4 0.010480      4   0.36073 0.38279 0.013390
5 0.010000      5   0.35025 0.38003 0.013350
```

```
> spam.rpart <- rpart(formula = yesno ~ crl.tot + dollar + bang + money + n000 + make,
method="class", data=spam.train, control=rpart.control(xval=10,cp=0.00001))
> printcp(spam.rpart)

Classification tree:
rpart(formula = yesno ~ crl.tot + dollar + bang + money + n000 +
    make, data = spam.train, method = "class", control = rpart.control(xval = 10,
    cp = 1e-05))

Variables actually used in tree construction:
[1] bang    crl.tot dollar  make    money   n000

Root node error: 1552/4000 = 0.388
```

	CP	nsplit	rel error	xerror	xstd
1	0.45747423	0	1.00000	1.00000	0.019858
2	0.05637887	1	0.54253	0.56637	0.016874
3	0.05219072	3	0.42977	0.48582	0.015938
4	0.01030928	4	0.37758	0.39240	0.014640
5	0.00966495	5	0.36727	0.38402	0.014511
6	0.00596005	6	0.35760	0.36791	0.014255
7	0.00547680	10	0.33376	0.36082	0.014140
8	0.00451031	12	0.32281	0.35052	0.013969
9	0.00322165	15	0.30928	0.34021	0.013794
10	0.00257732	17	0.30284	0.34021	0.013794
11	0.00193299	18	0.30026	0.34085	0.013805
12	0.00161082	27	0.28286	0.34214	0.013827
13	0.00150344	29	0.27964	0.34149	0.013816
14	0.00080541	32	0.27513	0.34536	0.013882
15	0.00064433	36	0.27191	0.34601	0.013893
16	0.00042955	47	0.26482	0.34794	0.013926
17	0.00032216	52	0.26224	0.34665	0.013904
18	0.00027614	60	0.25966	0.35052	0.013969
19	0.00001000	67	0.25773	0.35374	0.014023

```
spam.rpart.opt<-prune(spam.rpart,cp=0.004)

my.predictions <- predict(spam.rpart.opt, spam.test)

my.predictions[1:10]
 [1] 0.92592593 0.89619540 0.89619540 0.04642409
0.82926829 0.04761905 0.89619540 0.04642409 0.04642409
0.04642409

bin.pred<-1*(my.predictions > 0.4)
```

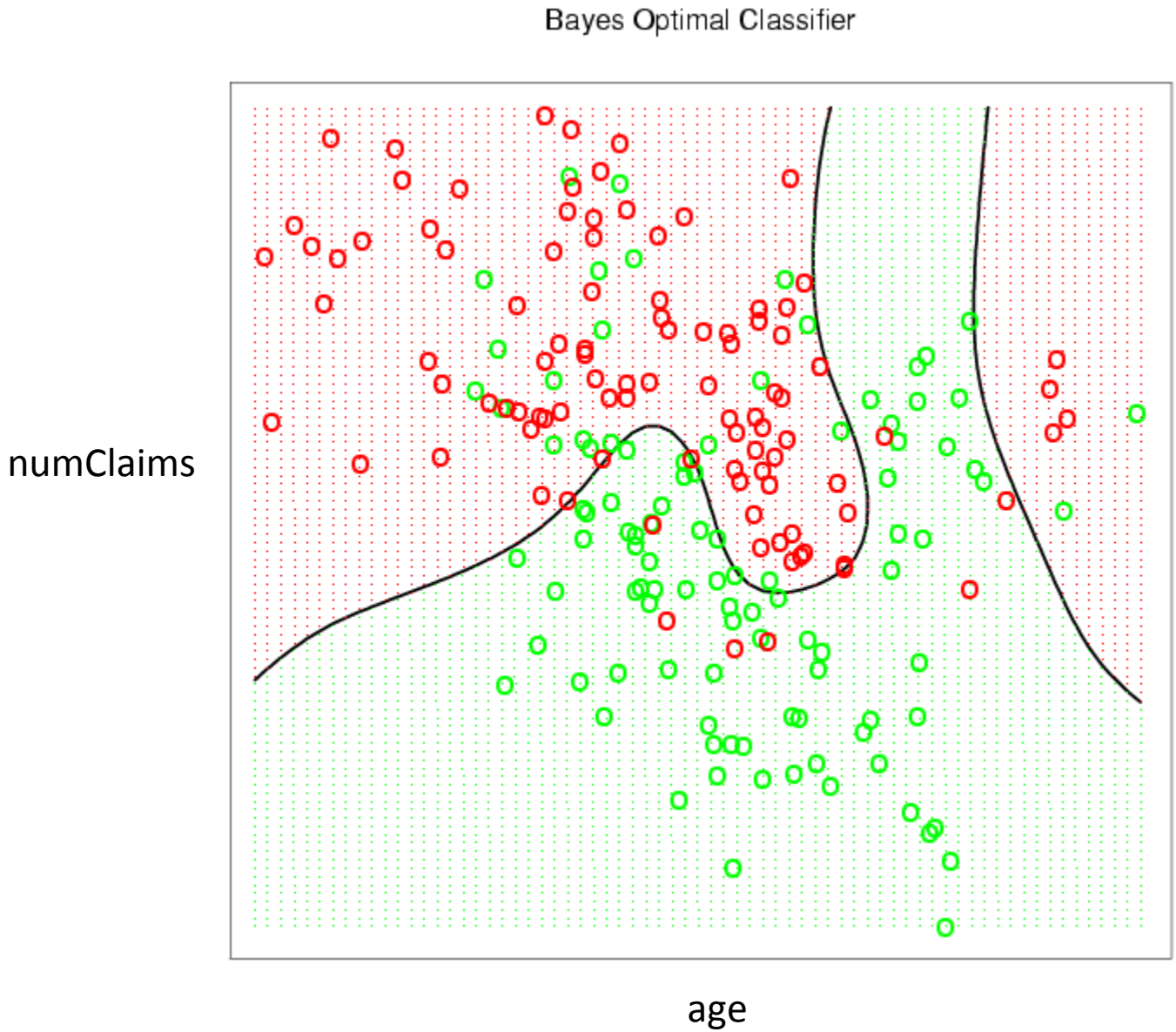
myModel<-rpart(as.logical(Y2_inhospital)~num_claims+sex, data=Train, control=rpart.control(xval=10,cp=0.001))

my.pred <- data.frame(memberid=my.test$memberid,Y2_inhospital=bin.pred)
submit(my.pred,"team name","rpart(Y2_hospital~…)")

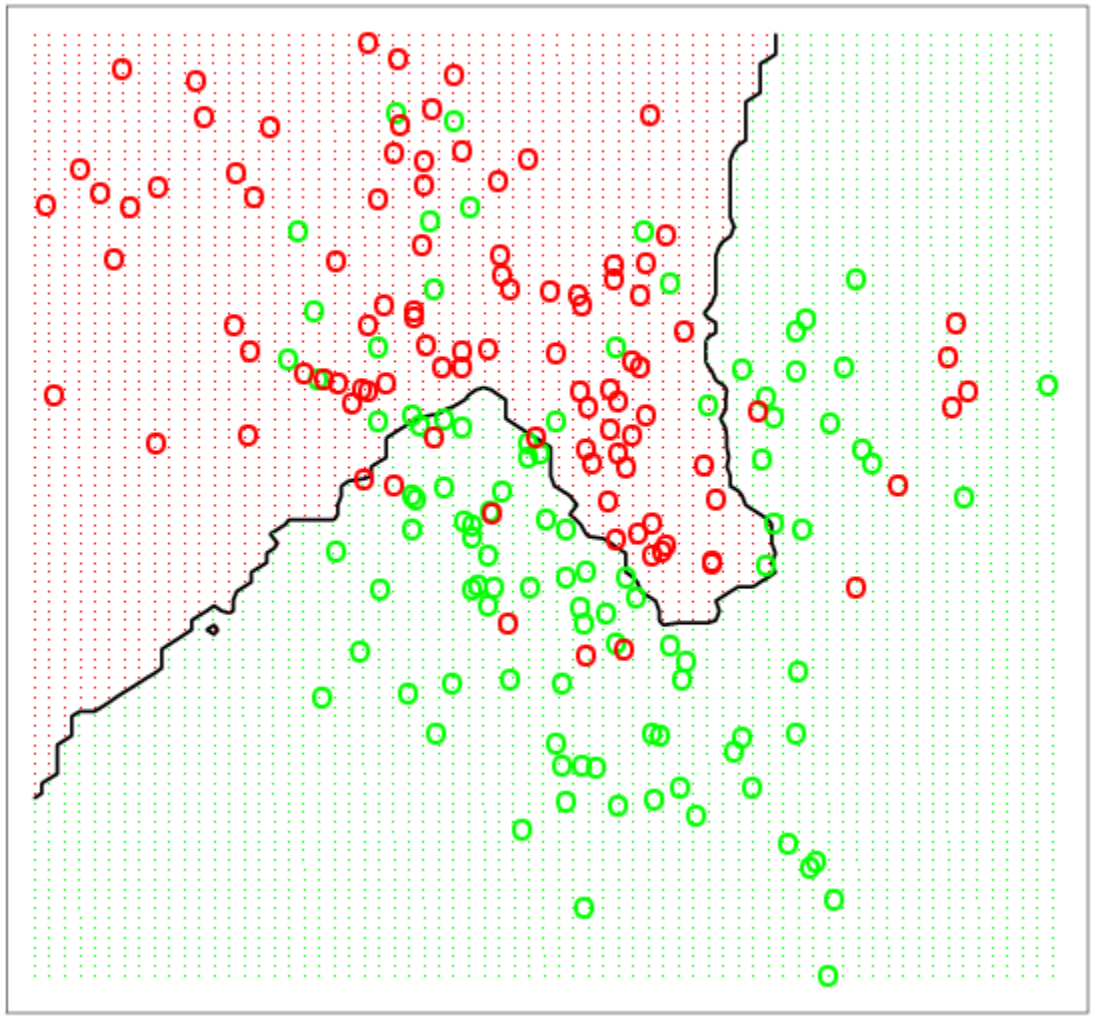
Nearest Neighbor Methods

Nearest Neighbor Methods

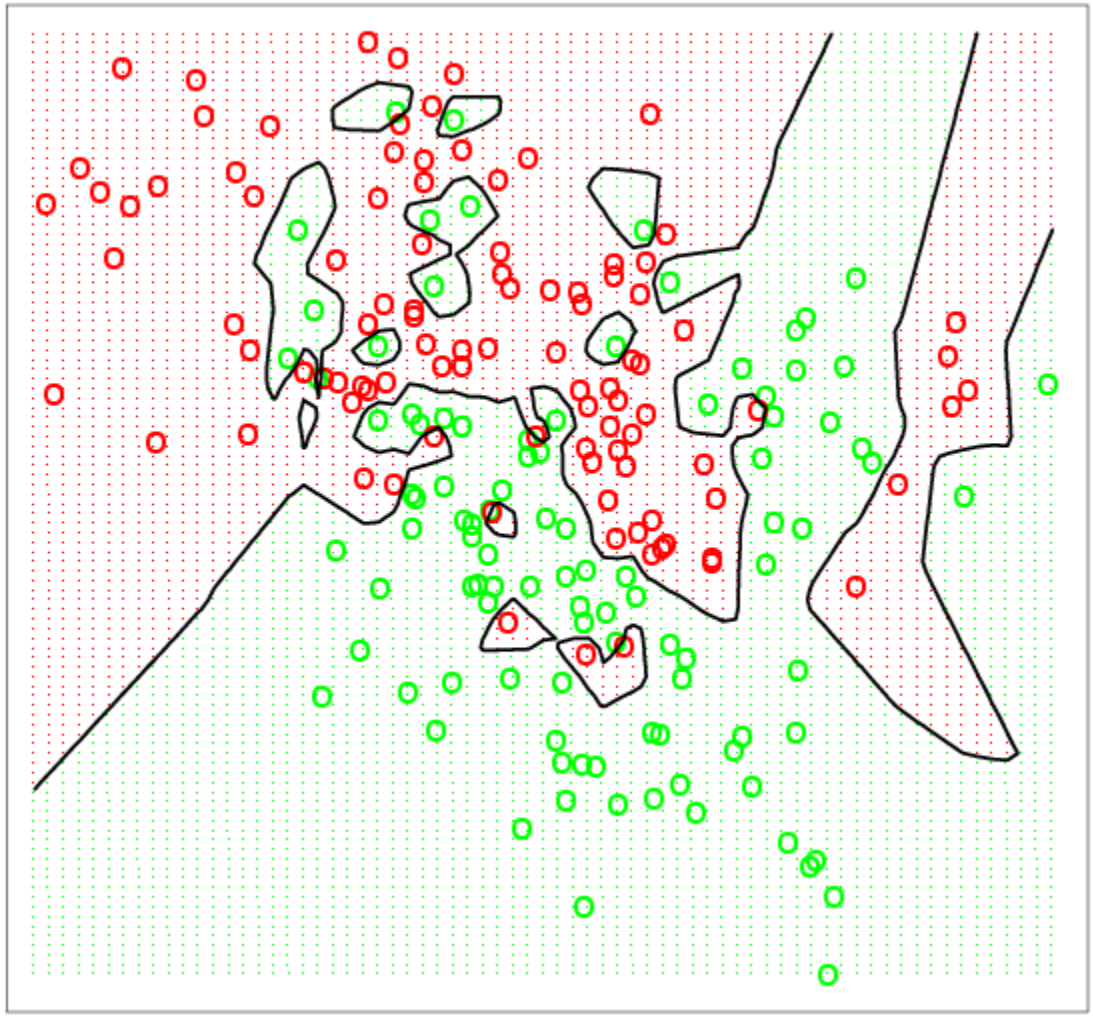
- *k*-NN assigns an unknown object to the most common class of its *k* nearest neighbors
- Choice of *k*?
- Choice of distance metric?

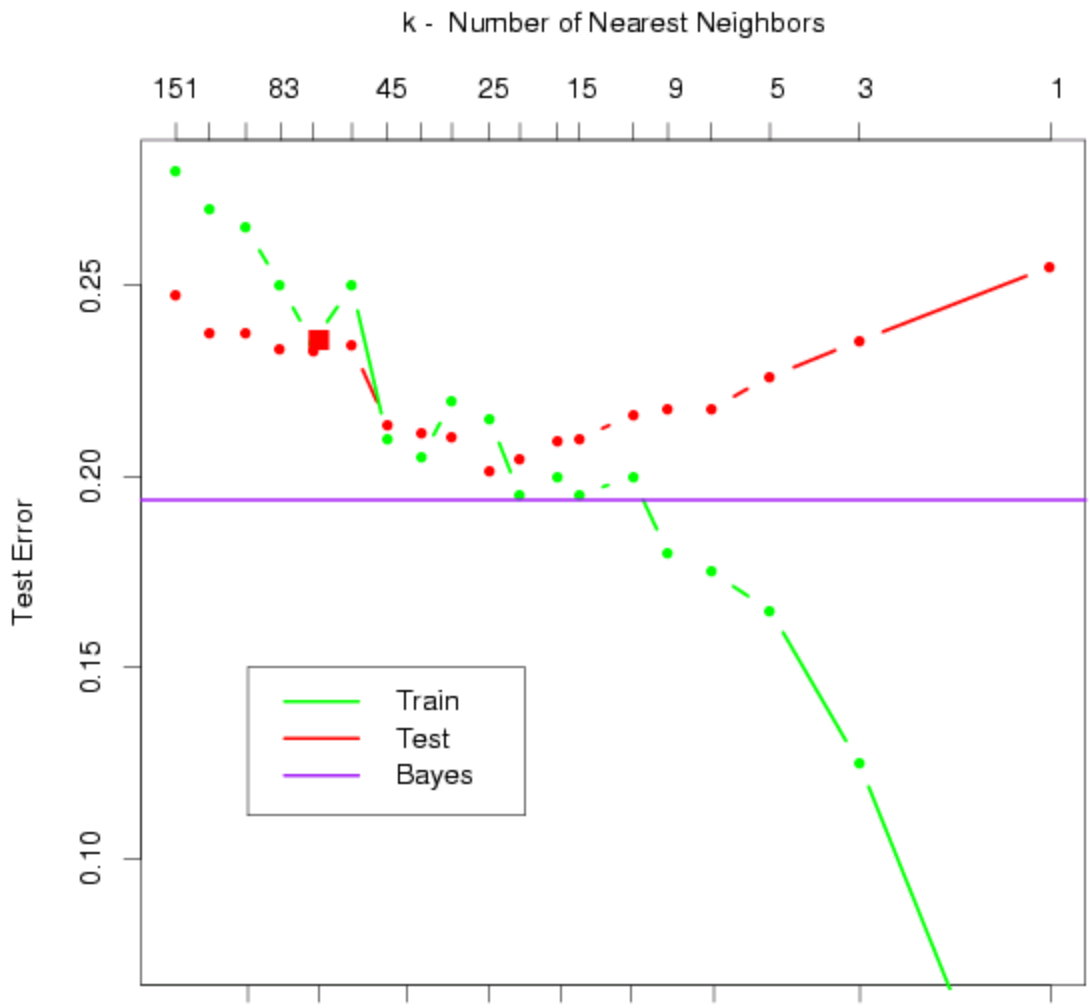

Bayes Optimal Classifier
numClaims
age

15-Nearest Neighbor Classifier



1-Nearest Neighbor Classifier




k - Number of Nearest Neighbors
151
83
45
25
15
9
5
3
1
Test Error
0.25
0.20
0.15
0.10
Train
Test
Bayes

```
Library(kknn)
myModel <- kknn(yesno~dollar+bang+money,spam.train,spam.test,k=3)
> names(myModel)
[1] "fitted.values" "CL"          "W"           "D"           "prob"        "response"    "distance"
"call"        "terms"
> myModel$fitted.values
  [1] n y y n y n y y y y n n n n n y n n y n n n y n n n y n y n n y n y n n n n n n n n y y n n n n n n n y n n
n n y n n n n y y y n n n n y n y n y n n n n n y n
 [78] n y n y n n n n n n n n y y n n y n n n y n n y y n y y y n n n y n n y n y n n n y y n n n n n n y
n n n n n n n y n n n n n n y y y n y y y n n n n y n n n
[155] y n y y n n y n y y n y n n y n y n n n n n n n n n y n n y n n y n y y n n n n n n n y y y n n y n n n
y n y n y n n n y n n y n y n y n y y n n n n n n y y
[232] n y n n y n y n n n n y y n n y n y n y n n y n n y n y n n n y n n n y n n y y n y y n n n y y y n
y n n y n n y n y y y y y y y y n y n n n n n y n y y y
[309] n y n n y n y y y y y n y n n y y n y y n n y y y y y y y y n n n n n y y n y n n n y y n n n n y y y
y n n y y y n n n n y y y y n y y y y n y n n n y n
[386] y n n n y y y n n y y n n n n n y n n y n n y n n n y y y n n n n y y n n y n n n n y y y n n y n n
y y y y y n y n n n n n n n n n n n n y n n n n n n n y
[463] n n n y y n n n n n n n y n n n y n n y n n y y n n n y y y n n y n n n y n y y y y n y n n y y n n
y y n y y n y n n y y n n n n n y n y n n n n n y n n
[540] y n y n n n n y n n y y n n n y y n y n n n y y y n y y n y n y y y y y y n n n y y n n y y n y n n n n
y n y n y y y n y y y
```

```
Library(FNN)
myModel <- knn(Train[,-163], Test[,-163], Train[,163], k=3,
algorithm="kd_tree",prob=TRUE)

Error in knn(Train[, -163], Test[, -163], Train[, 163], k = 3, algorithm =
"kd_tree",  :
  no missing values are allowed

Train2 <- Train[apply(is.na(Train),1,sum)==0,]
Test2 <- Test[apply(is.na(Test),1,sum)==0,]

Train2[,1] <- as.numeric(Train2[,1])
Train2[,3] <- as.numeric(Train2[,3])
Test2[,1] <- as.numeric(Test2[,1])
Test2[,3] <- as.numeric(Test2[,3])
```

```
m <- dim(Train2)[1]
val <- sample(1:m, size = round(m/10), replace = FALSE)
my.train <- Train2[-val,]
my.test <- Train2[val,]

for (k in1:100) {
	my.model <- knn(my.train[,-163], my.test[,-163], my.train[,163], k=k,
algorithm="kd_tree",prob=TRUE)
	attributes(my.model) <- NULL
	bin.pred <- my.model - 1
	cat("k=",k," accuracy: ",  (sum((my.model==1)&(!my.test[,163])) +
sum((my.model==2)&my.test[,163]))/length(my.model),"\n" )
}
```