Naïve Bayes II

Semester 1, 2023

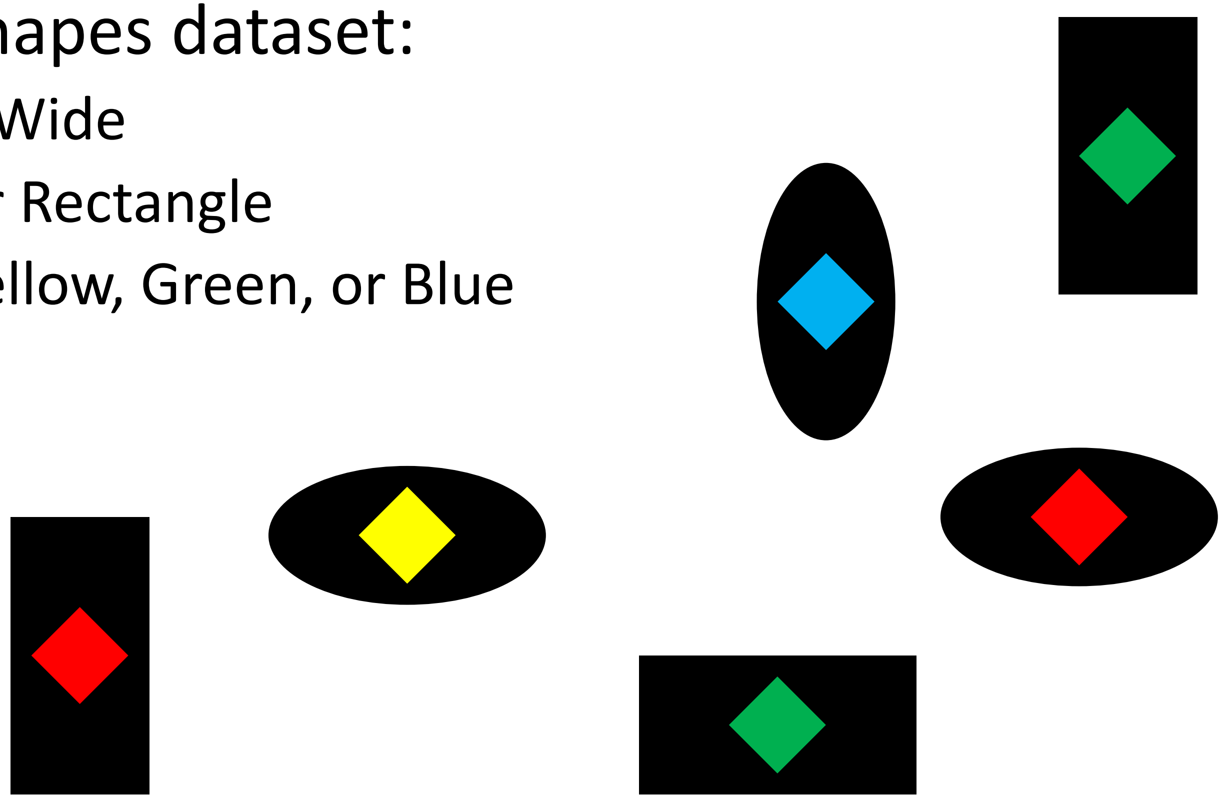
Kris Ehinger

Outline

- Naïve Bayes in practice
- Continuous data
- Converting data types
- Naïve Bayes with continuous data

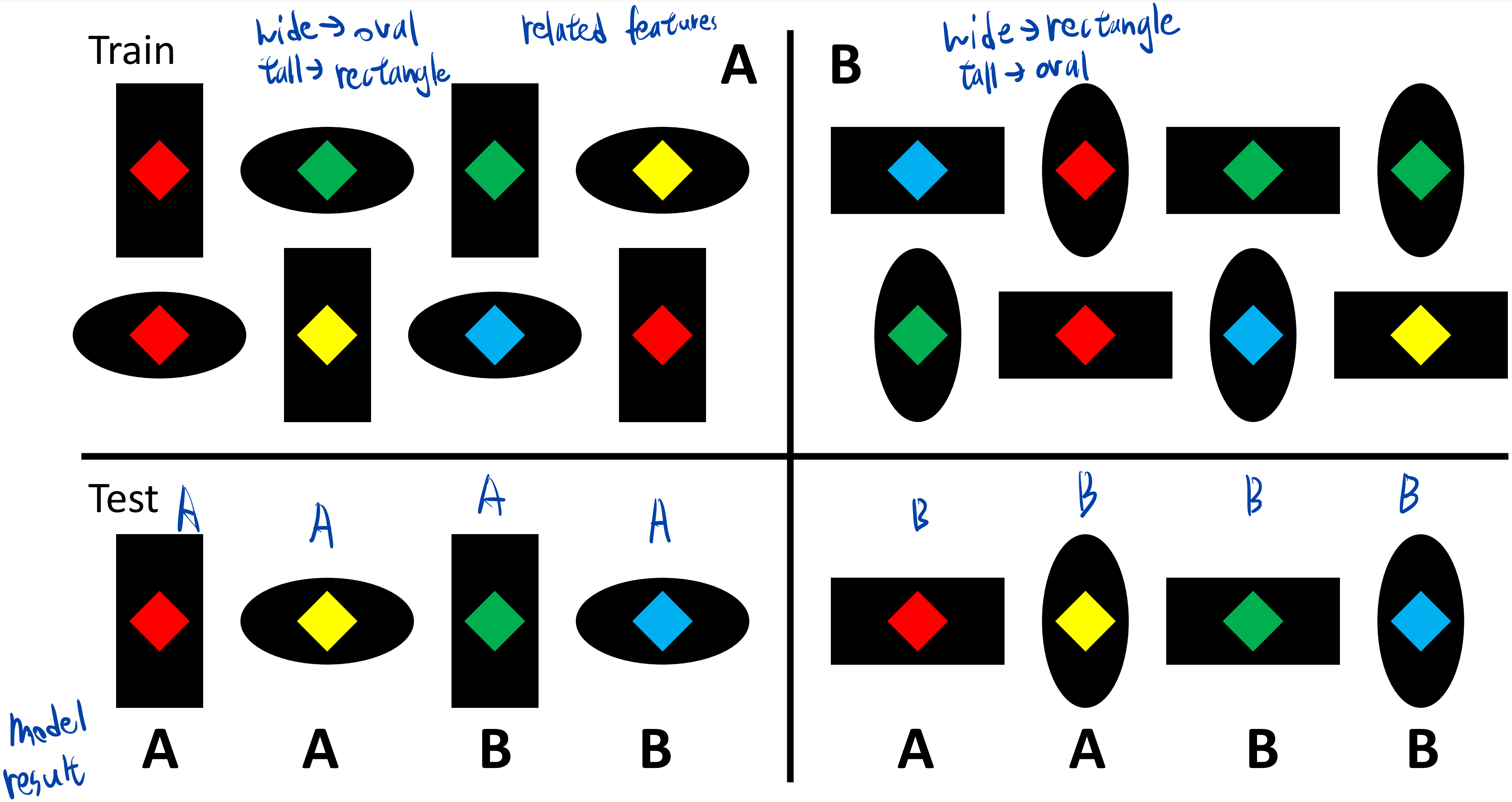
Naïve Bayes example

- Simple shapes dataset:
  - Tall or Wide
  - Oval or Rectangle
  - Red, Yellow, Green, or Blue



Tall, Rectangle, Red   Wide, Rectangle, Green

Naïve Bayes example



small training dataset

assumption: independent features

Naïve Bayes in practice

- Practical considerations
  - How to handle zero probabilities? ← Previous lecture
  - How to handle missing values?
  - How to handle small probabilities (underflow errors)
  - What if attributes aren’t conditionally independent?

Missing values

- What if an instance is missing some attribute?
- Missing values at test can simply be ignored – compute the likelihood of each class from the non-missing values
- Missing values in training can also be ignored – don't include them in the attribute-class counts, N and the probabilities will be based on the non-missing values

Avoiding underflow

- Multiplying a lot of numbers (0,1] can lead to underflow – numbers smaller than the computer can represent
- Common workaround: **log transformation**
  - take the log() of each probability and sum instead of multiplying

multiplying

$$\hat{c} = \arg\max_{c_j \in C} P(c_j) \prod_i P(x_i|c_j)$$

take log

$$= \arg\max_{c_j \in C} \left[\log\big(P(c_j)\big) + \sum_i \log\big(P(x_i|c_j)\big) \right]$$

Independence assumption

- Independence assumption is usually wrong
- But Naïve Bayes usually works anyway. Why?
  - We don’t need a perfect estimate of $P(c|T)$ for every class – we just need to know which class is most likely
  - Ignoring the fact that some attributes are correlated tends to make all the class probabilities higher, but doesn’t typically change their rank
  - Naïve Bayes is also robust to small errors in estimating $P(x_i|c_j)$ – these can change class probabilities but typically don’t change class rank

Naïve Bayes

- Strengths:
  - Simple to build, fast
  - Computations scale well to high-dimensional datasets (1000s of attributes)
  - Explainable – generally easy to understand why the model makes the decision it does
- Weaknesses:
  - Inaccurate when there are many missing liklihood $P(x_i|c_j)$ values
  - Conditional independence assumption becomes problematic for complex systems not all dataset

Continuous data

Continuous attributes

- Naïve Bayes (as discussed last lecture) assumes nominal data
  - What happens if we have continuous data?

Continuous attributes

- How to compute probabilities $P(x_i|c_j)$?

Wind	Temp	Rain?
north	32.1	no
east	18.4	**yes**
east	19.5	no
north	23.5	no
north	20.3	**yes**
east	19.7	**yes**

P(temp = 32.1|Rain = no) = 1/3
P(temp = 19.5|Rain = no) = 1/3
P(temp = 23.5|Rain = no) = 1/3
P(temp = 18.4|Rain = yes) = 1/3
P(temp = 20.3|Rain = yes) = 1/3
P(temp = 19.7|Rain = yes) = 1/3

This doesn't look right…

Converting data types

Data types

- The input to a machine learning system consists of instances, which have:
  - Attributes
  - Class labels (if supervised)
- Attributes and class labels can be:
  - Nominal / categorical
  - Ordinal
  - Continuous / numeric

Attribute types

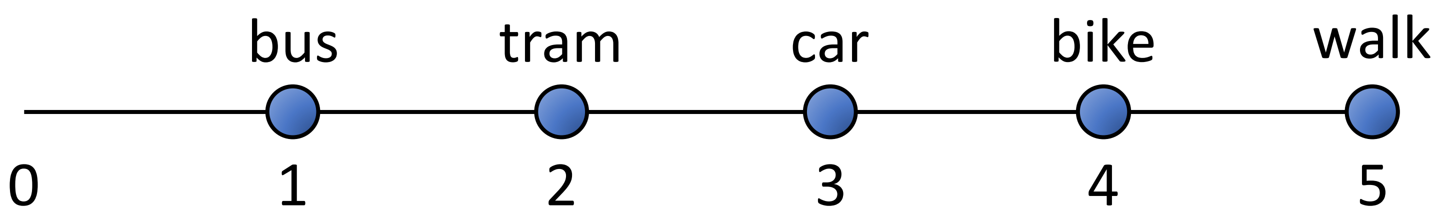
- Machine learning algorithms typically assume attributes have a particular data type
- Algorithms that assume nominal attributes:
  - Naïve Bayes (as described in last lecture)
  - Decision trees
- Algorithms that assume numeric attributes:
  - Support vector machines (SVM)
  - Perceptron, neural network

Attribute types

- What if we have attributes of the wrong type for a given model?
- Options:
  - Discard those attributes
  - Convert the attributes to match the model
  - Change the model assumptions to match the data

Nominal -> numeric

- Option 1: Convert category names to numbers
  - Attribute: mode of transport
  - Nominal: “bus,” “tram,” “car,” “bike,” “walk” not ordinal
  - Numeric: 0, 1, 2, 3, 4
- Problem: creates an artificial ordering when no order exists, makes some categories seem more/less similar to each other

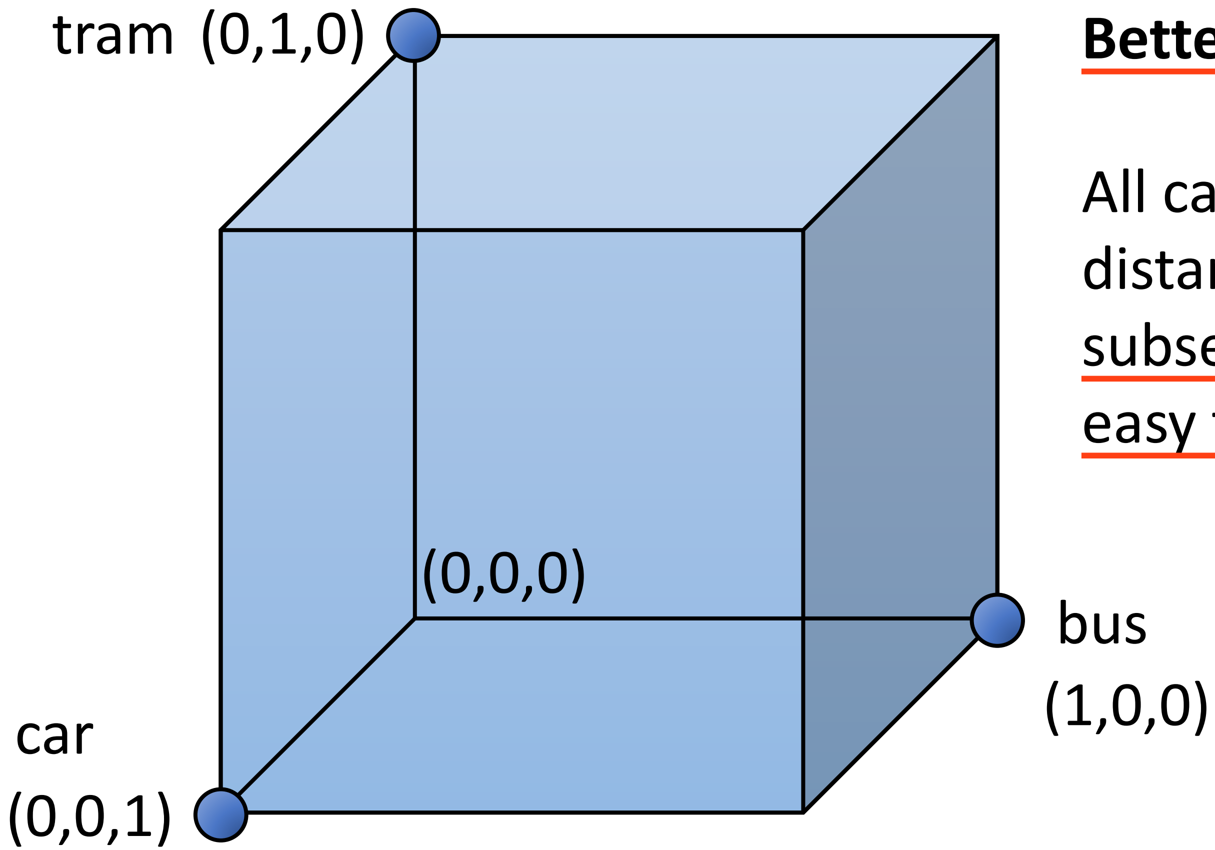


Nominal -> numeric

- Option 2: One-hot encoding
  - Attribute with *m* possible values -> *m* boolean attributes
  - “bus” = [1, 0, 0, 0, 0]
  - “tram” = [0, 1, 0, 0, 0]
  - “car” = [0, 0, 1, 0, 0]
  - “bike” = [0, 0, 0, 1, 0]
  - “walk” = [0, 0, 0, 0, 1]
- Best way to represent nominal values in a continuous space, but increases dimensionality of the space

1 attribute with m values $\Rightarrow$ m attributes

Visualisation of option 2



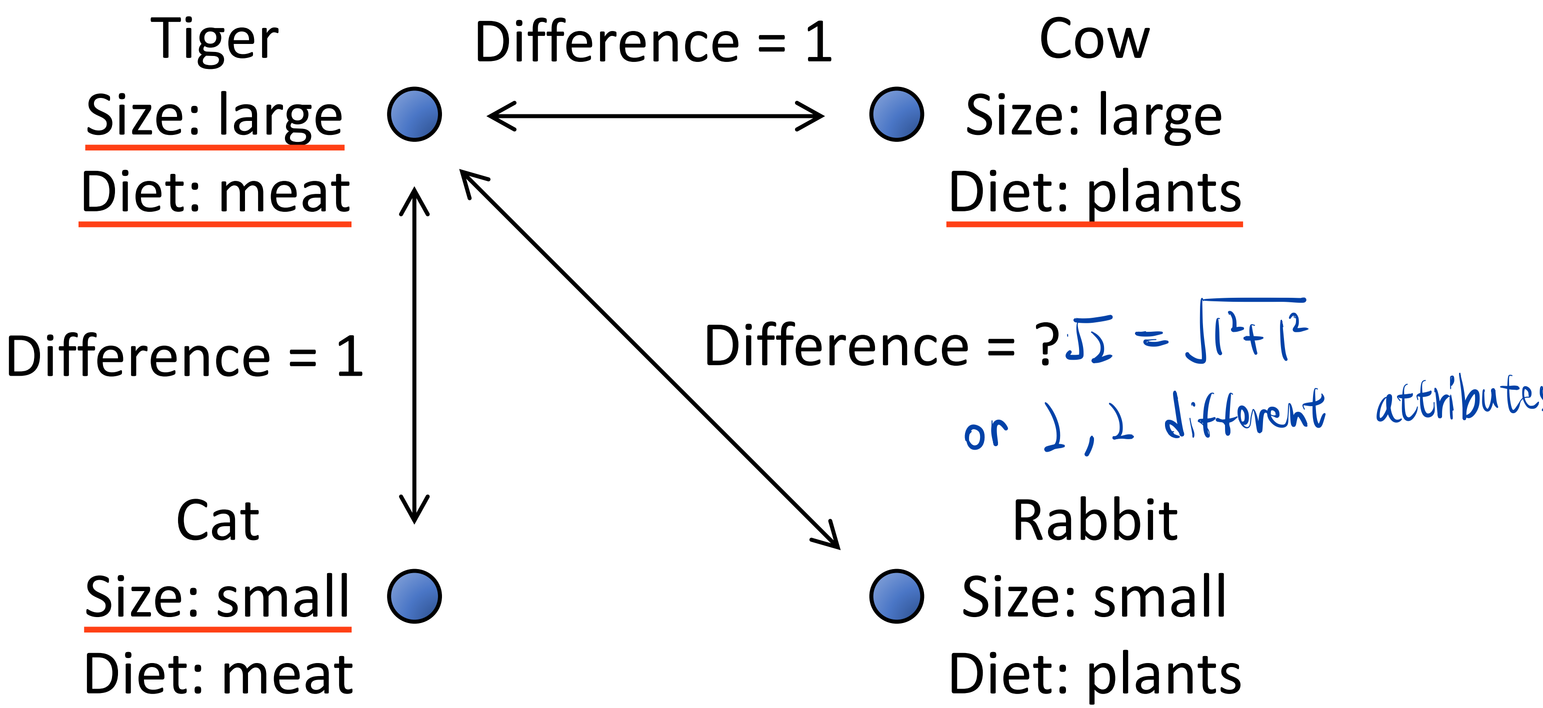
Better encoding:

All categories equal distance apart; any subset is equally easy to group

Computing distances

- How to compute distances between nominal attributes?
- Consider these boolean attributes for an animal classifier:
  - Size: large/small
  - Diet: meat/plants

Computing distances (difference)



Computing distances

- How to compute distances between nominal attributes?
  - Euclidean distance
    - If A and B differ on N attributes, $dist_E(A, B) = \sqrt{N}$

$$dist_E(A, B) = \sqrt{\sum_i (a_i - b_i)^2}$$

  - Hamming distance
    - If A and B differ on N attributes, $dist_H(A, B) = N$

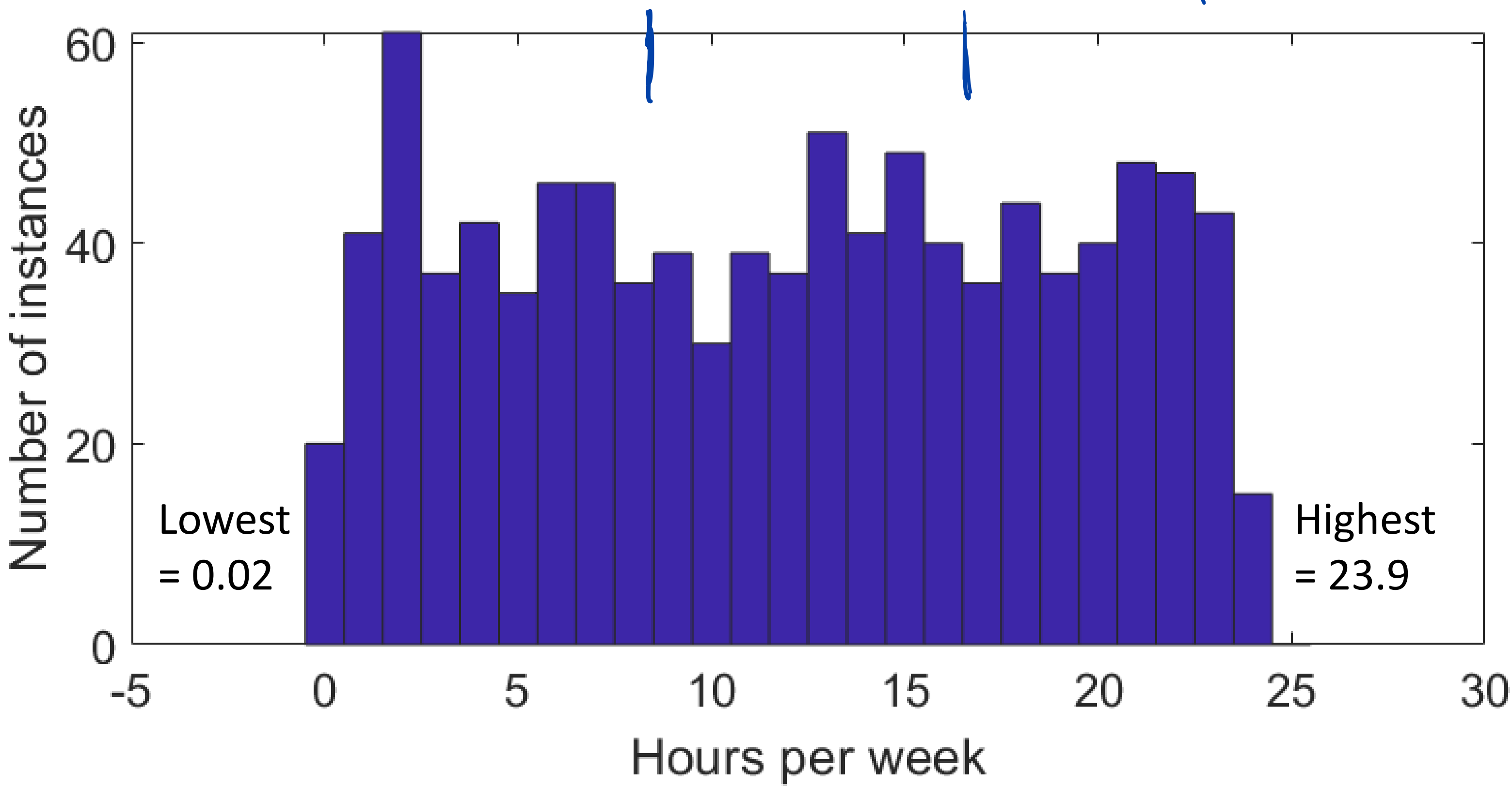
$$dist_H(A, B) = \sum_i \begin{cases} 0, & a_i == b_i \\ 1, & otherwise \end{cases}$$

Numeric -> nominal

- **Discretisation** is the translation of continuous numeric attributes to discrete nominal attributes
  - Example: map temperatures to “hot,” “mild,” “cool”
- Discretisation is generally a two-step process:
  - Decide how many nominal values (= intervals) onto which you will map the numeric values #bins
  - Decide where to place the boundaries for these intervals

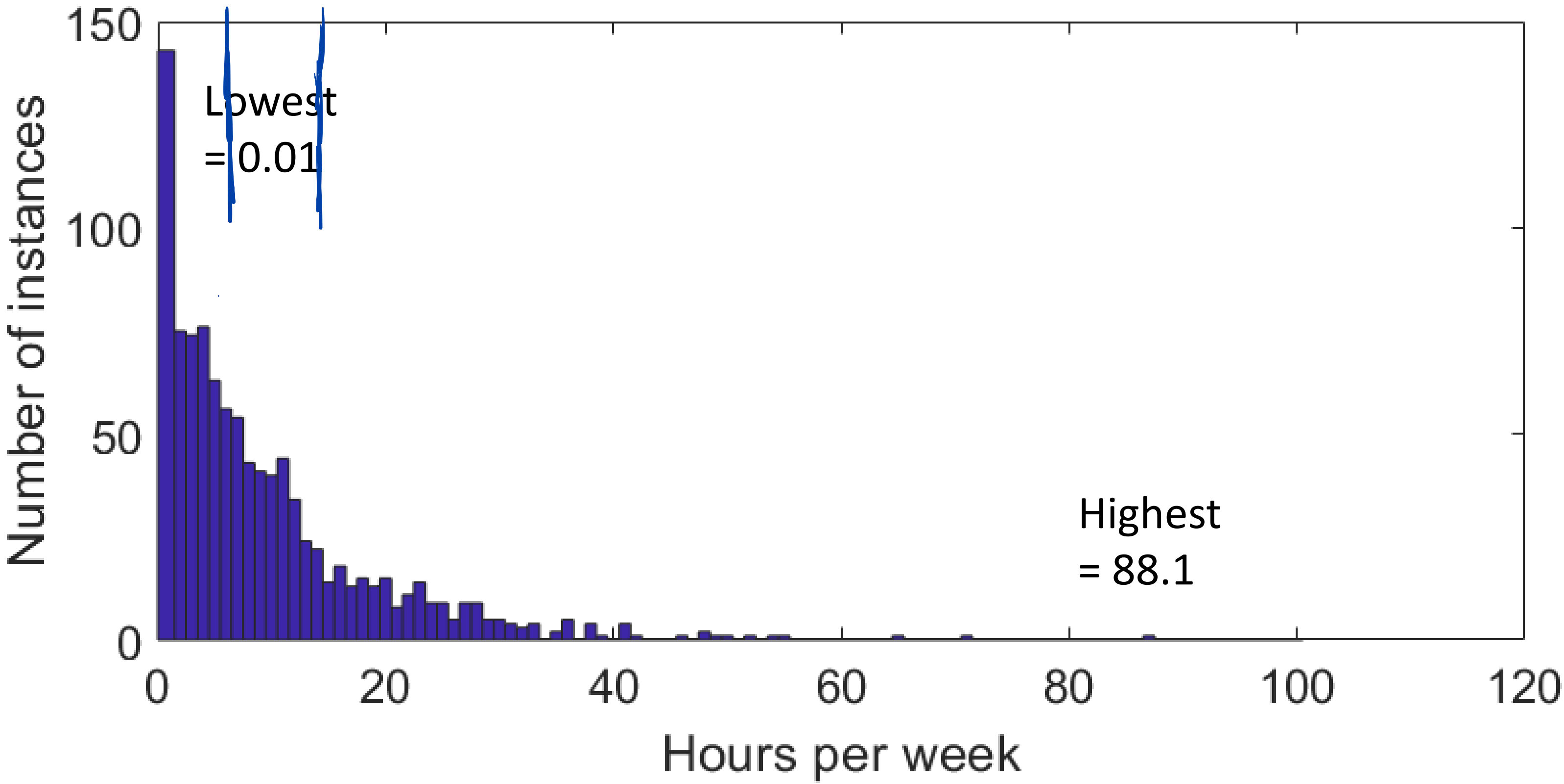
Discretisation example 1

You measured app usage (hours/week) from 1000 users. How would you recode this attribute into 3 levels: low, medium, high? equal-width



Discretisation example 2

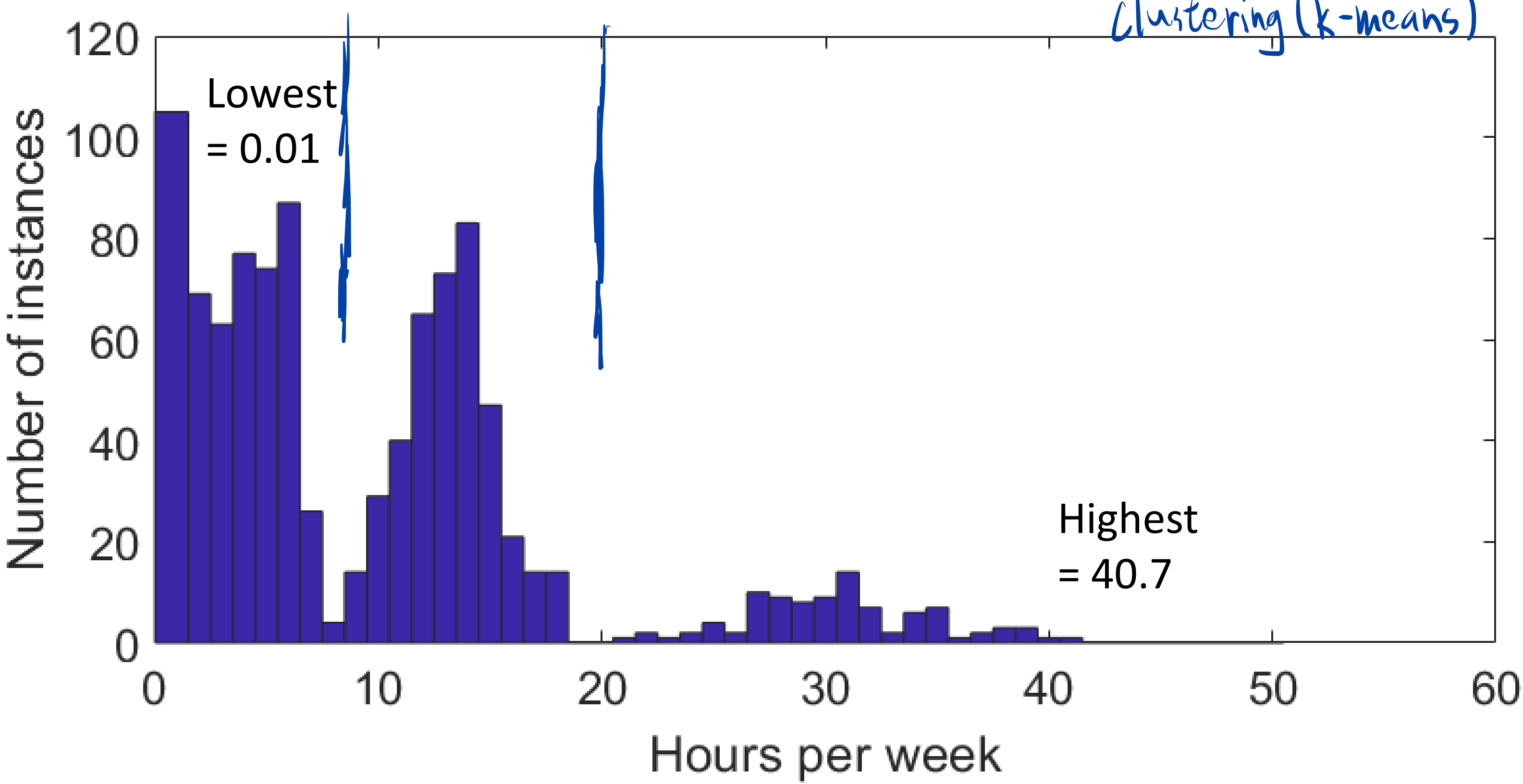
You measured app usage (hours/week) from 1000 users. How would you recode this attribute into 3 levels: low, medium, high? equal frequency



Discretisation example 3

You measured app usage (hours/week) from 1000 users. How would you recode this attribute into 3 levels: low, medium, high?

clustering (k-means)



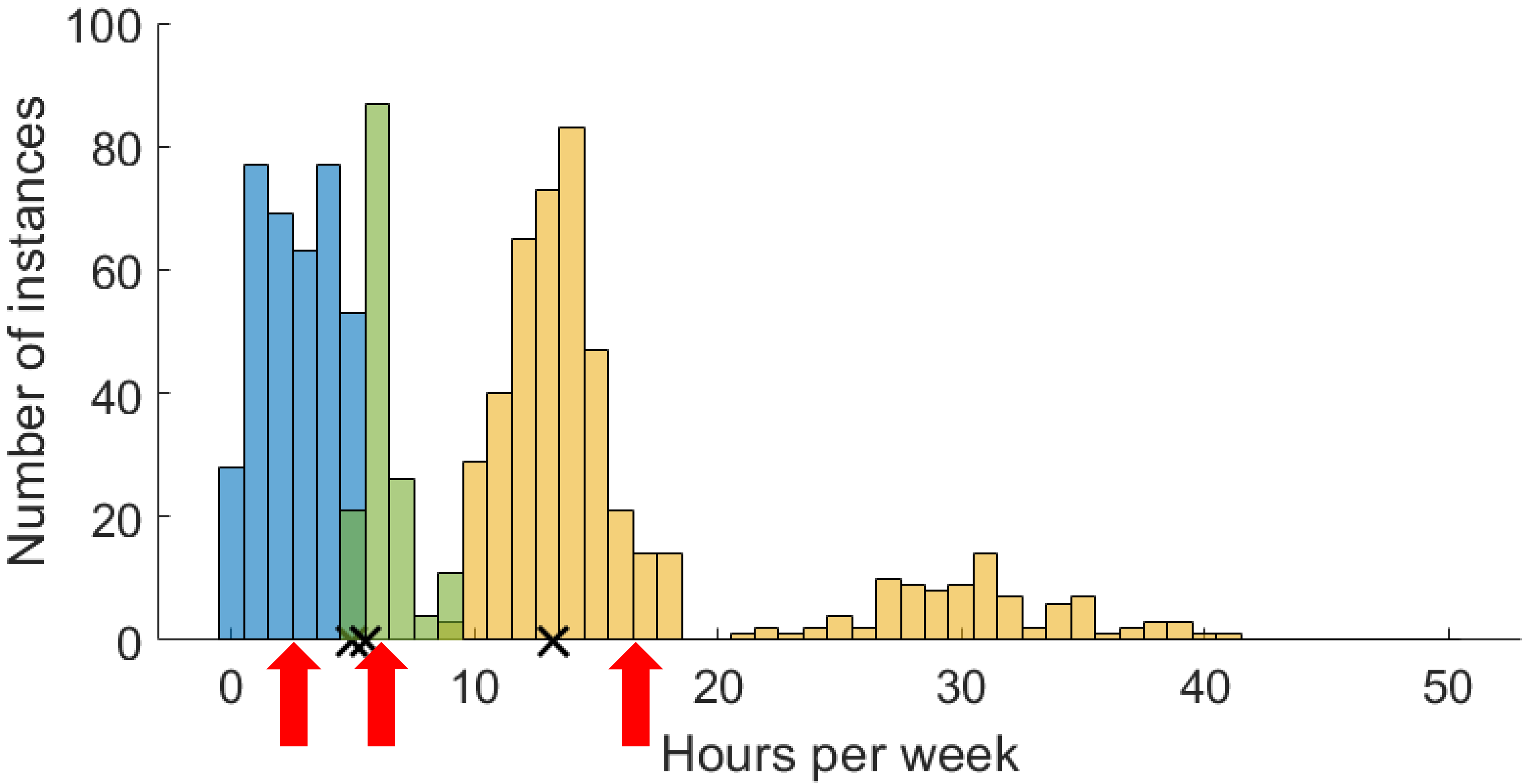
Discretisation options

- e.g. 1 **Equal-width** discretisation – find min/max of range, partition into n bins of width (max-min)/n
- e.g. 2 **Equal-frequency** discretisation – sort values, find breakpoints that produce n bins with (approximately) equal numbers of items
- Disadvantages:
  - Arbitrary group boundaries
  - Equal-width is sensitive to outliers, equal-frequency is sensitive to sample bias
  - User must choose n

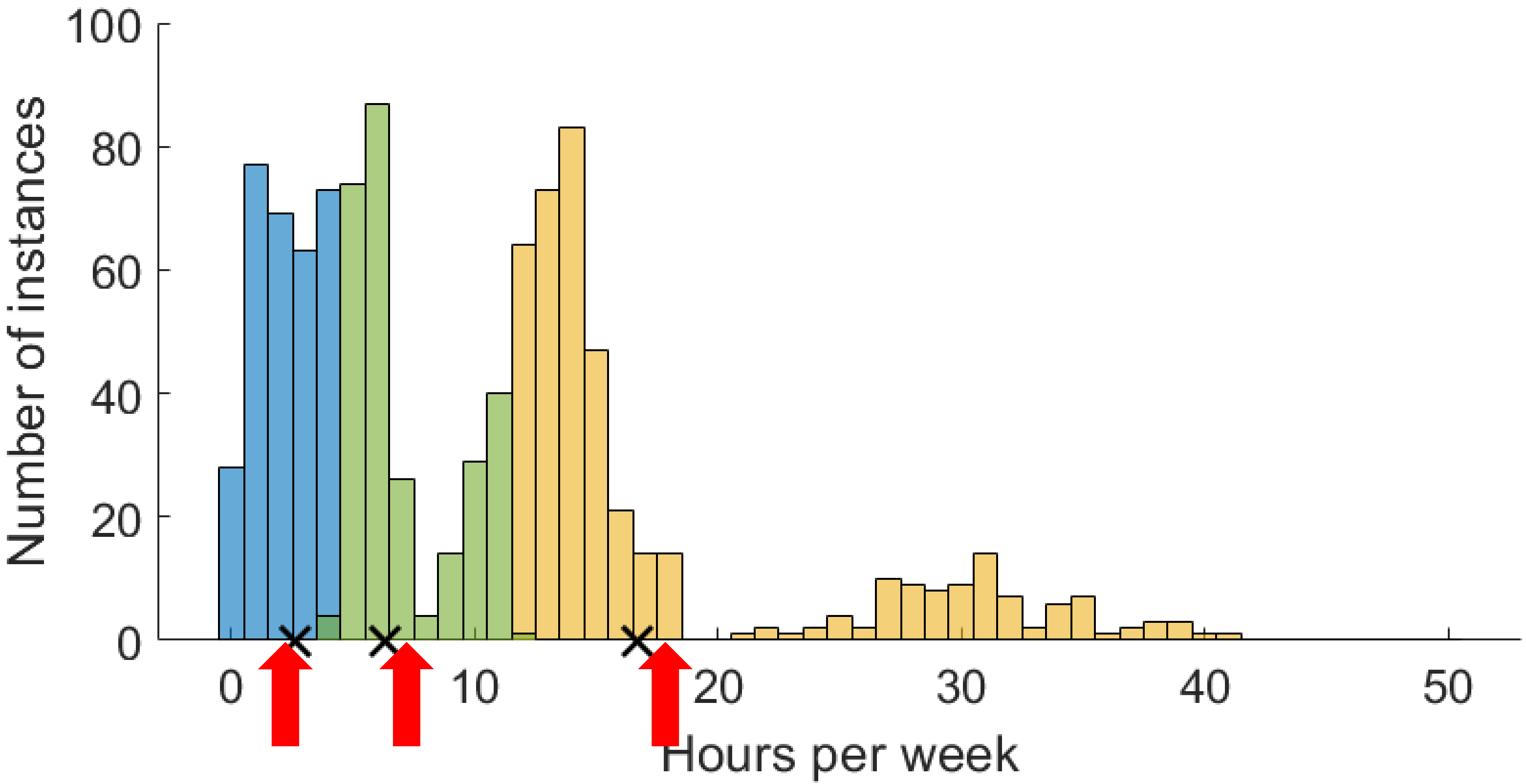
Discretisation options

- Use some clustering method to discover natural breakpoints within your data (e.g., k-means clustering)
- Disadvantages
  - More complicated than equal-width or -frequency
  - If data doesn't have natural "groups," k-means result is the same as equal-width discretisation
  - Sensitive to outliers
  - User must choose n

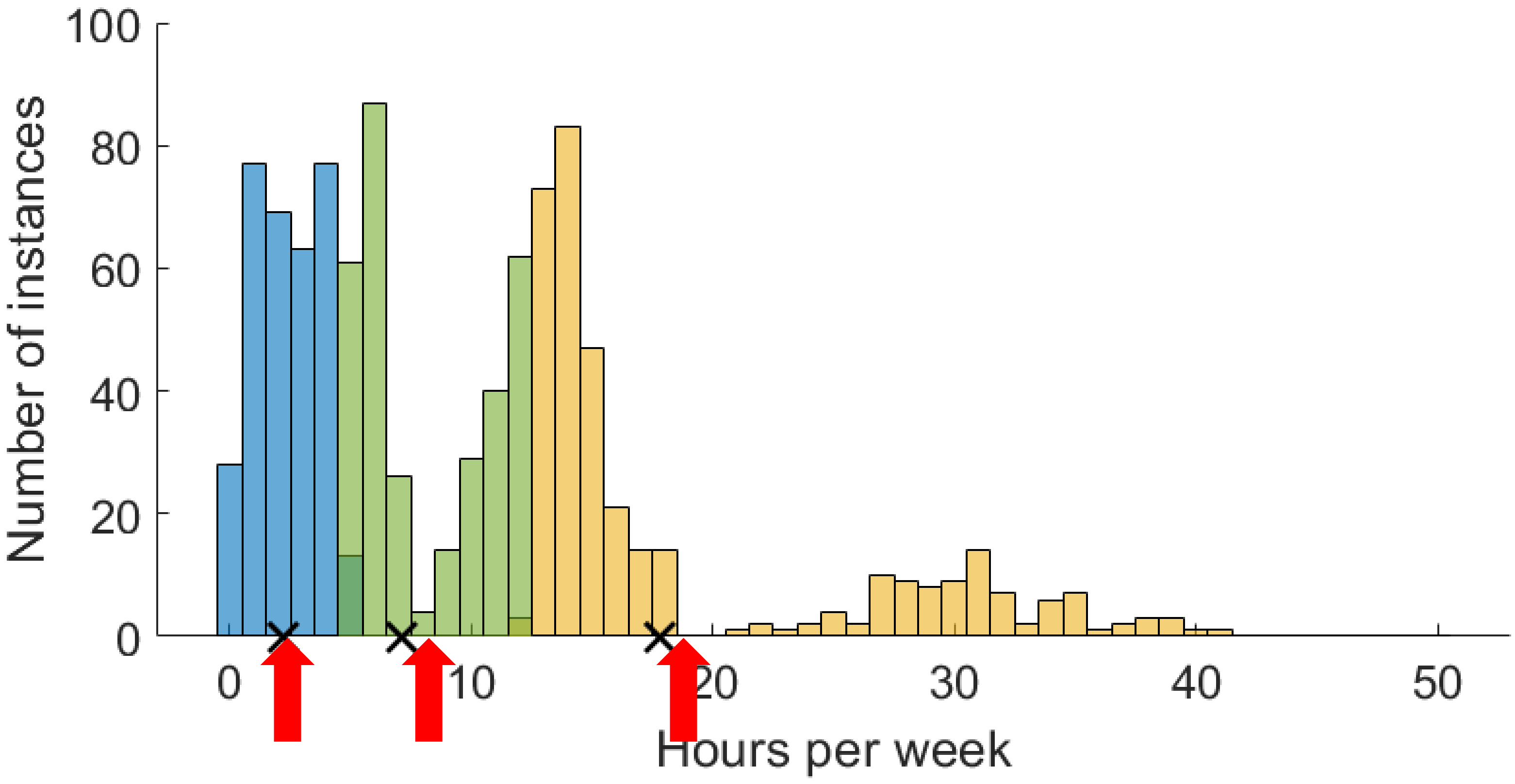
Discretisation: K-means clustering



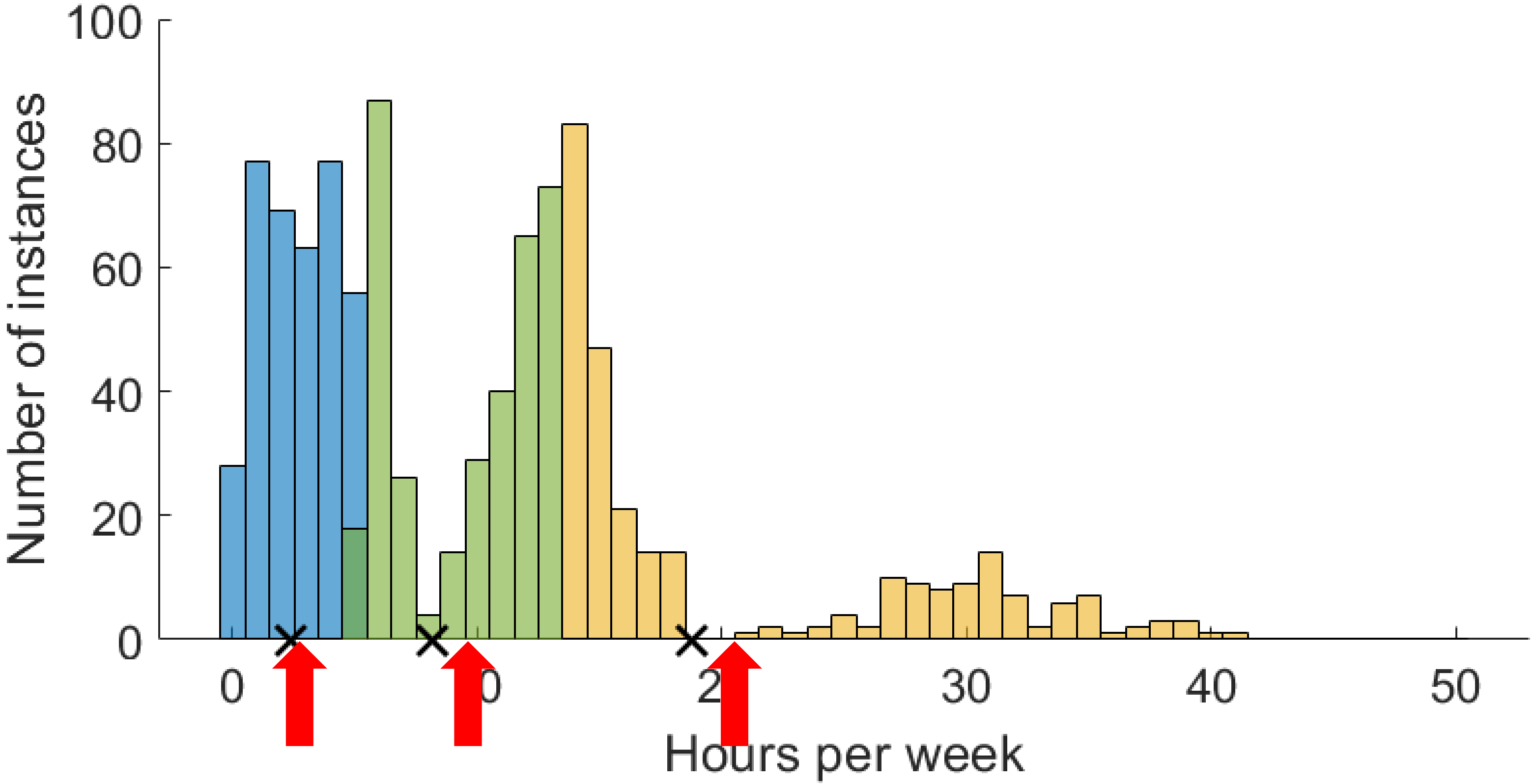
Discretisation: K-means clustering



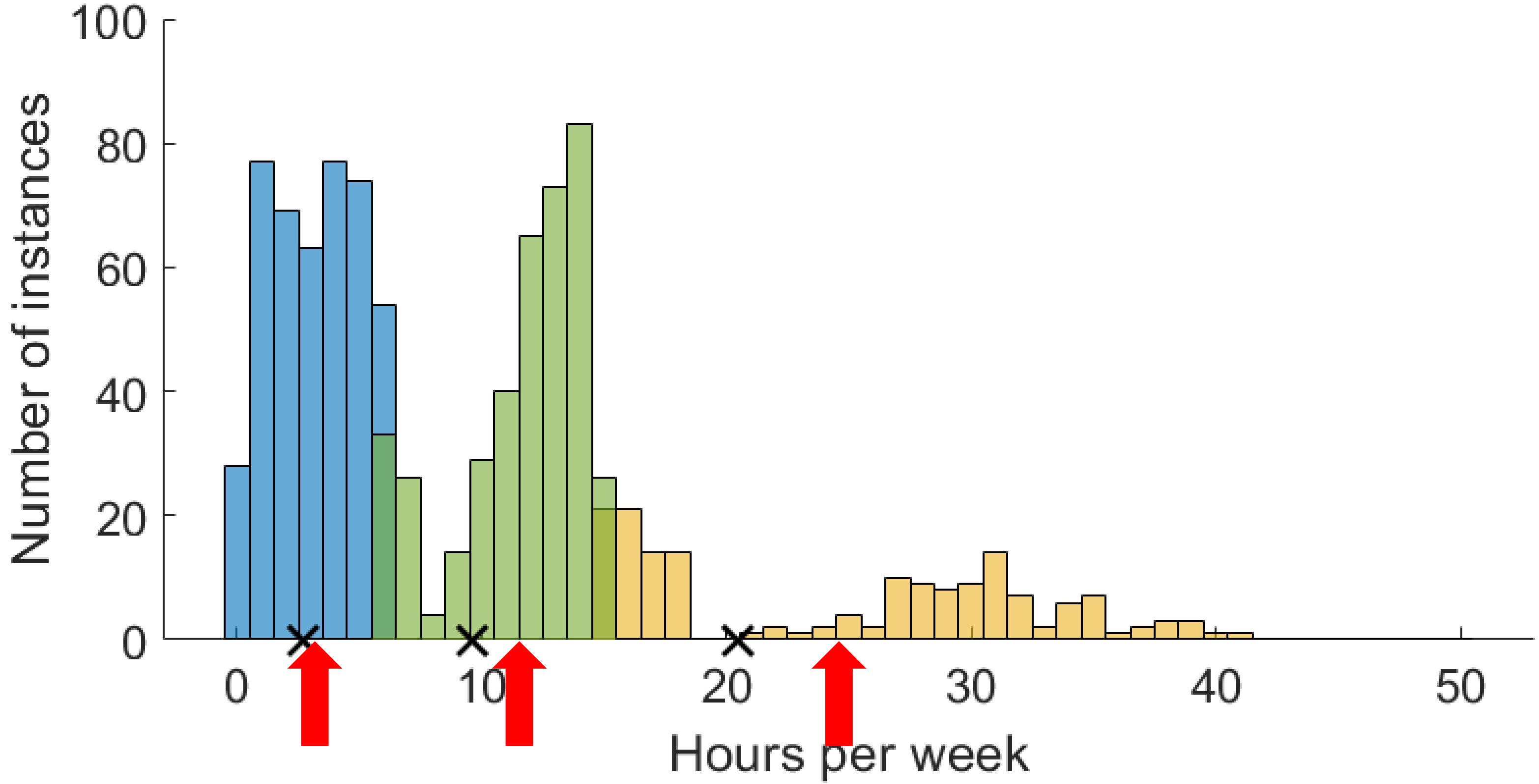
Discretisation: K-means clustering



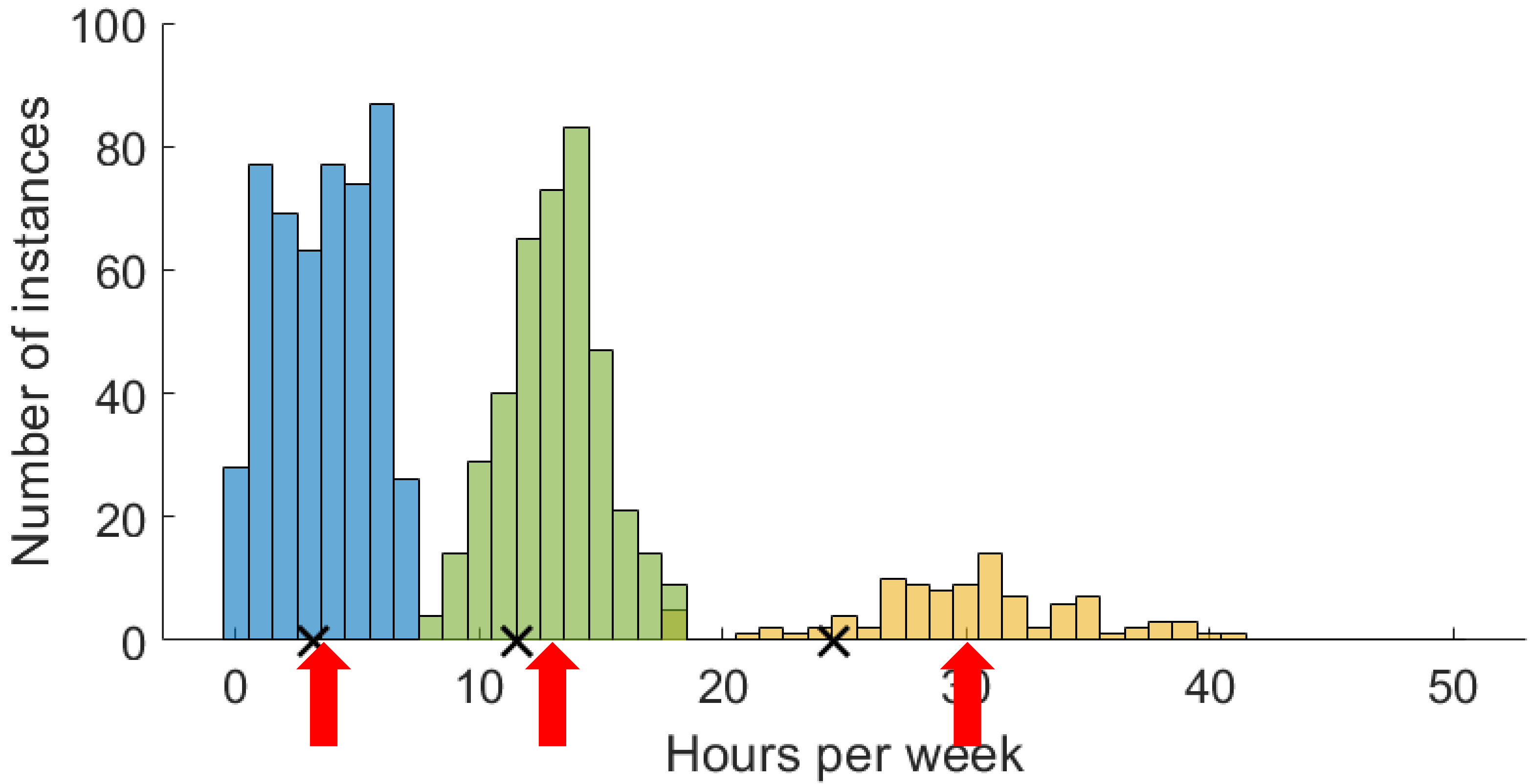
Discretisation: K-means clustering



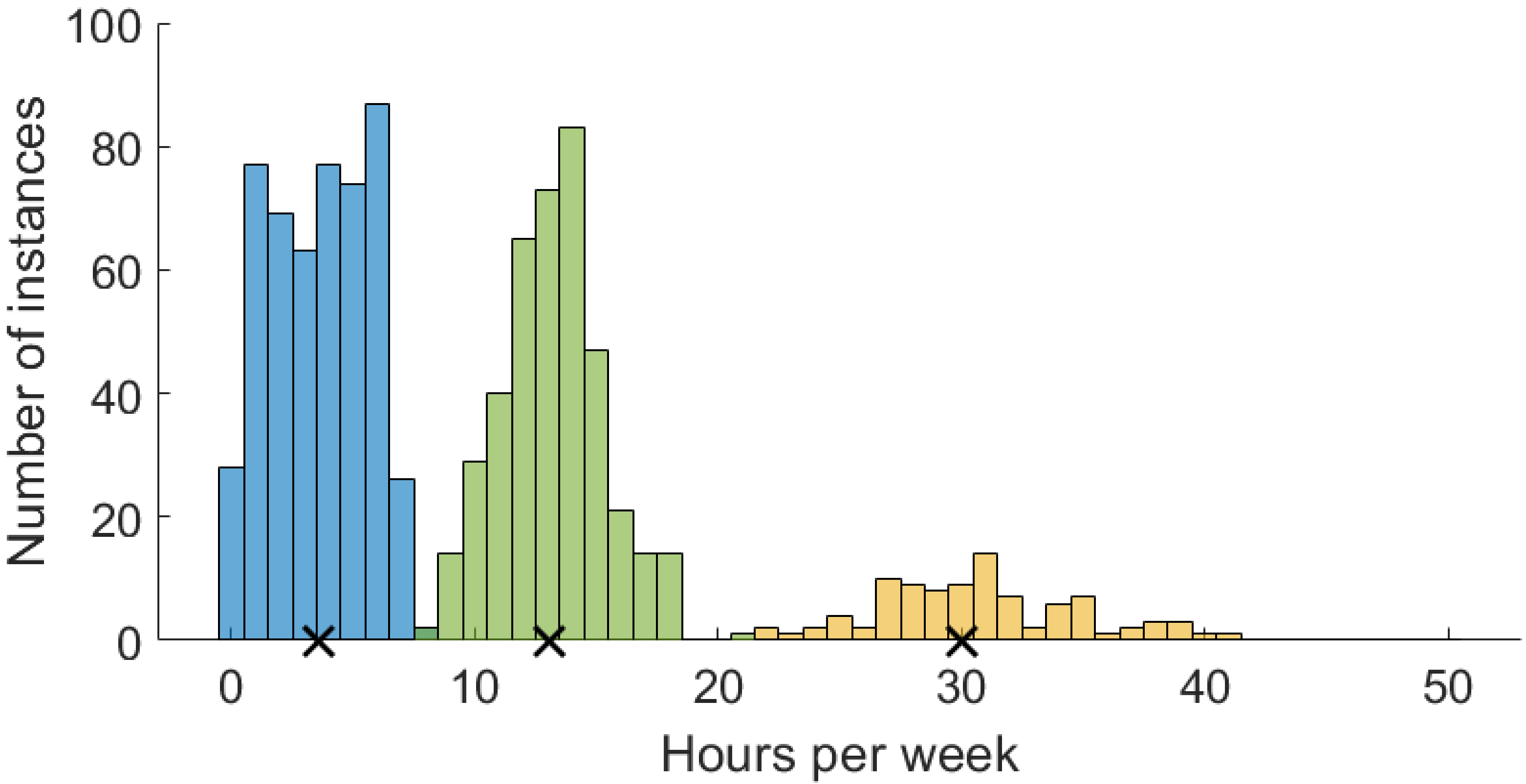
Discretisation: K-means clustering



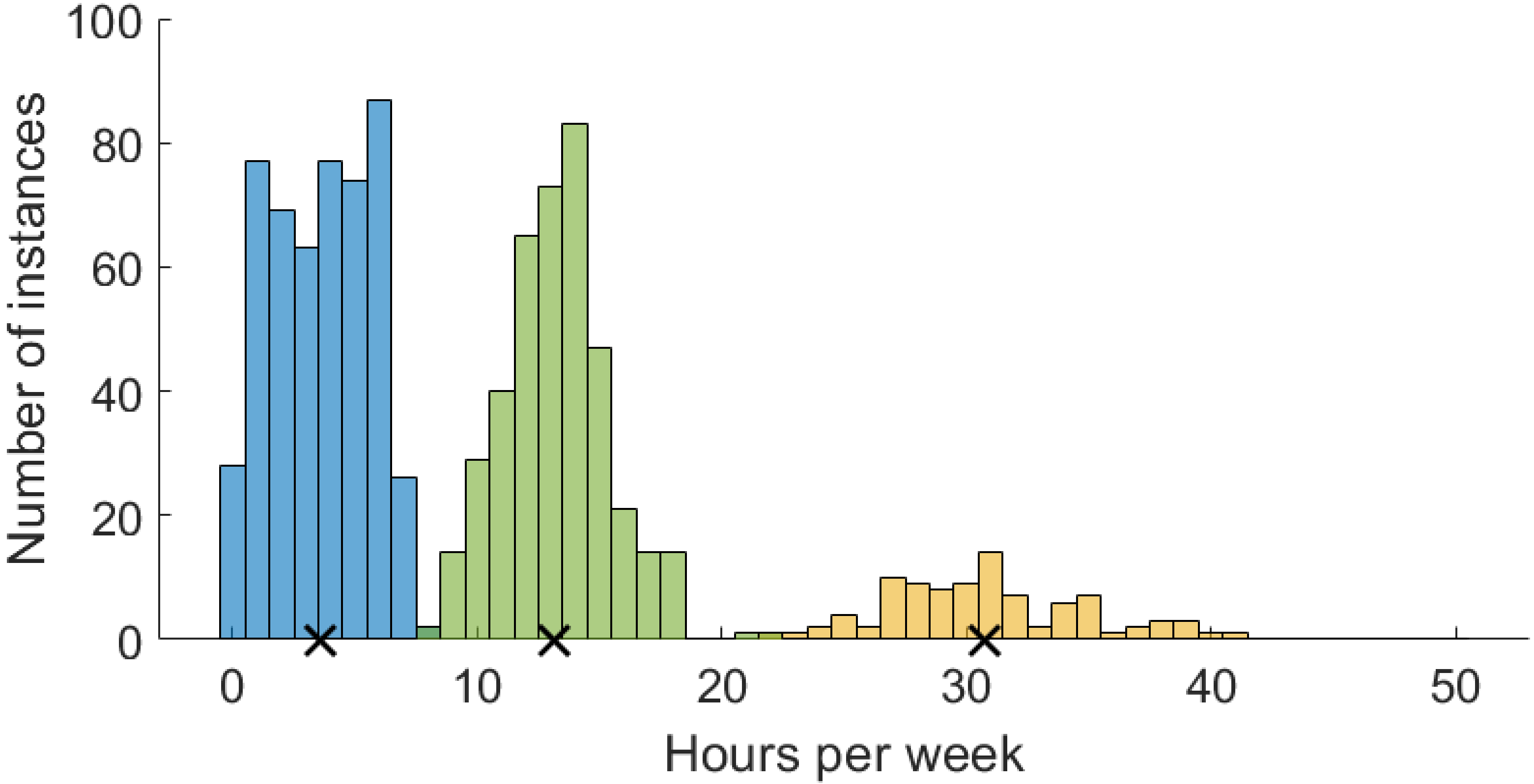
Discretisation: K-means clustering



Discretisation: K-means clustering

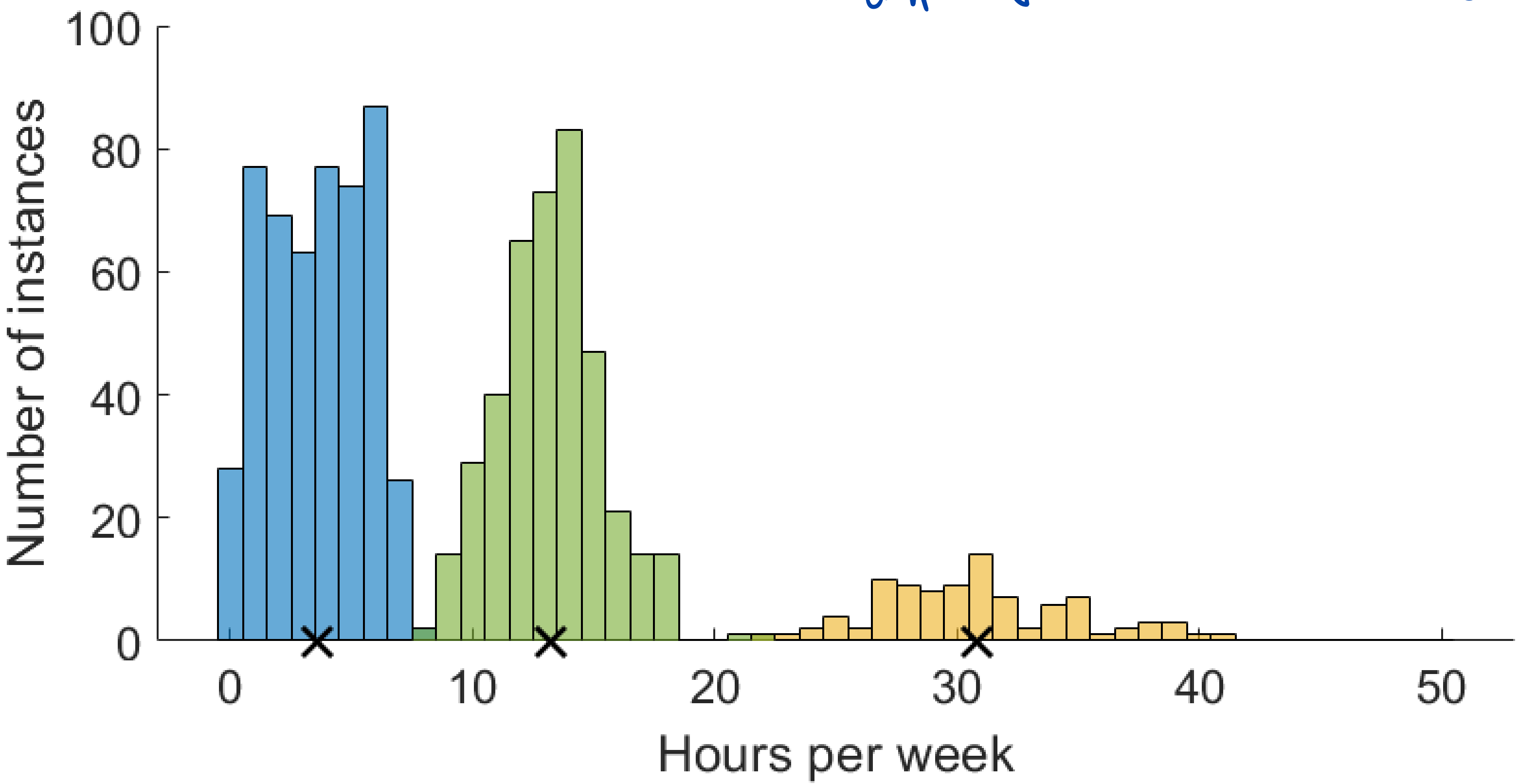


Discretisation: K-means clustering



Discretisation: K-means clustering

converge → centre unchange



Supervised discretisation

- Group values into class-contiguous intervals
  - Sort values, and identify breakpoints in class membership

rainy   not rainy   rainy → label

temp → 1 | 2 | 4 5 7 | 8 9 | 9 11

  - Reposition breakpoints if the numeric value is the same

1 | 2 | 4 5 7 | 8 9 9 | 11

  - Set the breakpoints midway between the neighbouring values

1 | 2 | 4 5 7 | 8 9 9 | 11

1.5 3 7.5 10

∴ 5 bins

Supervised discretisation

- Supervised discretisation may help you find "groups" that are most relevant for your classification task
  to the label
- Disadvantages
  - Arbitrary group boundaries
  - Arbitrary number of groups (tends to produce too many groups)
  - Tends to **overfit** training set

Discretisation summary

- Various options; each has advantages and disadvantages
- Discretisation means throwing out some information
  - This can help simplify data to make it easier for the model to learn
  - But you can potentially lose details that would have been useful

Naïve Bayes with continuous data

Naïve Bayes

- In naïve Bayes, we choose the class c_j that maximizes:

posterior

$$P(c_j)\prod_i P(x_i|c_j)$$

Prior probability of class c_j

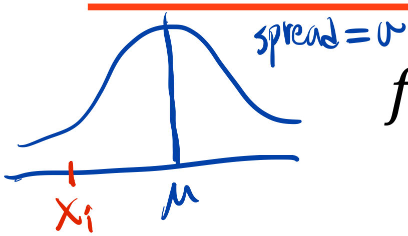
Likelihood of feature x_i in class c_j

- $P(x_i|c_j)$ can be computed differently for different data types:
  - If x_i is a nominal attribute with multiple levels, count number of times each level occurs in class c_j
  - If x_i is a numeric attribute, compute its **probability density function**

pdf

Probability density function

- A popular pdf for continuous data is the **Gaussian distribution** (or **normal distribution**)
- The probability of observing value x from a variable with mean (expected value) μ and standard deviation σ:



$$f(x) = \phi_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

ϕ_σ is a Gaussian distribution with mean = 0 and standard deviation = σ

- Important properties of this distribution:
  - Symmetric about the mean μ
  - Area under the curve sums to 1

Probability density function

- Where do the **mean** μ and **standard deviation** σ come from?
- These are descriptive statistics that can be estimated from our data

Mean of N samples of an attribute X:

$$\mu_x = \sum_{i=1}^{N} \frac{x_i}{N}$$

Standard deviation of N samples of an attribute X:

$$\sigma_x = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N-1}}$$

Why Gaussian?

- In practice, the normal distribution is a reasonable approximation for many events
  - Including binomial, Poisson, chi-square, and Student's t-distributions
  - E.g., height, weight, shoe size, reaction time …
- Easy to compute mean and standard deviation from data
- Even if the data isn't quite normally distributed, assuming a normal distribution often works well enough

Example: Gaussian naïve Bayes

- Compute probability distribution for each class:

Wind	Temp	Rain?
north	32.1	no
east	18.4	**yes**
east	19.5	no
north	23.5	no
north	20.3	**yes**
east	19.7	**yes**

When rain=no, temp is [32.1, 19.5, 23.5]   N=3

Mean:

$$\mu_{no} = (32.1 + 19.5 + 23.5)/3 = 25.0$$

Standard deviation:

$$\sigma_{no} = \sqrt{\frac{(32.1 - 25.0)^2 + (19.5 - 25.0)^2 + (23.5 - 25.0)^2}{3 - 1}} = 6.4$$

Example: Gaussian naïve Bayes

- Compute probability distribution for each class:

Wind	Temp	Rain?
north	32.1	no
east	18.4	**yes**
east	19.5	no
north	23.5	no
north	20.3	**yes**
east	19.7	**yes**

When rain=yes, temp is
[18.4, 20.3, 19.7]

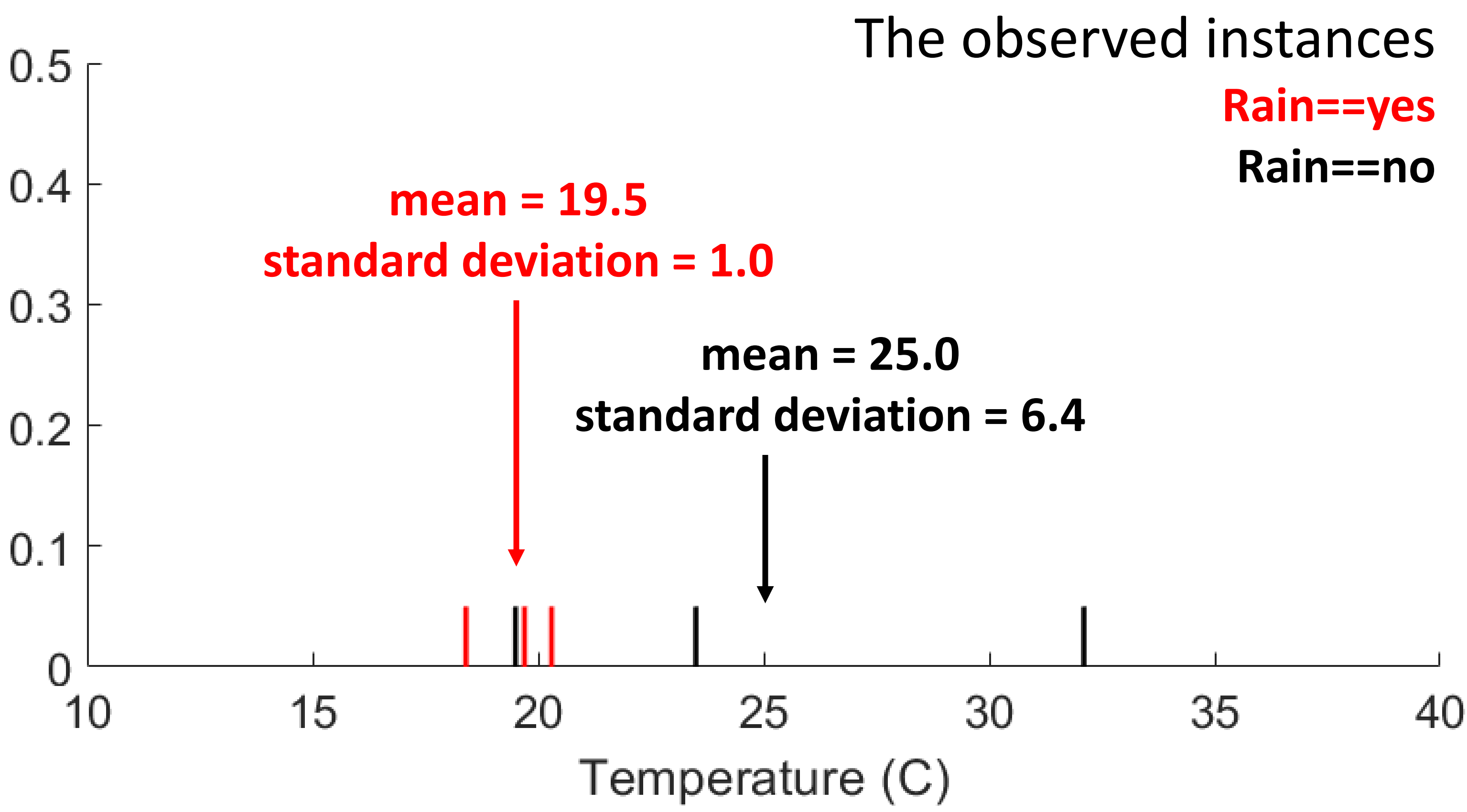
Mean:

$$\mu_{yes} = (18.4 + 20.3 + 19.7)/3 = 19.5$$

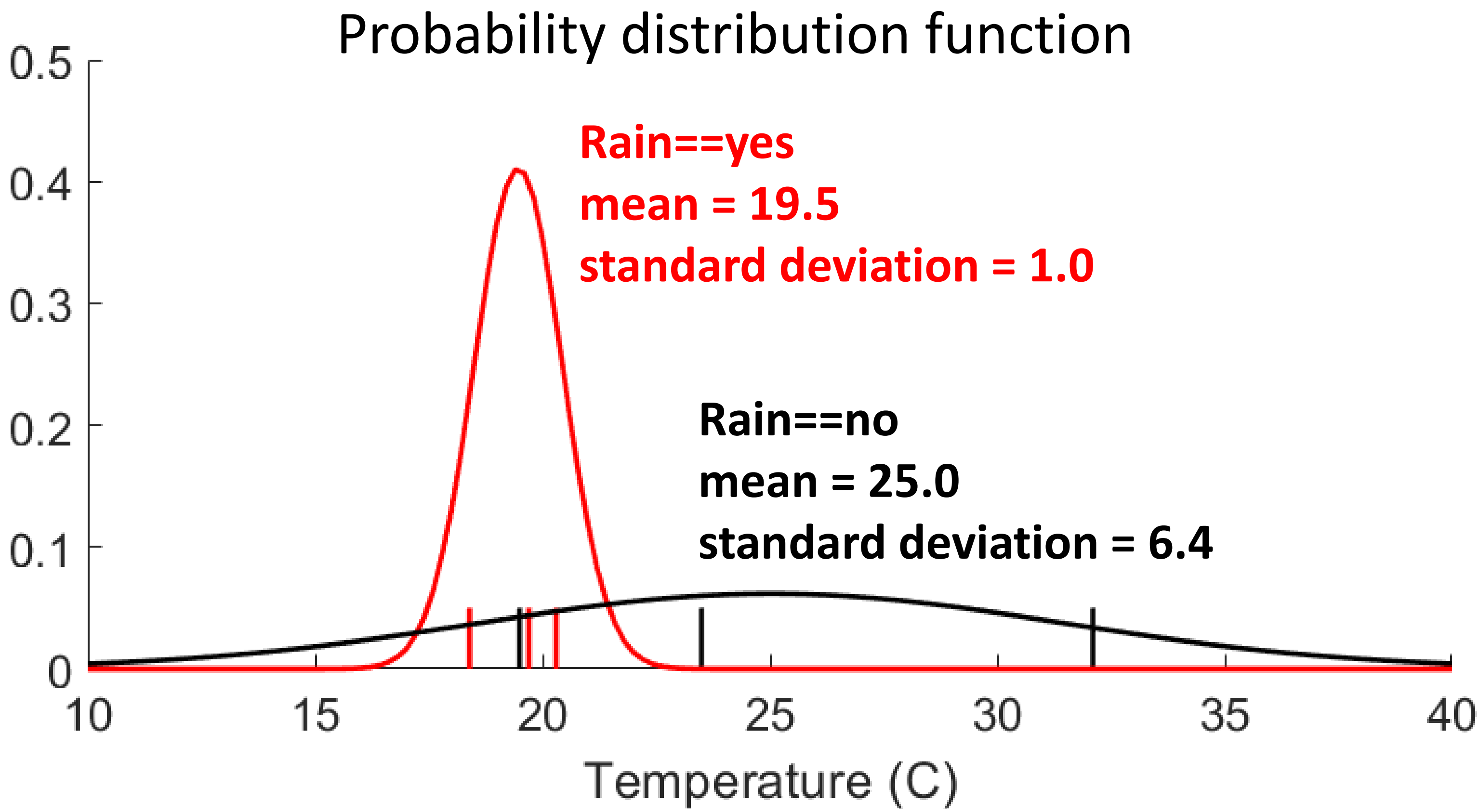
Standard deviation:

$$\sigma_{yes} = \sqrt{\frac{(18.4 - 19.5)^2 + (20.3 - 19.5)^2 + (19.7 - 19.5)^2}{3 - 1}} = 1.0$$

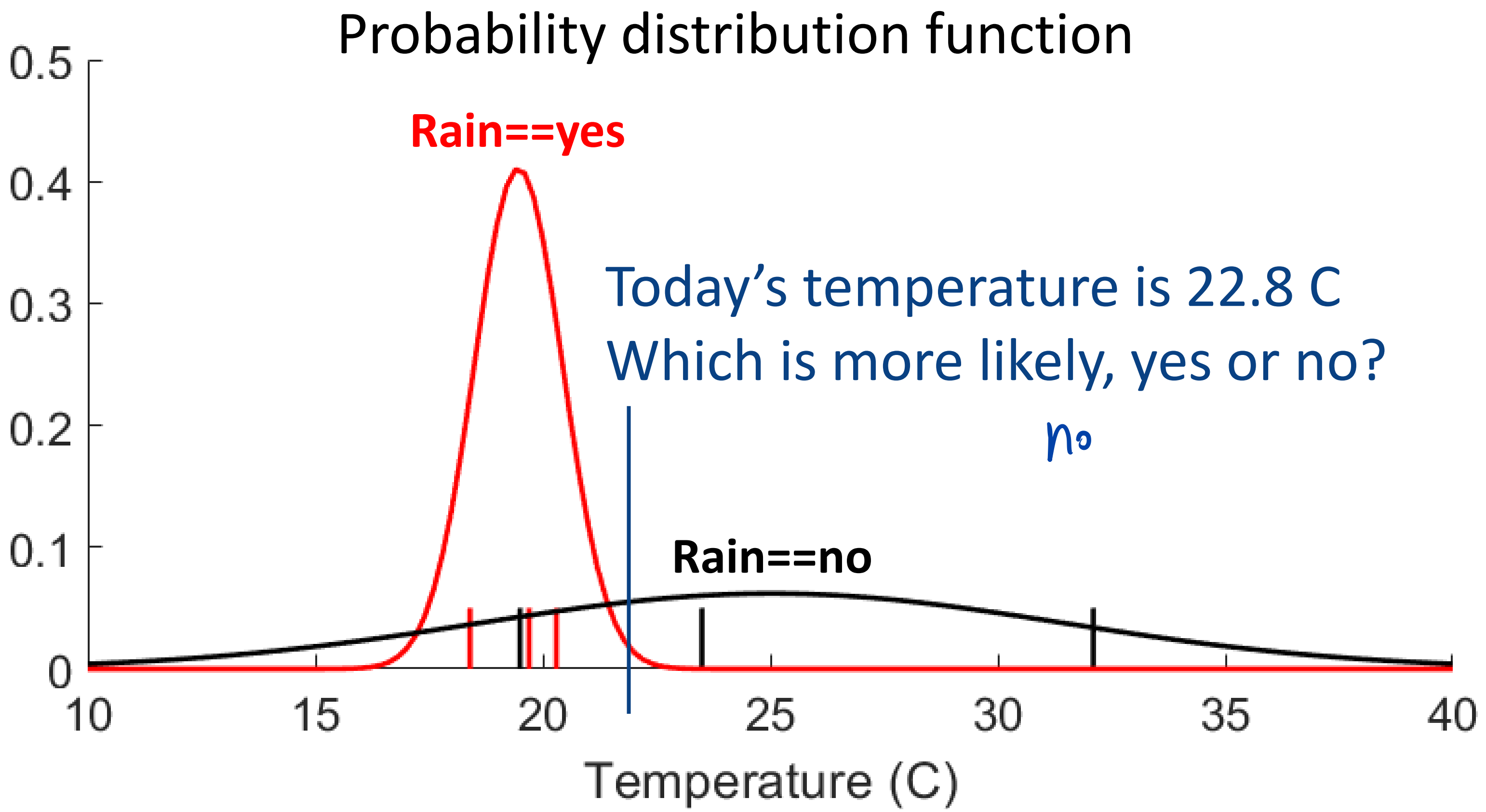
Example: Gaussian naïve Bayes



Example: Gaussian naïve Bayes



Example: Gaussian naïve Bayes



Example: Gaussian naïve Bayes

- Compute the probability of the test instance by substituting the μ and σ for each class:

$$P(rain = no|temp = 22.8) \overset{(\sim)}{=} P(rain = no)P(temp = 22.8|rain = no)$$

$$P(rain = yes|temp = 22.8) \overset{(\sim)}{=} P(rain = yes)P(temp = 22.8|rain = yes)$$

Example: Gaussian naïve Bayes

- Compute the probability of the test instance by substituting the μ and σ for each class:

$$P(rain = no|temp = 22.8) \approx P(rain = no)\phi_{\mu_{no},\sigma_{no}}(22.8)$$

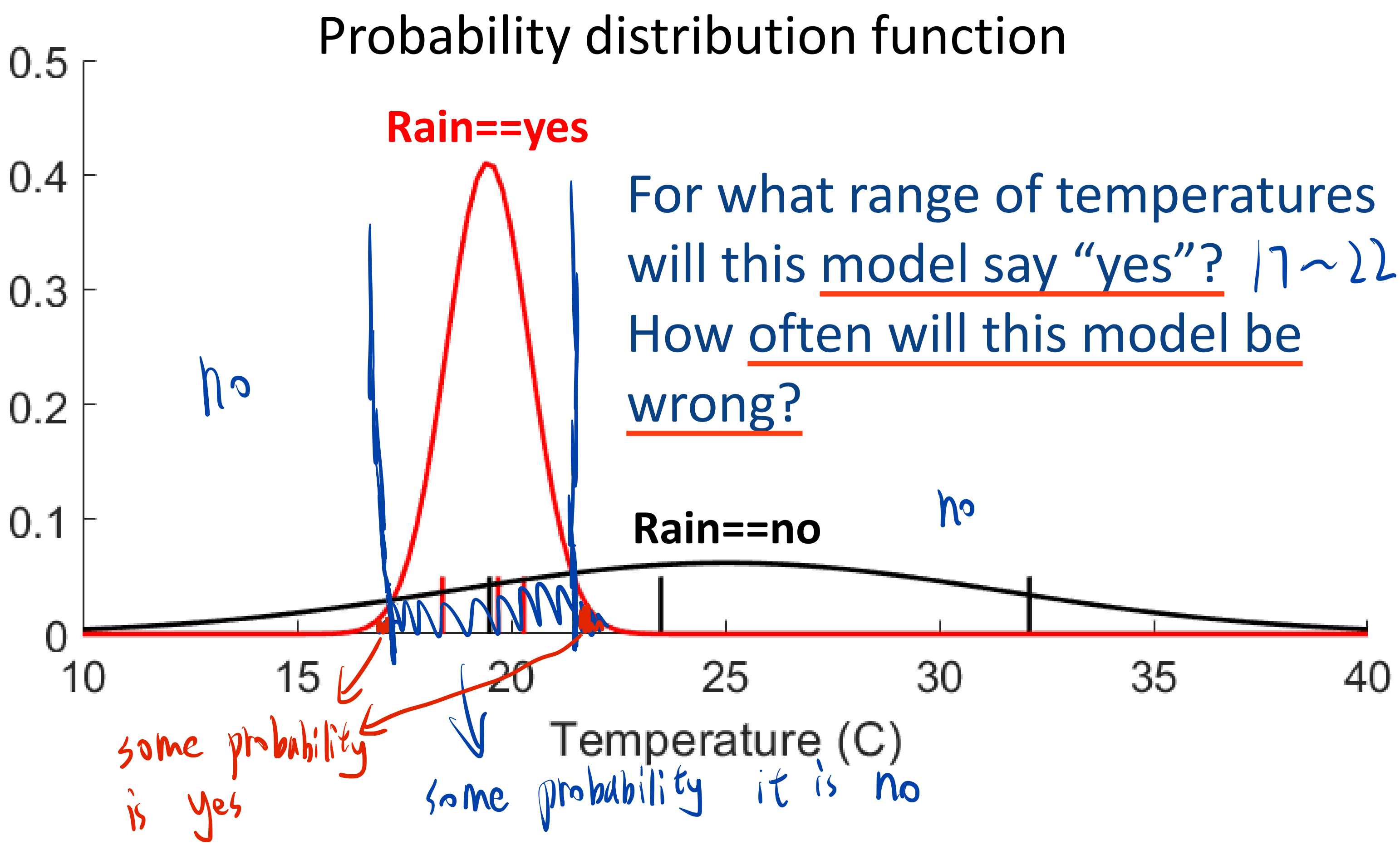
$$= (0.5)\frac{1}{6.4\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{22.8-25.0}{6.4}\right)^2} = \mathbf{0.0292}$$

$$P(rain = yes|temp = 22.8) \approx P(rain = yes)\phi_{\mu_{yes},\sigma_{yes}}(22.8)$$

$$= (0.5)\frac{1}{1.0\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{22.8-19.5}{1.0}\right)^2} = \mathbf{0.0006}$$

take log better

Explaining naïve Bayes



For what range of temperatures will this model say "yes"? 17~22

How often will this model be wrong?

Naïve Bayes for mixed data types

- In naïve Bayes, we choose the class c_j that maximizes:

$$P(c_j)\prod_i P(x_i|c_j)$$

Prior probability of class c_j

Likelihood of feature x_i in class c_j

- How you compute $P(x_i|c_j)$ depends on the data type of x_i and its probability distribution or density function

Types of naïve Bayes

- **Multivariate**: attributes are nominal and can take any of a fixed number of values
- Binomial (or Bernoulli): model assumptions: attributes are binary
  - Special case of multivariate
- Multinomial: model assumptions: attributes are natural numbers corresponding to frequency
  - Probability $P(a_k = m|c_j) \approx P(a_k = 1|c_j)^m/(m!)$
- **Gaussian**: model assumptions: attributes are numeric, and we can assume they come from a Gaussian distribution
- Kernel density estimation: attributes are numeric, and come from an arbitrary distribution learn the distribution from data

Naïve Bayes summary

- Naïve Bayes works with various data types
  - Just need to change how you compute $P(x_i|c_j)$ for each attribute
- Mix numeric and nominal attributes in one model – no need to convert data types
- Optimal combination of attributes
  - Assuming conditional independence, and correct estimates of the probability distributions