Memory-based reasoning: nearest neighbors

Lecture 18

Classification example: bankruptcy dataset

Dataset

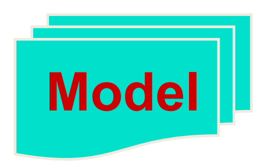
Late payments, L	Spending ratio, R	Bankruptcy
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1.0	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes

Class labels

New customer

L	R	B
2	0.3	?

Classify



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

Memory-based reasoning

Seems poisonous

Amanita muscaria

Classification by similarity

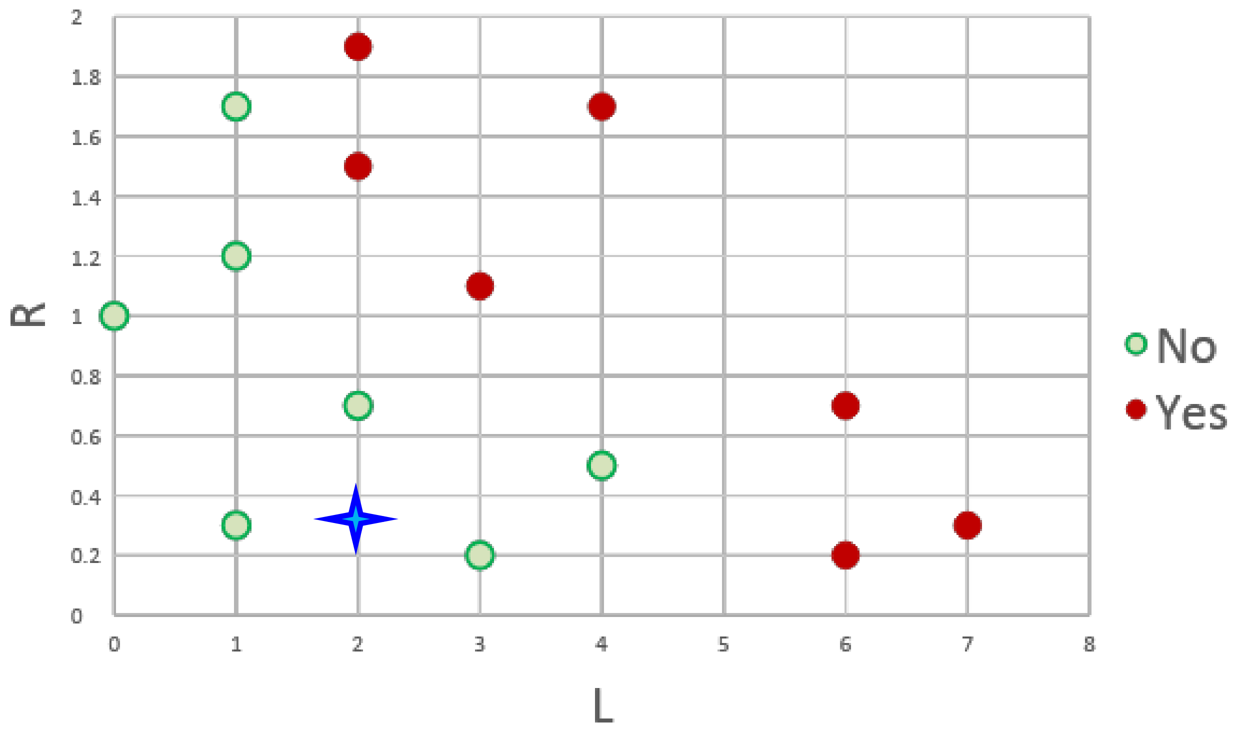
“ If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a … [Class label] ”

New classifier: Nearest Neighbor

- Remember the 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

Predicting bankruptcy: nearest neighbor

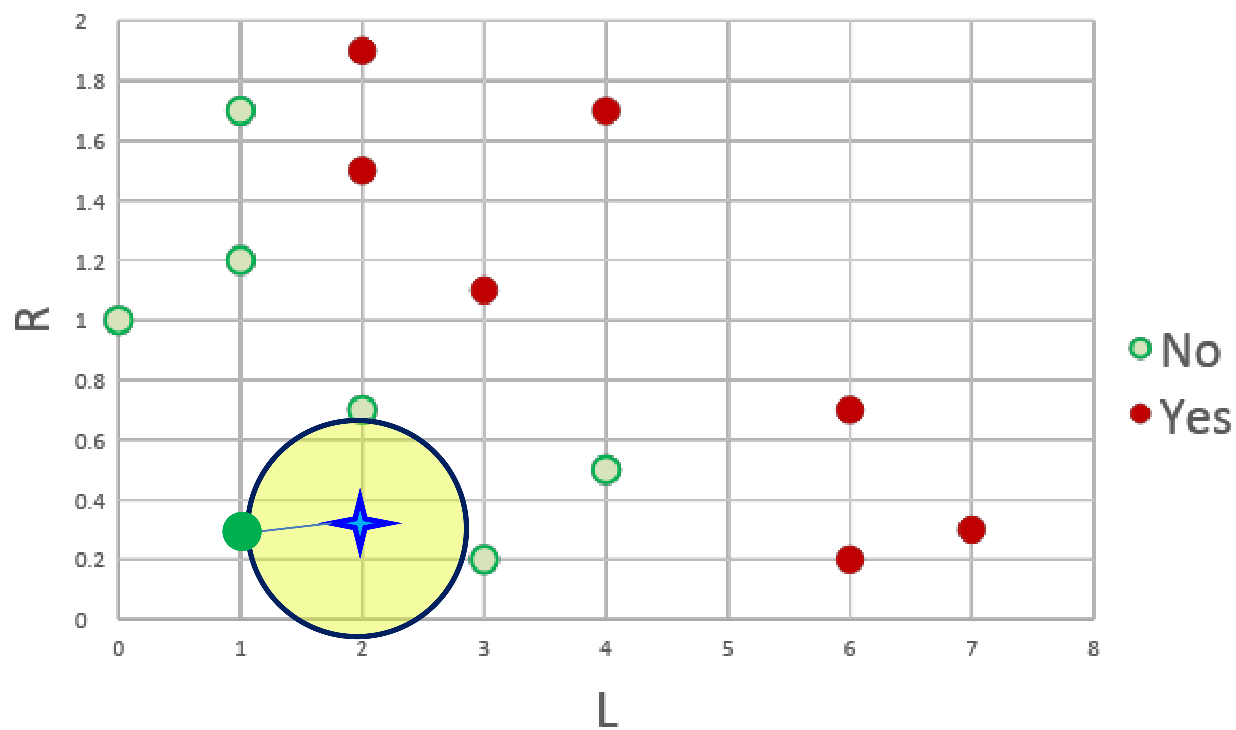
L	R
2	0.3



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

Predicting bankruptcy: nearest neighbor

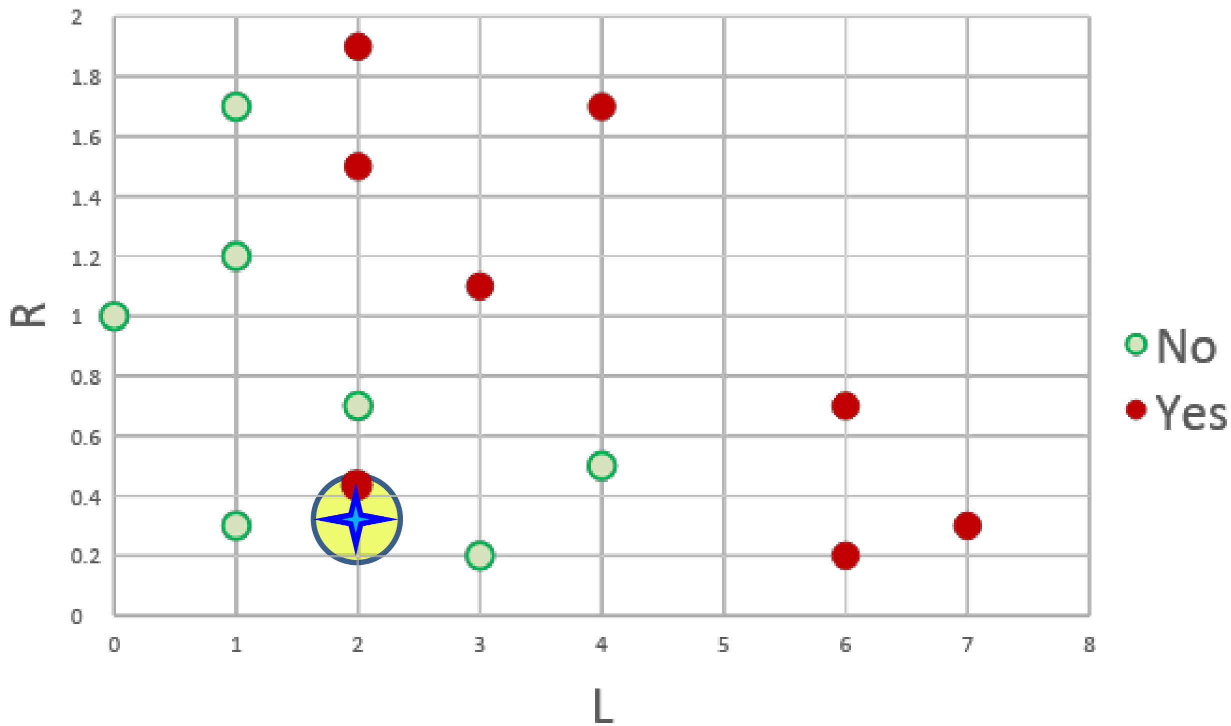
L	R
2	0.3



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

Predicting bankruptcy: noise

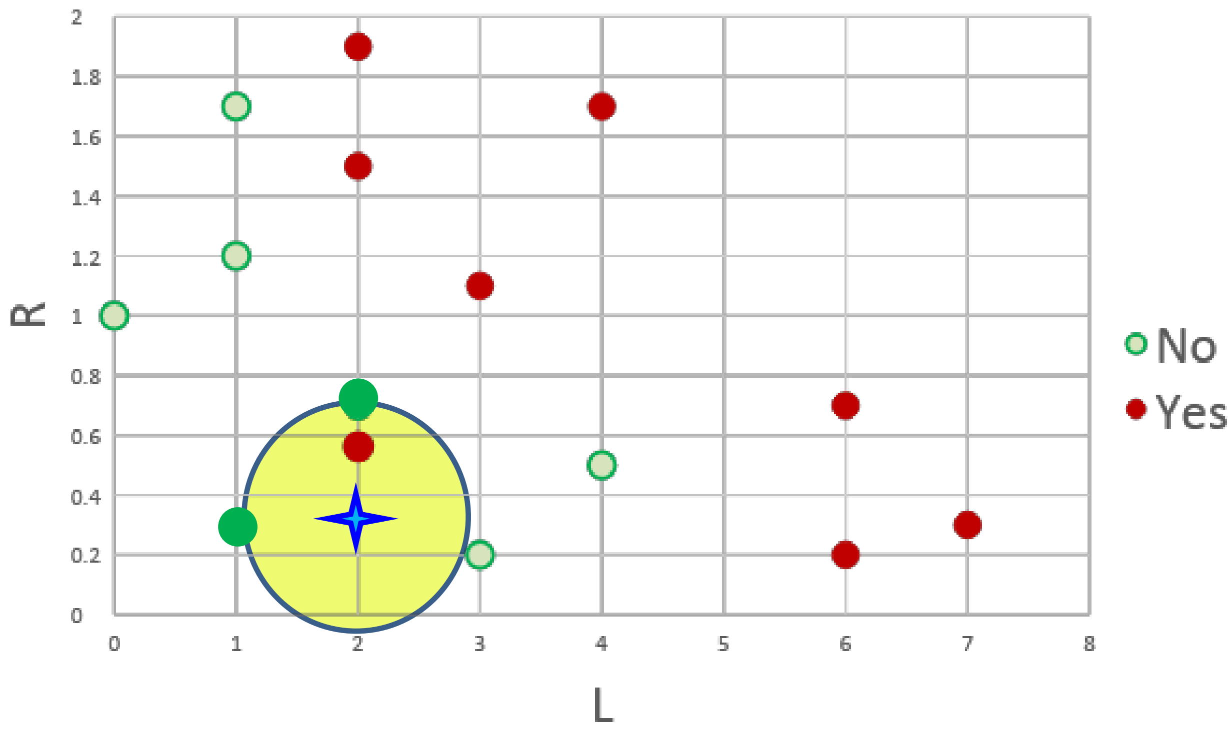
L	R
2	0.3



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: lazy classifier

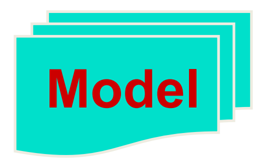
Dataset

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	?

Classify



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

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:

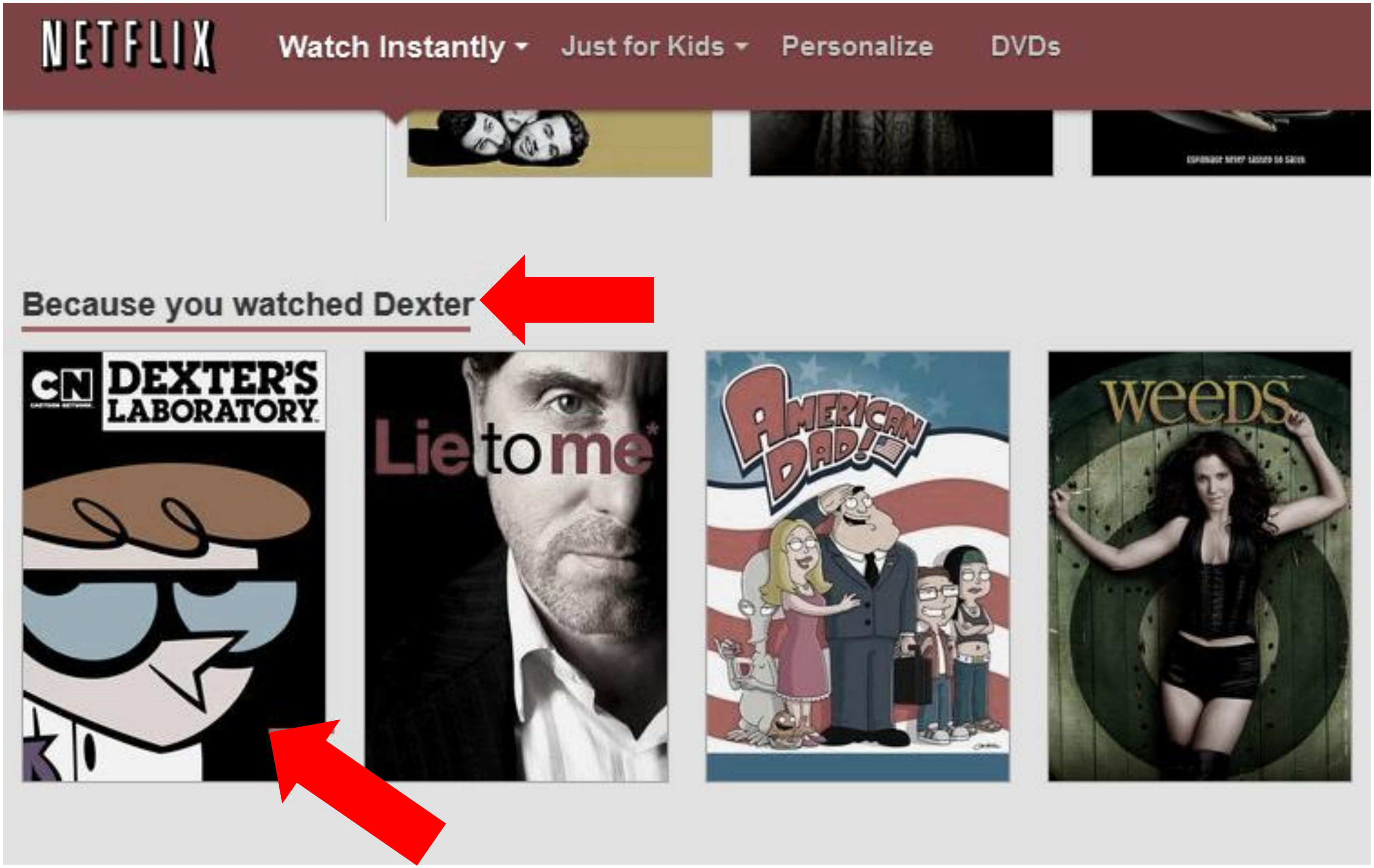
return ***combination*** $(\boldsymbol{L}_{1\ldots K})$

We will discuss:

- Distance/similarity function
- Combining neighbor class labels
- How many neighbors: choice of K *K should be odd*
- The best number of dimensions

Proximity between data records

How do we define proximity?



Numeric *proximity* (similarity or distance) between data records

- Combination of proximity measures for each attribute
- In this approach, each attribute is considered a separate and independent (orthogonal) dimension of the data

Similarity/distance across a single dimension

- First step: translate all fields into numeric variables, to be able to compute similarity (distance) across each dimension
- How to make all attributes numeric: see slide deck 02. Describing and visualizing (slides 6-13)

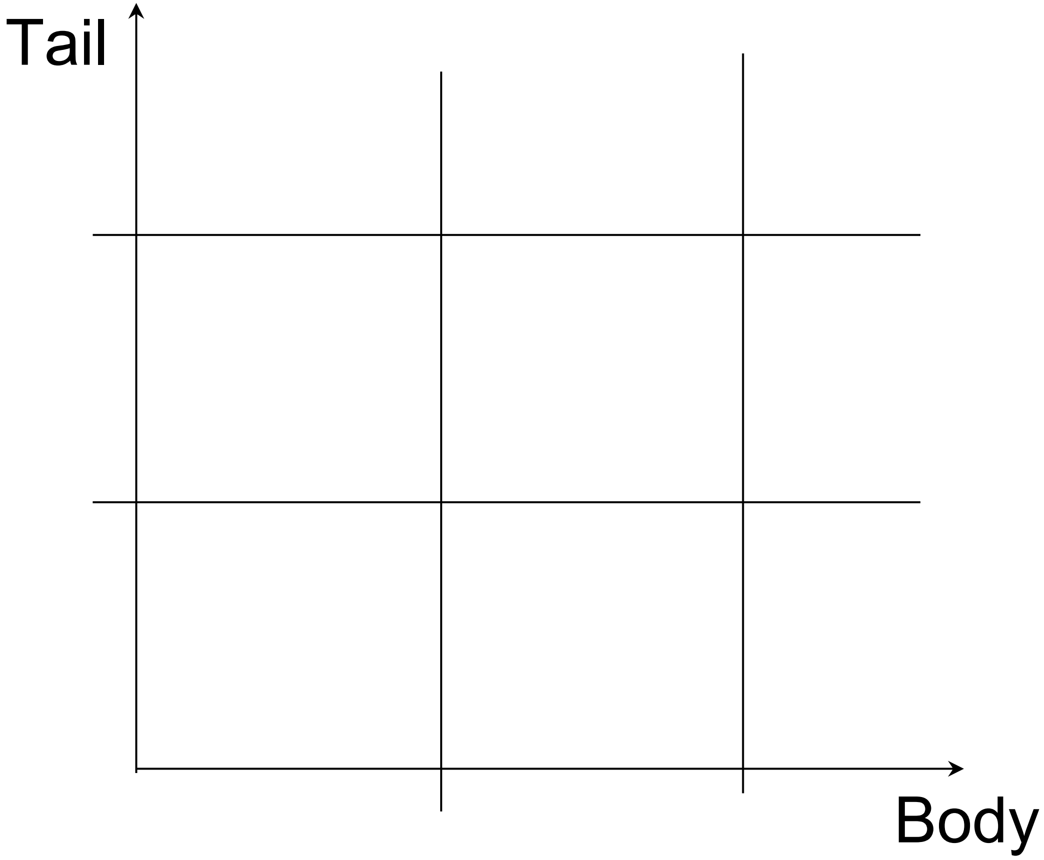
Summary on proximity measures for a single attribute

Attribute type	Distance (dissimilarity)	Similarity
True measures	d=|x-y|	s=-d, $\boxed{s=1/(1+d)}$, s=1-(d-min_d)/(max_d-min_d)
Ordinal	d=|x-y|/(n-1) (values mapped to integers 0 to n-1 where n is the number of values)	s=1-d
Nominal (Categorical)	d= 0 if x=y d=1 if x≠y	s=1 if x=y s=0 if x≠y

Combining measures of separate attributes into a proximity measure between a pair of data records

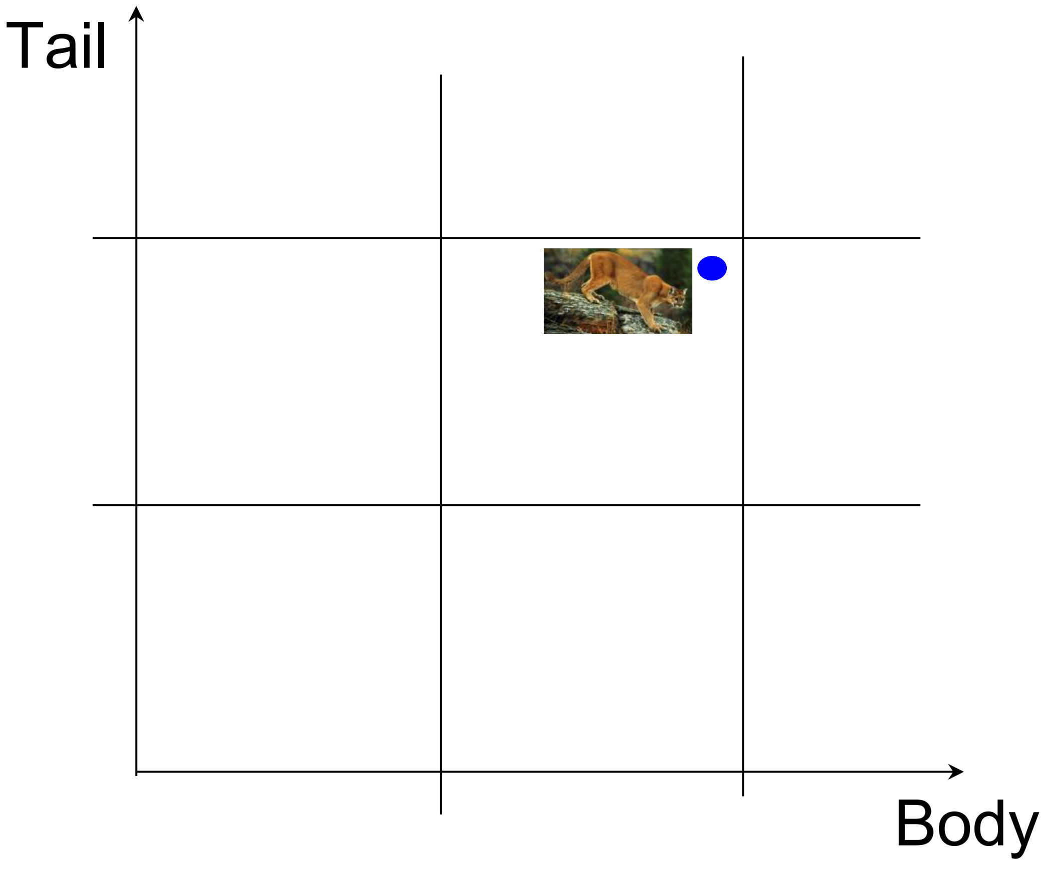
- Hundreds of similarity measures were proposed
- We learned:
  - Manhattan distance
  - Euclidean distance
  - Jaccard index
  - Tanimoto coefficient
  - Cosine similarity

Example: Cat or bear classifier



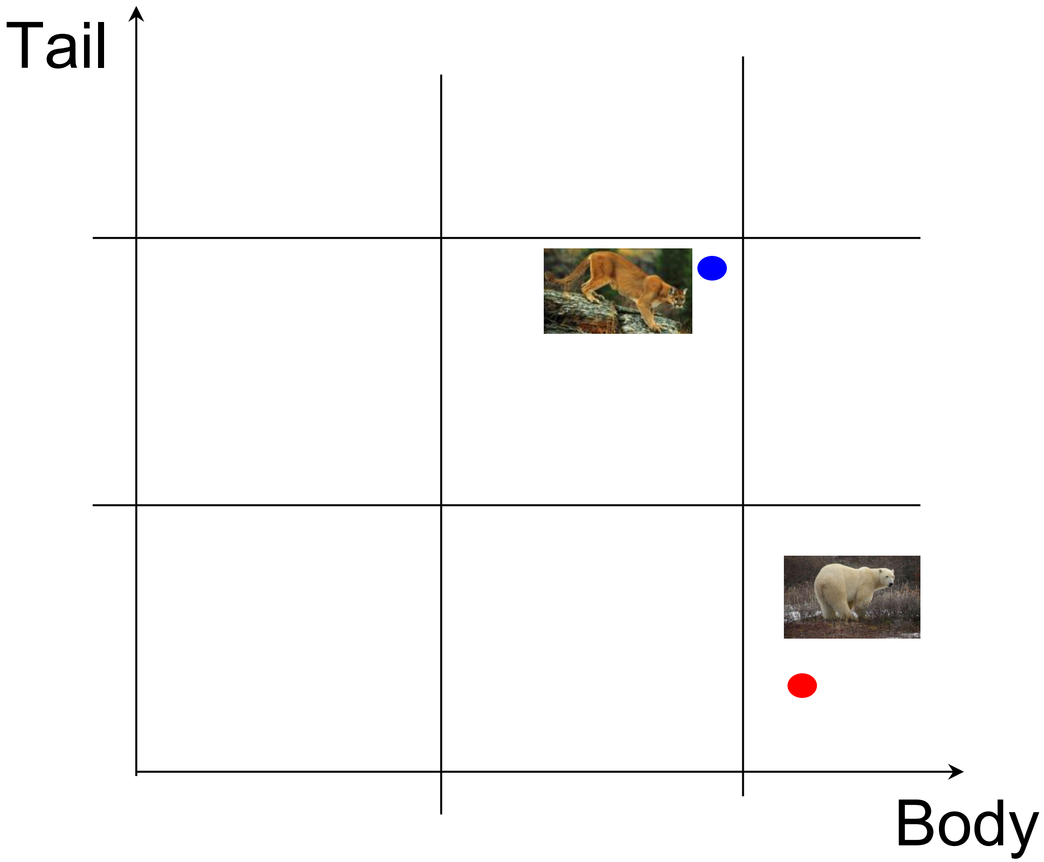
Cat or bear?

Cat or bear classifier

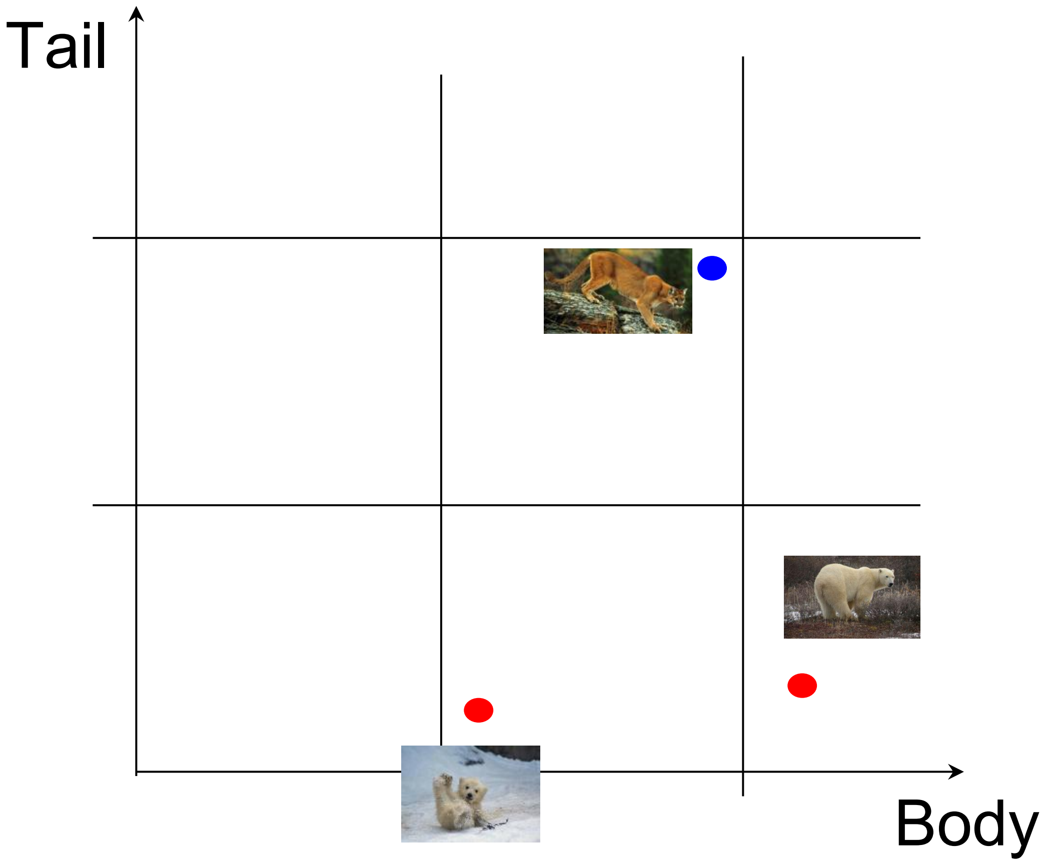


Cat or bear?

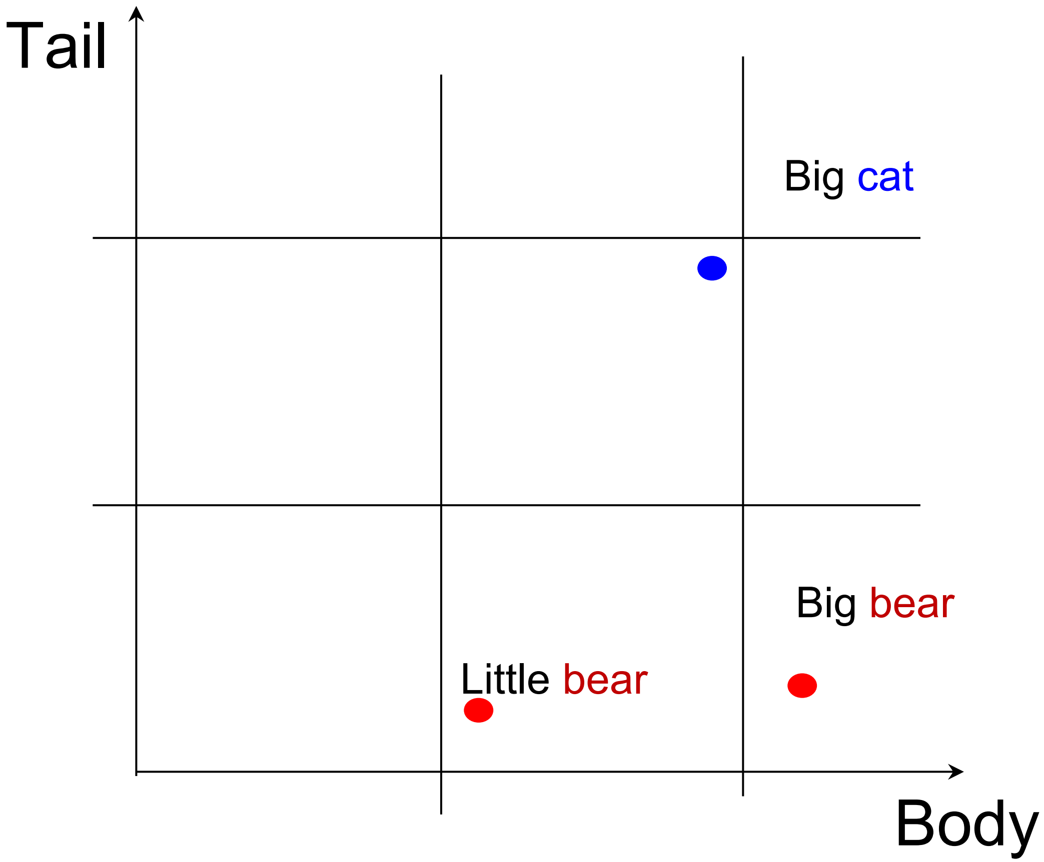
Cat or bear classifier



Cat or bear?

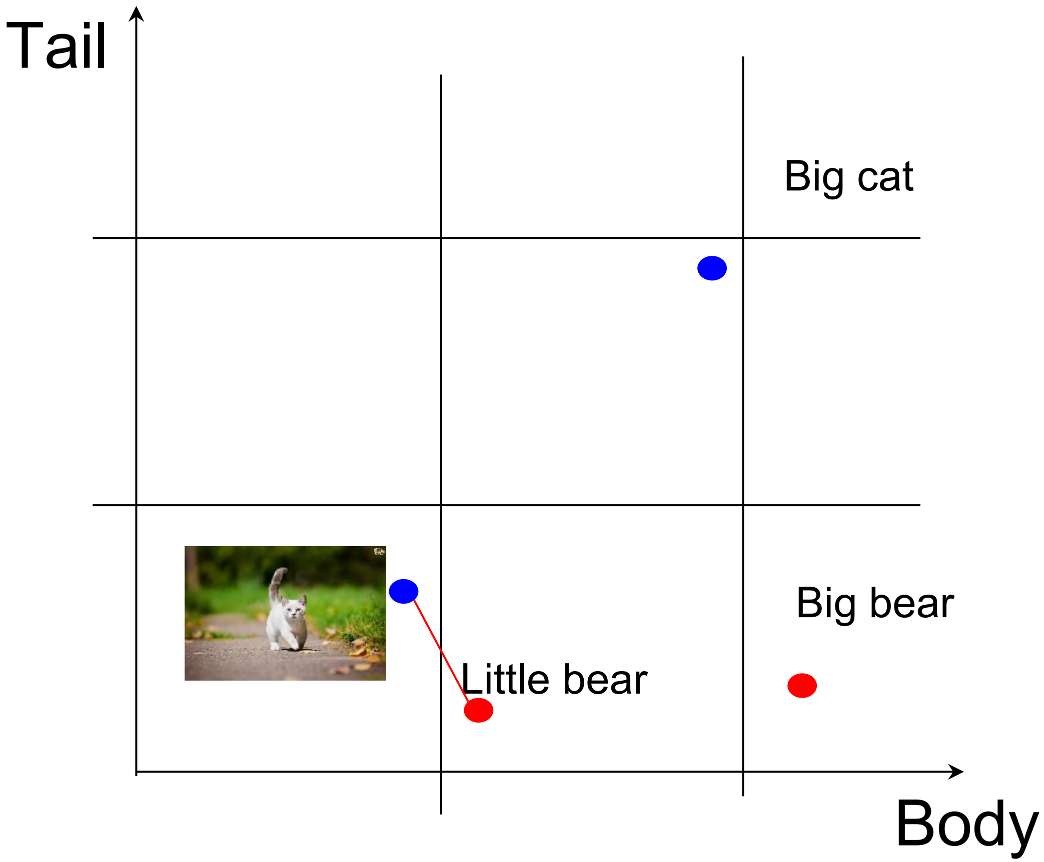


Cat or bear classifier

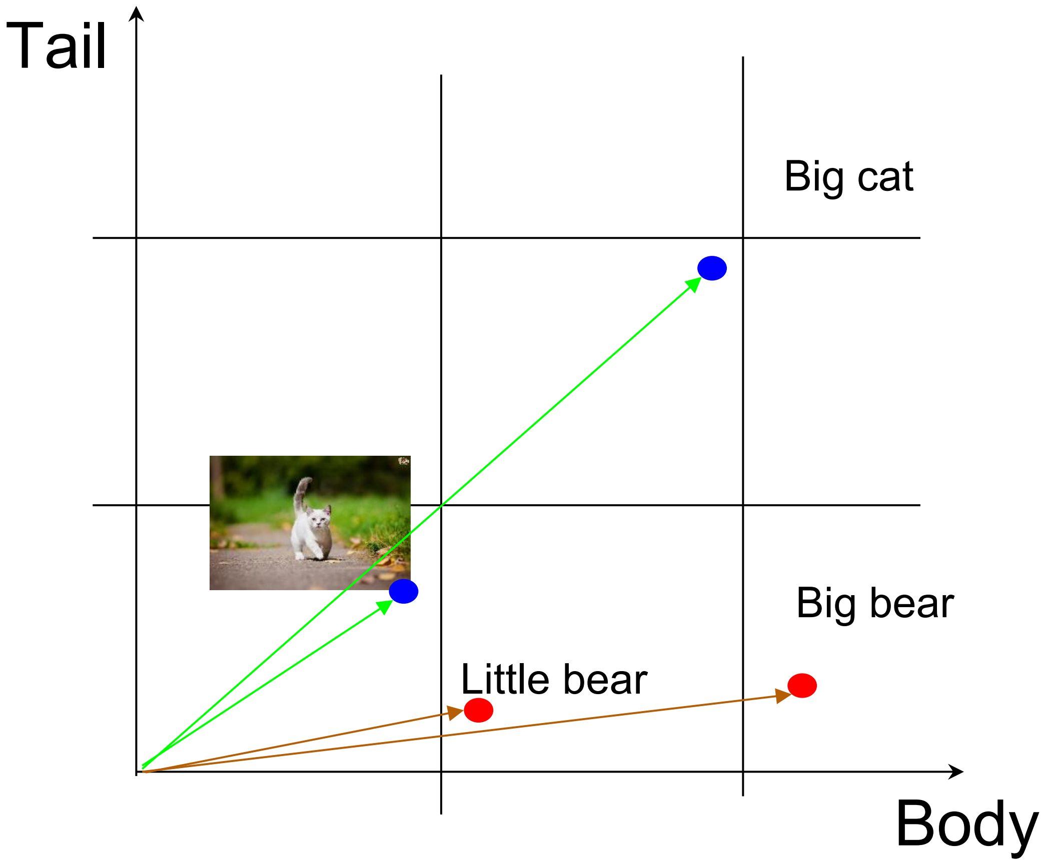


Cat or bear?

Cat or bear?



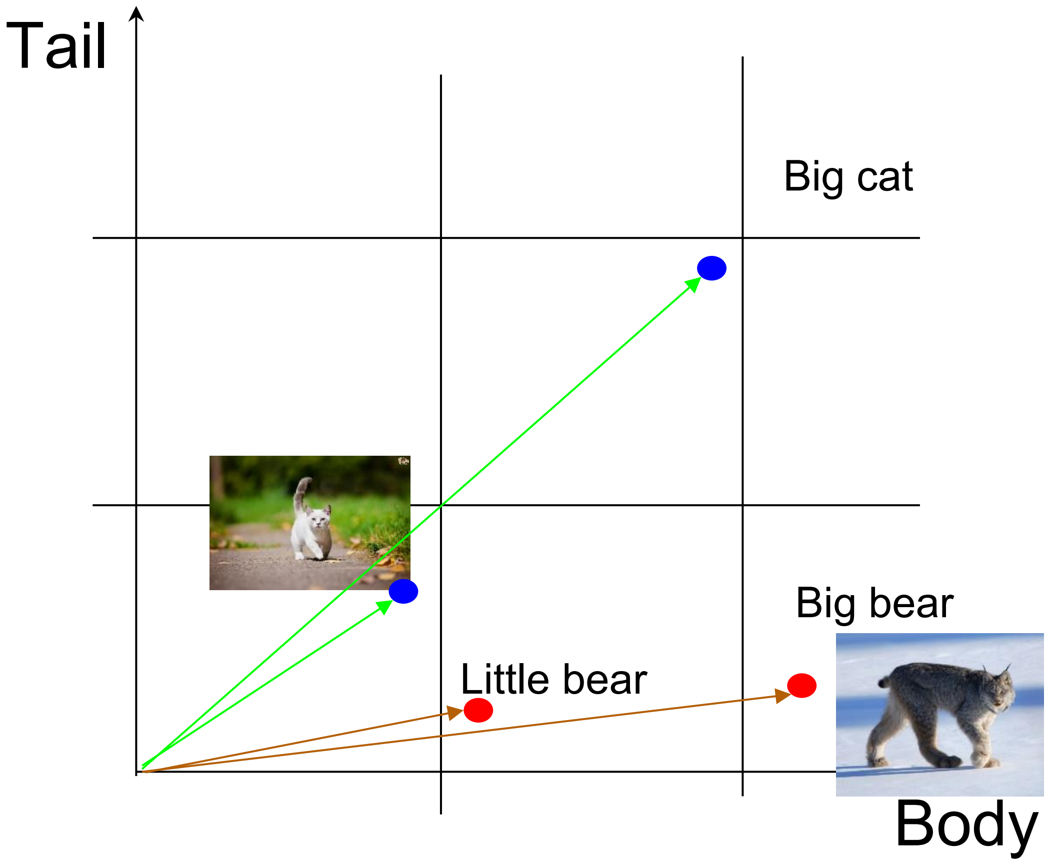
Cat or bear?
Consider angle between vectors



Cat or bear?

Canadian Lynx

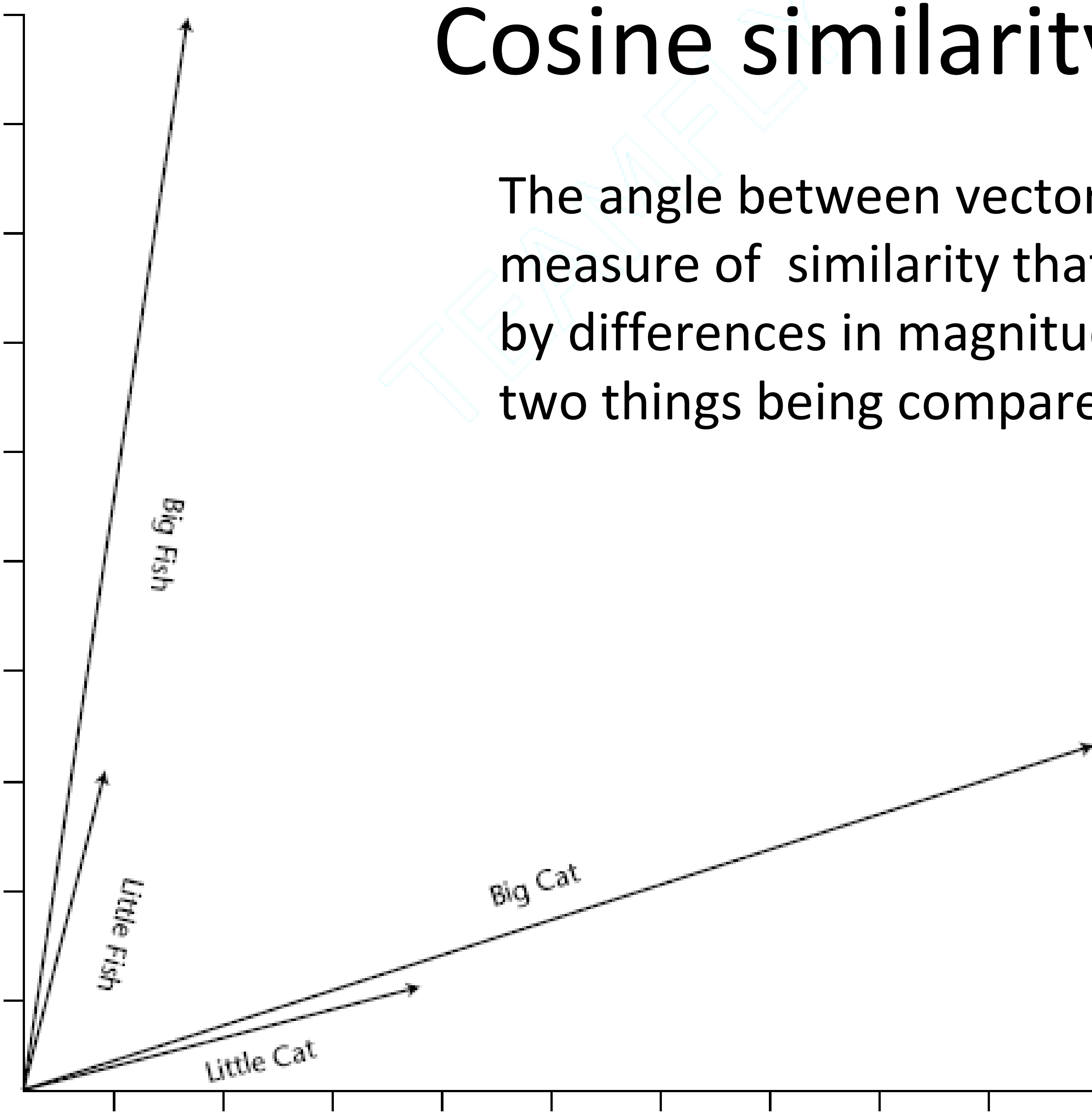
Cat or bear?
Consider angle between vectors



Cosine similarity

- Sometimes it makes more sense to compare records based on the ratio of fields ***within each record***
- Sardines should be closer to cod and tuna, while kittens closer to cougars and lions, but if we use the Euclidean distance of body-part lengths, the sardine is closer to a kitten than it is to a catfish
- Solution: we use a different vector interpretation. Instead of thinking of *X and Y as points in space*, we think of them as *position vectors and measure the angle between them*

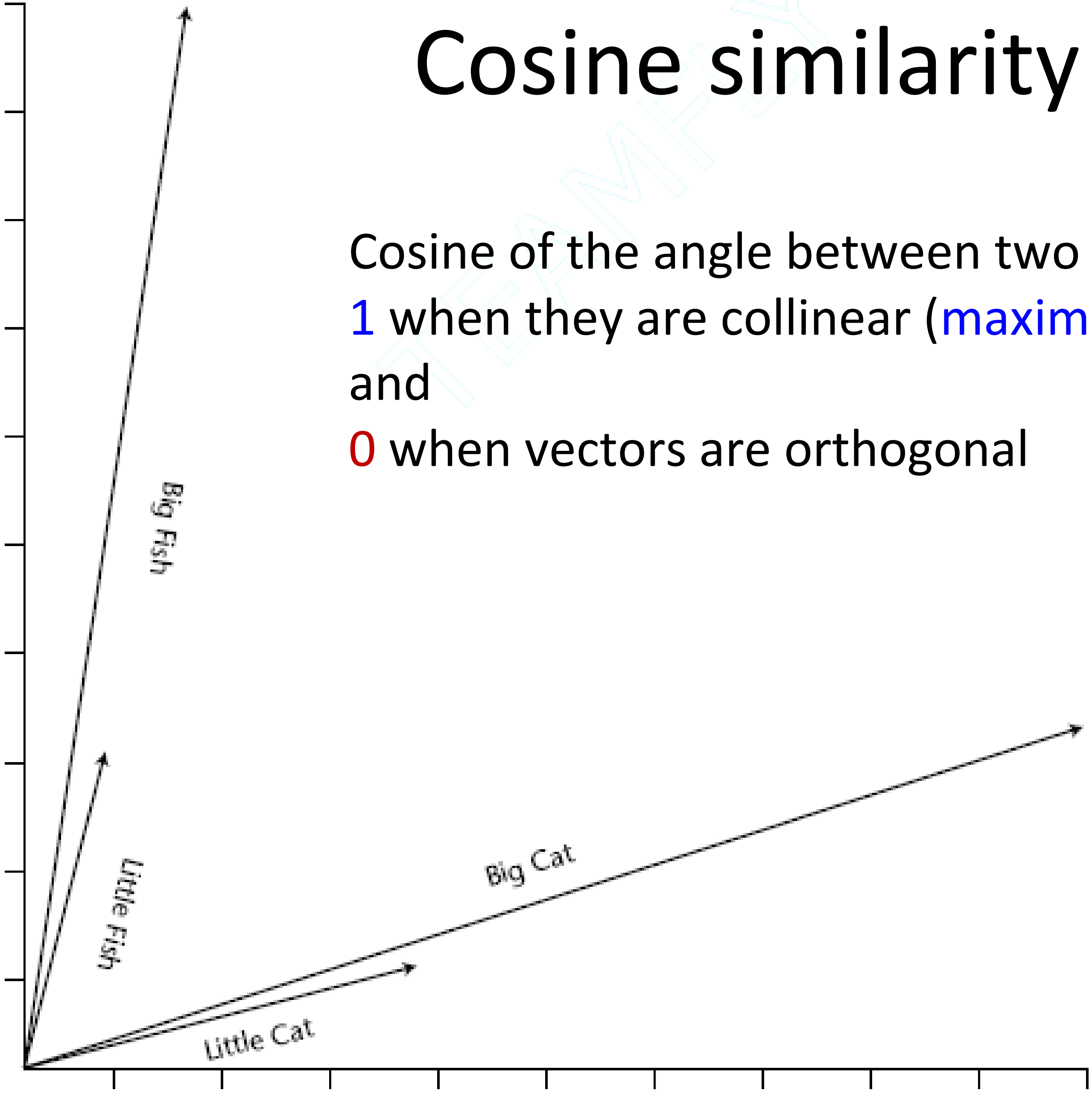
Cosine similarity



The angle between vectors provides a measure of similarity that is not influenced by differences in magnitude between the two things being compared

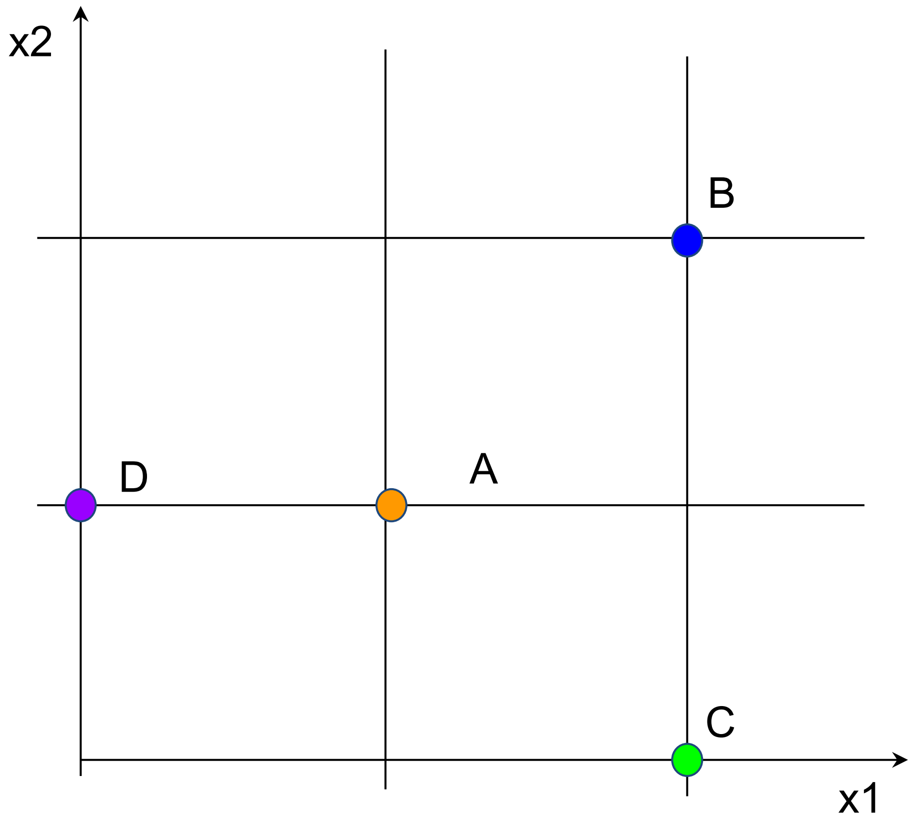
Cosine similarity

Cosine of the angle between two vectors is
1 when they are collinear (maximum similarity)
and
0 when vectors are orthogonal



Cosine similarity

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



Cosine Similarity for document vectors

	w1	w2	w3	w4	w5	w6	
x=(	1	0	0	0	0	0	)
y=(	0	0	0	1	2	0	)
z=(	0	0	0	4	8	0	)

Cosine between **x** and **y** is 0 (dot-product is 0). These documents are not similar.

Cosine between **y** and **z** is 1: though the number of times each word occurs in y and z is different, these documents are about the same topic

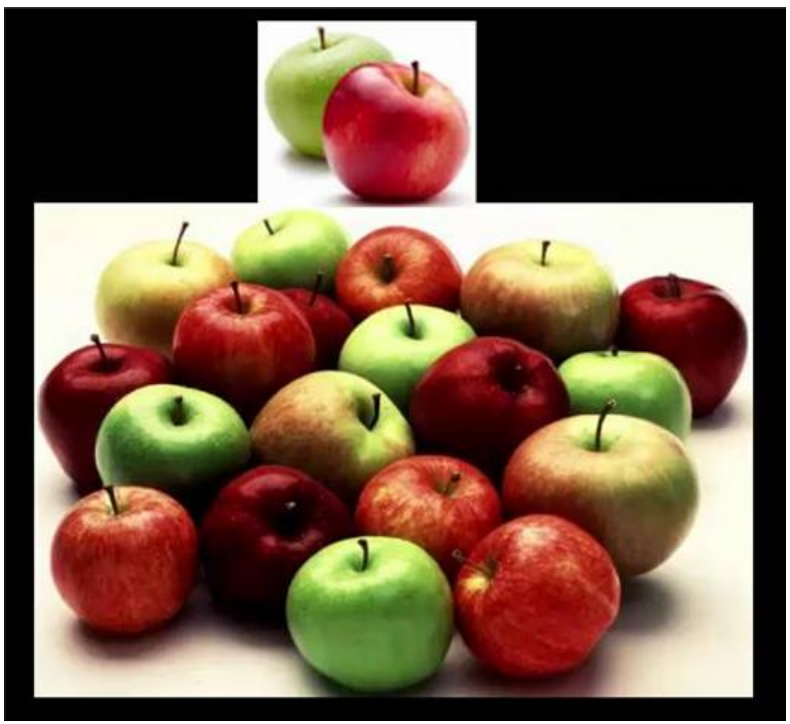
Encode expert knowledge with weights

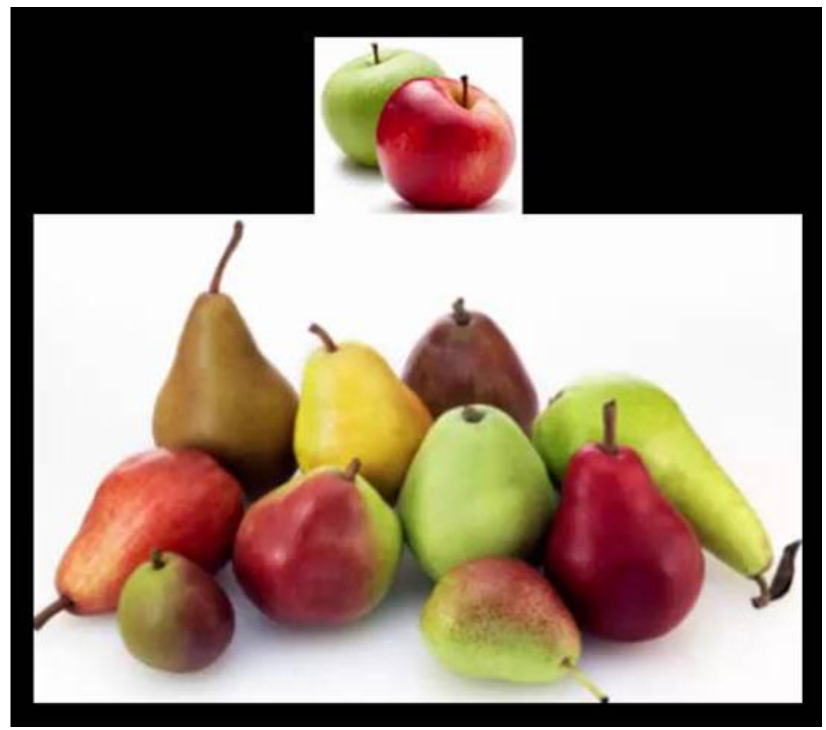
- Changes in one variable should not be more significant only because of differences in magnitudes of values
- To address this, we apply ***min-max scaling*** and map all values to the interval 0…1.

- After scaling to get rid of bias due to units, we can use weights to introduce bias based on expert knowledge of context:
  - 2 families with the same income and number of children are more similar than 2 families living in the same neighborhood
  - Number of children is more important than the number of credit cards

Data-dependent proximity

Two Apples among Apples | are less similar than | Two Apples among Pears



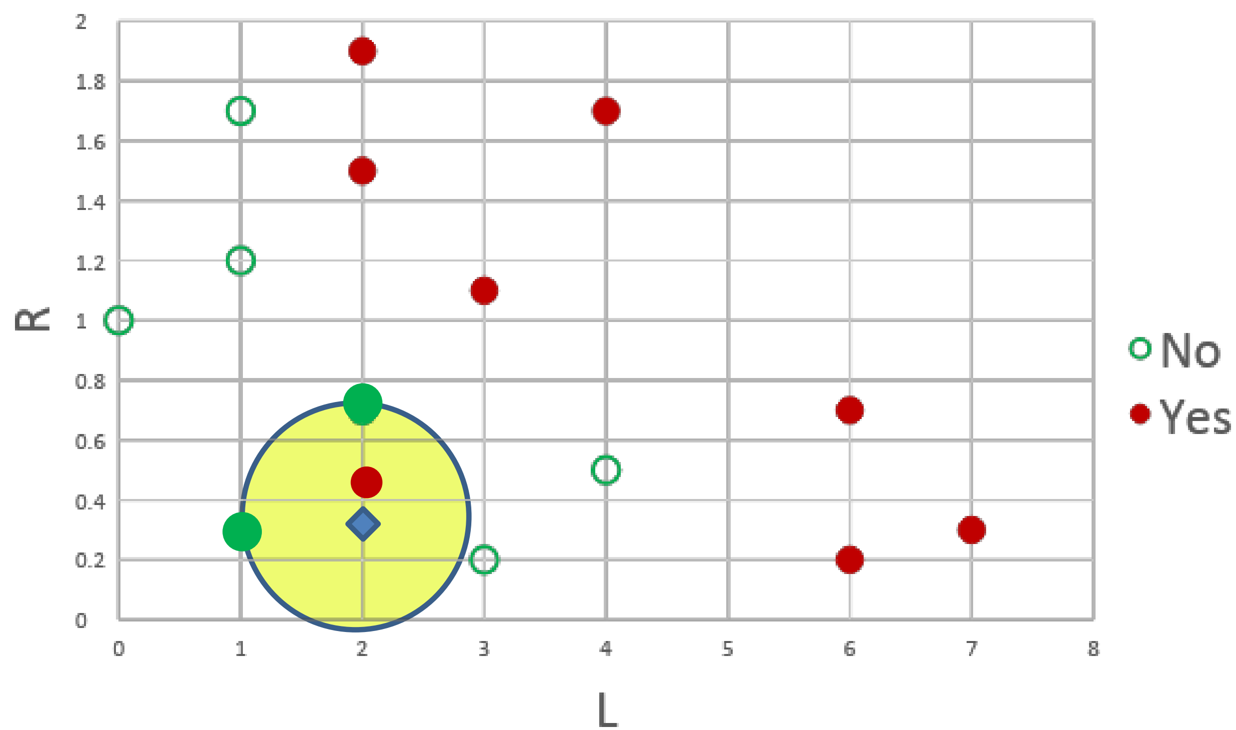


S. Aryal, K. M. Ting, G. Haffari and T. Washio, "Mp-Dissimilarity: A Data Dependent Dissimilarity Measure," *ICDM*, 2014

Combining neighbors' vote

Majority voting (democracy)

L	R
2	0.3



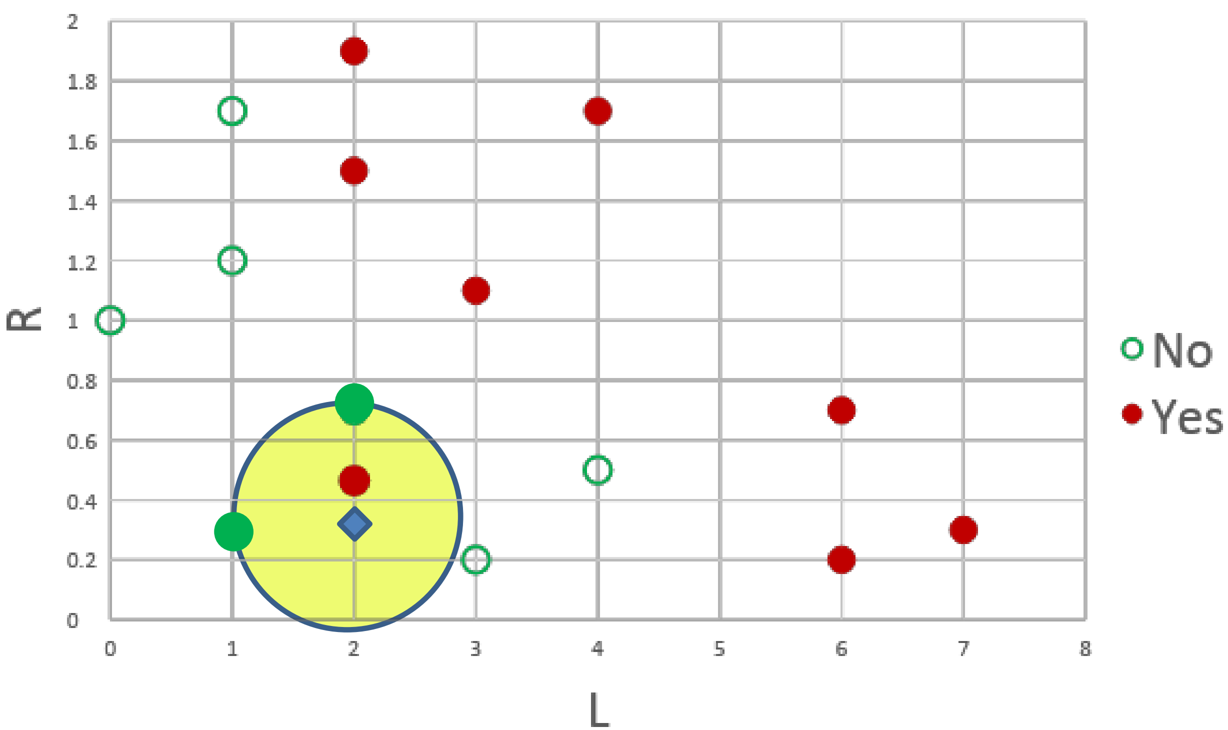
L: #late payments / year
R: expenses / income ratio

Blue diamond is classified as **No** (No bankrupt)

Weighted voting (shareholder democracy)

L	R
2	0.3

The weight of each neighbor is inversely proportional to its distance



1/0.15 Yes+1/1 No + 1/0.5 No=6.7 Yes + 3 No = Yes!

The closest neighbor outweighs the majority class

How to find best value of *K*

Recap: Holdout

- *Holdout procedure*: method of splitting original data into training and test set
  - Dilemma: ideally both training set ***and*** test set should be large!

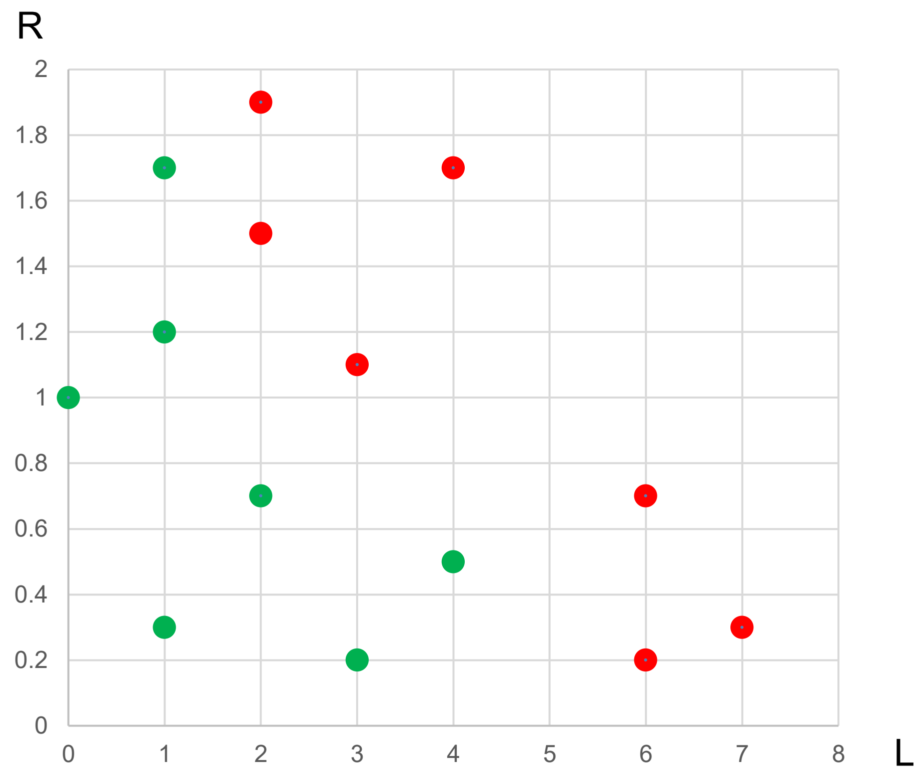
- Holdout reserves a certain amount for testing and uses the remainder for training
  - Usually: 1/3 for testing, the rest for training

- Problem: the samples might not be representative
  - Example: one class might be missing in the test data

- Advanced version uses *stratification*
  - Ensures that each class is represented with approximately equal proportions in both subsets (but what about the attribute values?)

Example: holdout

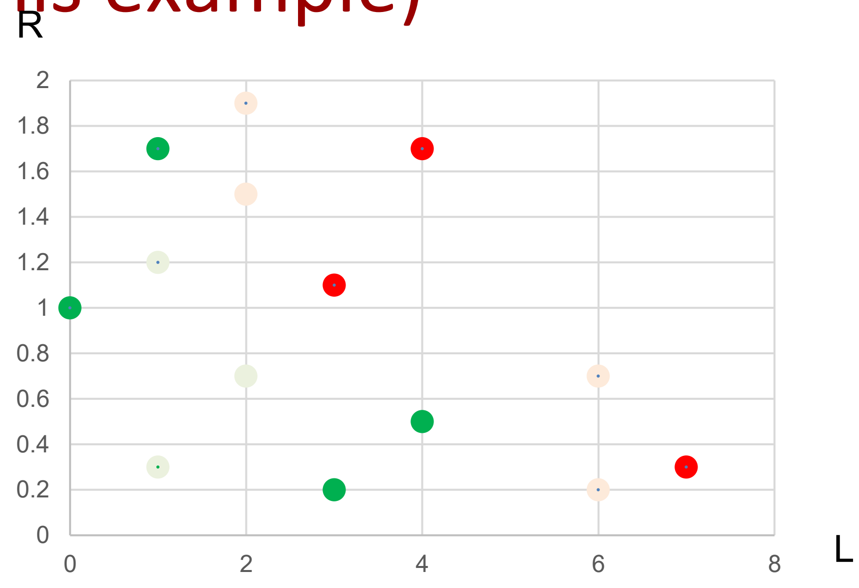
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1.0	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



Split into train and test
(1:1 in this example)

Train set

L	R	B
3	0.2	No
4	0.5	No
0	1.0	No
1	1.7	No
7	0.3	Yes
3	1.1	Yes
4	1.7	Yes



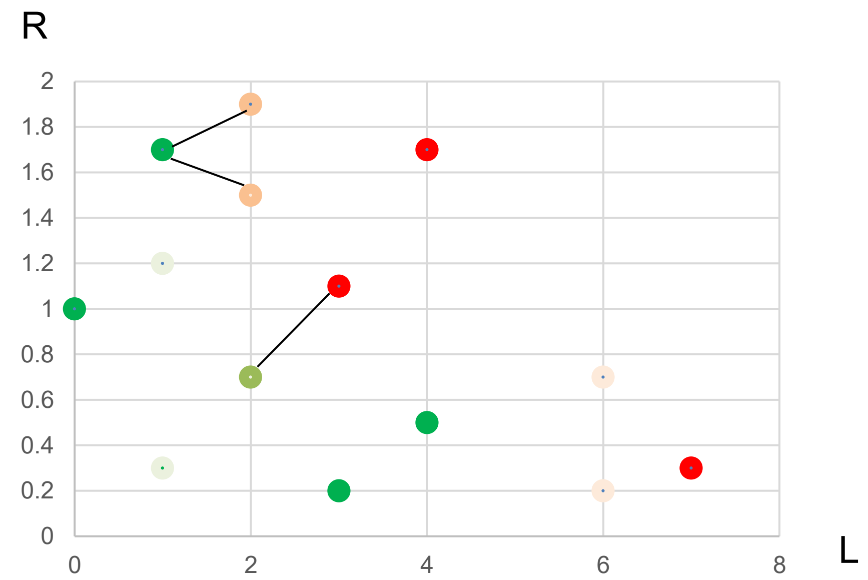
Test set

L	R	B
1	0.3	No
2	0.7	No
1	1.2	No
6	0.2	Yes
6	0.7	Yes
2	1.5	Yes
2	1.9	Yes

Predict test data using train model (K=1)

Train set

L	R	B
3	0.2	No
4	0.5	No
0	1.0	No
1	1.7	No
7	0.3	Yes
3	1.1	Yes
4	1.7	Yes



Test set

L	R	B	Pred	
1	0.3	No	No	
2	0.7	No	Yes	x
1	1.2	No	No	
6	0.2	Yes	Yes	
6	0.7	Yes	Yes	
2	1.5	Yes	No	x
2	1.9	Yes	No	x

Error rate: 3/7

Success rate: 4/7

Recap: cross-validation

k-fold cross-validation

- *Cross-validation* avoids overlapping test sets
  - **First step:** split data into *k* subsets of ~equal size
  - **Second step:** use each subset in turn for testing, the remainder for training

- Often the subsets are stratified before the cross-validation is performed
- The error estimates are averaged to yield an overall error estimate

- Standard method: ***stratified 10-fold cross-validation***

Leave-One-Out cross-validation

- Leave-One-Out: an extreme form of cross-validation
  - Set number of folds to **number of training instances:**

  for *n* training instances, build classifier *n* times using n-1 instances for training, and record the error rate of the left-out instance

✓ Makes best use of data

✓ Involves no random subsampling

❖ But - computationally expensive

Back to:

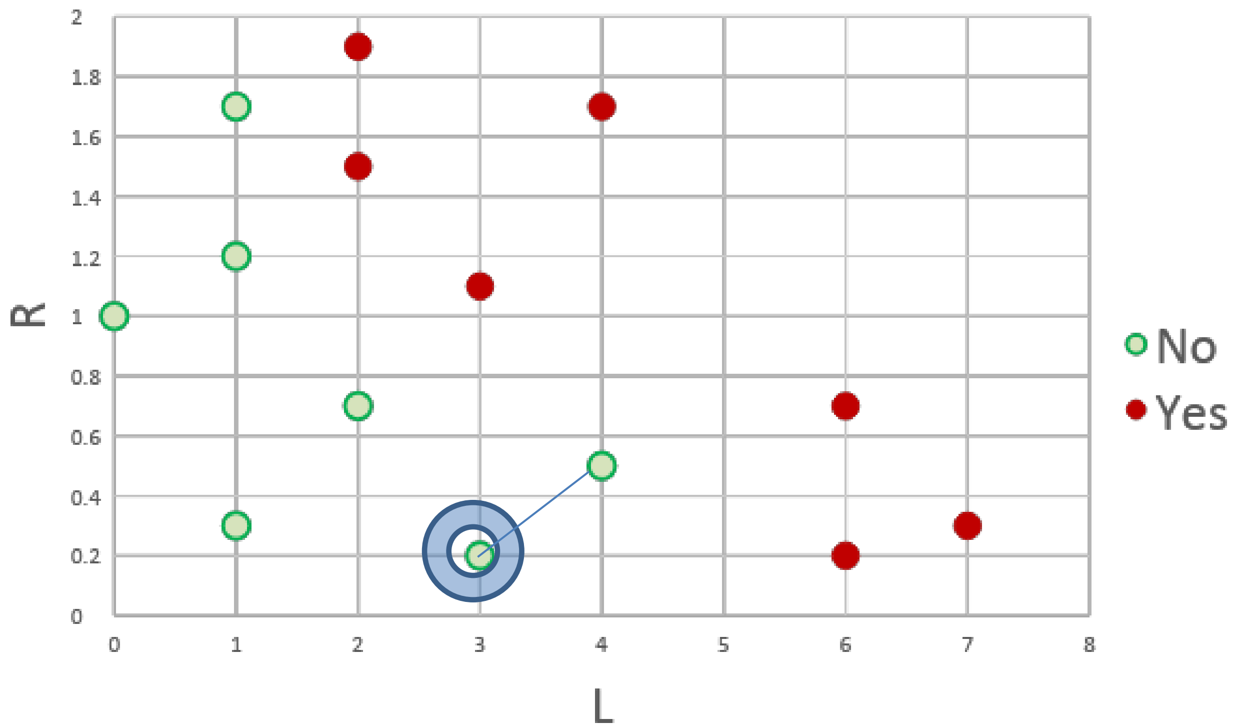
Choosing optimal value of *K*

How many neighbors? application-dependent

- Vary K from 1 to N (odd numbers)
- Use cross-validation to find optimal value of K

Leave-one-out cross validation: K=1

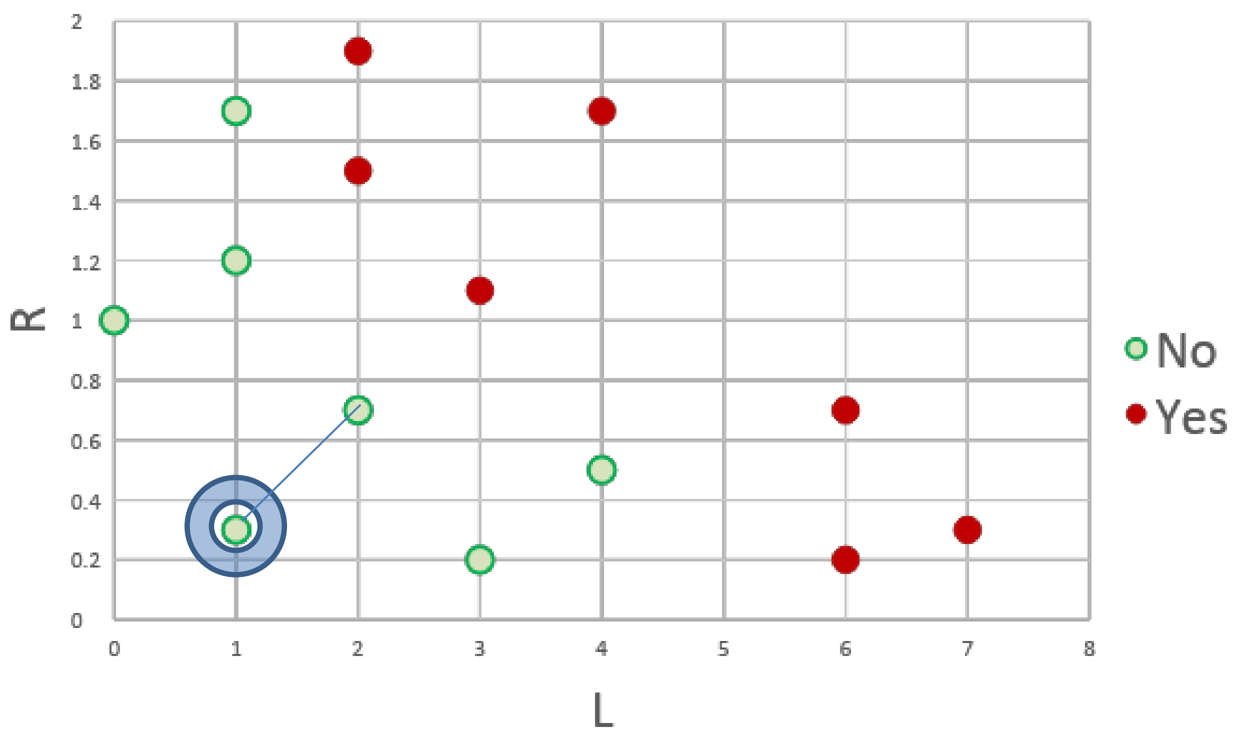
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

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

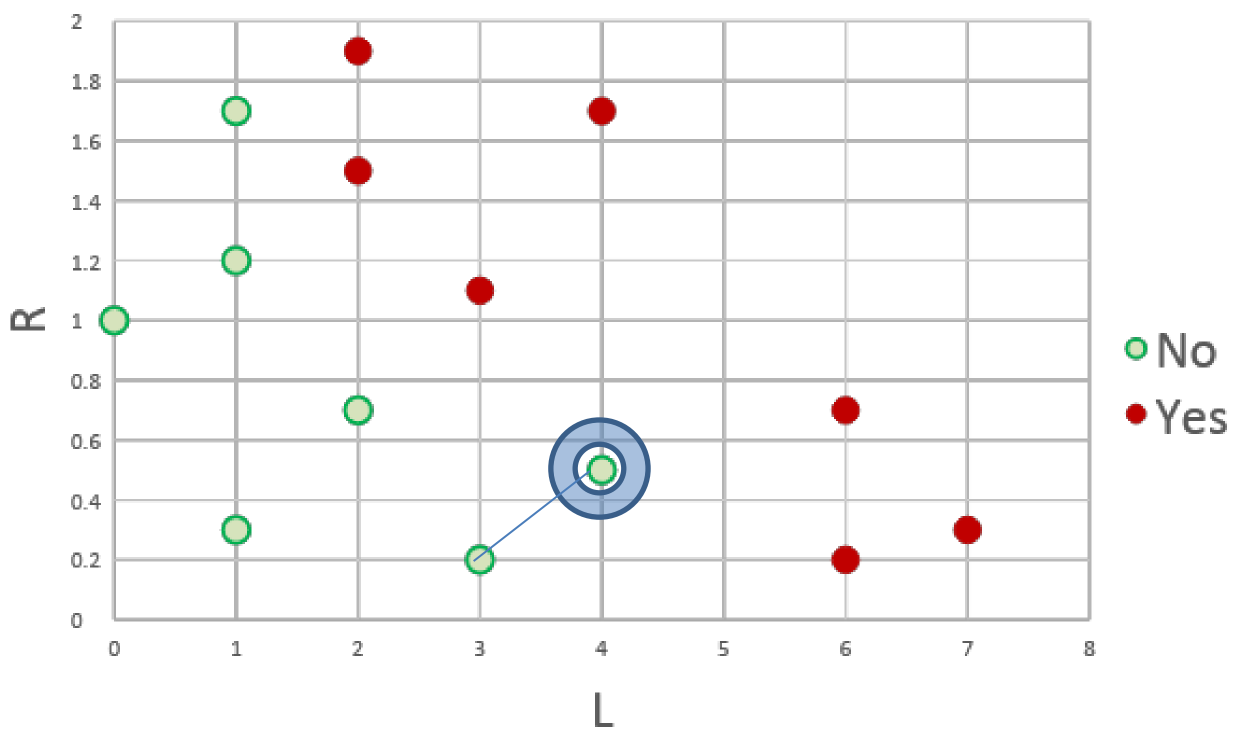
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: K=1

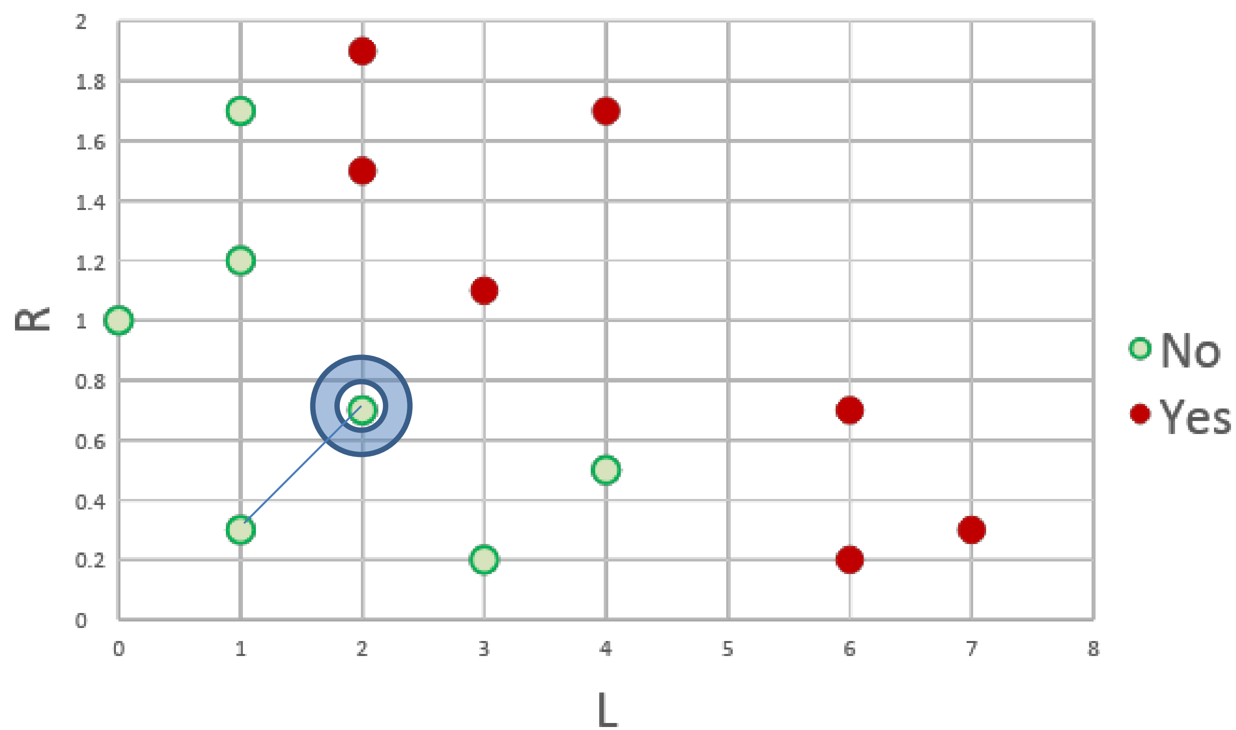
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

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

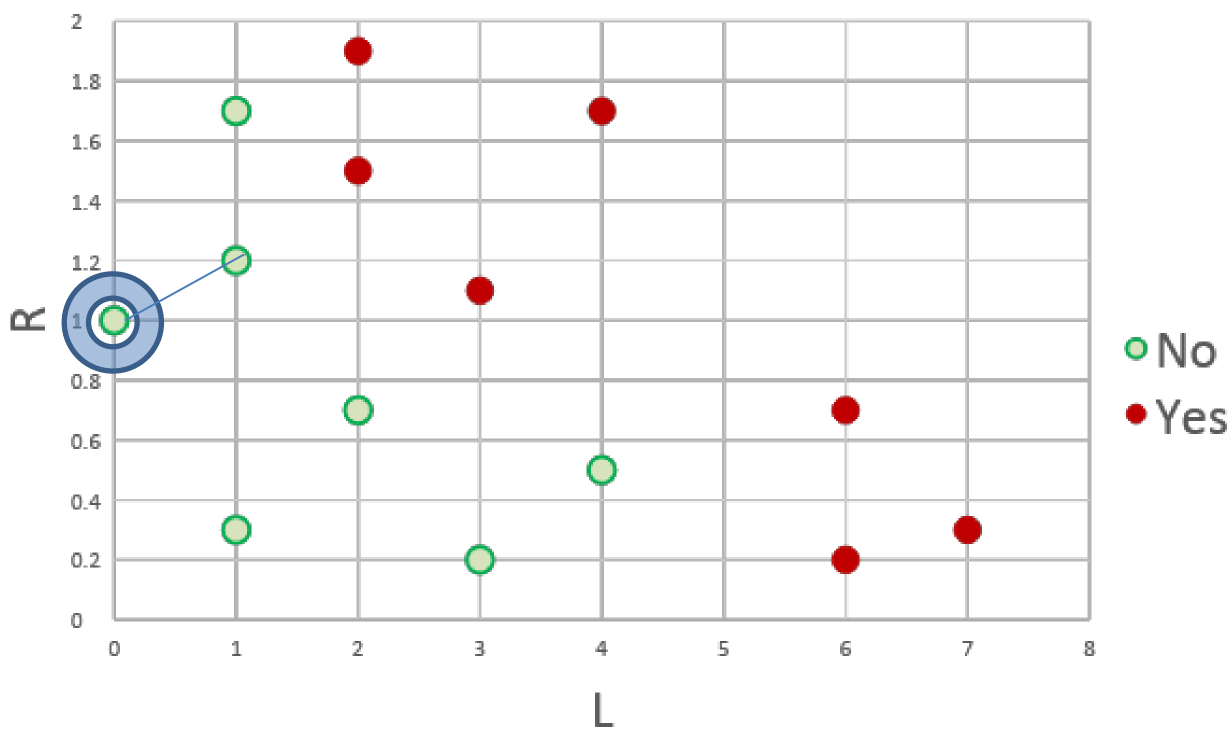
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

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

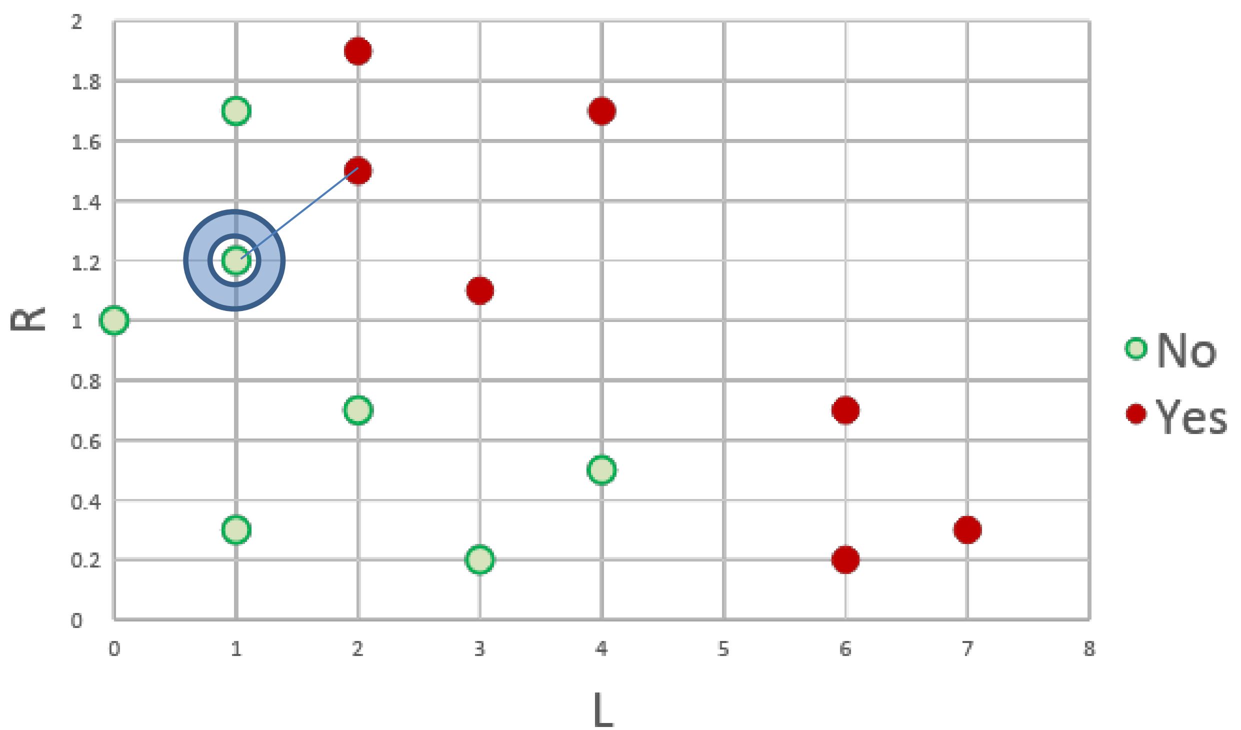
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: K=1

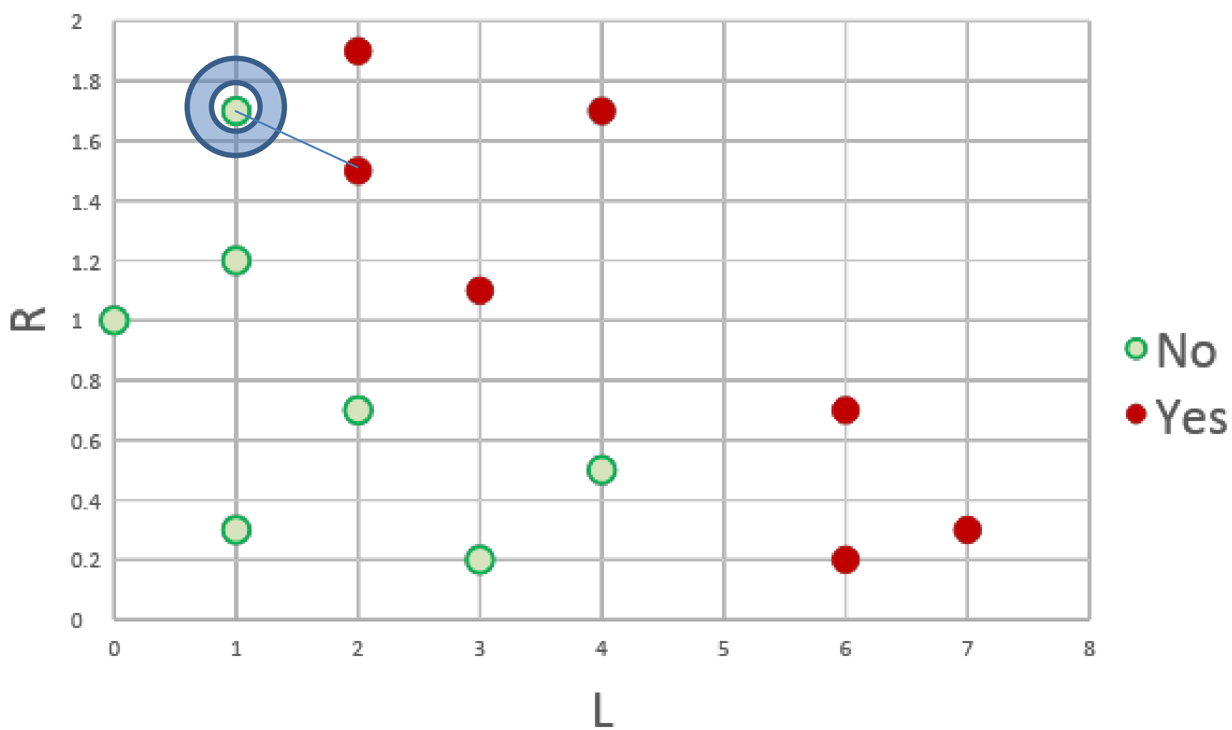
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

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

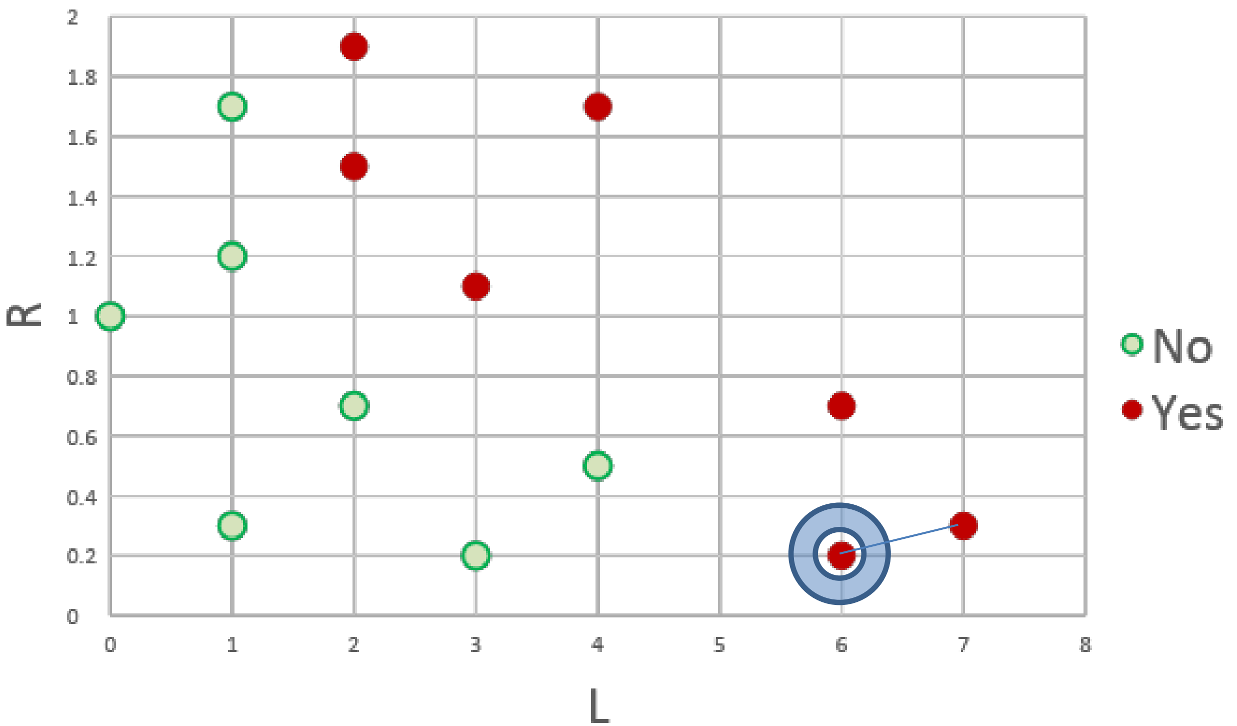
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: K=1

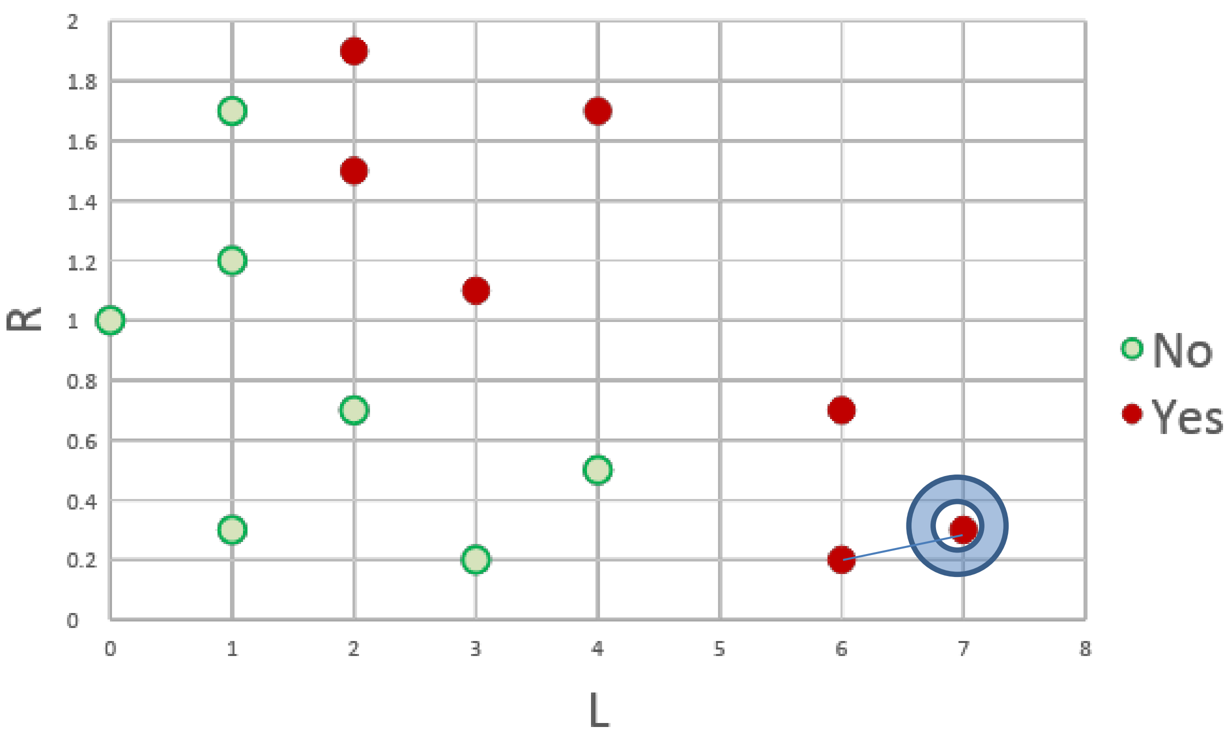
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

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

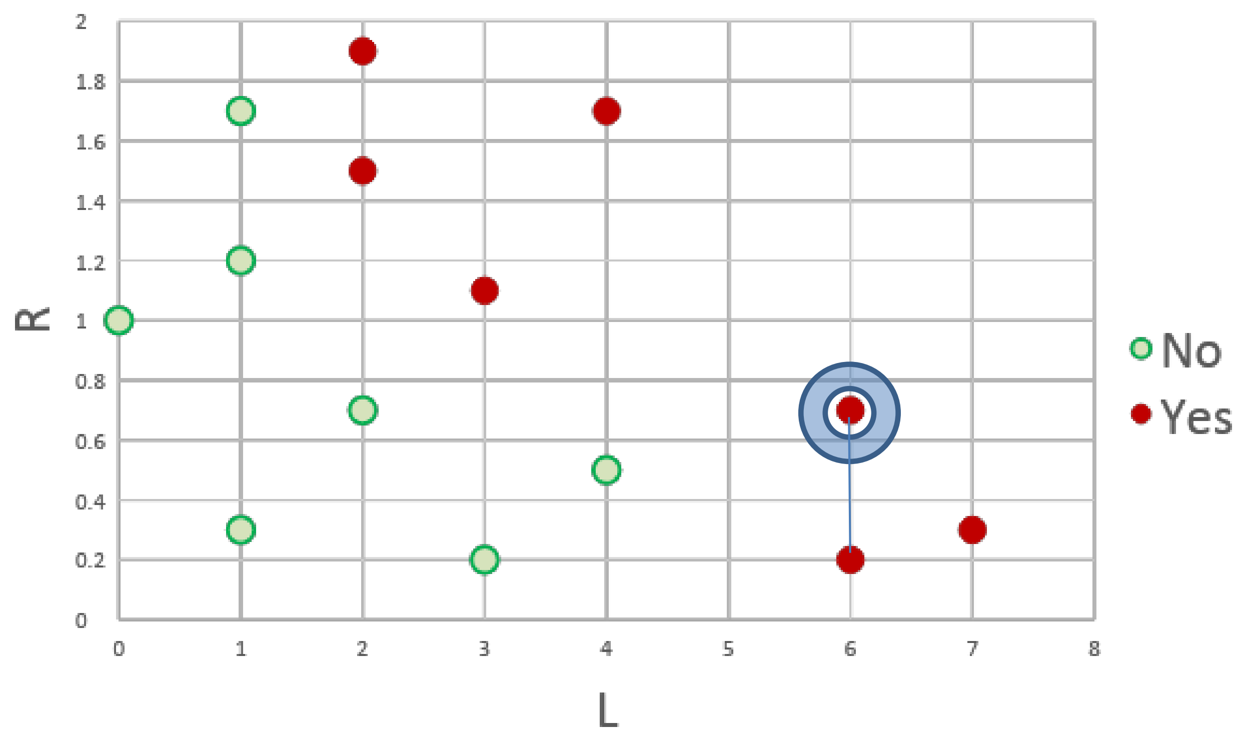
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

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

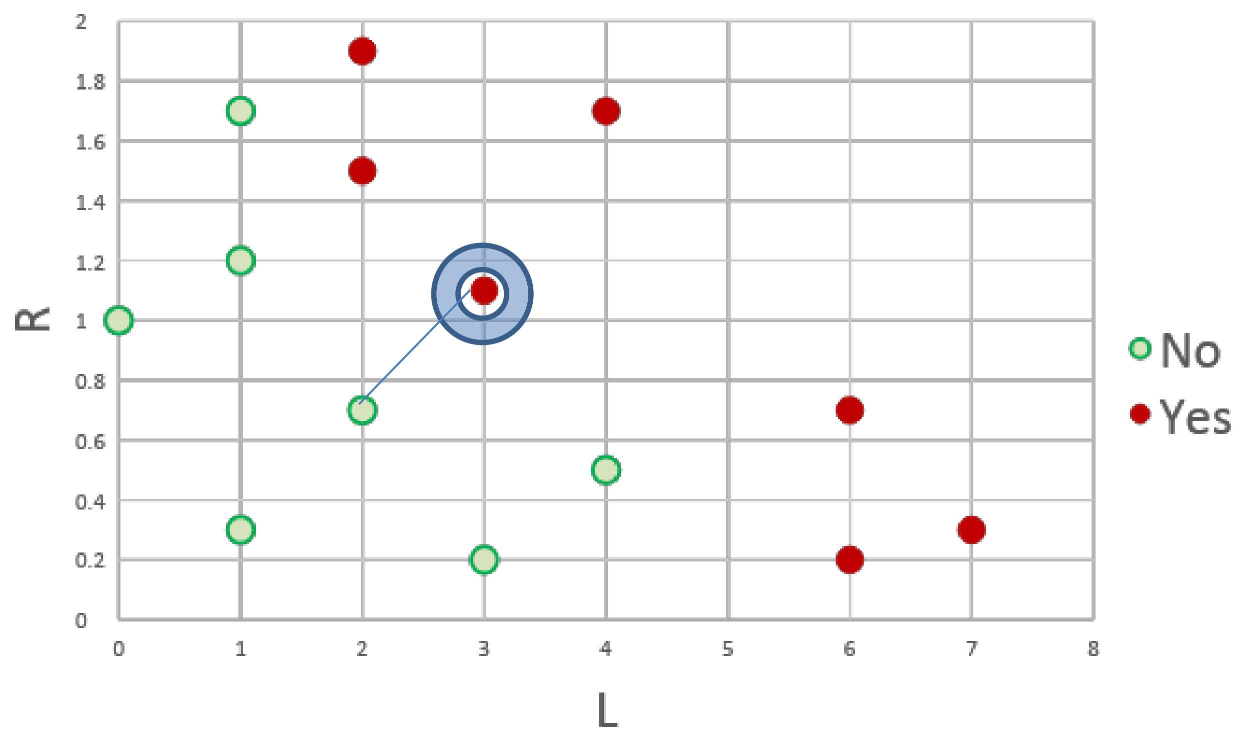
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

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

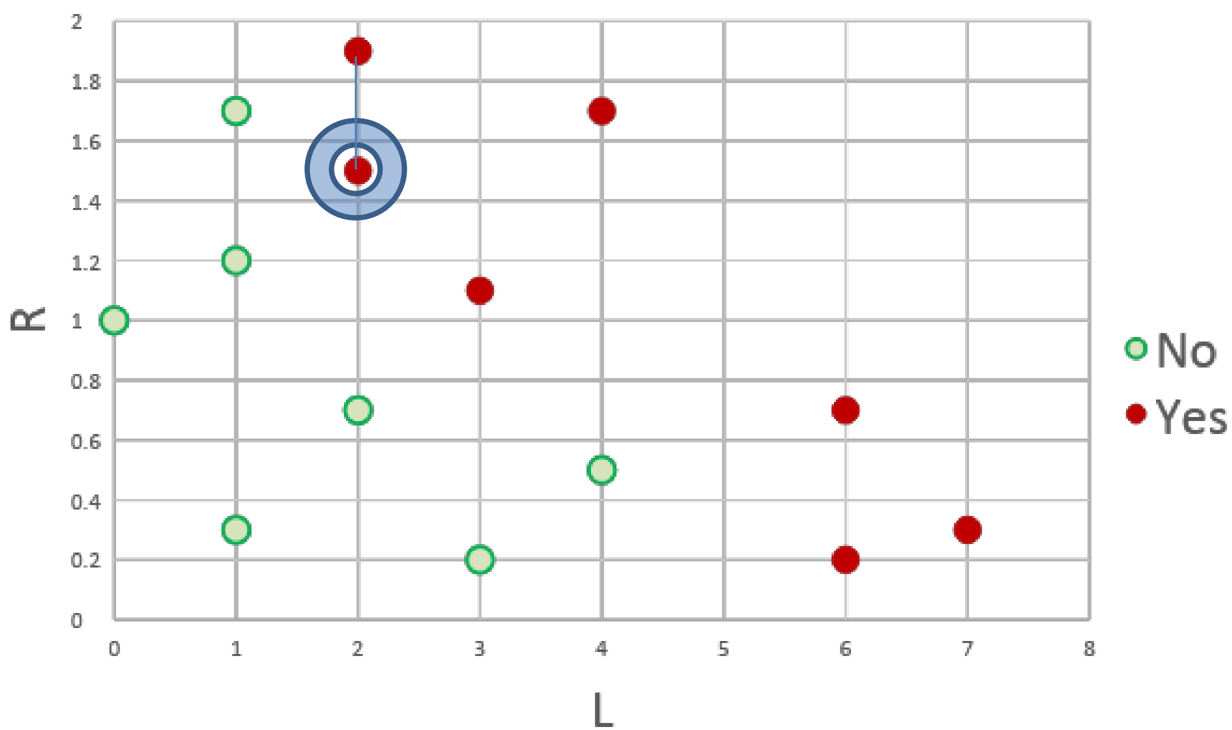
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: K=1

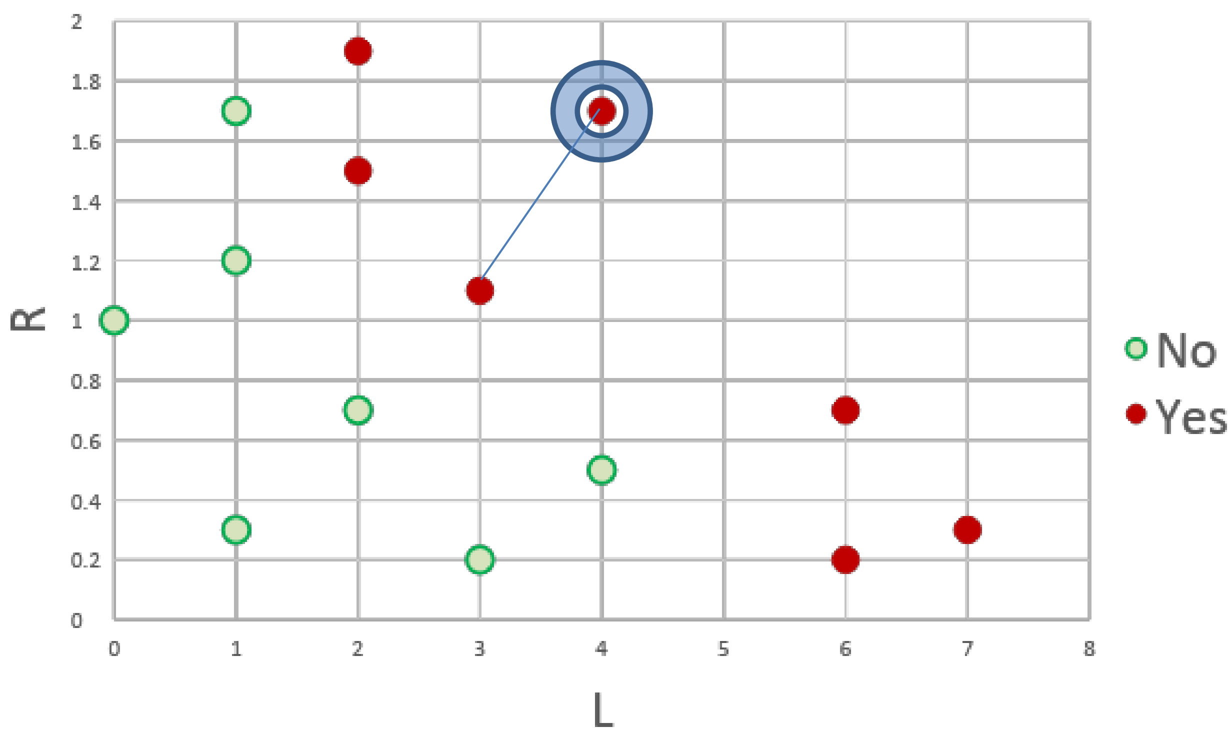
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

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

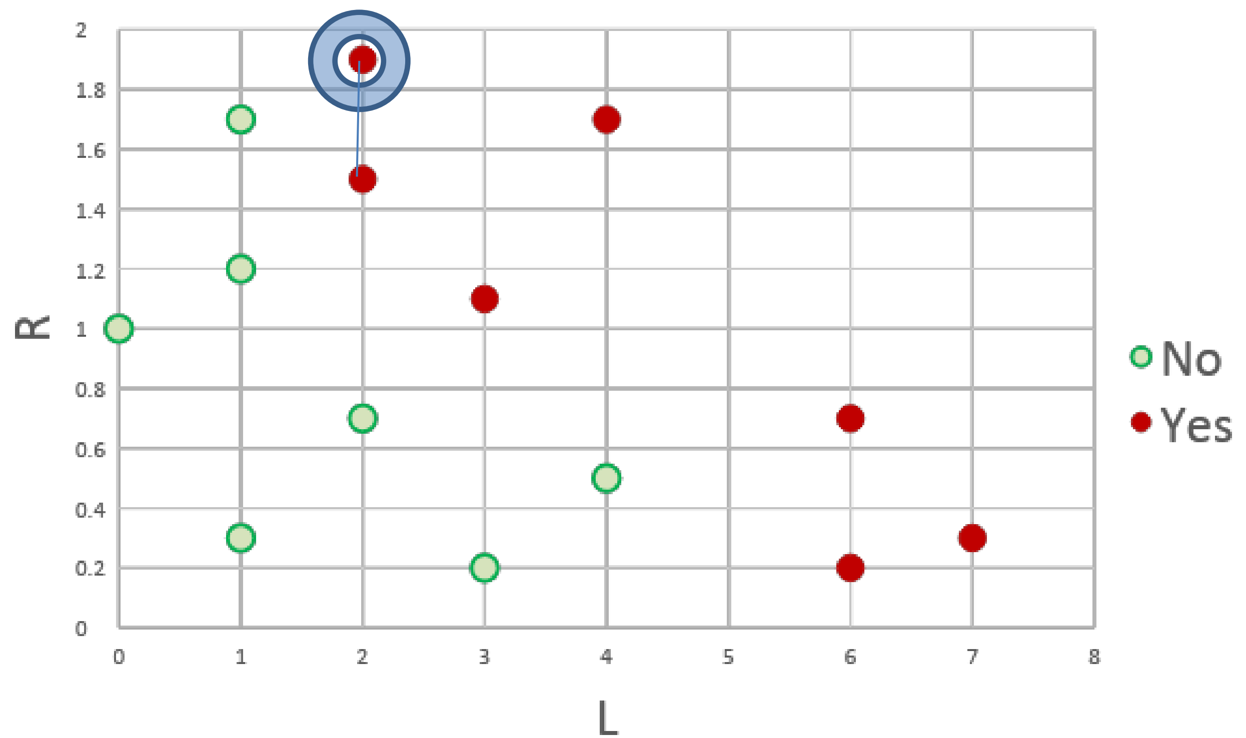
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

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

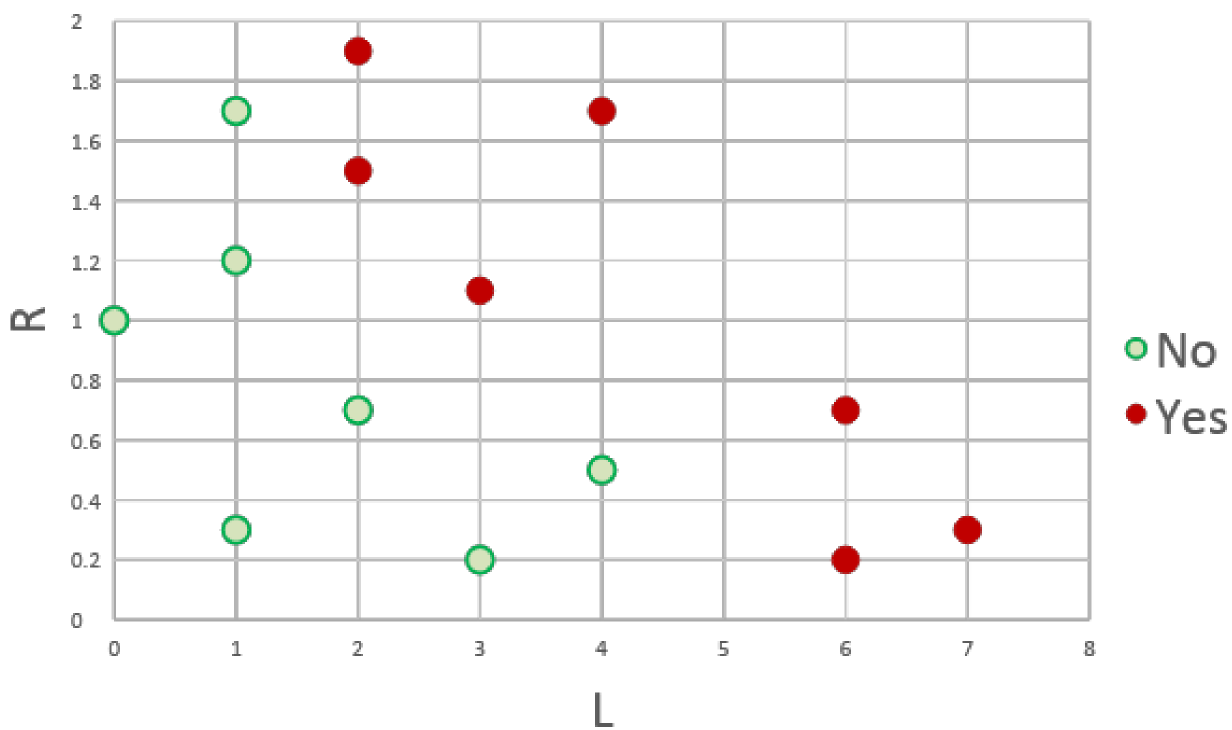
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

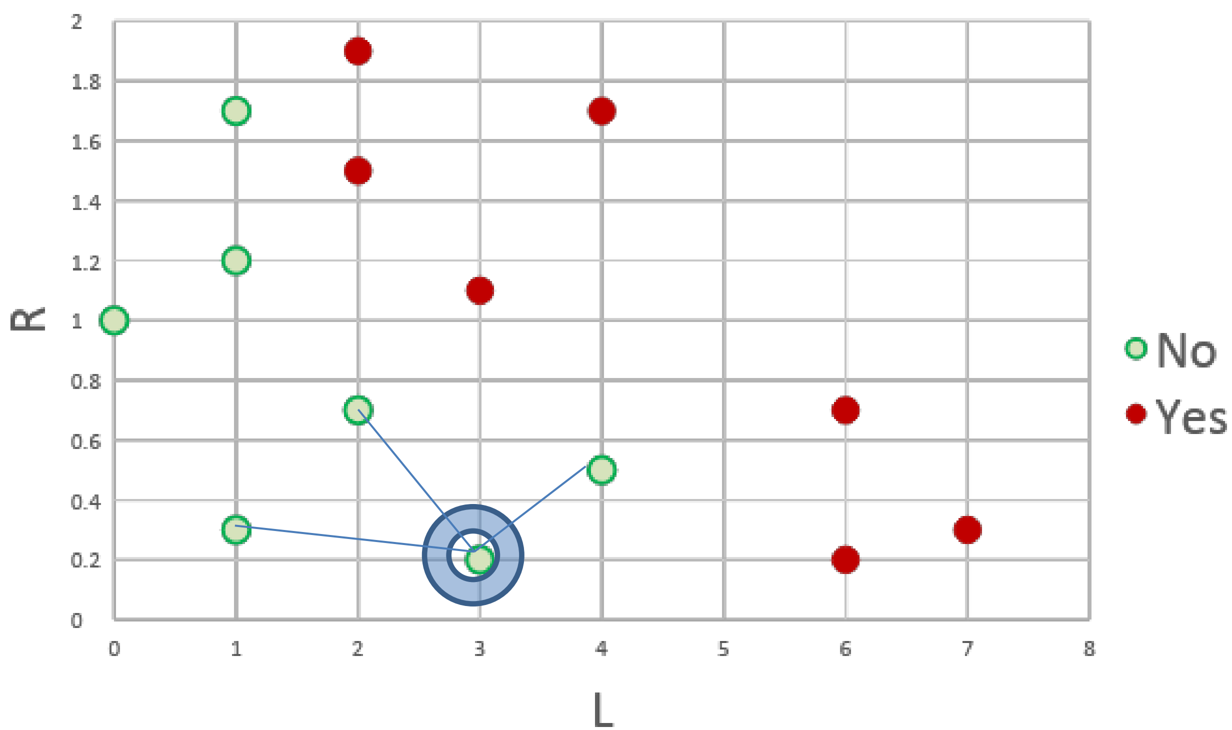


For *K*=1:

Error rate 3/14

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

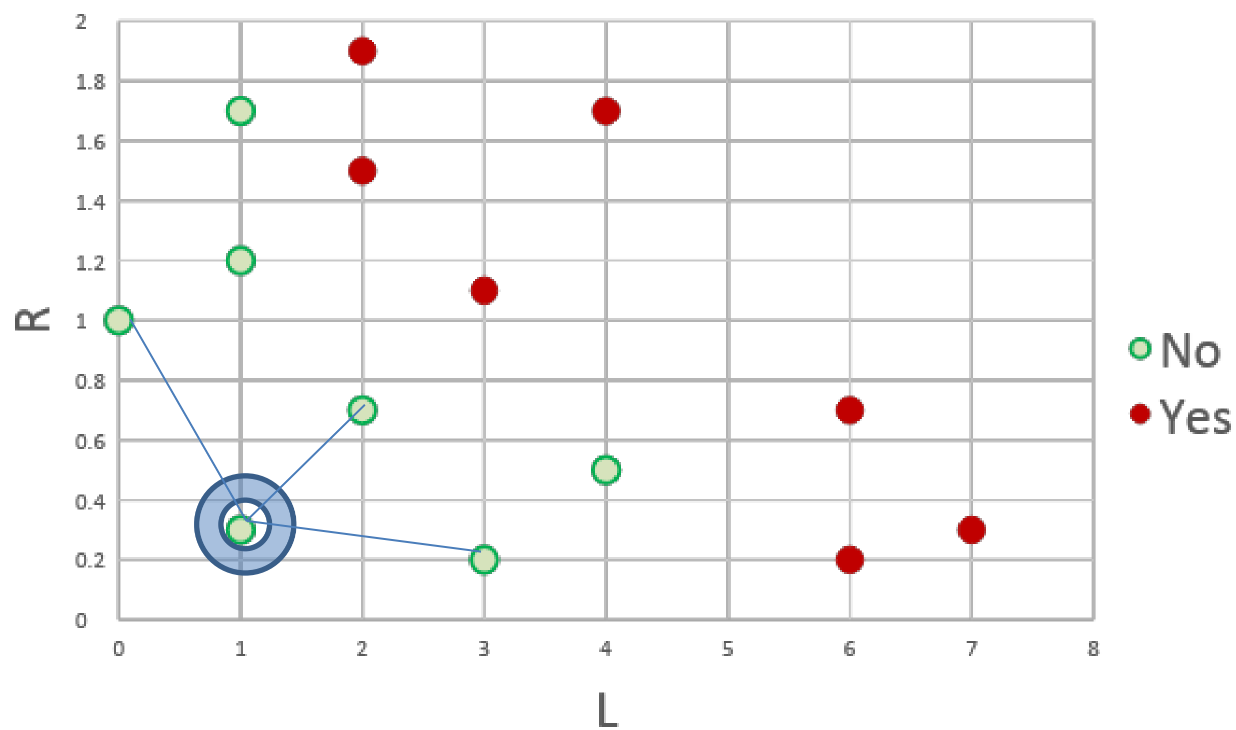
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

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

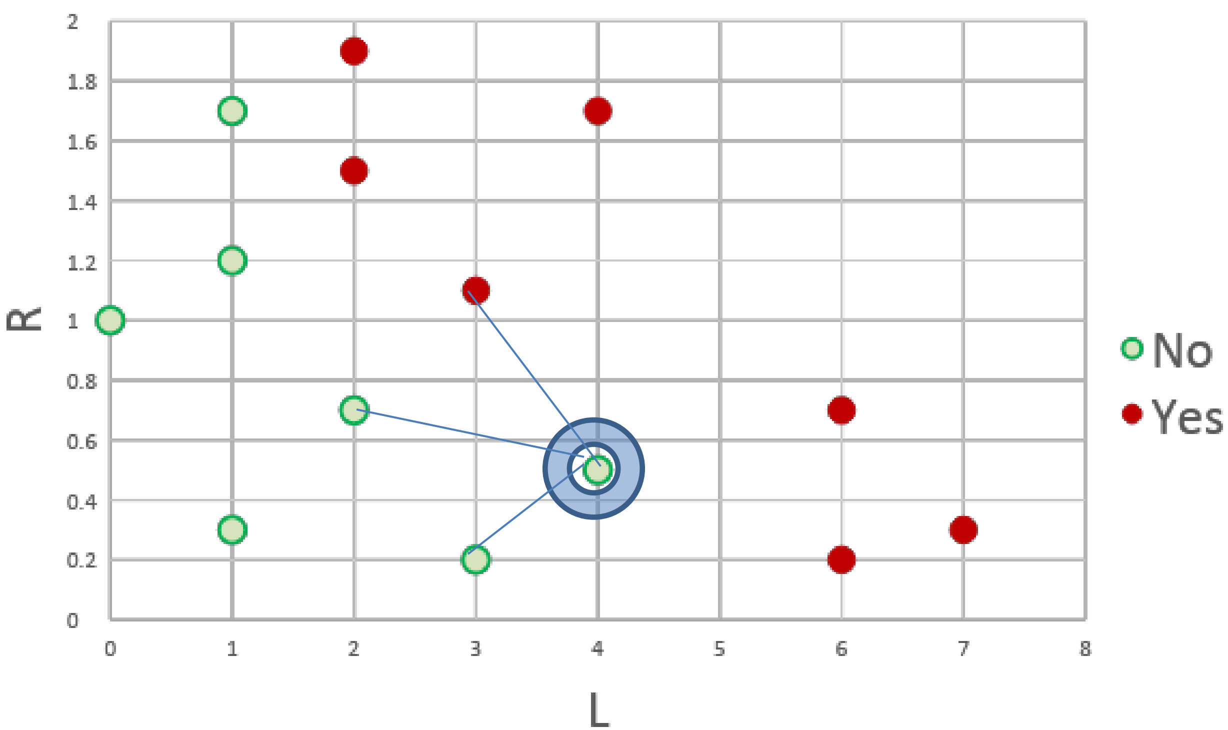
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

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

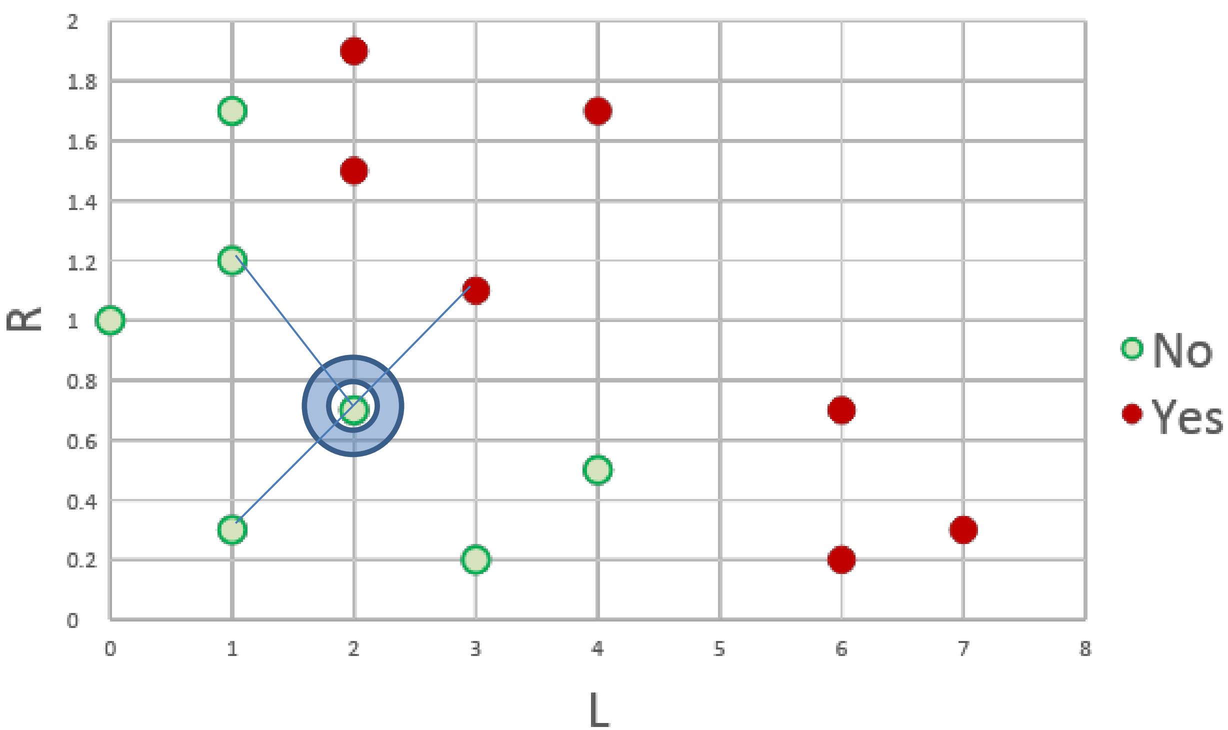
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

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

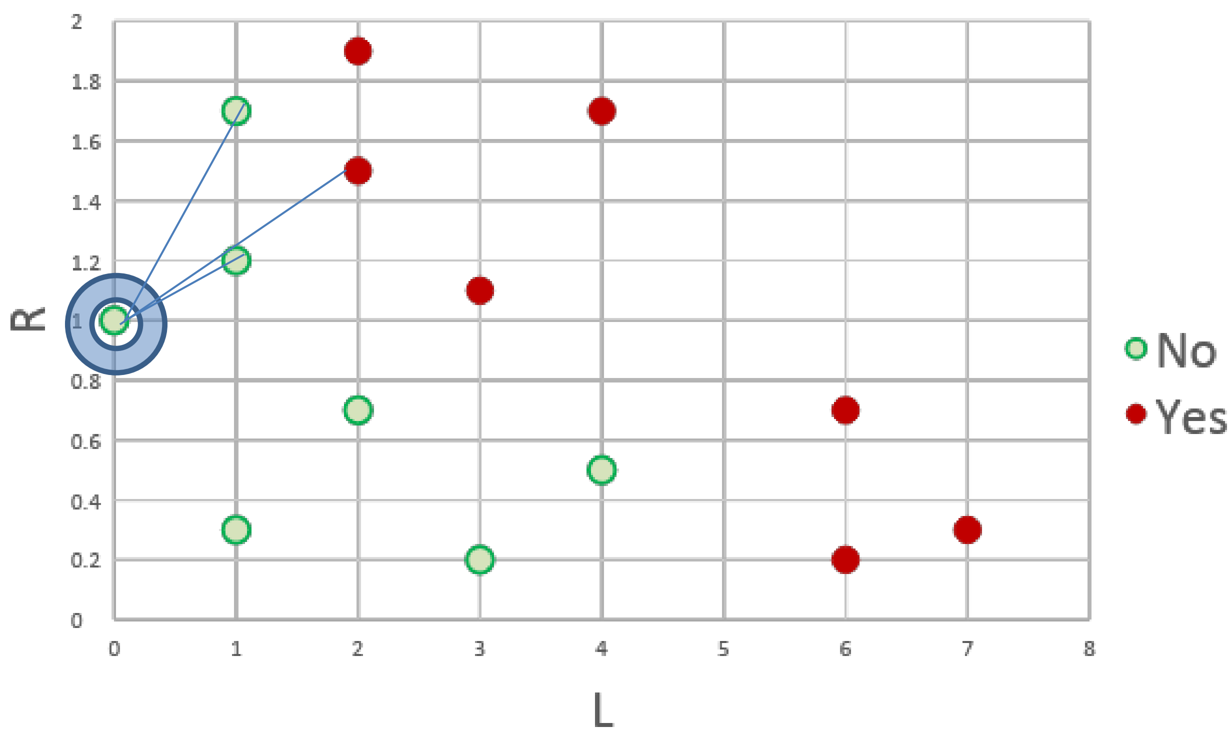
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

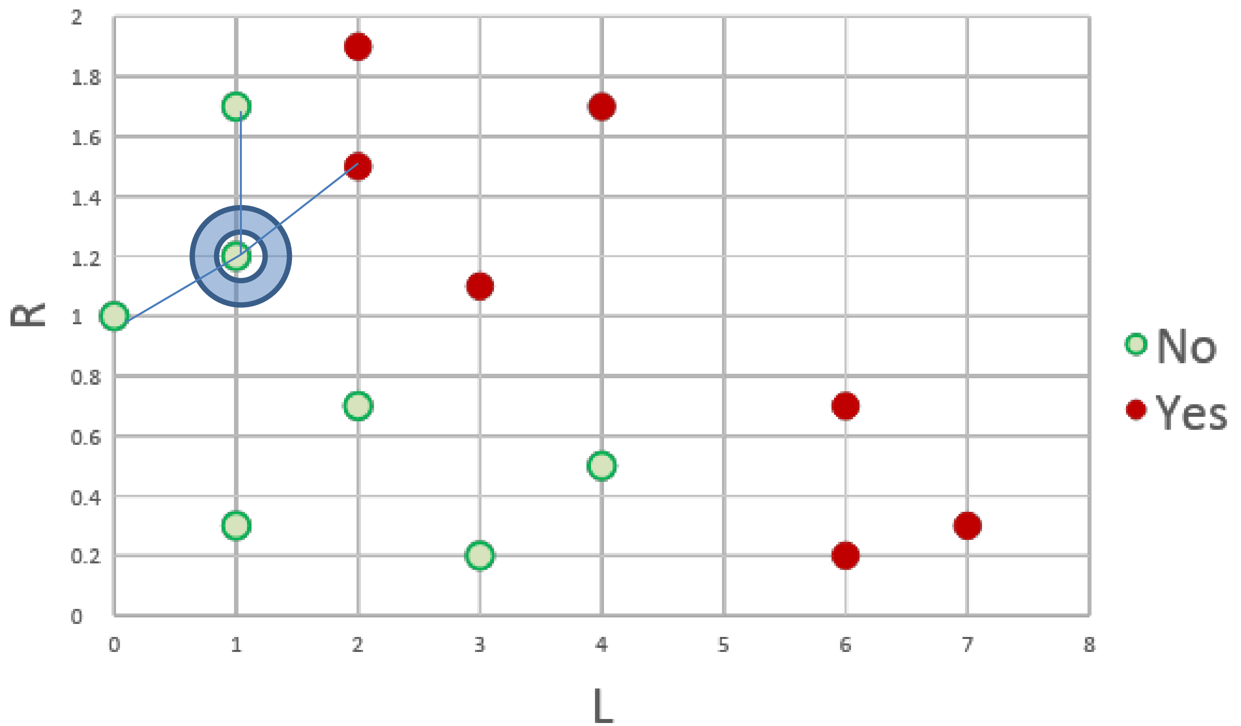


L: #late payments / year

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

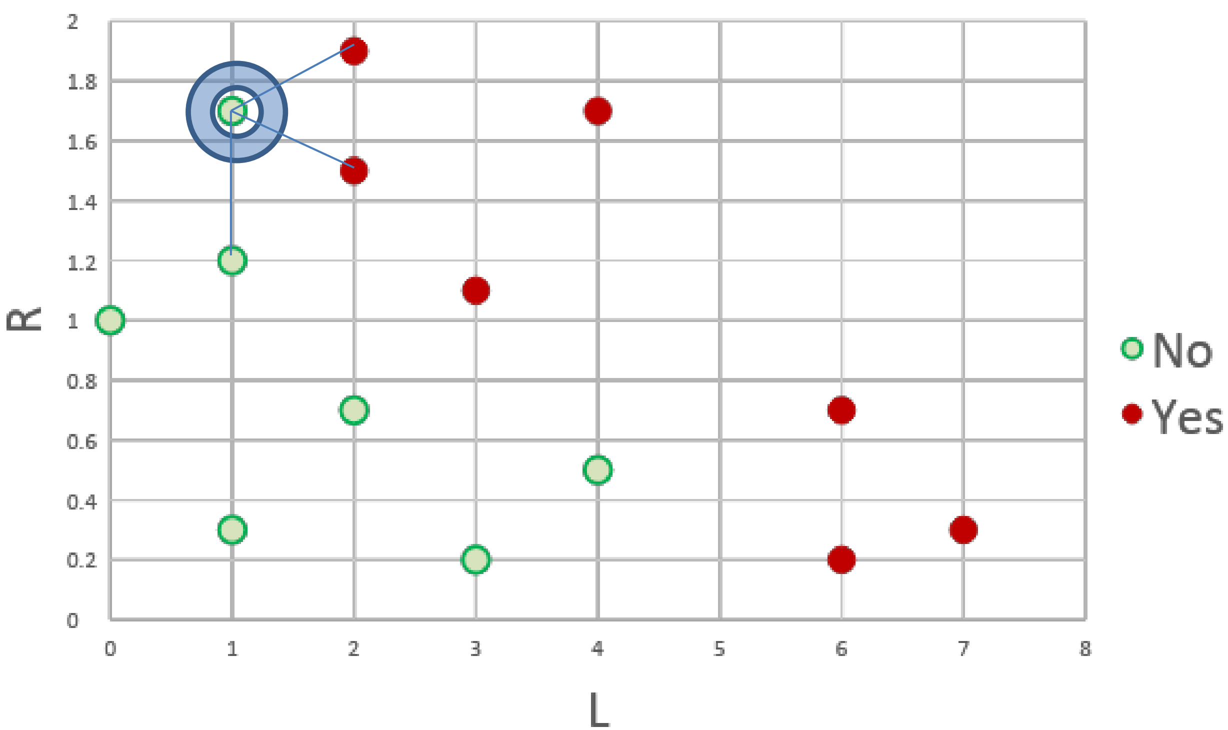


L: #late payments / year

R: expenses / income ratio

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

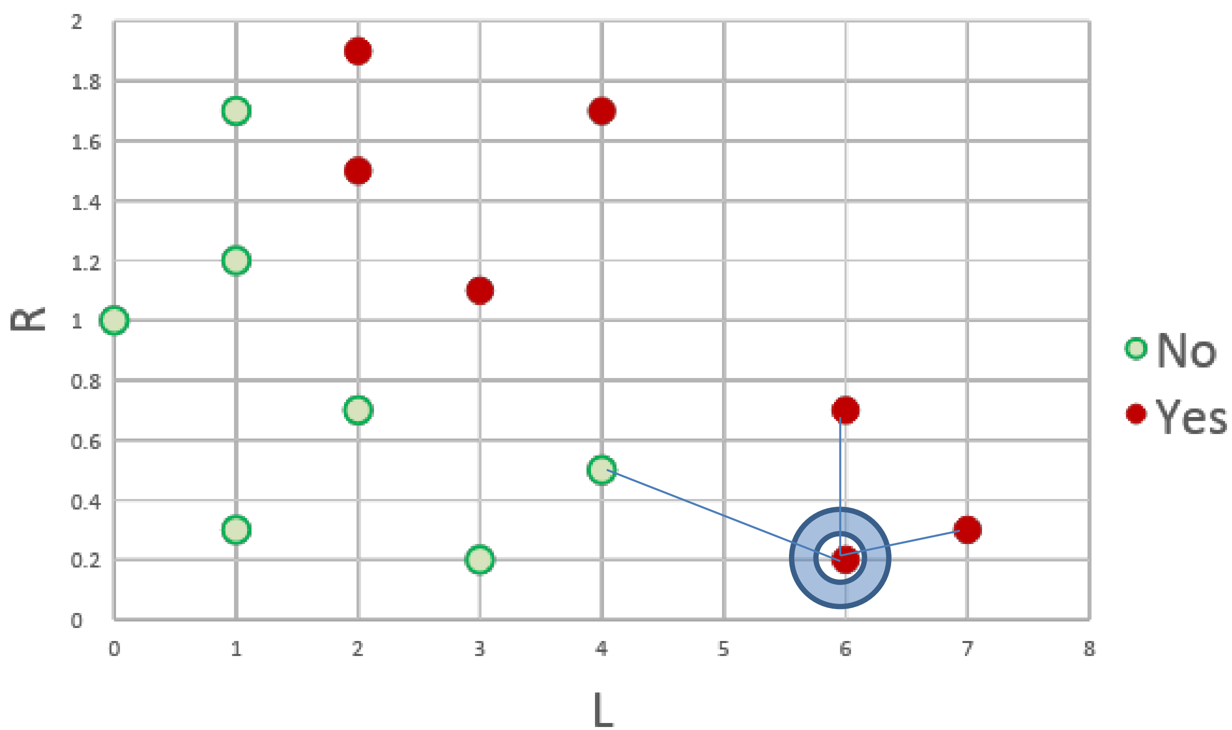
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: K=3

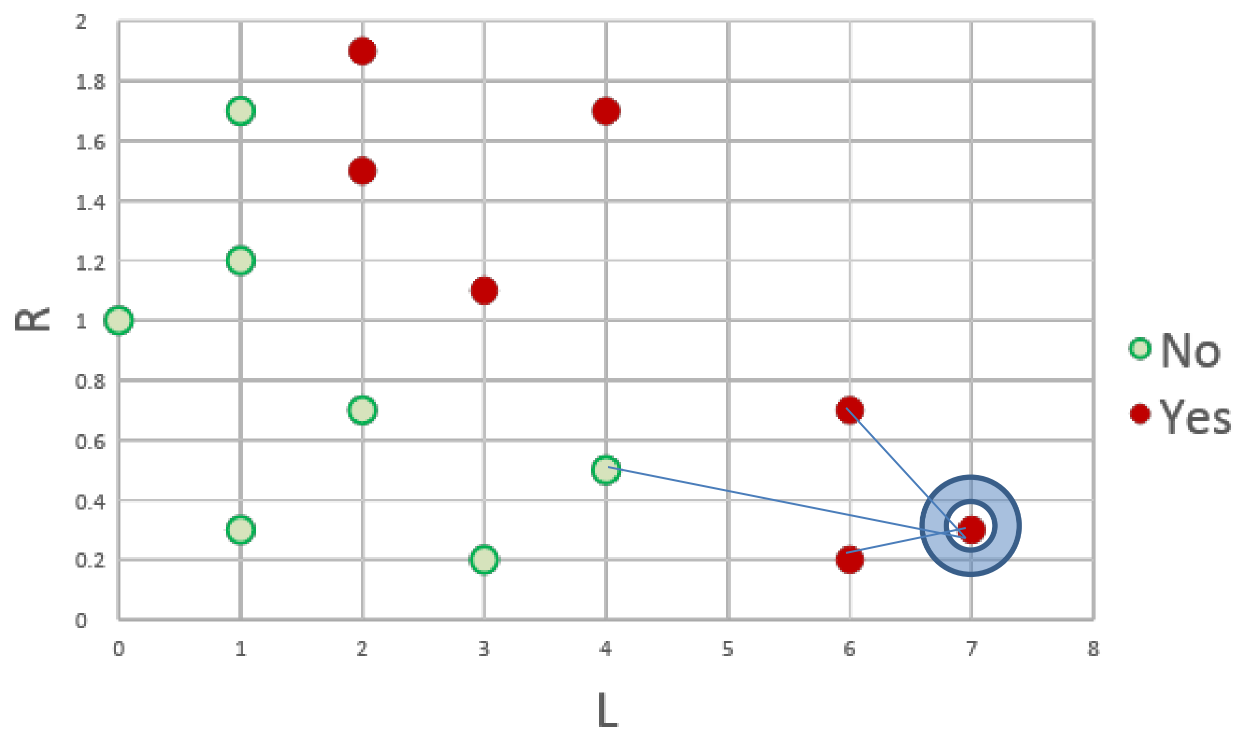
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

Leave-one-out cross validation: K=3

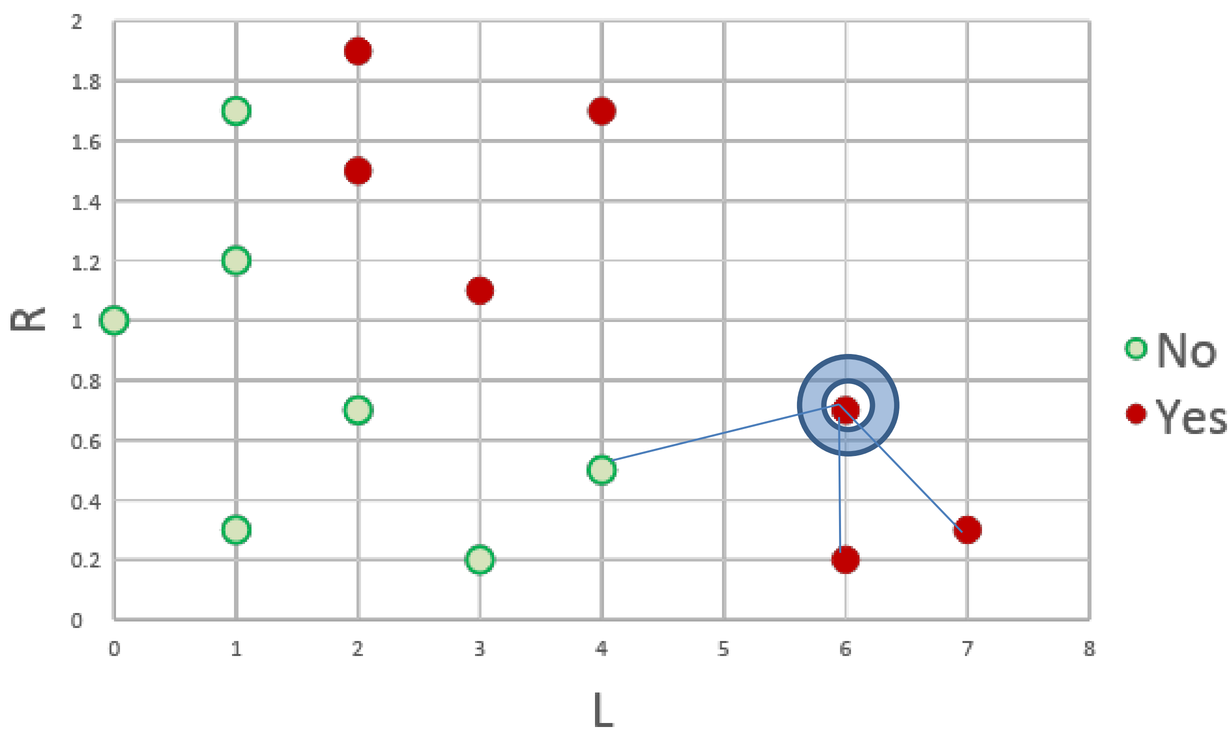
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

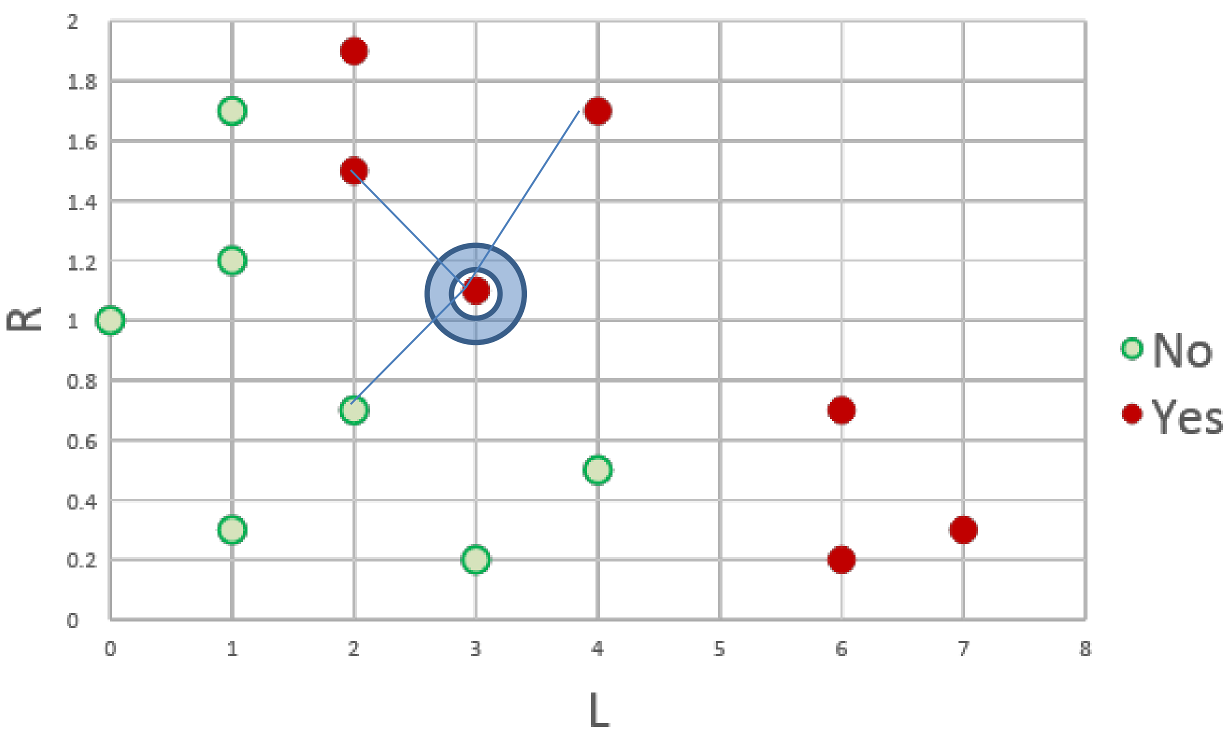


L: #late payments / year

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

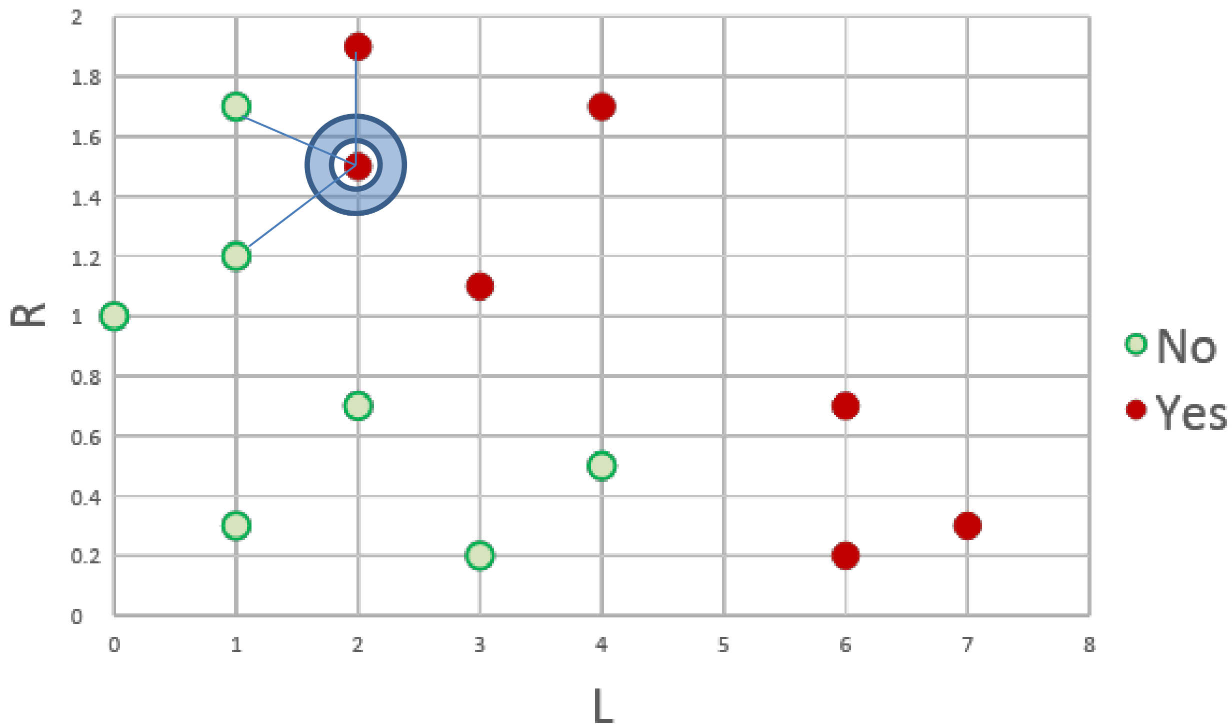


L: #late payments / year

R: expenses / income ratio

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

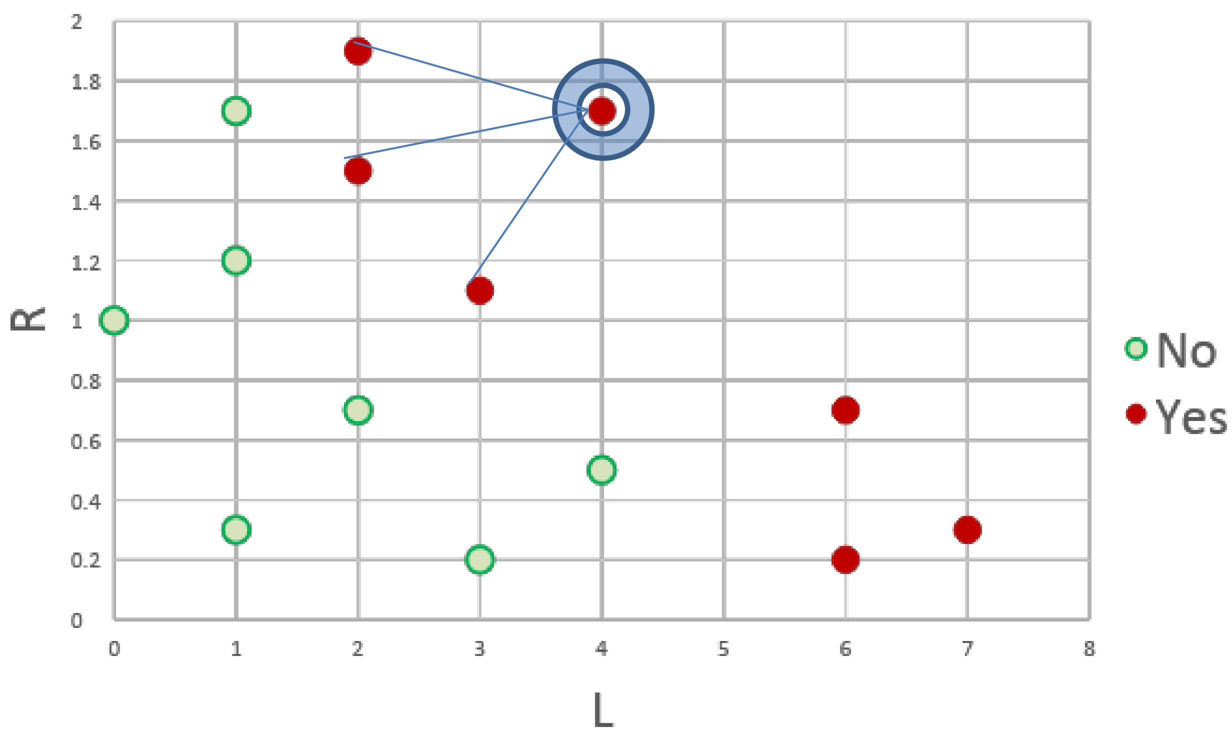
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

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

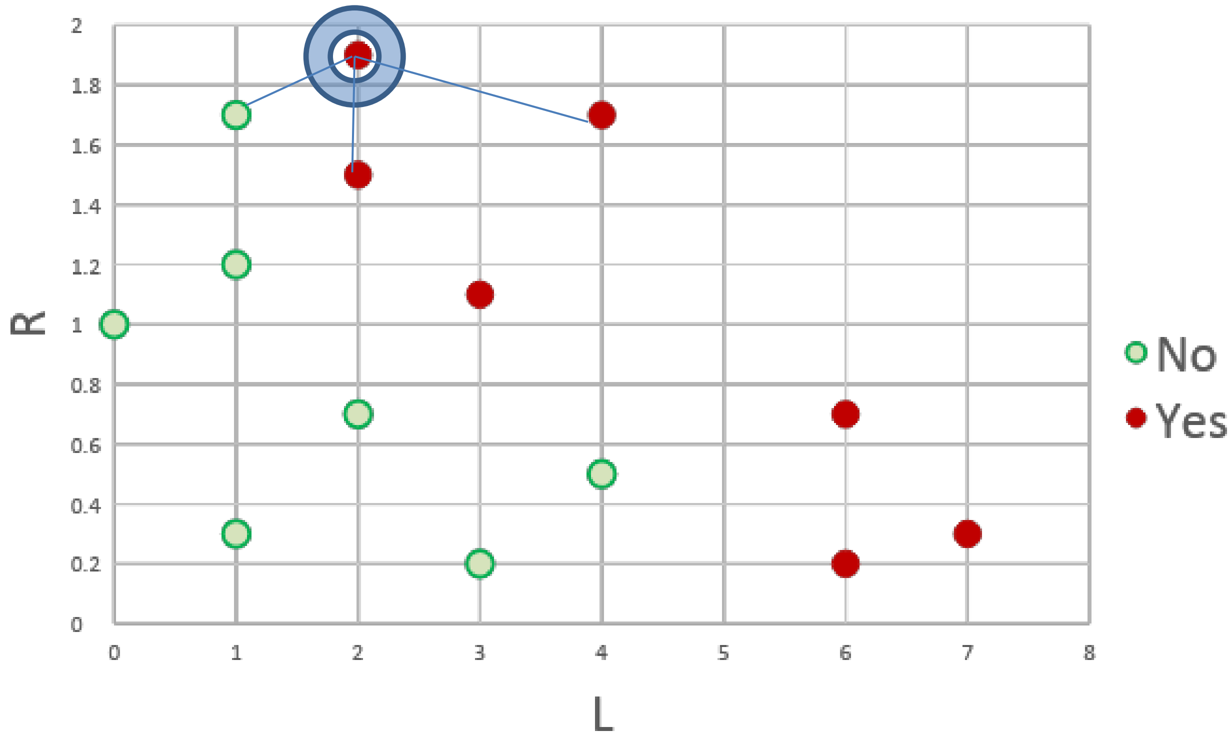
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



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

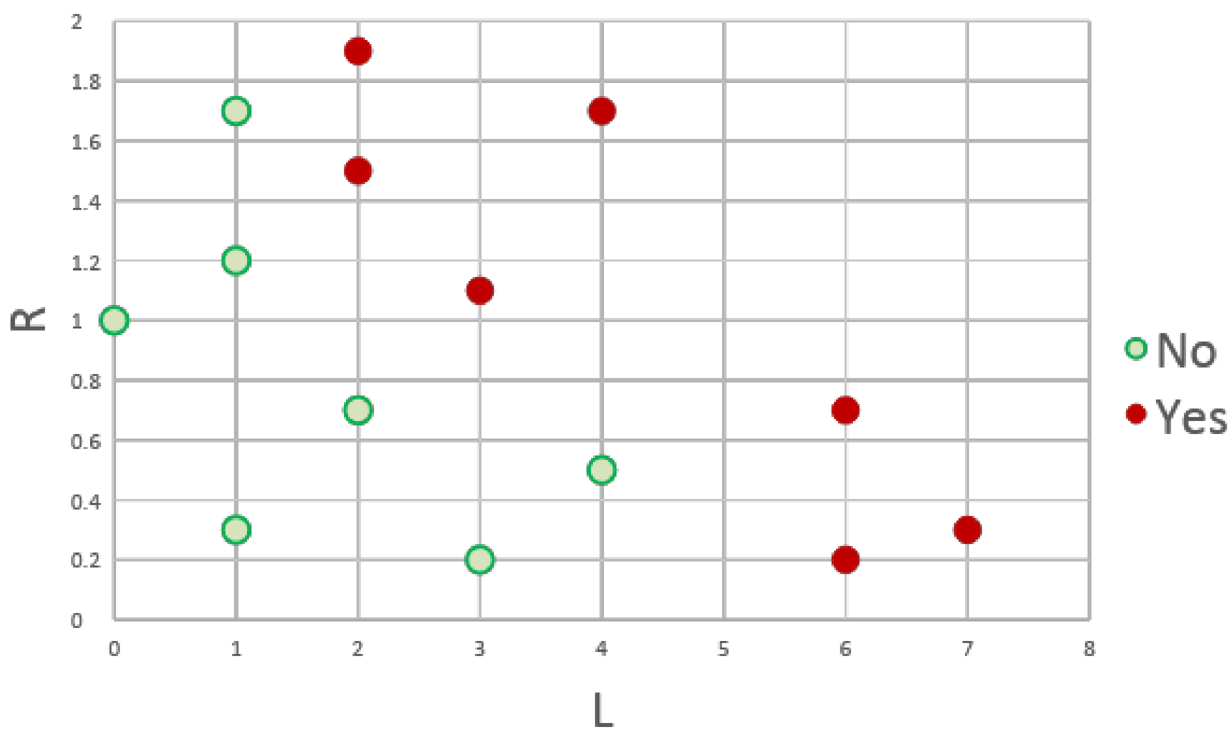


L: #late payments / year

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

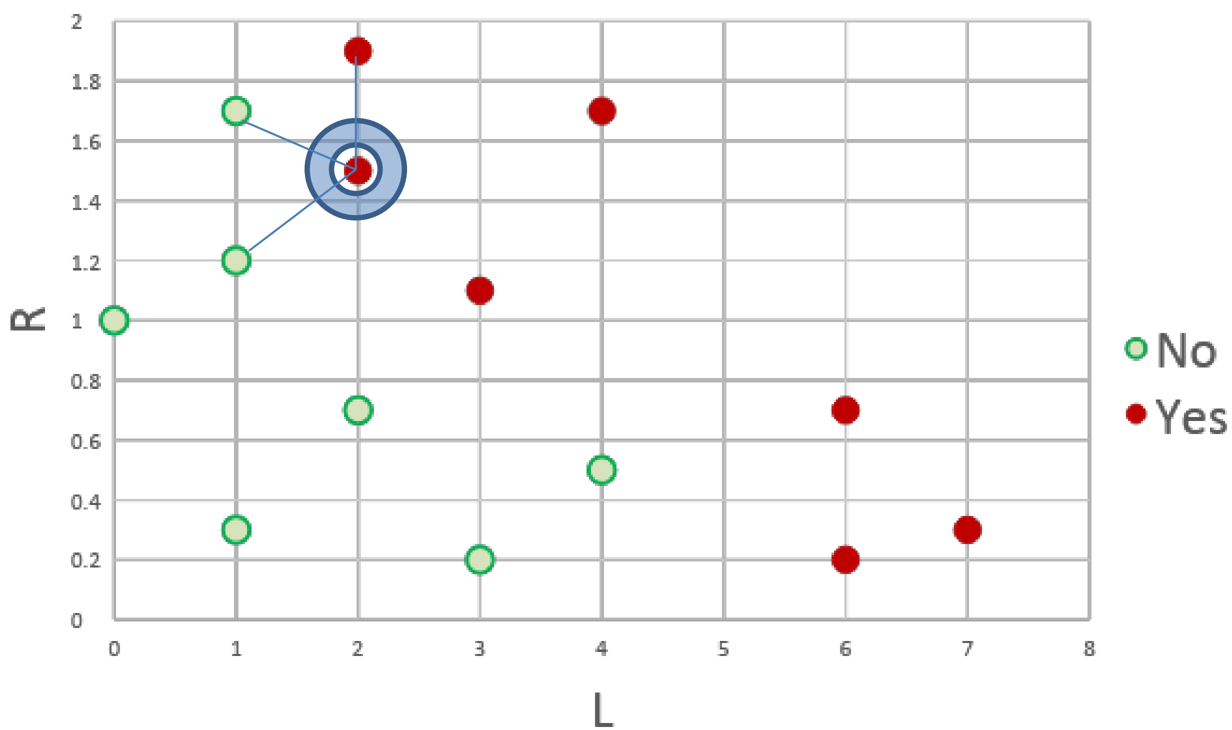


For ***K*=1**:
Error rate **3/14**

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

Number of data dimensions

More dimensions => better classifier?

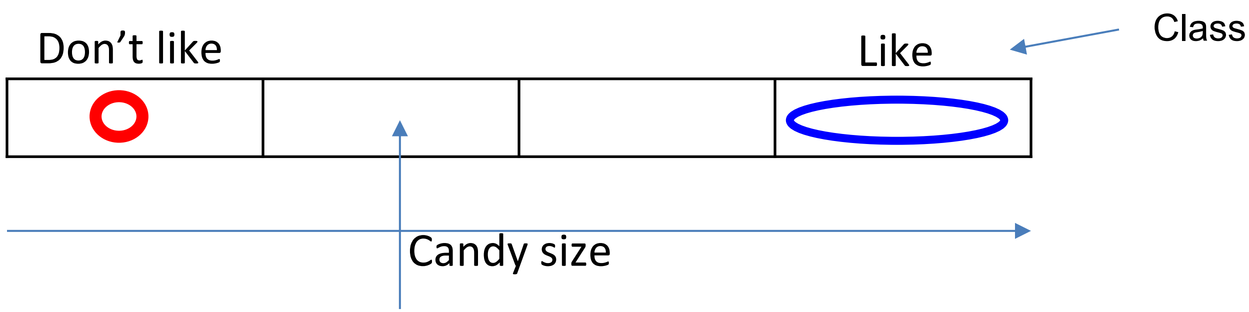
- Imagine:
  - Plot 100 data points on a straight line (where floor and a wall meet).
  - Plot the same 100 points on a wall.
  - Imagine a 3D room where you plot the same 100 points.

- The set of points becomes more and 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 is now separated by an exponentially large empty spaces.

- The prediction in sparse high-dimensional space will be **less** reliable.

The curse of dimensionality: example

1 dimension



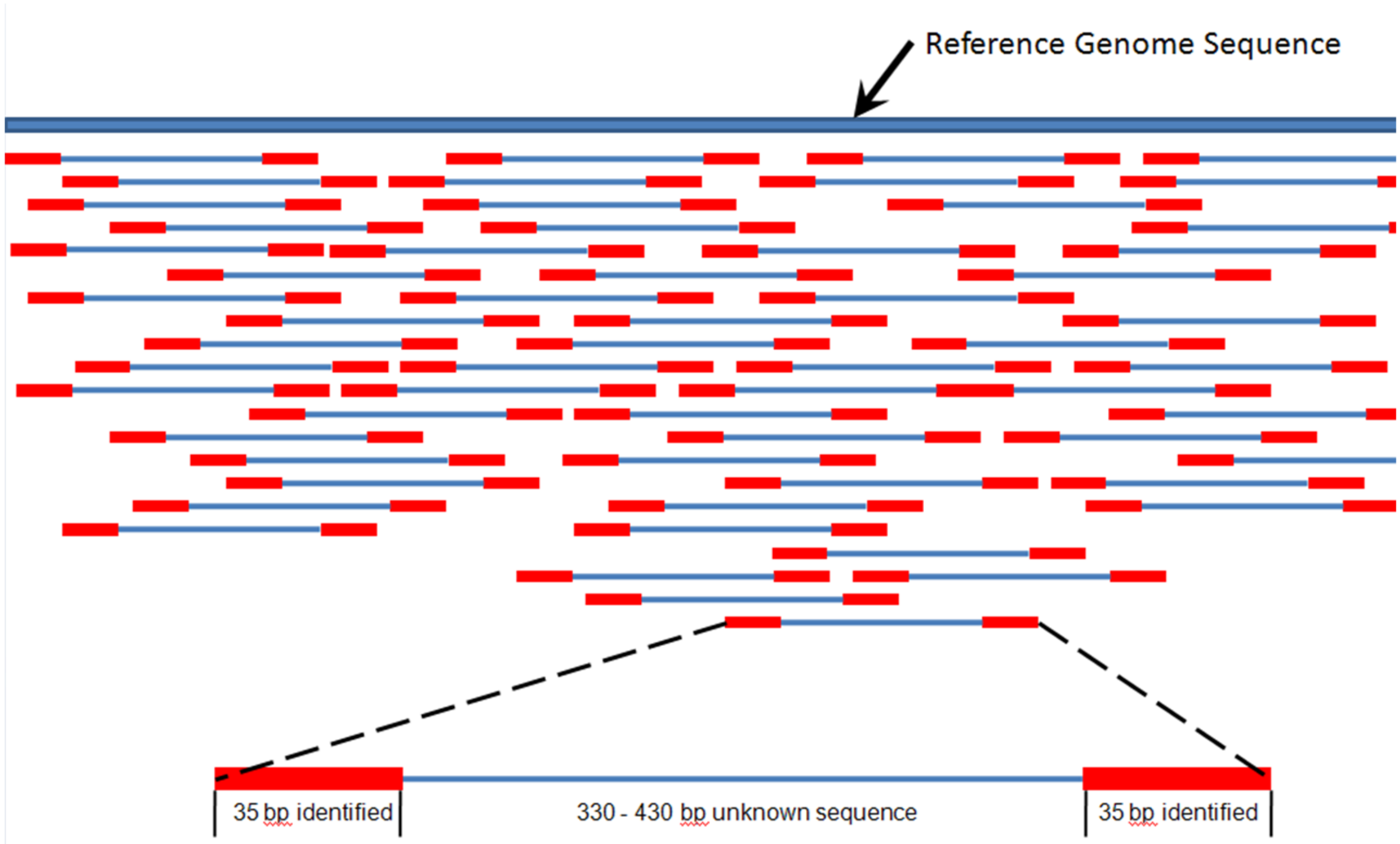
Predict: ? Prediction is always reliable

2 dimensions

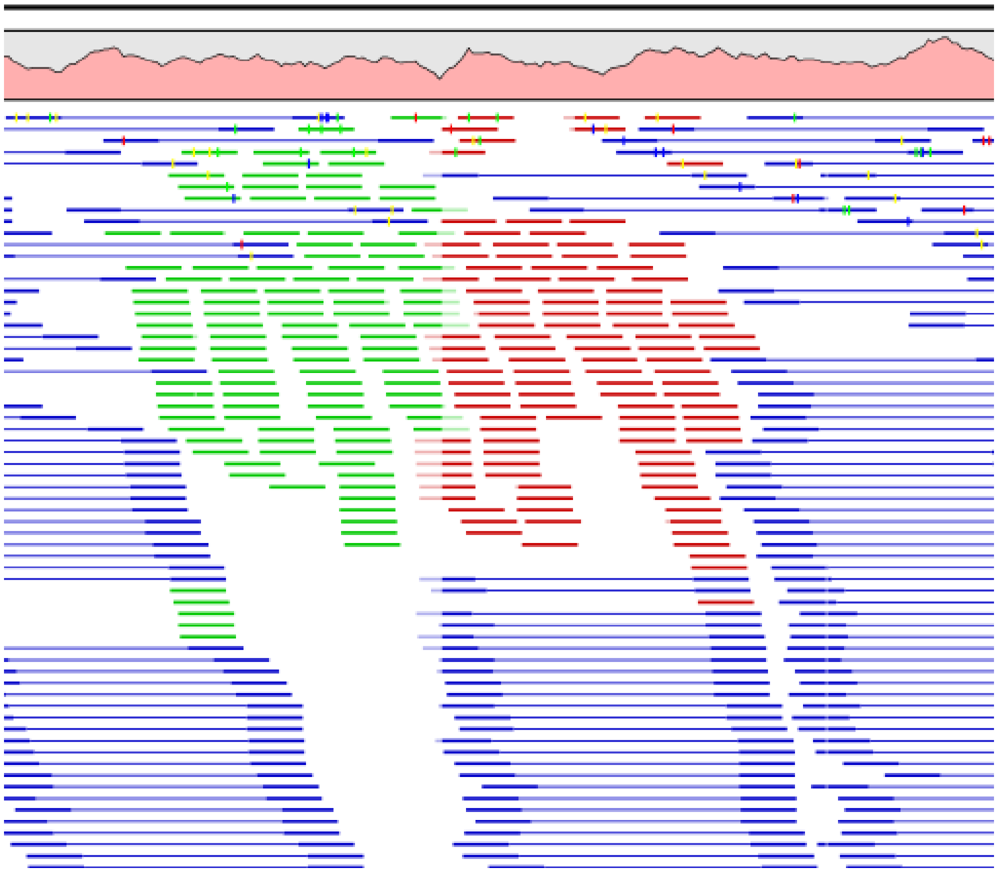
Predict: ? Prediction is not reliable: need more data points

Reducing number of dimensions: Principal Component Analysis
PCA: main idea

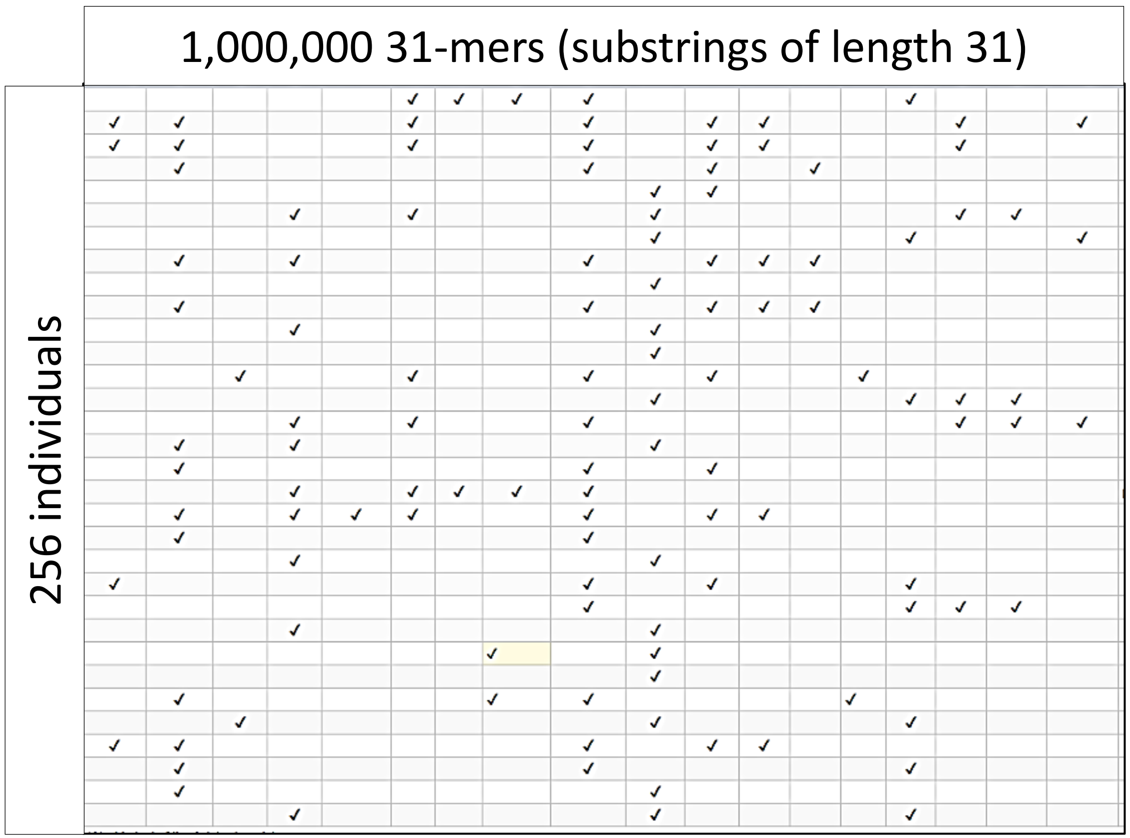
Practical example: Short raw DNA reads



Input Dataset:
20 TB of short reads from 232 individuals



Experiment: inferring populations without reference



https://barsky.ca/marina/UTOR/experiments/bio_example/index.html

K-NN algorithm. Summary

- The training set ***is the*** model
- Advantages:
  - Building classifier: zero work
  - Updating the model with every new record: zero work
  - Interpretable: we can justify our classification
  - Good also for predicting numeric values (Regressor)
- Disadvantages:
  - The query is computationally expensive: O(N*M), where N-number of points, M-number of dimensions