Feature Spaces

Feature Spaces

- Features can be much more complex

Feature Spaces

- Features can be much more complex
- Drawn from bigger discrete set

Feature Spaces

- Features can be much more complex

- Drawn from bigger discrete set
  - If set is unordered (4 different makes of cars, for example), use binary attributes to encode the values (1000, 0100, 0010, 0001)

Feature Spaces

- Features can be much more complex

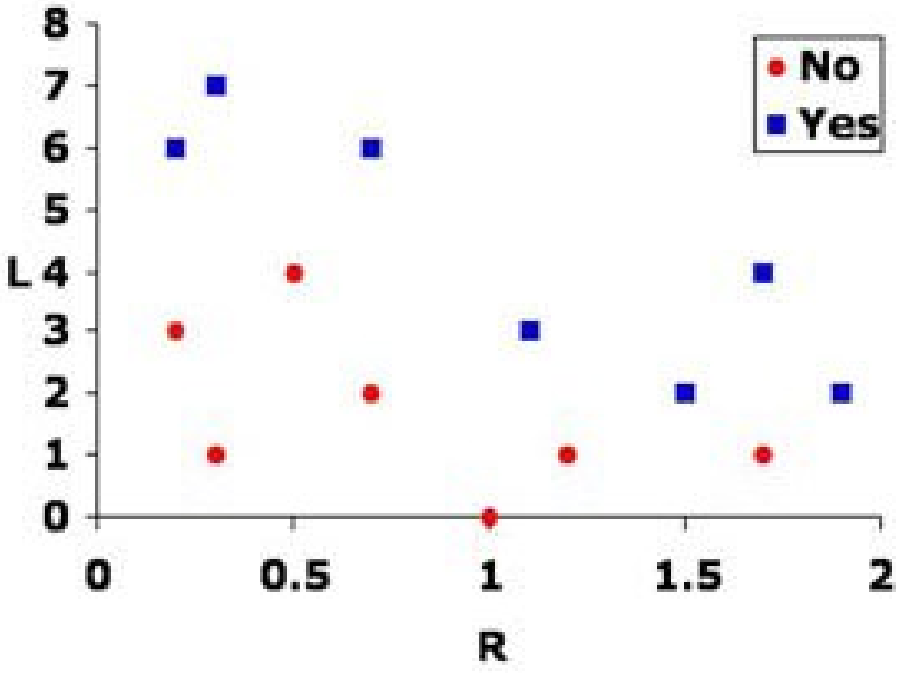
- Drawn from bigger discrete set
  - If set is unordered (4 different makes of cars, for example), use binary attributes to encode the values (1000, 0100, 0010, 0001)
  - If set is ordered, treat as real-valued

Feature Spaces

- Features can be much more complex
- Drawn from bigger discrete set
  - If set is unordered (4 different makes of cars, for example), use binary attributes to encode the values (1000, 0100, 0010, 0001)
  - If set is ordered, treat as real-valued
- Real-valued: bias that inputs whose features have "nearby" values ought to have "nearby" outputs

Predicting Bankruptcy

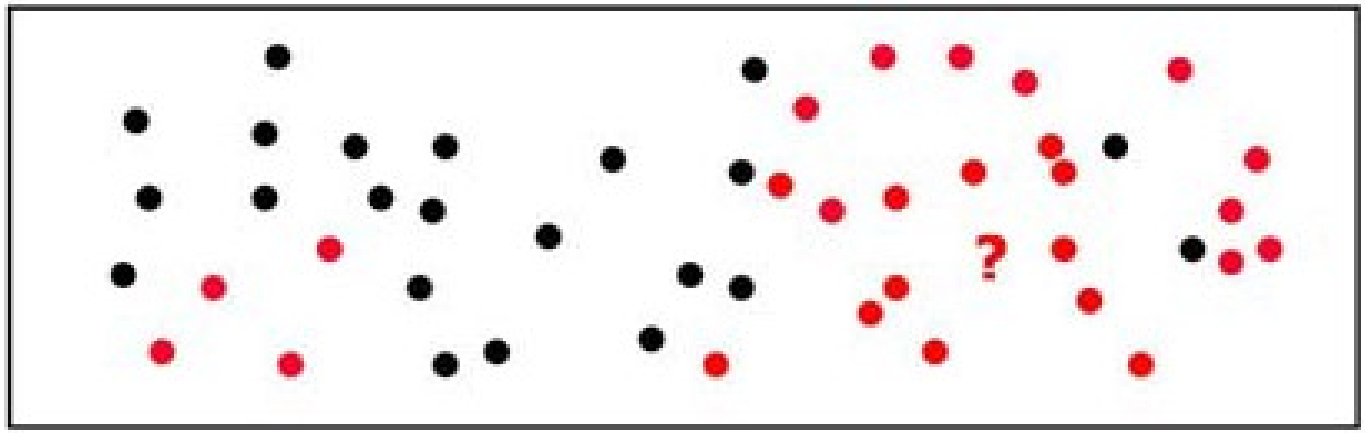
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1.0	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



L: #late payments / year
R: expenses / income

Love thy Nearest Neighbor

- Remember all your data
- When someone asks a question,
  - find the nearest old data point
  - return the answer associated with it



What do we mean by "Nearest"?

- Need a distance function on inputs
- Typically use Euclidean distance (length of a straight line between the points)

$$D(x^i, x^k) = \sqrt{\sum_j (x_j^i - x_j^k)^2}$$

What do we mean by "Nearest"?

- Need a distance function on inputs
- Typically use Euclidean distance (length of a straight line between the points)

$$D(x^l, x^k) = \sqrt{\sum_j (x_j^l - x_j^k)^2}$$

- Distance between character strings might be number of edits required to turn one into the other

Scaling

- What if we're trying to predict a car's gas mileage?
  - f_1 = weight in pounds
  - f_2 = number of cylinders

Scaling

- What if we're trying to predict a car's gas mileage?
  - f_1 = weight in pounds
  - f_2 = number of cylinders
- Any effect of f_2 will be completely lost because of the relative scales

Scaling

- What if we're trying to predict a car's gas mileage?
  - f_1 = weight in pounds
  - f_2 = number of cylinders
- Any effect of f_2 will be completely lost because of the relative scales
- So, re-scale the inputs

Scaling

- What if we're trying to predict a car's gas mileage?
  - f_1 = weight in pounds
  - f_2 = number of cylinders
- Any effect of f_2 will be completely lost because of the relative scales
- So, re-scale the inputs to have mean 0 and variance 1:

$$x' = \frac{x - \bar{x}}{\sigma_x}$$

$\bar{x}$: average

σ_x: standard deviation

Scaling

- What if we're trying to predict a car's gas mileage?
  - f_1 = weight in pounds
  - f_2 = number of cylinders
- Any effect of f_2 will be completely lost because of the relative scales
- So, re-scale the inputs to have mean 0 and variance 1:

$$x' = \frac{x - \bar{x}}{\sigma_x}$$

$\bar{x}$: average

σ_x: standard deviation

- Or, build knowledge in by scaling features differently

Scaling

- What if we're trying to predict a car's gas mileage?
  - f_1 = weight in pounds
  - f_2 = number of cylinders
- Any effect of f_2 will be completely lost because of the relative scales
- So, re-scale the inputs to have mean 0 and variance 1:

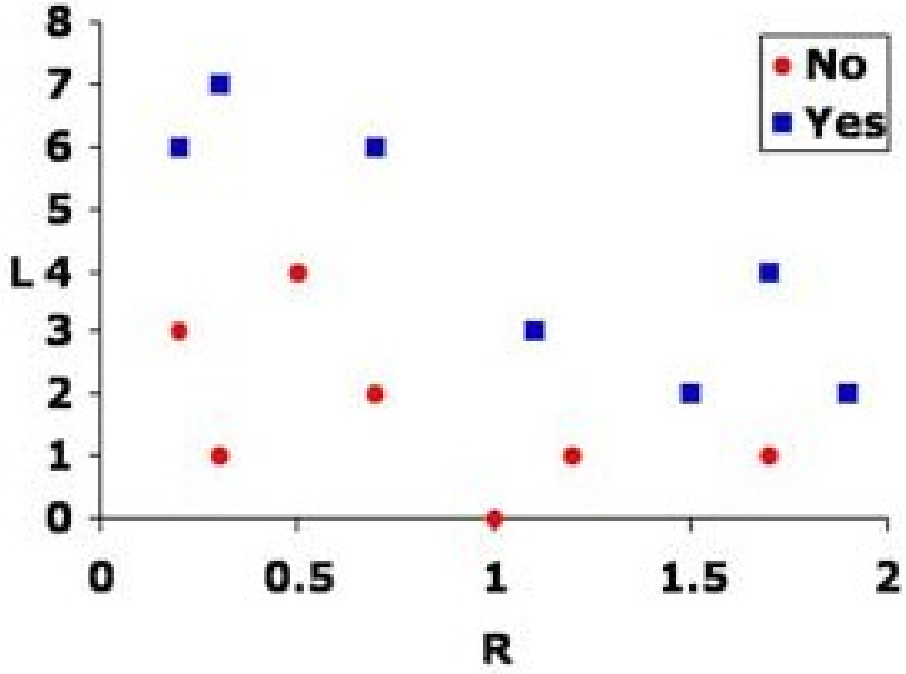
$$x' = \frac{x - \bar{x}}{\sigma_x}$$

$\bar{x}$: average

σ_x: standard deviation

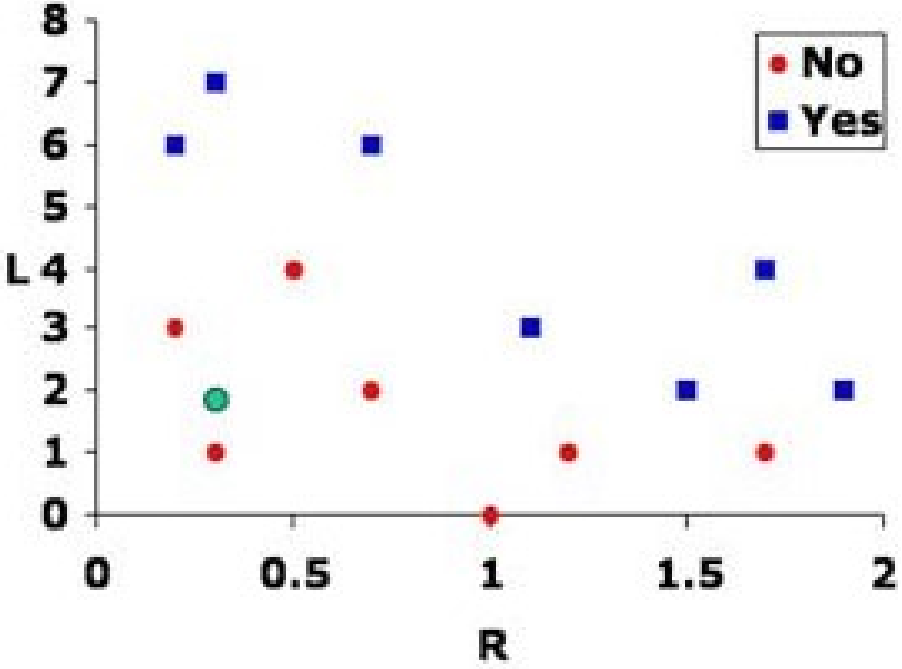
- Or, build knowledge in by scaling features differently
- Or use cross-validation to choose scales

Predicting Bankruptcy



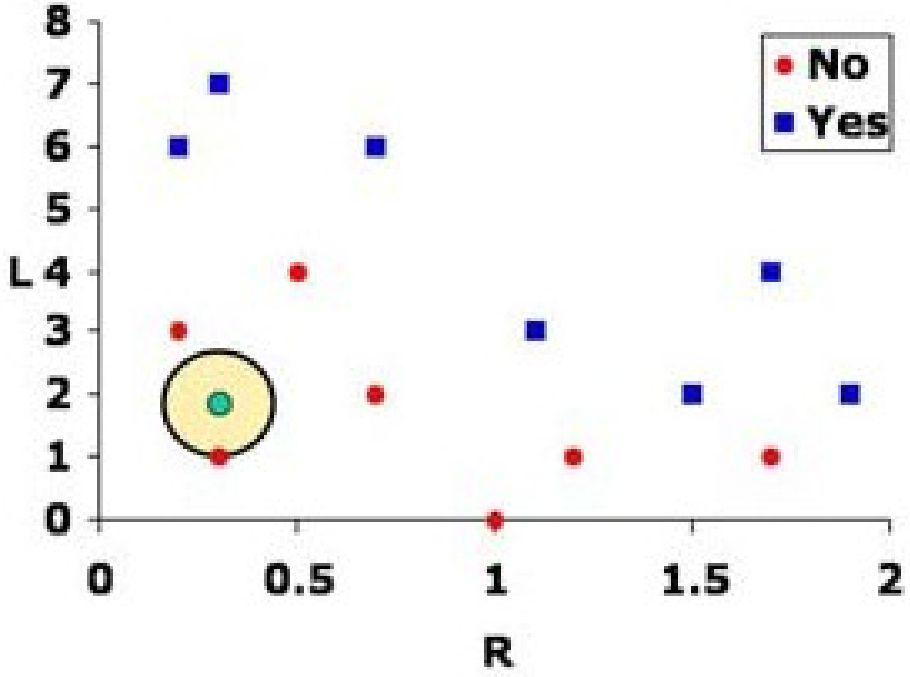
$$D(x^l, x^k) = \sqrt{\sum_j (L^l - L^k)^2 + (5R^l - 5R^k)^2}$$

Predicting Bankruptcy



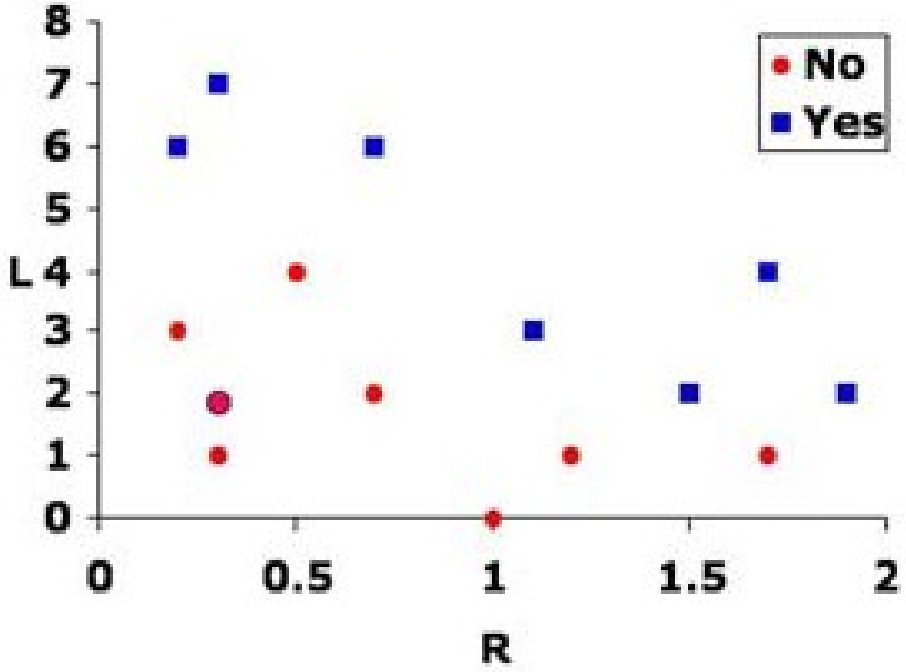
$$D(x^l, x^k) = \sqrt{\sum_j (L^l - L^k)^2 + (5R^l - 5R^k)^2}$$

Predicting Bankruptcy



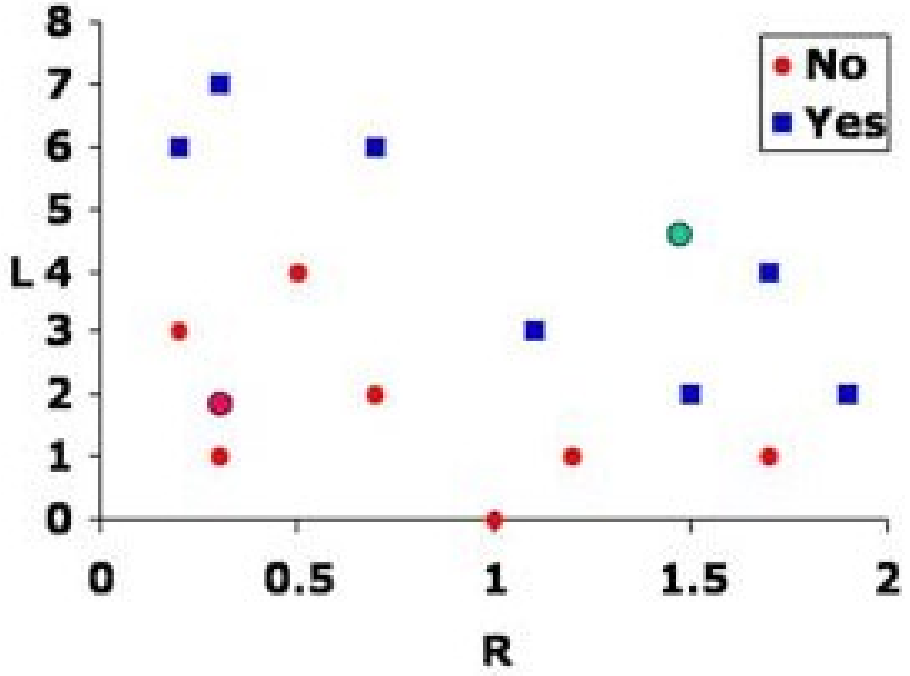
$$D(x^l, x^k) = \sqrt{\sum_j (L^l - L^k)^2 + (5R^l - 5R^k)^2}$$

Predicting Bankruptcy



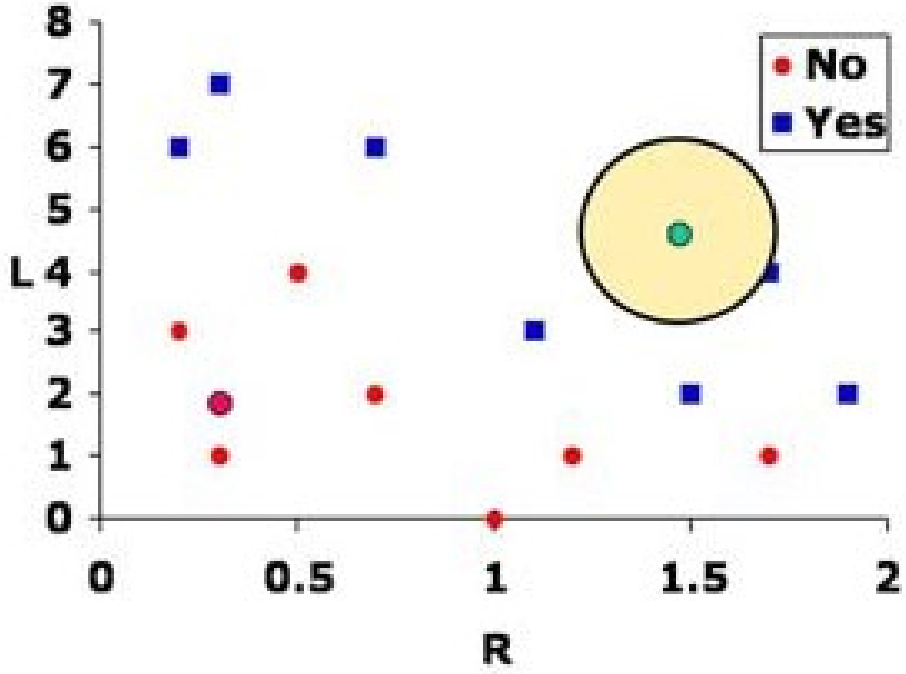
$$D(x^l, x^k) = \sqrt{\sum_j (L^l - L^k)^2 + (5R^l - 5R^k)^2}$$

Predicting Bankruptcy



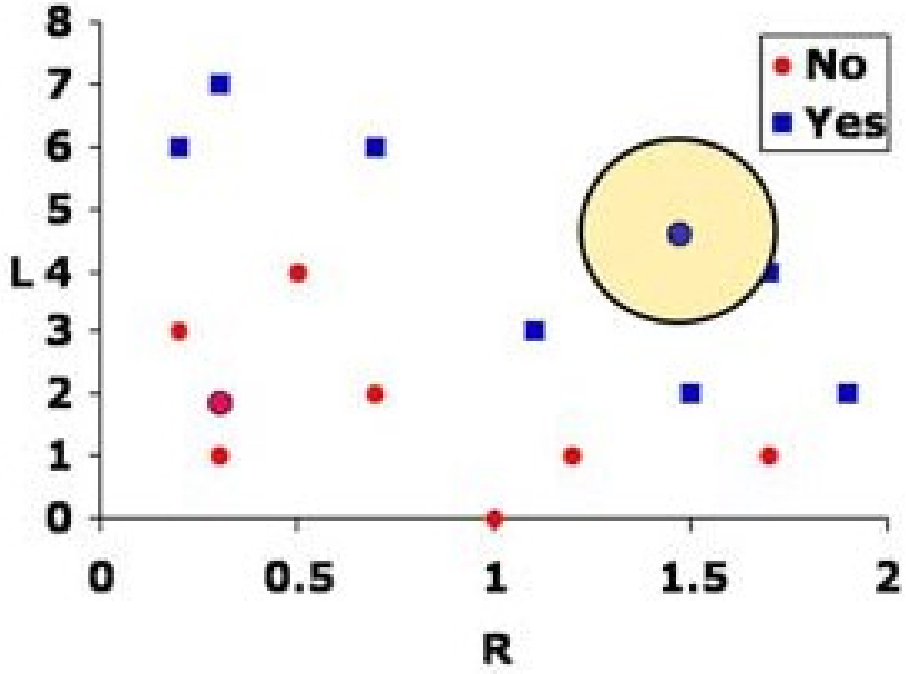
$$D(x^l, x^k) = \sqrt{\sum_j (L^l - L^k)^2 + (5R^l - 5R^k)^2}$$

Predicting Bankruptcy



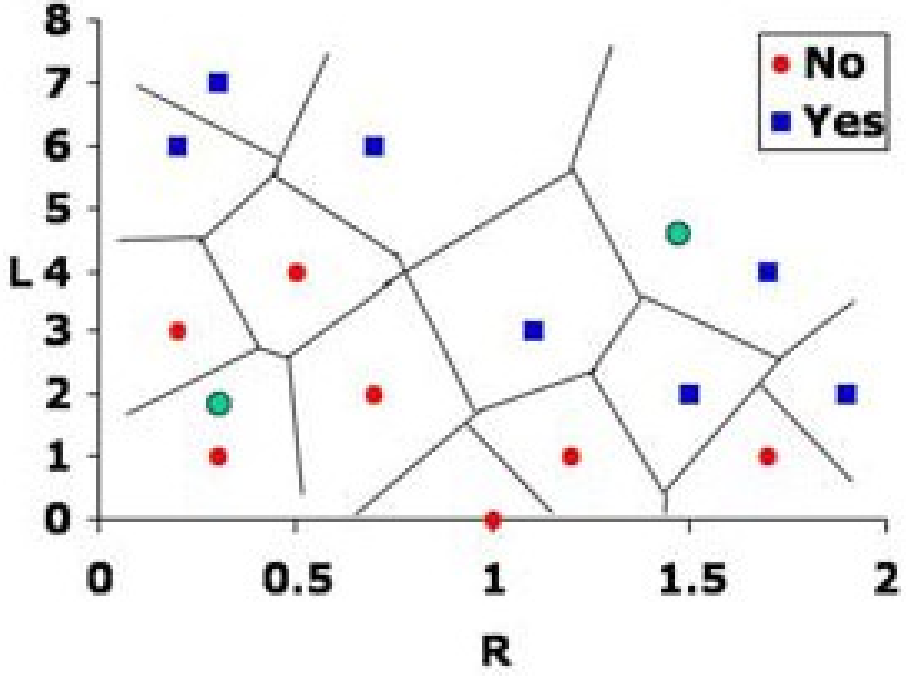
$$D(x^l, x^k) = \sqrt{\sum_j (L^l - L^k)^2 + (5R^l - 5R^k)^2}$$

Predicting Bankruptcy



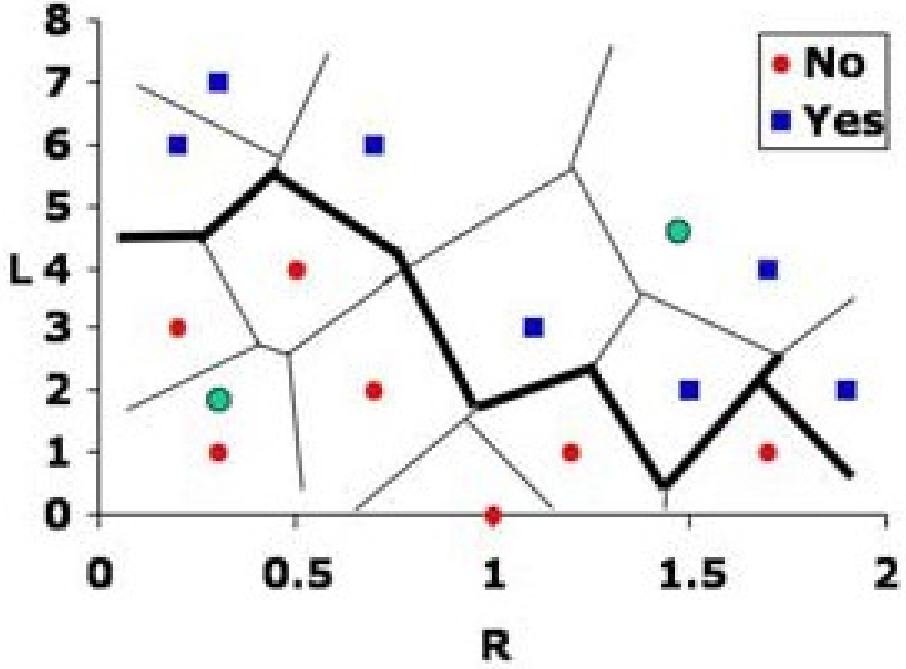
$$D(x^l, x^k) = \sqrt{\sum_j (L^l - L^k)^2 + (5R^l - 5R^k)^2}$$

Hypothesis



$$D(x^l, x^k) = \sqrt{\sum_j (L^l - L^k)^2 + (5R^l - 5R^k)^2}$$

Hypothesis



$$D(x^l, x^k) = \sqrt{\sum_j (L^l - L^k)^2 + (5R^l - 5R^k)^2}$$

Time and Space

- Learning is fast

Time and Space

- Learning is fast
- Lookup takes about m*n computations

Time and Space

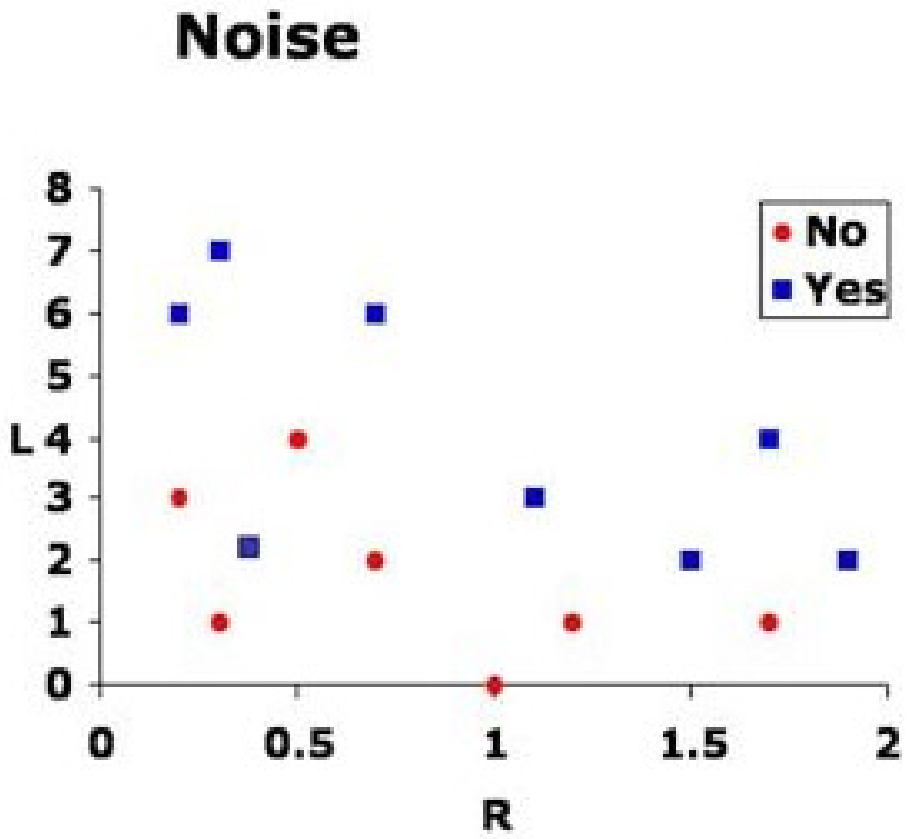
- Learning is fast
- Lookup takes about m*n computations
  - storing data in a clever data structure (KD-tree) reduces this, on average, to log(m)*n

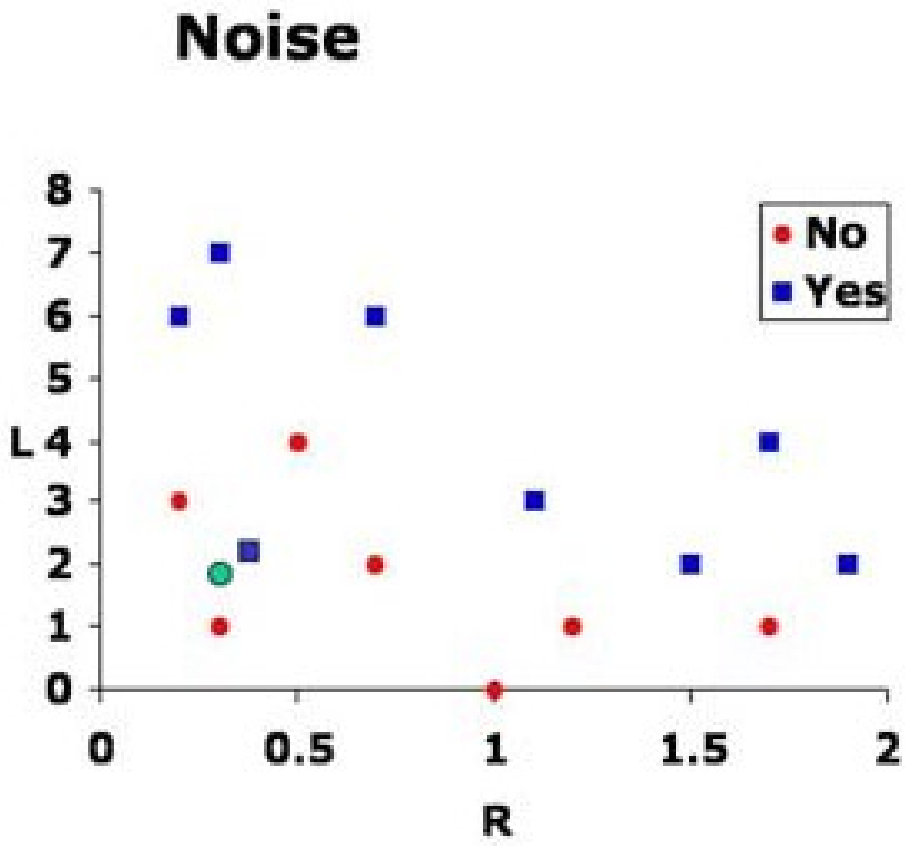
Time and Space

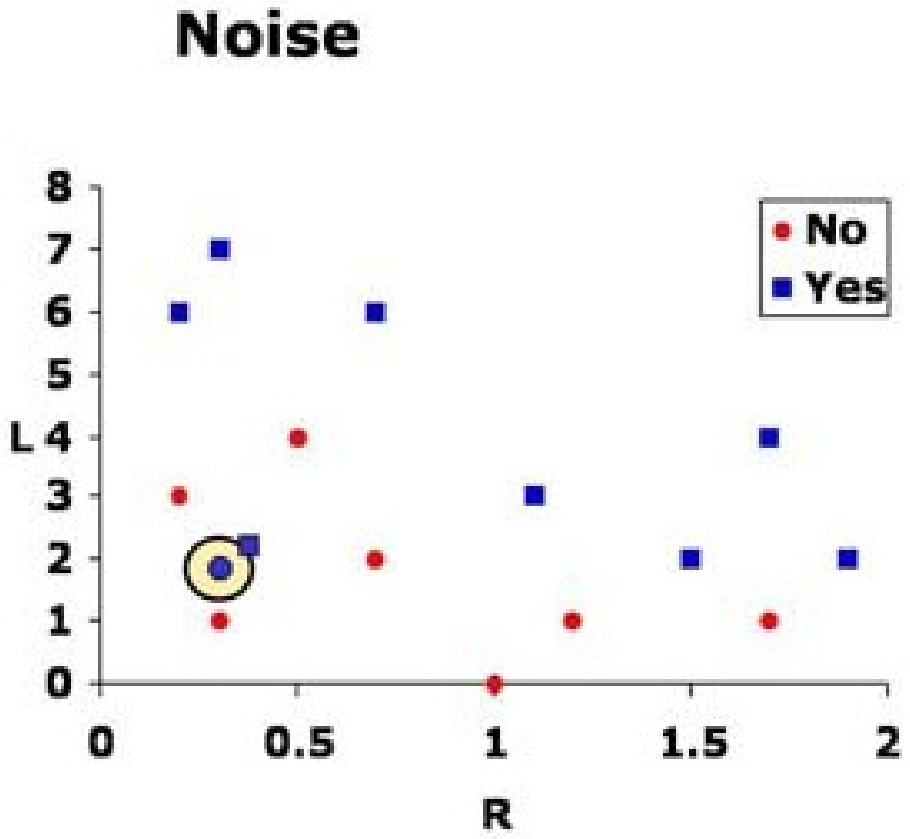
- Learning is fast
- Lookup takes about m*n computations
  - storing data in a clever data structure (KD-tree) reduces this, on average, to log(m)*n
- Memory can fill up with all that data

Time and Space

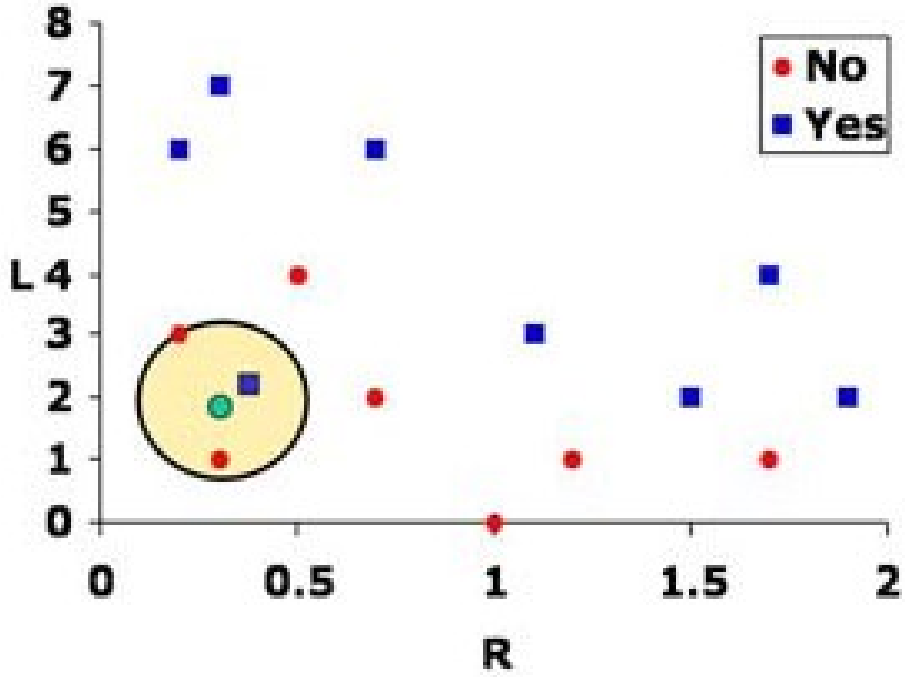
- Learning is fast
- Lookup takes about m*n computations
  - storing data in a clever data structure (KD-tree) reduces this, on average, to log(m)*n
- Memory can fill up with all that data
  - delete points that are far away from the boundary


Noise
No
Yes
L
R
0
1
2
3
4
5
6
7
8
0
0.5
1
1.5
2


Noise
No
Yes
L
R
0
1
2
3
4
5
6
7
8
0
0.5
1
1.5
2

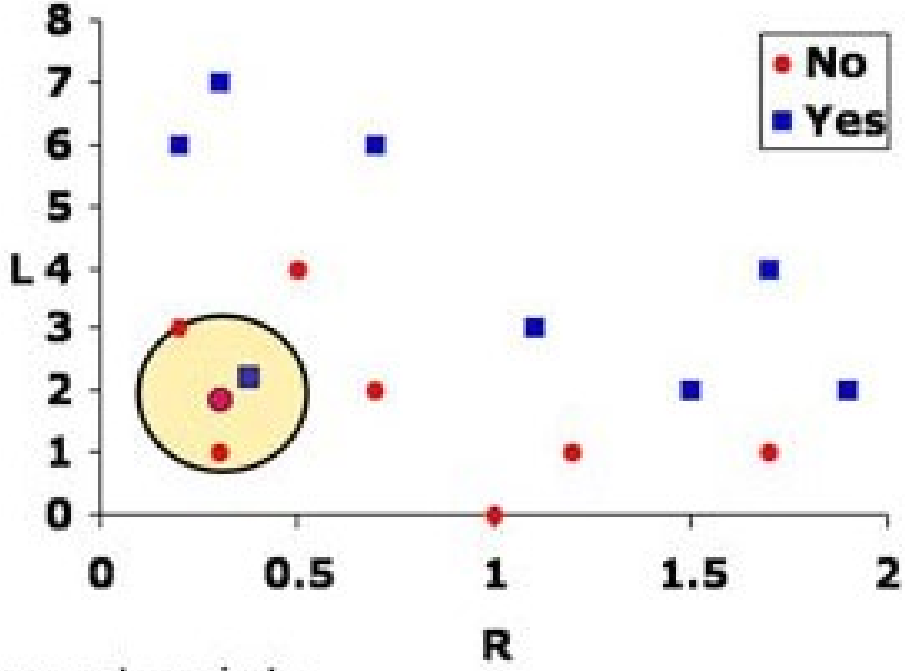

Noise
No
Yes
L
R
0
1
2
3
4
5
6
7
8
0
0.5
1
1.5
2

k-Nearest Neighbor



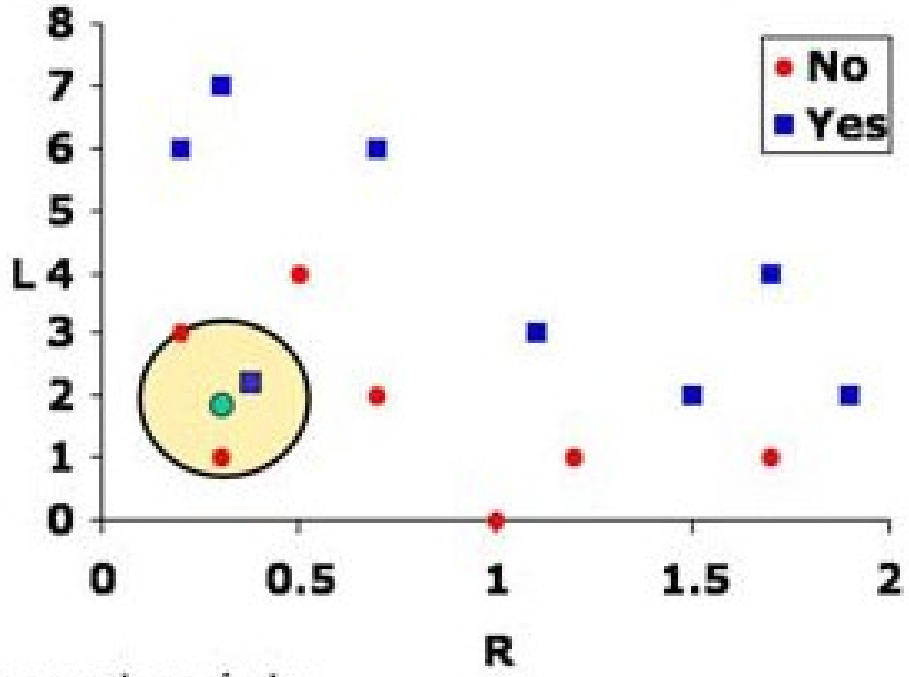
- Find the k nearest points

k-Nearest Neighbor



- Find the k nearest points
- Predict output according to the majority

k-Nearest Neighbor



- Find the k nearest points
- Predict output according to the majority
- Choose k using cross-validation

Curse of Dimensionality

- Nearest neighbor is great in low dimensions (up to about 6)
- As n increases, things get weird:

Curse of Dimensionality

- Nearest neighbor is great in low dimensions (up to about 6)
- As n increases, things get weird:
  - In high dimensions, almost all points are far away from one another
  - They're almost all near the boundaries

Curse of Dimensionality

- Nearest neighbor is great in low dimensions (up to about 6)
- As n increases, things get weird:
  - In high dimensions, almost all points are far away from one another
  - They're almost all near the boundaries
- Imagine sprinkling data points uniformly within a 10-dimensional unit cube
  - To capture 10% of the points, you'd need a cube with sides of length .63!

Curse of Dimensionality

- Nearest neighbor is great in low dimensions (up to about 6)
- As n increases, things get weird:
  - In high dimensions, almost all points are far away from one another
  - They're almost all near the boundaries
- Imagine sprinkling data points uniformly within a 10-dimensional unit cube
  - To capture 10% of the points, you'd need a cube with sides of length .63!
- Cure: feature selection or more global models

Test Domains

Test Domains

- Heart Disease: predict whether a person has significant narrowing of the arteries, based on tests
  - 26 features
  - 297 data points

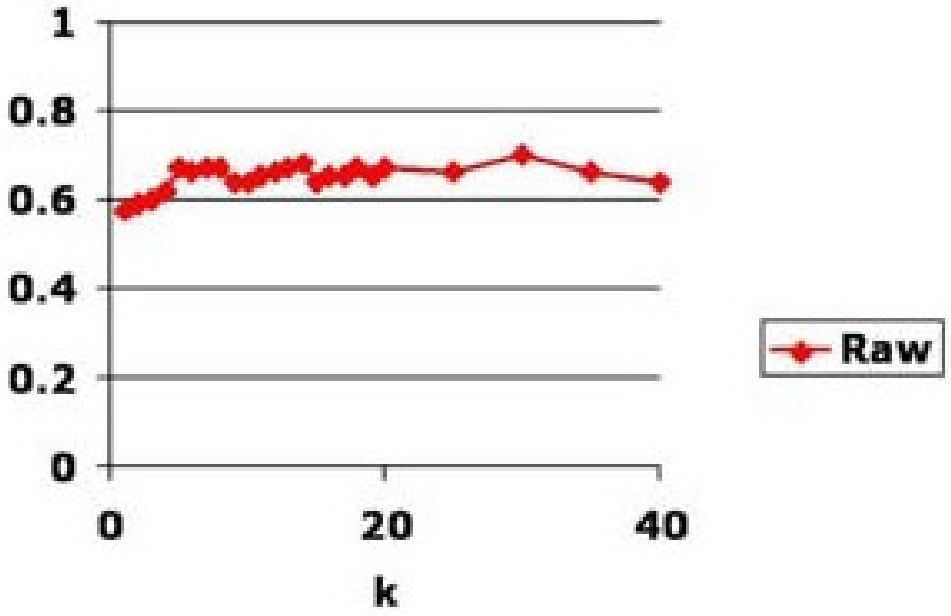
Test Domains

- Heart Disease: predict whether a person has significant narrowing of the arteries, based on tests
  - 26 features
  - 297 data points

- Auto MPG: predict whether a car gets more than 22 miles per gallon, based on attributes of car
  - 12 features
  - 385 data points

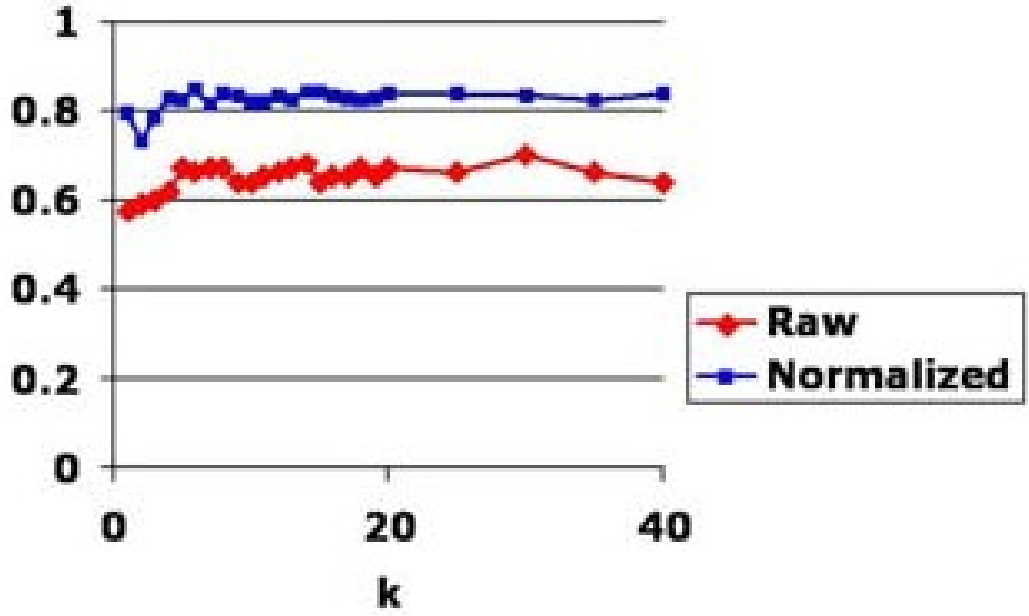
Heart Disease

- Relatively insensitive to k



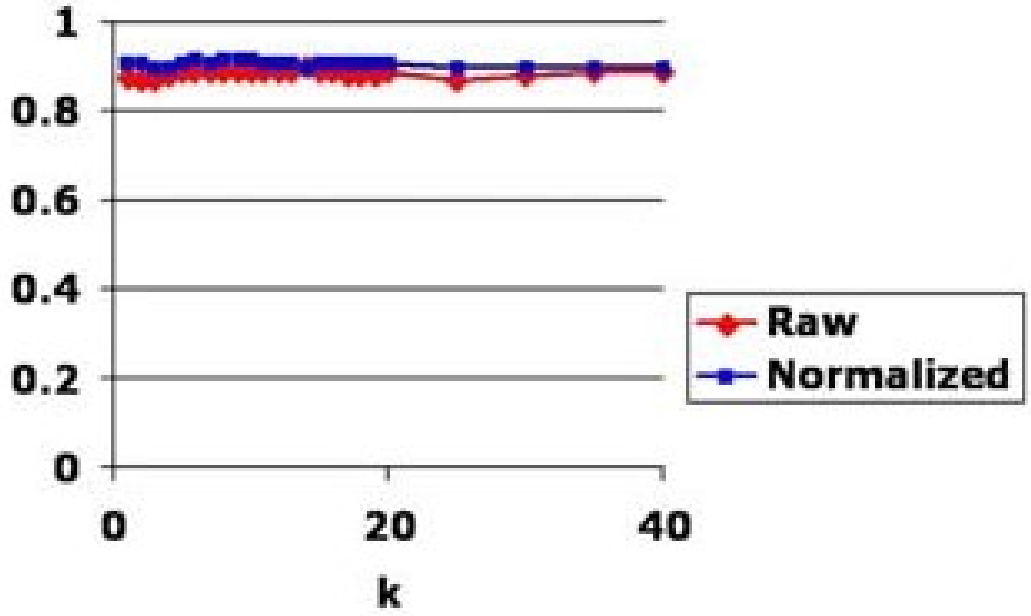
Heart Disease

- Relatively insensitive to k
- Normalization matters!



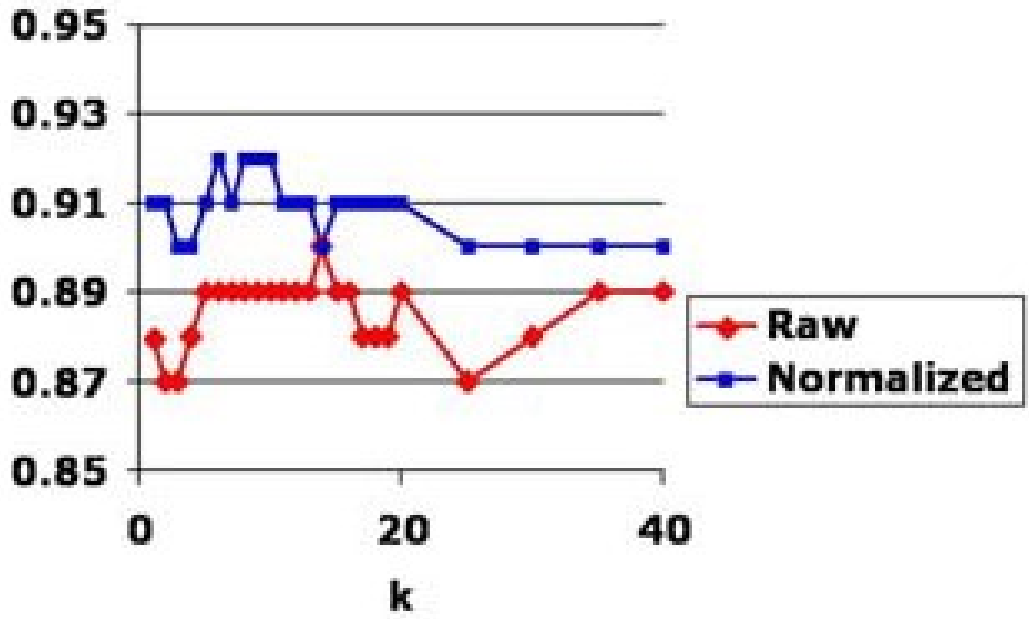
Auto MPG

- Relatively insensitive to k
- Normalization doesn't matter much



Auto MPG

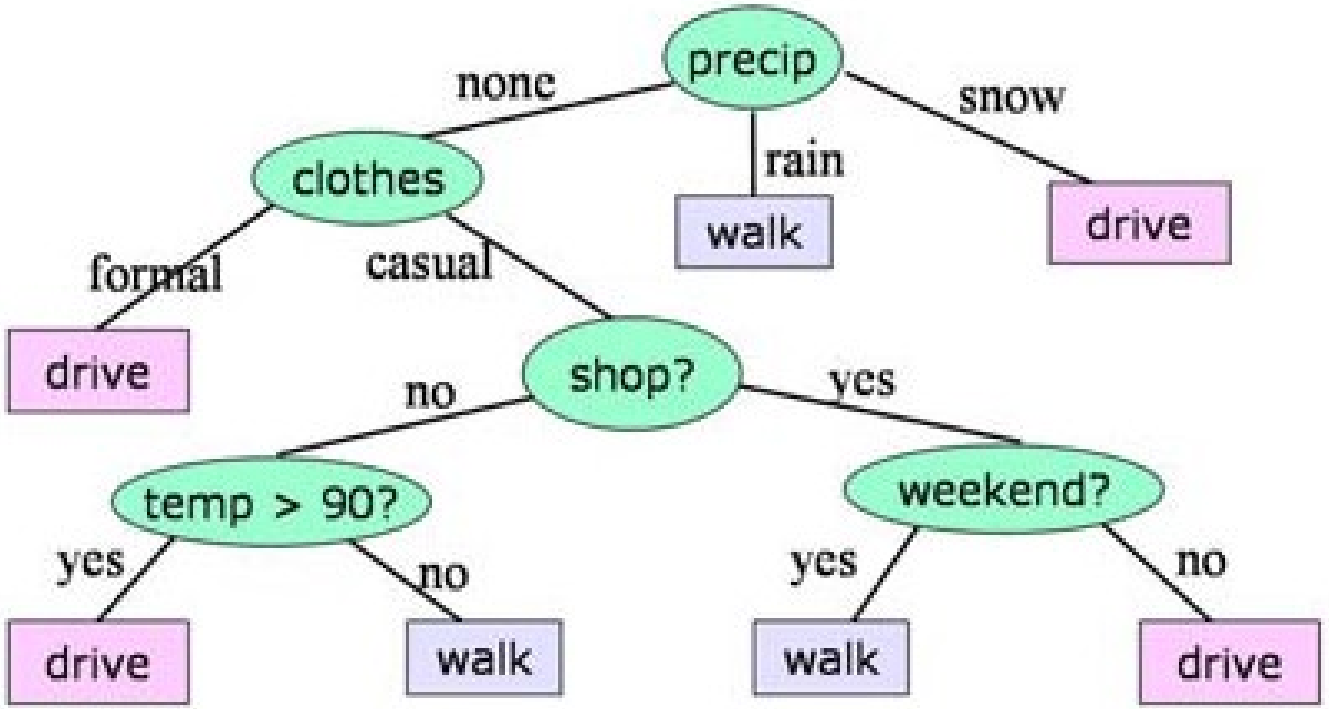
- Now normalization matters a lot!
- Watch the scales on your graphs



Decision Trees with Numeric Inputs

Remember Decision Trees

Use all the data to build a tree of questions with answers at the leaves



Numerical Attributes

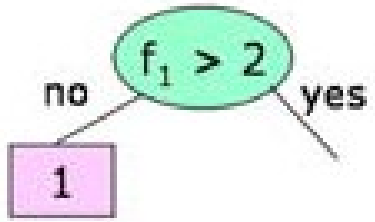
- Tests in nodes can be of the form x_j > constant

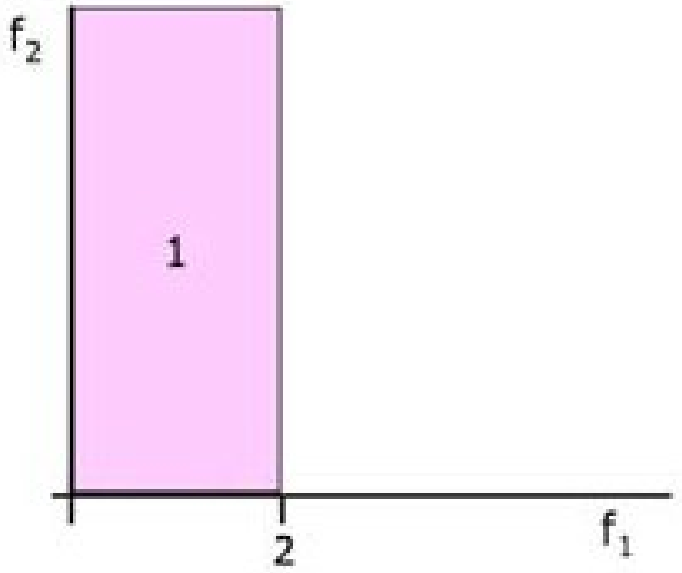
Numerical Attributes

- Tests in nodes can be of the form x_j > constant
- Divides the space into axis-aligned rectangles

Numerical Attributes

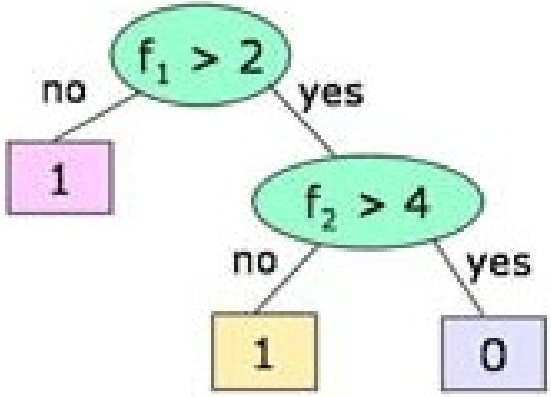
- Tests in nodes can be of the form x_j > constant
- Divides the space into axis-aligned rectangles

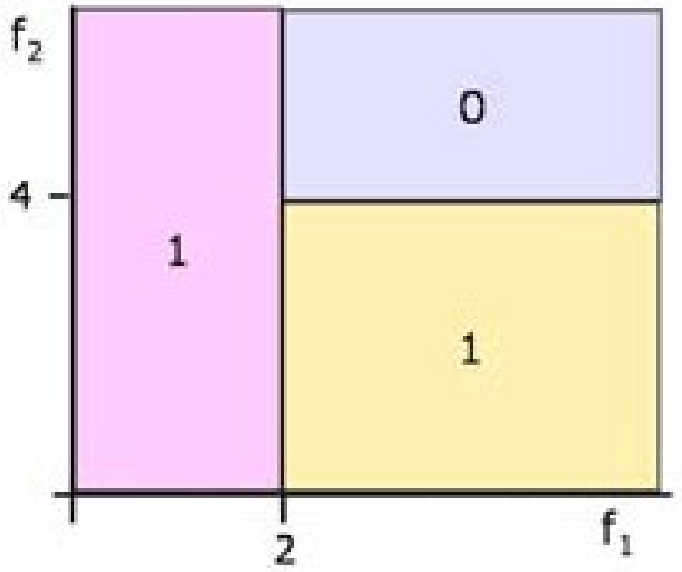




Numerical Attributes

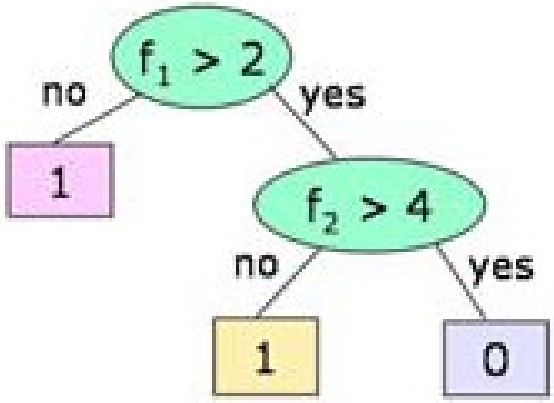
- Tests in nodes can be of the form x_j > constant
- Divides the space into axis-aligned rectangles

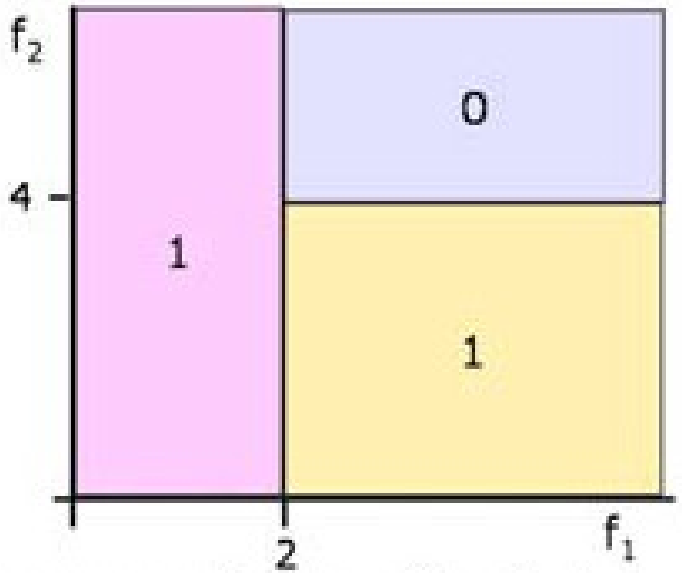




Numerical Attributes

- Tests in nodes can be of the form x_j > constant
- Divides the space into axis-aligned rectangles

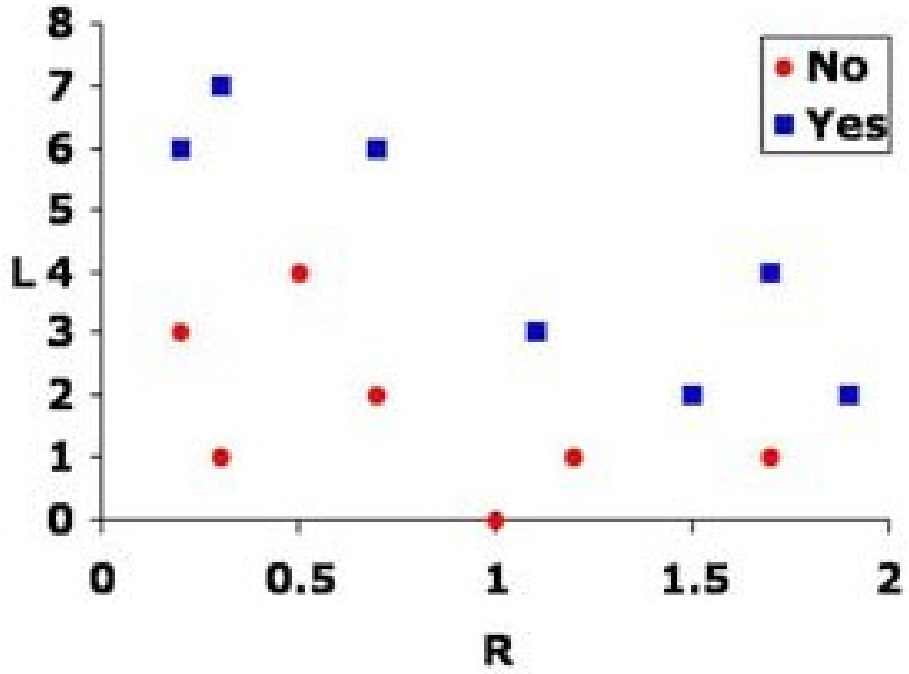




- Non-axis aligned hypotheses can be smaller but hard to find

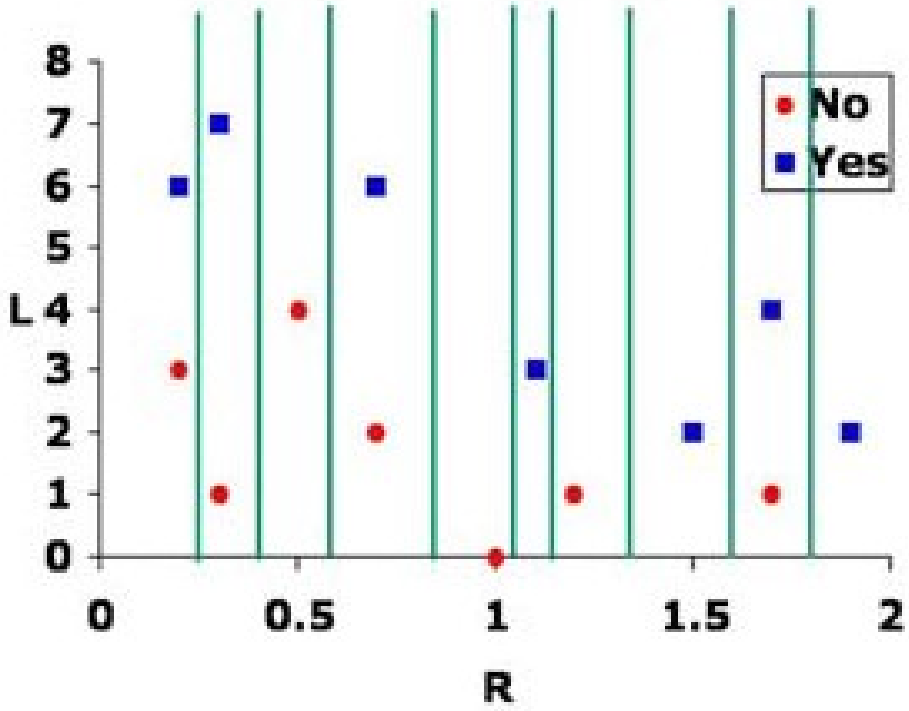
Considering Splits

- Consider a split between each point in each dimension



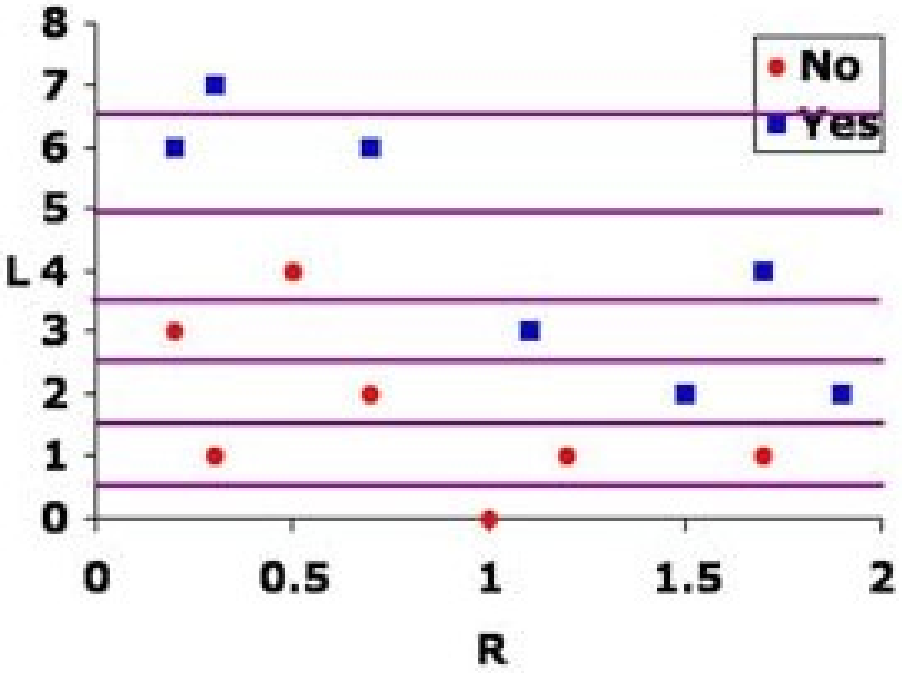
Considering Splits

- Consider a split between each point in each dimension



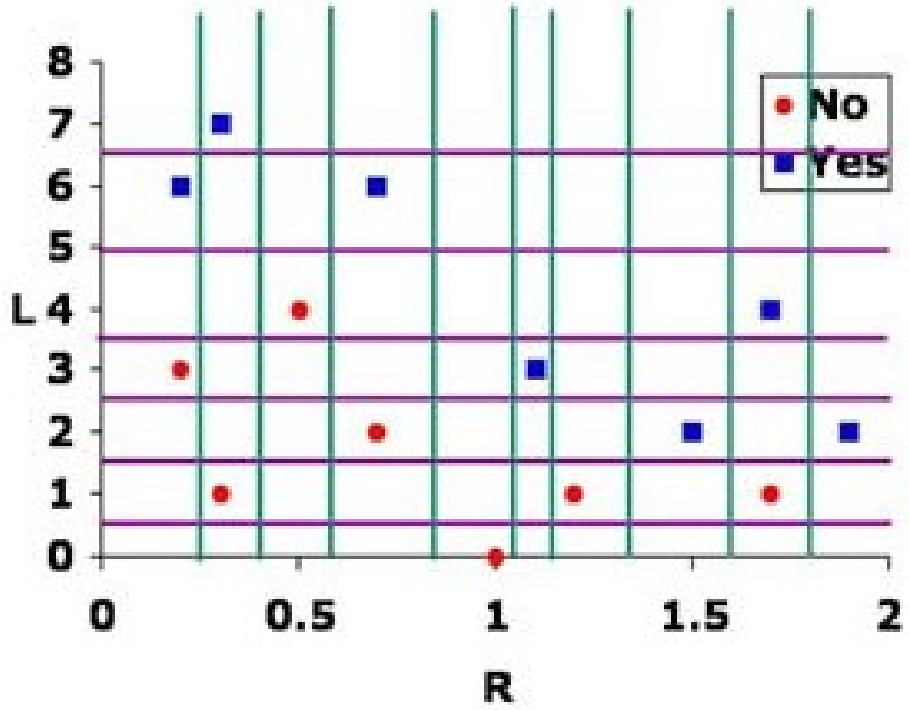
Considering Splits

- Consider a split between each point in each dimension



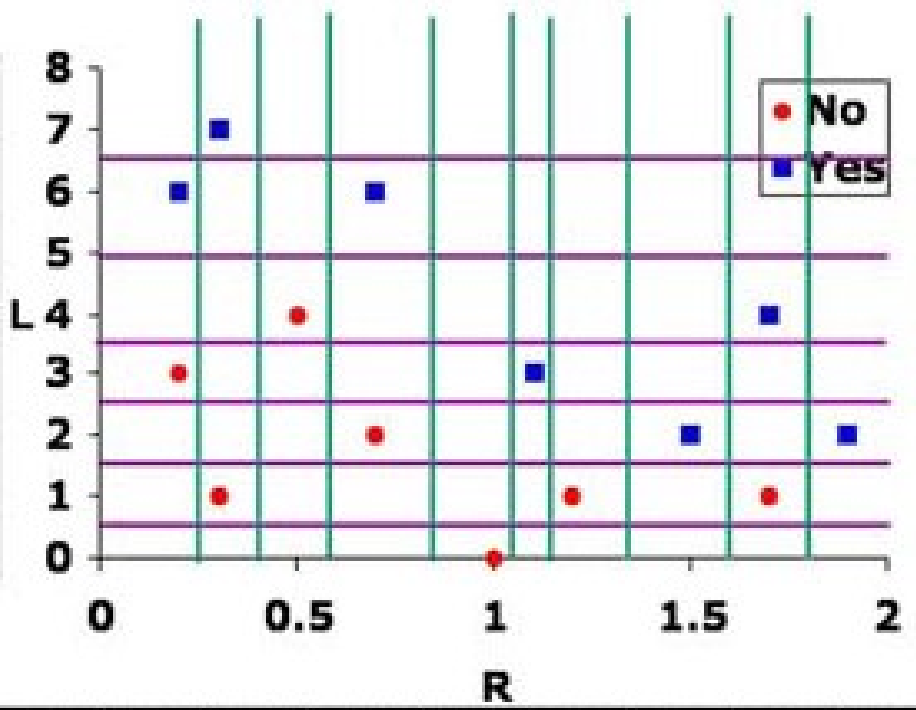
Considering Splits

- Choose split that minimizes average entropy of child nodes



Bankruptcy Example

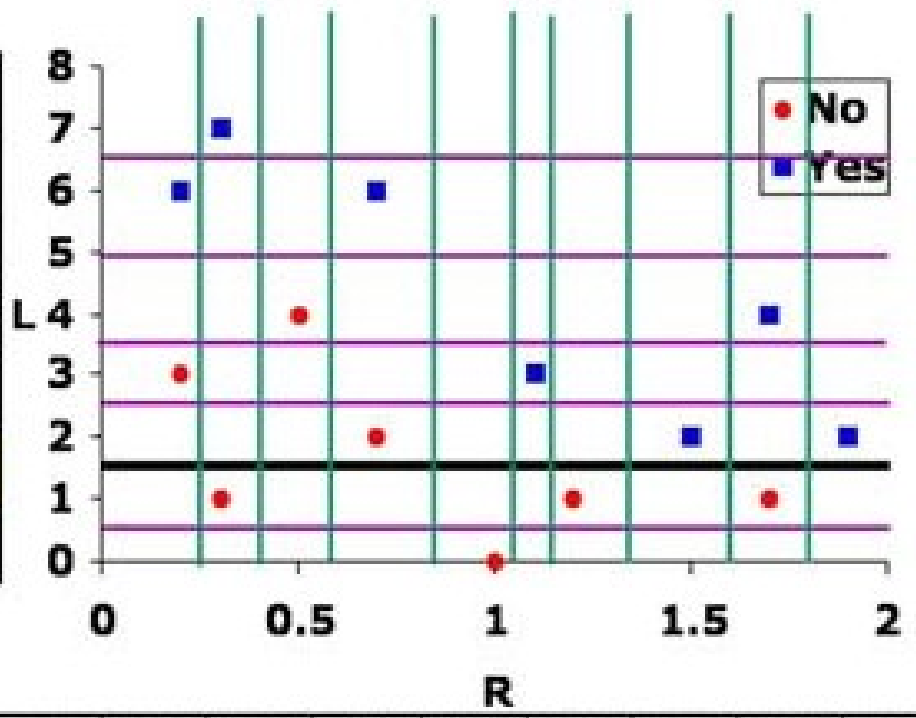
L<y	NL	PL	NR	PR	AE
6.5	7	6	0	1	0.93
5.0	7	4	0	3	0.74
3.5	6	3	1	4	0.85
2.5	5	2	2	5	0.86
1.5	4	0	3	7	0.63
0.5	1	0	6	7	0.93
	# neg to left	# pos to left	# neg to right	# pos to right	



AE	1.00	1.00	0.98	0.98	0.94	0.98	0.92	0.98	0.92
R<x	0.25	0.40	0.60	0.85	1.05	1.15	1.35	1.60	1.80

Bankruptcy Example

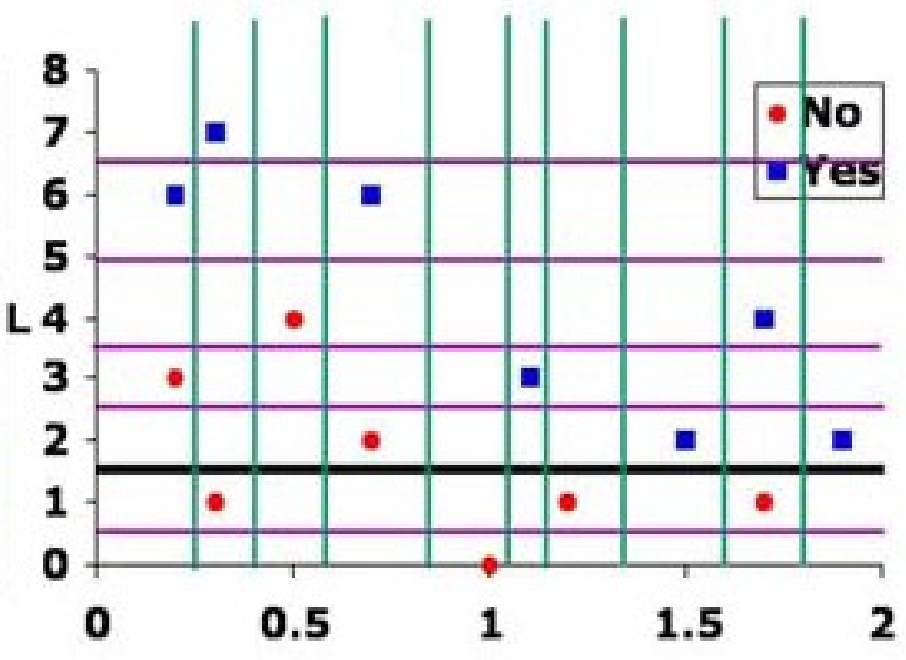
L<y	NL	PL	NR	PR	AE
6.5	7	6	0	1	0.93
5.0	7	4	0	3	0.74
3.5	6	3	1	4	0.85
2.5	5	2	2	5	0.86
1.5	4	0	3	7	0.63
0.5	1	0	6	7	0.93

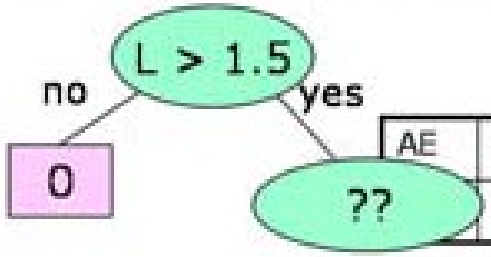


AE	1.00	1.00	0.98	0.98	0.94	0.98	0.92	0.98	0.92
R<x	0.25	0.40	0.60	0.85	1.05	1.15	1.35	1.60	1.80

Bankruptcy Example

L<y	NL	PL	NR	PR	AE
6.5	7	6	0	1	0.93
5.0	7	4	0	3	0.74
3.5	6	3	1	4	0.85
2.5	5	2	2	5	0.86
1.5	4	0	3	7	0.63
0.5	1	0	6	7	0.93

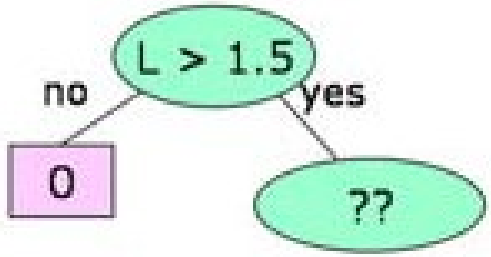


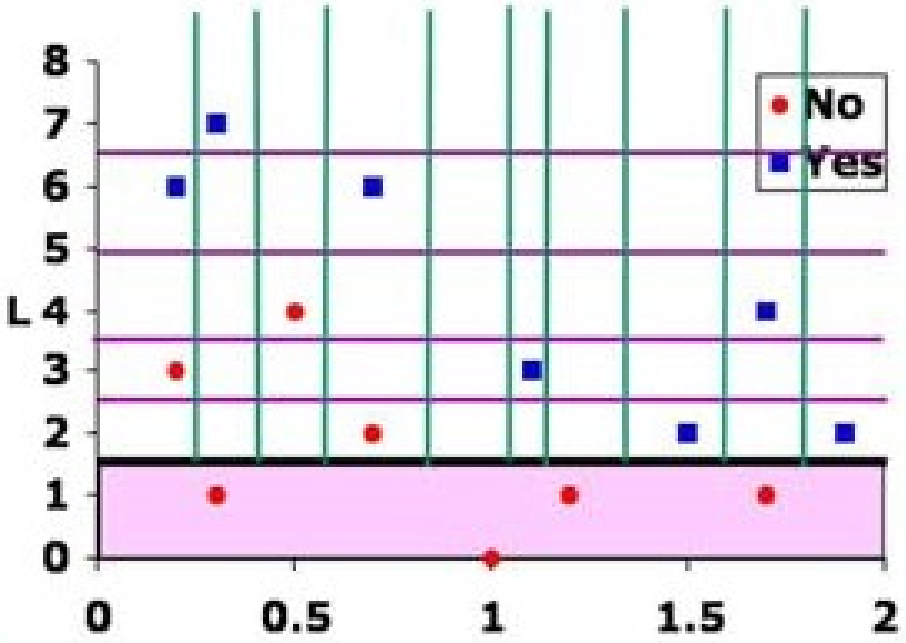


AE	1.00	1.00	0.98	0.98	0.94	0.98	0.92	0.98	0.92
	0.25	0.40	0.60	0.85	1.05	1.15	1.35	1.60	1.80

Bankruptcy Example

L<y	NL	PL	NR	PR	AE
6.5	6	3	0	1	0.83
5.0	4	3	0	3	0.69
3.5	3	2	4	1	0.85
2.5	2	1	5	2	0.88

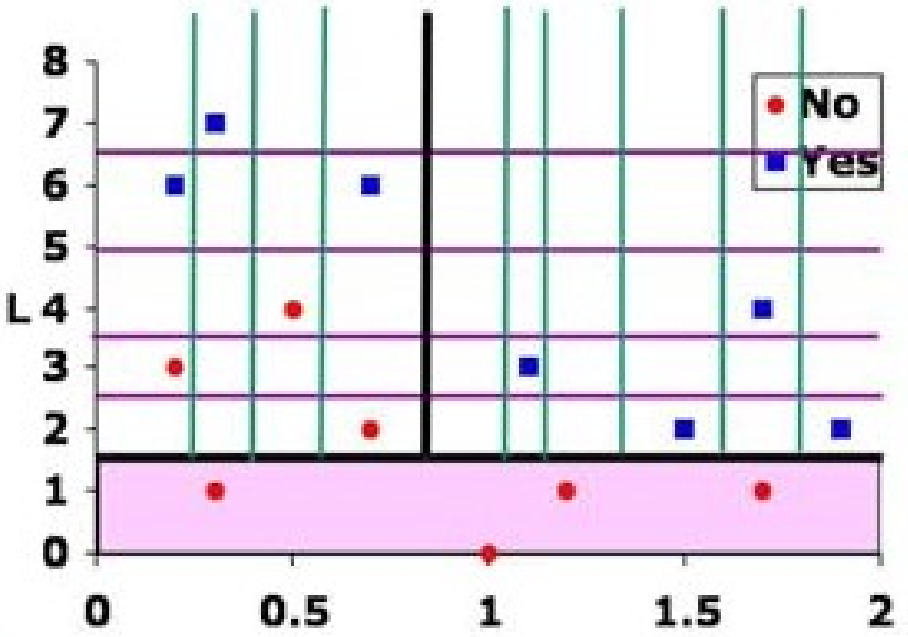


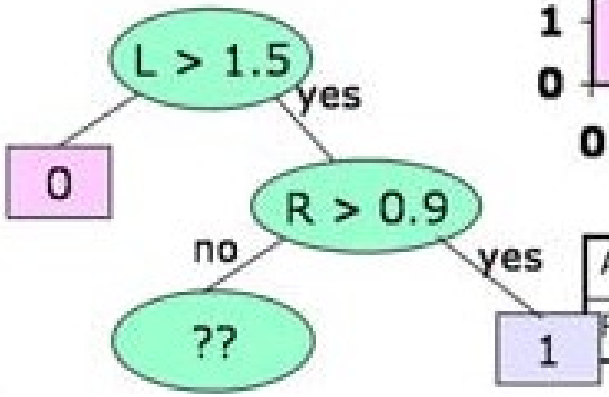


AE	0.85	0.88	0.79	0.60	0.69	0.76	0.83
R<x	0.25	0.40	0.60	0.90	1.30	1.60	1.80

Bankruptcy Example

L<y	NL	PL	NR	PR	AE
6.5	6	3	0	1	0.83
5.0	4	3	0	3	0.69
3.5	3	2	4	1	0.85
2.5	2	1	5	2	0.88

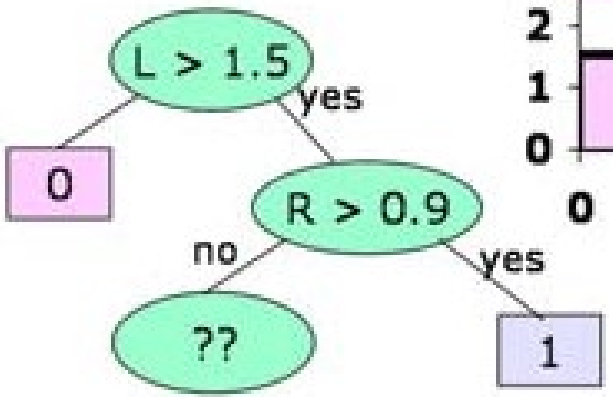


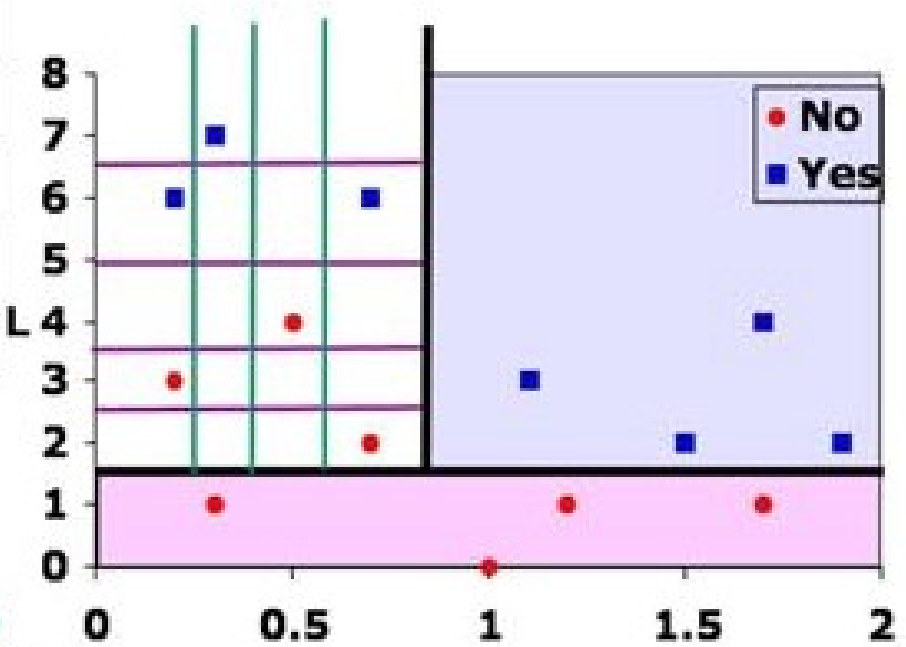


AD	0.85	0.88	0.79	0.60	0.69	0.76	0.83
R<x	0.25	0.40	0.60	0.90	1.30	1.60	1.80

Bankruptcy Example

L<y	NL	PL	NR	PR	AE
6.5	3	2	0	1	0.81
5.0	3	0	0	3	0.00
3.5	2	0	1	3	0.54
2.5	1	0	2	3	0.81

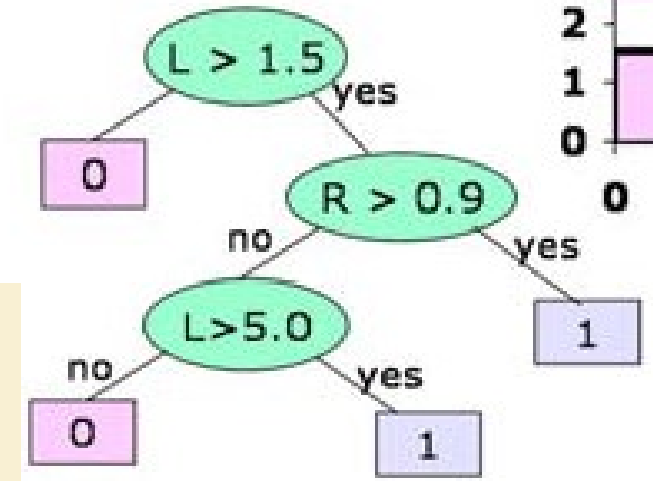


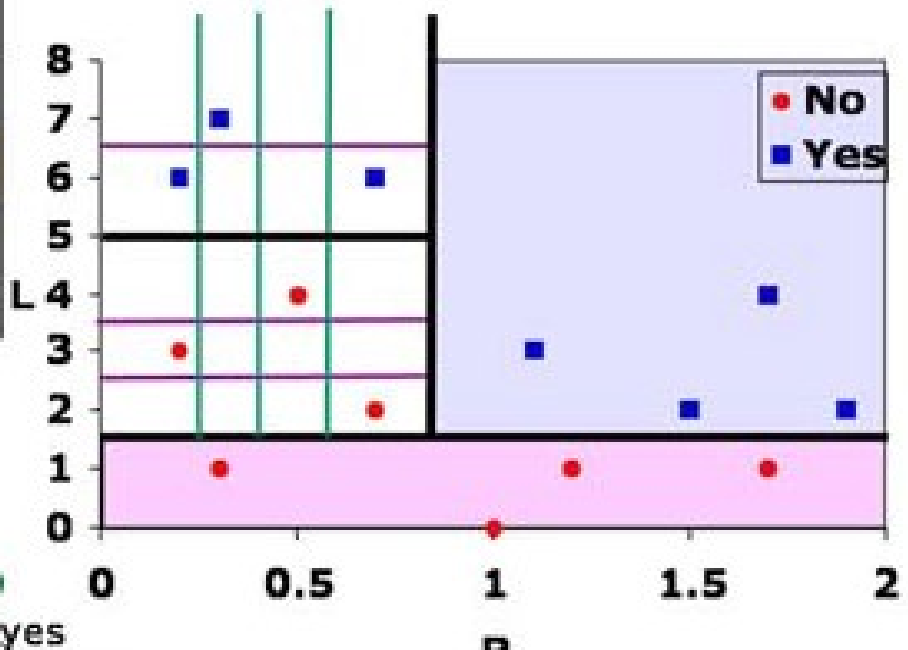


AE	1.00	0.92	1.00
R<x	0.25	0.40	0.60

Bankruptcy Example

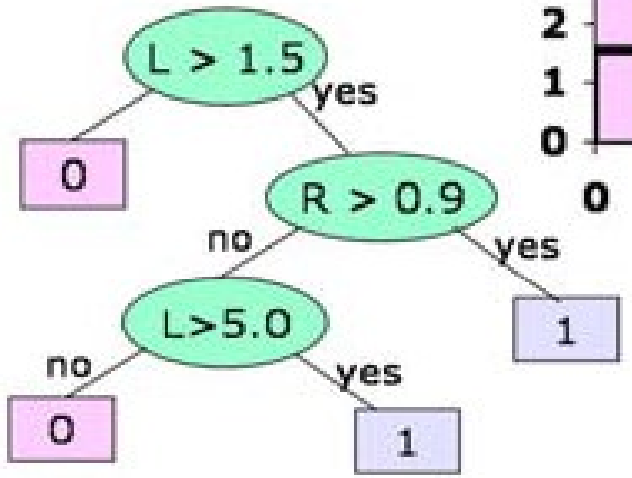
L<y	NL	PL	NR	PR	AE
6.5	3	2	0	1	0.81
5.0	3	0	0	3	0.00
3.5	2	0	1	3	0.54
2.5	1	0	2	3	0.81

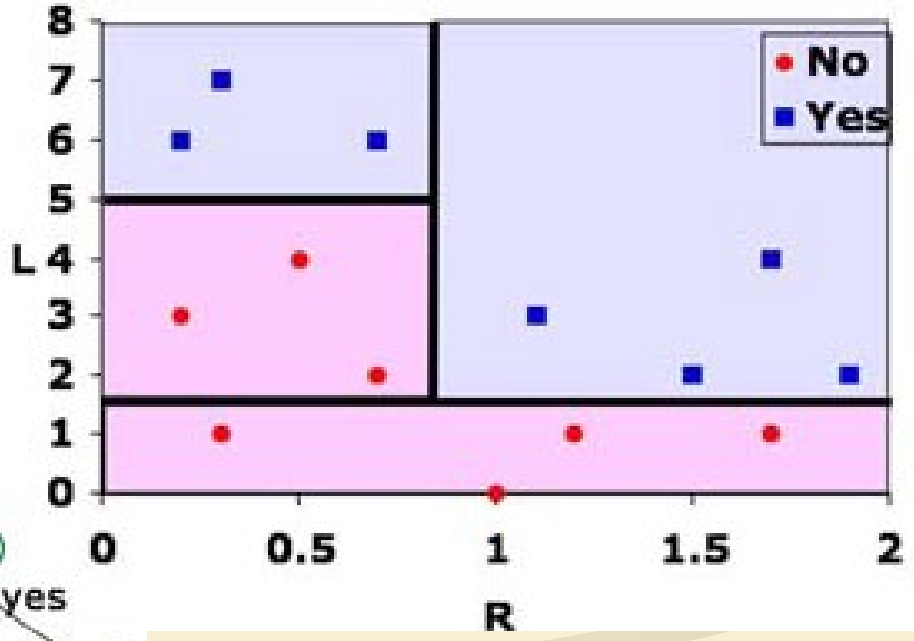




AE	1.00	0.92	1.00
R<x	0.25	0.40	0.60

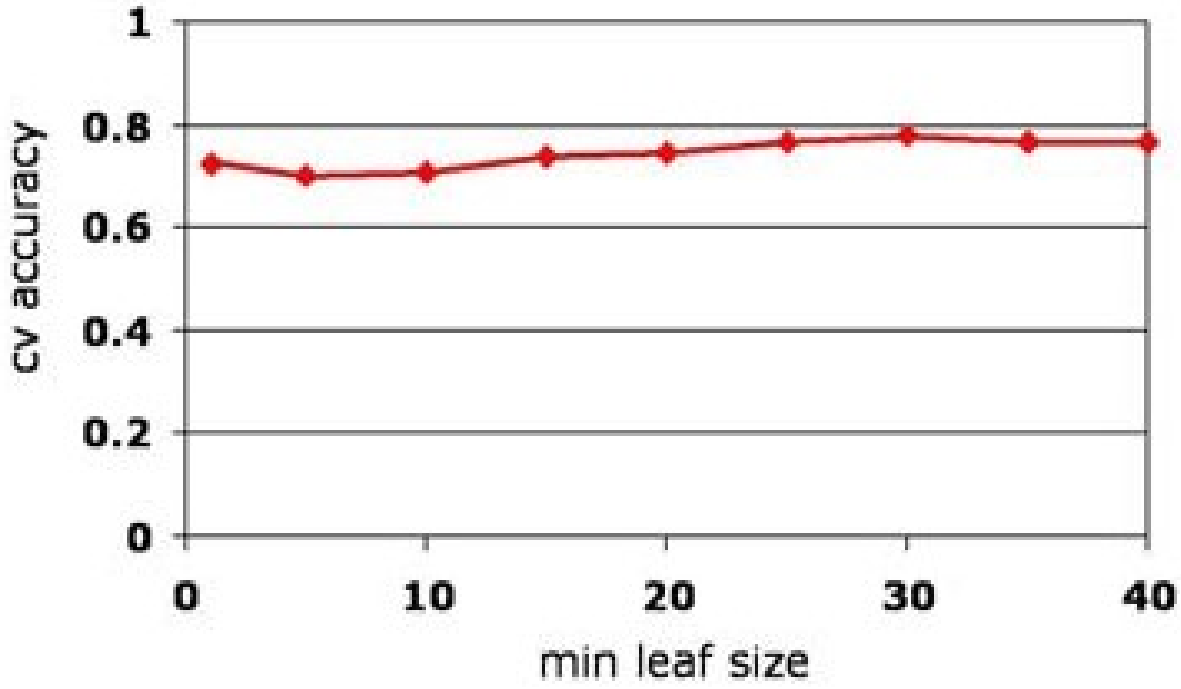
Bankruptcy Example





Heart Disease

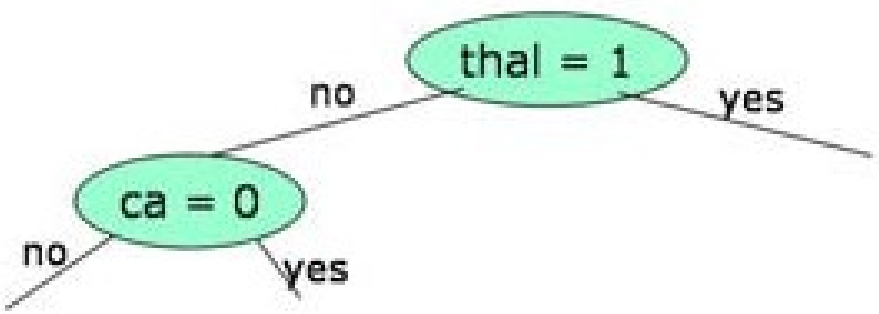
- Best performance (.77) slightly worse than nearest neighbor (.81)



Heart Disease

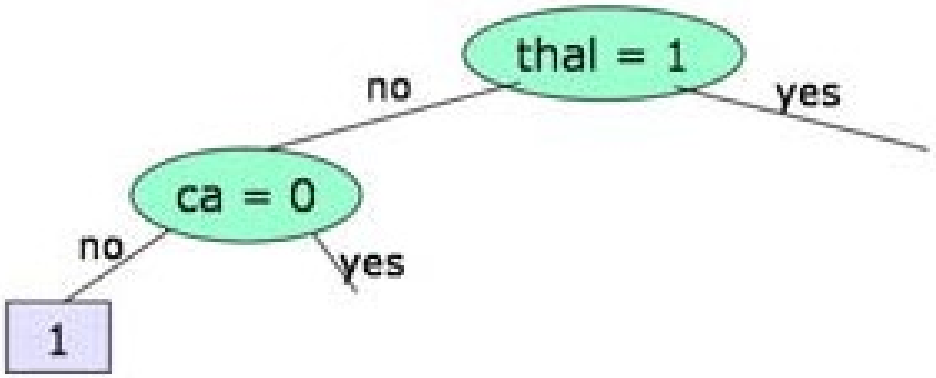
thal = 1: normal exercise thallium scintigraphy test

Heart Disease



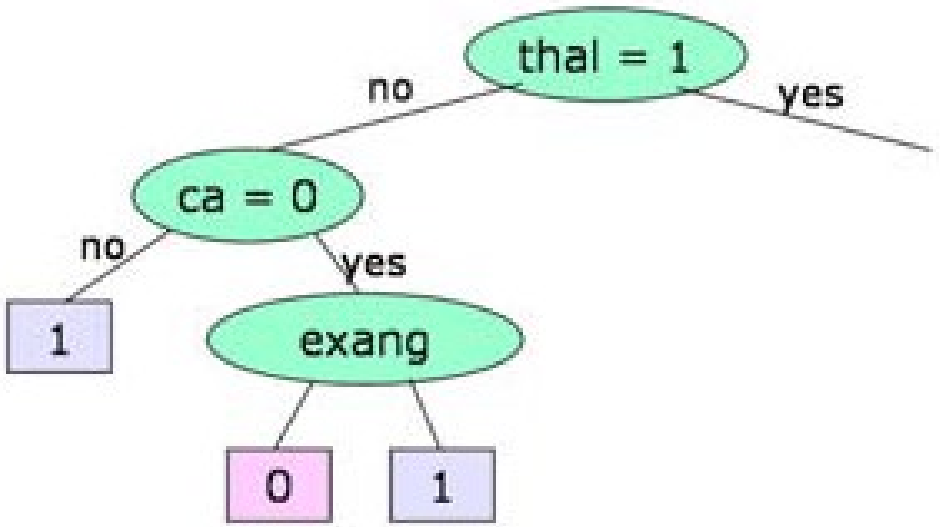
thal = 1: normal exercise thallium scintigraphy test
ca = 0: no vessels colored by fluoroscopy

Heart Disease



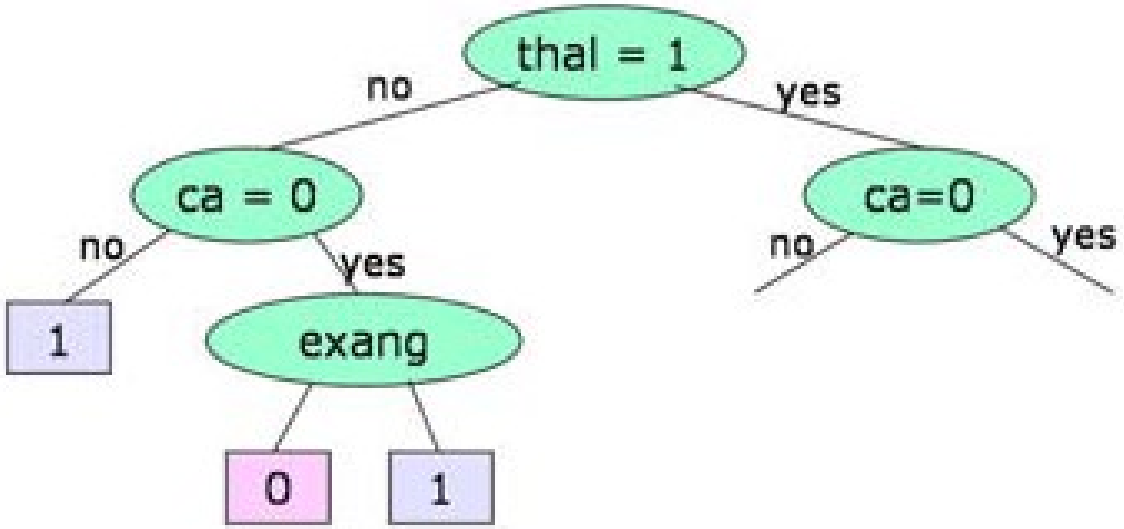
thal = 1: normal exercise thallium scintigraphy test
ca = 0: no vessels colored by fluoroscopy

Heart Disease



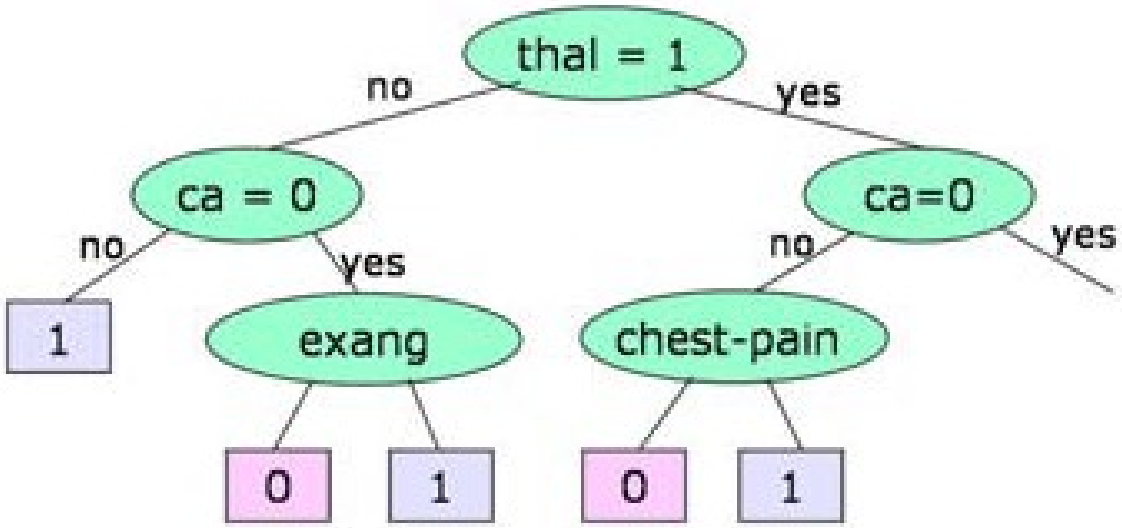
thal = 1: normal exercise thallium scintigraphy test
ca = 0: no vessels colored by fluoroscopy
exang: exercise induced angina

Heart Disease



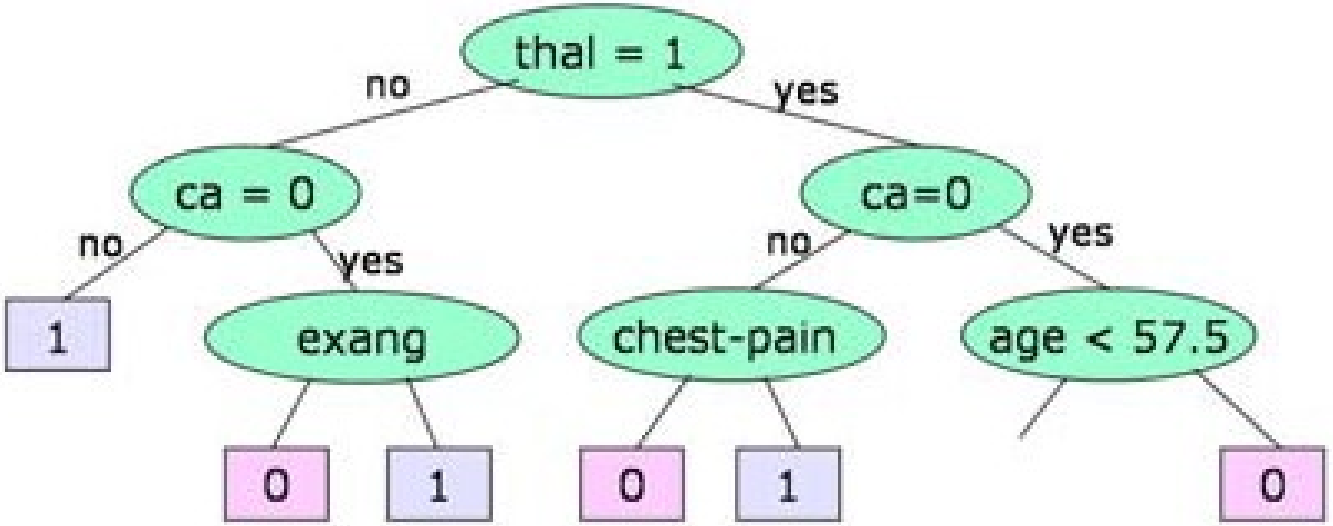
thal = 1: normal exercise thallium scintigraphy test
ca = 0: no vessels colored by fluoroscopy
exang: exercise induced angina

Heart Disease



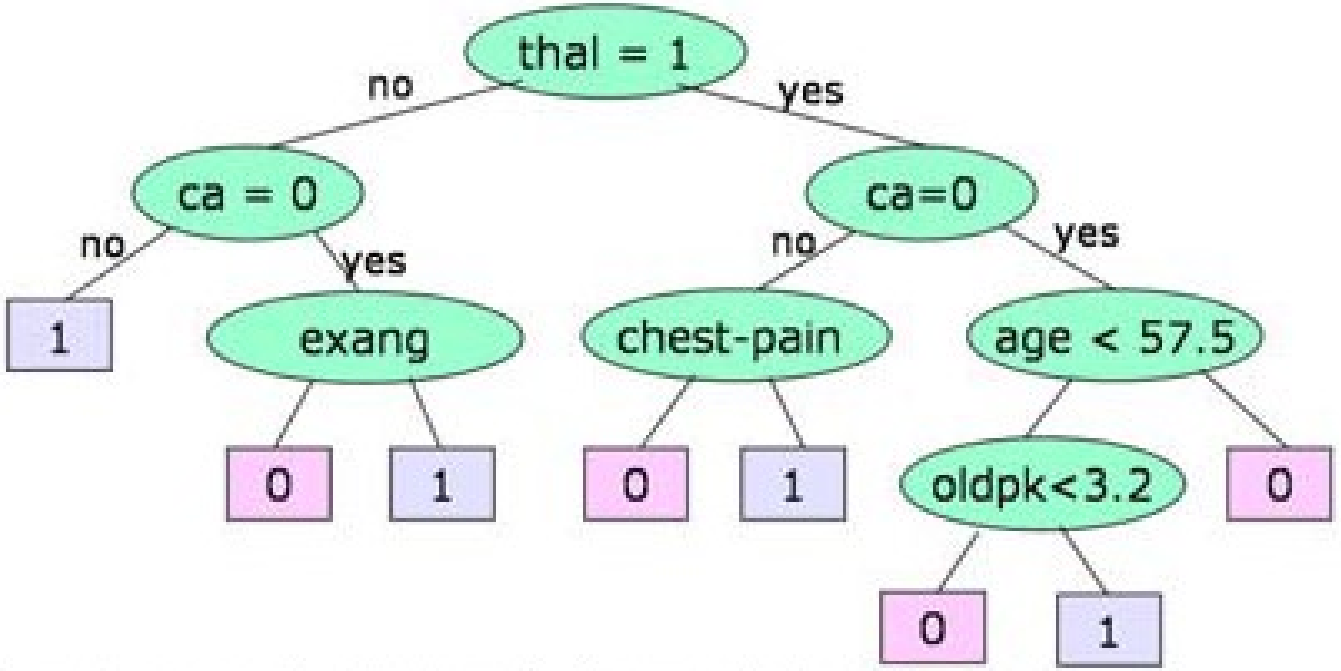
thal = 1: normal exercise thallium scintigraphy test
ca = 0: no vessels colored by fluoroscopy
exang: exercise induced angina

Heart Disease



thal = 1: normal exercise thallium scintigraphy test
ca = 0: no vessels colored by fluoroscopy
exang: exercise induced angina

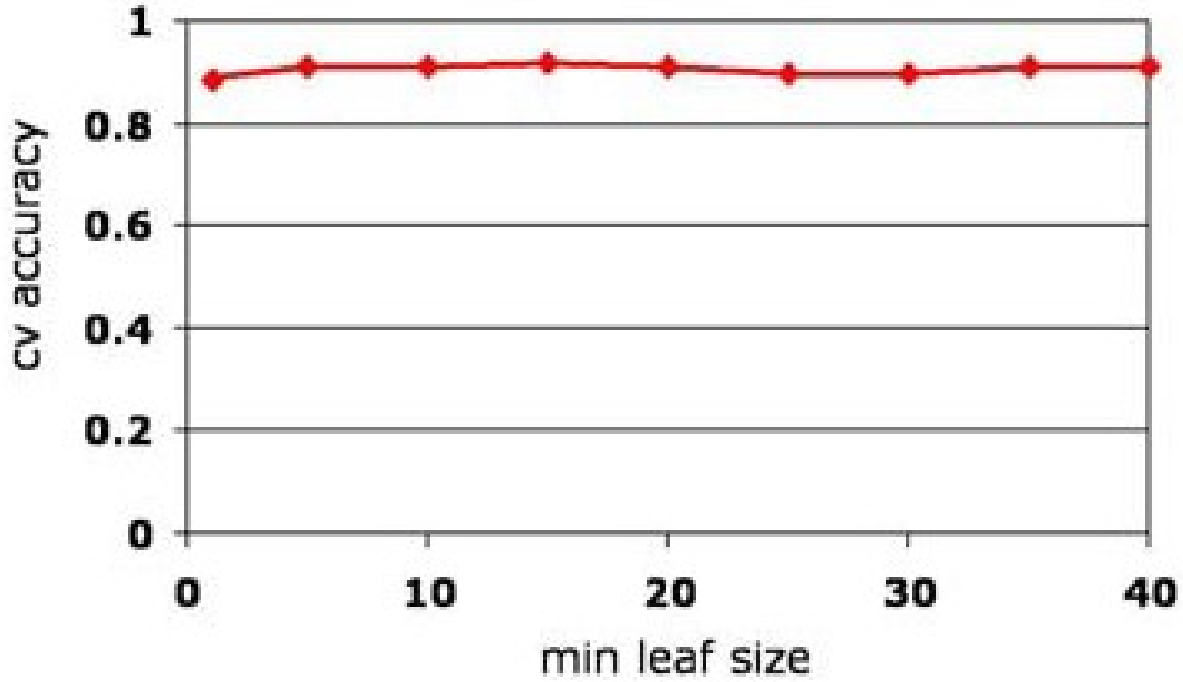
Heart Disease



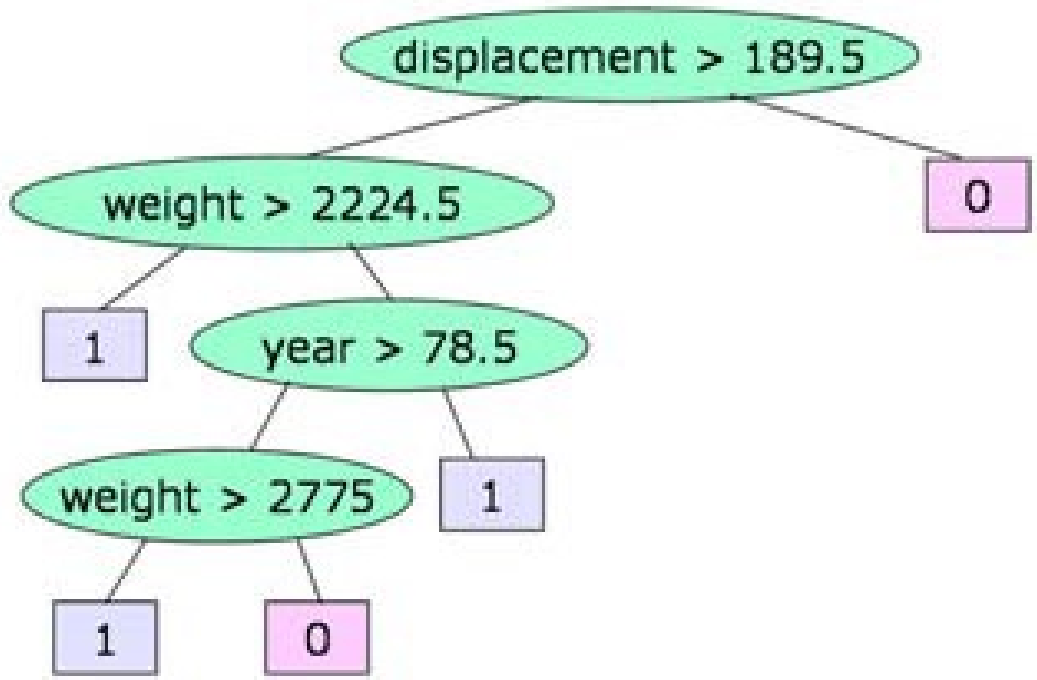
thal = 1: normal exercise thallium scintigraphy test
ca = 0: no vessels colored by fluoroscopy
exang: exercise induced angina
oldpk: feature of cardiogram

Auto MPG

- Performance (.91) essentially the same as nearest neighbor



More than 22 MPG?



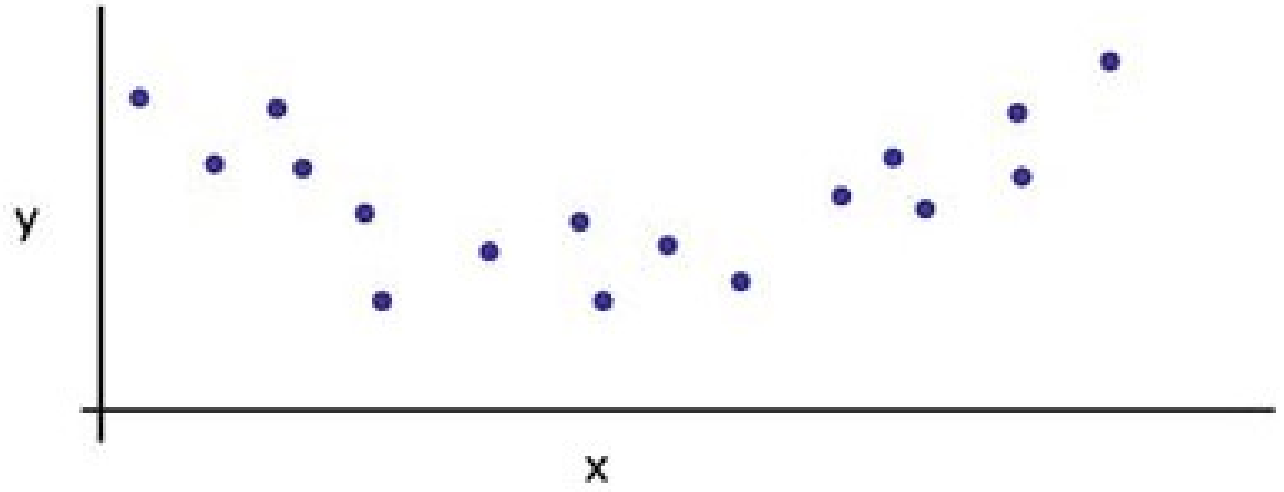
Regression

Regression

- Output is a continuous numeric value
  - Locally-weighted averaging
  - Regression trees

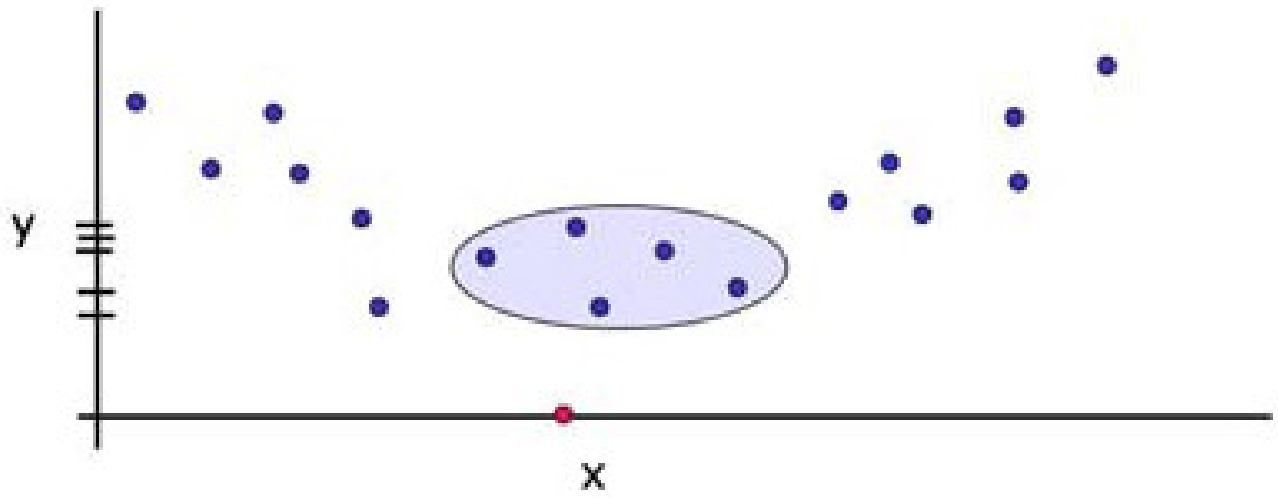
Local Averaging

- Remember all your data



Local Averaging

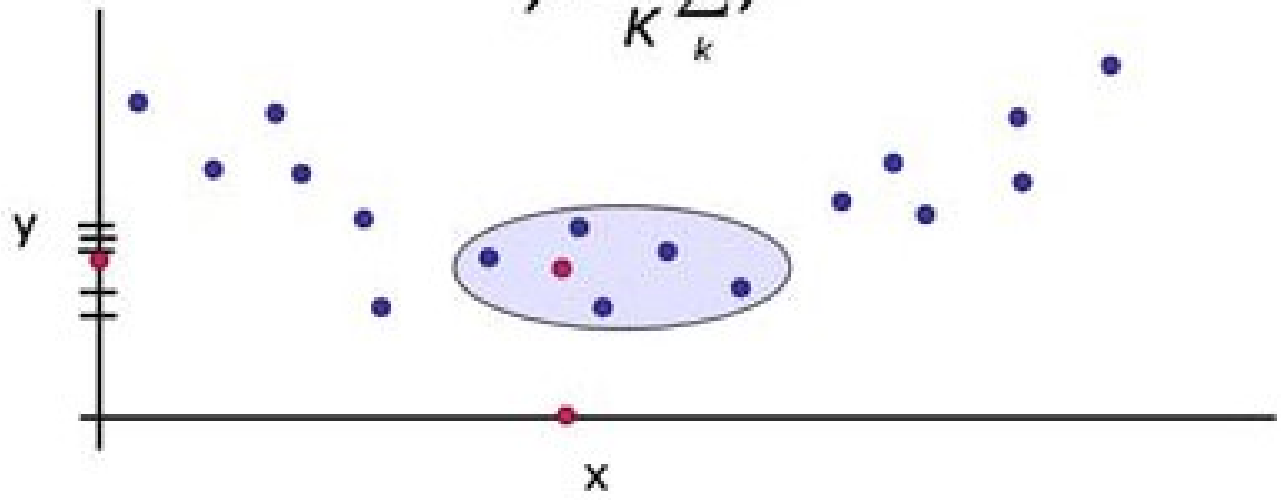
- Remember all your data
- When someone asks a question,
  - find the K nearest old data points



Local Averaging

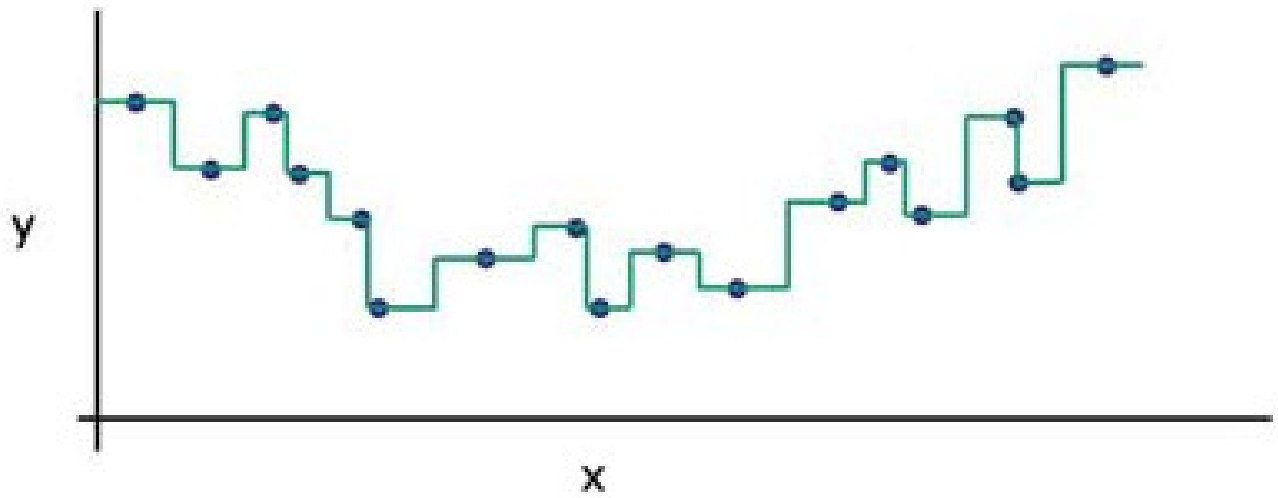
- Remember all your data
- When someone asks a question,
  - find the K nearest old data points
  - return the average of the answers associated with them

$$y = \frac{1}{K}\sum_{k} y^k$$



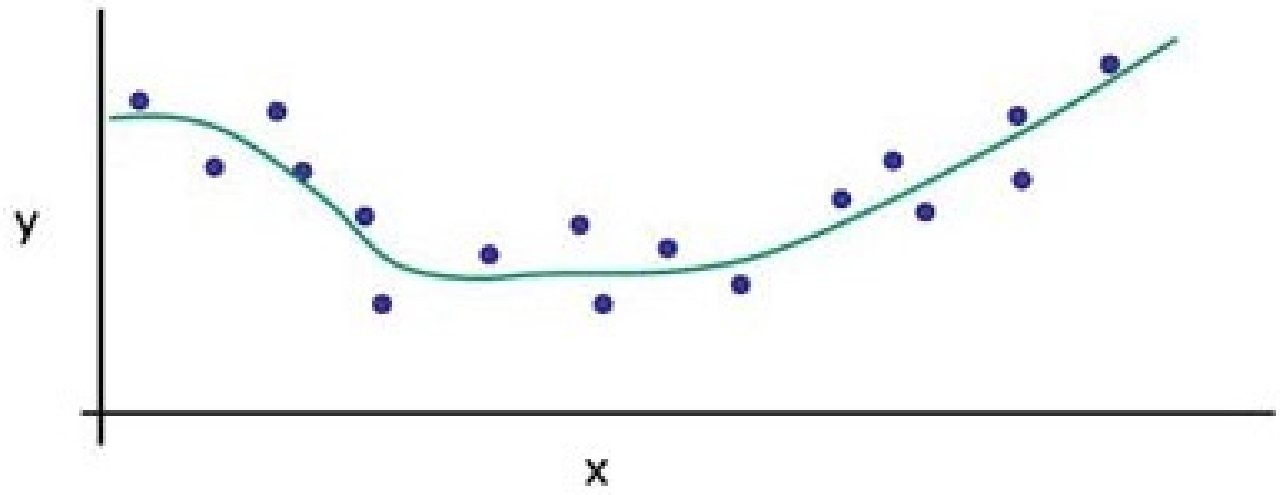
K = 1

- Tracks data very closely
- Prone to overfitting



Bigger K

- Smoothes out variations in data
- May introduce too much bias



Locally Weighted Averaging

Locally Weighted Averaging

- Find all points within distance λ from target point
- Average the outputs, weighted according to how far away they are from the target point

Locally Weighted Averaging

- Find all points within distance λ from target point
- Average the outputs, weighted according to how far away they are from the target point
- Given a target x, with k ranging over neighbors,

$$y = \frac{\sum_k K(x, x^k) y^k}{\sum_k K(x, x^k)}$$

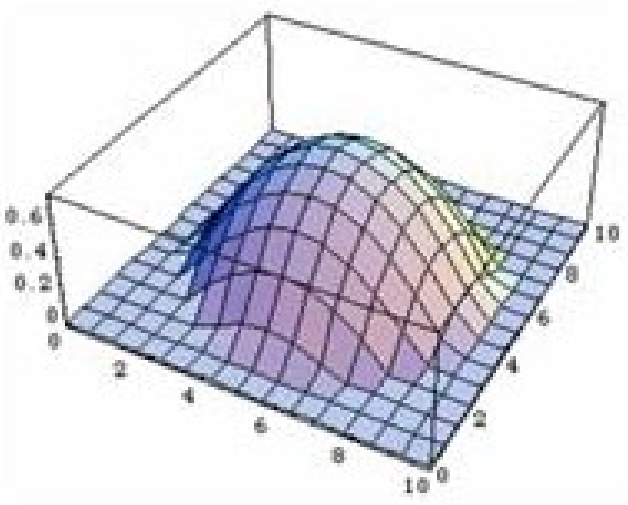
weighting "kernel"

Epanechnikov Kernel

- D is Euclidean distance

$$K(x, x^k) = \max\left(\frac{3}{4}\left(1 - \frac{D(x, x^k)^2}{\lambda^2}\right), 0\right)$$

- x = <5,5>
- $\lambda = 4$



- Many other possible choices of kernel K

Smoooth

- How should we choose λ?
  - If small, then we aren't averaging many points
    - Worse at averaging out noise
    - Better at modeling discontinuities
  - If big, we are averaging a lot of points
    - Good at averaging out noise
    - Smears out discontinuities

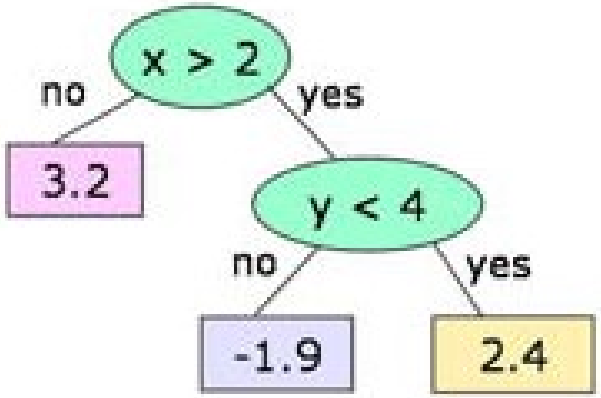
- Can use cross-validation to choose λ
- May be better to let it vary according to local density of points

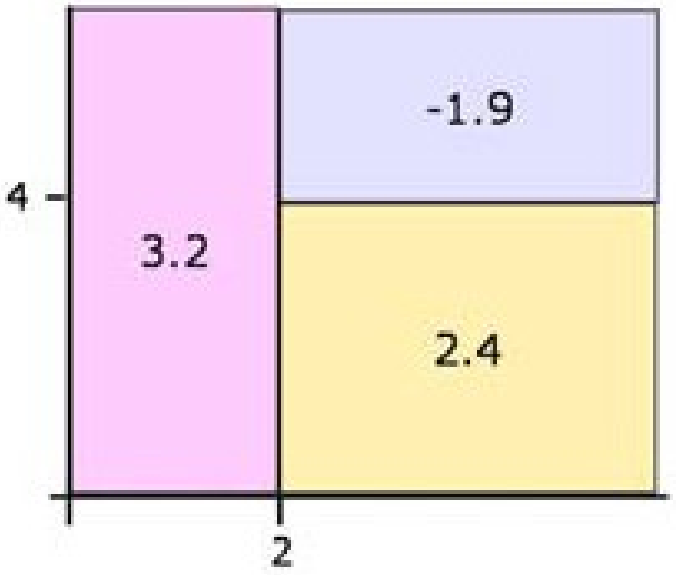
Regression Trees

- Like decision trees, but with real-valued constant outputs at the leaves

Regression Trees

- Like decision trees, but with real-valued constant outputs at the leaves





Leaf Values

- Assign a leaf node the average of the y values of the data points that fall there.

Leaf Values

- Assign a leaf node the average of the y values of the data points that fall there.
- We'd like to have groups of points in a leaf that have similar y values (because then the average is a good representative)

Variance

- Measure of how much a set of numbers is spread out

Variance

- Measure of how much a set of numbers is spread out
- Mean of m values, z_1 through z_m :

$$\mu = \frac{1}{m}\sum_{k=1}^{m} z_k$$

Variance

- Measure of how much a set of numbers is spread out
- Mean of m values, z_1 through z_m :

$$\mu = \frac{1}{m}\sum_{k=1}^{m} z_k$$

- Variance: average squared difference between z's and the mean:

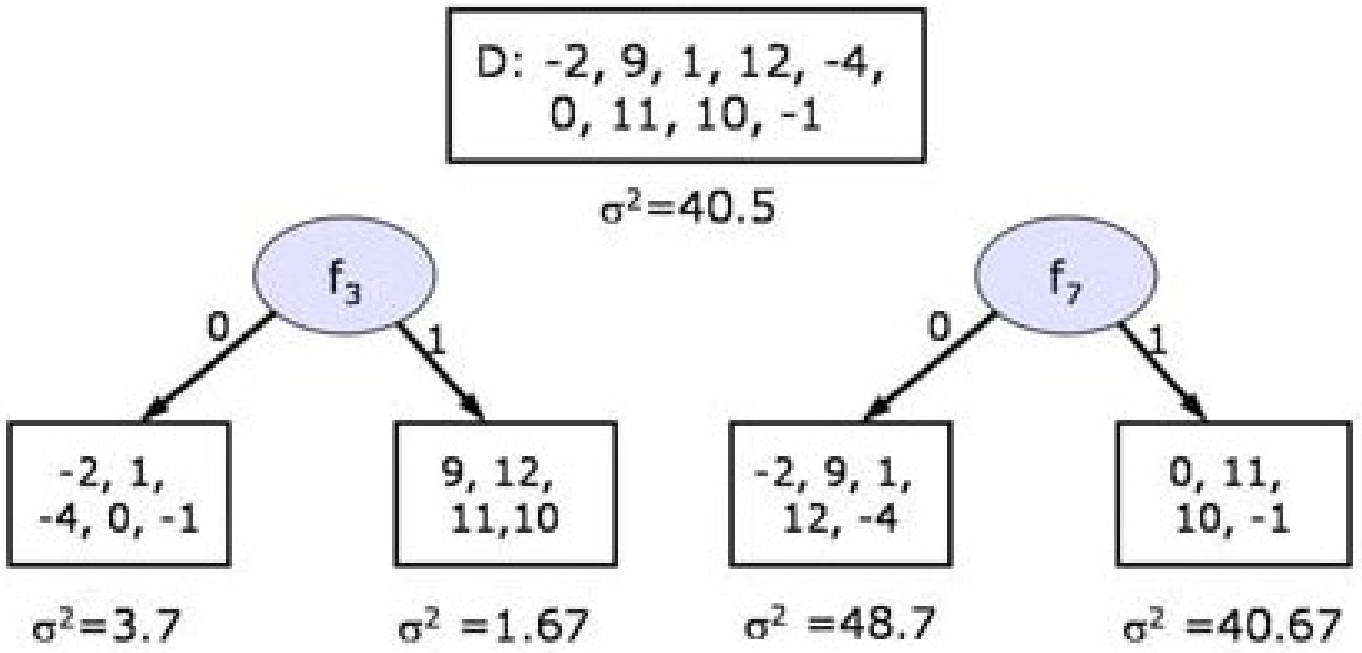
$$\sigma^2 = \frac{1}{m-1}\sum_{k=1}^{m} (z_k - \mu)^2$$

Let's Split

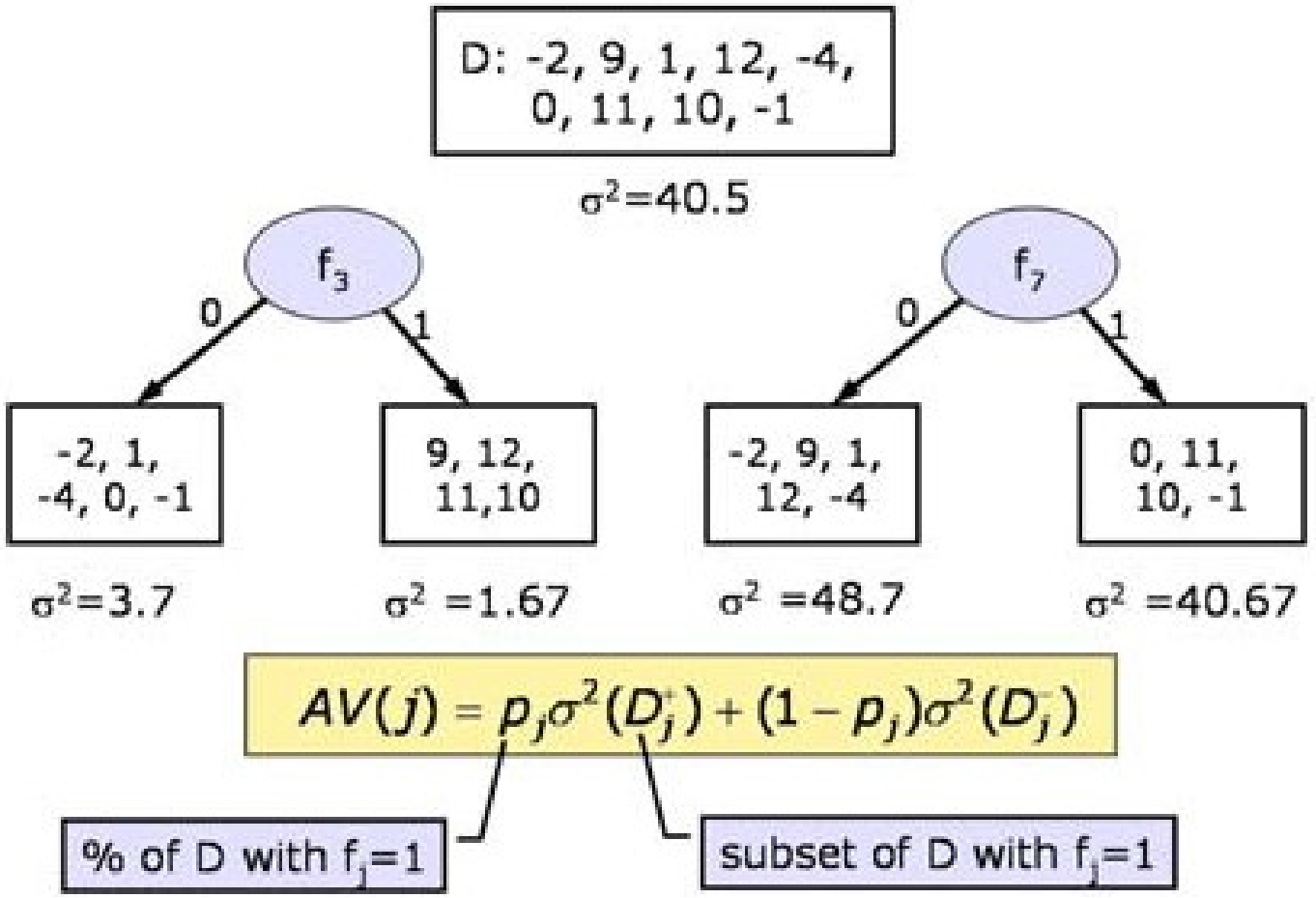
D: -2, 9, 1, 12, -4, 0, 11, 10, -1

$\sigma^2=40.5$

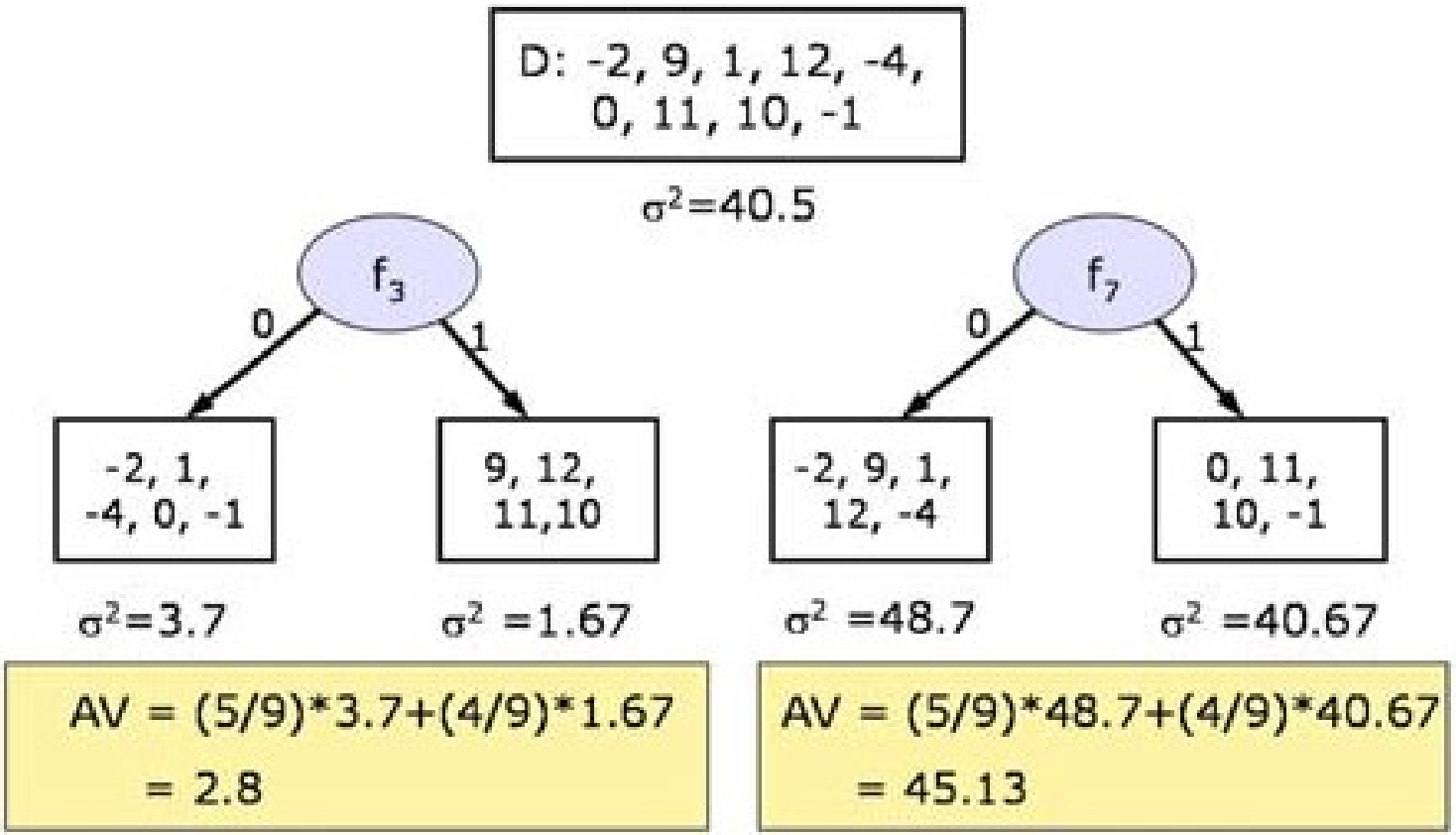
Let's Split


D: -2, 9, 1, 12, -4, 0, 11, 10, -1
σ²=40.5
f3
0
1
f7
0
1
-2, 1, -4, 0, -1
9, 12, 11,10
-2, 9, 1, 12, -4
0, 11, 10, -1
σ²=3.7
σ² =1.67
σ² =48.7
σ² =40.67

Let's Split


D: -2, 9, 1, 12, -4, 0, 11, 10, -1
σ²=40.5
f3
0
1
f7
0
1
-2, 1, -4, 0, -1
9, 12, 11,10
-2, 9, 1, 12, -4
0, 11, 10, -1
σ²=3.7
σ² =1.67
σ² =48.7
σ² =40.67
AV(j) = p_j σ²(D_j^+) + (1 − p_j) σ²(D_j^−)
% of D with f_j=1
subset of D with f_j=1

Let's Split


D: -2, 9, 1, 12, -4, 0, 11, 10, -1
σ²=40.5
f3
0
1
f7
0
1
-2, 1, -4, 0, -1
9, 12, 11,10
-2, 9, 1, 12, -4
0, 11, 10, -1
σ²=3.7
σ² =1.67
σ² =48.7
σ² =40.67
AV = (5/9)*3.7+(4/9)*1.67 = 2.8
AV = (5/9)*48.7+(4/9)*40.67 = 45.13

Stopping

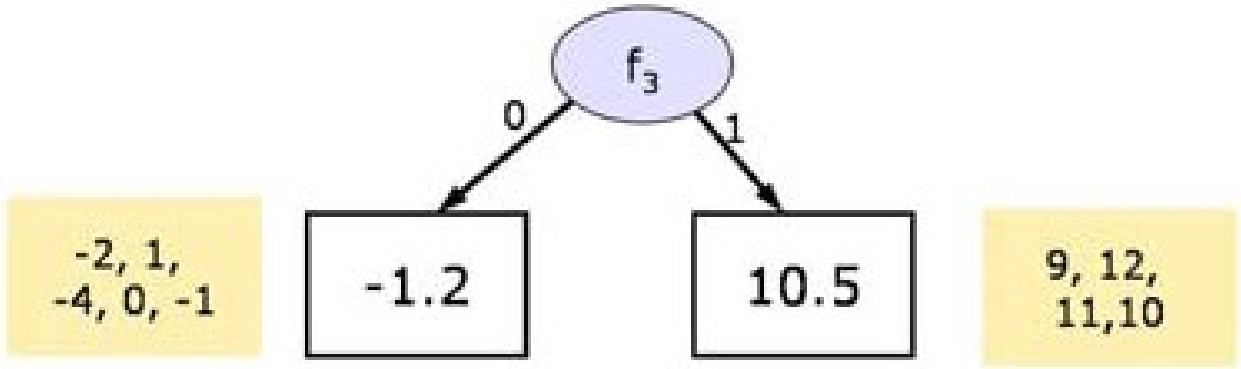
- Stop when variance at a leaf is small enough

Stopping

- Stop when variance at a leaf is small enough
- Or when you have fewer than min-leaf elements at a leaf

Stopping

- Stop when variance at a leaf is small enough
- Or when you have fewer than min-leaf elements at a leaf
- Set y at a leaf to be the mean of the y values of the elements



References

- Flach, P. (2012). Machine Learning: The Art and Science of Algorithms that Make Sense of Data. Cambridge University Press
- Russell S., & Norvig, P. (2021). Artificial Intelligence A Modern Approach. 4th Edition: Pearson Education https://www.academia.edu/45126798/Artificial_Intelligence_A_Modern_Approach_4th_Edition_?auto=download
- Rich E., Knight K. & Nair, S. B. (2011). Artificial Intelligence 3rd Edition: Tata McGraw Hill
- Han, J.., Kamber, M., Pei, J (2012). Data mining: concepts and techniques.Elsevier Inc.
- Nilsson, N.J. (1998). Artificial Intelligence: a new synthesis. Morgan Kaufmann.
- Lucila Ohno-Machado, and Peter Szolovits. HST.947 Medical Artificial Intelligence. Spring 2005. Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu. License: Creative Commons BY-NC-SA.
- https://archive.ics.uci.edu/ml/datasets/heart+Disease
- https://archive.ics.uci.edu/ml/datasets/auto+mpg

Thank You