Nearest neighbor classification

DSE 220

Decision Trees

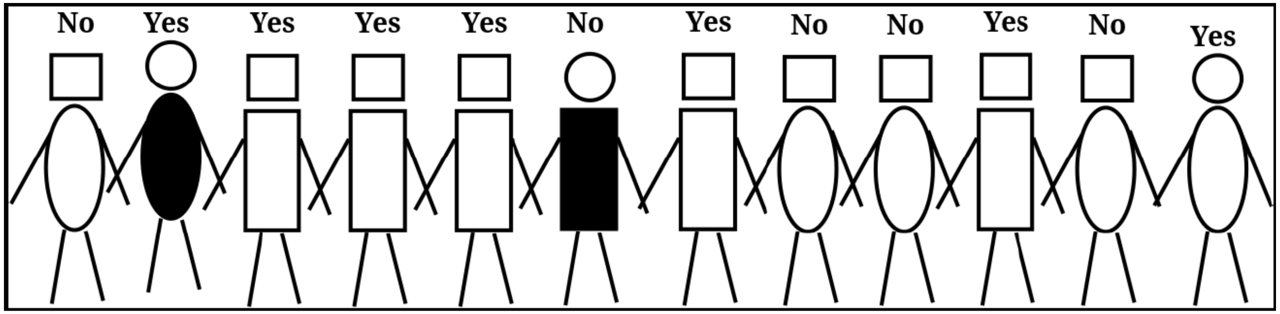
Target variable
Label
Dependent variable
Output space

Person ID	Age	Gender	Income	Balance	Mortgage payment
123213	32	F	25000	32000	Y
17824	49	M	12000	-3000	N
232897	60	F	8000	1000	Y
288822	28	M	9000	3000	Y
....	....	....	....	....	....

Decision Trees

- How can we judge whether a variable contains important information about the target variable?
- How can we (automatically) obtain a selection of the more informative variables with respect to predicting the value of the target variable?
- Even better, can we obtain the ranking of the variables?

Decision Trees



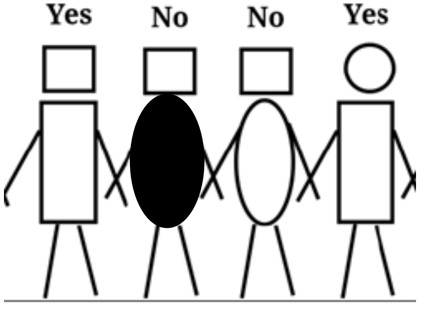
- Attributes:
  - head-shape: square, circular
  - body-shape: rectangular, oval
  - body-color: black, white
- Target variable: Yes, No

Decision Trees

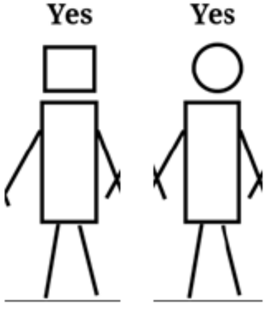
- Which attribute is the most informative? Or the most useful for distinguishing between data instances?
- If we split our data according to this variable, we would like the resulting groups to be as *pure* as possible.
- By pure we mean *homogeneous with respect to the target variable*.
- If every member of a group has the same value for the target, then the group is totally pure.

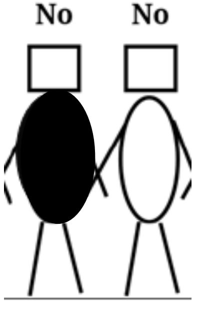
Example

- If this is our entire dataset:

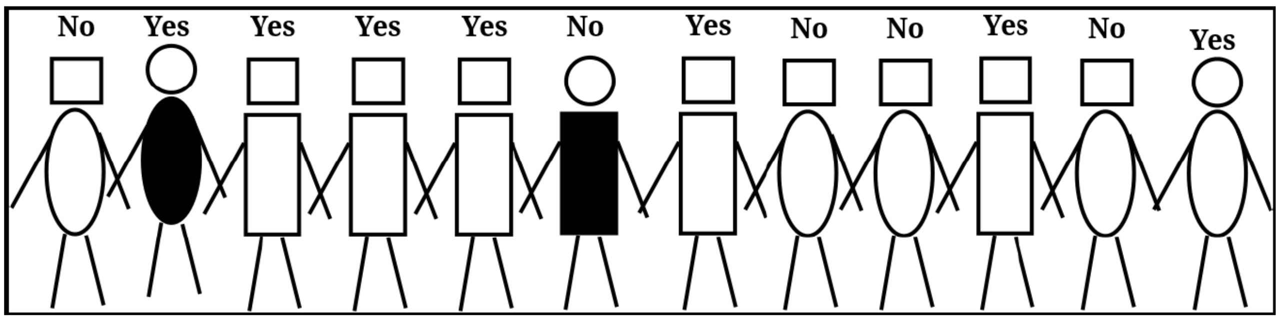


- Then, we can obtain two pure groups by splitting according to body shape:





Concerns



- Attributes rarely split a group perfectly.
- Even if one subgroup happens to be pure, the other may not.
- Is a very small, pure group, a good thing?
- How should continuous and categorical attributes be handled?

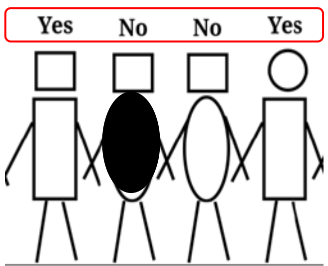
Entropy and Information Gain

- Target variable has two (or more) categories: 1, 2 (,…m)
- Probability P1 for category 1
- Probability P2 for category 2
- …
- Entropy:

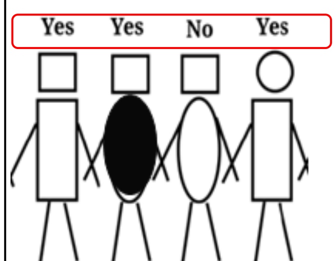
$$H(X) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 - \ldots - p_m \log_2 p_m$$

Entropy

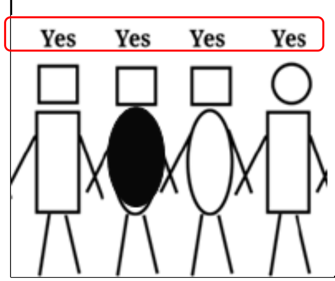
$$H(X) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 - \ldots - p_m \log_2 p_m$$



$$H(X) = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$



$$H(X) = -0.75 \log_2 0.75 - 0.25 \log_2 0.25 = 0.81$$

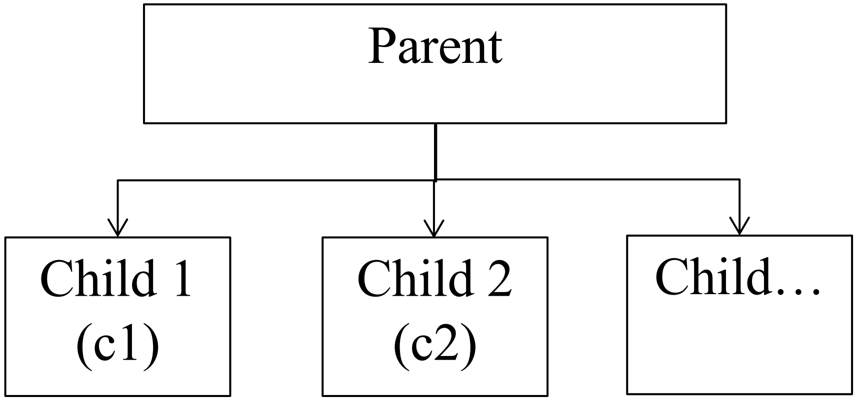


$$H(X) = -1 \log_2 1 = 0$$

Information Gain

- Calculation of information gain (IG):
- IG (parent, children) =

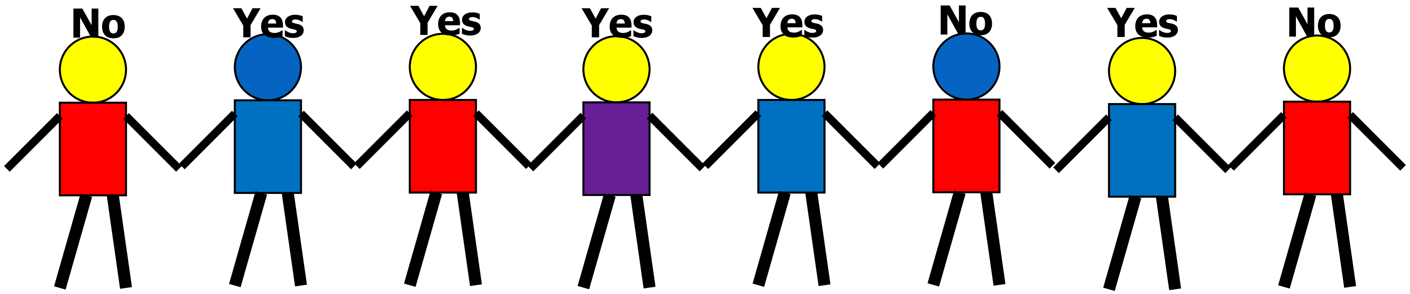
entropy(parent)−[p(c1)×entropy(c1)+p(c2)×entropy(c2) +…]



Note: Higher IG indicates a more informative split by the variable.

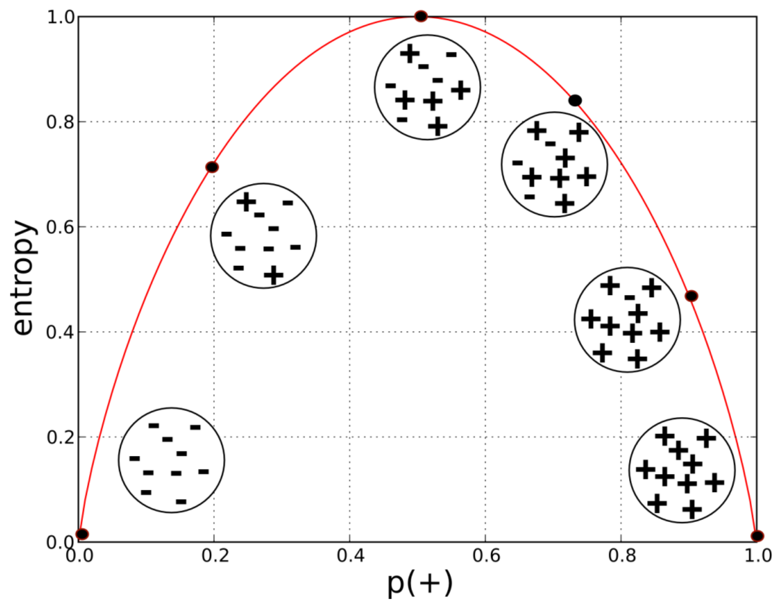
Selecting Informative Attributes

- Objective: Based on customer attributes, partition the customers into subgroups that are less impure – with respect to the class (i.e., such that in each group as many instances as possible belong to the same class)



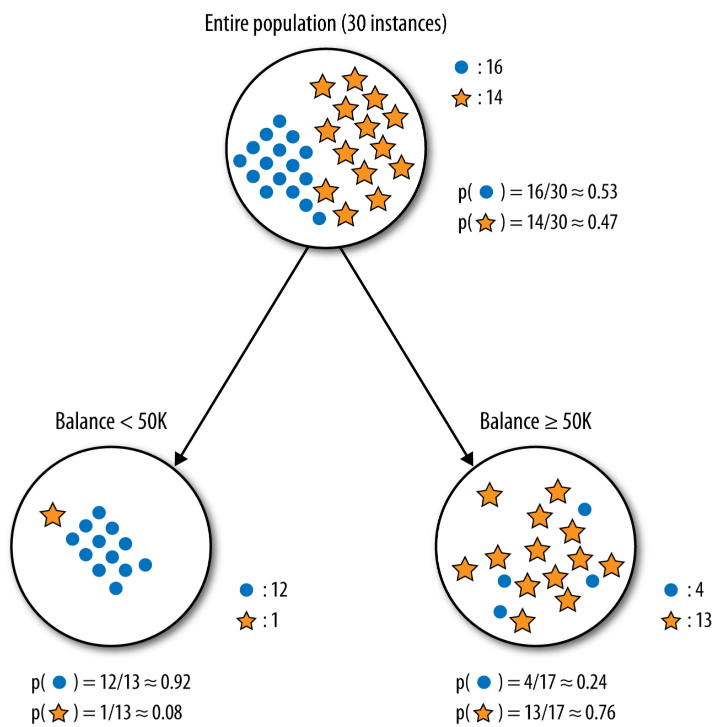
Selecting Informative Attributes

- The most common splitting criterion is called **information gain** (IG)
  - It is based on a **purity measure** called **entropy**
    - $entropy = -\mathrm{p}_1 \log_2(p_1) - p_2 \log_2(p_2) - ..$
    - Measures the general disorder of a set



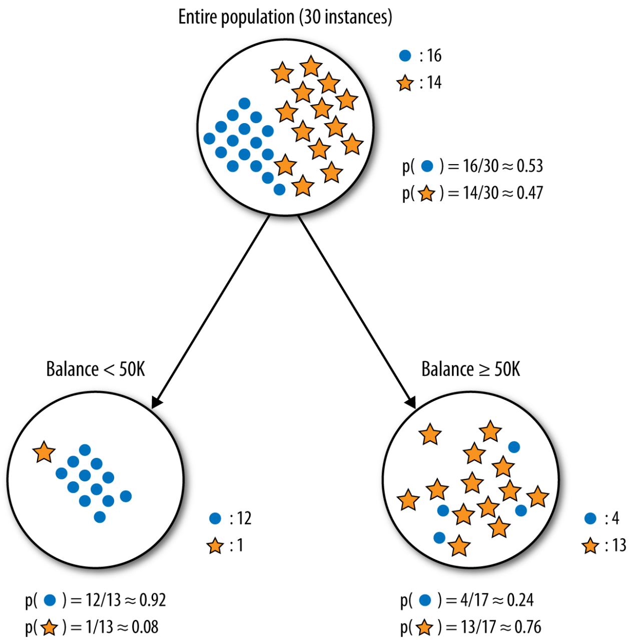
Information Gain

- Information gain measures the *change* in entropy due to any amount of new information being added



$$IG(parent, children) = entropy(parent) - [p(c_1) \times entropy(c_1) + p(c_2) \times entropy(c_2) + \cdots]$$

Information Gain



$$
\begin{aligned}
entropy(parent) &= -[p(\bullet) \times \log_2 p(\bullet) + p(\star) \times \log_2 p(\star)] \\
&\approx -[0.53 \times -0.9 + 0.47 \times -1.1] \\
&\approx 0.99 \quad \text{(very impure)}
\end{aligned}
$$

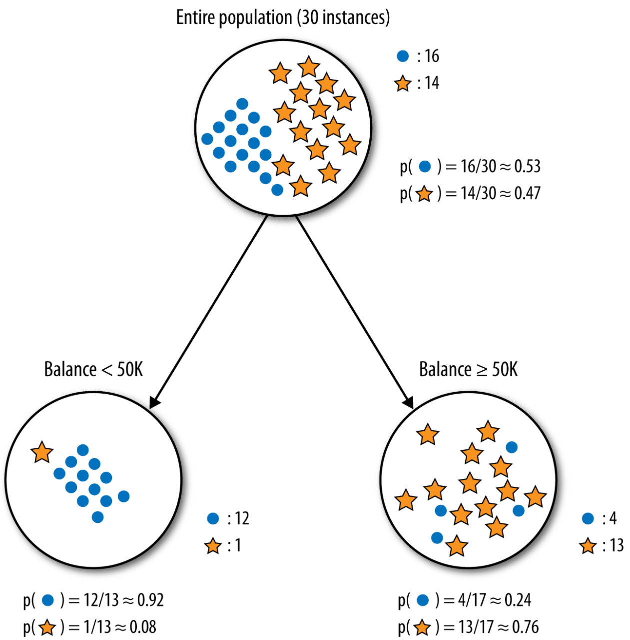
The entropy of the *left* child is:

$$
\begin{aligned}
entropy(Balance < 50K) &= -[p(\bullet) \times \log_2 p(\bullet) + p(\star) \times \log_2 p(\star)] \\
&\approx -[0.92 \times (-0.12) + 0.08 \times (-3.7)] \\
&\approx 0.39
\end{aligned}
$$

The entropy of the *right* child is:

$$
\begin{aligned}
entropy(Balance \geq 50K) &= -[p(\bullet) \times \log_2 p(\bullet) + p(\star) \times \log_2 p(\star)] \\
&\approx -[0.24 \times (-2.1) + 0.76 \times (-0.39)] \\
&\approx 0.79
\end{aligned}
$$

Information Gain



$$
\begin{aligned}
IG \quad = \quad & entropy(parent) - [p(\text{Balance} < 50\text{K}) \times entropy(\text{Balance} < 50\text{K}) \\
& \qquad + p(\text{Balance} \geq 50\text{K}) \times entropy(\text{Balance} \geq 50\text{K})] \\
\approx \quad & 0.99 - [0.43 \times 0.39 + 0.57 \times 0.79] \\
\approx \quad & 0.37
\end{aligned}
$$

So far…

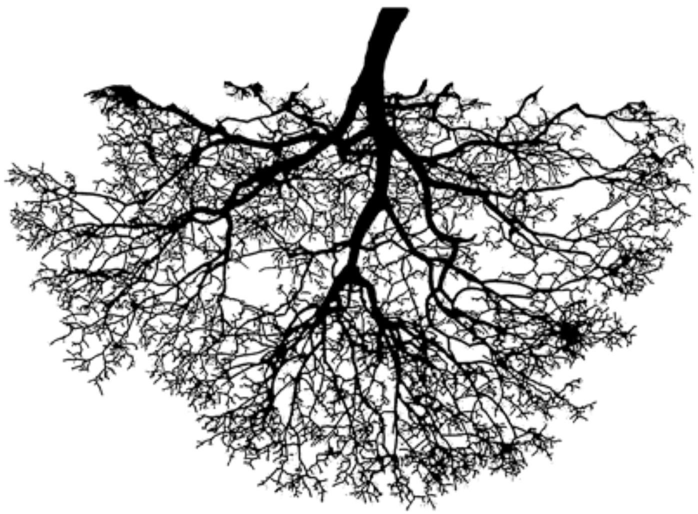
- We have measures of:
  - Purity of the data (entropy)
  - How informative is (a split by) a variable
- We can identify and rank informative variables
- Next – we will use this method to build our first supervised learning classifier – a decision tree

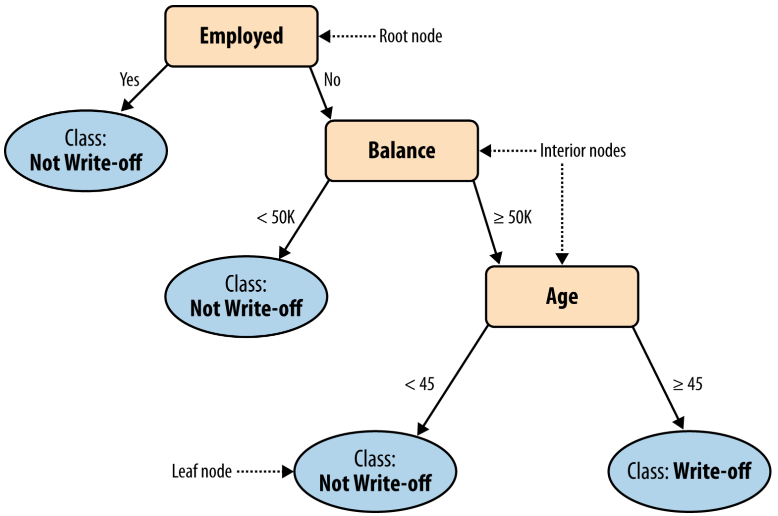
Multivariate Supervised Segmentation

- If we select the *single* variable that gives the most information gain, we create a very *simple* segmentation

- If we select multiple attributes each giving some information gain, how do we put them together?

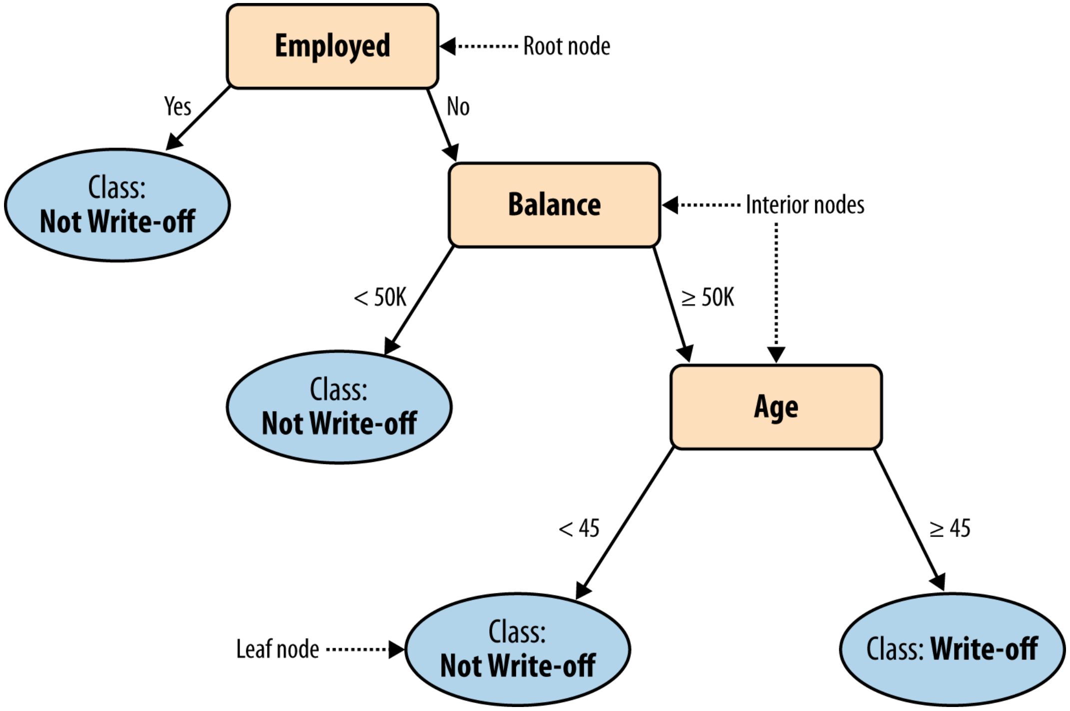
Tree-Structured Models





Tree-Structured Models

- Classify 'John Doe'
  - Balance=115K, Employed=No, and Age=40



Tree-Structured Models: “Rules”

- No two parents share descendants
- There are no cycles
- The branches always “point downwards”
- Every example always ends up at a leaf node with some specific class determination
    - Probability estimation trees, regression trees (*to be continued*..)

Tree Induction

- How do we create a classification tree from data?
  - **divide-and-conquer** approach
  - take each data subset and ***recursively*** apply attribute selection to find the best attribute to partition it

- When do we stop?
  - The nodes are pure,
  - there are no more variables, or
  - even earlier (over-fitting – *to be continued*..)

Why trees?

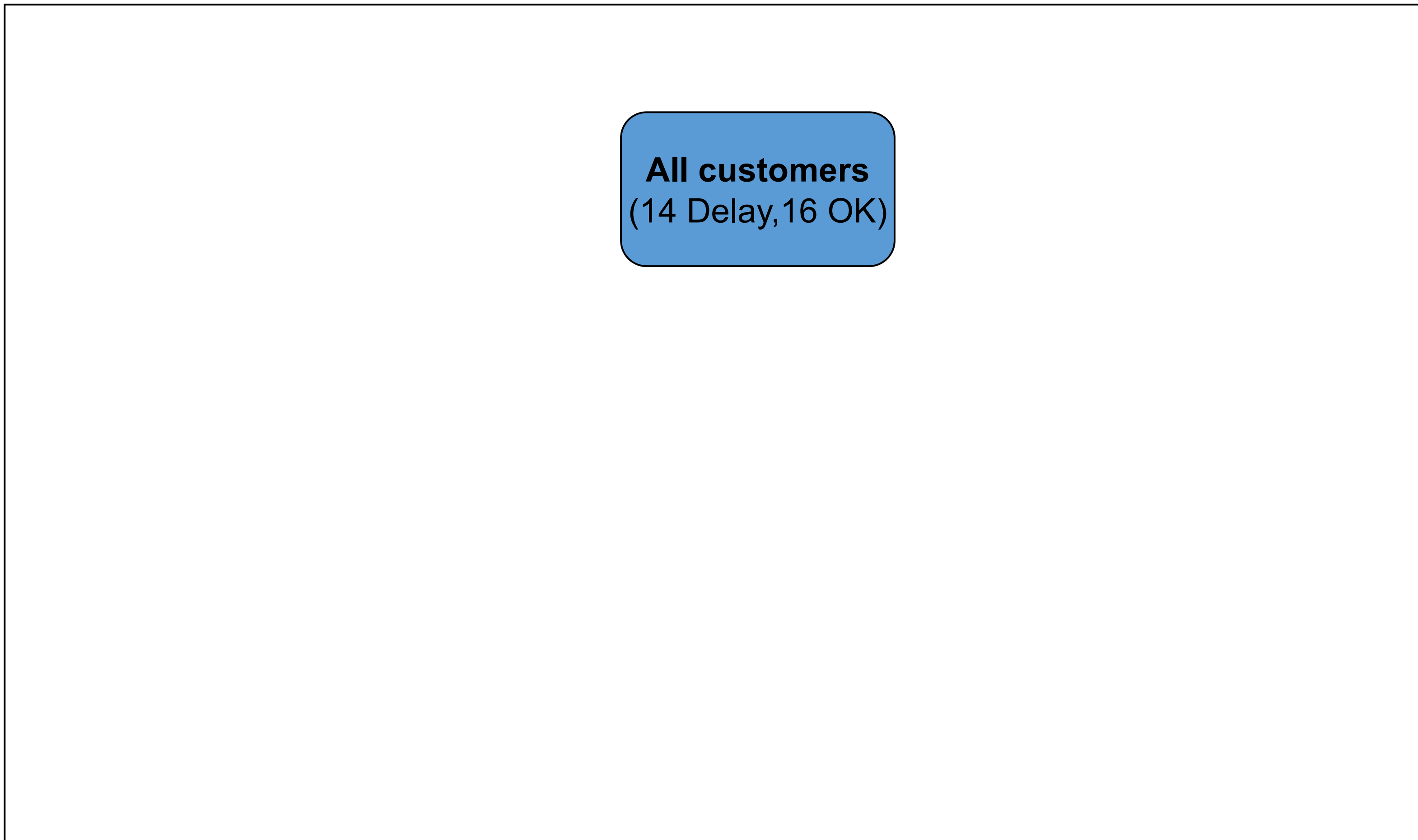
- Decision trees (DTs), or classification trees, are one of the most popular data mining tools
  - (along with linear and logistic regression)
- They are:
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Computationally cheap
- Almost all data mining packages include DTs
- They have advantages for model comprehensibility, which is important for:
  - model evaluation
  - communication to non-DM-savvy stakeholders

Dataset

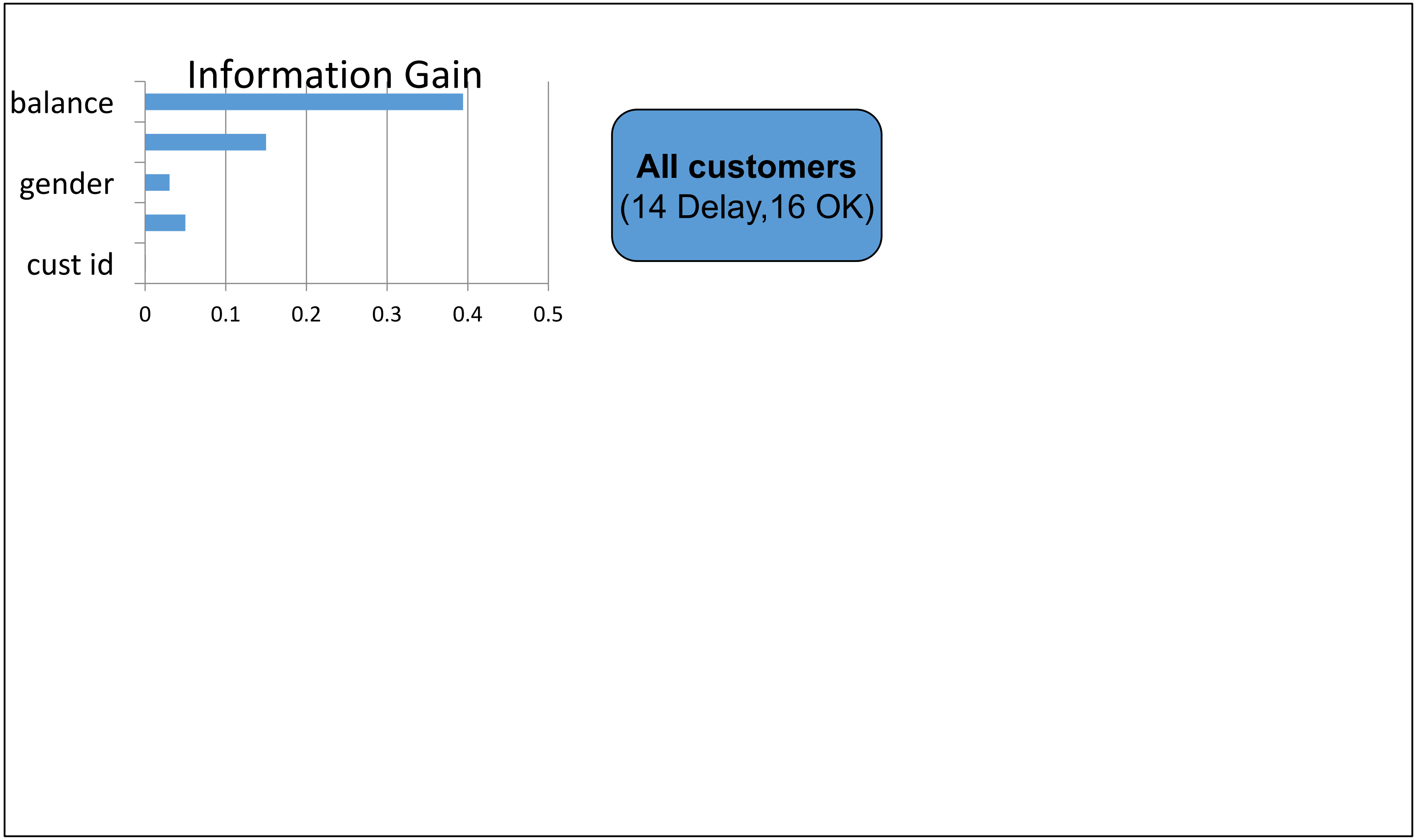
person id	age>50	gender	residence	Balance>= 50,000	mortgage payment delay
123213	N	F	own	N	delayed
17824	Y	M	own	Y	OK
232897	N	F	rent	N	delayed
288822	Y	M	other	N	delayed
….	….	….	….	….	….

Based on this dataset we will build a tree-based classifier.

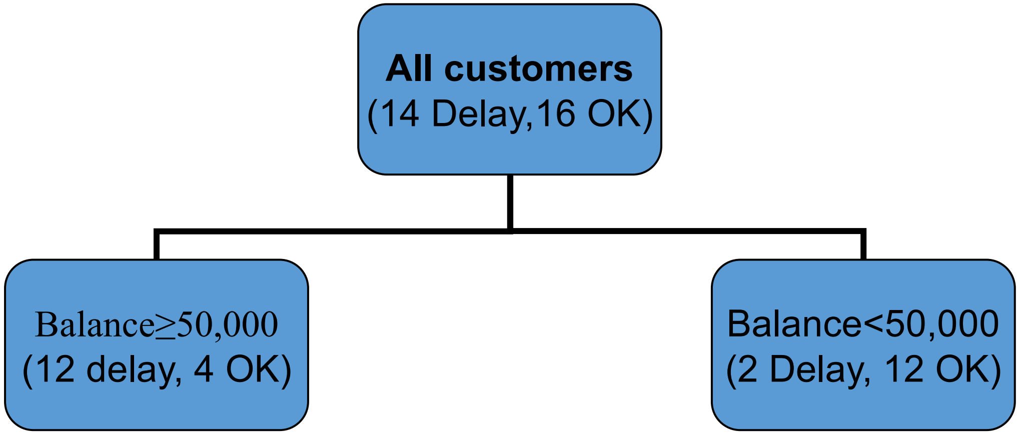
Tree Structure



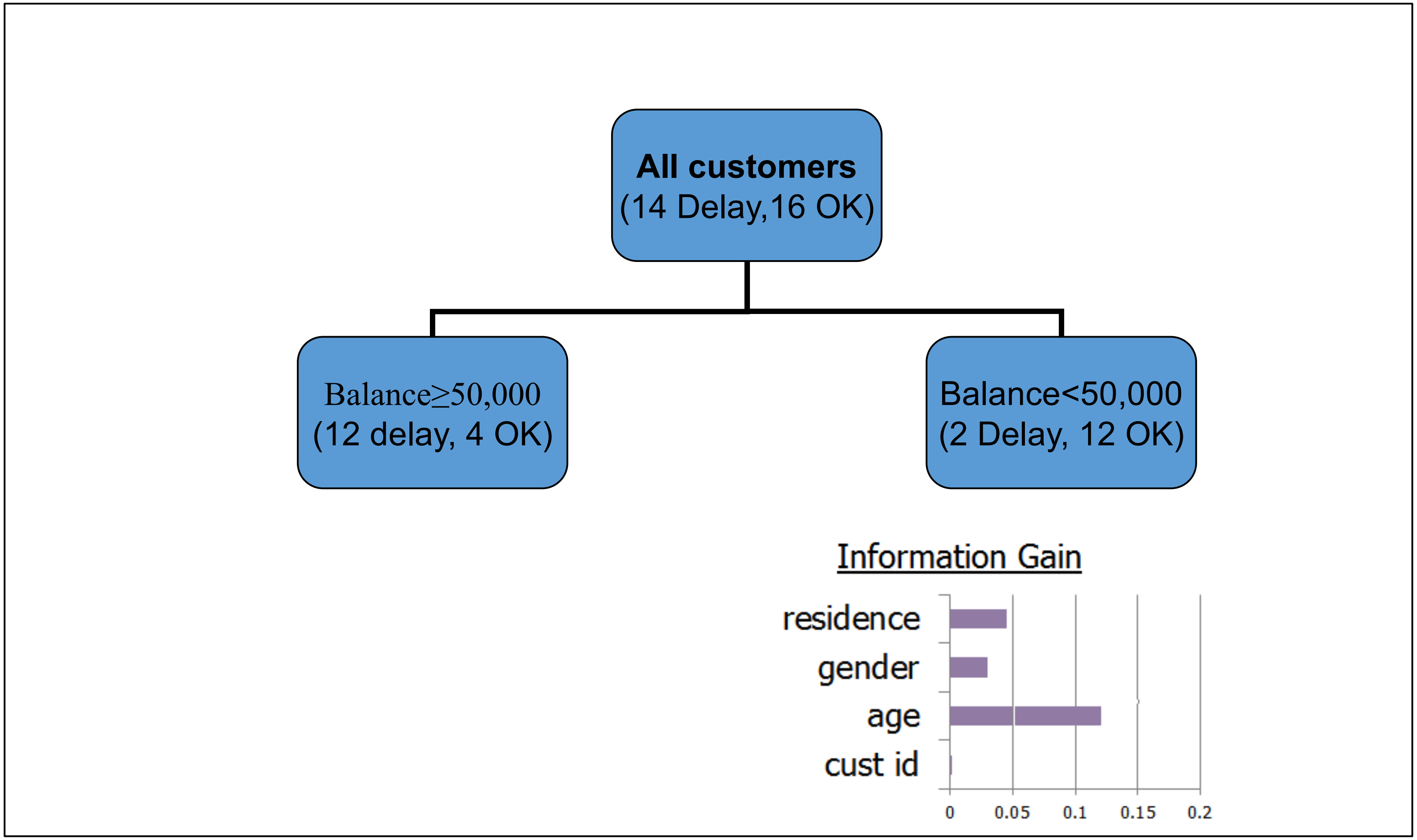
Tree Structure



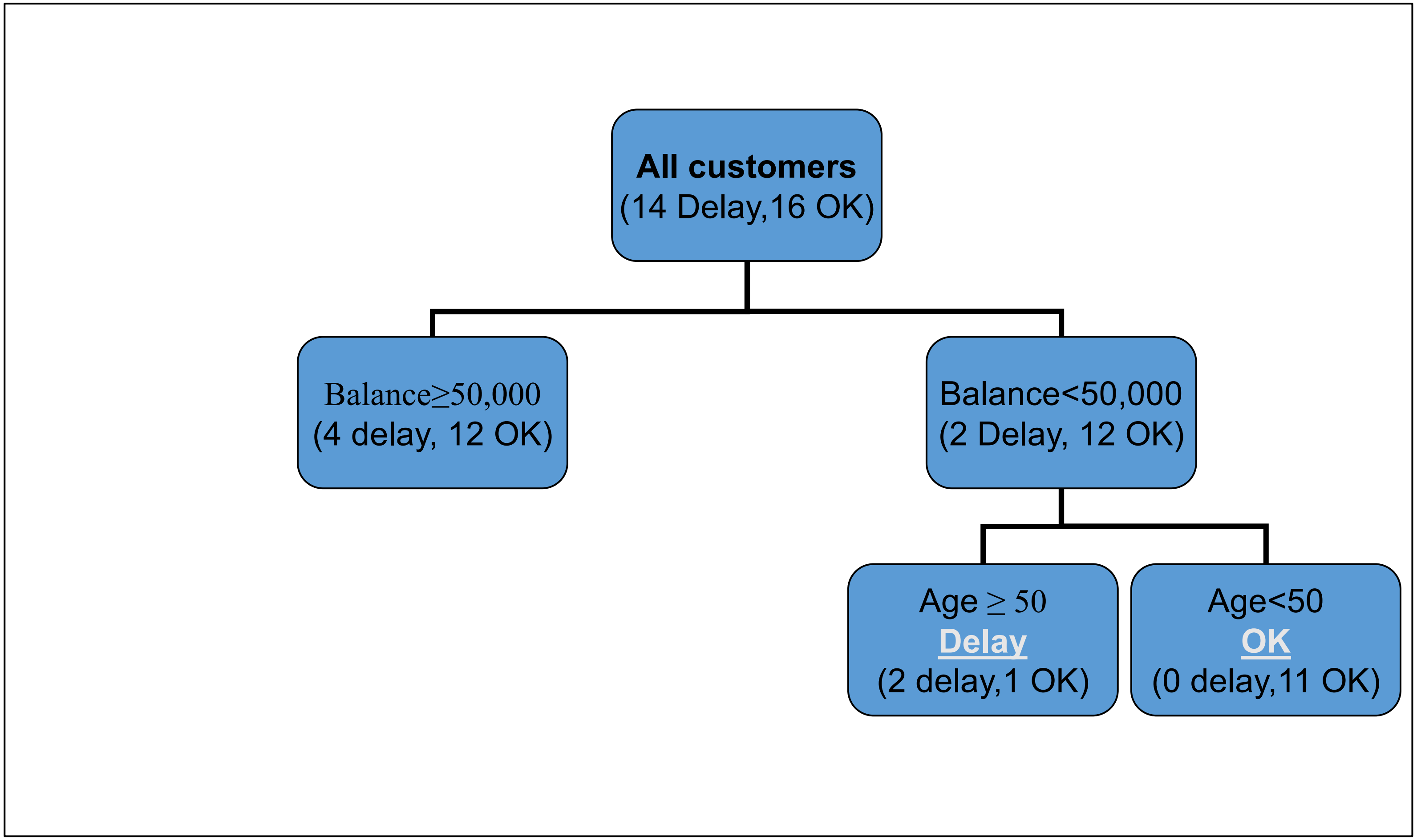
Tree Structure



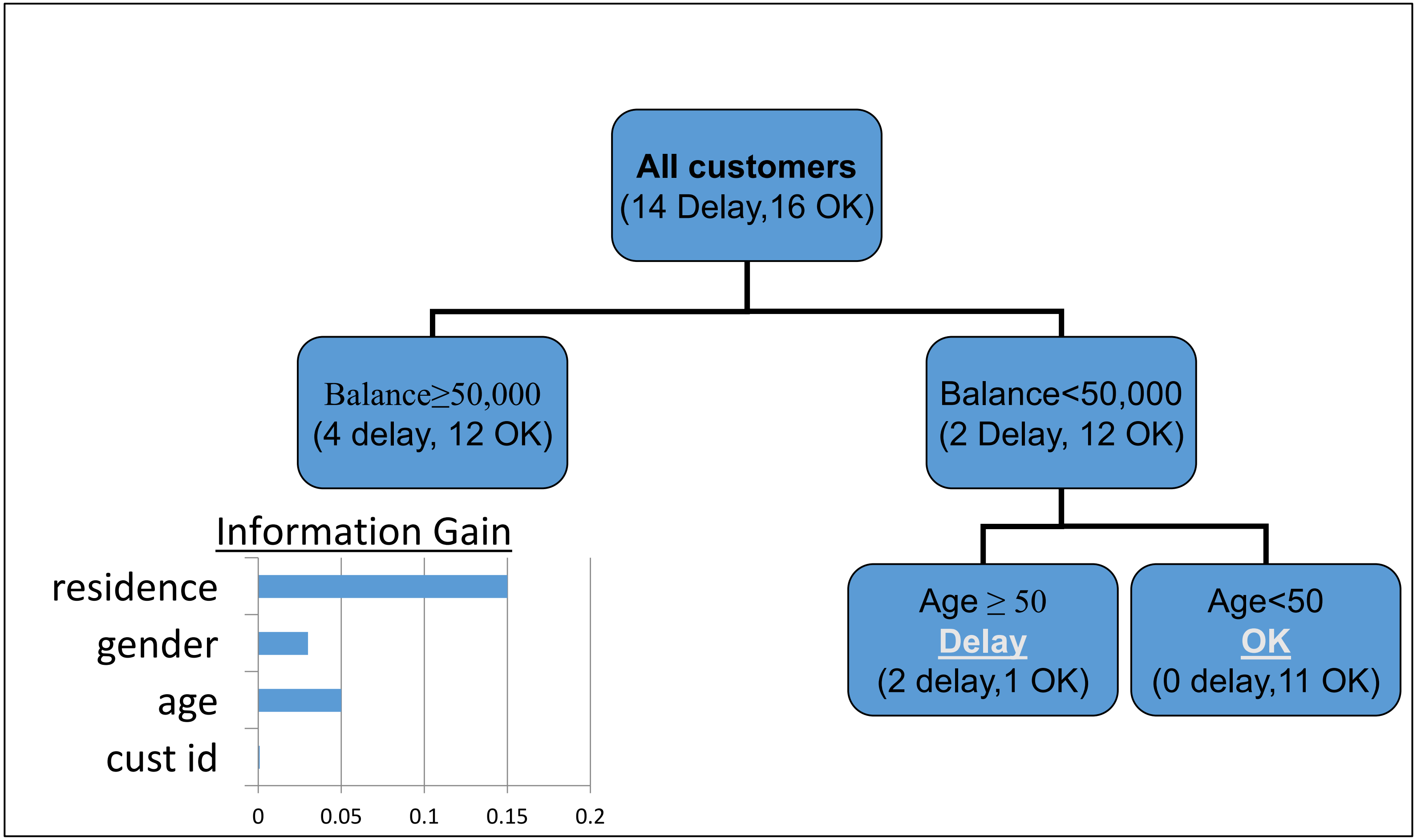
Tree Structure



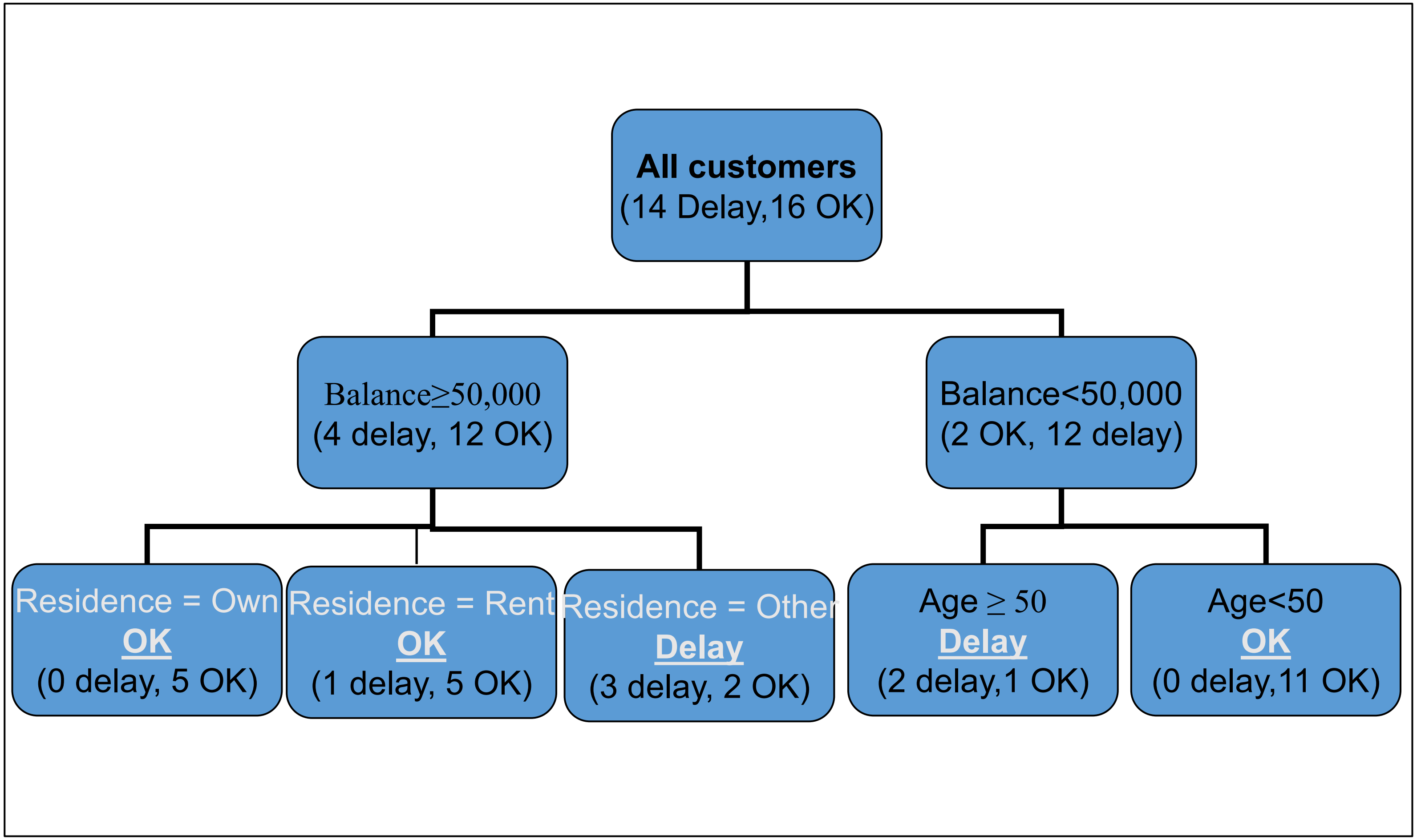
Tree Structure



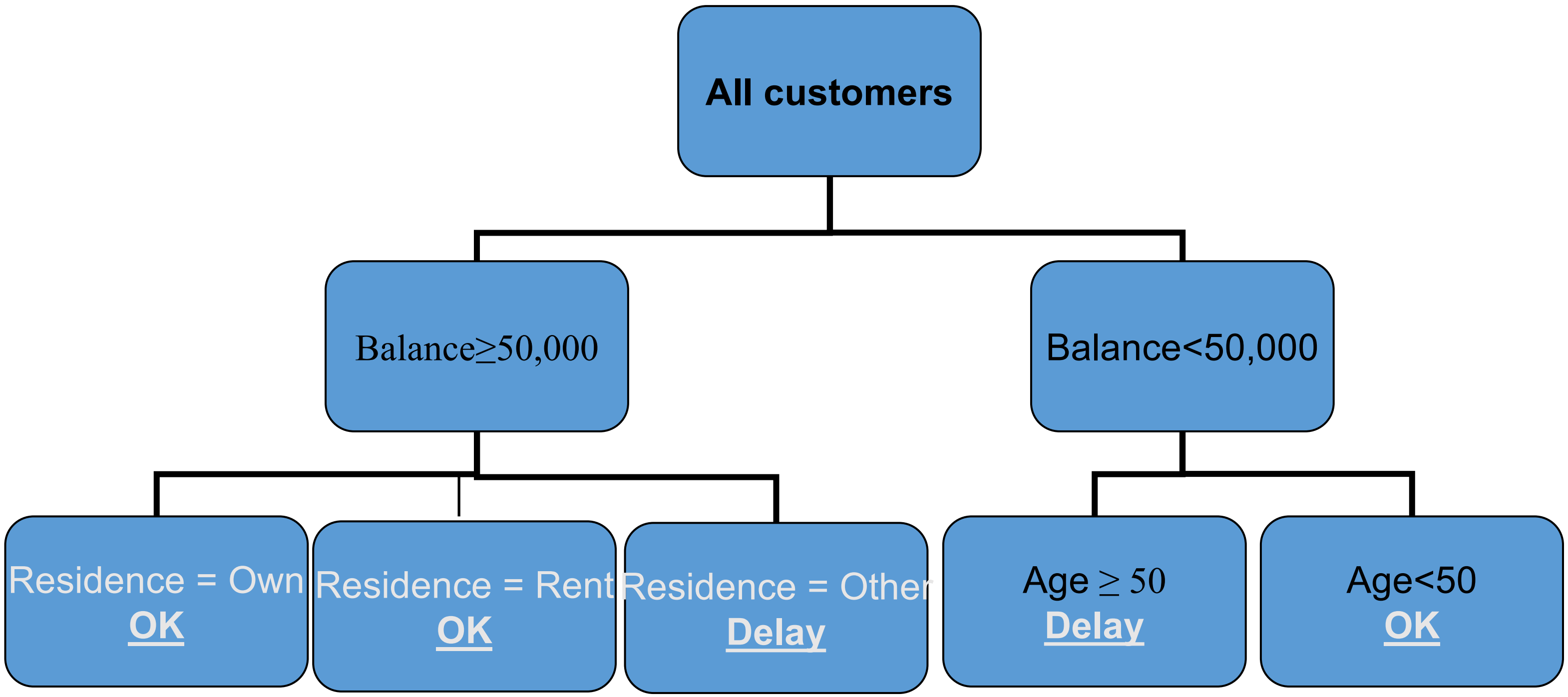
Tree Structure



Tree Structure

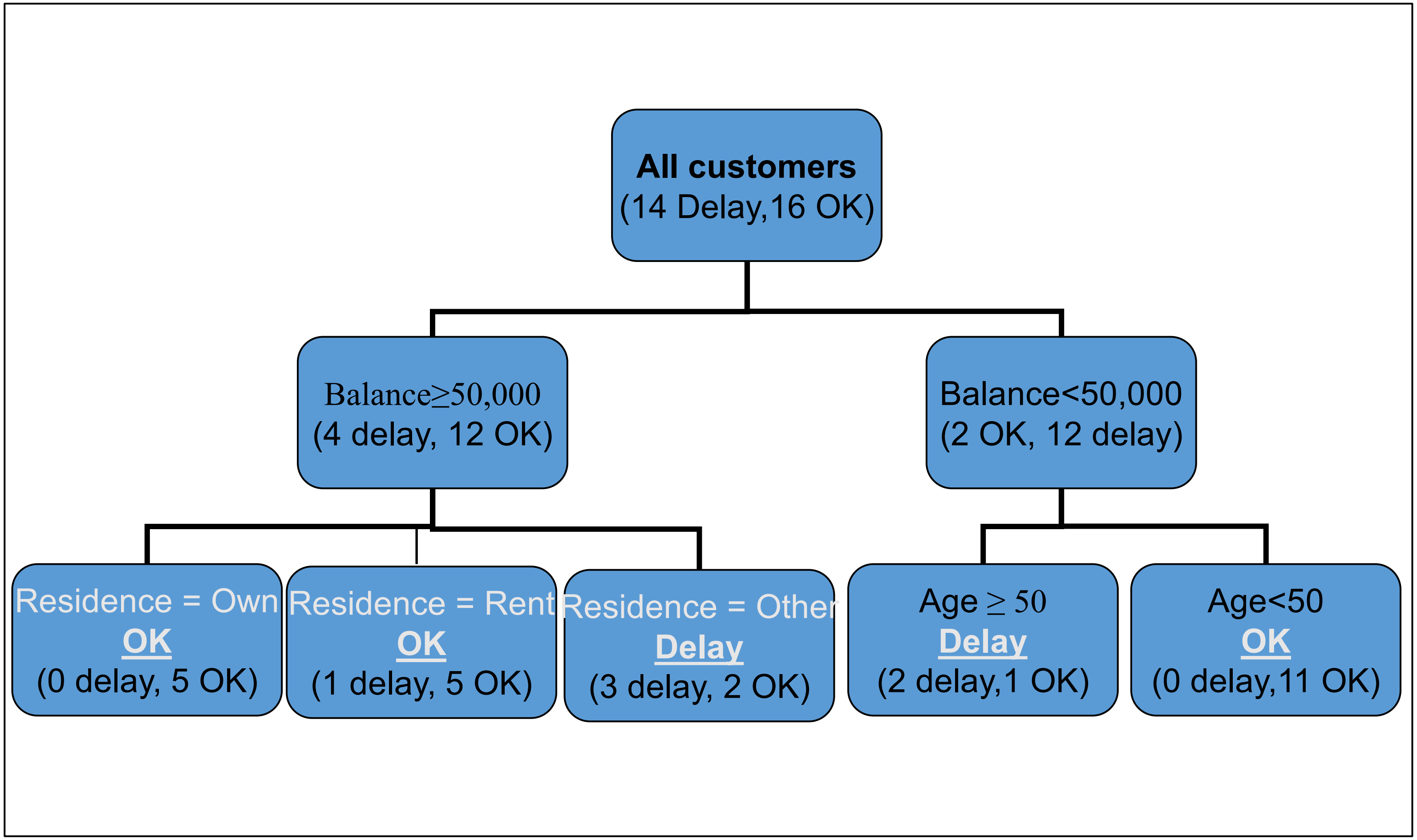


Tree Structure



iD	Age>50	Gender	Residenc e	Balance >=50K	Delay
87594	Y	F	own	<50K	???

Open Issues

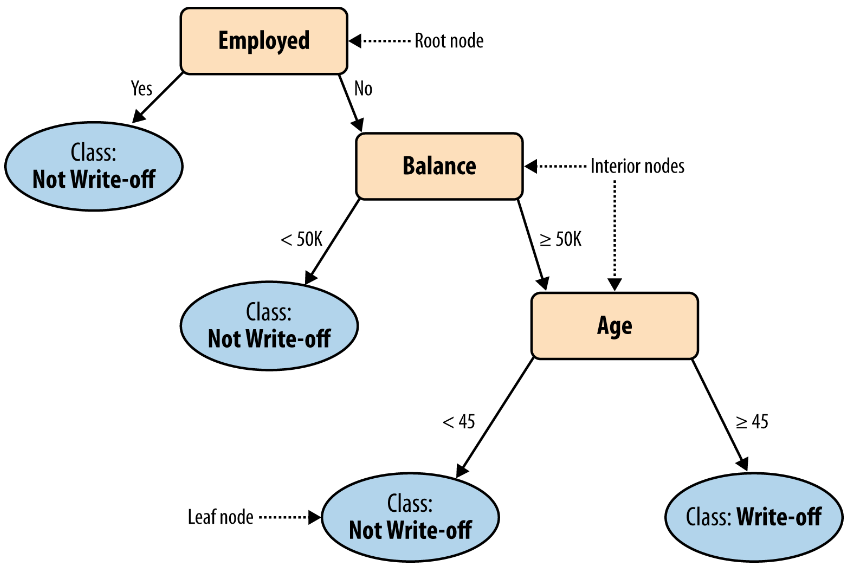


All customers
(14 Delay,16 OK)
Balance≥50,000
(4 delay, 12 OK)
Balance<50,000
(2 OK, 12 delay)
Residence = Own
OK
(0 delay, 5 OK)
Residence = Rent
OK
(1 delay, 5 OK)
Residence = Other
Delay
(3 delay, 2 OK)
Age ≥ 50
Delay
(2 delay,1 OK)
Age<50
OK
(0 delay,11 OK)

Trees as Sets of Rules

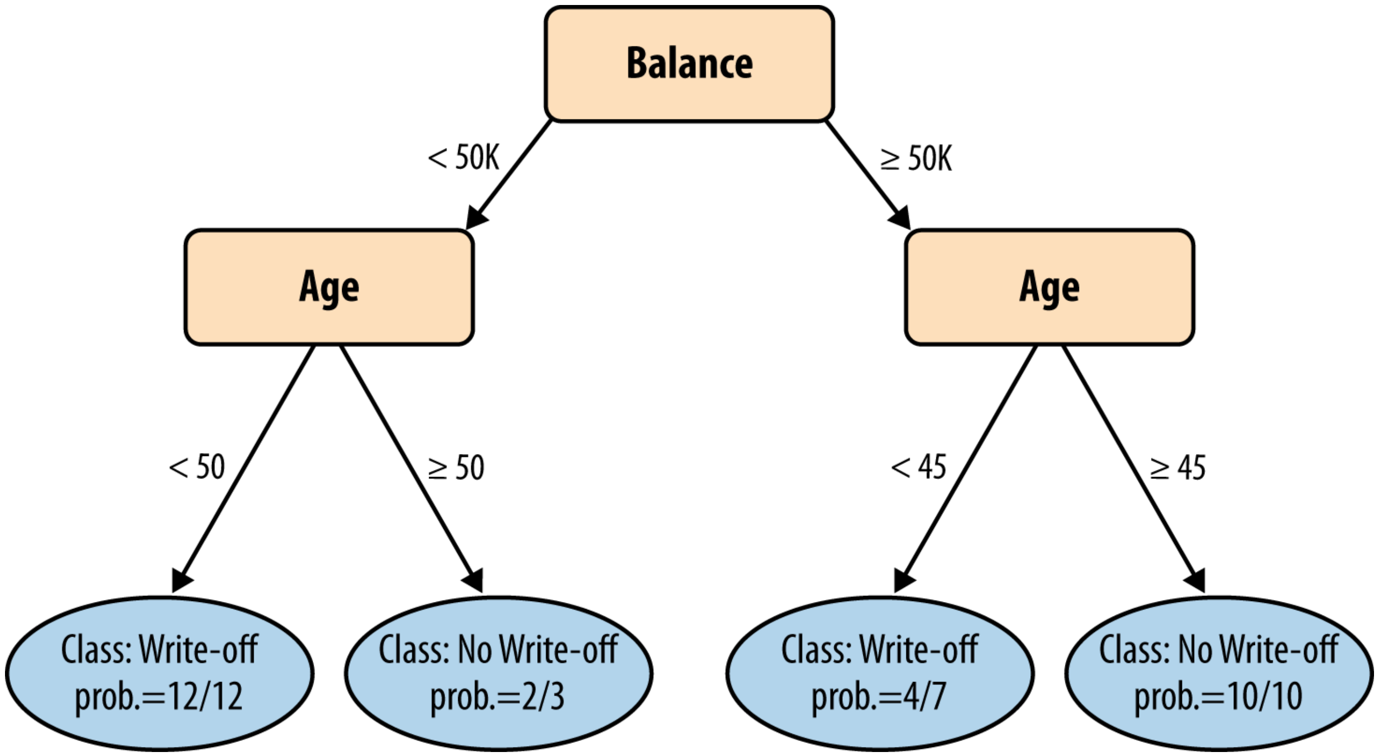
- The classification tree is equivalent to set of rules
- Each rule consists of the attribute tests along the path connected with **AND**

Trees as Sets of Rules

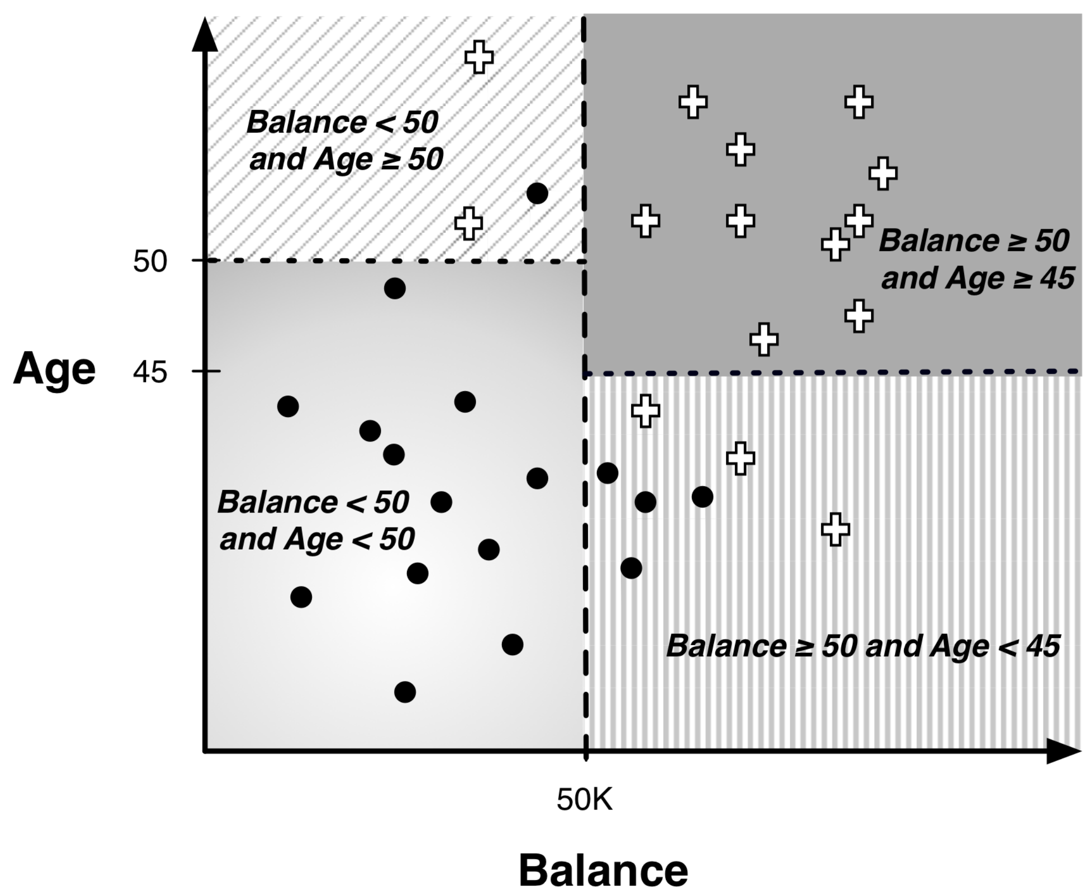


- IF (Employed = Yes) THEN Class=No Write-off
- IF (Employed = No) AND (Balance < 50k) THEN Class=No Write-off
- IF (Employed = No) AND (Balance ≥ 50k) AND (Age < 45) THEN Class=No Write-off
- IF (Employed = No) AND (Balance ≥ 50k) AND (Age ≥ 45) THEN Class=Write-off

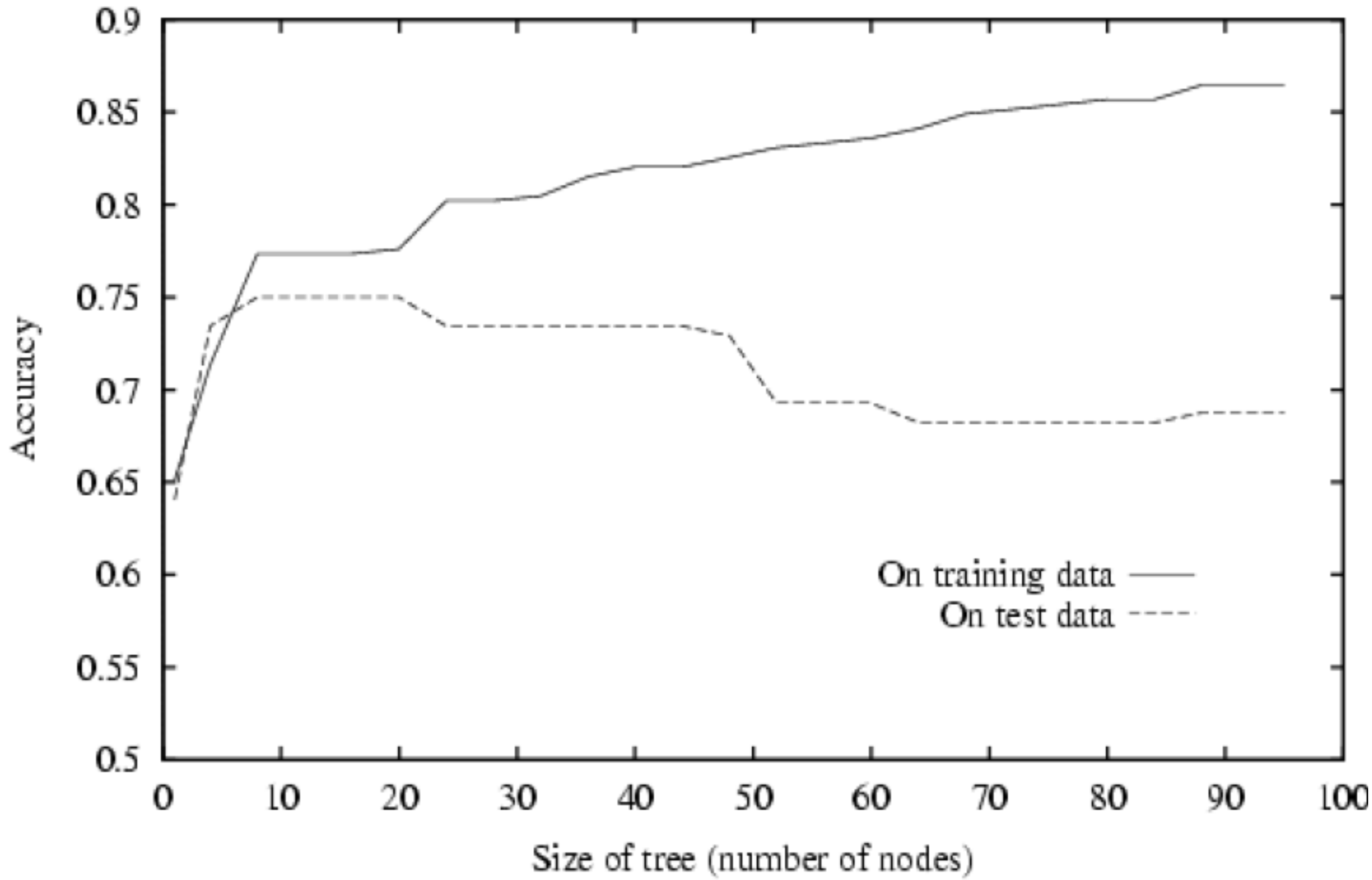
Visualizing Segmentations



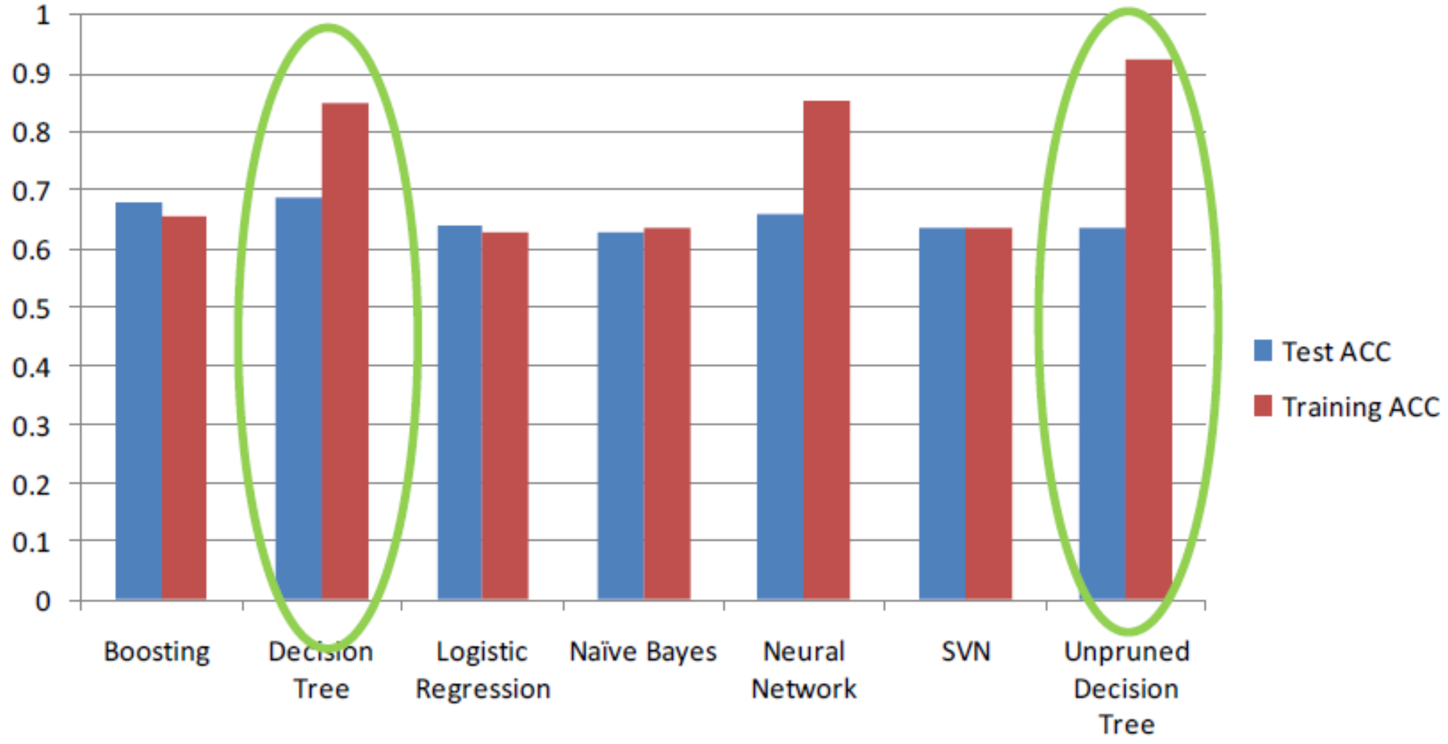
Visualizing Segmentations



Tree Complexity and Over-fitting



Trees on Churn



Pruning

- Pruning simplifies a decision tree to prevent over-fitting to noise in the data
- **Post-pruning**:
  - takes a fully-grown decision tree and discards unreliable parts
- **Pre-pruning**:
  - stops growing a branch when information becomes unreliable
- Post-pruning preferred in practice

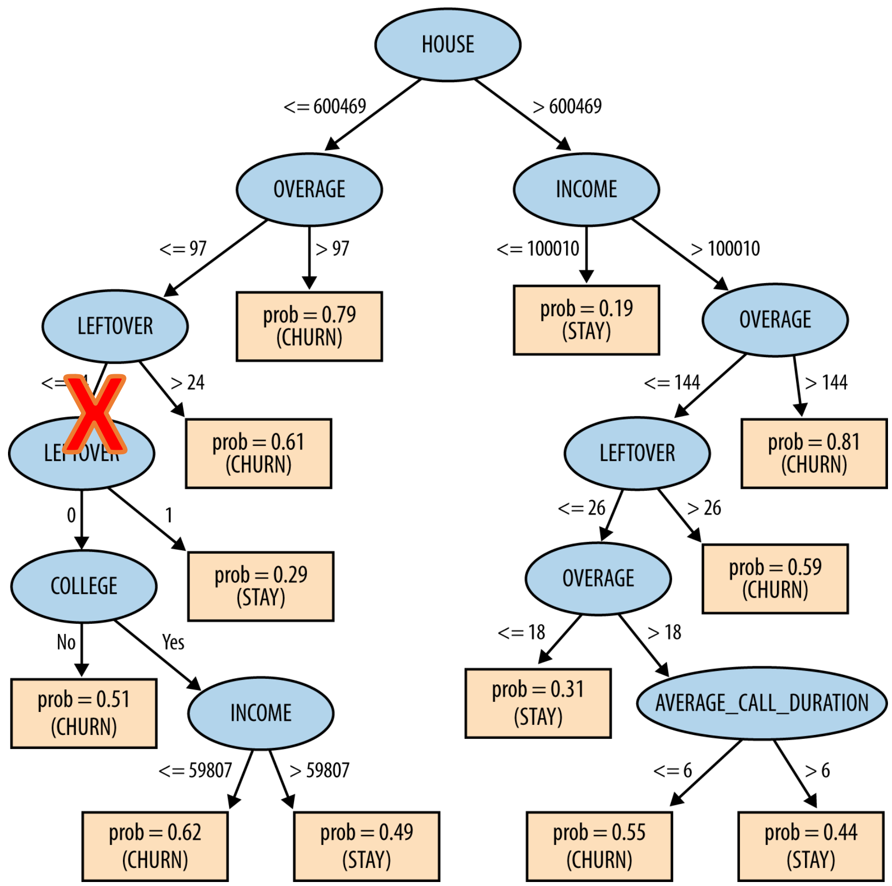
Decision Tree Pruning Methods

- Validation set – withhold a subset (~1/3) of training data to use for pruning
  - Note: you should randomize the order of training examples

Reduced-Error Pruning

- Classify examples in validation set – some might be errors
- For each node:
  - Sum the errors over entire subtree
  - Calculate error on same example if converted to a leaf with majority class label
- Prune node with highest reduction in error
- Repeat until error no longer reduced

MegaTelCo: Predicting Churn with Tree Induction



From Classification Trees to Probability Estimation Trees

- **Frequency-based estimate**
  - **Basic assumption**: Each member of a segment corresponding to a tree leaf has the same probability to belong in the corresponding class
  - If a leaf contains n positive instances and m negative instances (binary classification), the probability of any new instance being positive may be estimated as $\frac{n}{n+m}$

- Prone to **over-fitting**..

Laplace Correction

- $p(c) = \frac{n+1}{n+m+2}$,
  - where n is the number of examples in the leaf belonging to class c, and m is the number of examples not belonging to class c

The many faces of classification: Classification / Probability Estimation / Ranking

- Classification Problem
  - Most general case: The target takes on discrete values that are **NOT** ordered
  - Most common: binary classification where the target is either 0 or 1
- 3 Different Solutions to Classification
  - Classifier model: Model predicts the same set of **discrete value** as the data had
  - In binary case:
    - **Ranking**: Model predicts **a score** where a higher score indicates that the model think the example to be more likely to be in one class
    - **Probability estimation**: Model predicts **a score between 0 and 1 that is meant to be the probability** of being in that class

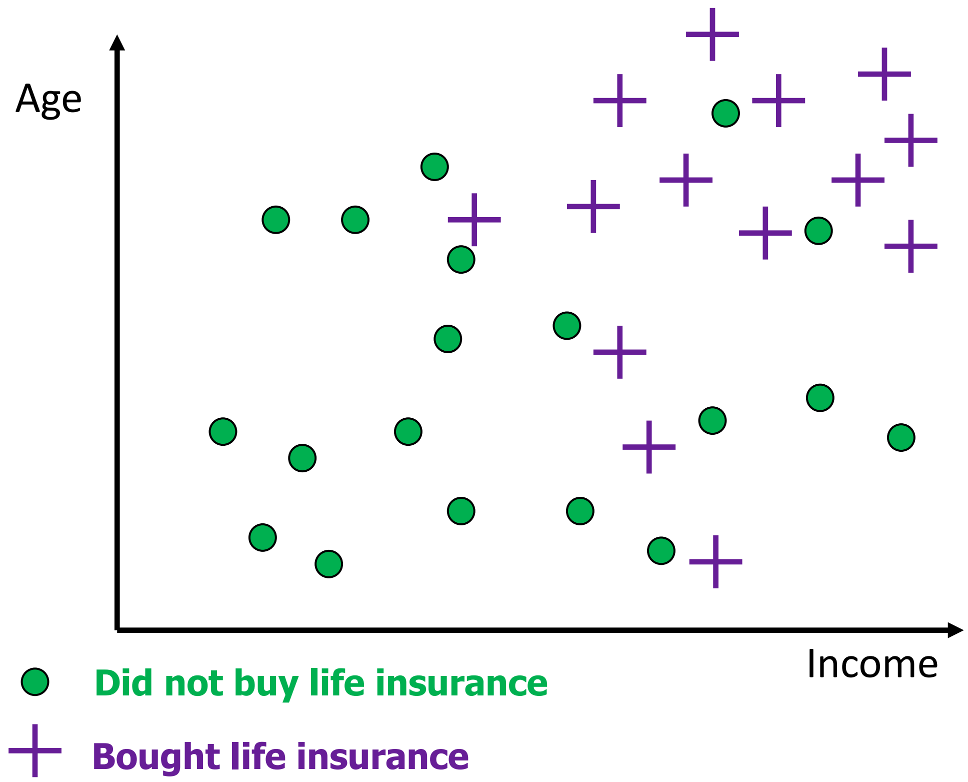
The many faces of classification: Classification / Probability Estimation / Ranking

Increasing difficulty

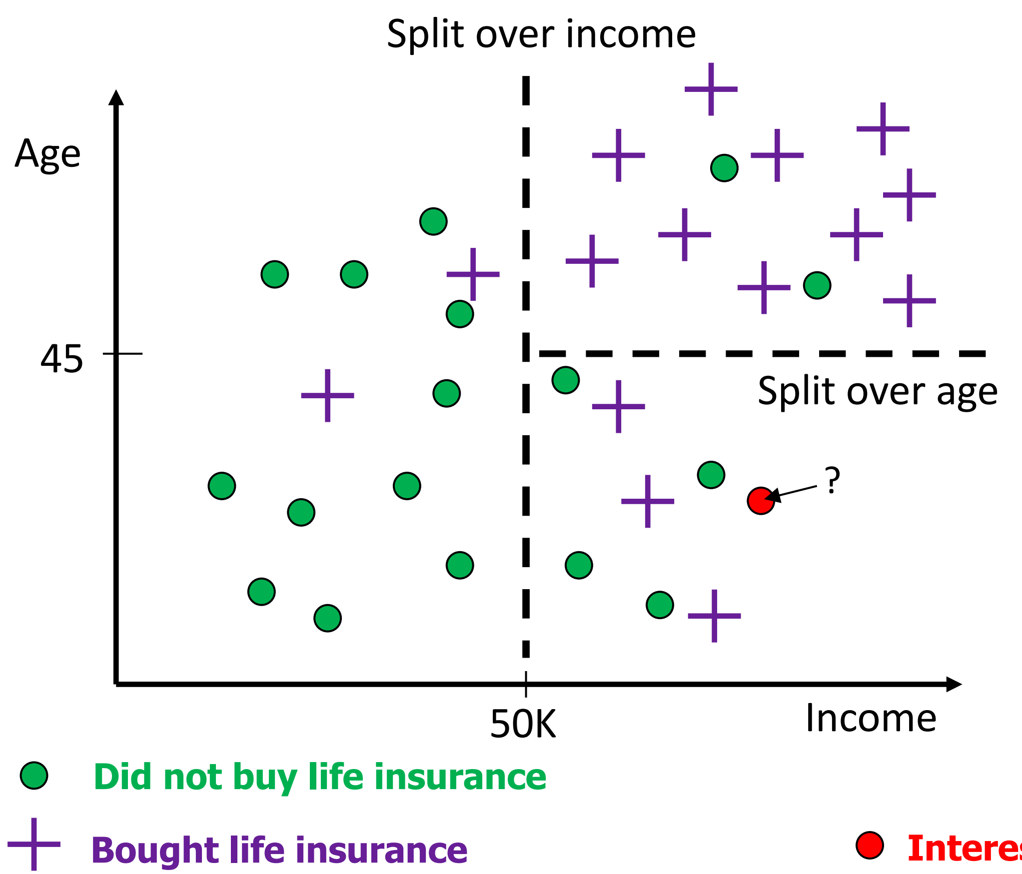
Classification → *Ranking* → *Probability*

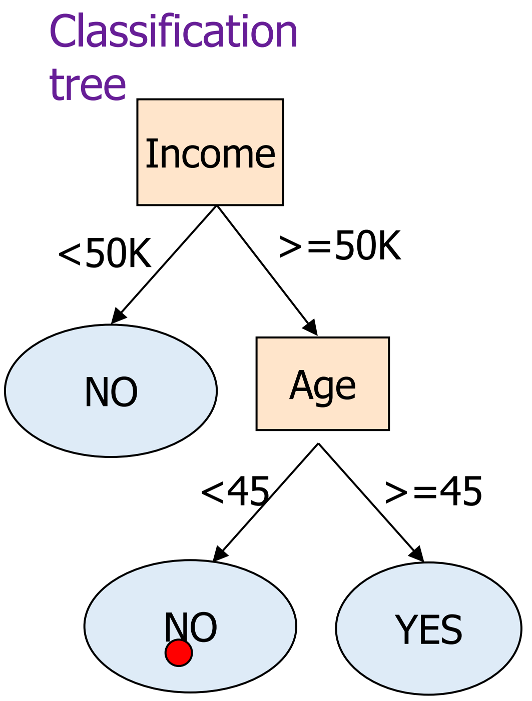
- Ranking:
  - business context determines the number of actions ("how far down the list")
  - cost/benefit is constant, unknown, or difficult to calculate
- Probability:
  - you can always rank / classify if you have probabilities!
  - cost/benefit is not constant across examples and known relatively precisely

Example


Age
Income
Did not buy life insurance
Bought life insurance

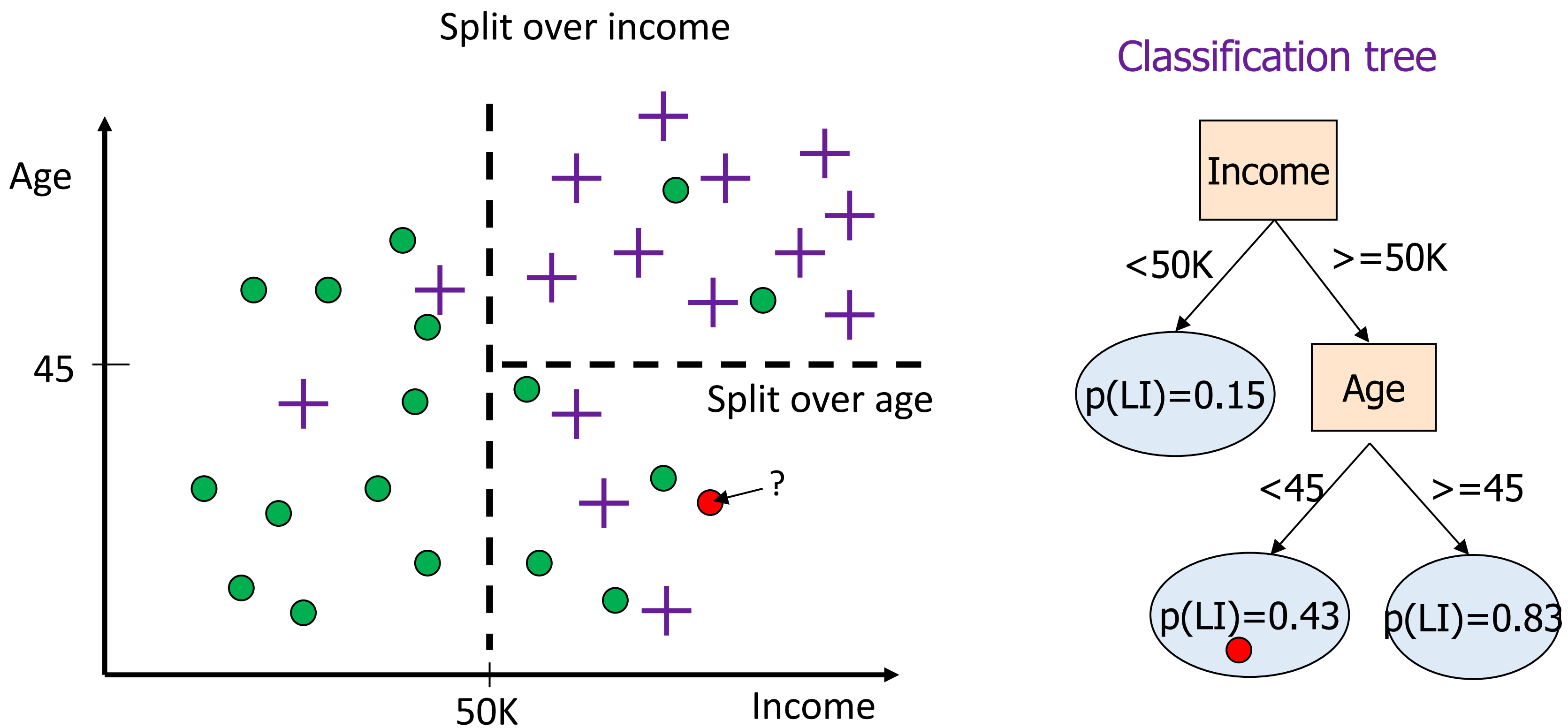
Example





Interested in LI? = NO

Example



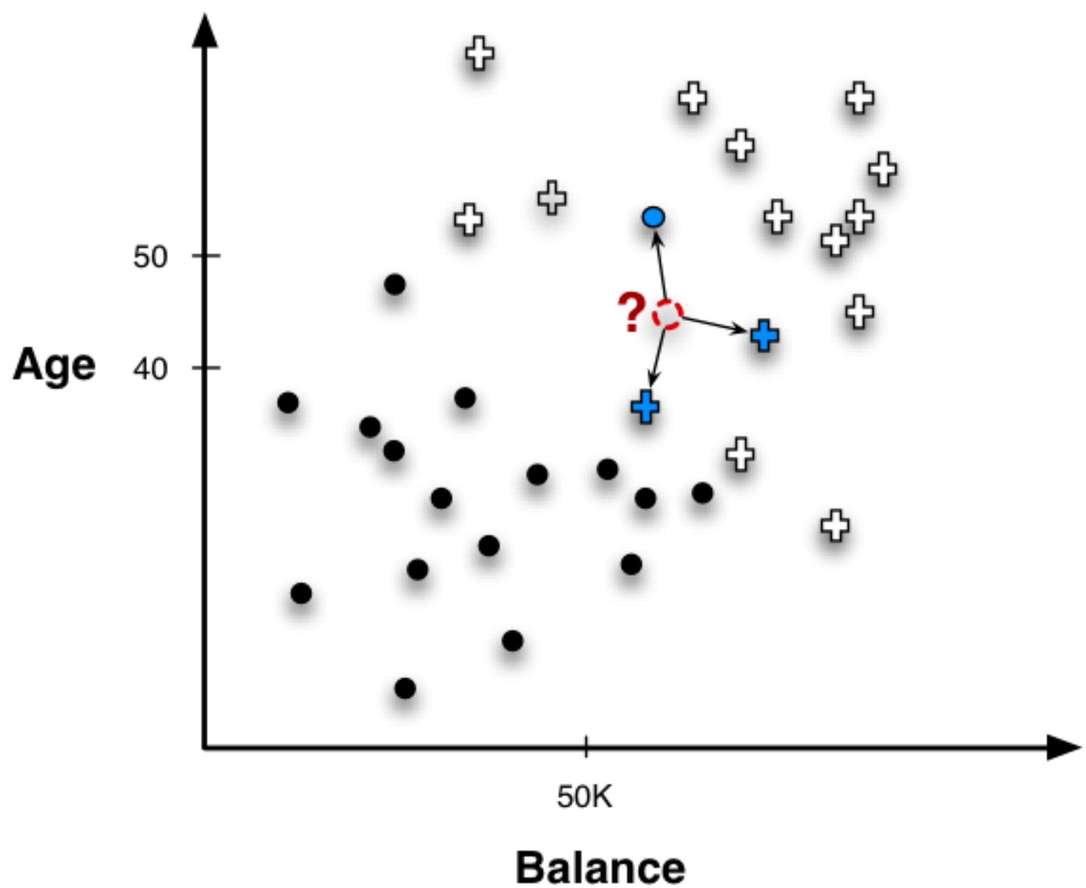
Did not buy life insurance

Bought life insurance

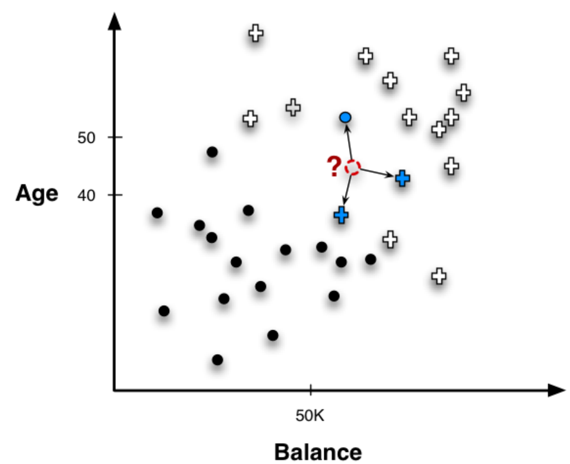
Interested in LI? = 3/7

Nearest neighbor classification

Nearest neighbors for predictive modeling



Nearest neighbor in classification



- Majority vote
- How many neighbors?
- Also consider the distance to vary the influence of the neighbors

Customer	Age	Income (1000s)	Cards	Response (target)	Distance from David
David	*37*	*50*	*2*	*?*	
John	35	35	3	Yes	$\sqrt{(35-37)^2+(35-50)^2+(3-2)^2} = 15.16$
Rachael	22	50	2	No	$\sqrt{(22-37)^2+(50-50)^2+(2-2)^2} = 15$
Ruth	63	200	1	No	$\sqrt{(63-37)^2+(200-50)^2+(1-2)^2} = 152.23$
Jefferson	59	170	1	No	$\sqrt{(59-37)^2+(170-50)^2+(1-2)^2} = 122$
Norah	25	40	4	Yes	$\sqrt{(25-37)^2+(40-50)^2+(4-2)^2} = 15.74$

Name	Distance	Similarity Weight	Contribution	Class
Rachael	15.0	0.004444	0.344	No
John	15.2	0.004348	0.336	Yes
Norah	15.7	0.004032	0.312	Yes
Jefferson	122.0	0.000067	0.005	No
Ruth	152.2	0.000043	0.003	No

Nearest neighbor classification

Given a labeled training set $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$

Example: the MNIST dataset of handwritten digits.

To classify a new instance x :

Find its nearest neighbor amongst the $x^{(i)}$, Return $y^{(i)}$

The data space

We need to choose a distance function.

Each image is 28 × 28 grayscale.
One option: Treat images as 784-dimensional vectors, and use Euclidean (l_2) distance:

$$\|x - x'\| = \sqrt{\sum_{i=1}^{784} (x_i - x'_i)^2}.$$

Summary:

- Data space $X = R^{784}$ with l_2 distance
- Label space $Y = \{0, 1, . . . , 9\}$

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points?

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**

  In general, **training error** is an overly optimistic predictor of future performance.

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.
- What test error would we expect for a random classifier?

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.
- What test error would we expect for a random classifier? **90%**

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.
- What test error would we expect for a random classifier? **90%**
- Test error of nearest neighbor: **3.09%.**

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.
- What test error would we expect for a random classifier? **90%**
- Test error of nearest neighbor: **3.09%.**

Examples of errors:

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.
- What test error would we expect for a random classifier? **90%**
- Test error of nearest neighbor: **3.09%.**

Examples of errors:

Ideas for improvement: (1) *k*-NN (2) better distance function.

K -nearest neighbor classification

Classify a point using the labels of its *k* nearest neighbors among the training points.

K -nearest neighbor classification

Classify a point using the labels of its k nearest neighbors among the training points.

MNIST:

k	1	3	5	7	9	11
Test error (%)	3.09	2.94	3.13	3.10	3.43	3.34

K -nearest neighbor classification

Classify a point using the labels of its *k* nearest neighbors among the training points.

MNIST:

k	1	3	5	7	9	11
Test error (%)	3.09	2.94	3.13	3.10	3.43	3.34

How to choose *k* in general?

K -nearest neighbor classification

Classify a point using the labels of its k nearest neighbors among the training points.

MNIST:

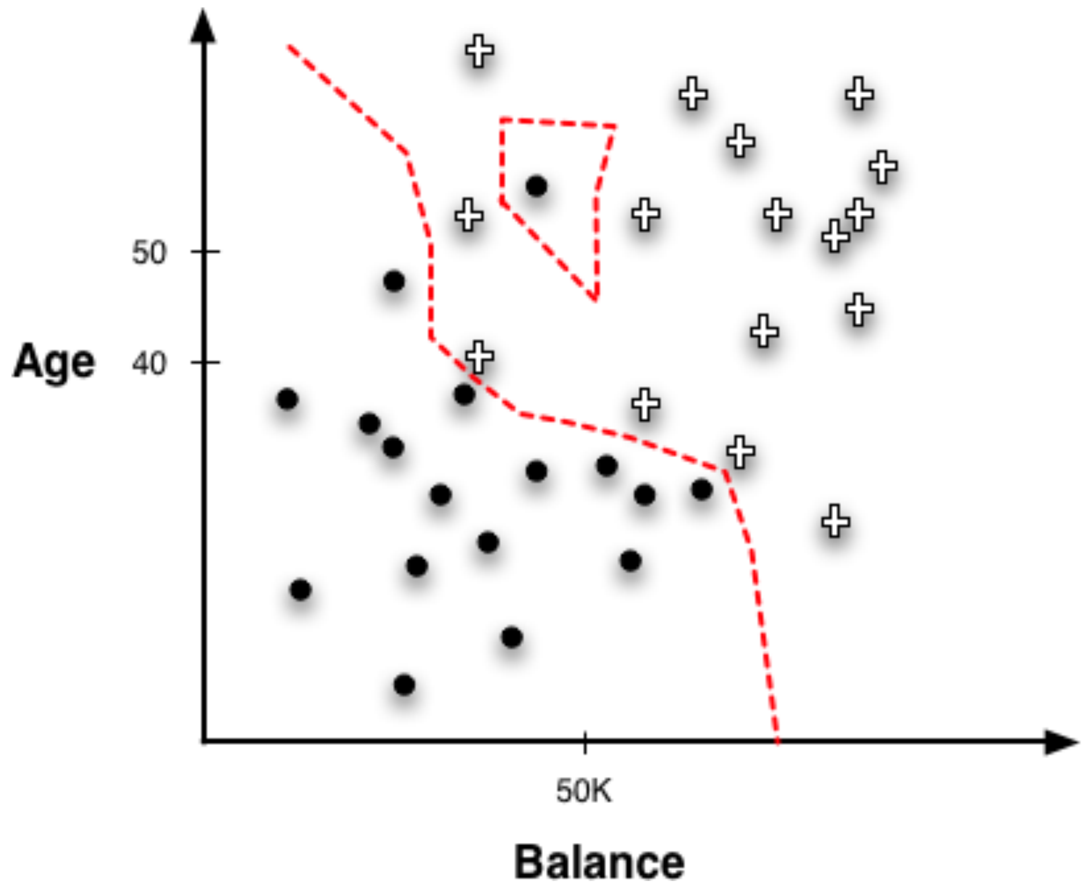
k	1	3	5	7	9	11
Test error (%)	3.09	2.94	3.13	3.10	3.43	3.34

How to choose k in general?

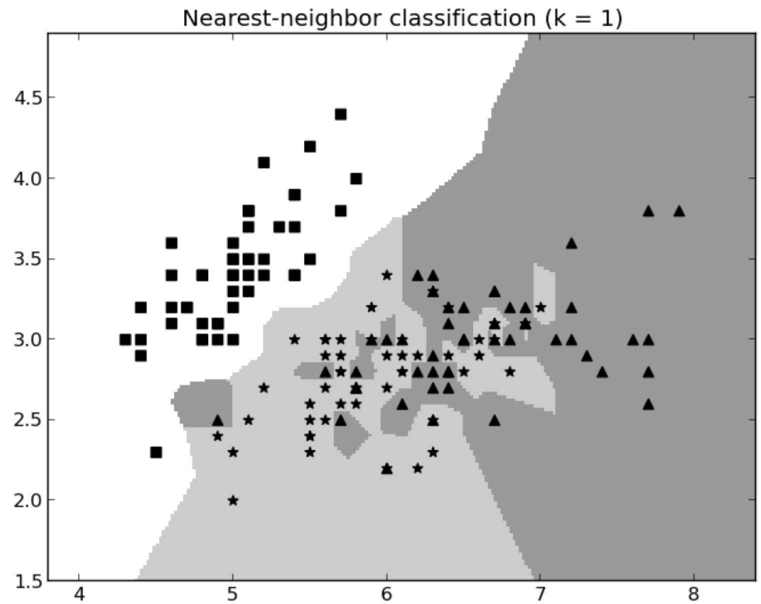
- Let $S \in Z^n$ be the training set, where $Z = X \times Y$ is the space of *labeled* points.
- Select a Hold-out set **V** – also called the Validation set
- Train the k-NN on (S – V) for some value of k
- Check the performance of this classifier on **V**
- Repeat for a different value of k
- Pick the best k (one with minimum test error)

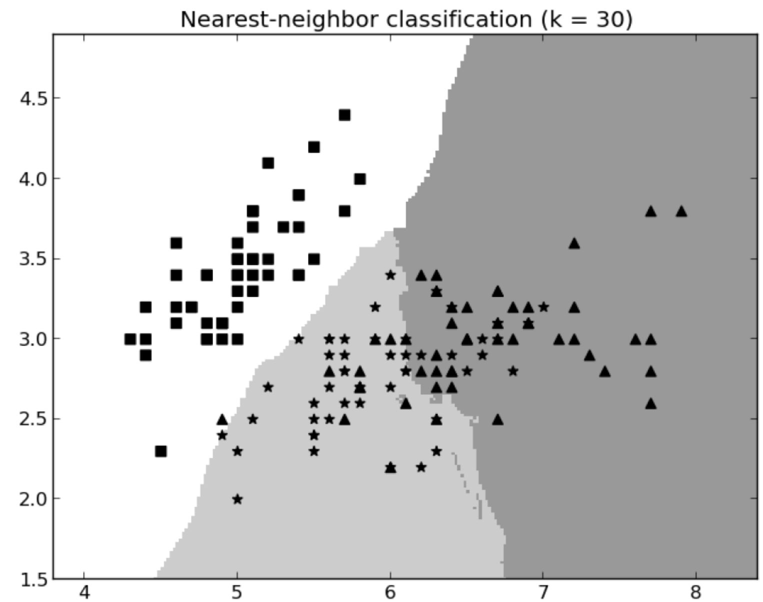
The above procedure can also be performed using Leave-one-out cross validation in which we choose one training point at a time as the validation set

Geometric Interpretation, Over-fitting, and Complexity



Nearest neighbor in classification





- Nearest neighbor classifiers follow very specific boundaries
- 1-NN strongly tends to overfit (k is a complexity parameter!)
- Use cross-validation or nested holdout testing

Better distance functions

Let *x* be an image. Consider an image *x'* that is just like *x* , but is either:

- shifted one pixel to the right, or
- Rotated slightly.

Then $\|x - x'\|$ could easily be quite large.

Better distance functions

Let *x* be an image. Consider an image *x'* that is just like *x* , but is either:

- shifted one pixel to the right, or
- Rotated slightly.

Then $\|x - x'\|$ could easily be quite large.

It makes sense to choose distance measures that are invariant under:

- Small translations and rotations. E.g. *tangent distance.*
- A broader family of natural deformations. E.g. *shape context*

Better distance functions

Let *x* be an image. Consider an image *x'* that is just like *x* , but is either:

- shifted one pixel to the right, or
- Rotated slightly.

Then $\|x - x'\|$ could easily be quite large.

It makes sense to choose distance measures that are invariant under:

- Small translations and rotations. E.g. *tangent distance.*
- A broader family of natural deformations. E.g. *shape context*

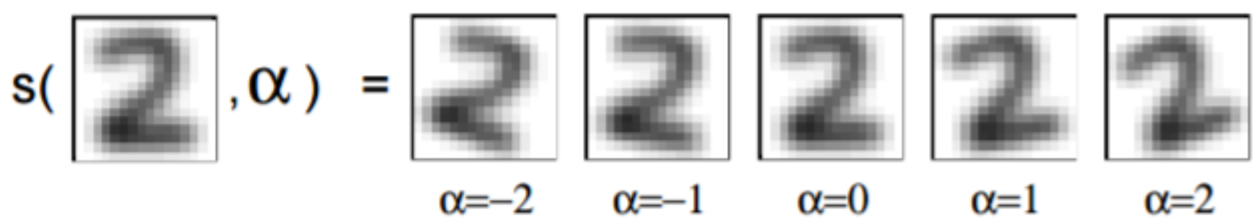
Test error rates:

ℓ_2	tangent distance	shape context
3.09	1.10	0.63

Better distance functions

Tangent Distance

- Is an invariant distance measure
- Especially effective for OCR
- Small transformations of certain image objects does not affect class membership



Refer: http://yann.lecun.com/exdb/publis/pdf/simard-00.pdf for more details

Better distance functions

Shape Context

- Is a feature descriptor in object recognition
- A way of describing shapes that allows for measuring shape similarity and the recovering of point correspondences
- pick n points on the contours of a shape

(a) (b)

References:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.441.6897&rep=rep1&type=pdf
https://en.wikipedia.org/wiki/Shape_context

Better distance functions

Shape Context

- Is a feature descriptor in object recognition
- A way of describing shapes that allows for measuring shape similarity and the recovering of point correspondences
- pick n points on the contours of a shape

(a) (b)

Are there other families of distance functions that are often useful?

ℓ_p norms

How can we measure the length of a vector in $\mathbb{R}^m$?
Usual choice is the *Euclidean norm*:

$$\|x\|_2 = \sqrt{\sum_{i=1}^{m} x_i^2}.$$

ℓ_p norms

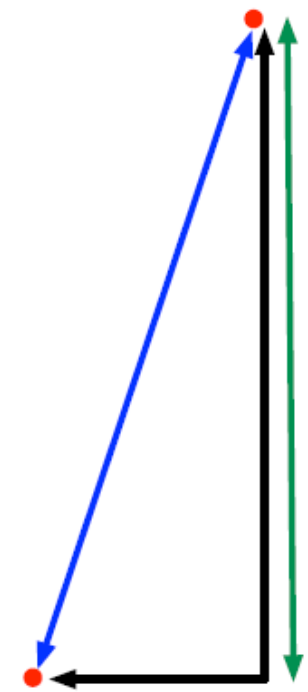
How can we measure the length of a vector in $\mathbb{R}^m$?
Usual choice is the *Euclidean norm*:

$$\|x\|_2 = \sqrt{\sum_{i=1}^{m} x_i^2}.$$

Generalization: For $p \geq 1$, the ℓ_p norm is

$$\|x\|_p = \left(\sum_{i=1}^{m} |x_i|^p \right)^{1/p}$$

- $p = 2$: Euclidean norm
- ℓ_1 norm: $\|x\|_1 = \sum_{i=1}^{m} |x_i|$
- ℓ_∞ norm: $\|x\|_\infty = \max_i |x_i|$



Quick quiz

Suppose data lie in R^p

1. What is the l_2 norm of the all-ones vector?

Quick quiz

Suppose data lie in R^p

1. What is the l_2 norm of the all-ones vector?
2. Suppose $\|x\|_1 = 1$.

What is the maximum value of the l_2 norm?

Quick quiz: Answers

Suppose data lie in R^p

1. What is the l_2 norm of the all-ones vector? **$1/(p)^{1/2}$**
2. Suppose $\|x\|_1 = 1$.

What is the maximum value of the l_2 norm? **One.**

Metric spaces

A more general notion is a *metric space*.

Metric spaces

A more general notion is a *metric space*.

Let X be the space in which data lie. A distance function $d : X \times X \rightarrow \mathrm{R}$ is a *metric* if it satisfies these properties:

- $d(x, y) \geq 0$ (non-negativity)
- $d(x, y) = 0$ if and only if $x = y$
- $d(x, y) = d(y, x)$ (symmetry)
- $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality)

Metric spaces

A more general notion is a *metric space*.

Let X be the space in which data lie. A distance function $d : X \times X \to \mathrm{R}$ is a *metric* if it satisfies these properties:

- $d(x, y) \geq 0$ (non-negativity)
- $d(x, y) = 0$ if and only if $x = y$
- $d(x, y) = d(y, x)$ (symmetry)
- $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality)

For instance:

- $\mathcal{X} = \mathbb{R}^m$ and $d(x, y) = \|x - y\|_p$
- $\mathcal{X} = \{\text{strings over some alphabet}\}$ and $d =$ edit distance.

Outline

- Nearest neighbor classification
- Statistical analysis
- Algorithmic analysis
- Some questions to ponder

The statistical learning framework

Why does training data help us develop a model that will work well on future data?

The statistical learning framework

Why does training data help us develop a model that will work well on future data? Because they come from the same source: the same underlying distribution.

The statistical learning framework

Why does training data help us develop a model that will work well on future data? Because they come from the same source: the same underlying distribution.

Model of reality: there is an (unknown) underlying distribution, Call it P, from which pairs (X, Y) are generated.

- Training points (x, y) come from this distribution.
- Future test points also come from this distribution.

The statistical learning framework

Why does training data help us develop a model that will work well on future data? Because they come from the same source: the same underlying distribution.

Model of reality: there is an (unknown) underlying distribution, Call it *P,* from which pairs *(X, Y)* are generated.

- Training points *(x, y)* come from this distribution.
- Future test points also come from this distribution.

We want a classifier

$$f : X \rightarrow Y$$

Which will do well on future data, i.e., do well on distribution *P.*

But we don't know *P*, so we treat the training data as a proxy for it.

The underlying distribution: example

A new strain of flu is highly contagious and seems to particularly affect children and the elderly. We want to predict whether patients waiting in an emergency room have this flu, based only on age and temperature.

The underlying distribution: example

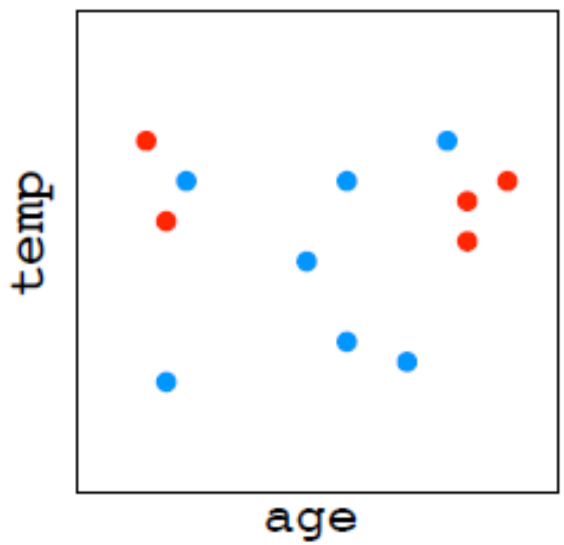
A new strain of flu is highly contagious and seems to particularly affect children and the elderly. We want to predict whether patients waiting in an emergency room have this flu, based only on age and temperature.

- $X = \mathbf{R}^2$ (age, temperature)
- $Y = \{0, 1\}$ (1 = flu)

The underlying distribution: example

A new strain of flu is highly contagious and seems to particularly affect children and the elderly. We want to predict whether patients waiting in an emergency room have this flu, based only on age and temperature.

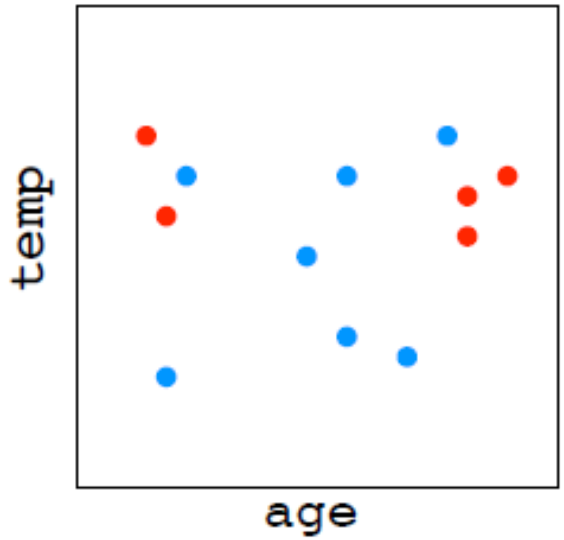
- $X = \mathbf{R}^2$ (age, temperature)
- $Y = \{0, 1\}$ (1 = flu)

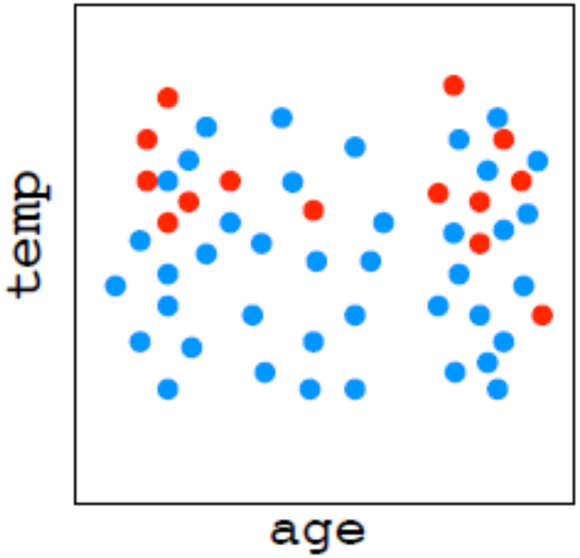


The underlying distribution: example

A new strain of flu is highly contagious and seems to particularly affect children and the elderly. We want to predict whether patients waiting in an emergency room have this flu, based only on age and temperature.

- $X = \mathbf{R}^2$ (age, temperature)
- $Y = \{0, 1\}$ (1 = flu)





As we get more and more samples, we get a better idea of the underlying distribution over X x Y

Three ways to sample

Three ways to draw a sample from a distribution P over pairs (x, y):

Three ways to sample

Three ways to draw a sample from a distribution P over pairs (x, y):

1. Draw the pair (x, y) in one go.

Three ways to sample

Three ways to draw a sample from a distribution P over pairs (x, y):

1. Draw the pair (x, y) in one go.
2. First draw y according to its *marginal distribution*.
   Then draw x according to its *conditional distribution* given y.

Three ways to sample

Three ways to draw a sample from a distribution P over pairs (x, y):

1. Draw the pair (x, y) in one go.
2. First draw y according to its *marginal distribution*.
   Then draw x according to its *conditional distribution* given y.
3. First draw x according to its marginal distribution.
   Then draw y according to its conditional distribution given x.

The underlying distribution

Reality ≡ the underlying distribution on pairs (X, Y)

The underlying distribution

Reality ≡ the underlying distribution on pairs (X, Y)

Can factor this distribution into two parts:

- The distribution of X. Call this μ.
- The distribution over labels Y given X. In the binary case ($y = \{0, 1\}$) this can be specified as $\eta(x) = \Pr(Y = 1 | X = x)$.

The underlying distribution

Reality ≡ the underlying distribution on pairs (X, Y)

Can factor this distribution into two parts:

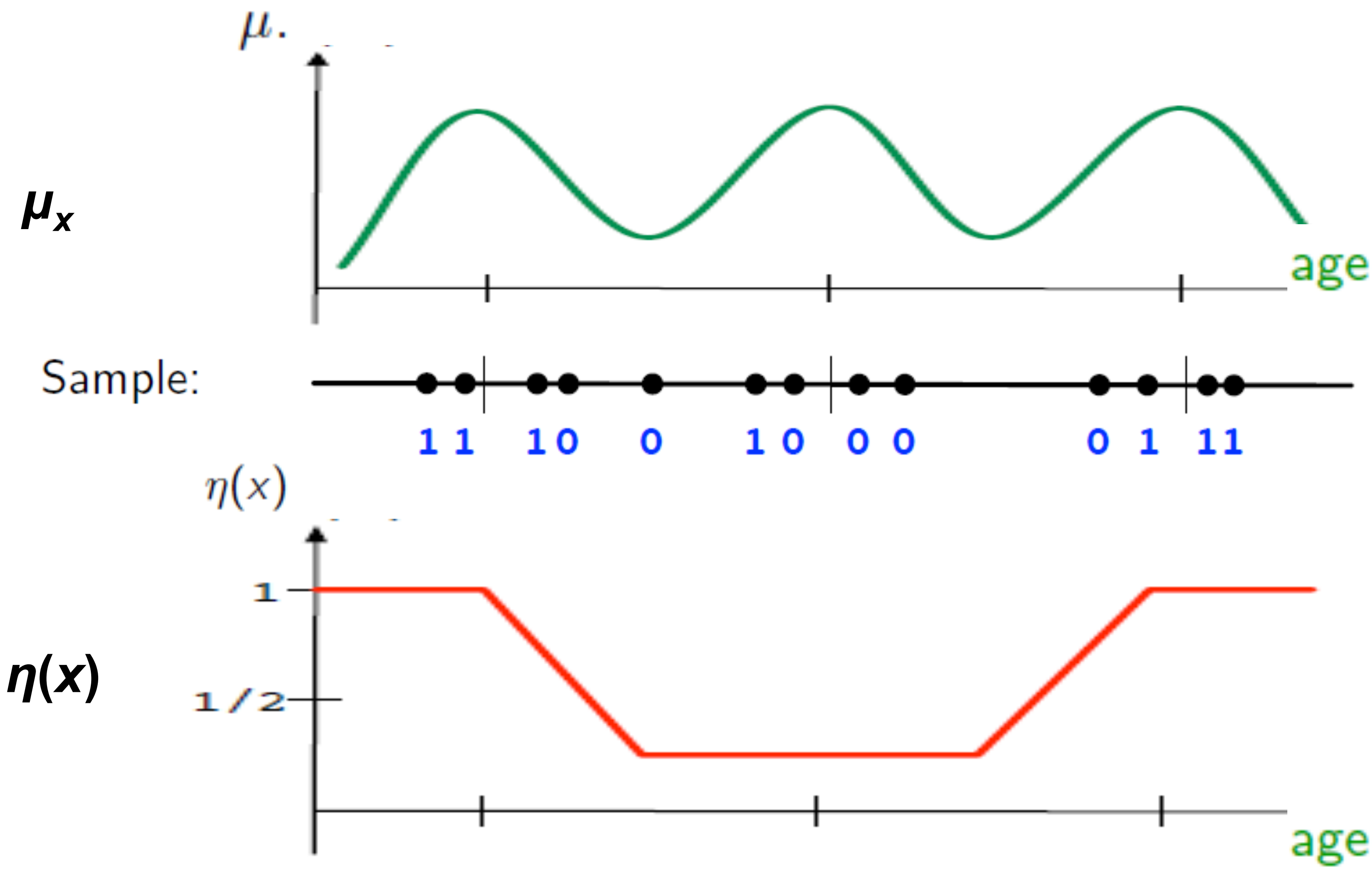
- The distribution of X. Call this μ.
- The distribution over labels Y given X. In the binary case ($y = \{0, 1\}$) this can be specified as $\eta(x) = \Pr(Y = 1 | X = x)$.

For instance, distribution of patients in a community:

x = age
y = 1 if visited doctor in the last year

The underlying distribution

Recall: x = age
y = 1 if visited doctor in the last year



Quick quiz

Suppose there are two possible labels, $Y \in \{0, 1\}$, that each occur 50% of the time. Data points X lie in R and have the following distribution:

- When $Y = 0$, points X are distributed uniformly in $[-2, 1]$.
- When $Y = 1$, points X are distributed uniformly in $[-1, 2]$.

What is the marginal distribution μ of X?

What is the conditional probability distribution $\eta(x) = \Pr(Y = 1|X = x)$?

Quick quiz: Answers

What is the marginal distribution μ of X ?

$$\mu_X = \begin{cases} \frac{1}{6} \text{ when } X \in [-2, -1] \\ \frac{1}{3} \text{ when } X \in [-1, 1] \\ \frac{1}{6} \text{ when } X \in [1, 2] \end{cases}$$

What is the conditional probability distribution $\eta(x) = \Pr(Y = 1|X = x)$?

$$\eta(x) = \begin{cases} 0 \text{ when } X \in [-2, -1] \\ \frac{1}{6} \text{ when } X \in [-1, 1] \\ 1 \text{ when } X \in [1, 2] \end{cases}$$

Statistical learning framework, cont'd

Let's look at the binary case, $y = \{0, 1\}$.

- There is an (unknown) underlying probability distribution on X from which *all* points are generated. Call this distribution μ.
- The label of any point x can, in general, be *stochastic*. It is a coin flip with bias $\eta(x) = \Pr(Y = 1|X = x)$.
- A classifier is a rule $h : X \rightarrow \{0, 1\}$. Its misclassification rate, or *risk*, is $R(h) = \Pr(h(X) \neq Y)$.

Statistical learning framework, cont'd

Let's look at the binary case, $y = \{0, 1\}$.

- There is an (unknown) underlying probability distribution on X from which *all* points are generated. Call this distribution μ.
- The label of any point x can, in general, be *stochastic*. It is a coin flip with bias $\eta(x) = \Pr(Y = 1|X = x)$.
- A classifier is a rule $h : X \to \{0, 1\}$. Its misclassification rate, or *risk*, is $R(h) = \Pr(h(X) \neq Y)$.

The Bayes-optimal classifier

$$h^*(x) = \begin{cases} 1 & \text{if } \eta(x) > 1/2 \\ 0 & \text{otherwise} \end{cases}$$

has minimum risk,

$$R^* = R(h^*) = \mathbb{E}_X \min(\eta(X), 1 - \eta(X)).$$

Quick quiz, cont'd

Suppose there are two possible labels, $Y \in \{0, 1\}$, that each occur 50% of the time. Data points X lie in R and have the following distribution:

- When $Y = 0$, points X are distributed uniformly in $[-2, 1]$.
- When $Y = 1$, points X are distributed uniformly in $[-1, 2]$.

What is the Bayes risk R^* ?

What is the risk of a classifier that always predicts 0?

Quick quiz, cont'd: Answers

What is the Bayes risk R^* ?

$R^*(h) = \Pr(h^*(X) \neq Y) = 1/6$

What is the risk of a classifier that always predicts 0?

$R^*(h) = \Pr(h^*(X) \neq Y) = 1/2$

Statistical theory of nearest neighbor

Let h_n be a classifier based on n training points from the underlying distribution.

- Its risk is $R(h_n) = \Pr(h_n(X) \neq Y)$.
- We say it is **consistent** if, as n grows to infinity, i.e., $R(h_n) \to R^*$

k-NN is consistent when both *k* and *n* are large but 1-NN is not!

Outline

- Nearest neighbor classification
- Statistical analysis
- Algorithmic analysis
- Some questions to ponder

Fast NN search

Naive search is $O(n)$ for training set of size n: very slow.

Fast NN search

Naive search is $O(n)$ for training set of size n: very slow.

Two popular approaches to fast nearest neighbor search, for data set $S \subset X$ and query q.

1. Locality sensitive hashing

   Collection of special hash functions $h_1, \ldots, h_m : X \to Z$.
   Search for nearest neighbor in

$$\bigcup_{i=1}^{m} \{x \in S : h_i(x) = h_i(q)\}$$

   This set is smaller than S, and is likely to contain the nearest neighbor of q.

2. Tree-based search

   Build tree structure on S, and use it to discard subsets of S that are far from a query q. Common options: k-d tree, PCA tree, cover tree.

Hash Function Example

If a hash has the tendency to put nearby data into the same bin, then it is a Locality Sensitive Hash

For example:

- Hamming distance
  - is the number of positions at which the corresponding symbols are different
- Given a million samples in $\mathbf{R}^p$, we will have a million feature vectors
- Take a base vector *b*
- Group samples with less Hamming distance (based on some threshold) to *b* together
- Group samples with large Hamming distance to *b* together
- You may have multiple *b* vectors to make a more comprehensive system

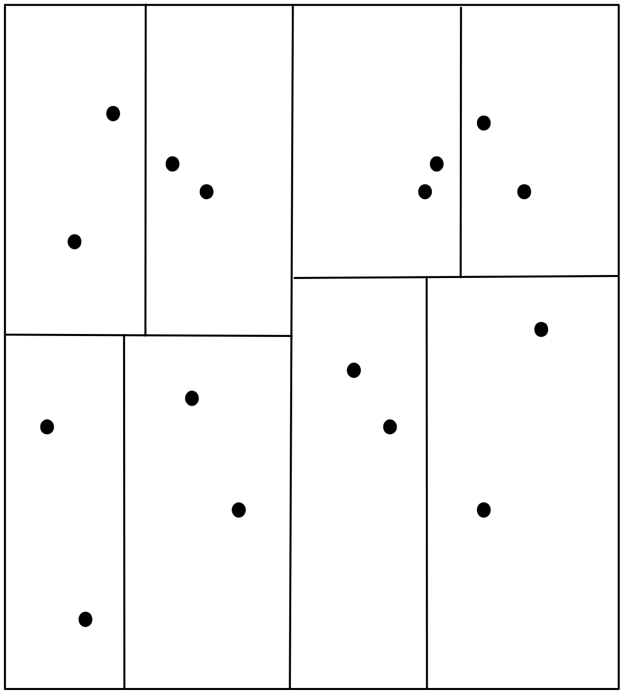
K -d trees for NN search

A hierarchical, rectilinear spatial partition.

For data set $S \subset \mathrm{R}^p$

- Pick a coordinate $1 \leq i \leq p$.
- Compute $v = \text{median}(\{x_i : x \in S\})$
- Split S into two halves:

$$S_L = \{x \in S : x_i < v\}$$
$$S_R = \{x \in S : x_i \geq v\}$$



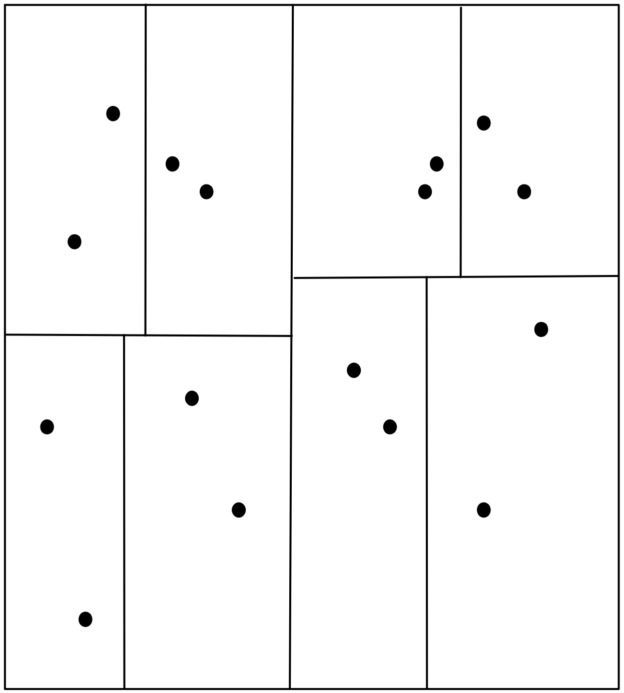
K -d trees for NN search

A hierarchical, rectilinear spatial partition.

For data set $S \subset R^p$

- Pick a coordinate $1 \leq i \leq p$.
- Compute $v = \text{median}(\{x_i : x \in S\})$
- Split S into two halves:

$$S_L = \{x \in S : x_i < v\}$$
$$S_R = \{x \in S : x_i \geq v\}$$



Two types of search, given a query $q \in \mathbf{R}^p$

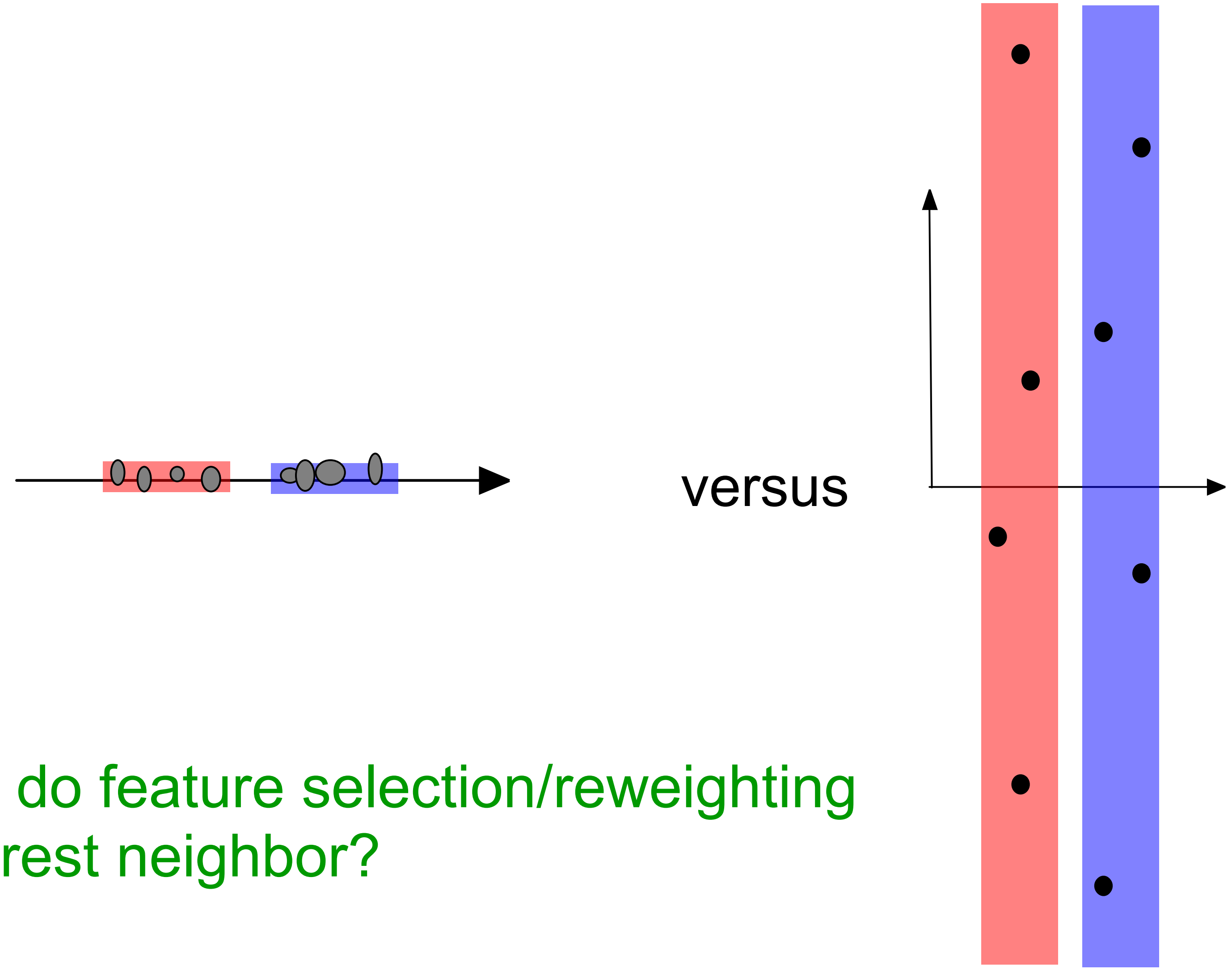
- *Defeatist search*: Route q to a leaf cell and return the NN in that cell. This might not be the true NN.
- *Comprehensive search*: Grow the search region to other cells that cannot be ruled out using the triangle inequality.

Sensitivity to noise

Adding a single sufficiently noisy feature can wreak havoc with nearest neighbor classifiers.

Sensitivity to noise

Adding a single sufficiently noisy feature can wreak havoc with nearest neighbor classifiers.



How to do feature selection/reweighting for nearest neighbor?