Memory-based reasoning, nearest neighbors, and collaborative filtering

Lecture 04.01

Classification example: bankruptcy dataset

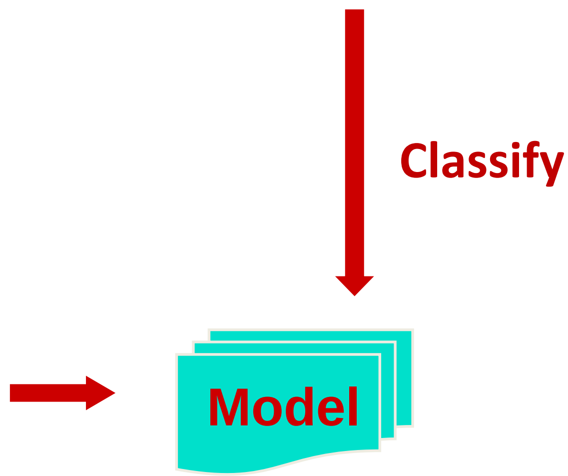
Training set

Late payments, L	Spending ratio, R	Bankruptcy
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

Class labels

New customer

L	R	B
2	0.3	?



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

Memory-based reasoning

Seems poisonous

Amanita muscaria

Classification by similarity

“ If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck. ”

New classifier: nearest neighbor

- Remember an entire labeled training set
- When a new sample comes:
  - Find the most similar sample in the labeled collection (**the nearest neighbor**)
  - Return the class label associated with it

Classification: eager classifier

Training set

Late payments, L	Spending ratio, R	Bankruptcy
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

Class labels

New customer

L	R	B
2	0.3	?

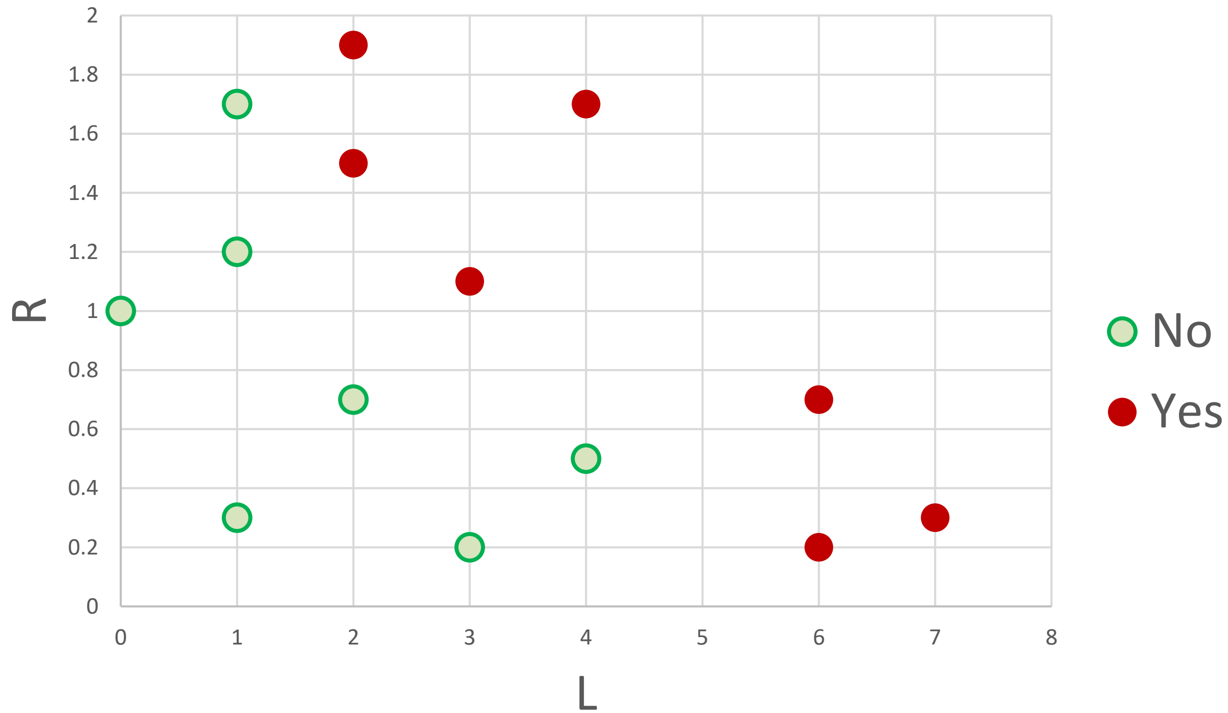
Classify

Model

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

Classification: lazy classifier

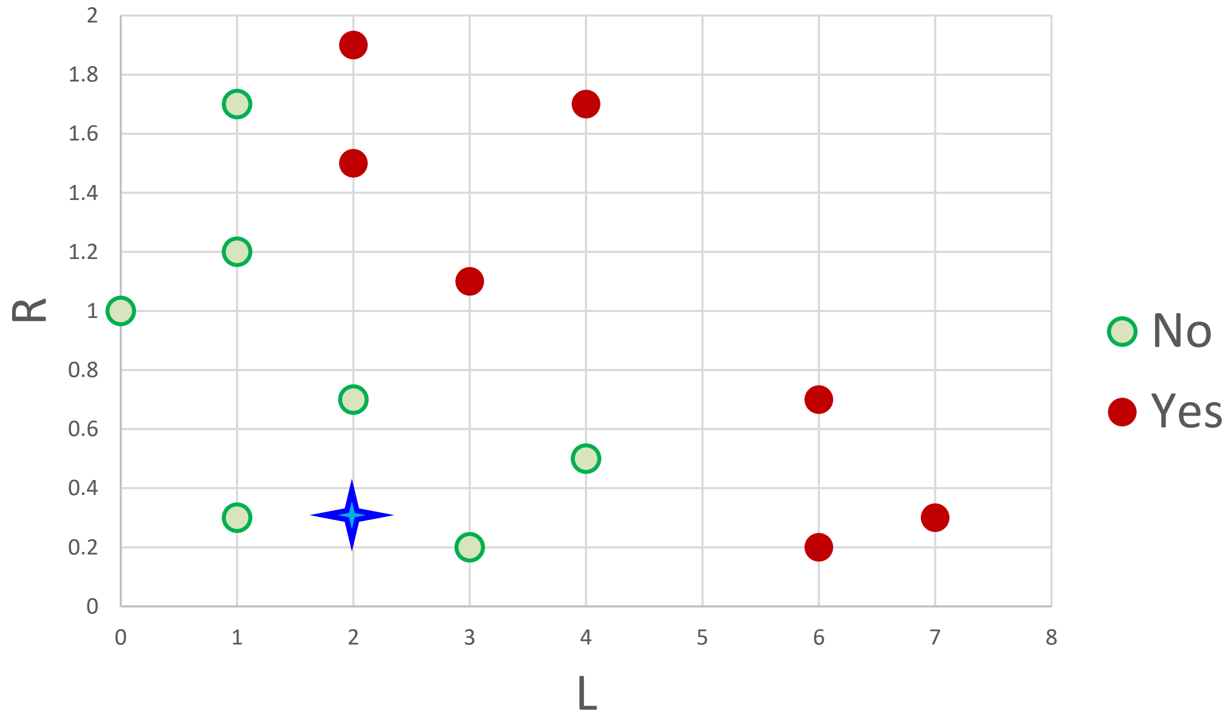
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Predicting bankruptcy: nearest neighbor

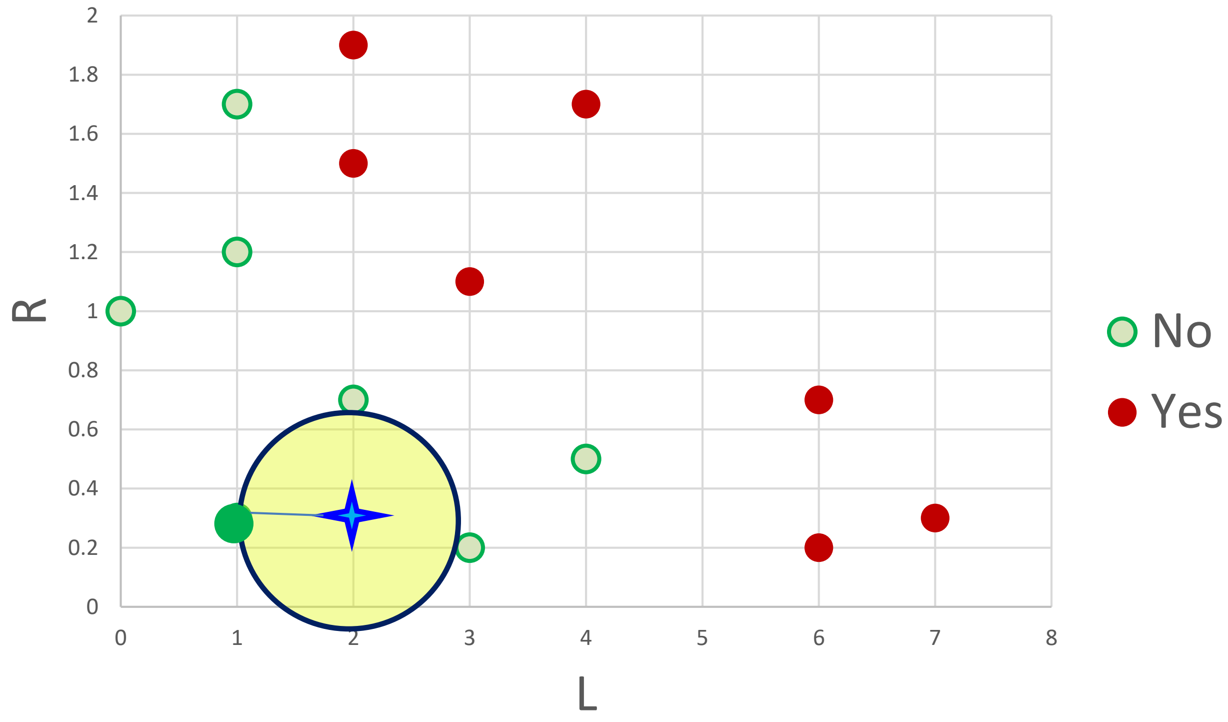
L	R
2	0.3



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

Predicting bankruptcy: nearest neighbor

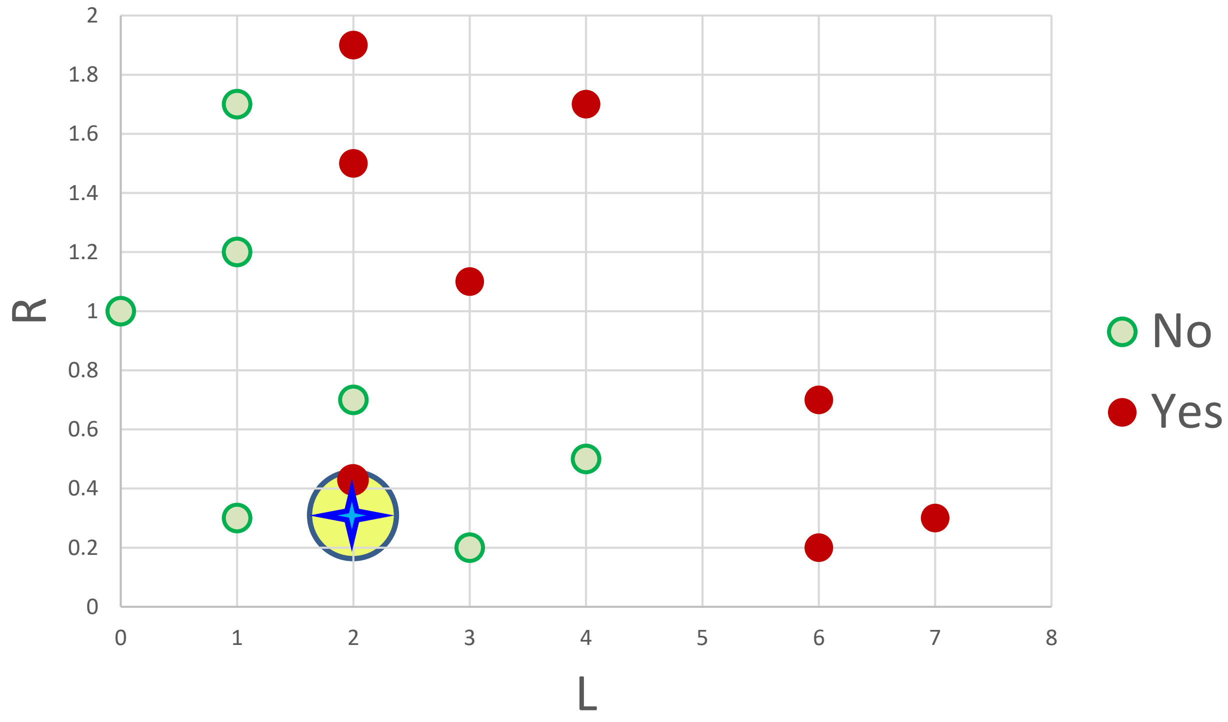
L	R
2	0.3



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

Predicting bankruptcy: noise

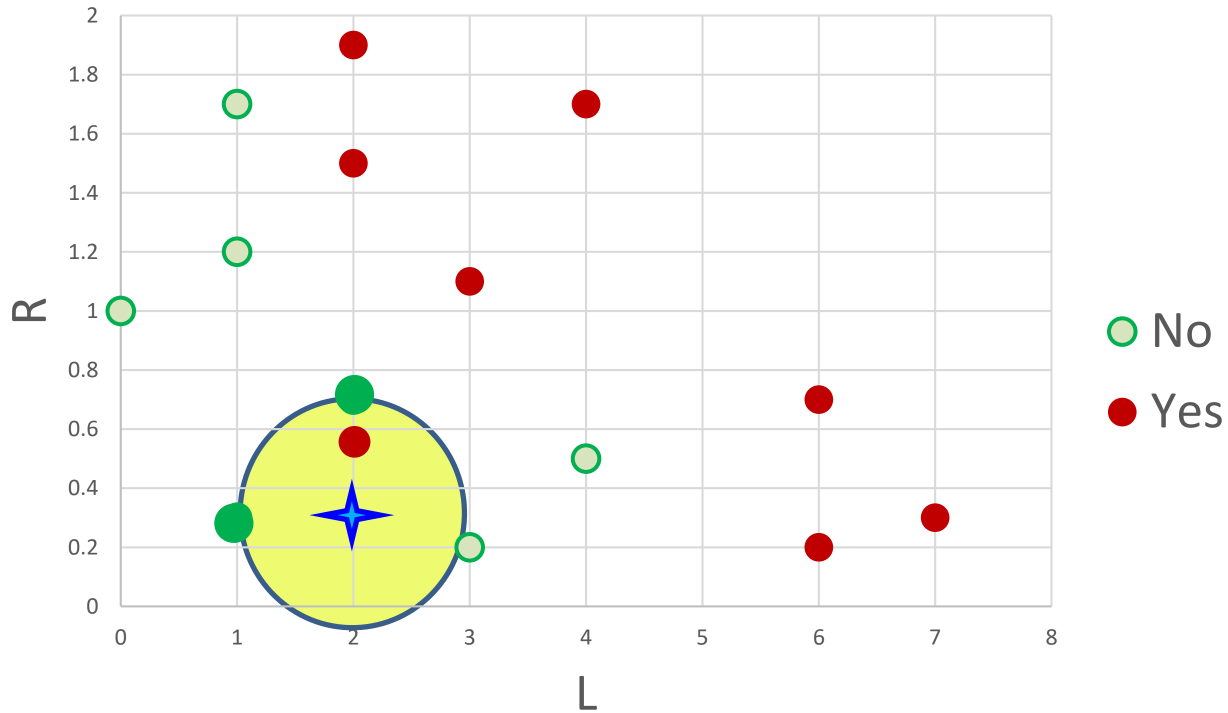
L	R
2	0.3



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

Predicting bankruptcy: *K* neighbors

L	R
2	0.3



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

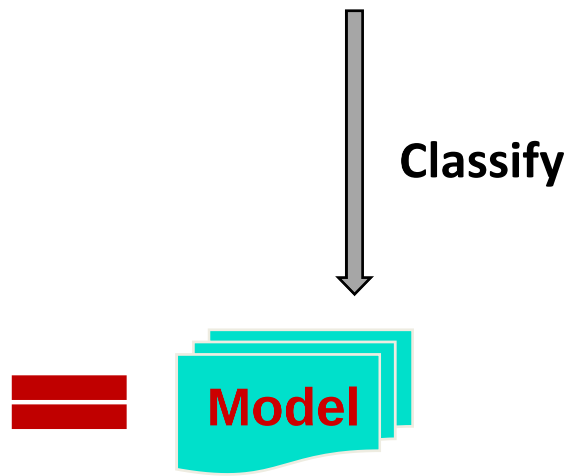
K-NN classifier

Training set

Late payments, L	Spending ratio, R	Bankruptcy
3	Very low	No
1	Very low	No
4	Low	No
2	Low	No
0	Normal	No
1	Medium	No
1	High	No
6	Very low	Yes
7	Very low	Yes
6	Low	Yes
3	Normal	Yes
2	Medium	Yes
4	High	Yes
2	High	Yes

New sample

L	R	B
2	Low	?



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

K-NN classification algorithm

Input:

set T of N labeled records,
K,
instance A to classify

Classification:

for *i* **from** 1 **to** *N*
compute ***distance*** $d\,(A,T_i)$
sort T *asc* by $d\,(A,T_i)$ into T_{sorted}
from top K records in T_{sorted}
extract class labels $L_{1 \ldots K}$

Output:

return ***combination*** $(L_{1 \ldots K})$

K-NN classification algorithm

Input:

set $\boldsymbol{T}$ of N labeled records,

$\boldsymbol{K}$,

instance $\boldsymbol{A}$ to classify

Classification:

for i **from** 1 **to** N

compute ***distance*** $d\,(\boldsymbol{A},\boldsymbol{T}_i)$

sort $\boldsymbol{T}$ *asc* by $d\,(\boldsymbol{A},\boldsymbol{T}_i)$ into $\boldsymbol{T}_{\text{sorted}}$

from top $\boldsymbol{K}$ records in $\boldsymbol{T}_{\text{sorted}}$

extract class labels $\boldsymbol{L}_{1\ldots K}$

Output:

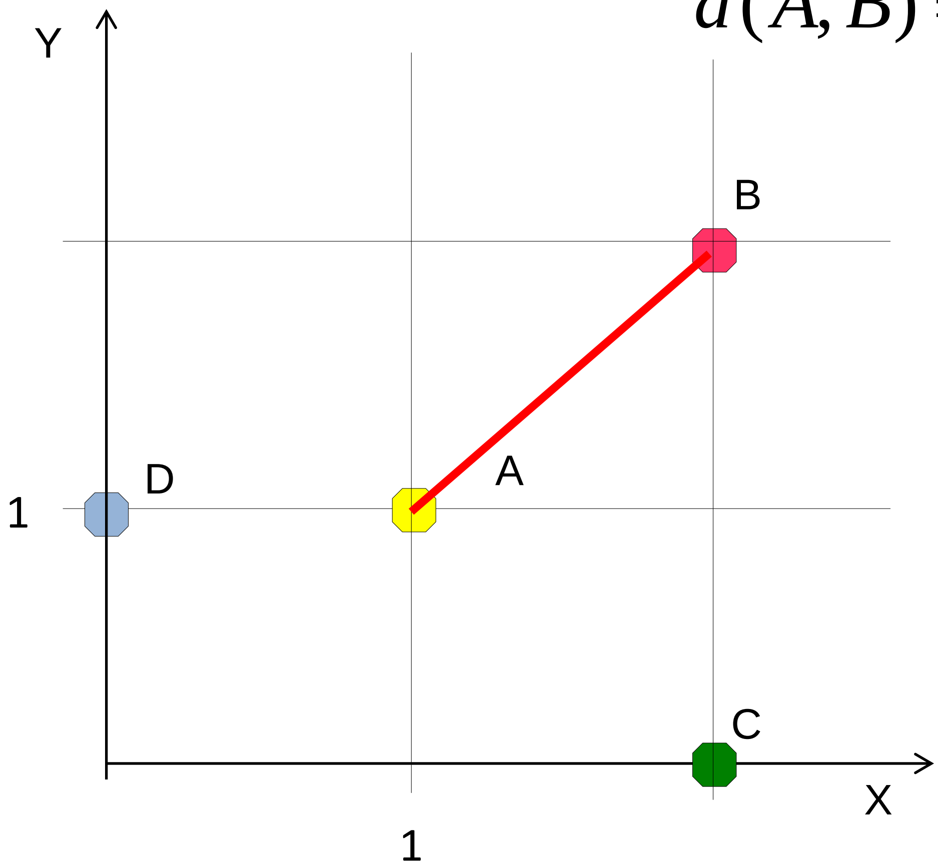
return ***combination*** $(\boldsymbol{L}_{1\ldots K})$

K-NN: round 1

- How many neighbors: choice of K
- Distance/similarity function
- Combining neighbor class labels

Simple distance function
Geometry: Euclidean distance

$$d(A,B) = \sqrt{|A_X - B_X|^2 + |A_Y - B_Y|^2}$$



$d(A,B)=?$

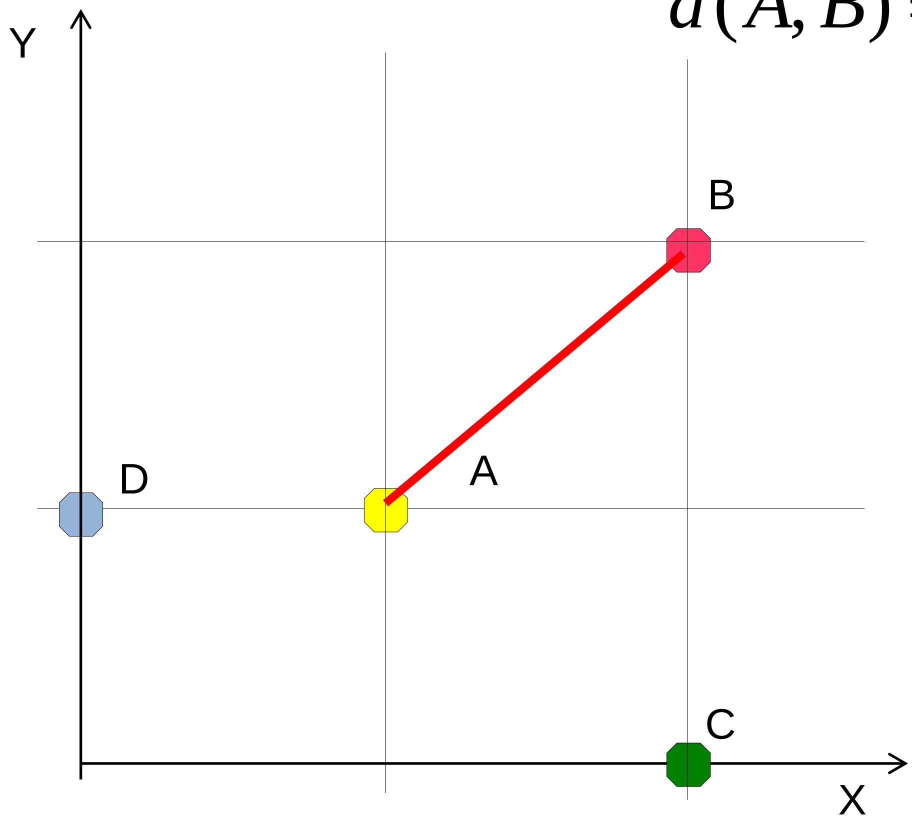
$d\ (B,C)=?$

$d(C,D)=?$

Simple distance function

Geometry: Euclidean distance

$$d(A,B) = \sqrt{|A_X - B_X|^2 + |A_Y - B_Y|^2}$$

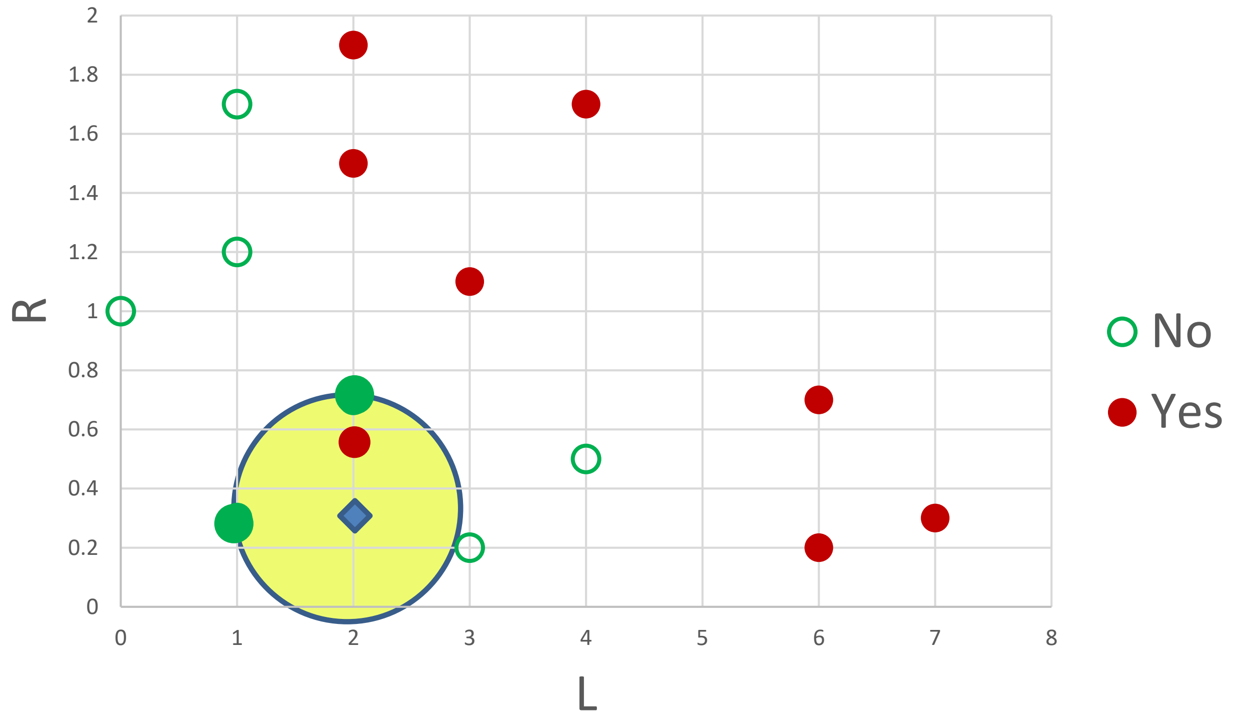


For *N* dimensions:

$$d(A,B) = \sqrt{\sum_{i=1}^{N} |A_i - B_i|^2}$$

Simple combination function for classification : **majority voting**

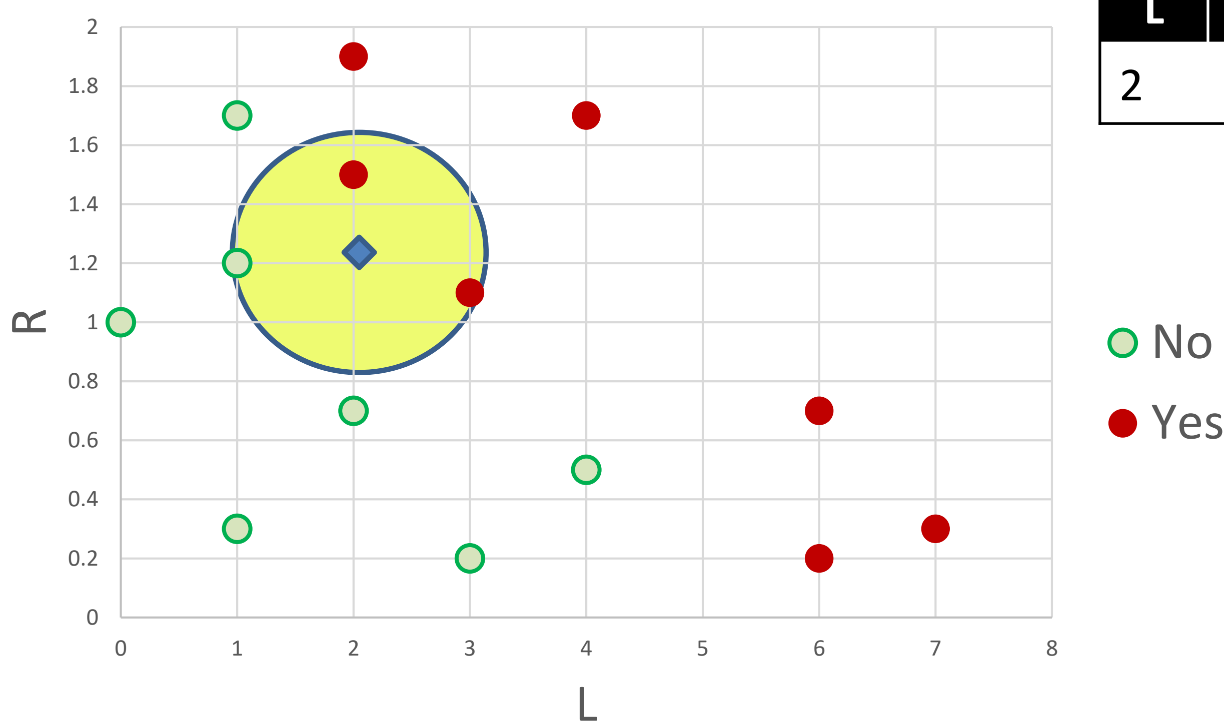
L	R
2	0.3



Classified as non-bankrupt

Simple combination function for prediction: **average**

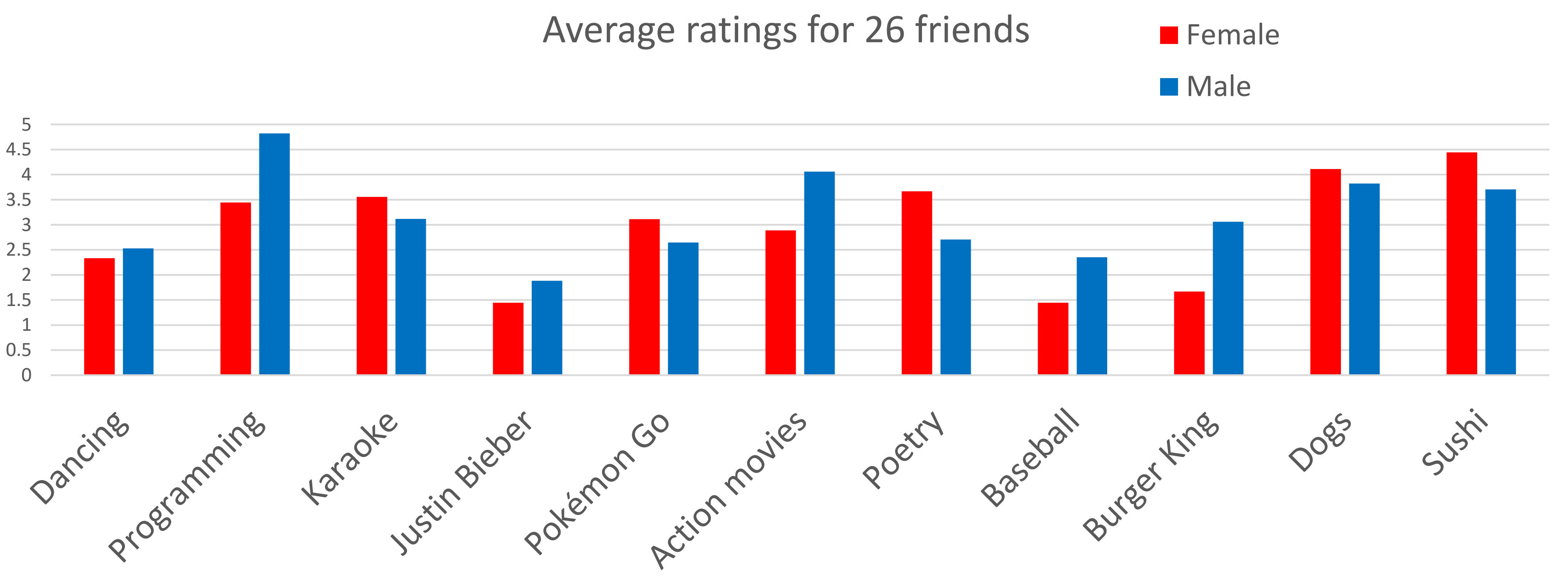
L	R	D
3	0.2	0
1	0.3	0
4	0.5	0
2	0.7	0
0	1	0
1	1.2	0
1	1.7	0
6	0.2	50K
7	0.3	100K
6	0.7	500K
3	1.1	25K
2	1.5	30K
4	1.7	150K
2	1.9	40K



L	R
2	1.2

Predicted default:
(0+30+25)/3=18K

My friends dataset

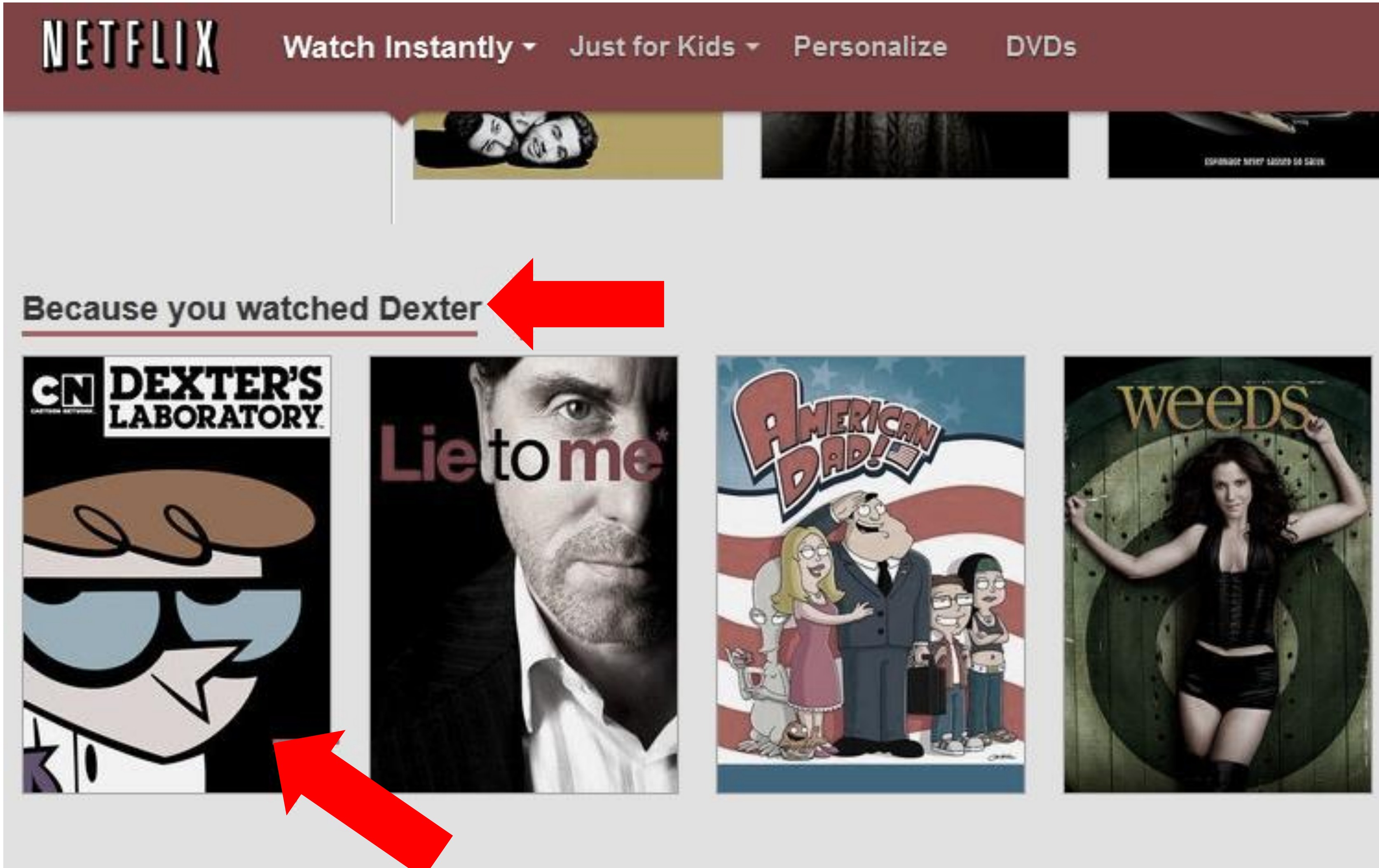


https://hope.simons-rock.edu/~mbarsky/intro19/lectures/data/predict/

K-NN: round 2

I. Distance/similarity function
II. How many neighbors: choice of K
III. Combining neighbor votes

How do we define proximity?

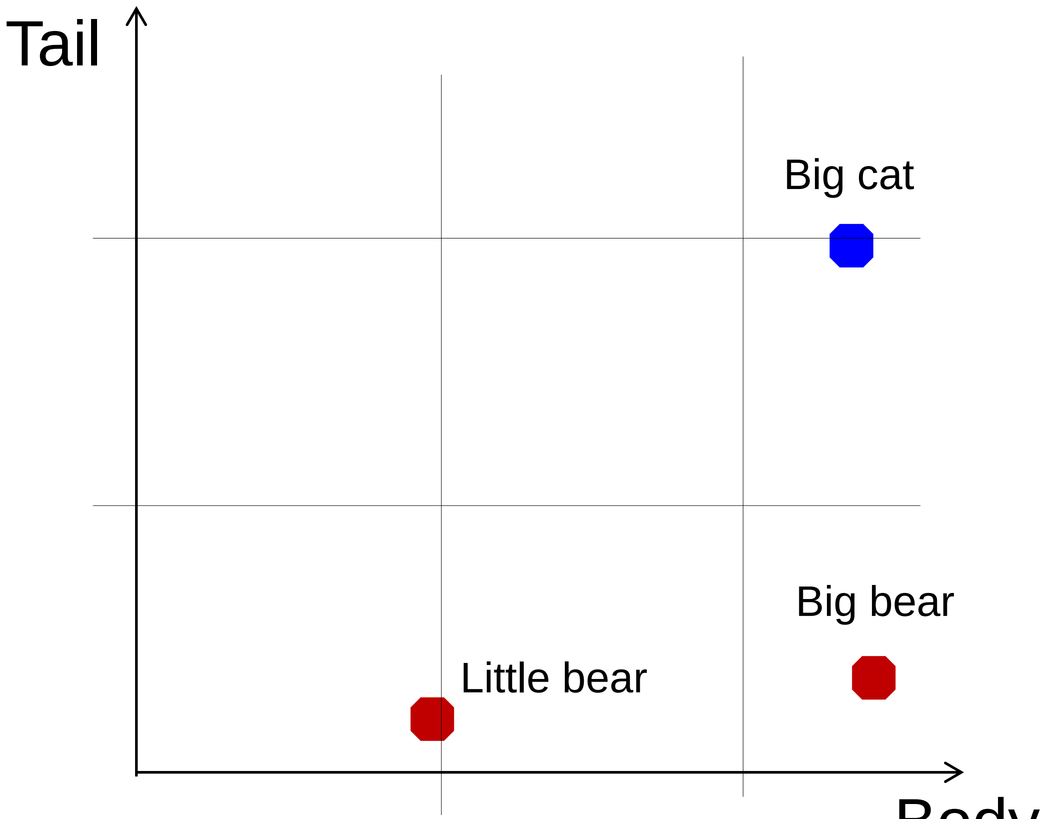


Cat or bear?

Cat or bear?

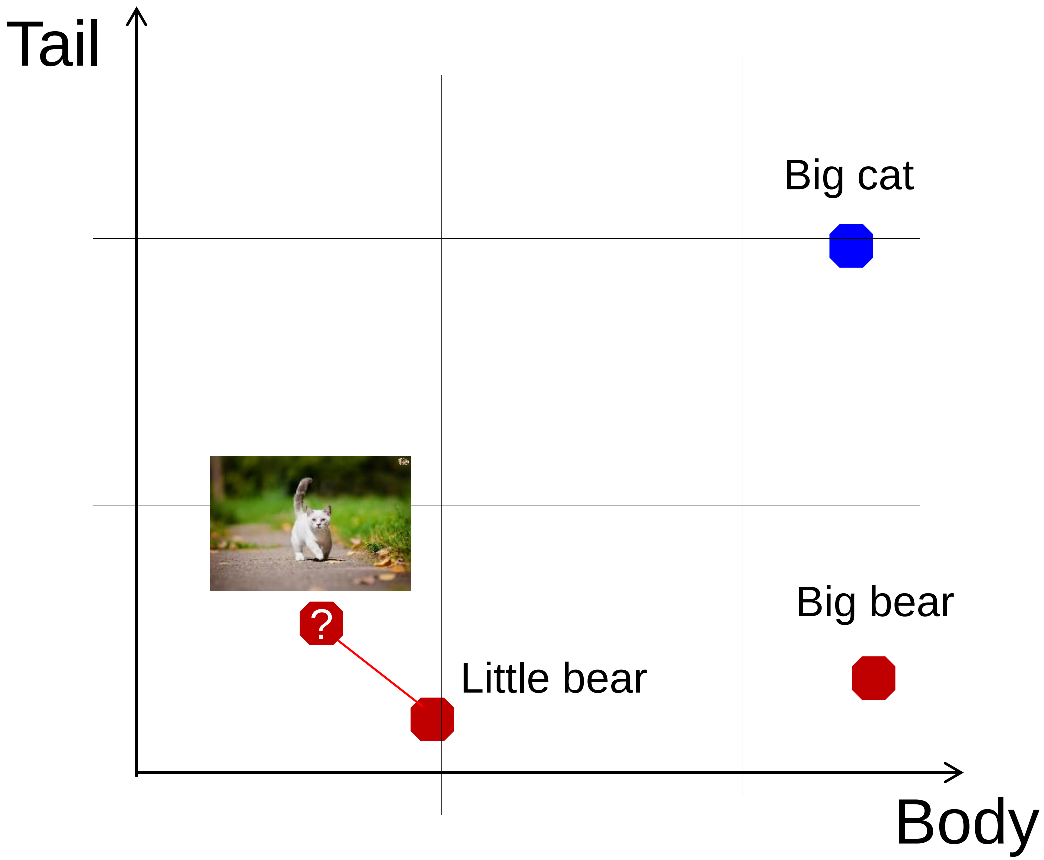
Cat or bear?

Cat or bear classifier



Cat or bear?

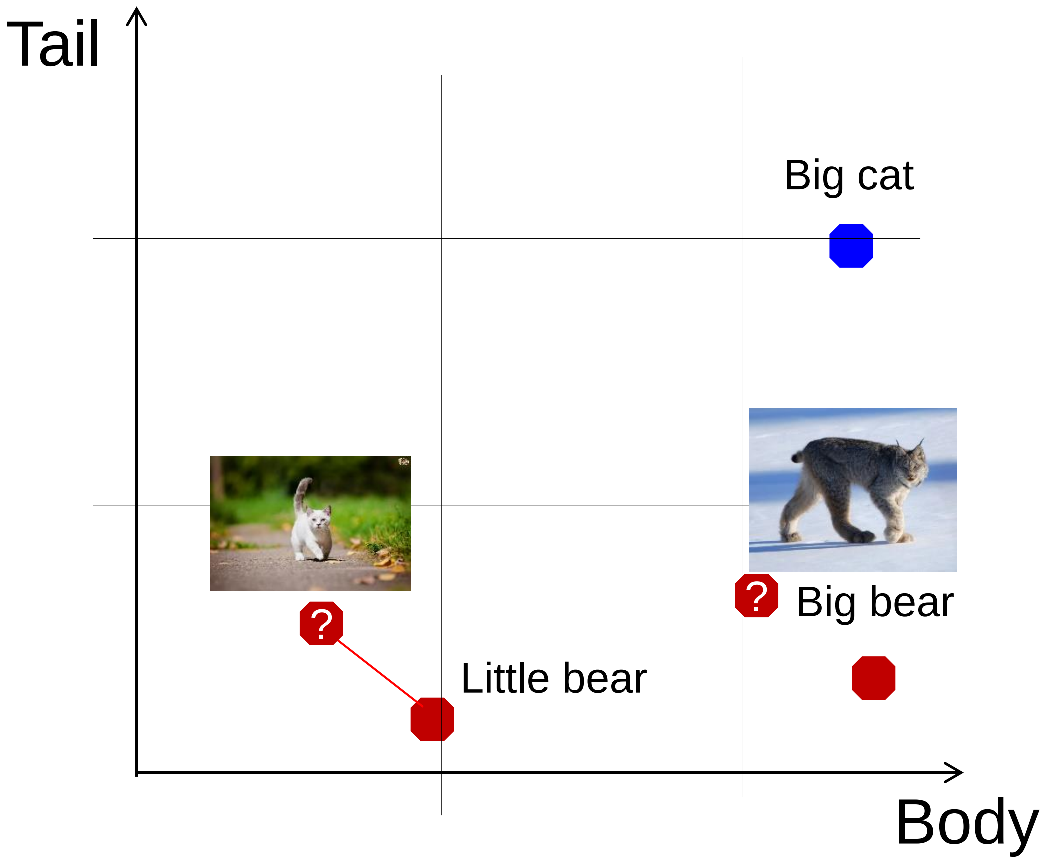
Cat or bear?



Cat or bear?

Canadian Lynx

Cat or bear?

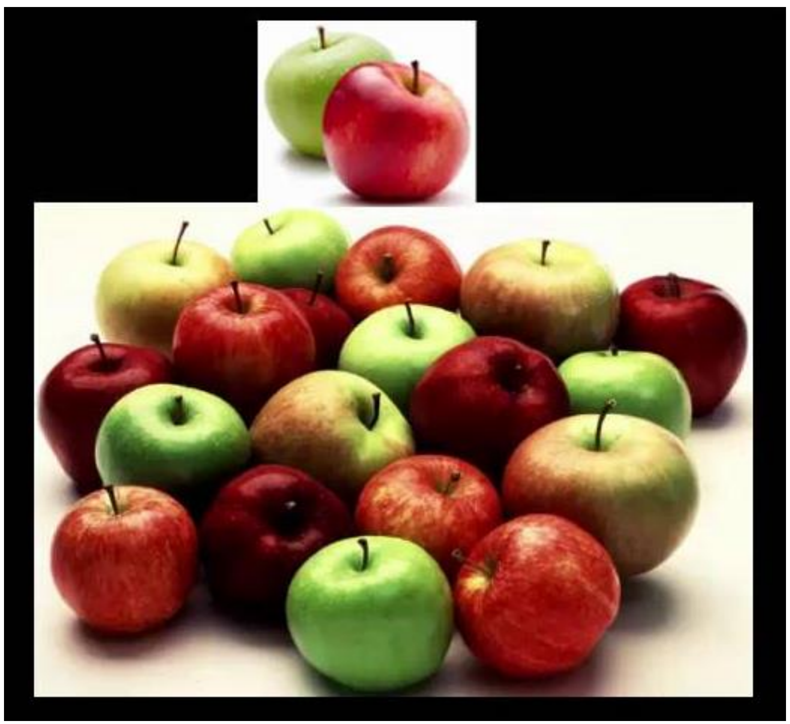


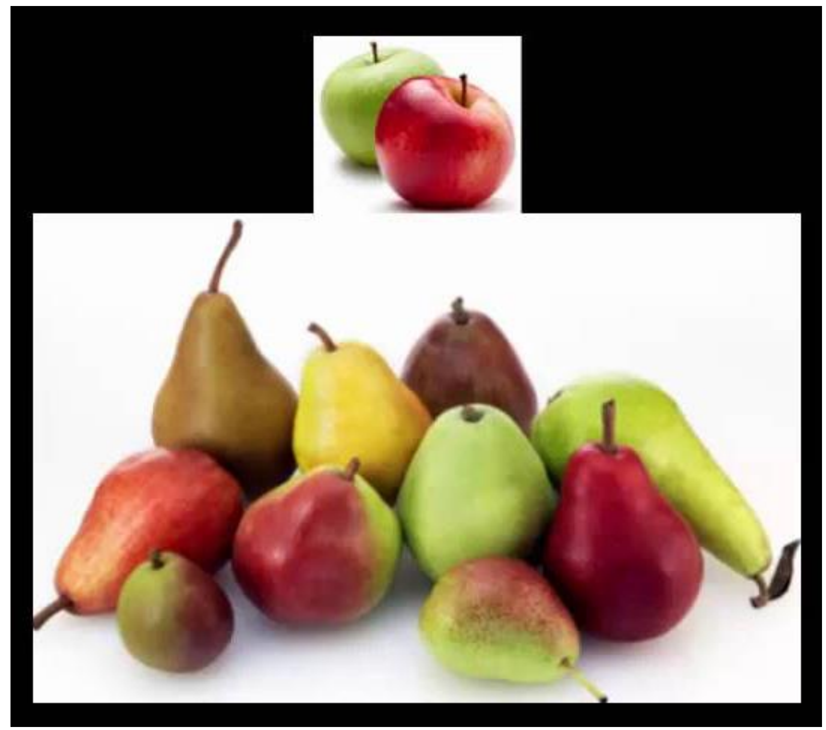
Data-dependent proximity

Two Apples among Apples

are less similar than

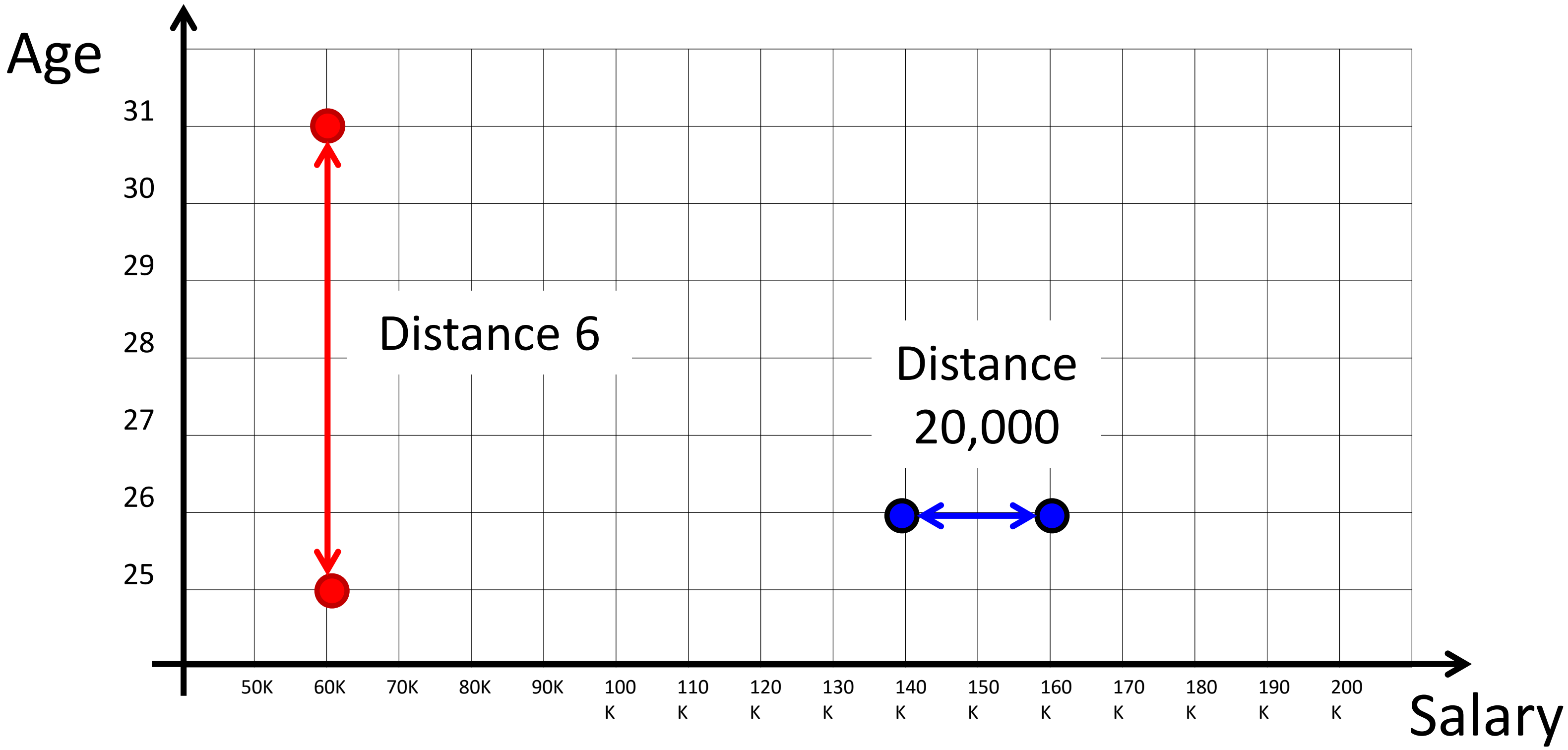
Two Apples among Pears





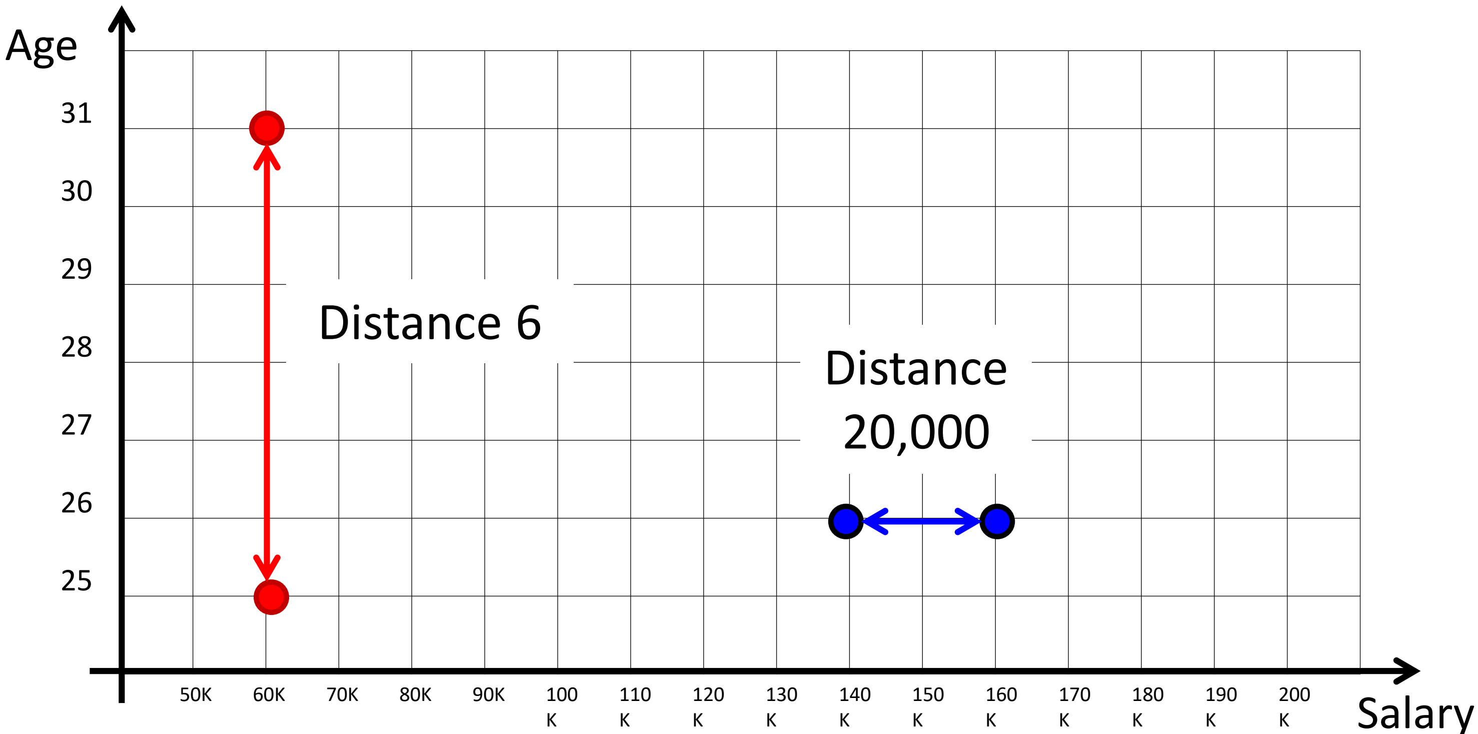
S. Aryal, K. M. Ting, G. Haffari and T. Washio, "Mp-Dissimilarity: A Data Dependent Dissimilarity Measure," *ICDM*, 2014

Need to scale numeric features



Scaling to a common scale

$$a_i = \frac{v_i - \min(all\ v)}{\max(all\ v) - \min(all\ v)}$$

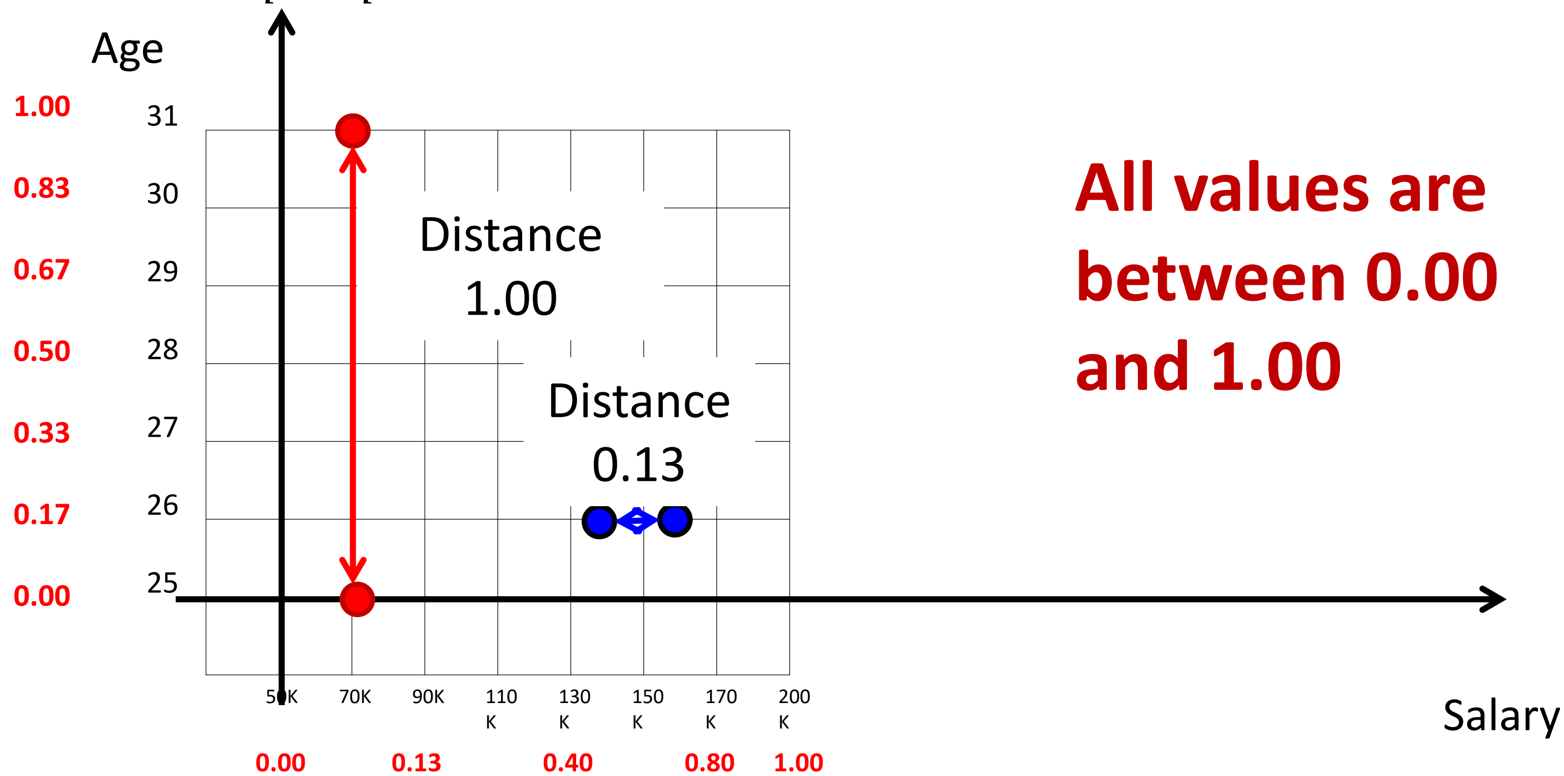


Scaling to a common scale

$$a_i = \frac{v_i - \min(all\ v)}{\max(all\ v) - \min(all\ v)}$$

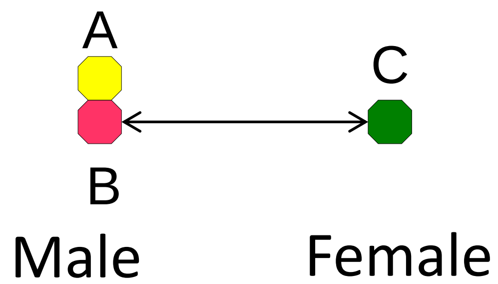
For Age: a_i=(v_i-25)/(31-25)

For Salary: a_i=(v_i-50,000)/(200,00-50,00)



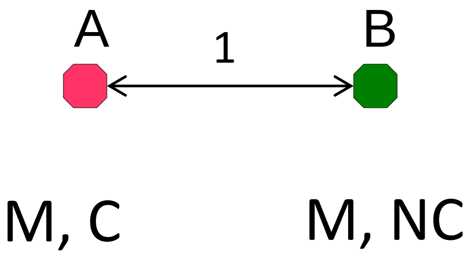
All values are between 0.00 and 1.00

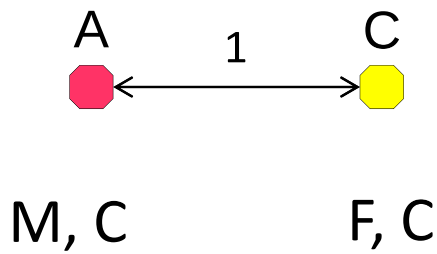
Distance between categorical attributes



- d(A,B)=?
- d(B,C)=?
- d(A,C)=?

Combining distance between categorical attributes



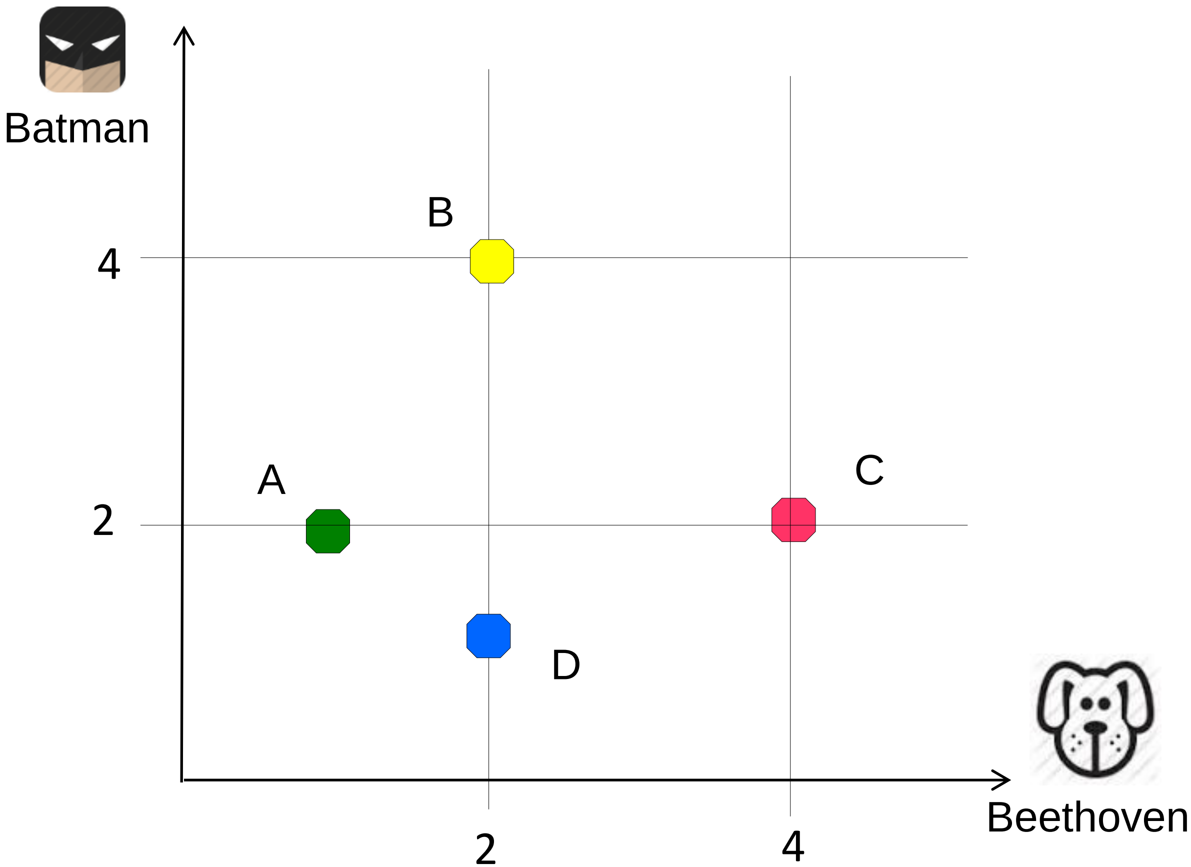


	Gender	Education	Age
A	Male	College	21
B	Male	No college	21
C	Female	College	21

$$d(\mathrm{A},\mathrm{B})=\sqrt{0+1+0}=1$$

$$d(\mathrm{A},\mathrm{C})=\sqrt{1+0+0}=1$$

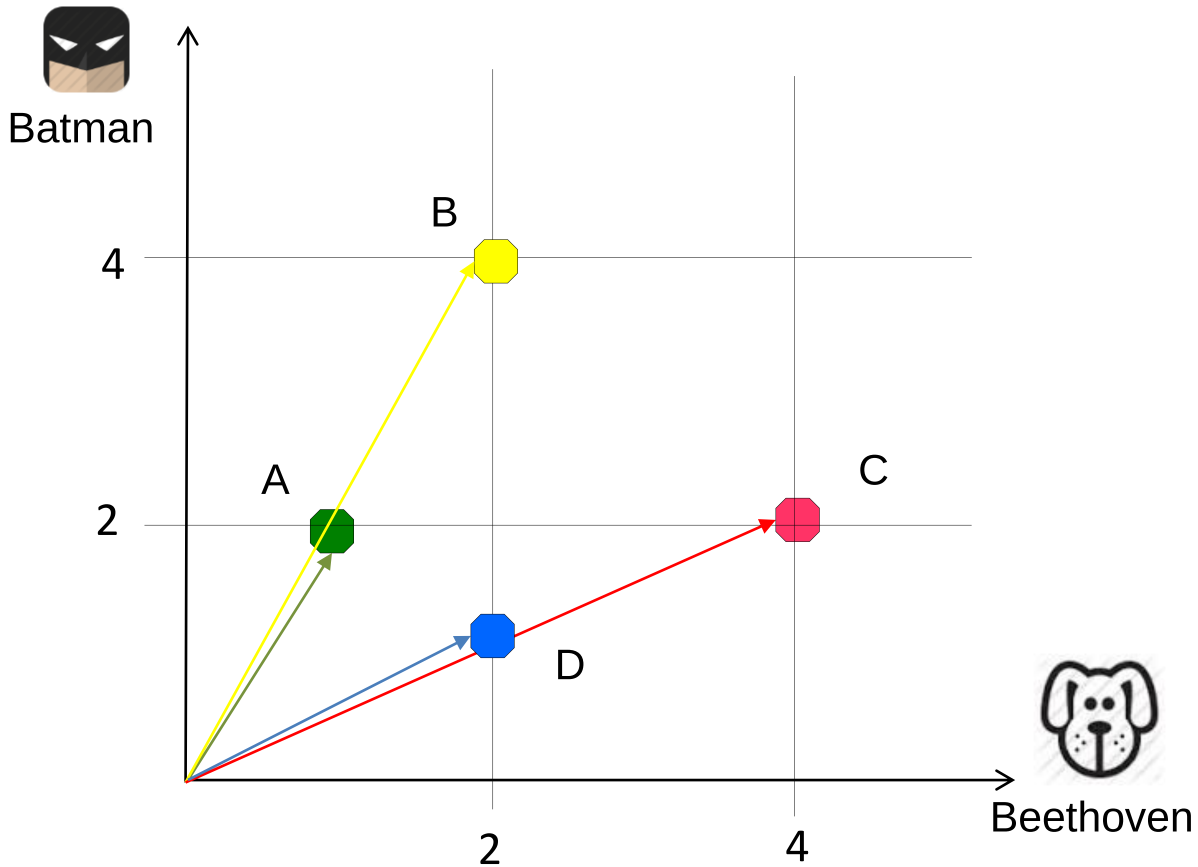
Coordinate-based similarity: Euclidean distance



A: (1,2)
B: (2,4)
C: (4,2)
D: (2,1)

Movie rankings

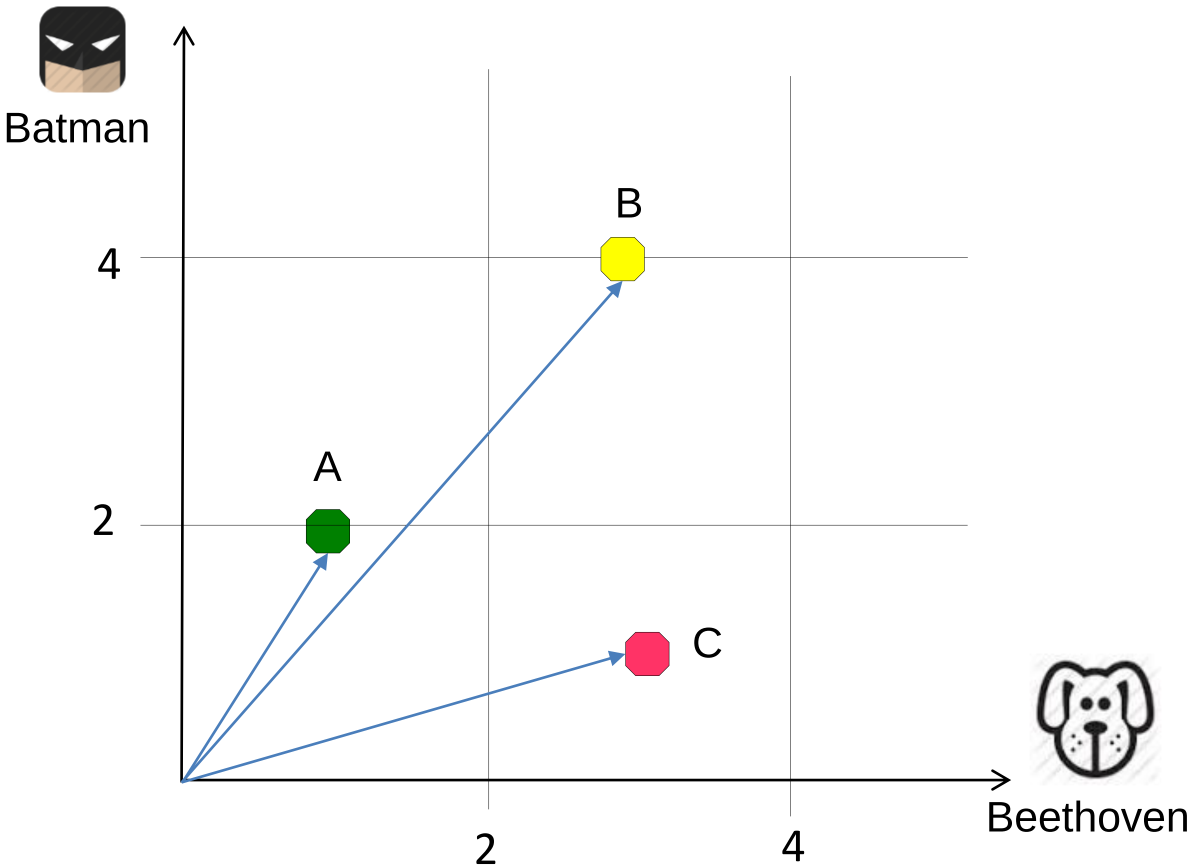
Vector-based similarity: cosine of an angle between vectors



A: (1,2)
B: (2,4)
C: (4,2)
D: (2,1)

Movie rankings

Vector-based similarity: cosine of an angle between vectors

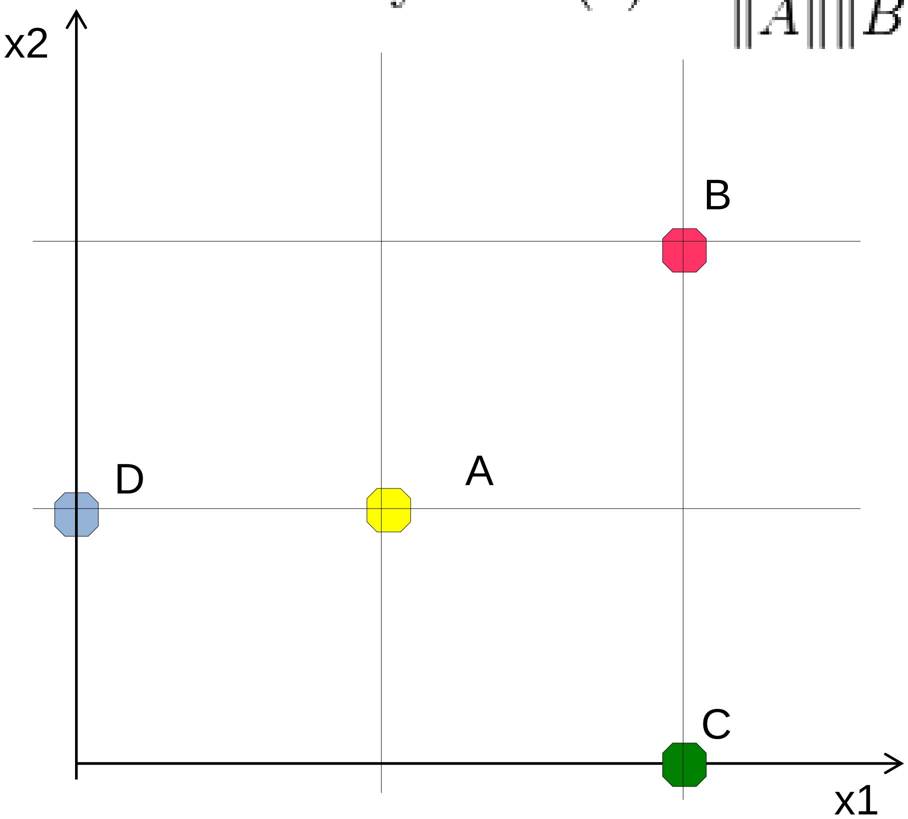


A: (1,2)
B: (3,4)
C: (3,1)

Movie rankings

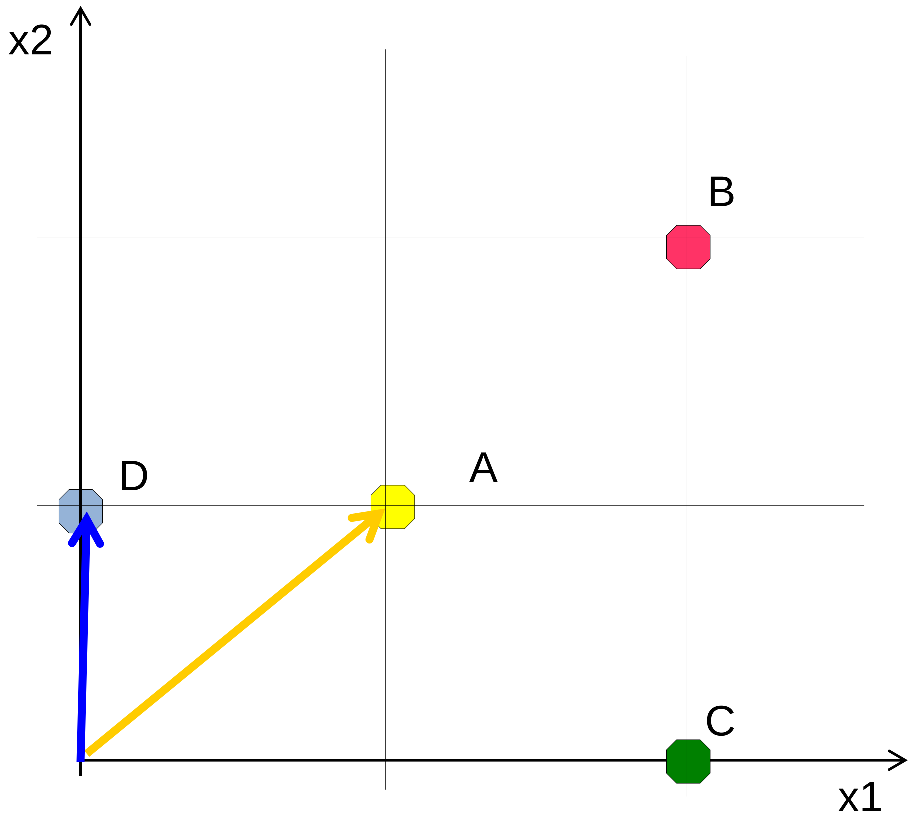
Cosine similarity formula

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$



Cosine similarity computation

$$s(A,B)=\cos(\mathbf{A},\mathbf{B}) = (\mathbf{A} \bullet \mathbf{B}) \,/\, |\mathbf{A}| \text{x} |\mathbf{B}|$$



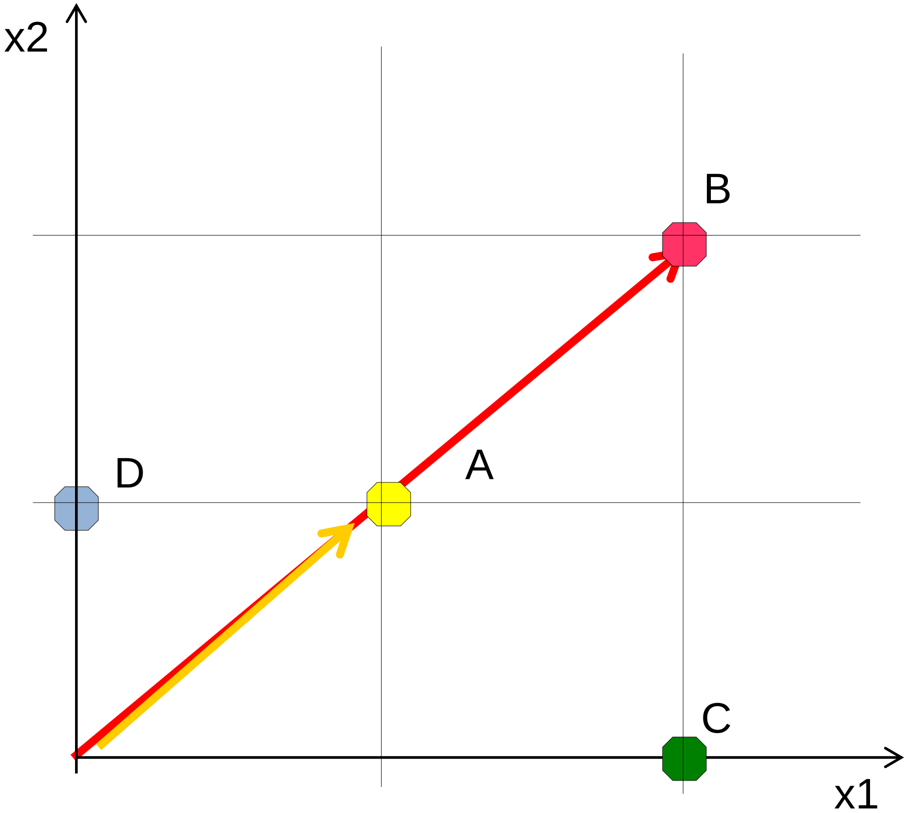
$\mathbf{A}=(1,1)$

$\mathbf{D}=(0,1)$

$$s(A,D)=\cos(\mathbf{A},\mathbf{D})$$
$$=1/\sqrt{2}\approx 0.7$$

Cosine similarity computation

$$s(A,B)=cos(\mathbf{A},\mathbf{B}) = (\mathbf{A} \bullet \mathbf{B}) / |\mathbf{A}| \text{x} |\mathbf{B}|$$

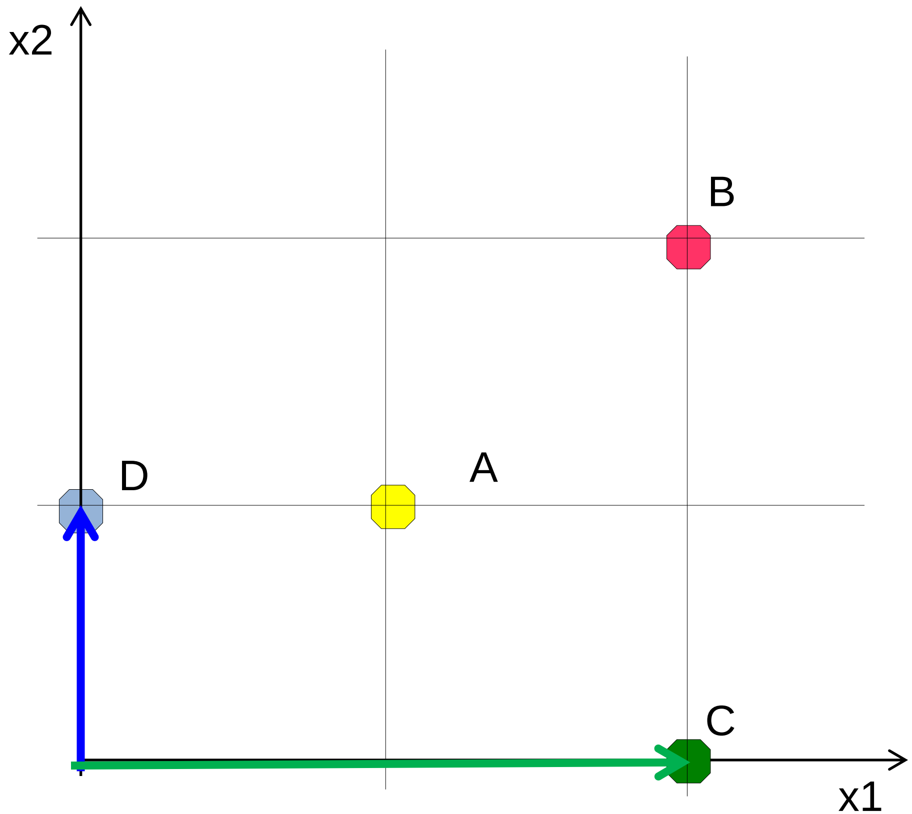


$\mathbf{A}=(1,1)$

$\mathbf{B}=(2,2)$

Cosine similarity computation

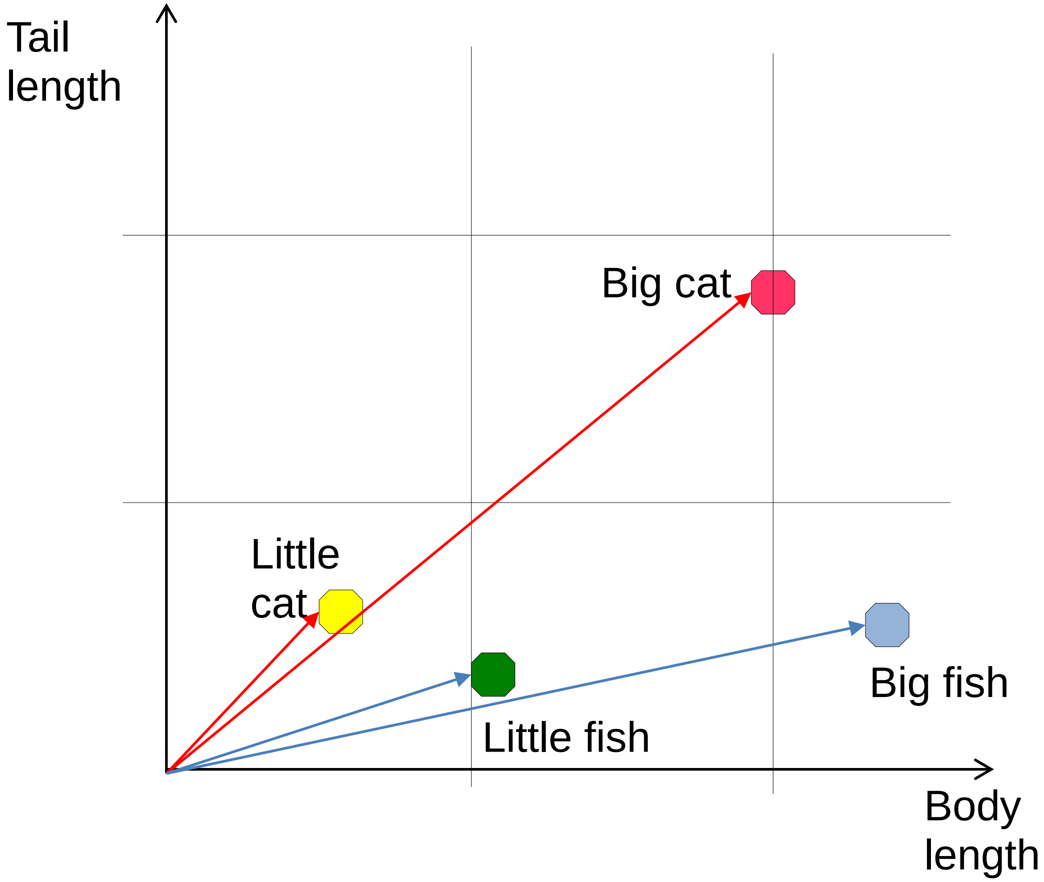
$$s(A,B)=cos(\mathbf{A},\mathbf{B}) = (\mathbf{A} \bullet \mathbf{B}) / |\mathbf{A}| x |\mathbf{B}|$$

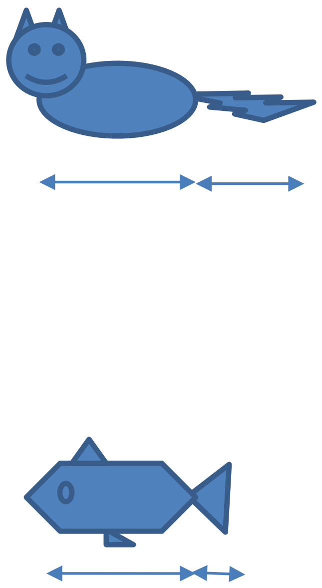


$\mathbf{C}=(2,0)$

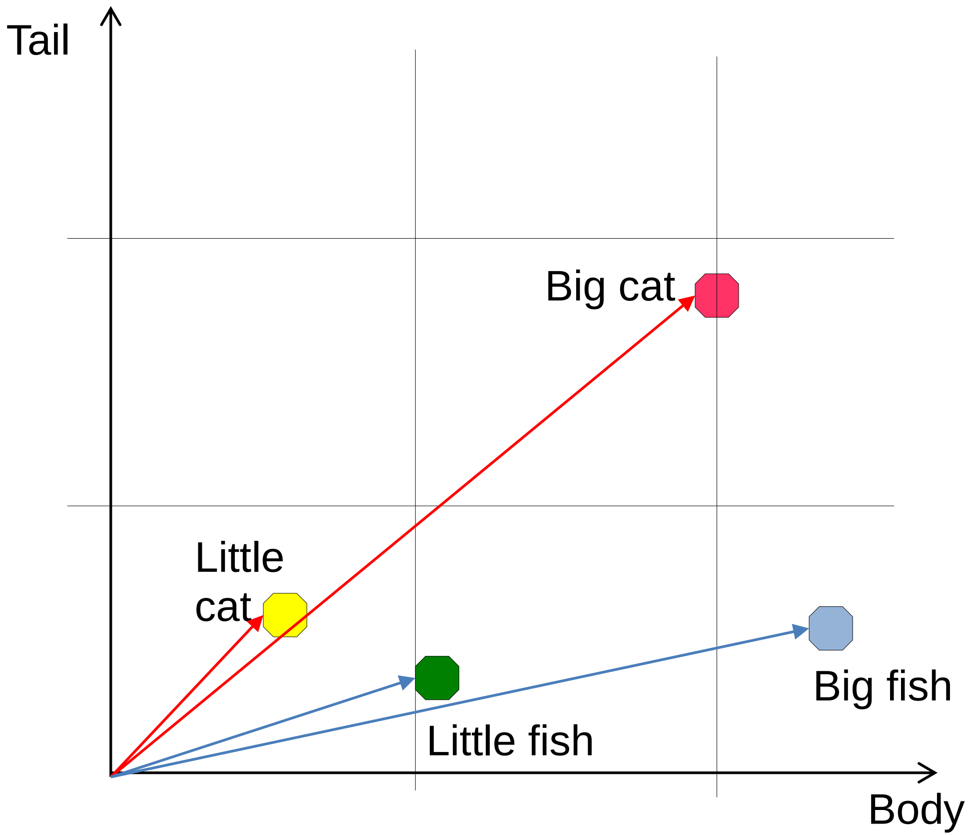
$\mathbf{D}=(0,1)$

Cosine similarity: fish vs. cat





Cosine similarity: fish vs. cat



Canadian Lynx

Siamese fighting fish

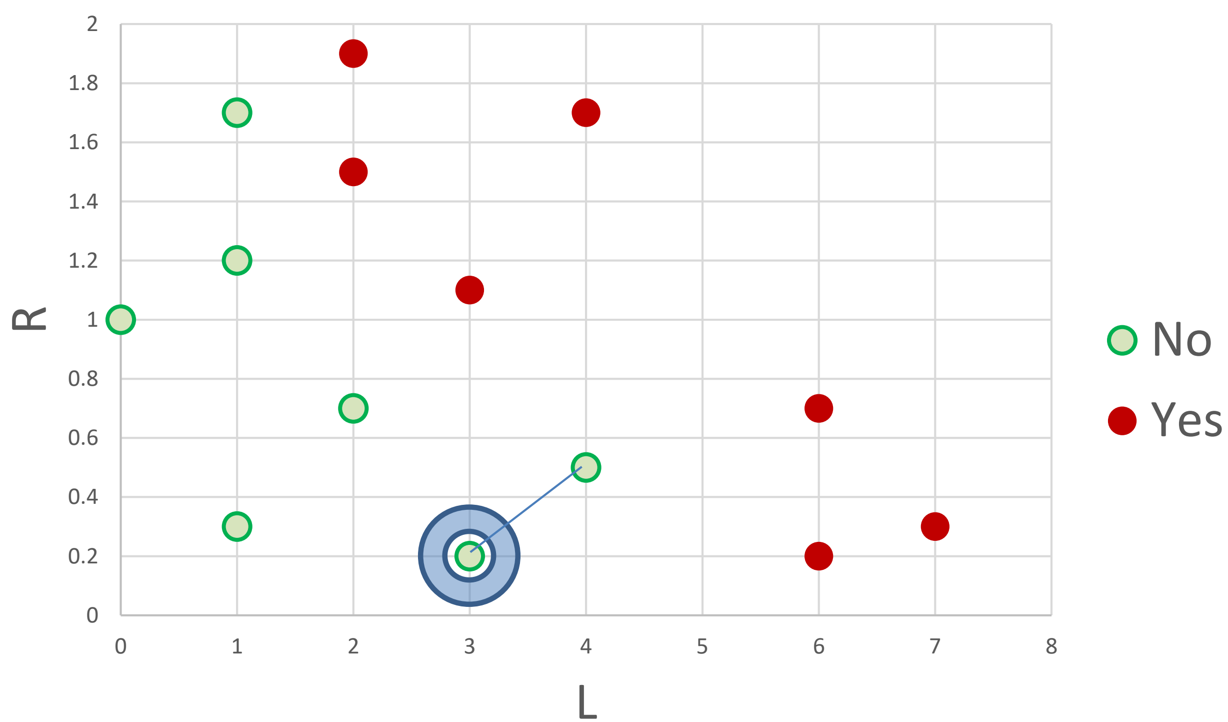
How many neighbors?

application-dependent

- Vary K from 1 to N
- Use cross-validation to find optimal value of K

Leave-one-out cross validation: *K*=1

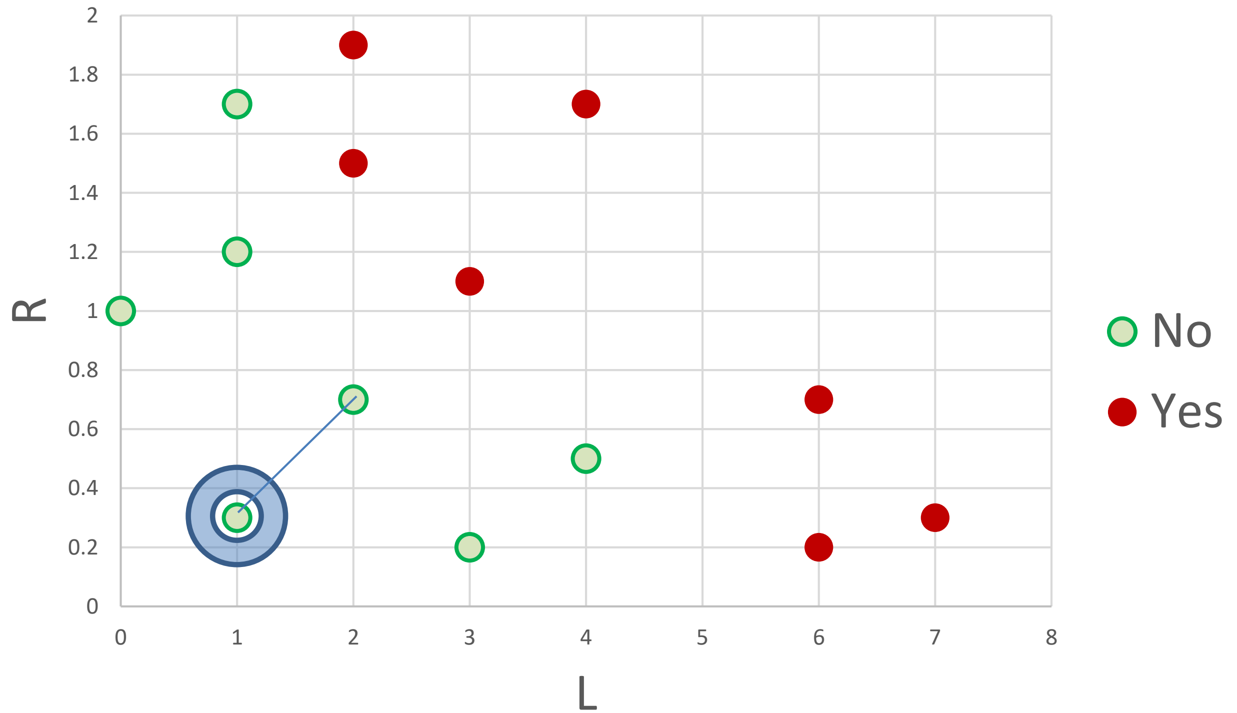
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=1

L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	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

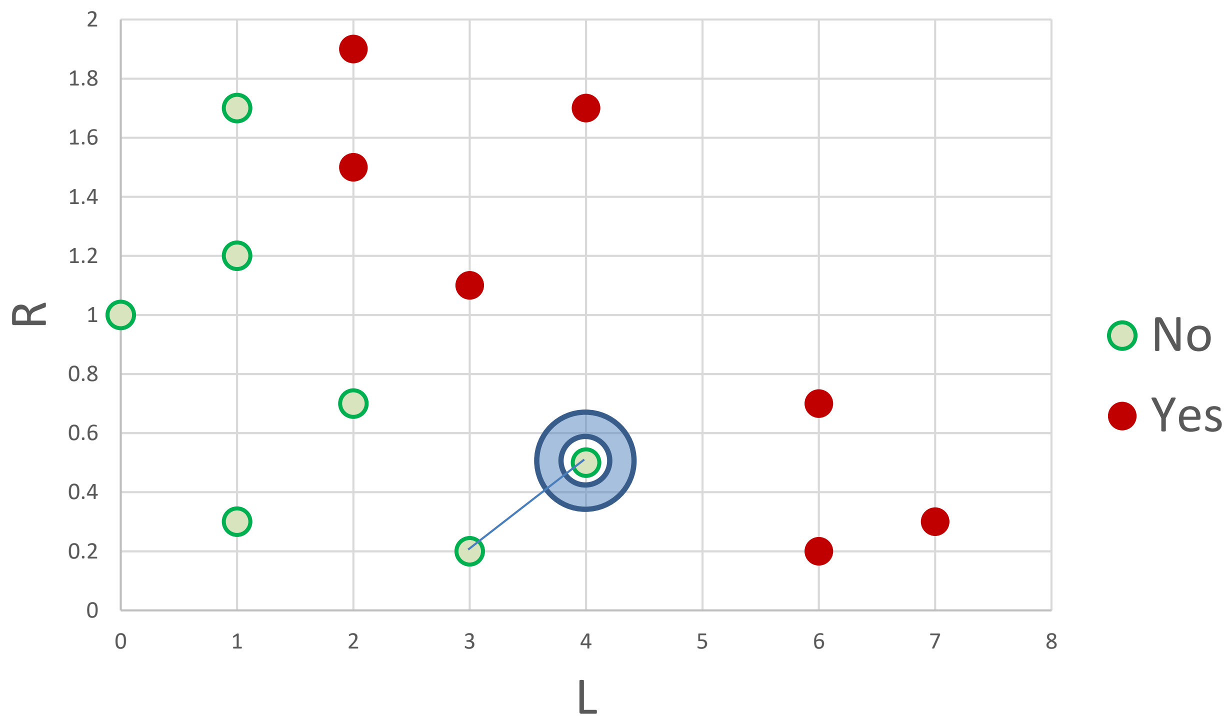


L: #late payments / year

R: expenses / income ratio

Leave-one-out cross validation: *K*=1

L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	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

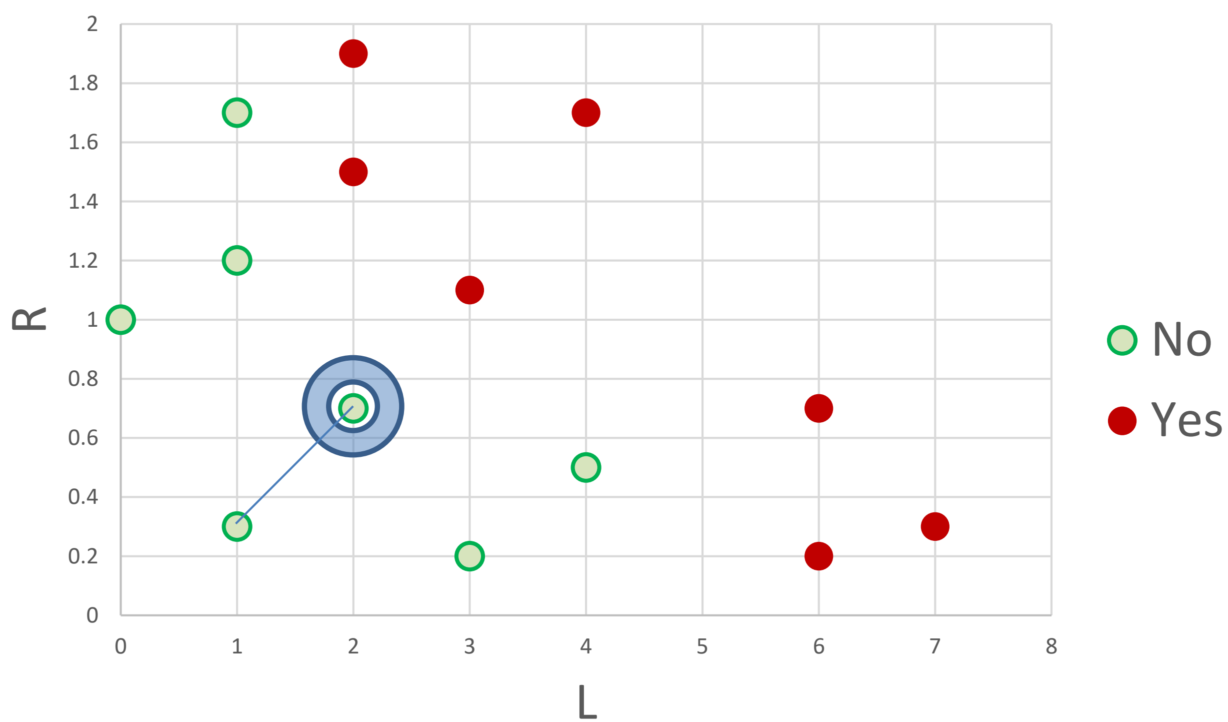


L: #late payments / year

R: expenses / income ratio

Leave-one-out cross validation: *K*=1

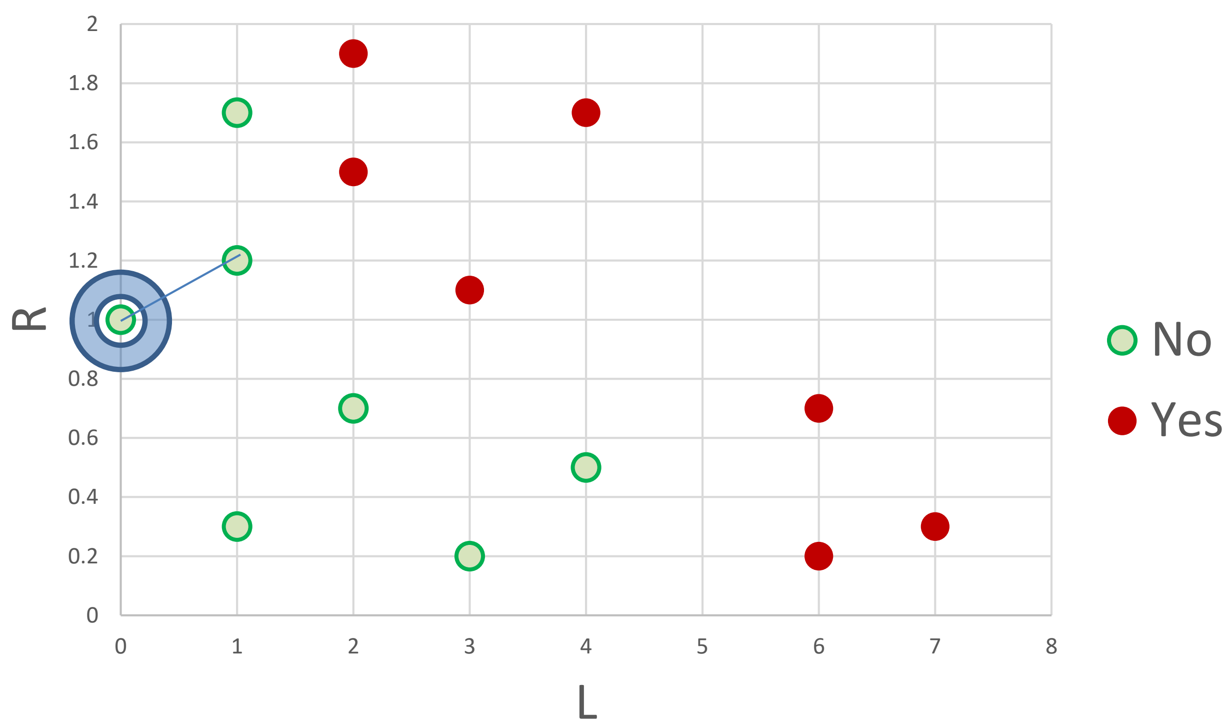
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=1

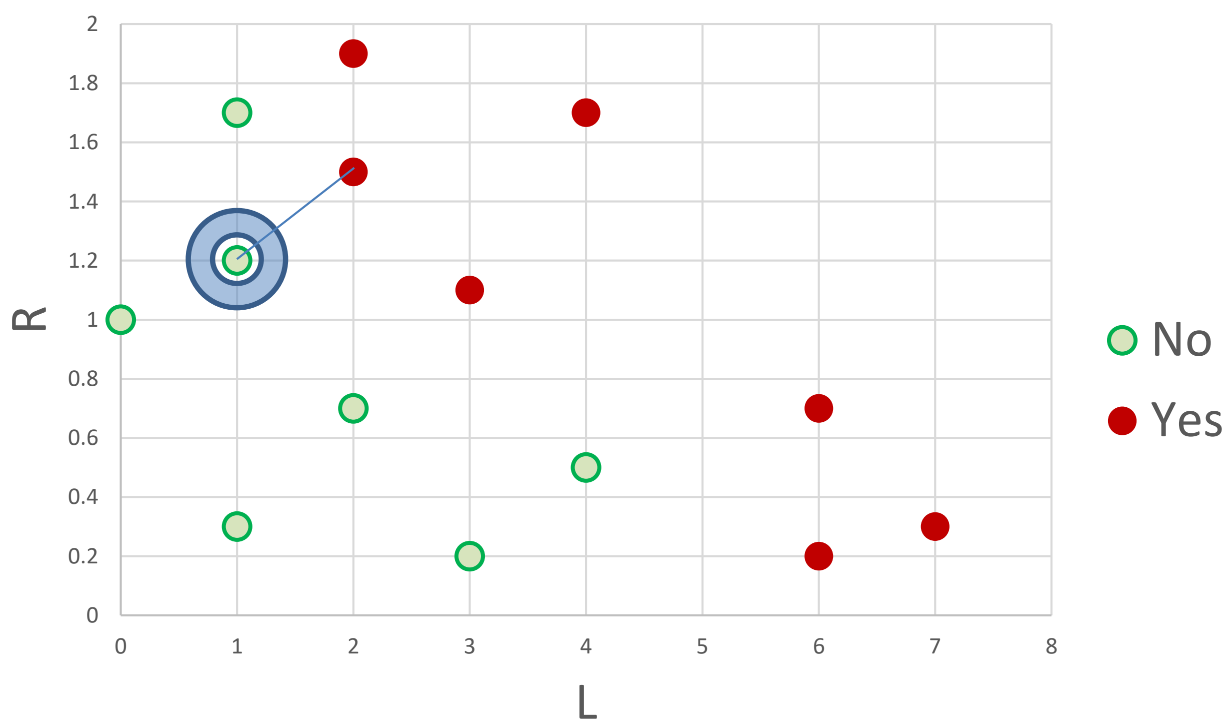
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=1

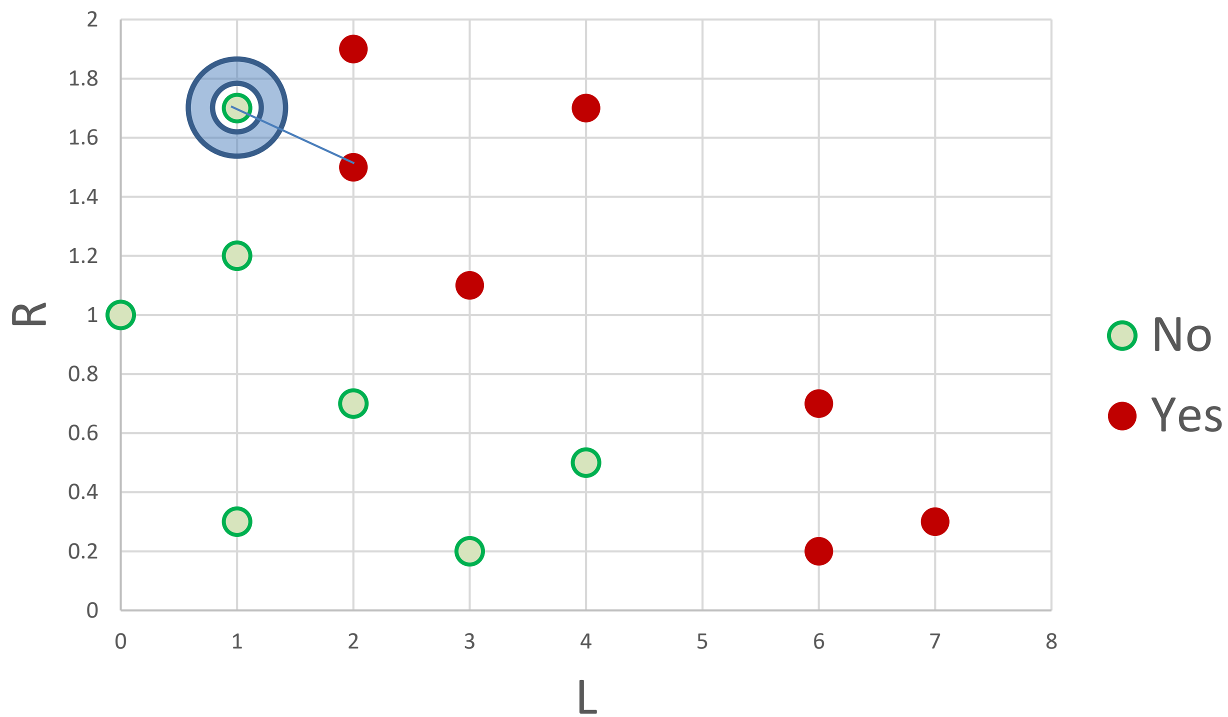
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=1

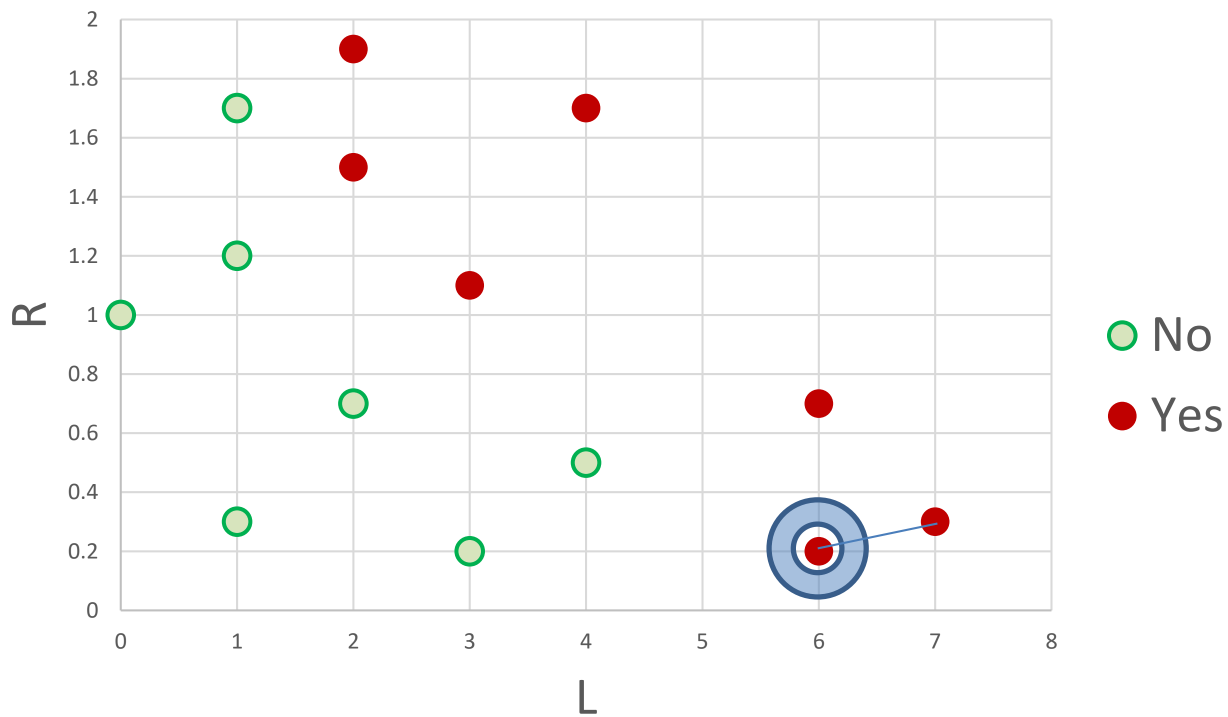
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=1

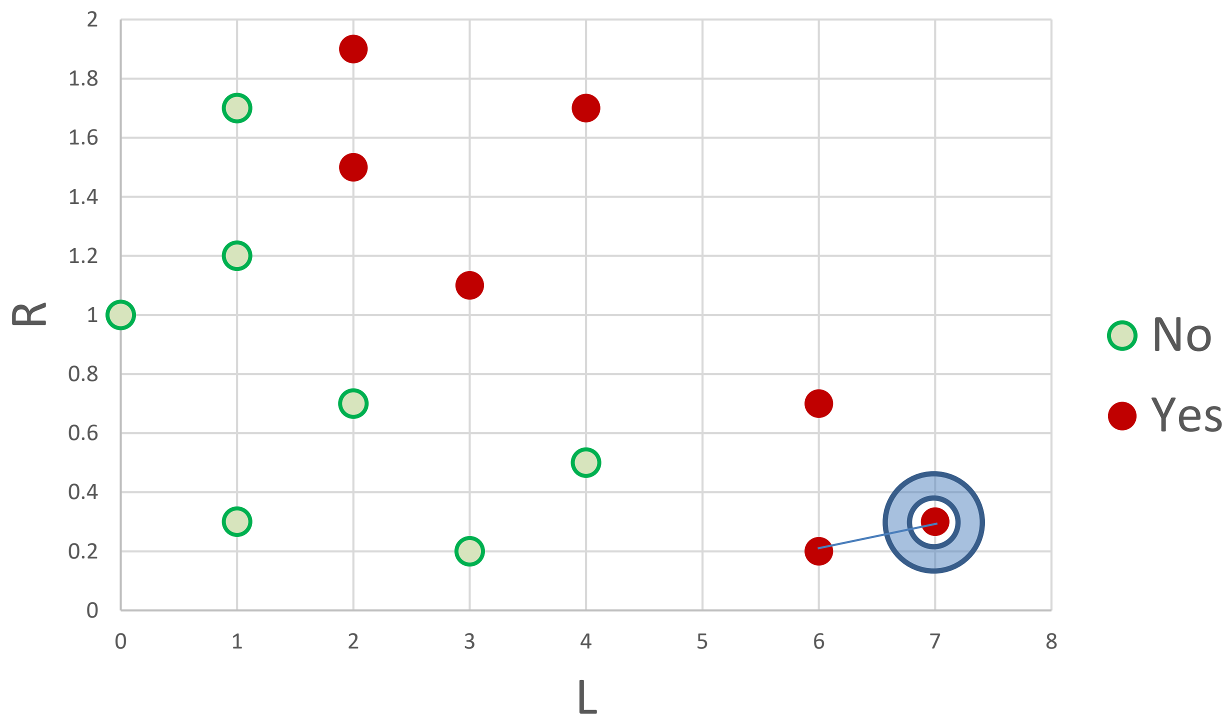
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=1

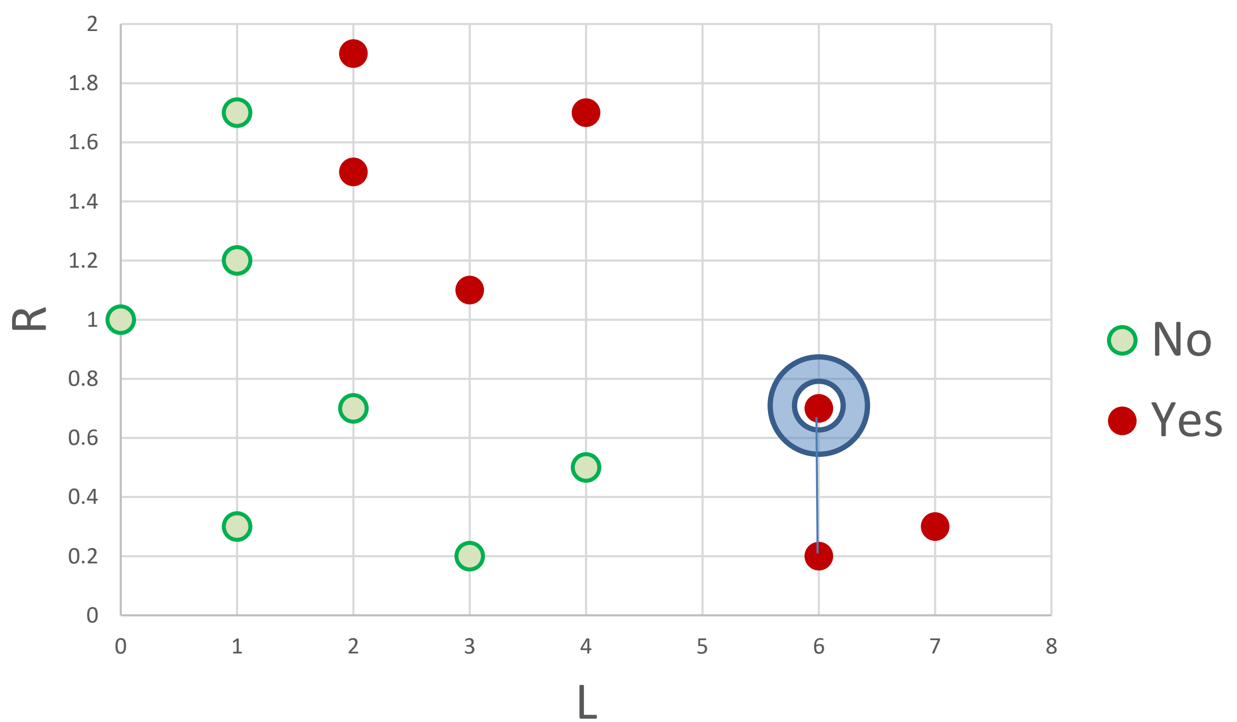
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=1

L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	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

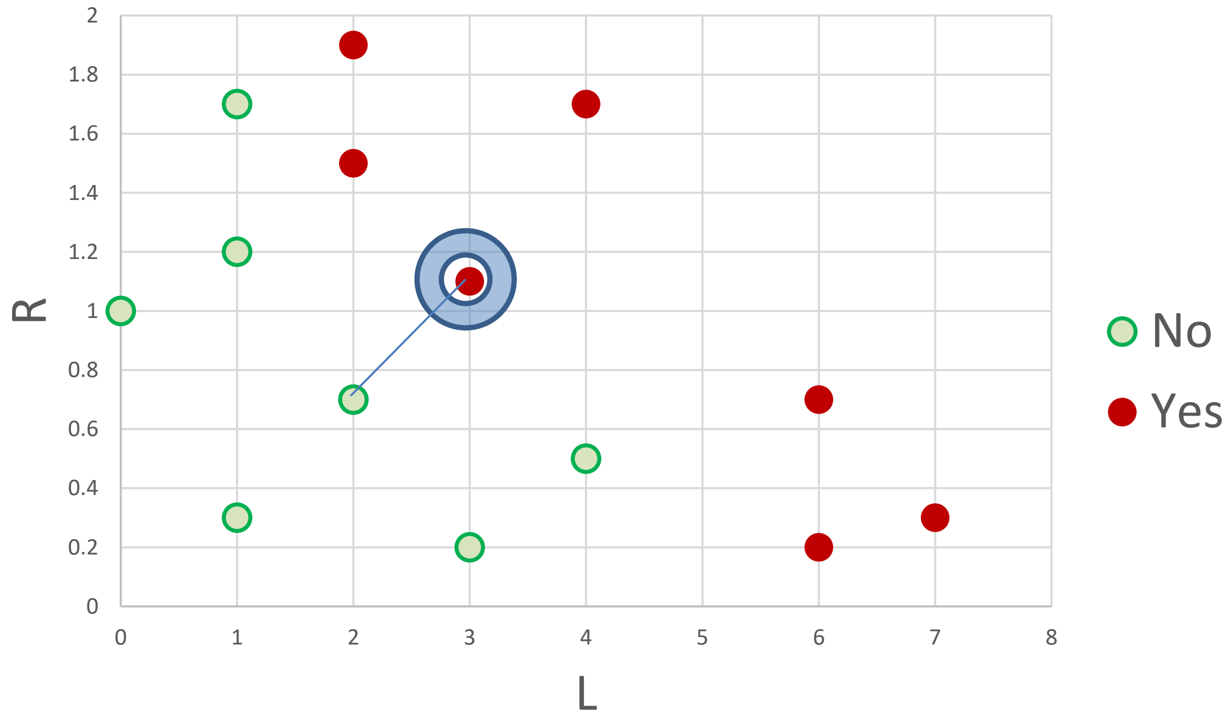


L: #late payments / year

R: expenses / income ratio

Leave-one-out cross validation: *K*=1

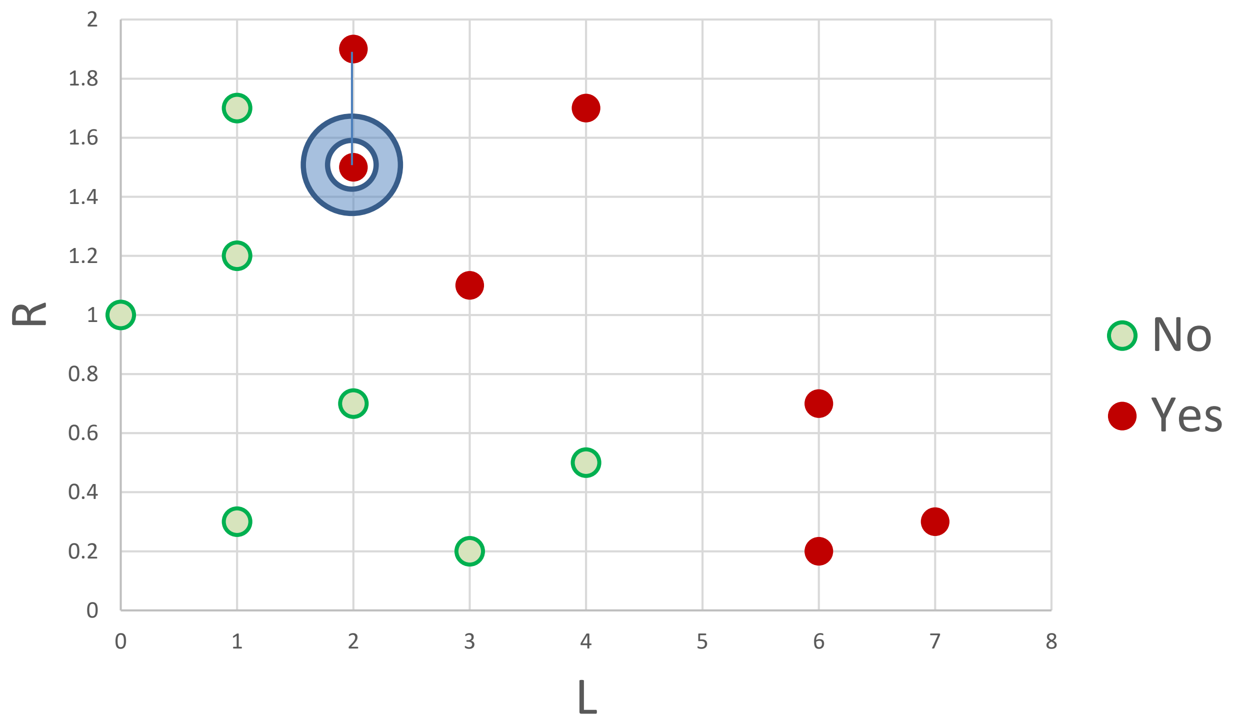
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=1

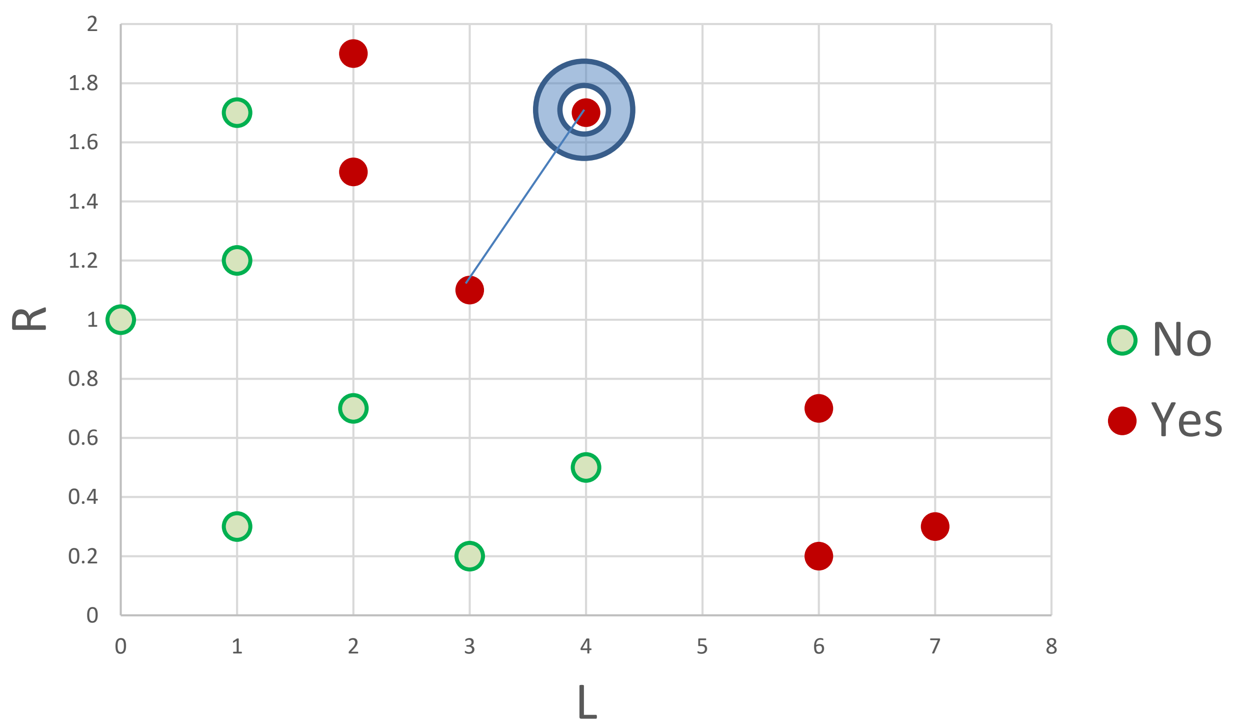
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=1

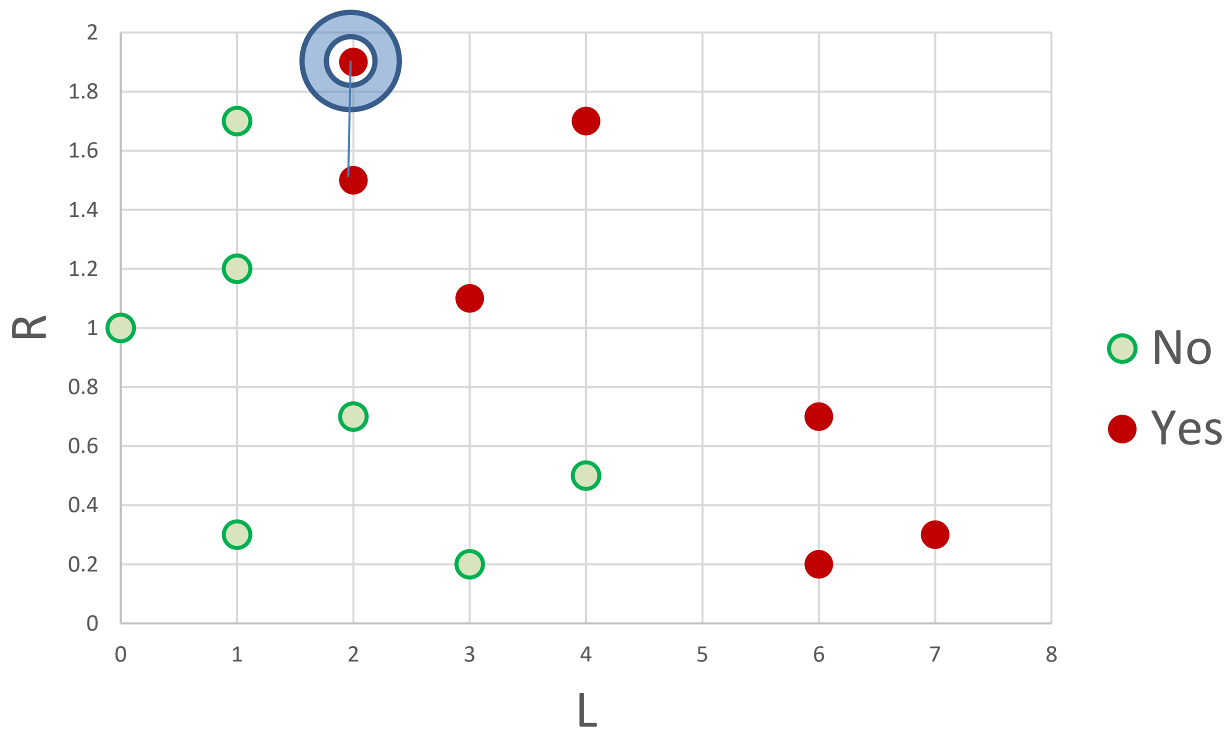
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: K=1

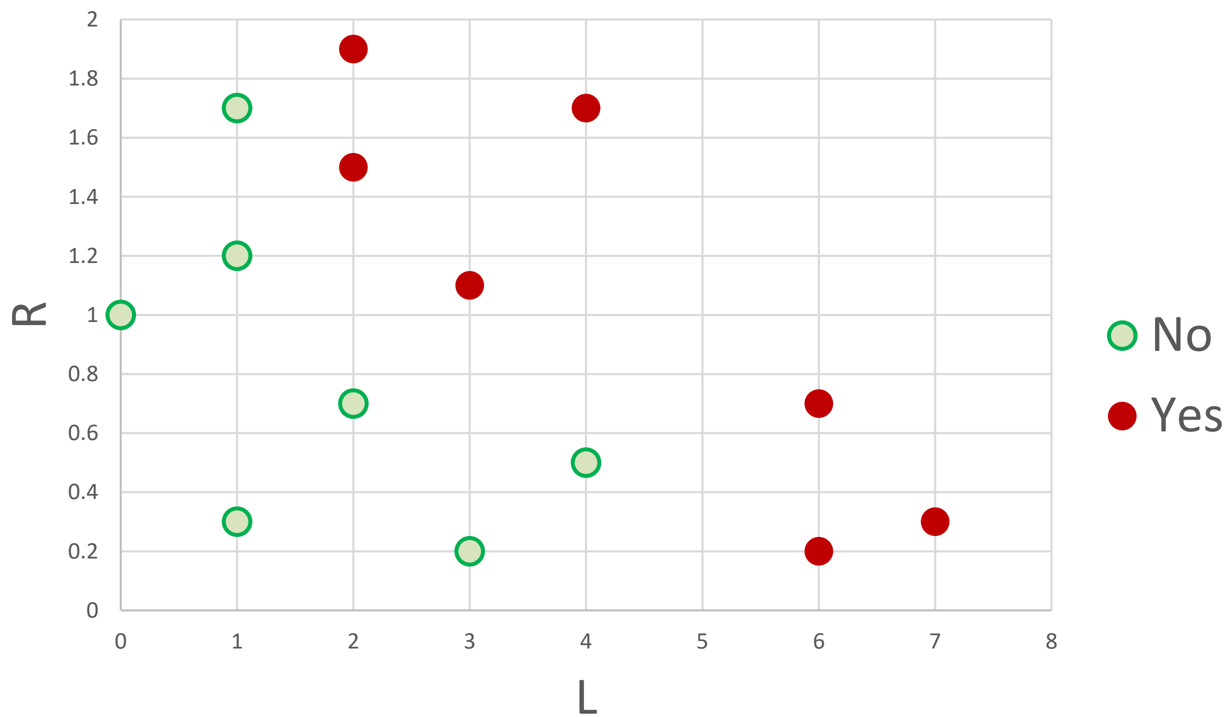
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=1

	L	R	B
	3	0.2	No
	1	0.3	No
	4	0.5	No
	2	0.7	No
	0	1	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

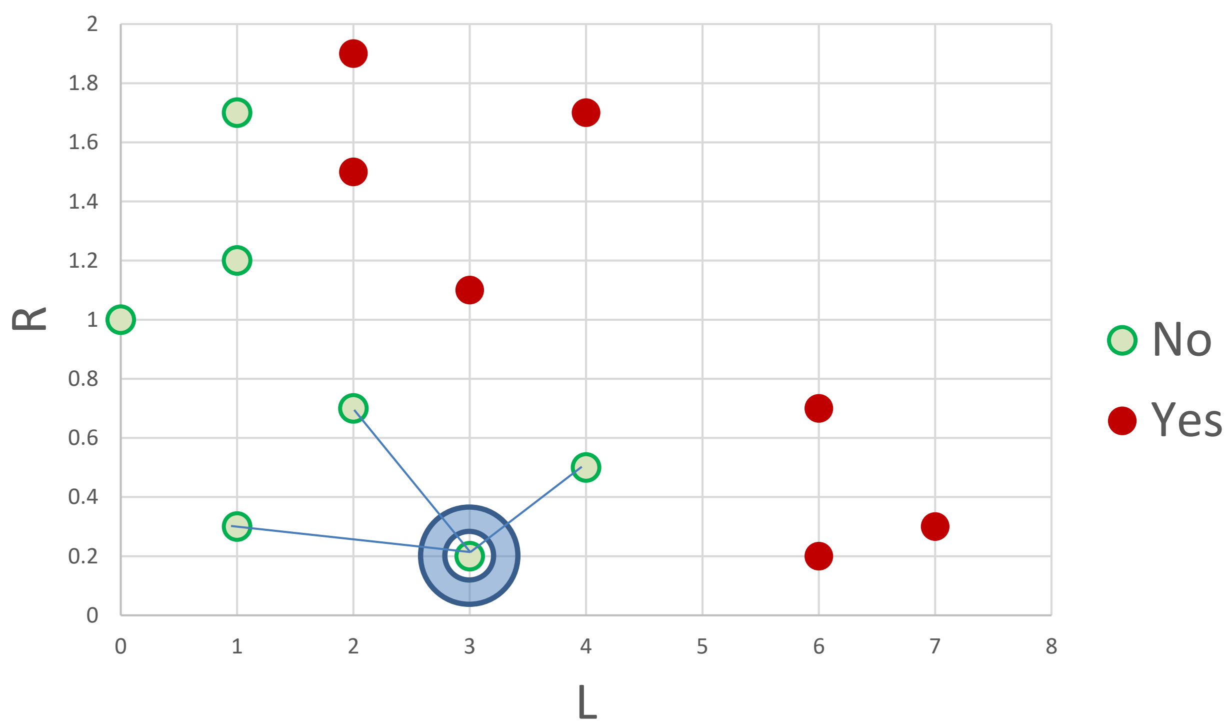


For *K*=1:

Error rate 3/14

Leave-one-out cross validation: *K*=3

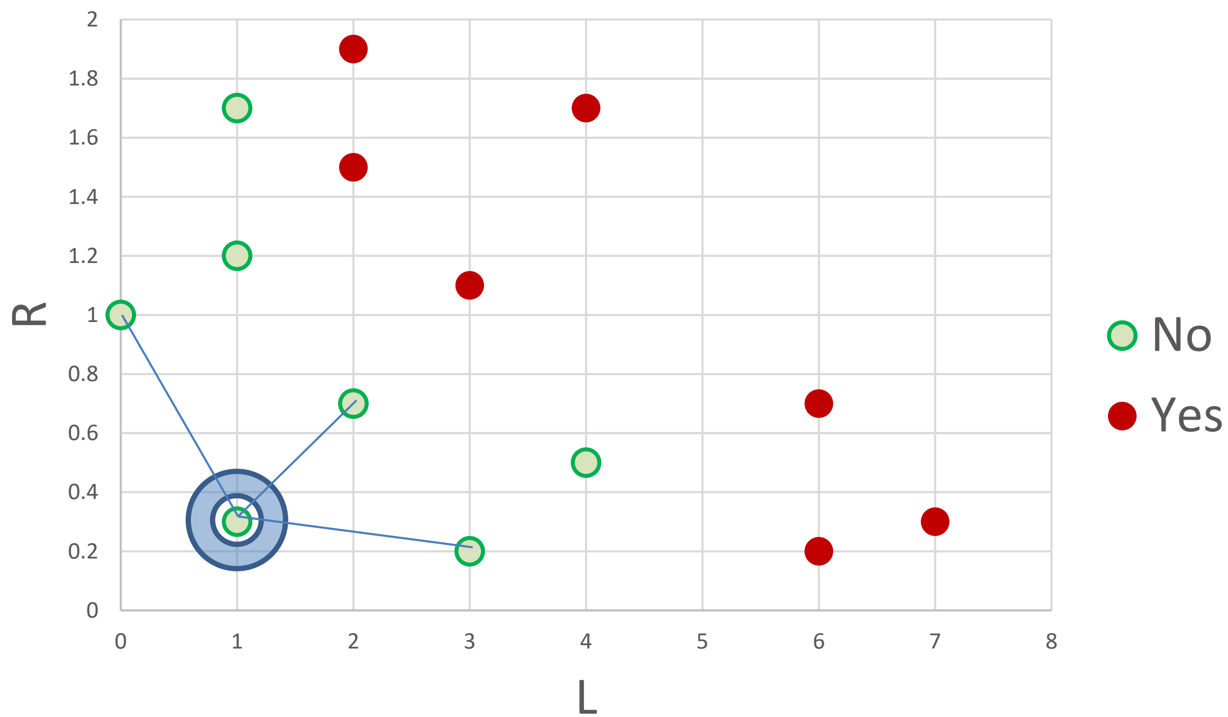
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=3

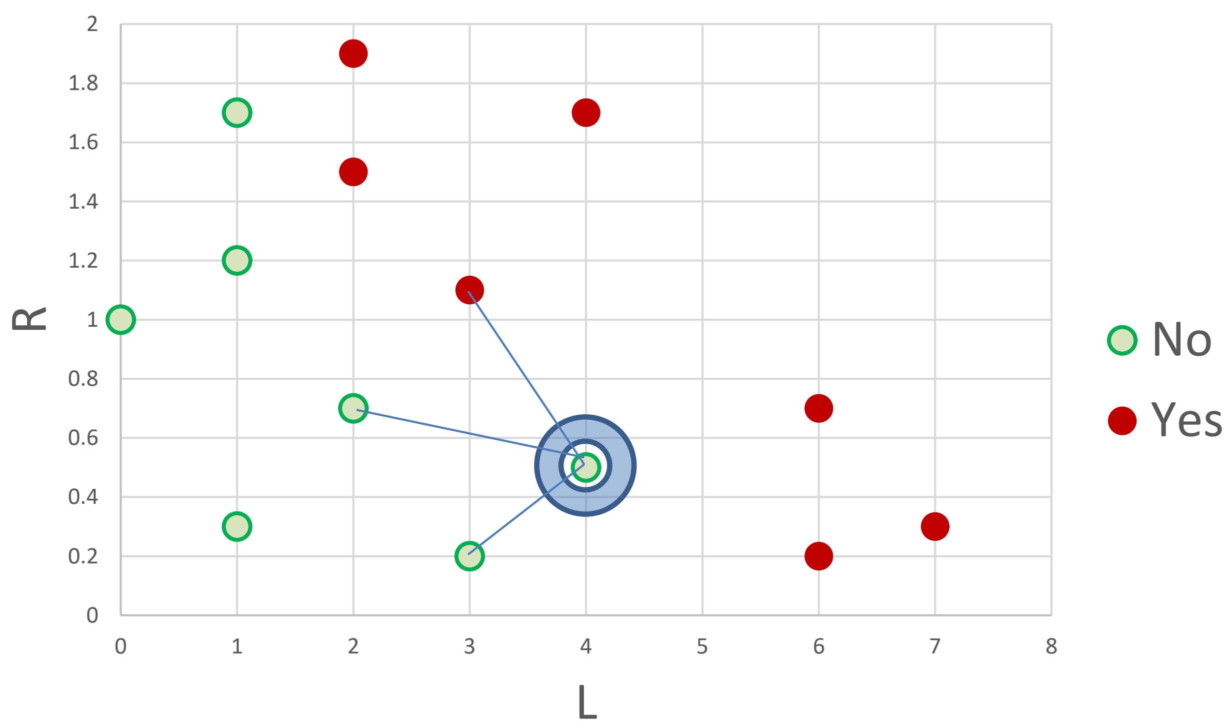
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=3

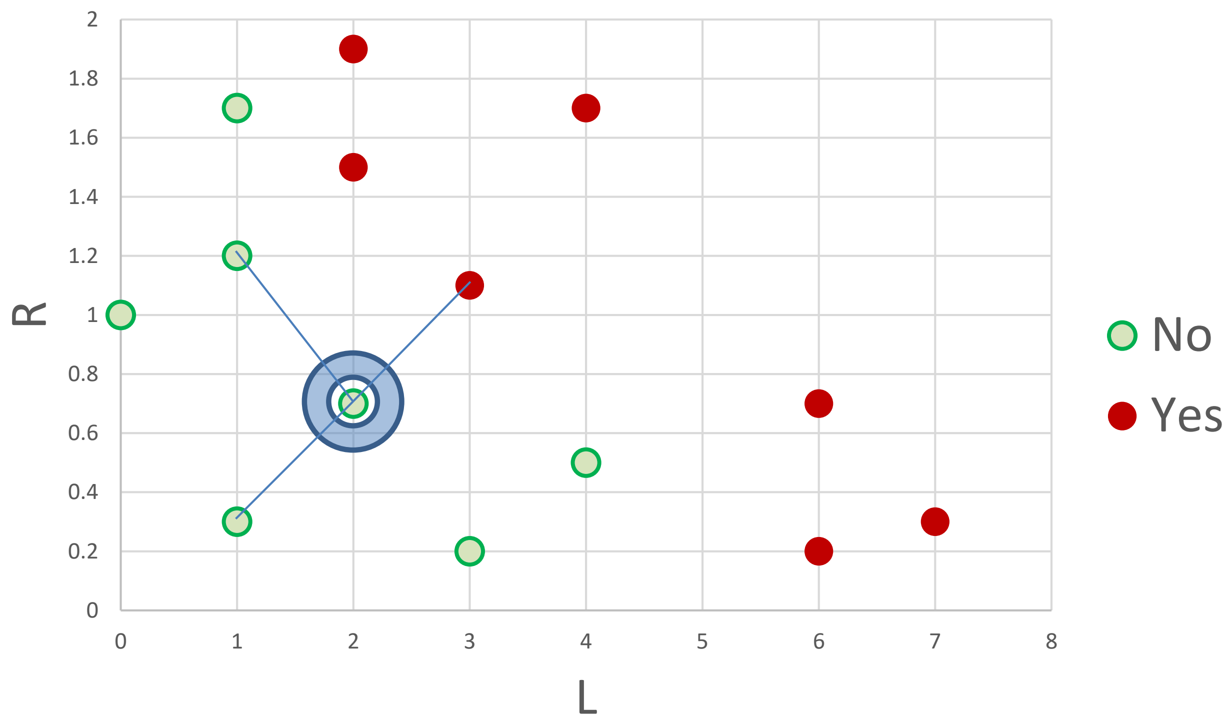
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=3

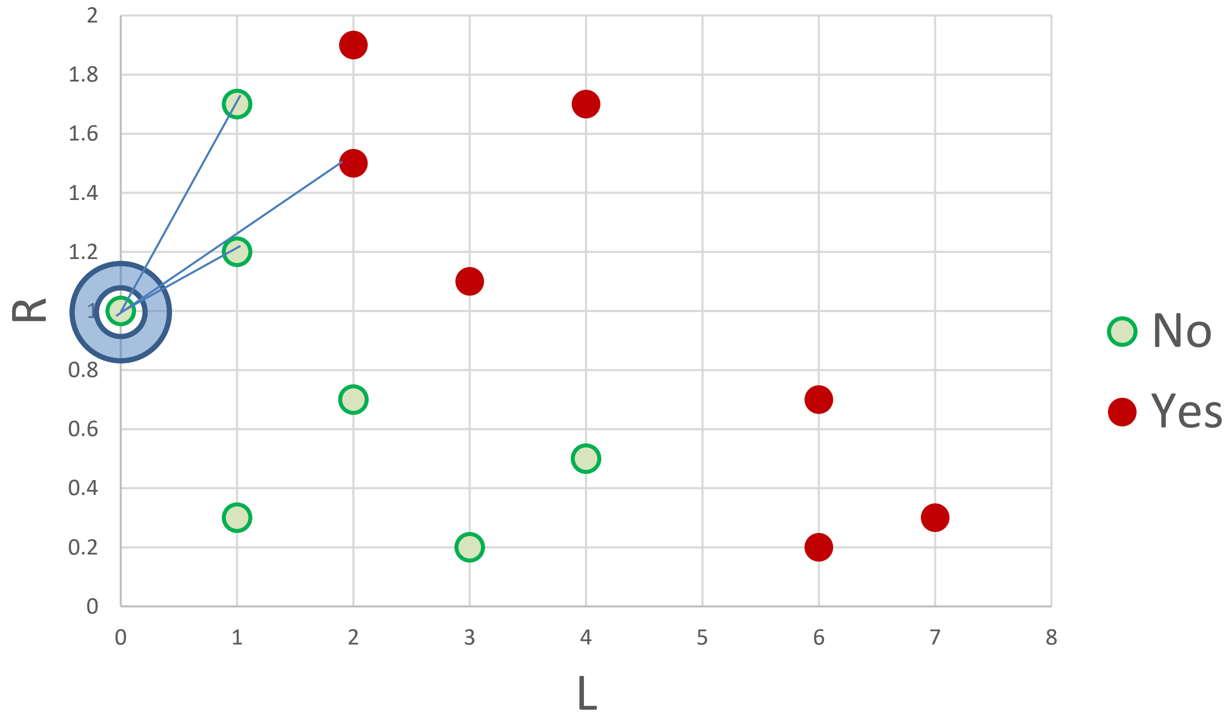
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=3

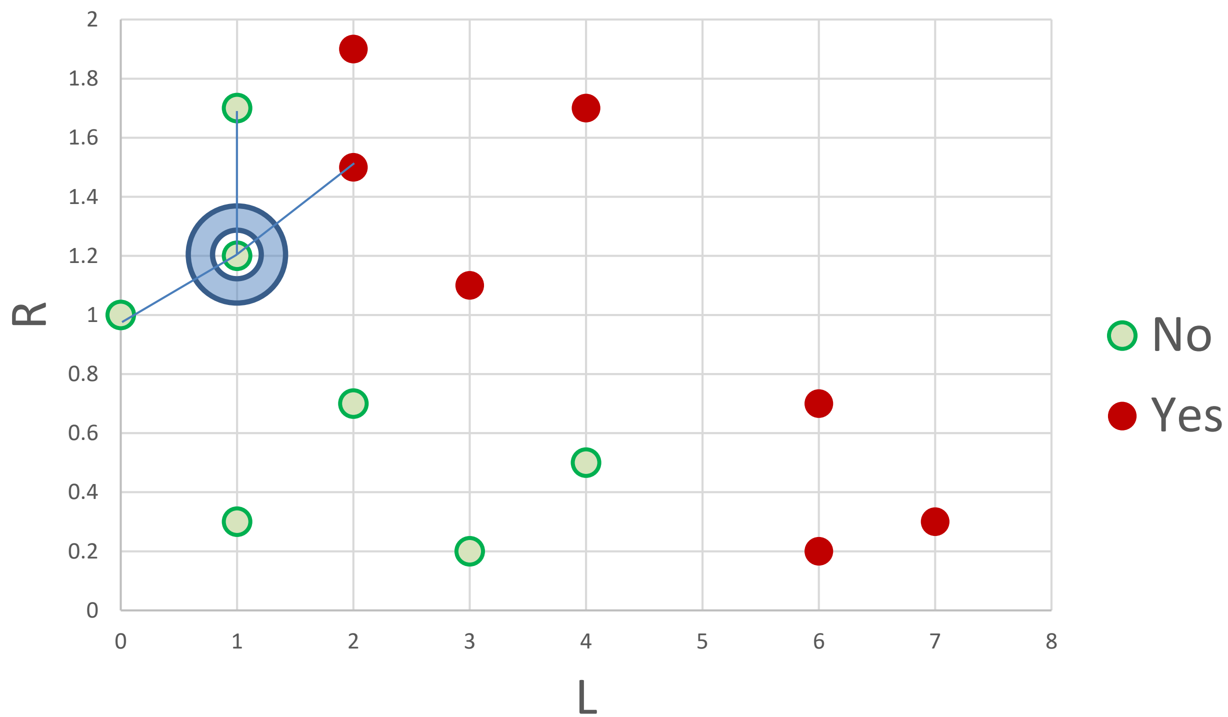
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=3

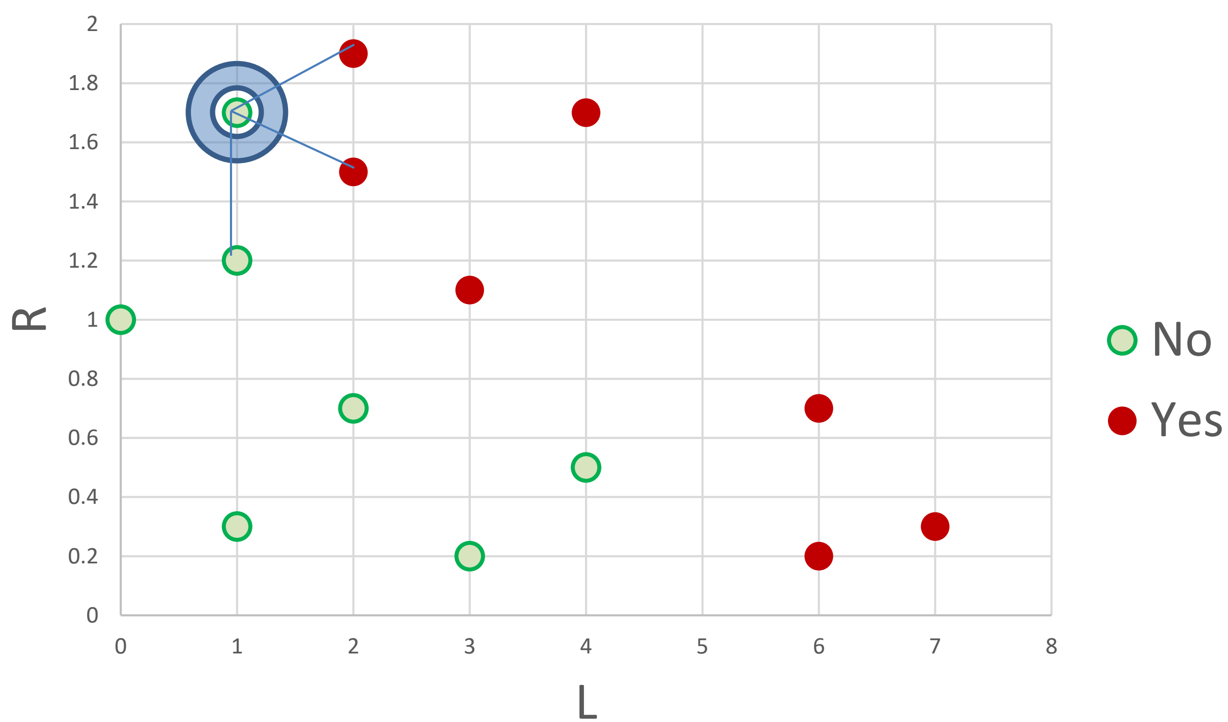
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=3

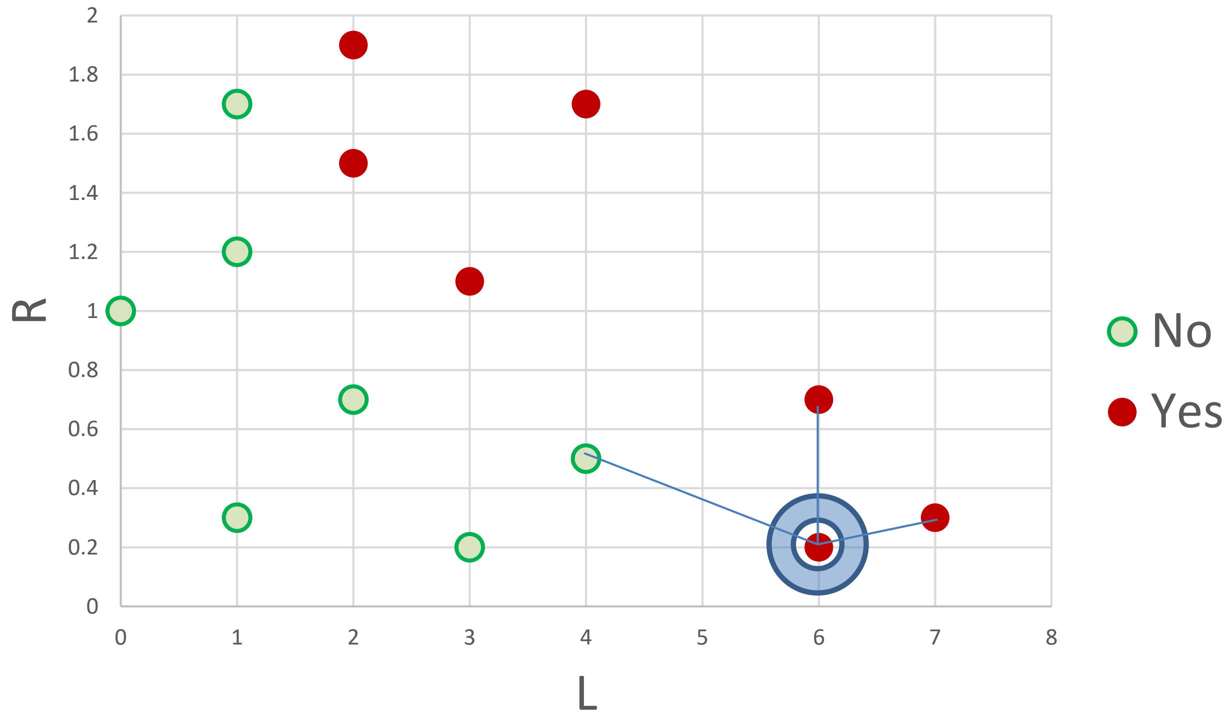
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=3

L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	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

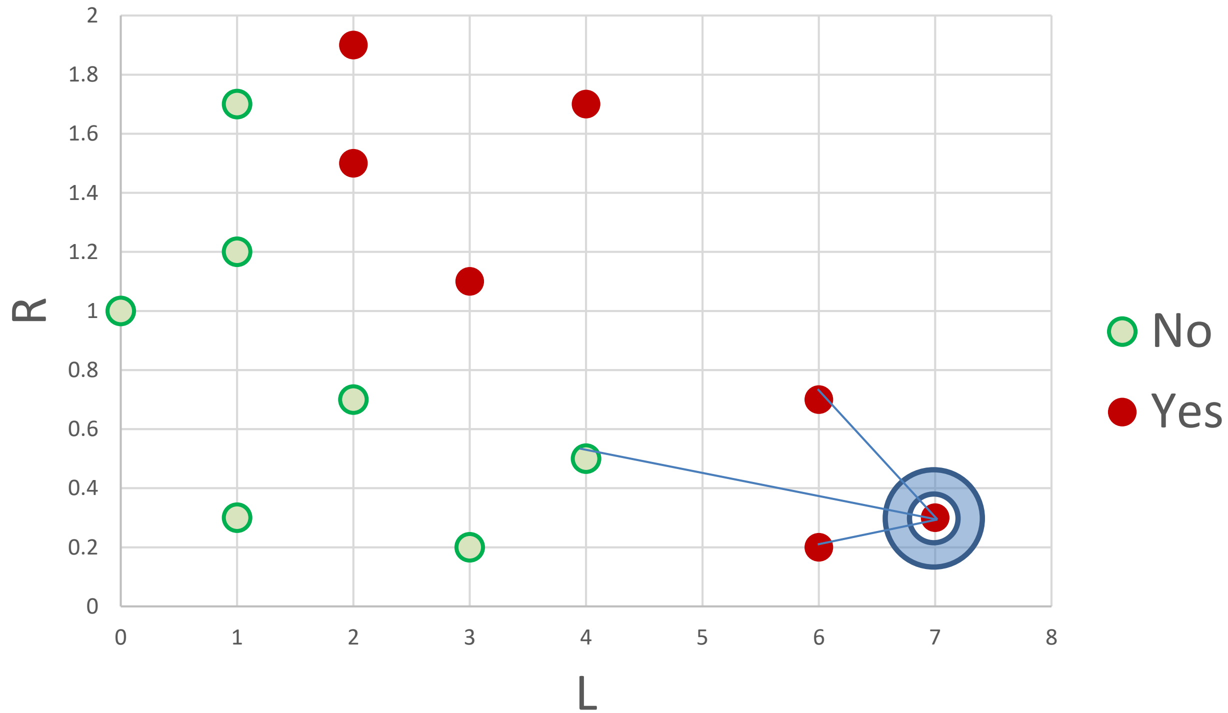


L: #late payments / year

R: expenses / income ratio

Leave-one-out cross validation: *K*=3

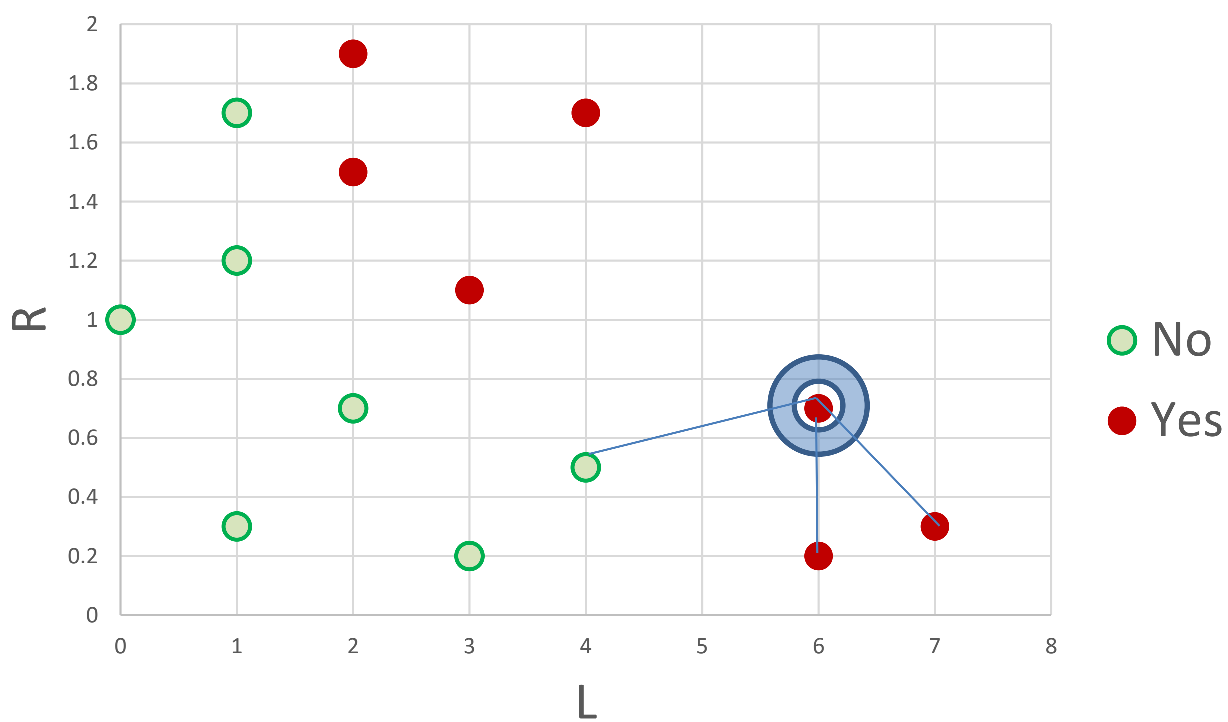
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=3

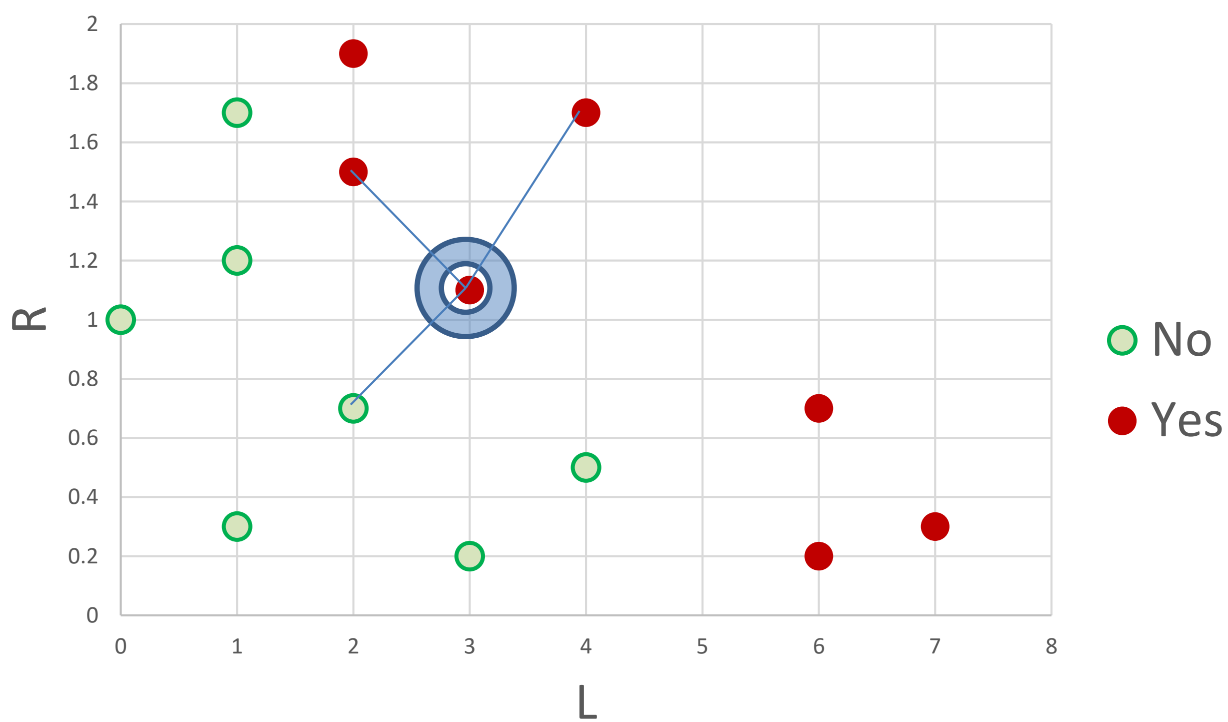
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=3

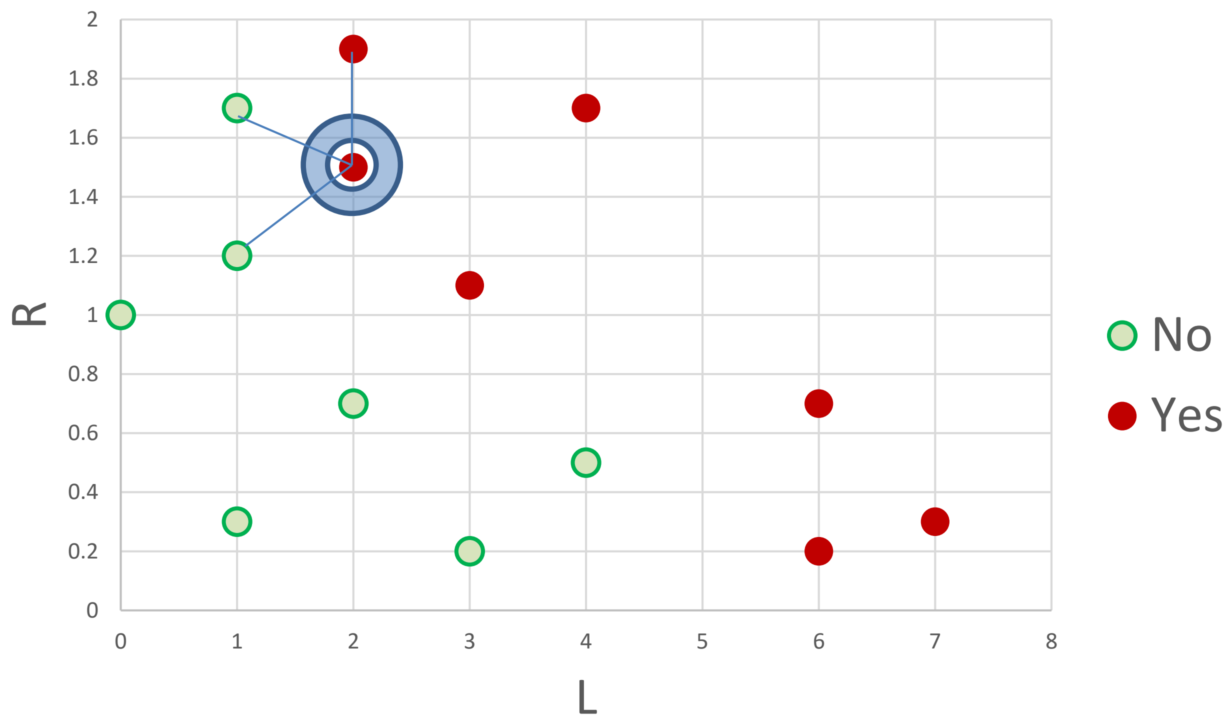
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: K=3

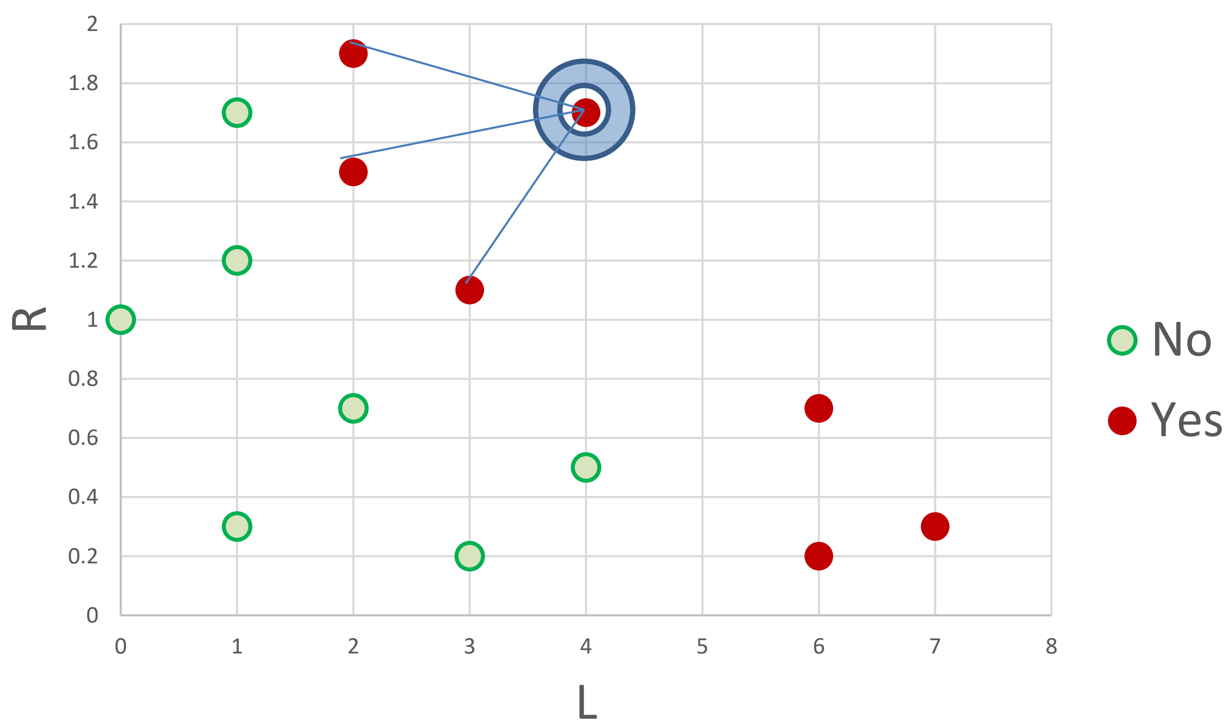
	L	R	B
	3	0.2	No
	1	0.3	No
	4	0.5	No
	2	0.7	No
	0	1	No
	1	1.2	No
⊗	1	1.7	No
	6	0.2	Yes
	7	0.3	Yes
	6	0.7	Yes
	3	1.1	Yes
⊗	2	1.5	Yes
	4	1.7	Yes
	2	1.9	Yes



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

Leave-one-out cross validation: *K*=3

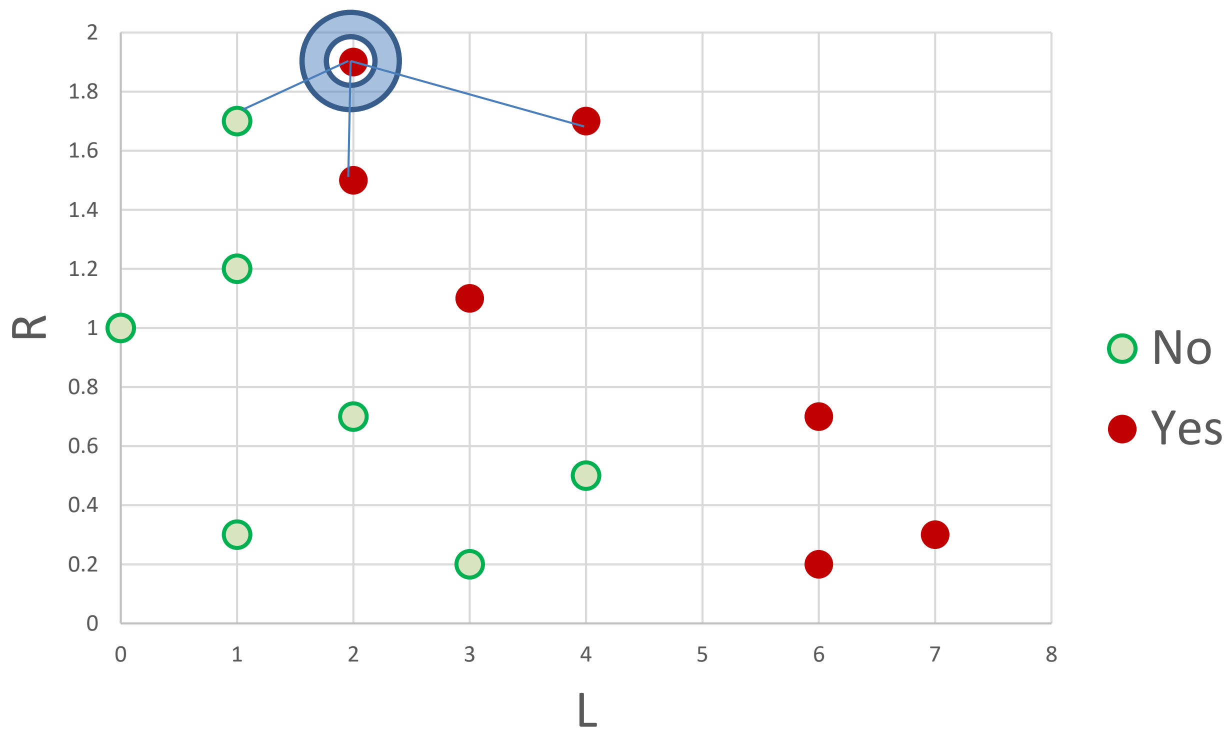
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=3

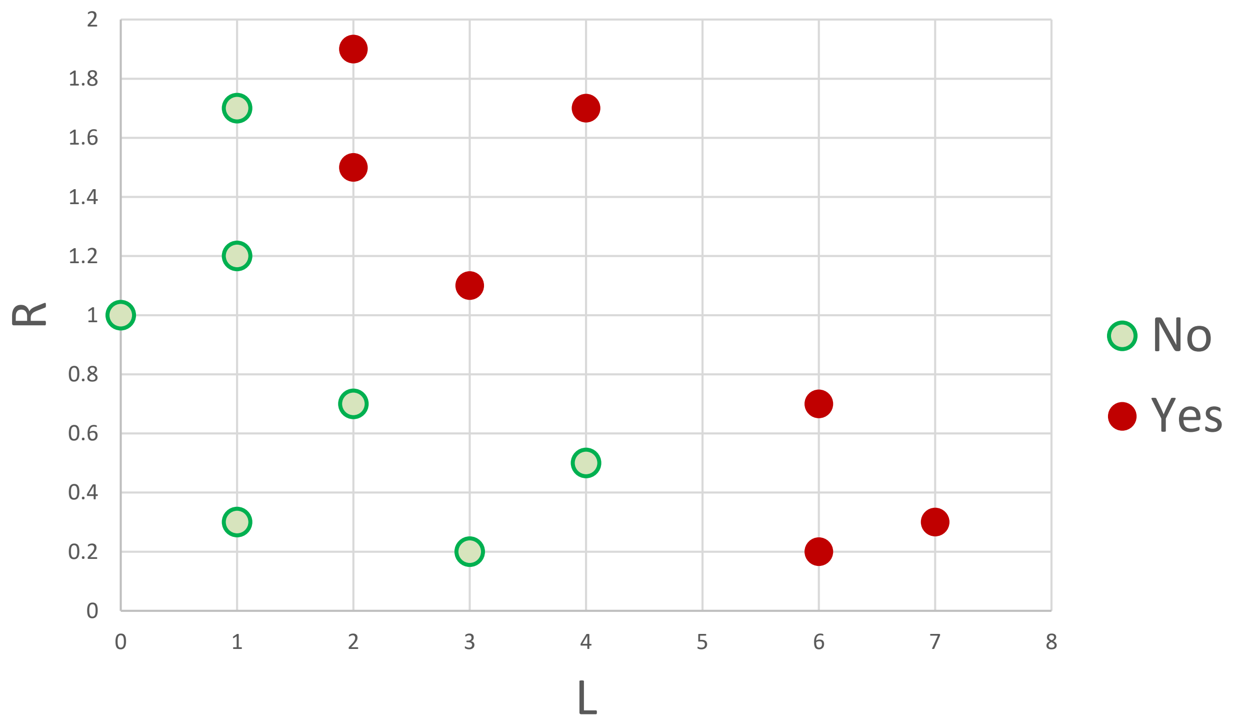
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: *K*=3

	L	R	B
	3	0.2	No
	1	0.3	No
	4	0.5	No
	2	0.7	No
	0	1	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

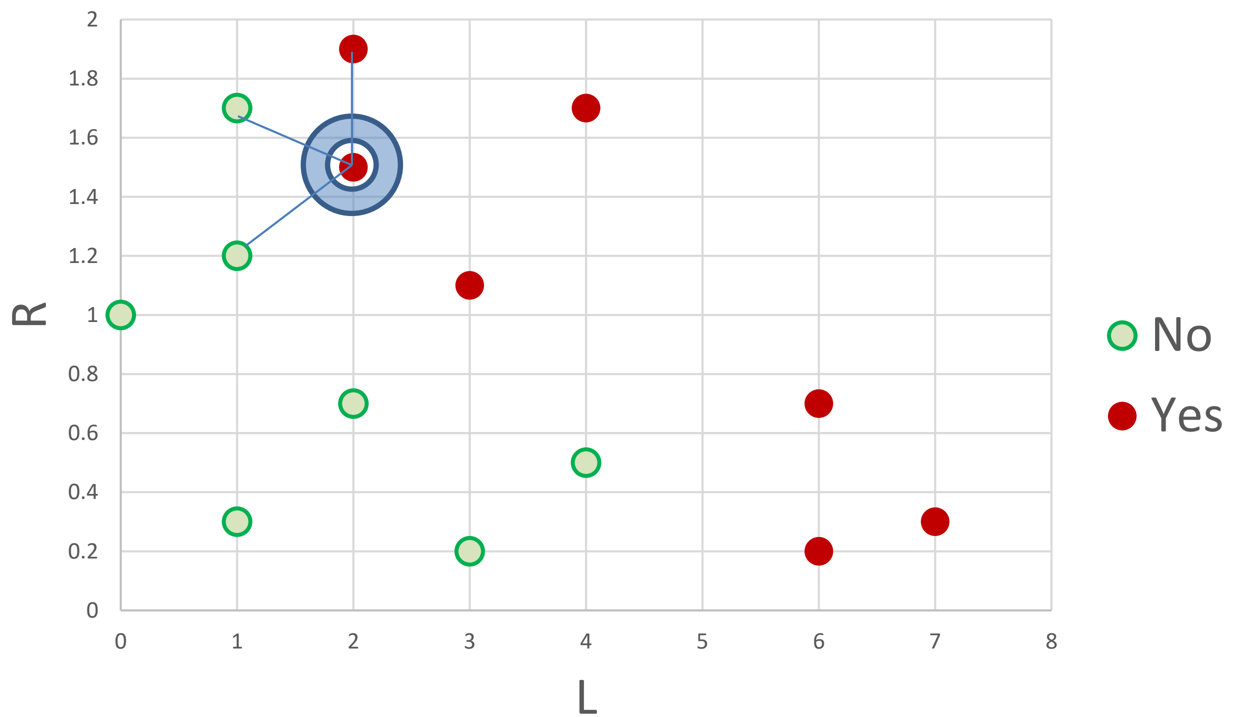


For ***K*=1**:
Error rate **3/14**

For ***K*=3**:
Error rate **2/14**

Leave-one-out cross validation: new error with *K*=3

L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	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

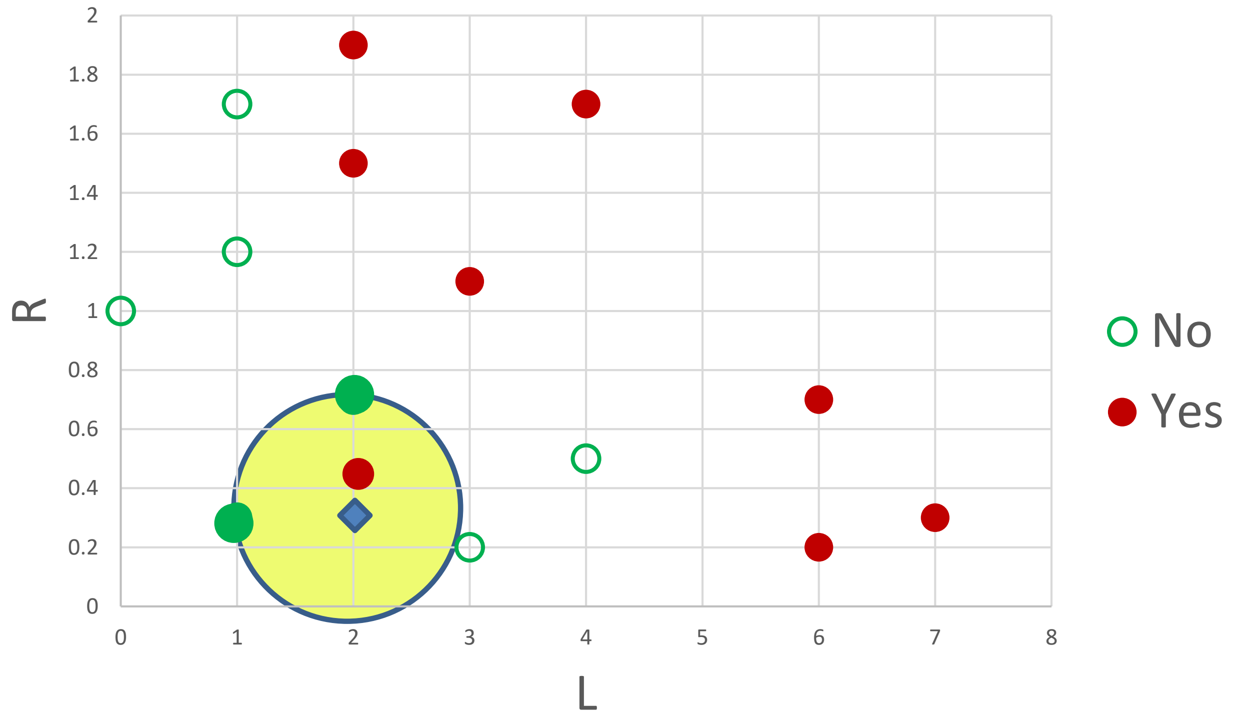


For *K*=1:
Error rate 3/14

For *K*=3:
Error rate 2/14

Majority voting (democracy)

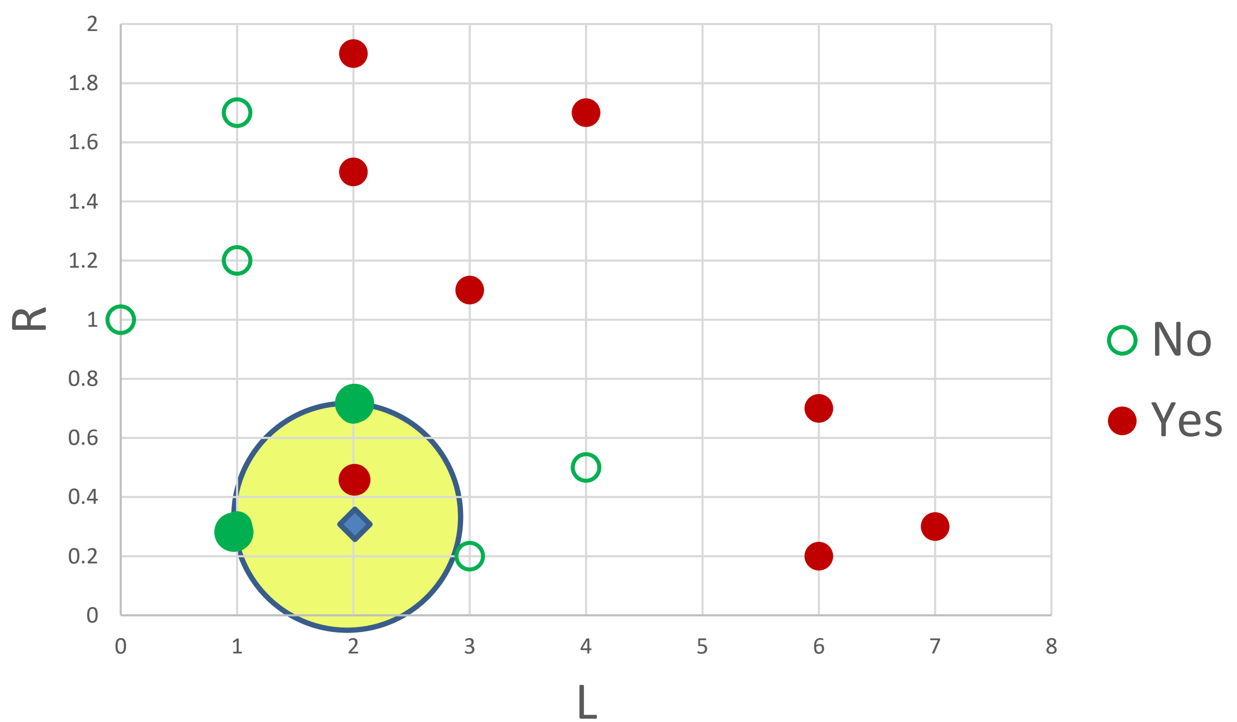
L	R
2	0.3



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

Weighted voting (shareholder democracy)

L	R
2	0.3



1/0.5 Yes+1/1.5 No + 1/1.5 No=2 Yes + 1.5 No = Yes

The closest neighbor outweighs the majority class

How many dimensions?

- Imagine you have one dimensional data which can be a straight line (the line where the floor and the wall meets) and plot 100 data points

- Now let's make this a 2D - a wall. Plot the same 100 points.

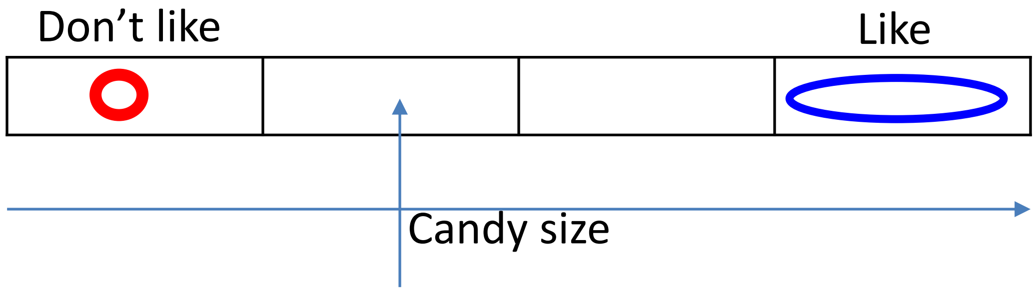
- Moving on, let's imagine a 3D which can be the room that has the wall in it. Again plot the 100 points.

- The points become more sparse as we move from a line to a wall and to a room. In a high dimensional space the same number of points are now separated by an exponentially large distance

- The prediction in sparse high-dimensional space will be less unreliable: The distance between points increases exponentially thus making predictions on sparse data becomes next to impossible.

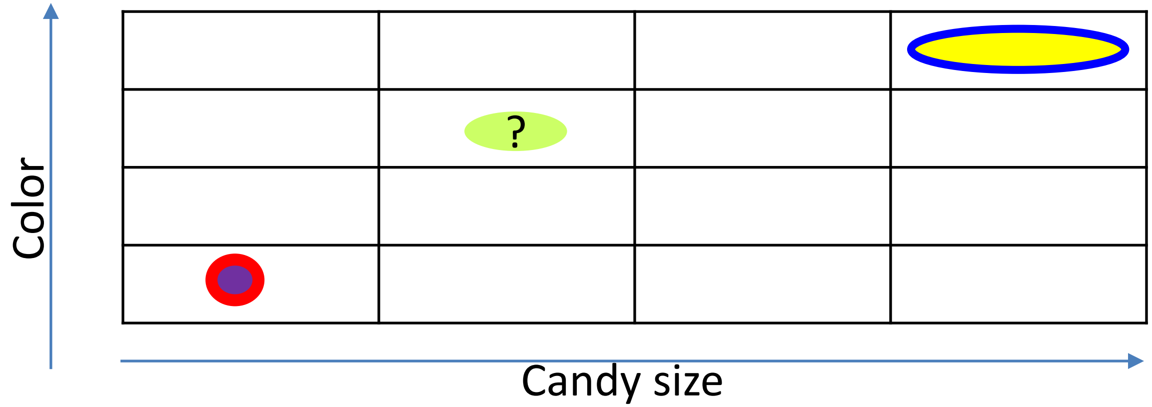
The curse of dimensionality: example

1 dimension



Predict: ? Prediction is always reliable

2 dimensions



Predict: ?

Prediction is not reliable:
need more data points

K-NN algorithm. Summary

- The training set ***is the*** model
- Advantages:
  - Building a classifier: zero work
  - Updating the model with every new record: zero work
  - Interpretable: we can justify our classification
  - Good in predicting numeric values
- Disadvantages:
  - The query is computationally expensive

A NEAREST NEIGHBOR APPROACH TO MAKING RECOMMENDATIONS

A nearest neighbor approach to making recommendations

Knowing that lots of people liked something is not enough. *Who* liked it is important.

?

Underworld: Awakening 3D 

Reviews - Trailer - IMDb

1hr 28min - Rated 18-A - Action/Adventure/Scifi/Fantasy/Horror - English
Director: Mans Marlind - Cast: Kate Beckinsale, Scott Speedman, India Eisley, Charles Dance, Michael Ealy - :

Selene escapes imprisonment to find herself in a world where humans have discovered the existence of both Vampire and Lycan clans, and are conducting an all-out war to eradicate both immortal species.

K-NN prediction

Item for which prediction is sought

Active user

	i_1	i_2	…	i_j	…	…	i_m
u_1							
u_2				5			
…							
u_a				?			
…							
…				5			
u_n							

K-NN prediction

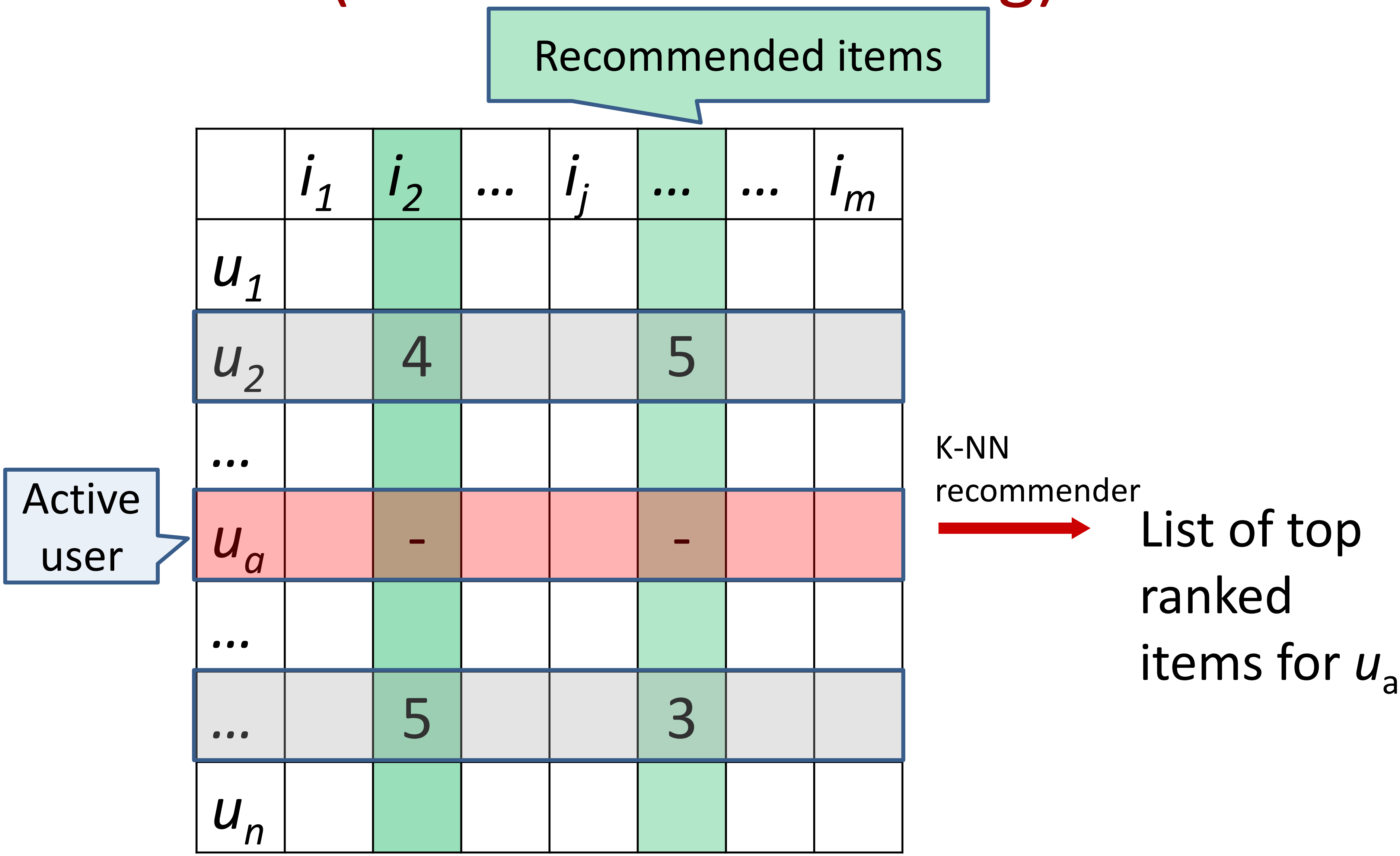
Item for which prediction is sought

	i_1	i_2	…	i_j	…	…	i_m
u_1							
u_2				5			
…							
u_a				?			
…							
…				5			
u_n							

Active user

$\xrightarrow{\text{K-NN}}$ predicted rank (i_j, u_a)

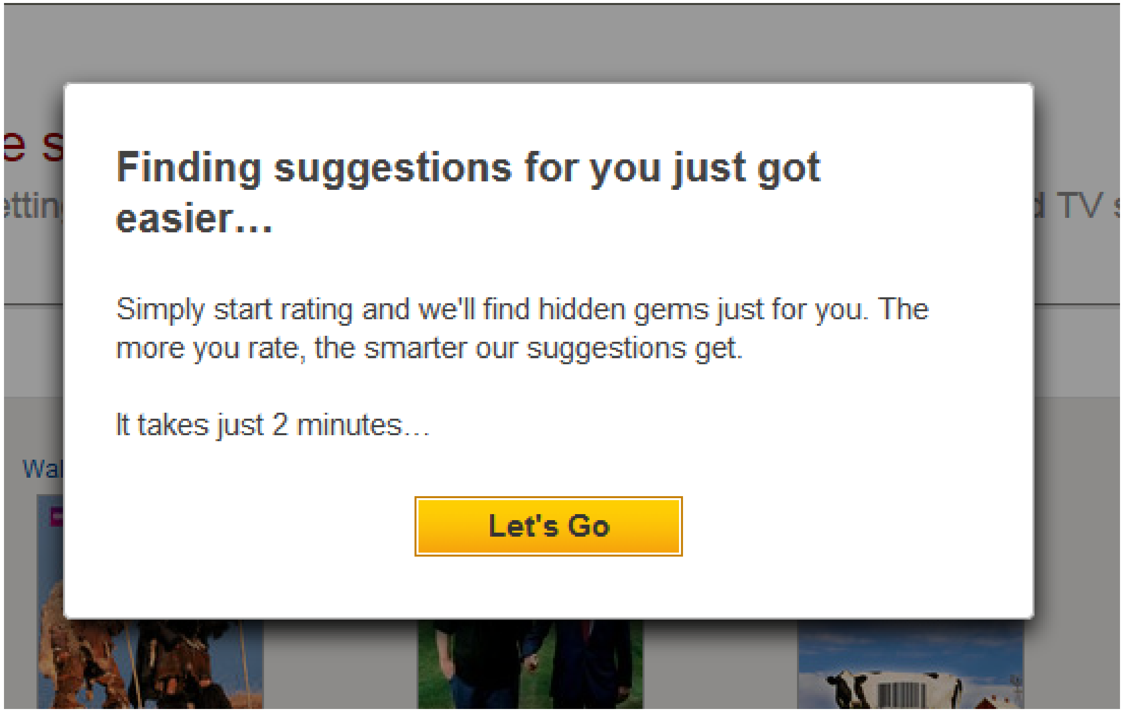
K-NN recommender (collaborative filtering)



Automated recommender system (collaborative filtering)

I. Build profile for active user
II. Compare active profile with the profiles of other customers, locate similar “neighbors”
III. Find and combine ratings of a peer group selected by similar tastes for the items that active user did not rank yet
IV. Rank predictions and output top-scored ones

Creating customer profile



Netflix.ca

Example: music recommender

You:

{Lady Gaga, Katy Perry, Justin Bieber, Maroon 5}

II. Finding users with similar tastes

1. April:
   {The Beatles, Lady Antebellum}

2. Ben:
   {Lady Gaga, Adele, Kelly Clarkson, The Dixie Chicks, Lady Antebellum}

3. Cory:
   {Kelly Clarkson, Lady Gaga, Katy Perry, Justin Bieber, Lady Antebellum}

4. Dave:
   {The Beatles, Maroon 5, Lady Antebellum}

5. Edgar:
   {Adele, Maroon 5, Katy Perry, Bruno Mars}

II. Finding users with similar tastes

	Adele	The Beatles	Justin Bieber	The Dixie Chicks	Kelly Clarkson	Lady Gaga	Lady Antebellum	Maroon 5	Bruno Mars	Katy Perry
April		Yes					Yes			
Ben	Yes			Yes	Yes	Yes	Yes			
Cory			Yes		Yes	Yes	Yes			Yes
Dave		Yes					Yes	Yes		
Edgar	Yes							Yes	Yes	Yes
You			**Yes**			**Yes**		**Yes**		**Yes**

Similarity measures for asymmetric binary data

Simple matching coefficient:

number of matches / total attributes

Jaccard index:

number of matches/total not-both-null attributes

II. Finding users with similar tastes (Jaccard index)

	Adele	The Beatles	Justin Bieber	The Dixie Chicks	Kelly Clarkson	Lady Gaga	Lady Antebellum	Maroon 5	Bruno Mars	Katy Perry
April		Yes					Yes			
Ben	Yes			Yes	Yes	Yes	Yes			
Cory			Yes		Yes	Yes	Yes			Yes
Dave		Yes					Yes	Yes		
Edgar	Yes							Yes	Yes	Yes

You			Yes			Yes		Yes		Yes

You vs April: 0/6
You vs Ben: 1/8
You vs. Cory: 3/6
You vs. Dave: 1/6
You vs. Edgar: 2/6

Your peer group:
Cory: similarity 0.50
Dave: similarity 0.17
Edgar: similarity 0.33

III. Combine ratings for new items (weighted voting)

	Adele	The Beatles	Justin Bieber	The Dixie Chicks	Kelly Clarkson	Lady Gaga	Lady Antebel lum	Maroon 5	Bruno Mars	Katy Perry
April		Yes					Yes			
Ben	Yes			Yes	Yes	Yes	Yes			
Cory			Yes		Yes	Yes	Yes			Yes
Dave		Yes					Yes	Yes		
Edgar	Yes							Yes	Yes	Yes

You			**Yes**			**Yes**		**Yes**		**Yes**

Your peer group:

Cory: similarity 0.50

Dave: similarity 0.17

Edgar: similarity 0.33

Predicted likes for new items:

Adele: 1 like *0.33 = 0.33

The Beatles: 1 like * 0.17 = 0.17

Kelly Clarkson: 1 like * 0.50 = 0.50

Lady Antebellum: 1 like *0.50 + 1 like * 0.17 = 0.67

Bruno Mars: 1 like * 0.33 = 0.33

IV. Output top-ranked

	Adele	The Beatles	Justin Bieber	The Dixie Chicks	Kelly Clarkson	Lady Gaga	Lady Antebel lum	Maroon 5	Bruno Mars	Katy Perry
April		Yes					Yes			
Ben	Yes			Yes	Yes	Yes	Yes			
Cory			Yes		Yes	Yes	Yes			Yes
Dave		Yes						Yes		
Edgar	Yes							Yes	Yes	Yes

You			**Yes**			**Yes**		**Yes**		**Yes**

Your peer group:

Cory: similarity 0.50

Dave: similarity 0.17

Edgar: similarity 0.33

Top ranked:

Lady Antebellum: 0.67

Kelly Clarkson: 0.50

These are your recommendations !

Project idea. Course recommender

- Provided:
  - Training set
  - Fully-labeled test set
- Task:
  - Develop recommender system: optimal K, optimal distance, optimal combination function
- Evaluation:
  - Recommend 3 courses for each user in a test set
  - Compare with actual rankings in a fully-labeled test set. The prediction is considered incorrect if it differs from the actual rank by more than 0.5

To improve recommender

- Choice of distance/similarity
- Weighted voting
- Optimal number of neighbors *k*
- Domain knowledge …

K-NN performance. Time and space

- Space: $O(N)$.

- Running time: $O(M \times N)$.

- M – number of attributes
- N – total instances in the training set

K-NN performance improvements (heuristics)

1. IB2: save memory, speed up classification
2. IB3: deal with noise

---

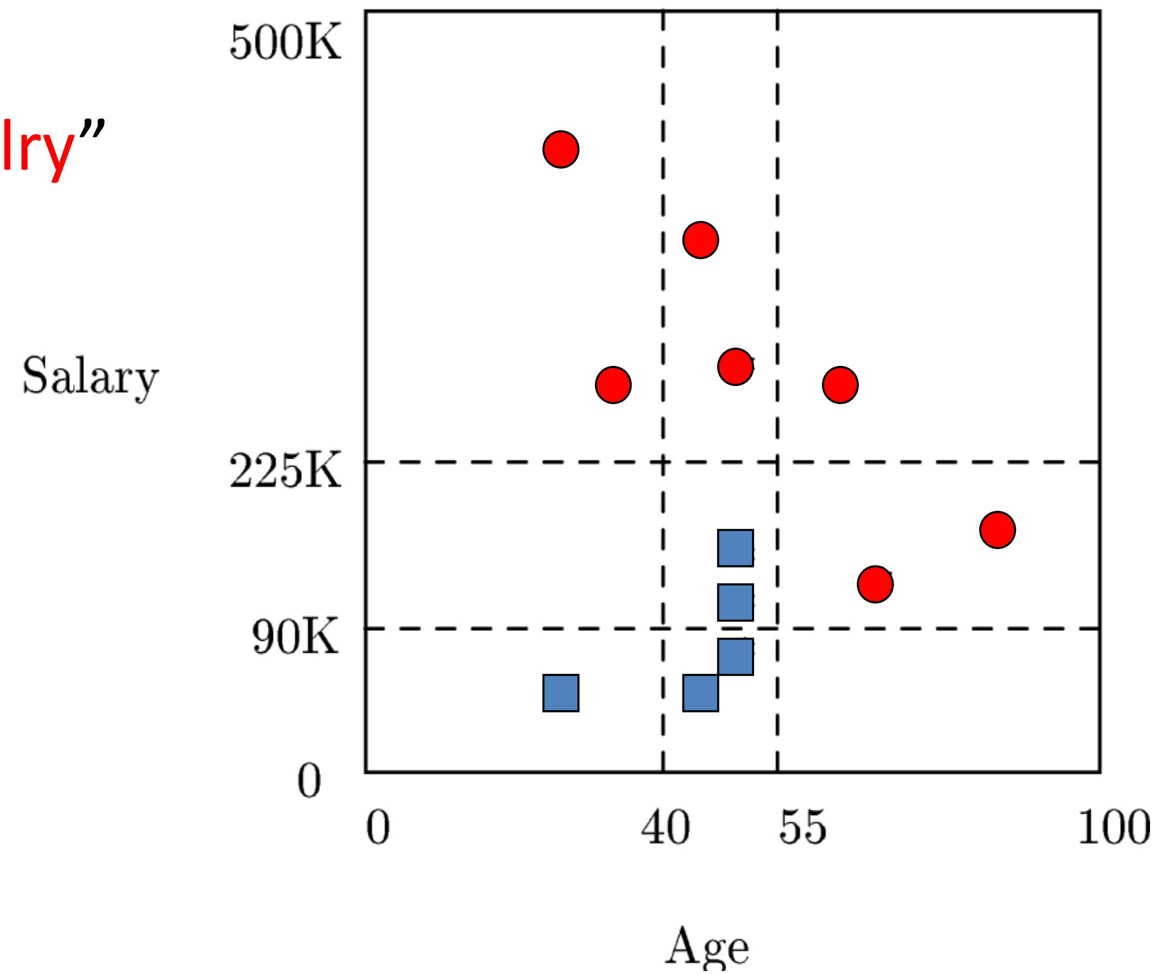
*) IB – Instance Based

IB2 main idea

- Work incrementally
- Only incorporate misclassified instances

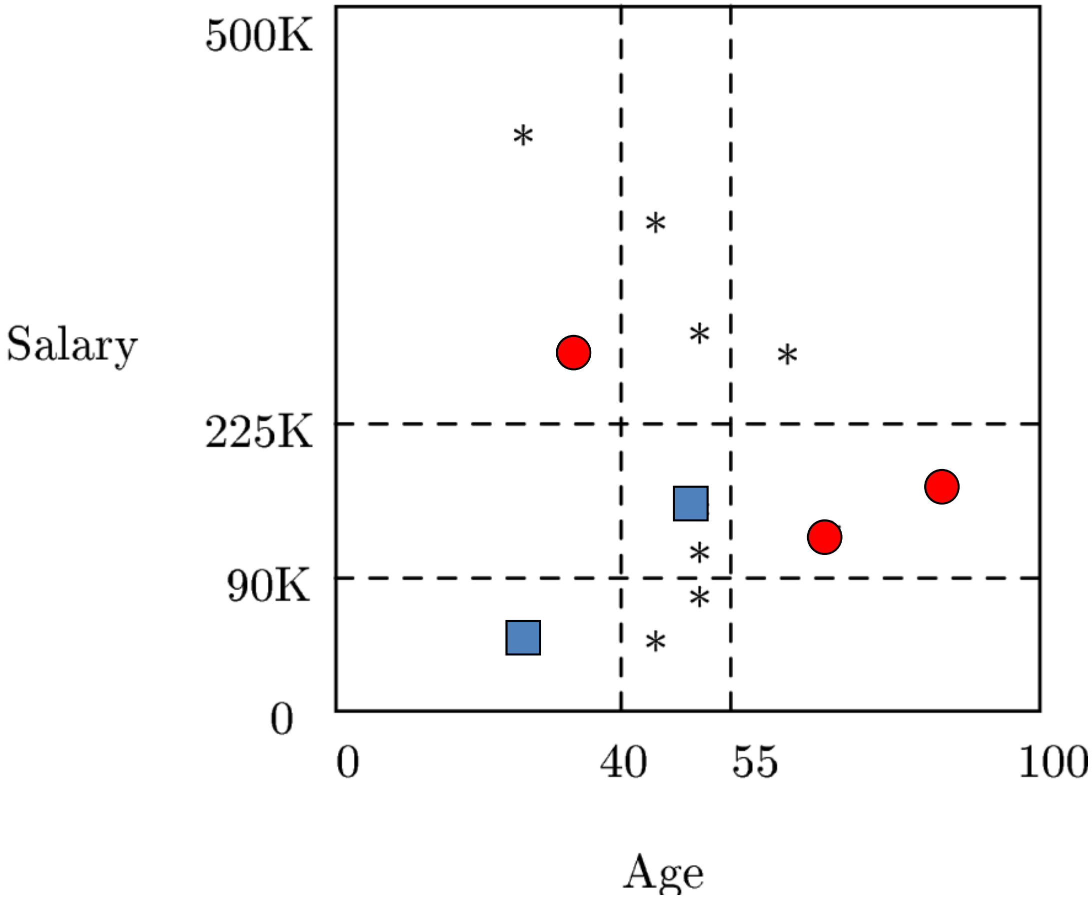
Example: IB2

Dataset:
"Who buys gold jewelry"



IB2 example

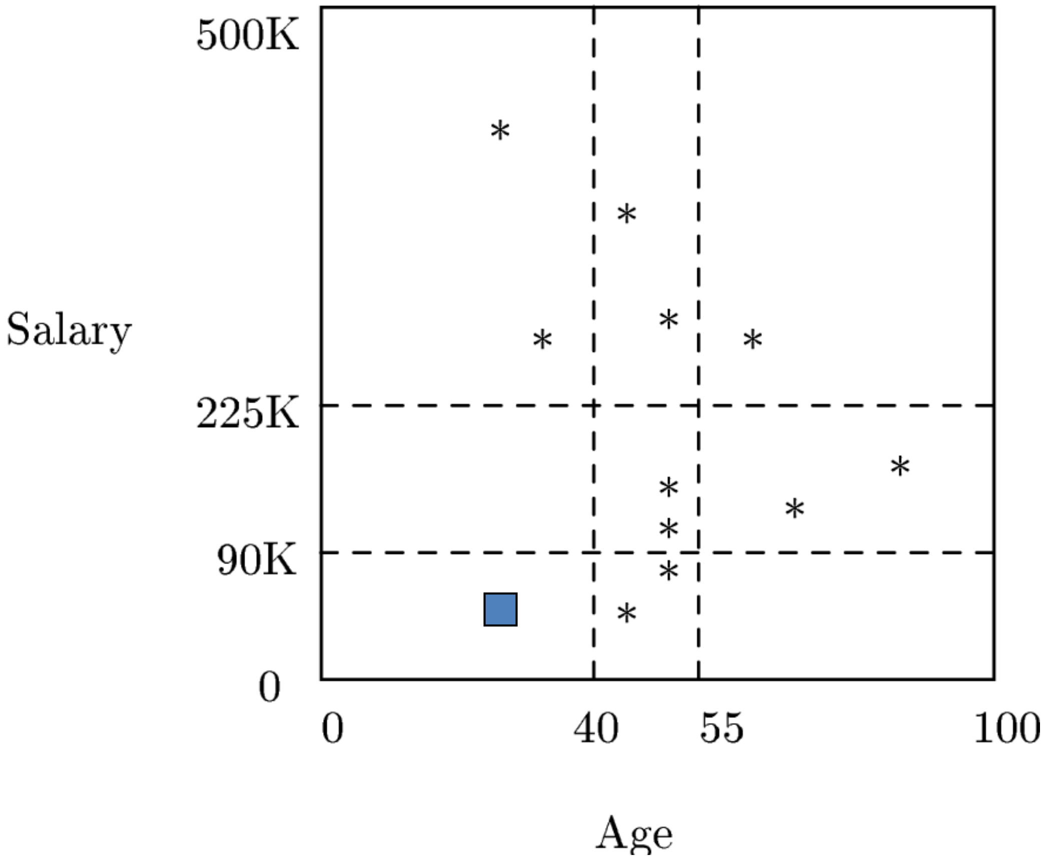
- Data:
  - (25,60,no)
  - (85,140,yes)
  - (45,60,no)
  - (30,260,yes)
  - (50,75,no)
  - (50,120,no)
  - (70,110,yes)
  - (25,400,yes)
  - (50,100,no)
  - (45,350,yes)
  - (50,275,yes)
  - (60,260,yes)



IB2 output: We memorize only these 5 points.

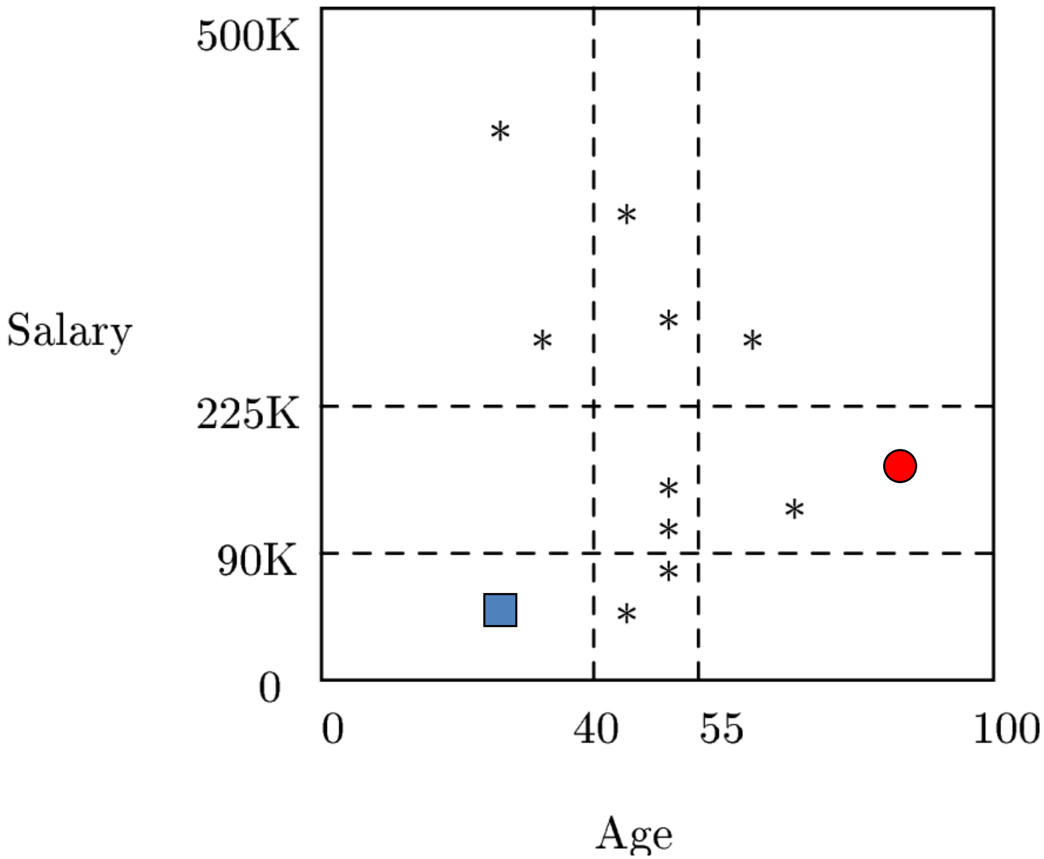
IB2 example

- Data:
  - (25,60,no)



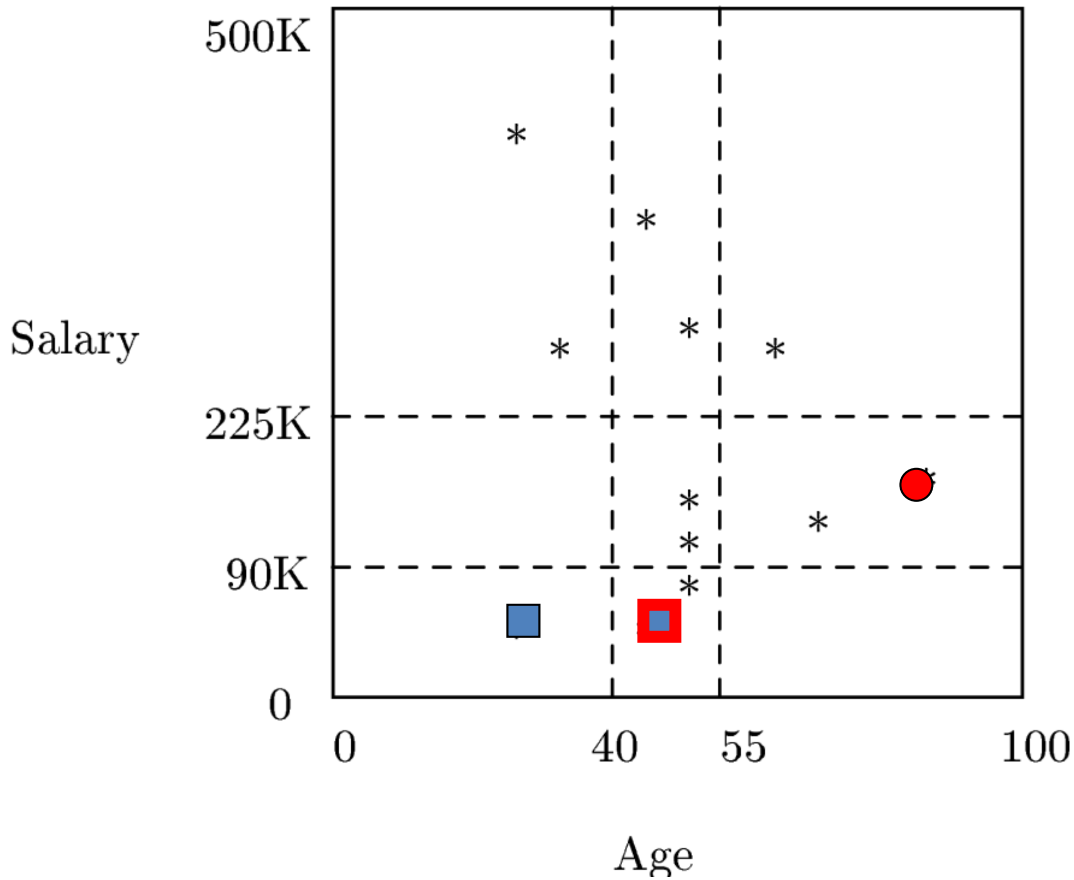
IB2 example

- Data:
  - (25,60,no)
  - *(85,140,yes)*



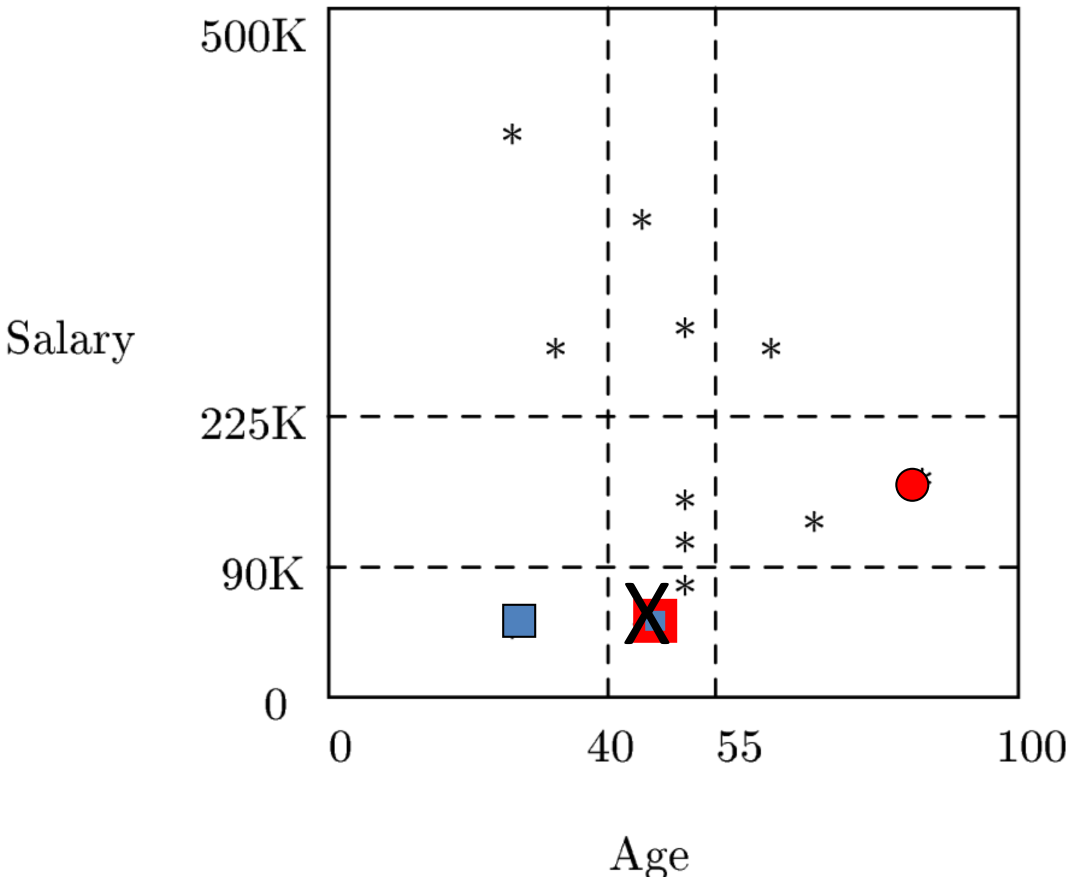
IB2 example

- Data:
  - (25,60,no)
  - (85,140,yes)
  - *(45,60,no)*



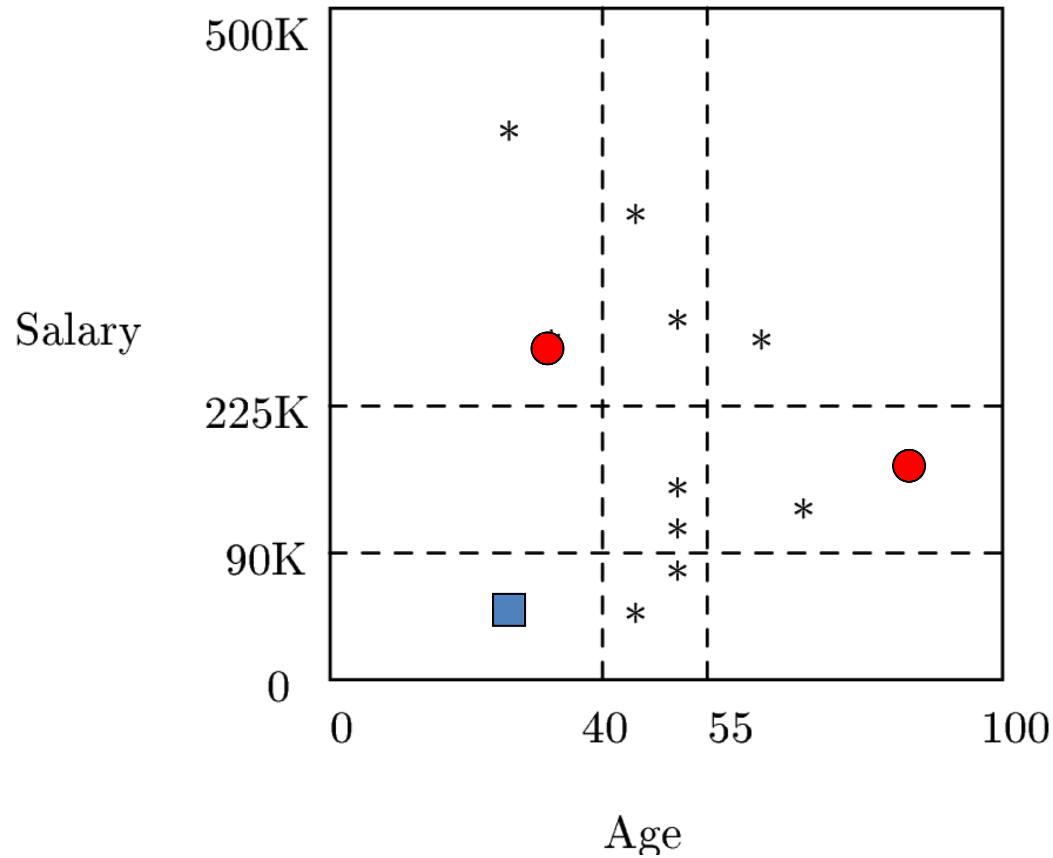
IB2 example

- Data:
  - (25,60,no)
  - (85,140,yes)
  - *(45,60,no)*



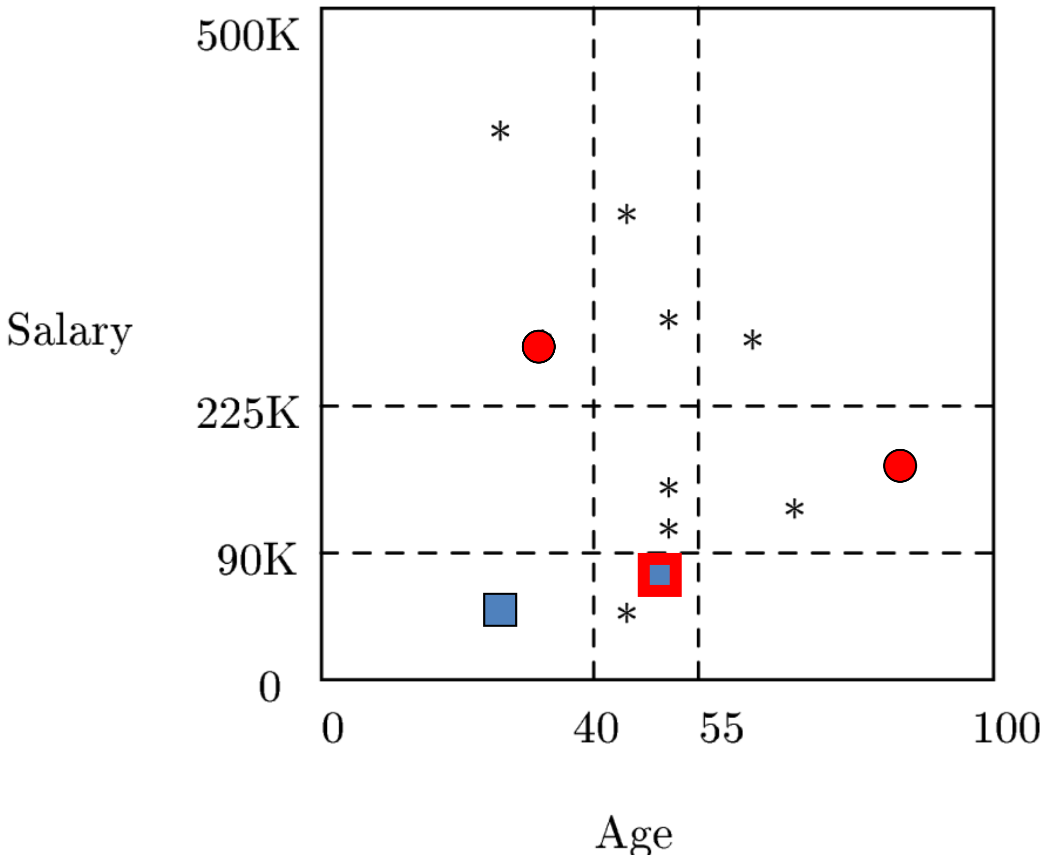
IB2 example

- Data:
  - (25,60,no)
  - (85,140,yes)
  - (45,60,no)
  - *(30,260,yes)*



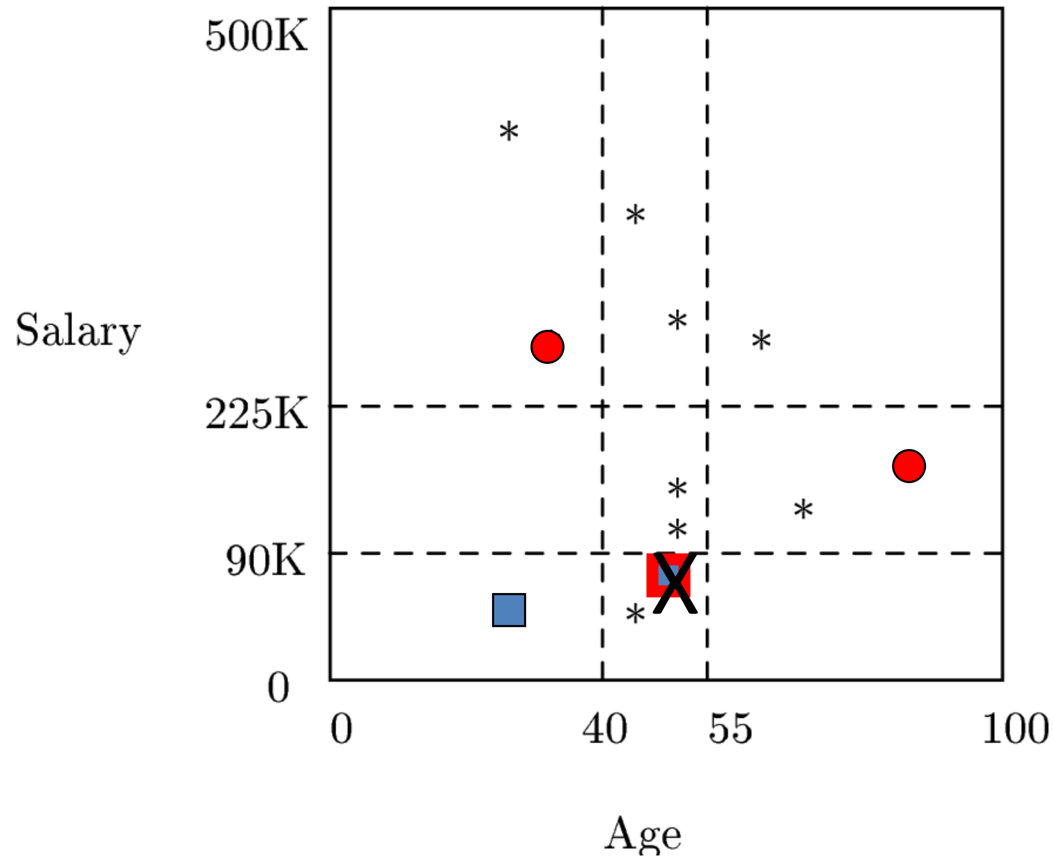
IB2 example

- Data:
  - (25,60,no)
  - (85,140,yes)
  - (45,60,no)
  - (30,260,yes)
  - *(50,75,no)*



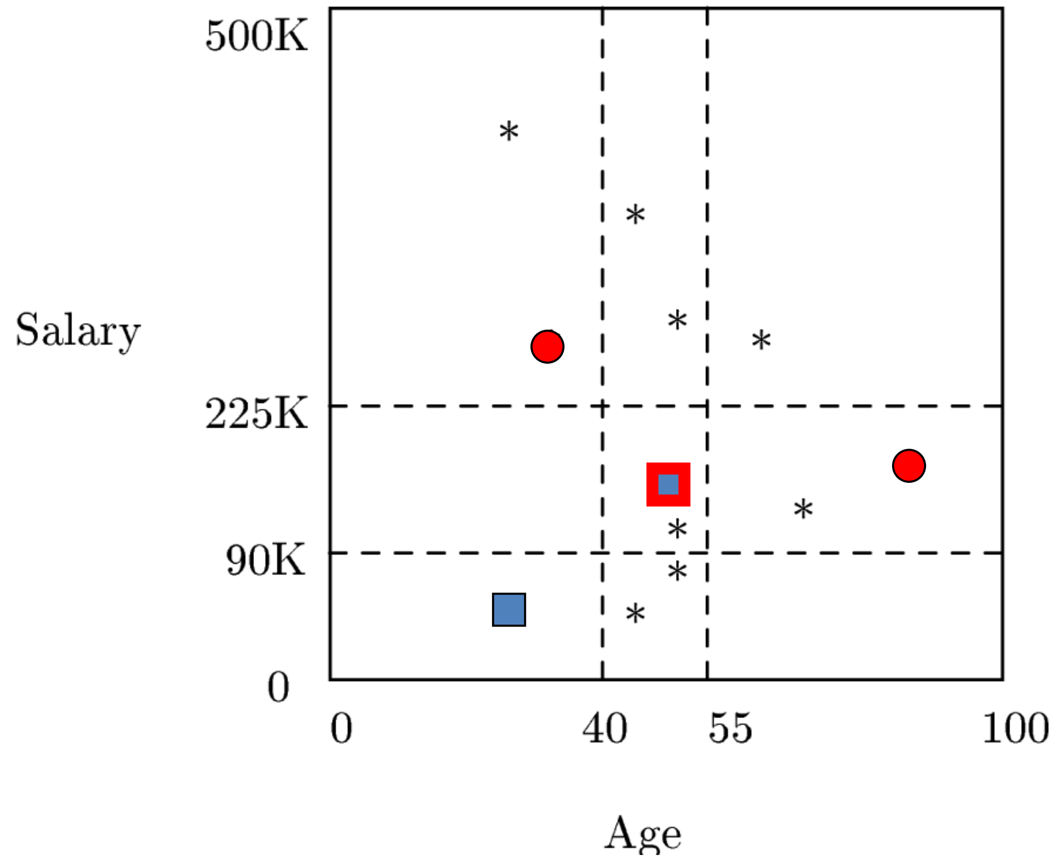
IB2 example

- Data:
  - (25,60,no)
  - (85,140,yes)
  - (45,60,no)
  - (30,260,yes)
  - *(50,75,no)*



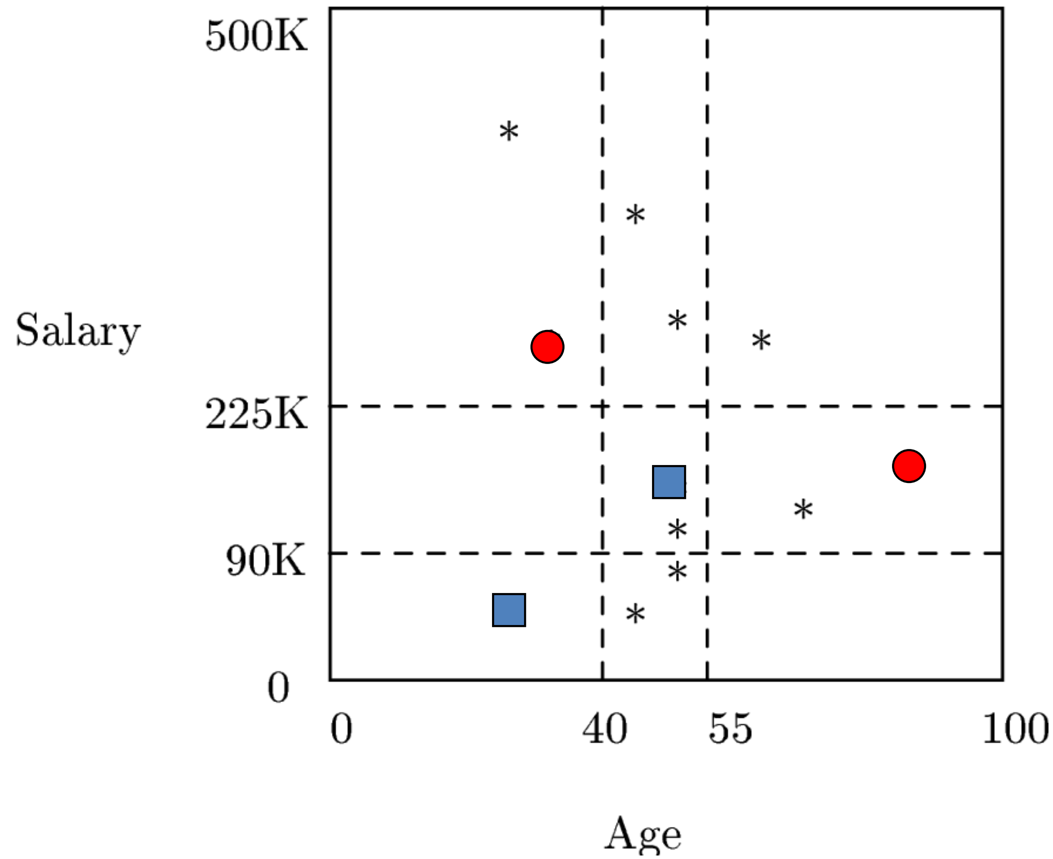
IB2 example

- Data:
  - (25,60,no)
  - (85,140,yes)
  - (45,60,no)
  - (30,260,yes)
  - (50,75,no)
  - *(50,120,no)*



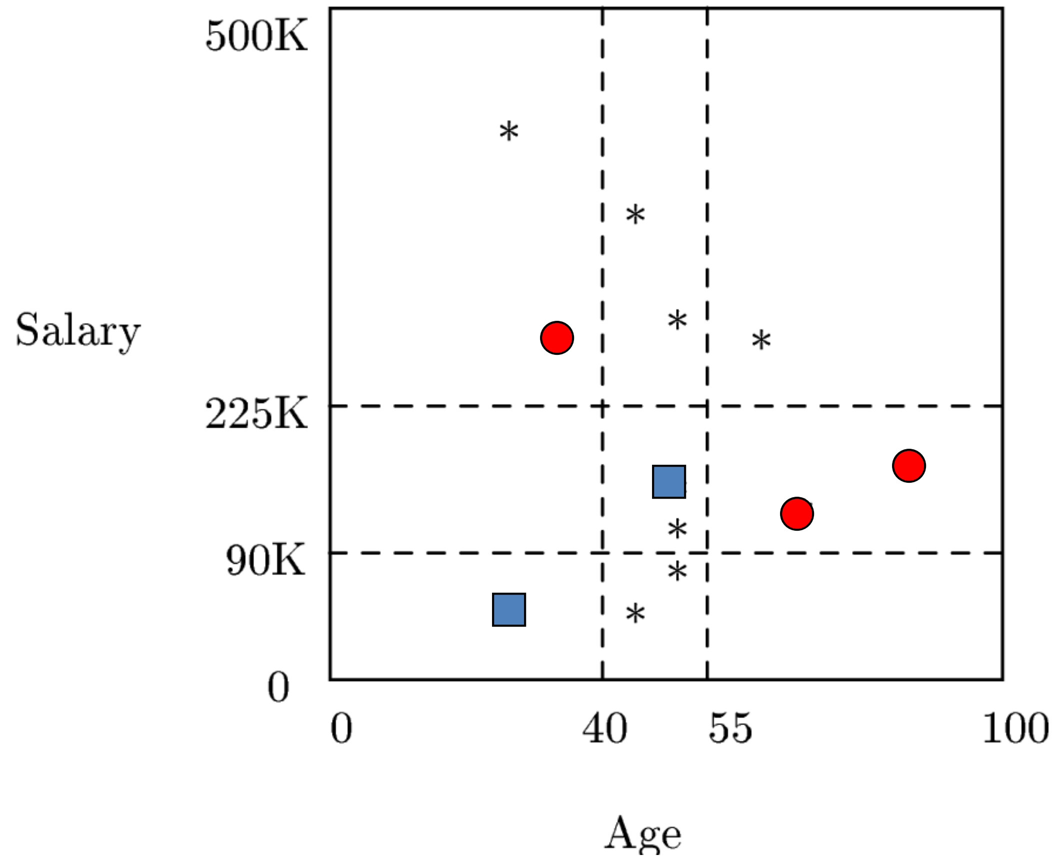
IB2 example

- Data:
  - (25,60,no)
  - (85,140,yes)
  - (45,60,no)
  - (30,260,yes)
  - (50,75,no)
  - *(50,120,no)*



IB2 example

- Continuing in a similar way, we finally get a smaller set to memorize.
  - The colored points are the ones that get memorized.



This is the final answer. I.e. we memorize only these 5 points.

IB2 summary

- Work incrementally
- Only incorporate misclassified instances

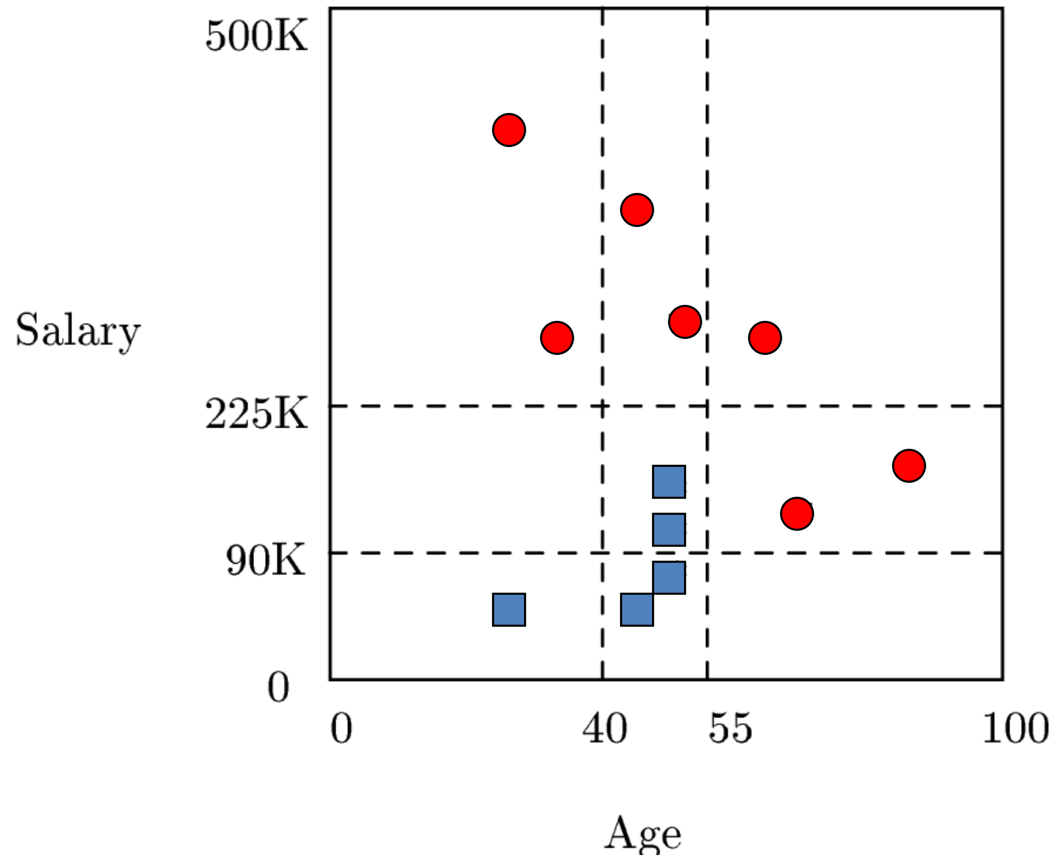
- Problem: noisy data might get incorporated

IB3 main idea

- Discard instances that don't perform well
- Keep a record of the number of correct and incorrect classification decisions that each exemplar makes.
- Two predetermined thresholds are set on success ratio. An instance is kept if:
  - If the number of incorrect classifications is $\leq$ the negative threshold ε
  - If the number of correct classifications $\geq$ the positive threshold γ.

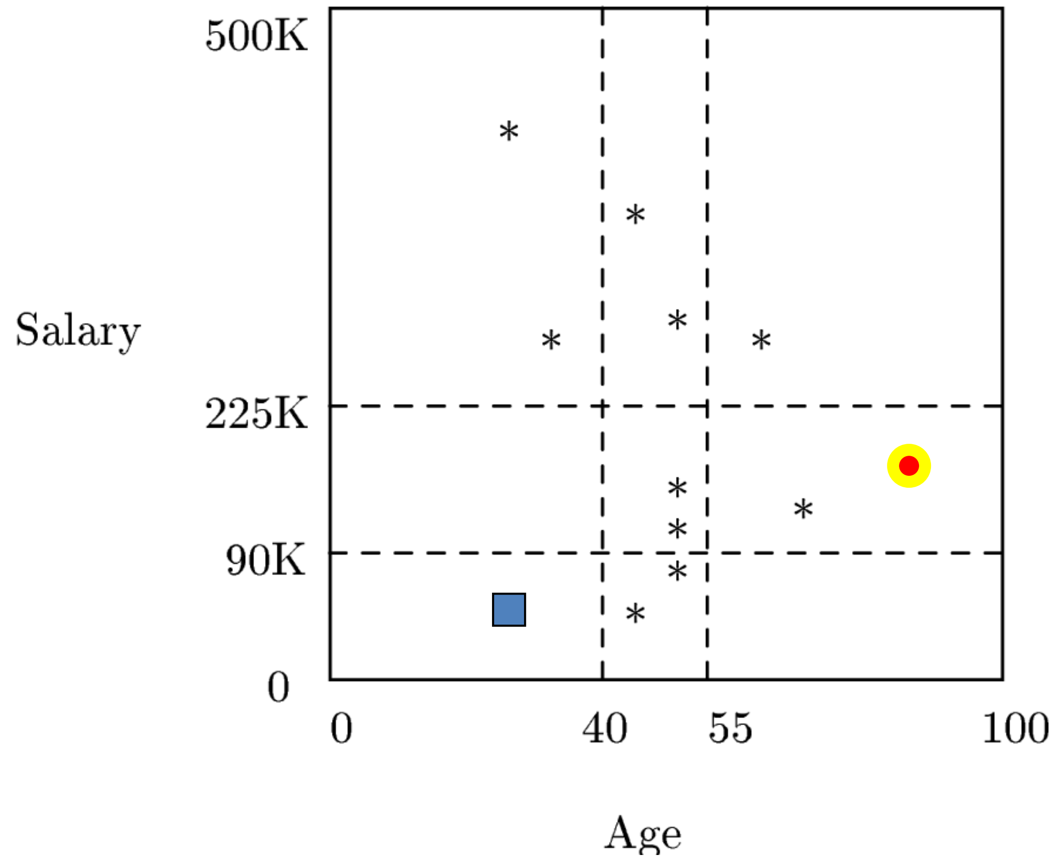
IB3 example

- Suppose the lower threshold ε=0, and the upper threshold γ=1.
- Shuffle the original dataset:
  - (25,60,no)
  - (85,140,yes)
  - (45,60,no)
  - (30,260,yes)
  - (50,75,no)
  - (50,120,no)
  - (70,110,yes)
  - (25,400,yes)
  - (50,100,no)
  - (45,350,yes)
  - (50,275,yes)
  - (60,260,yes)



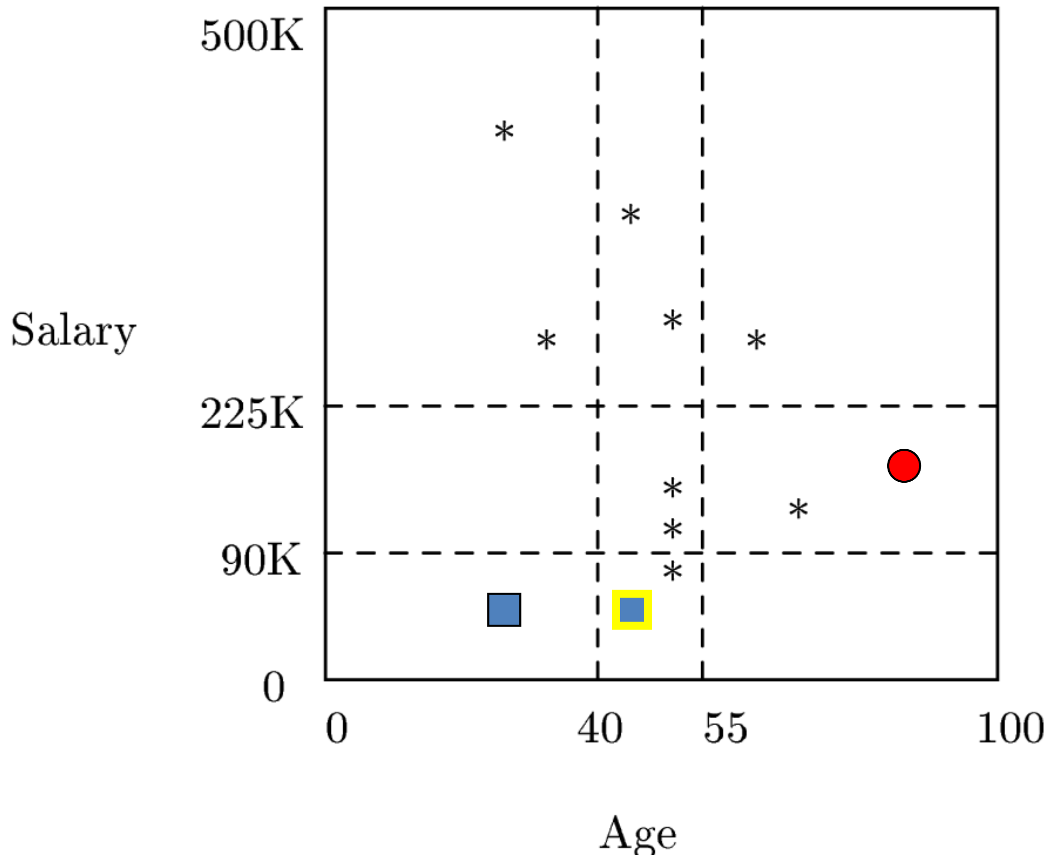
IB3 example

- ε=0, γ=1.
- Adding two first instances:
  - (25,60,no) [1, 0]
  - (85,140,yes)



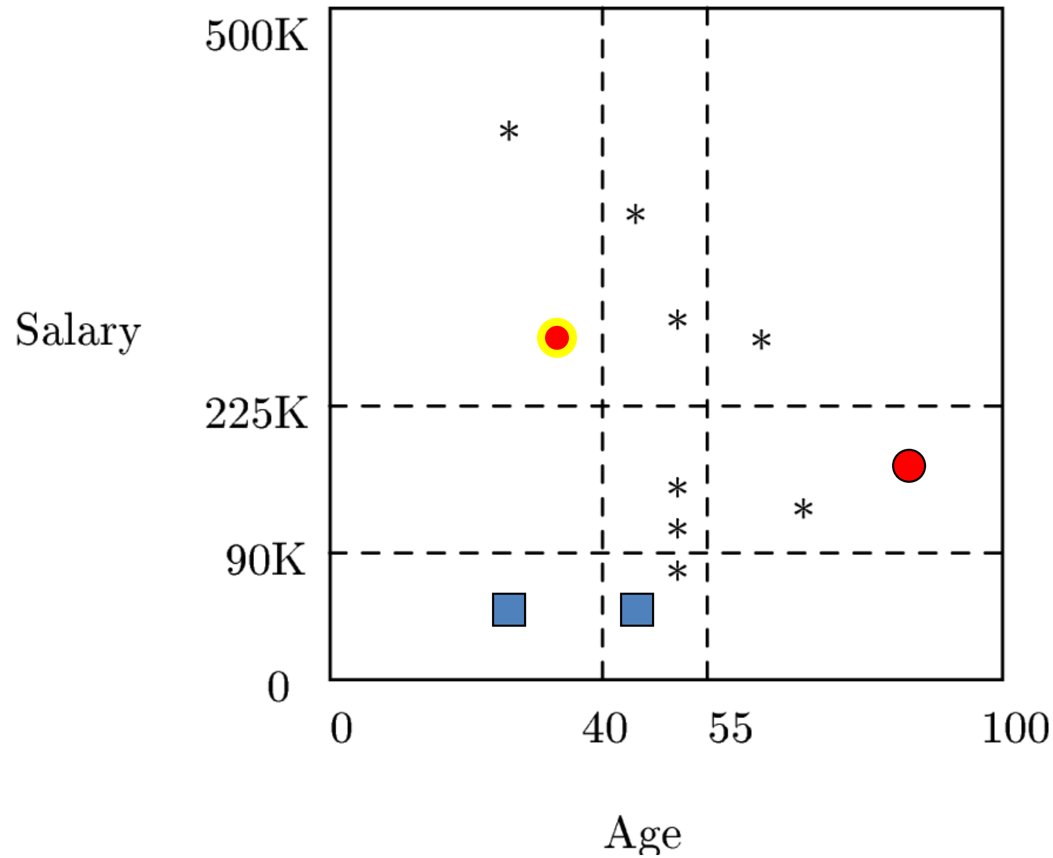
IB3 example

- ε=0, γ=1.
- Adding next:
  - (25,60,no) [1, 1]
  - (85,140,yes) [0, 0]
  - (45,60,no)



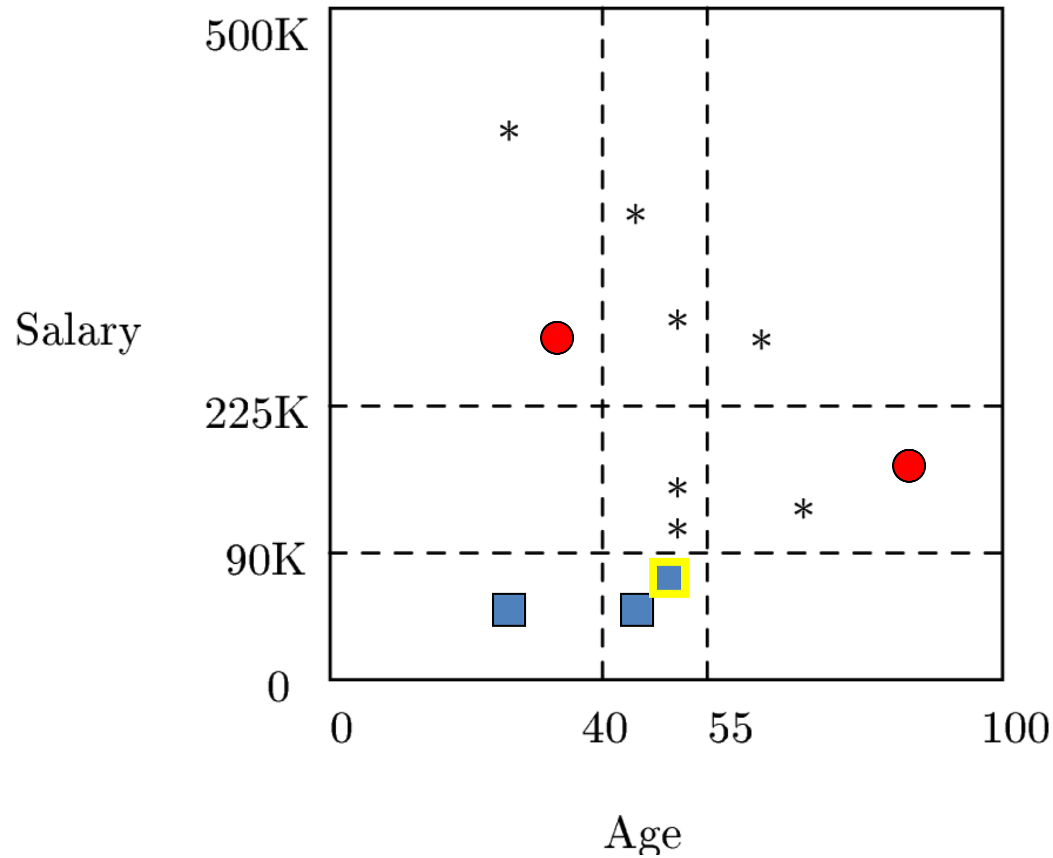
IB3 example

- ε=0, γ=1.
  - (25,60,no) [2,1]
  - (85,140,yes) [0,0]
  - (45,60,no) [0,0]
  - (30,260,yes)



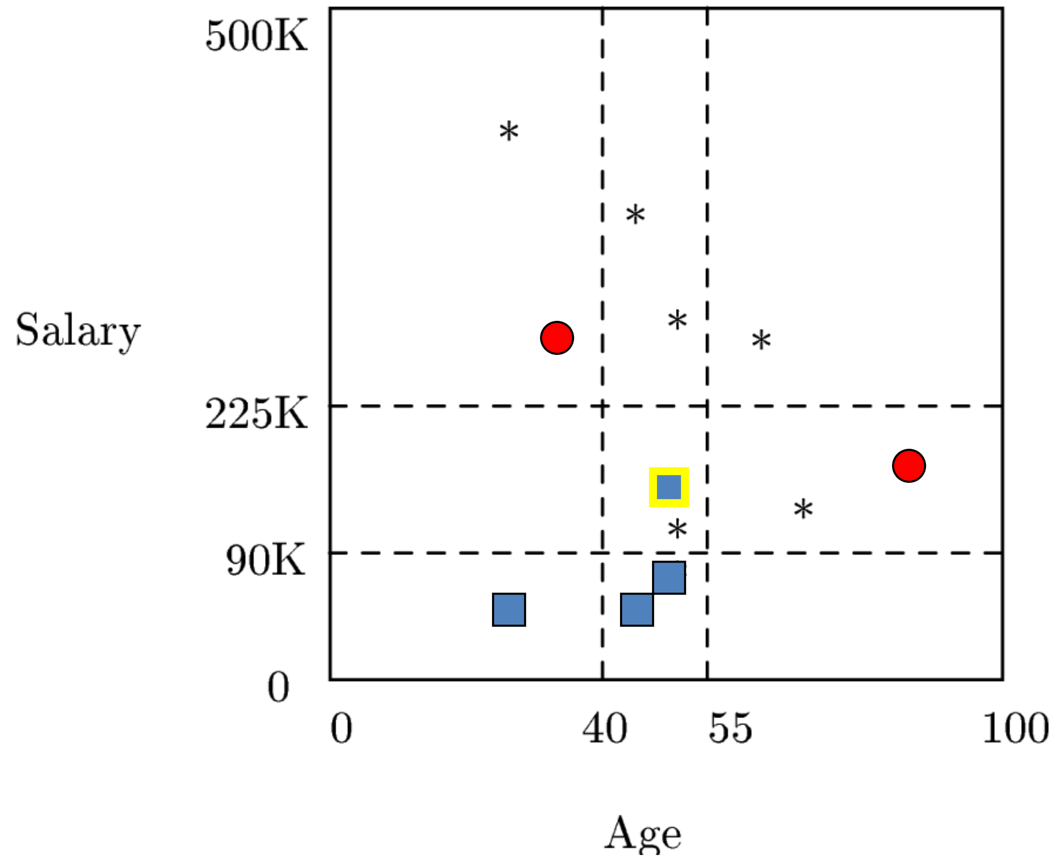
IB3 example

- ε=0, γ=1.
  - (25,60,no) [2,1]
  - (85,140,yes) [0,0]
  - (45,60,no) [0,1]
  - (30,260,yes) [0,0]
  - (50,75,no)



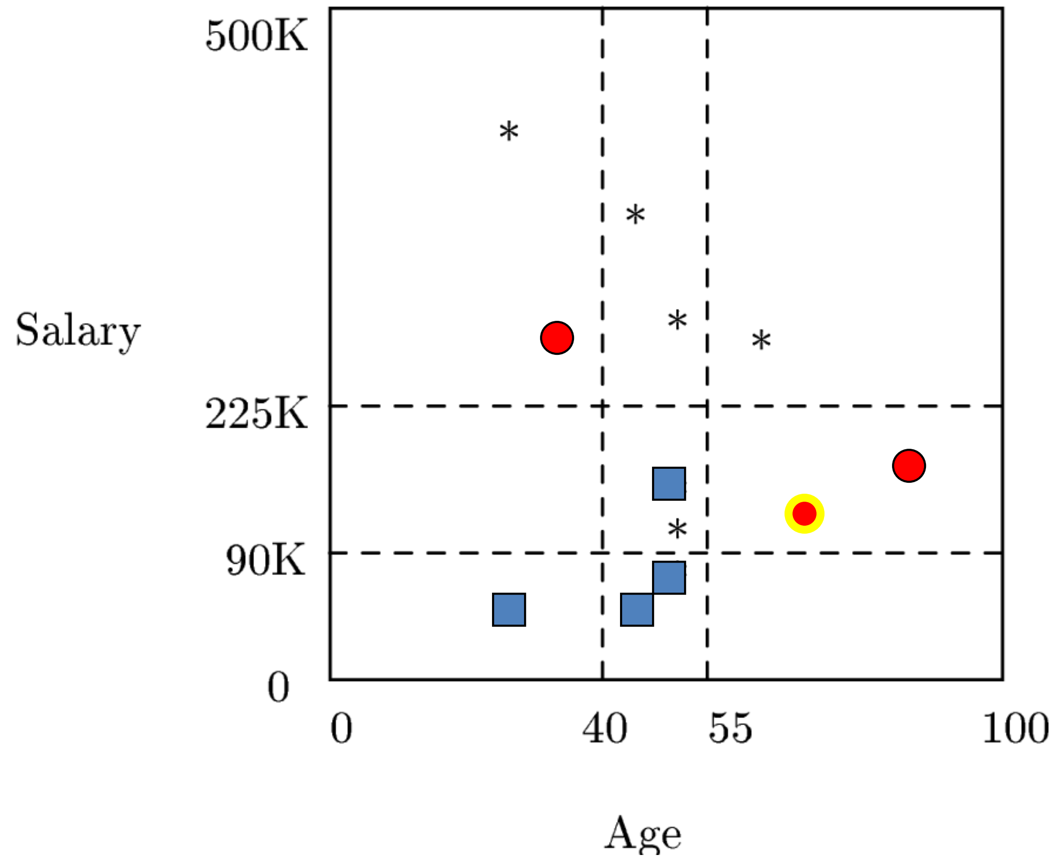
IB3 example

- ε=0, γ=1.
  - (25,60,no) [2,1]
  - (85,140,yes) [0,0]
  - (45,60,no) [0,1]
  - (30,260,yes) [0,0]
  - (50,75,no) [0,1]
  - (50,120,no)
  -



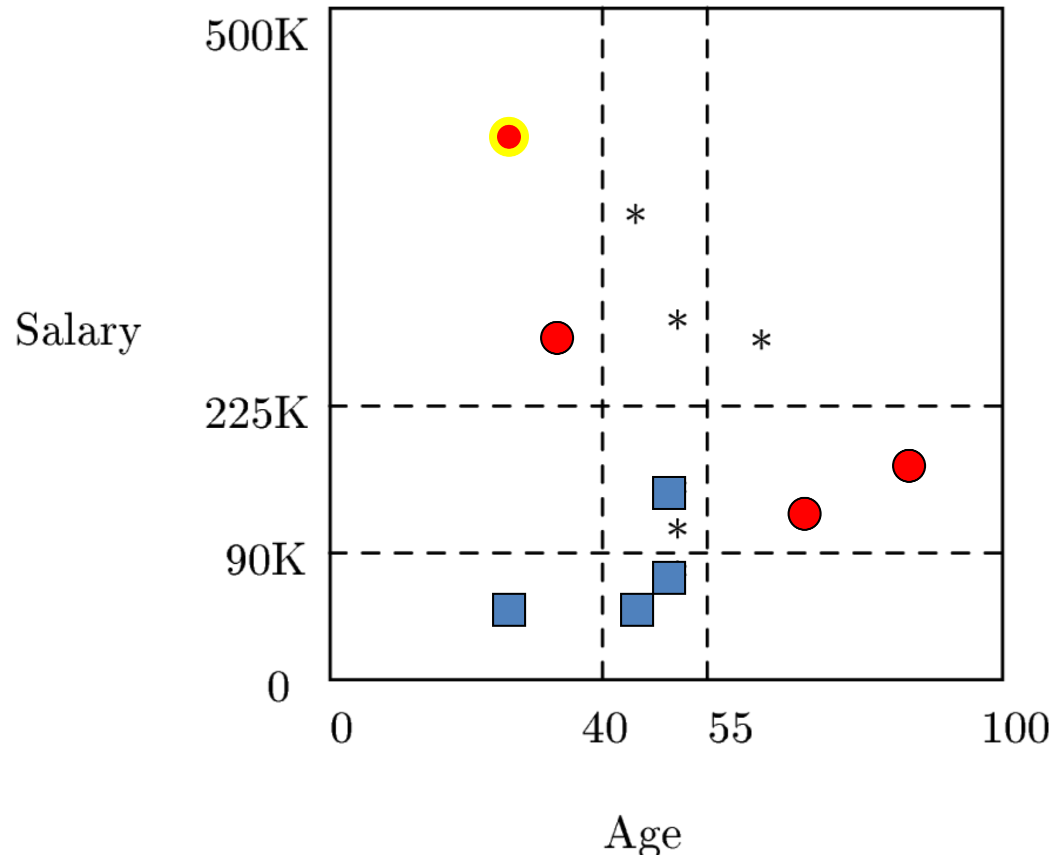
IB3 example

- ε=0, γ=1.
  - (25,60,no) [2,1]
  - (85,140,yes) [0,1]
  - (45,60,no) [0,1]
  - (30,260,yes) [0,0]
  - (50,75,no) [0,1]
  - (50,120,no) [0,0]
  - (70,110,yes)



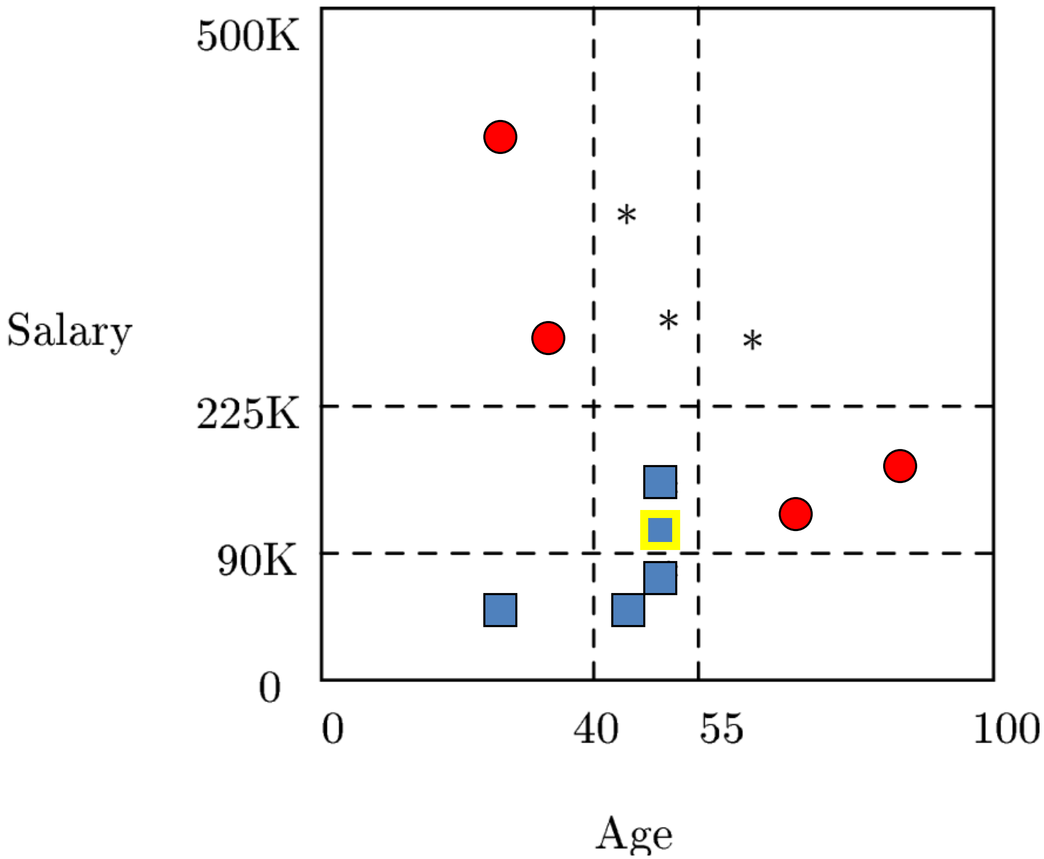
IB3 example

- ε=0, γ=1.
  - (25,60,no) [2,1]
  - (85,140,yes) [0,1]
  - (45,60,no) [0,1]
  - (30,260,yes) [0, 1]
  - (50,75,no) [0, 1]
  - (50,120,no) [0, 0]
  - (70,110,yes) [0, 0]
  - (25,400,yes)



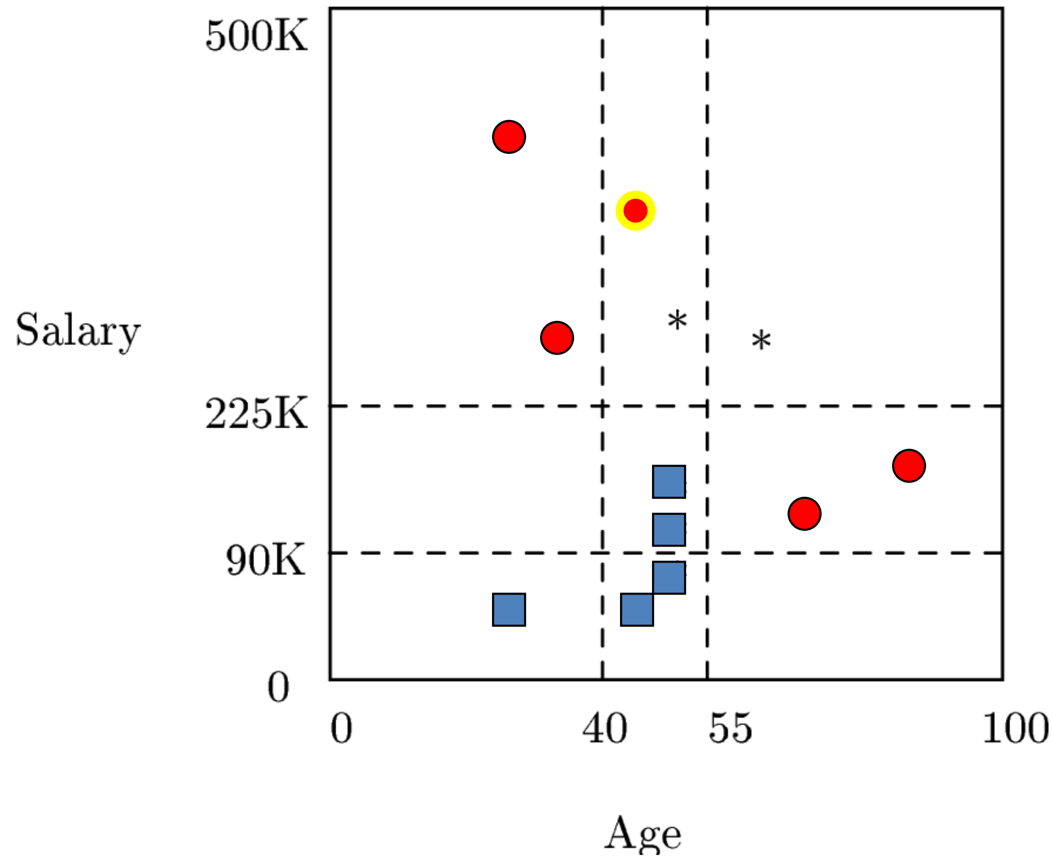
IB3 example

- ε=0, γ=1.
  - (25,60,no) [2, 1]
  - (85,140,yes) [0, 1]
  - (45,60,no) [0, 1]
  - (30,260,yes) [0, 1]
  - (50,75,no) [0, 1]
  - (50,120,no) [0, 1]
  - (70,110,yes) [0, 0]
  - (25,400,yes) [0, 0]
  - (50,100,no)



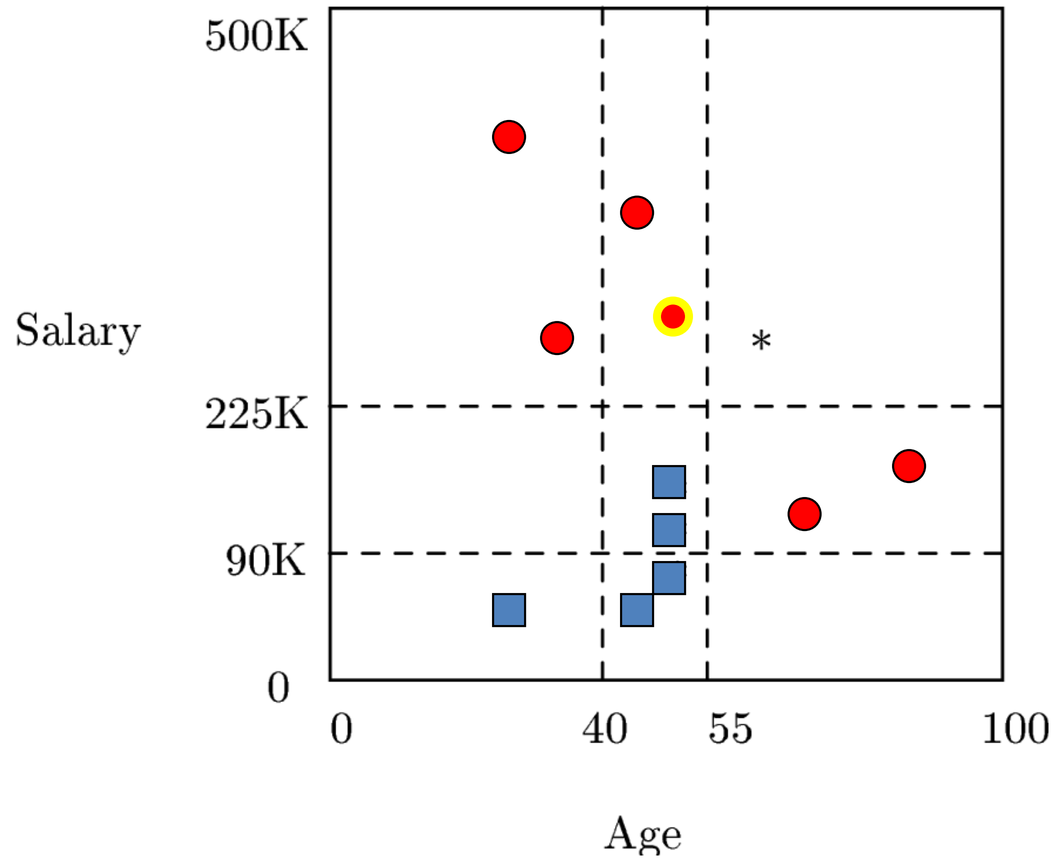
IB3 example

- ε=0, γ=1.
  - (25,60,no) [2, 1]
  - (85,140,yes) [0, 1]
  - (45,60,no) [0, 1]
  - (30,260,yes) [0, 1]
  - (50,75,no) [0, 1]
  - (50,120,no) [0, 1]
  - (70,110,yes) [0, 0]
  - (25,400,yes) [0, 1]
  - (50,100,no) [0, 0]
  - (45,350,yes)



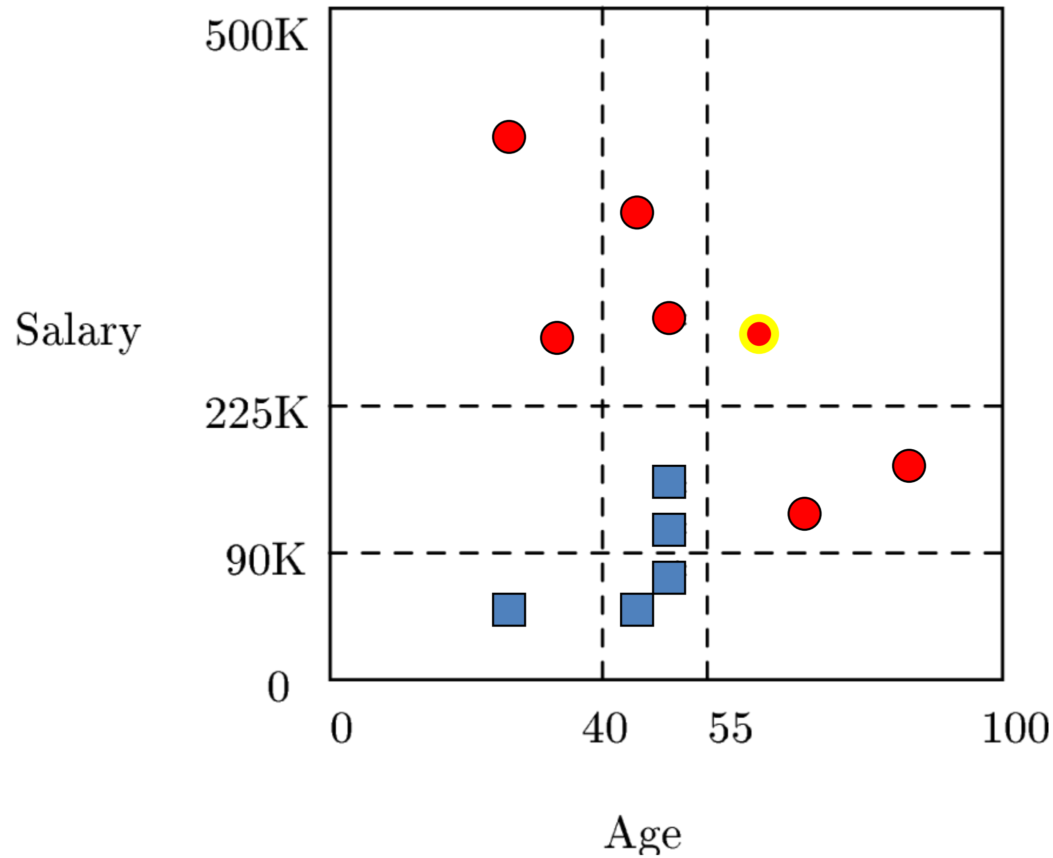
IB3 example

- ε=0, γ=1.
  - (25,60,no) [2, 1]
  - (85,140,yes) [0, 1]
  - (45,60,no) [0, 1]
  - (30,260,yes) [0, 2]
  - (50,75,no) [0, 1]
  - (50,120,no) [0, 1]
  - (70,110,yes) [0, 0]
  - (25,400,yes) [0, 1]
  - (50,100,no) [0, 0]
  - (45,350,yes) [0, 0]
  - (50,275,yes)



IB3 example

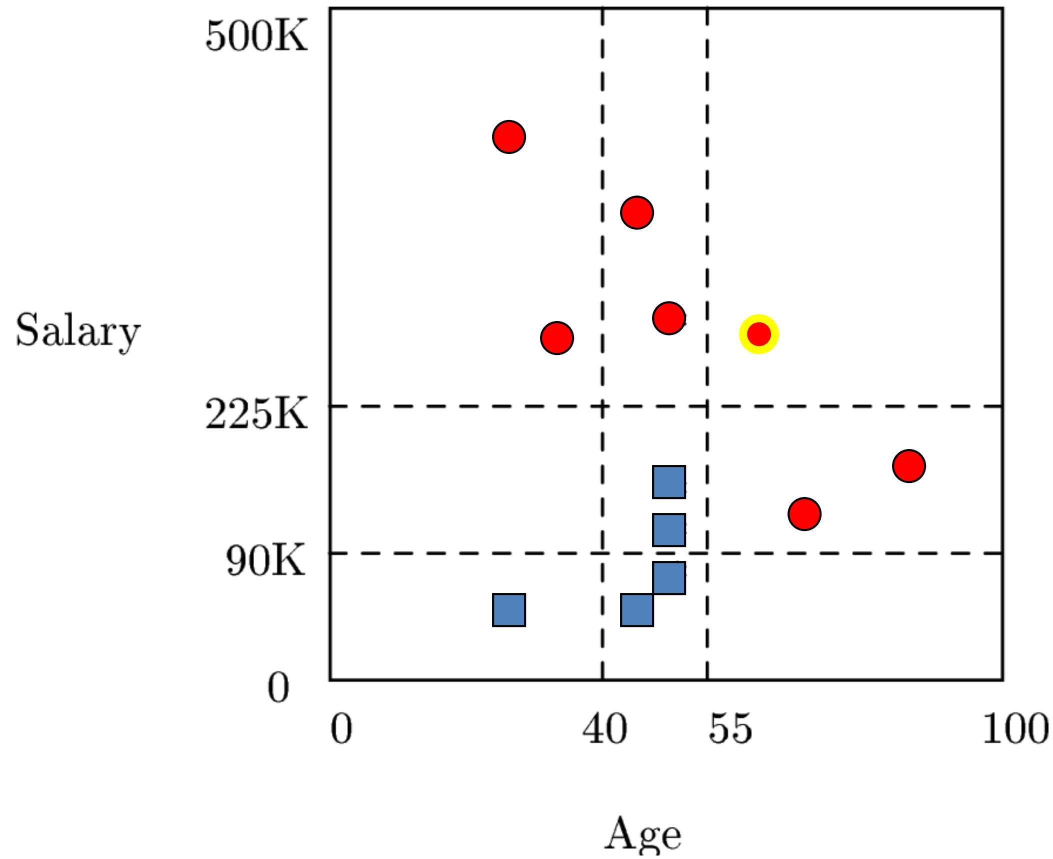
- ε=0, γ=1.
  - (25,60,no) [2, 1]
  - (85,140,yes) [0, 1]
  - (45,60,no) [0, 1]
  - (30,260,yes) [0, 2]
  - (50,75,no) [0, 1]
  - (50,120,no) [0, 1]
  - (70,110,yes) [0, 0]
  - (25,400,yes) [0, 1]
  - (50,100,no) [0, 0]
  - (45,350,yes) [0, 0]
  - (50,275,yes) [0, 1]
  - (60,260,yes)



What do we discard?

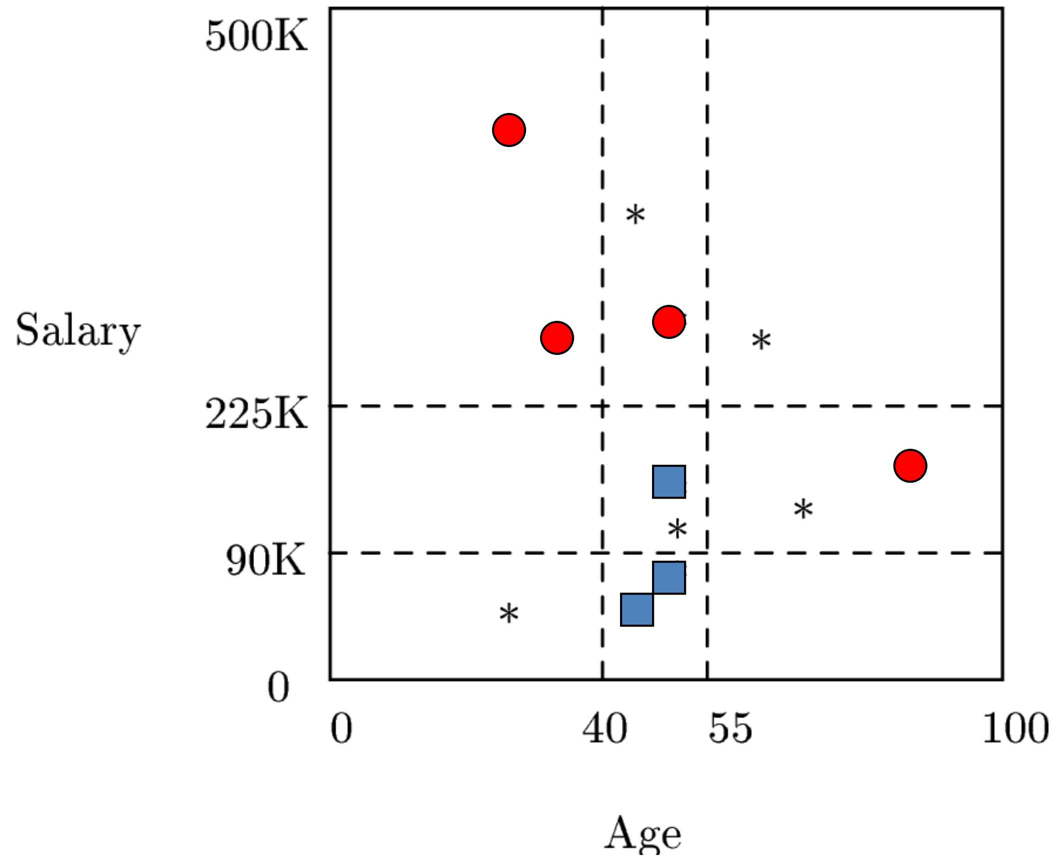
IB3 example

- ε=0, γ=1.
  - ~~(25,60,no) [2, 1]~~
  - (85,140,yes) [0, 1]
  - (45,60,no) [0, 1]
  - (30,260,yes) [0, 2]
  - (50,75,no) [0, 1]
  - (50,120,no) [0, 1]
  - ~~(70,110,yes) [0, 0]~~
  - (25,400,yes) [0, 1]
  - ~~(50,100,no) [0, 0]~~
  - ~~(45,350,yes) [0, 0]~~
  - (50,275,yes) [0, 1]
  - ~~(60,260,yes)~~



IB3 example

- The points that will be kept for classification are:
  - (85,140,yes) [0, 1]
  - (45,60,no) [0, 1]
  - (30,260,yes) [0, 2]
  - (50,75,no) [0, 1]
  - (50,120,no) [0, 1]
  - (25,400,yes) [0, 1]
  - (50,275,yes) [0, 1]



IB3 summary

- Discard instances that don't perform well
- Keep a record of the number of correct and incorrect classification decisions that each exemplar makes.
- After all instances have been added keep only the ones with:
  - The number of incorrect classifications is $\leq \varepsilon$
  - The number of correct classifications $\geq \gamma$.