

Discrete & Continuous Data

Semester 1, 2021

Kris Ehinger

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

$$P(\text{temp} = 32.1 | \text{Rain} = \text{no}) = 1/3$$

$$P(\text{temp} = 19.5 | \text{Rain} = \text{no}) = 1/3$$

$$P(\text{temp} = 23.5 | \text{Rain} = \text{no}) = 1/3$$

$$P(\text{temp} = 18.4 | \text{Rain} = \text{yes}) = 1/3$$

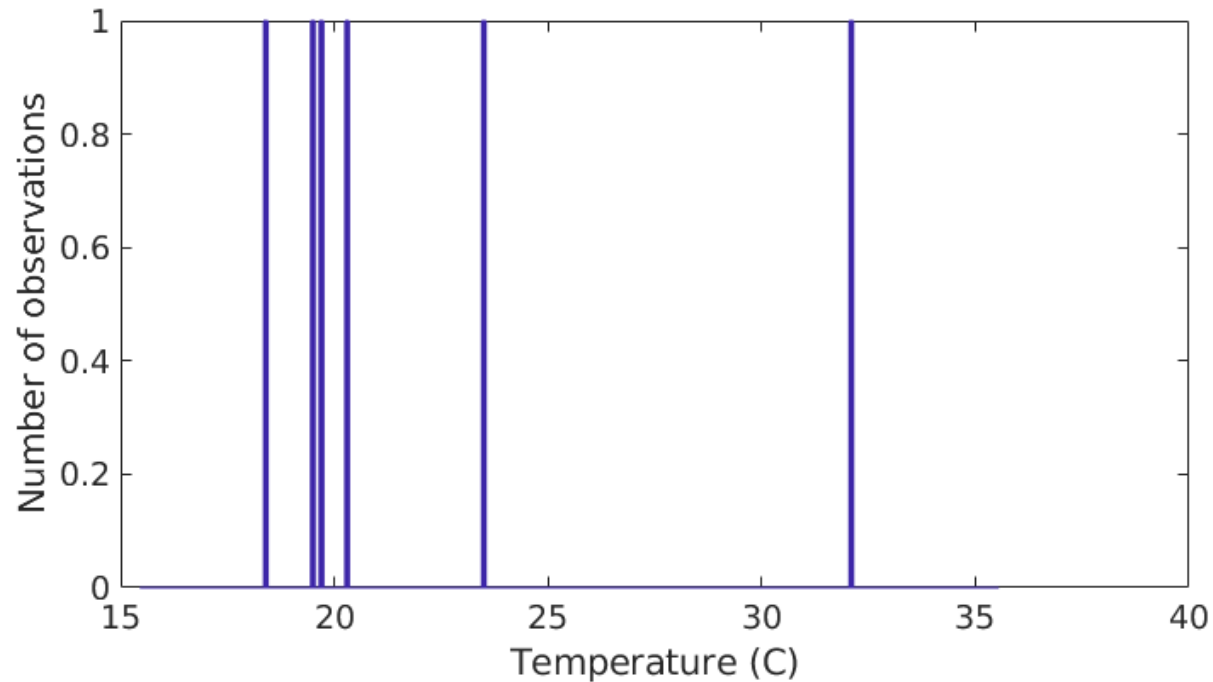
$$P(\text{temp} = 20.3 | \text{Rain} = \text{yes}) = 1/3$$

$$P(\text{temp} = 19.7 | \text{Rain} = \text{yes}) = 1/3$$

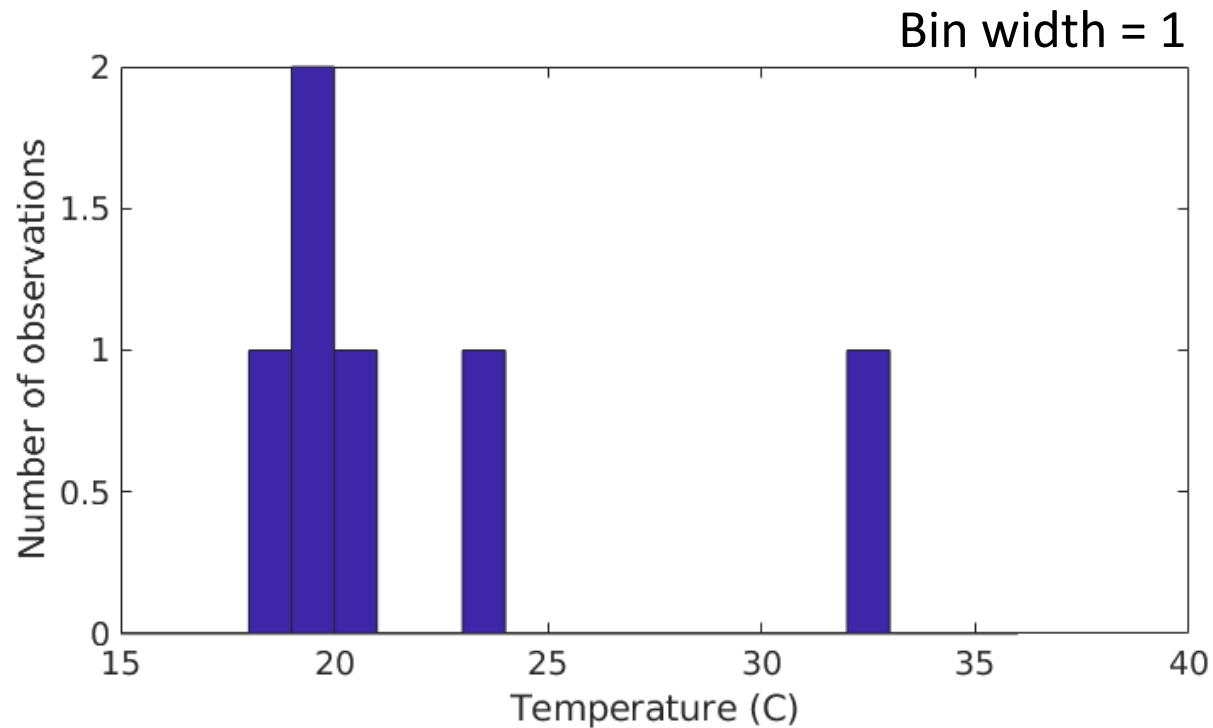
This doesn't look right...

Continuous attributes

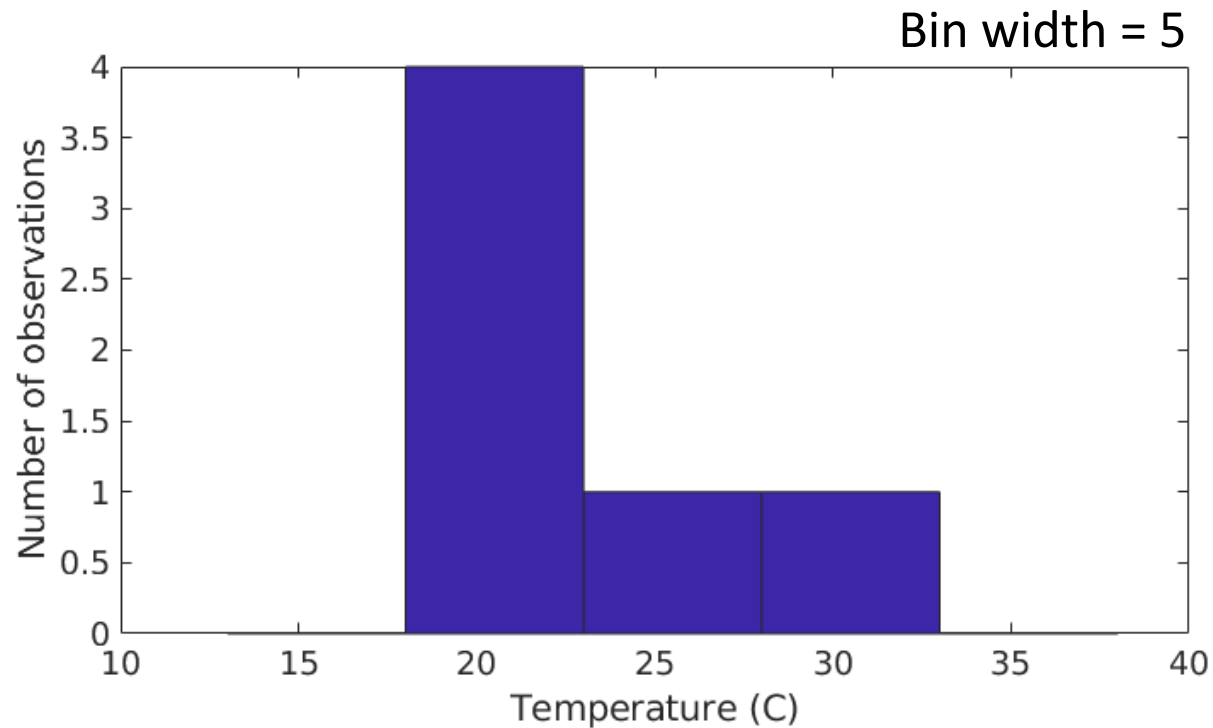
How the model perceives the training data:



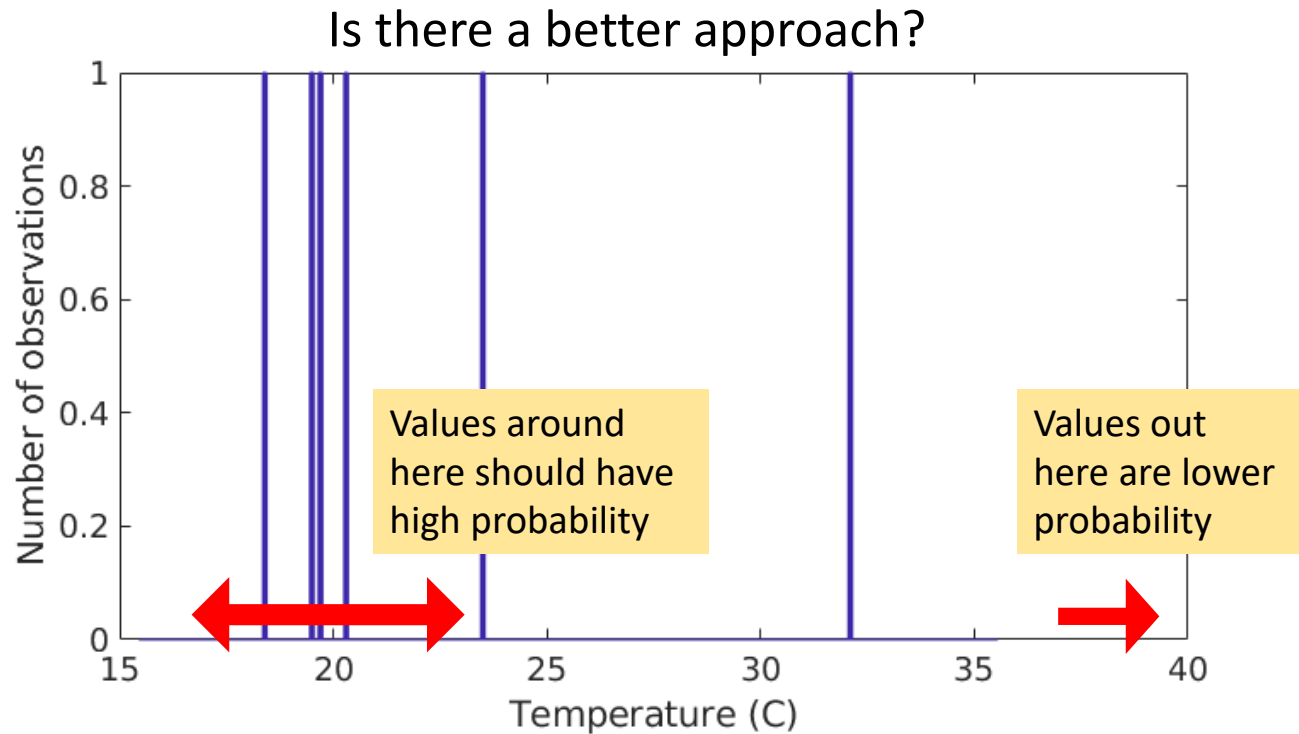
Continuous attributes



Continuous attributes



Continuous attributes



Outline

- Converting data types
 - Nominal -> numeric
 - Numeric -> nominal
- Naïve Bayes with continuous data
 - Gaussian naïve Bayes
 - Kernel Density Estimation (KDE)
 - KDE naïve Bayes

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

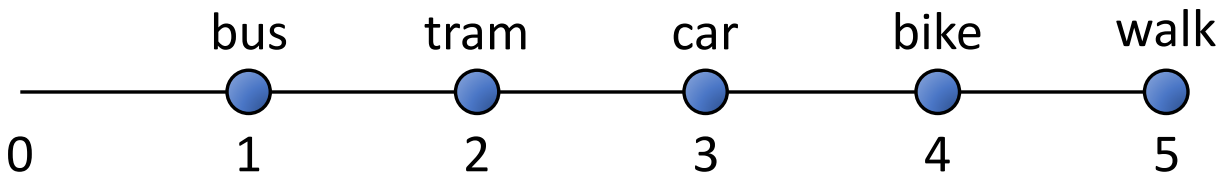
- Machine learning algorithms typically assume attributes have a particular data type
- Algorithms that assume nominal attributes:
 - Naïve Bayes (as described in last lecture)
 - Decision trees
- Algorithms that assume numeric attributes:
 - Support vector machines (SVM)
 - Perceptron, neural network

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

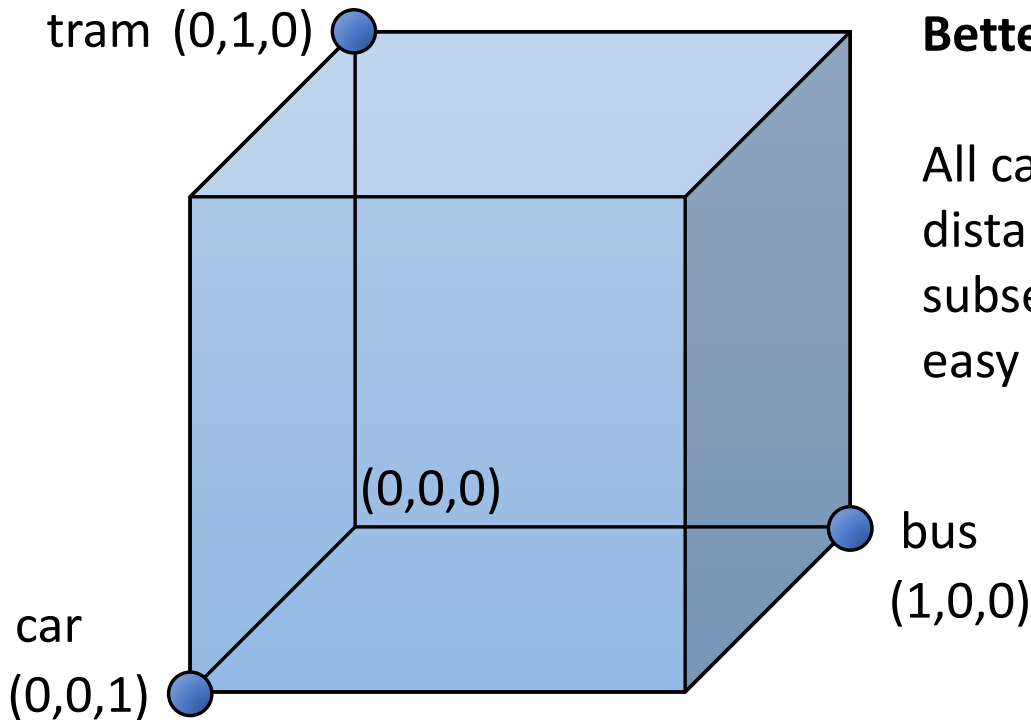
- Option 1: Convert category names to numbers
 - Attribute: mode of transport
 - Nominal: “bus,” “tram,” “car,” “bike,” “walk”
 - Numeric: 0, 1, 2, 3, 4
- Problem: creates an artificial ordering when no order exists, makes some categories seem more/less similar to each other



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

Visualisation of option 2



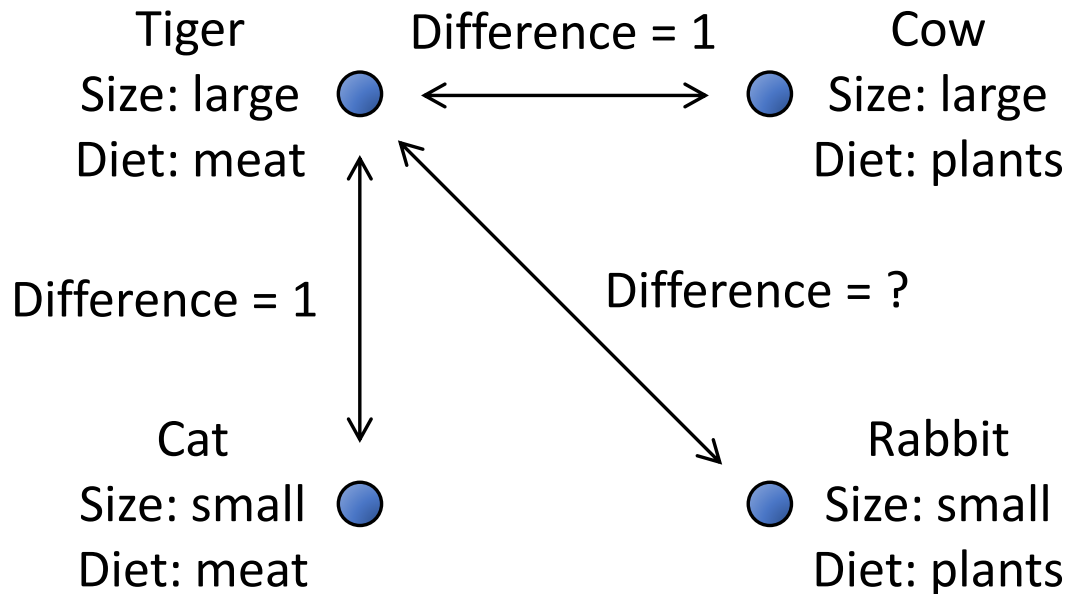
Better encoding:

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, $dist_H(A, B) = N$

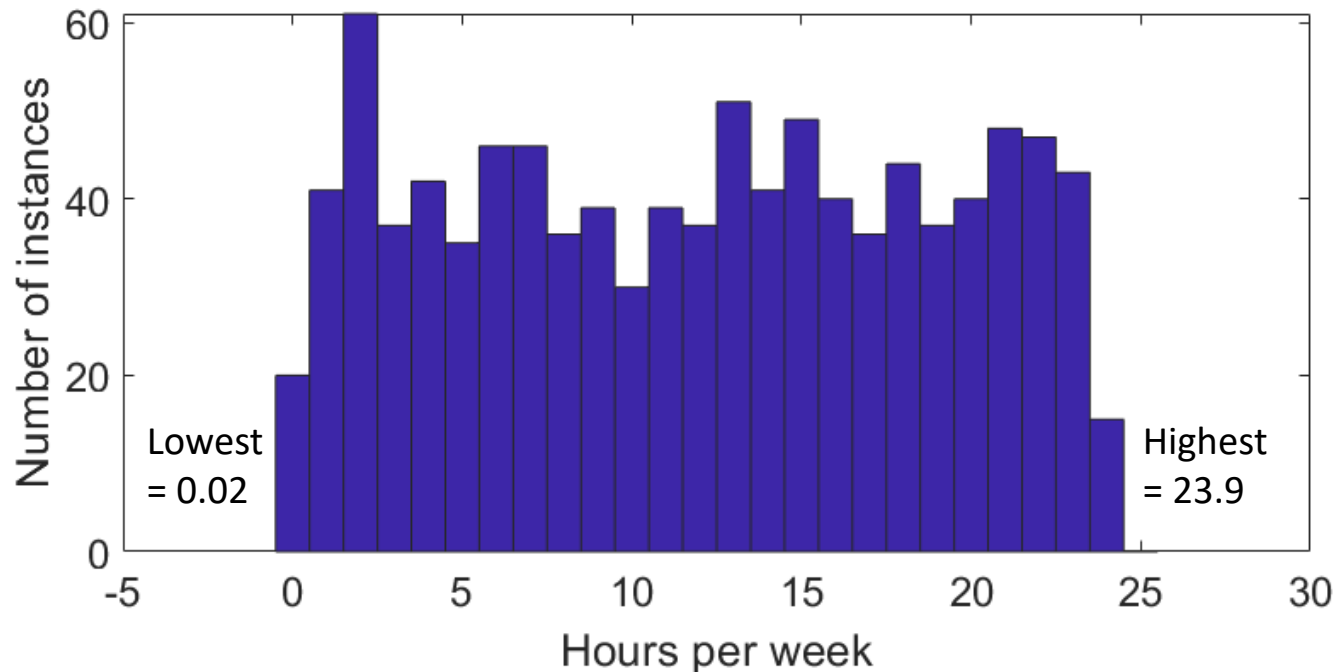
$$dist_H(A, B) = \sum_i \begin{cases} 0, & a_i == b_i \\ 1, & otherwise \end{cases}$$

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
 - Decide where to place the boundaries for these intervals

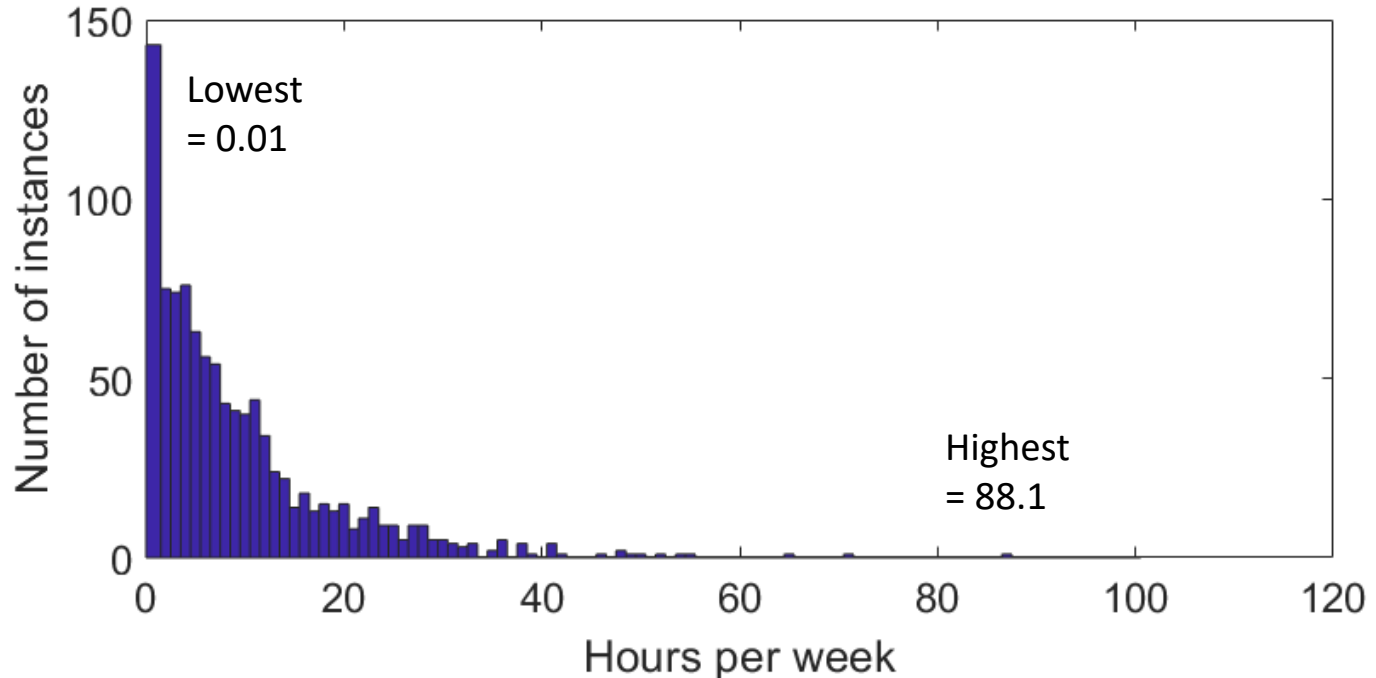
Discretisation example 1

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



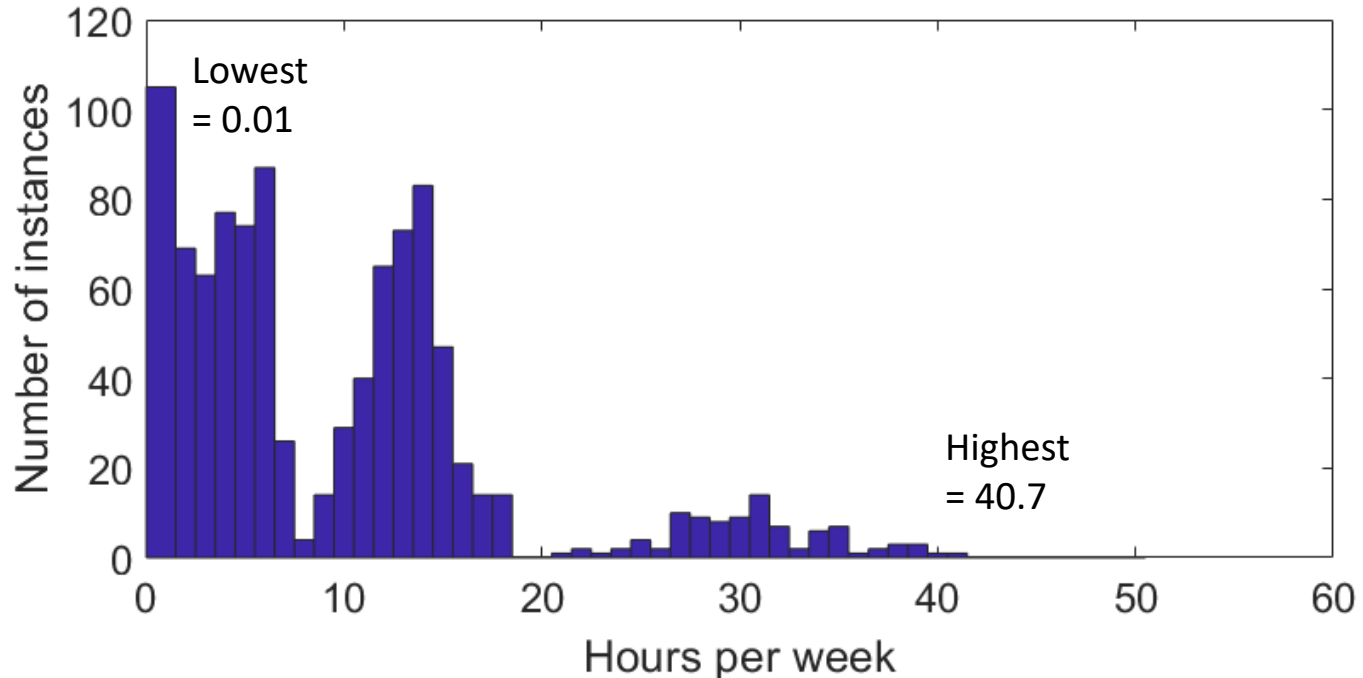
Discretisation example 2

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



Discretisation example 3

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



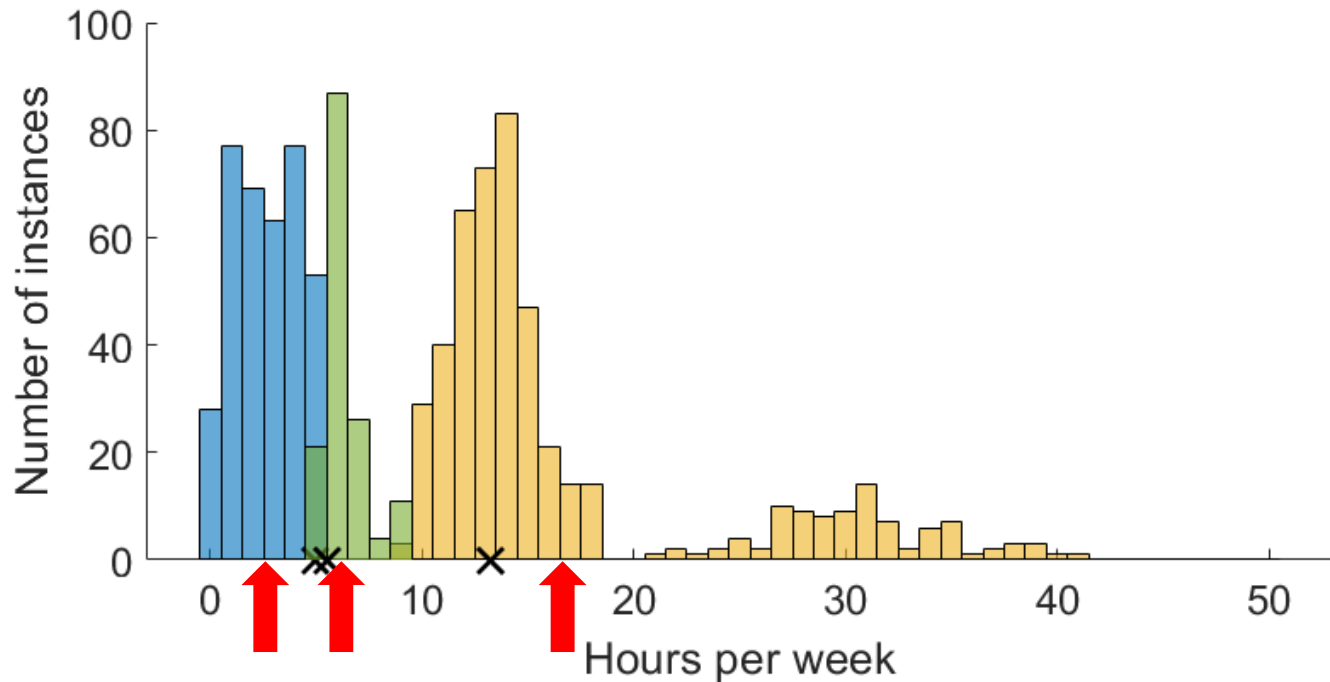
Discretisation options

- **Equal-width** discretisation – find min/max of range, partition into n bins of width $(\max - \min)/n$
- **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** *⇒ some numbers of a population are systematically more likely to be selected in a sample than others.*
 - User must choose n

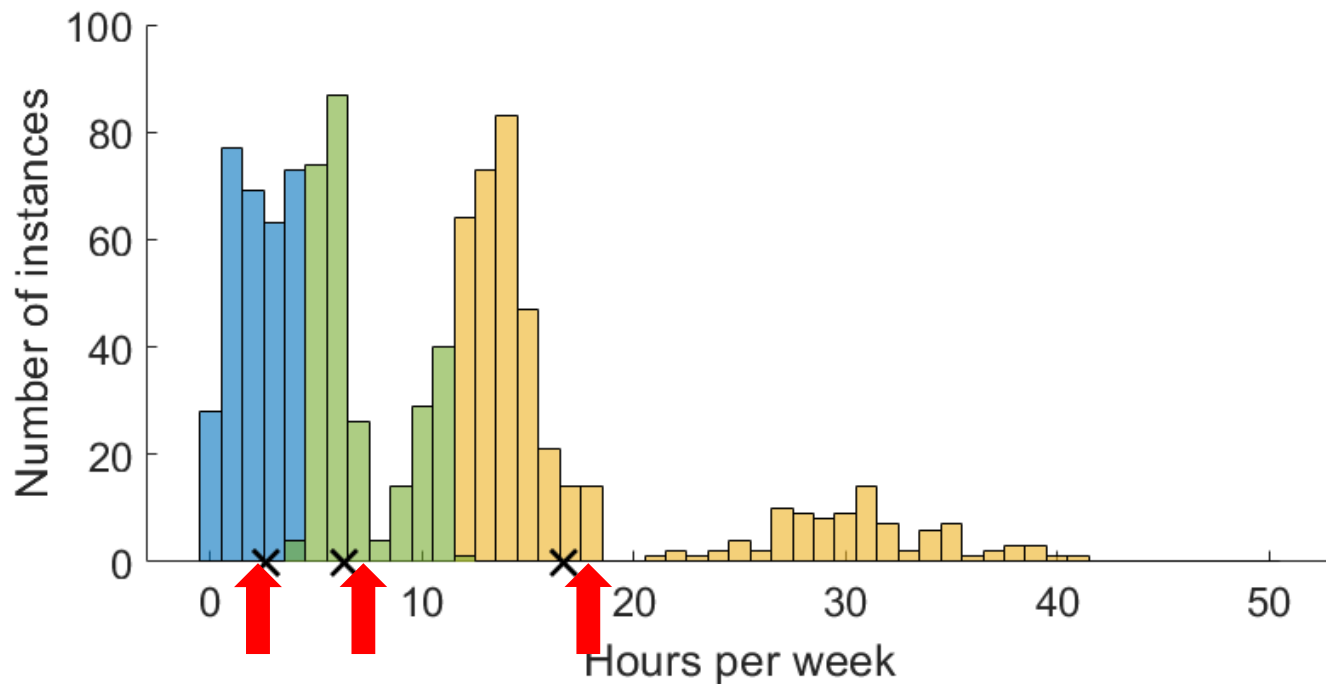
Discretisation options

- 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

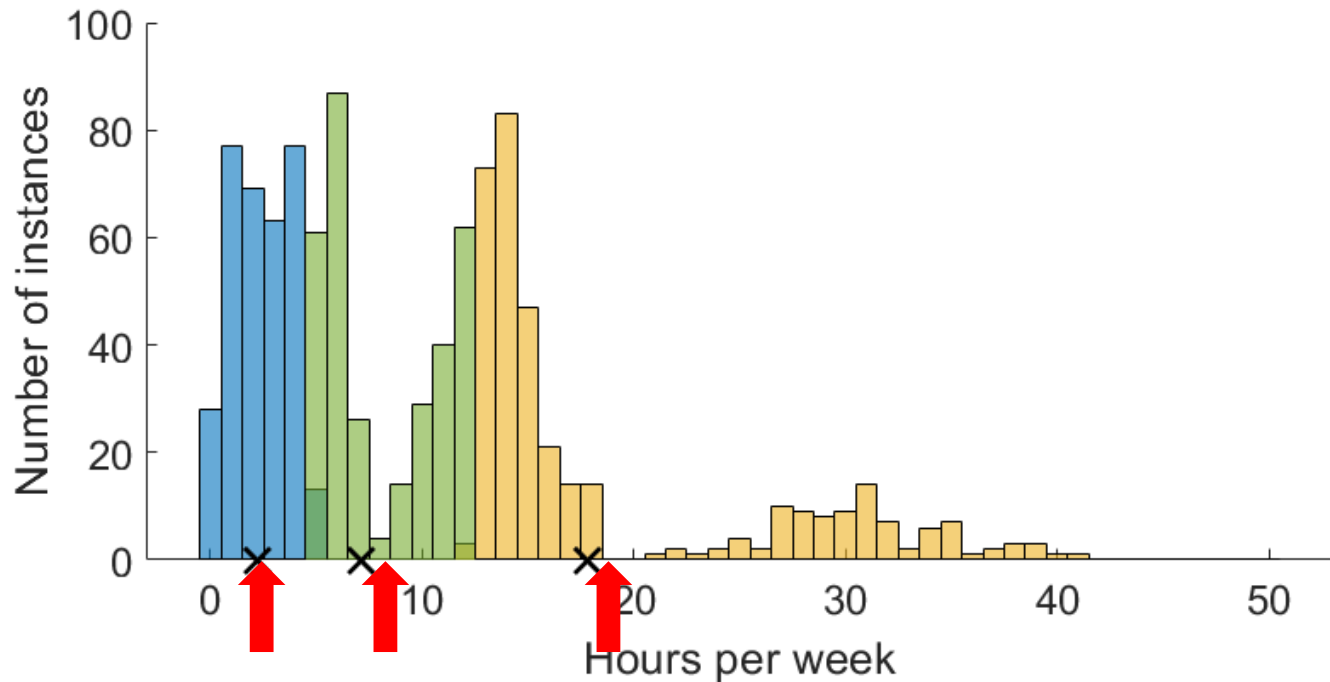
Discretisation: K-means clustering



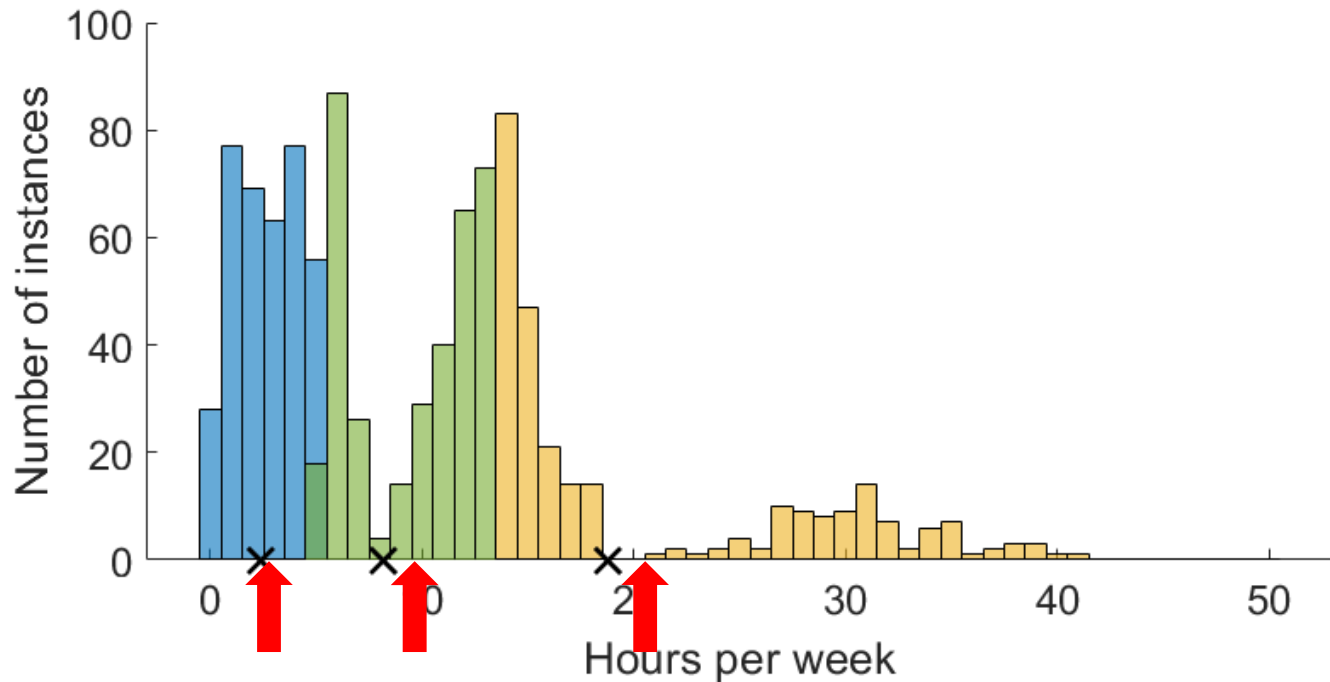
Discretisation: K-means clustering



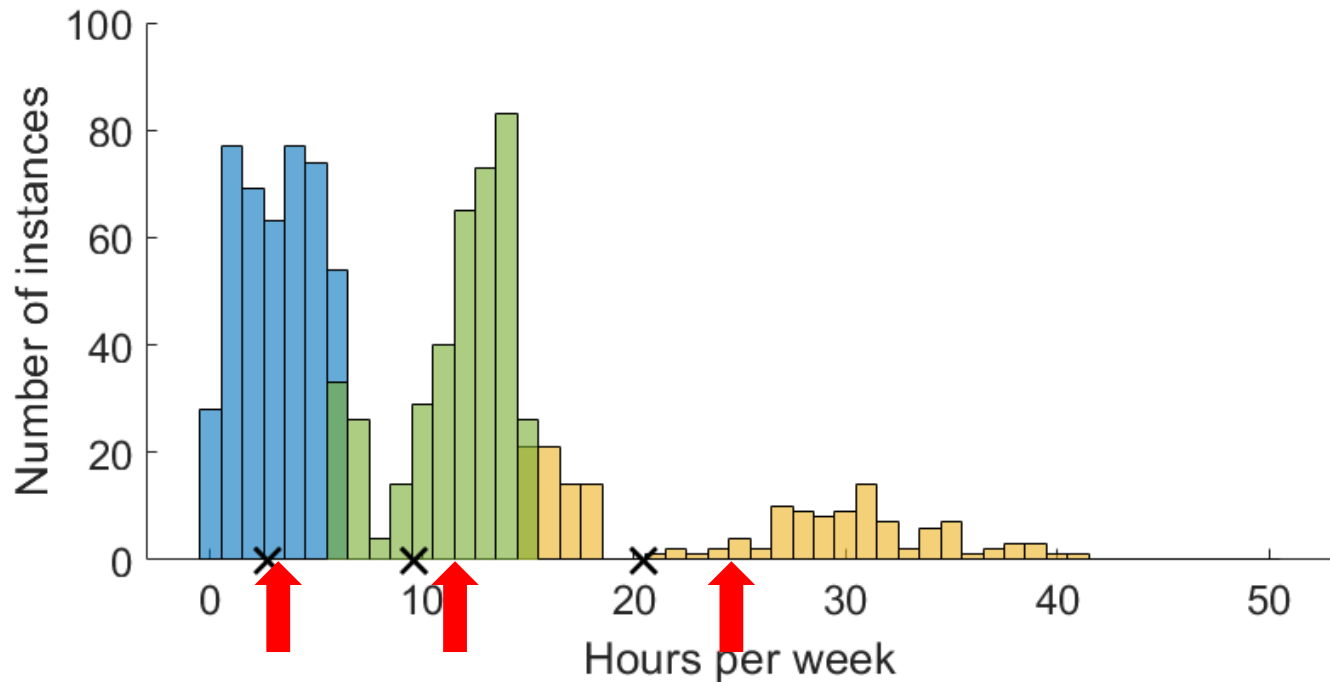
Discretisation: K-means clustering



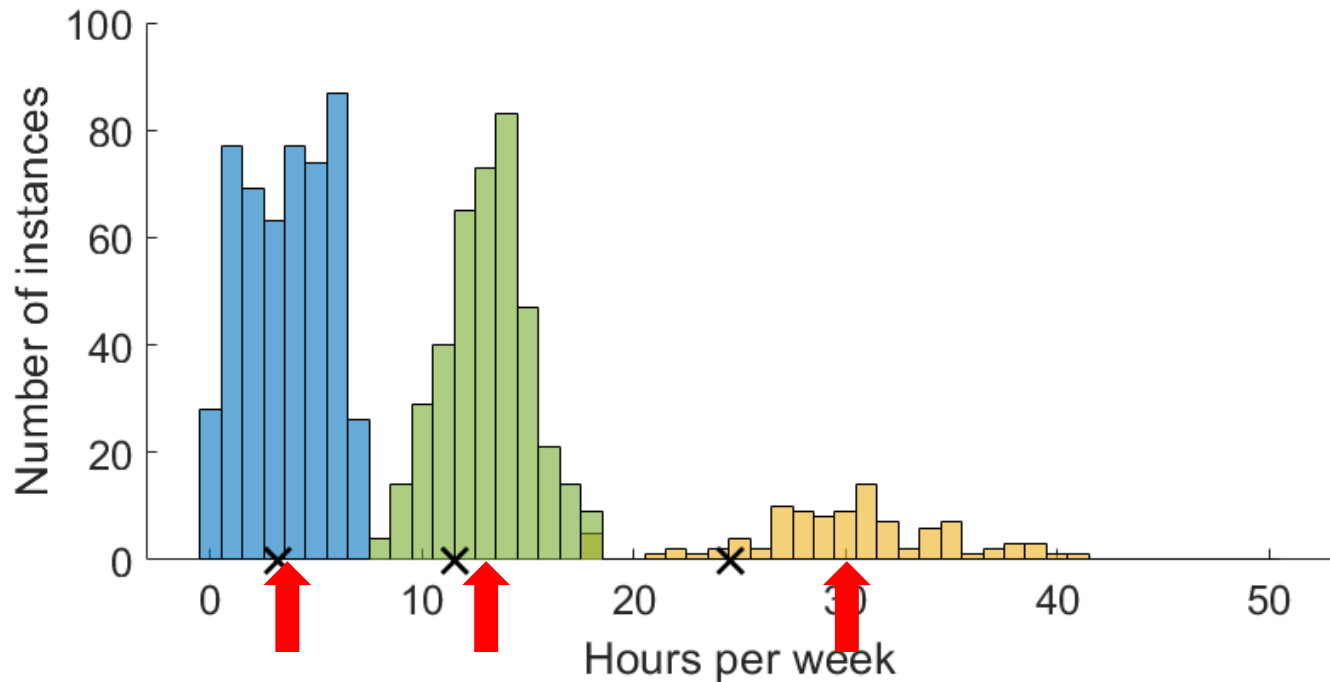
Discretisation: K-means clustering



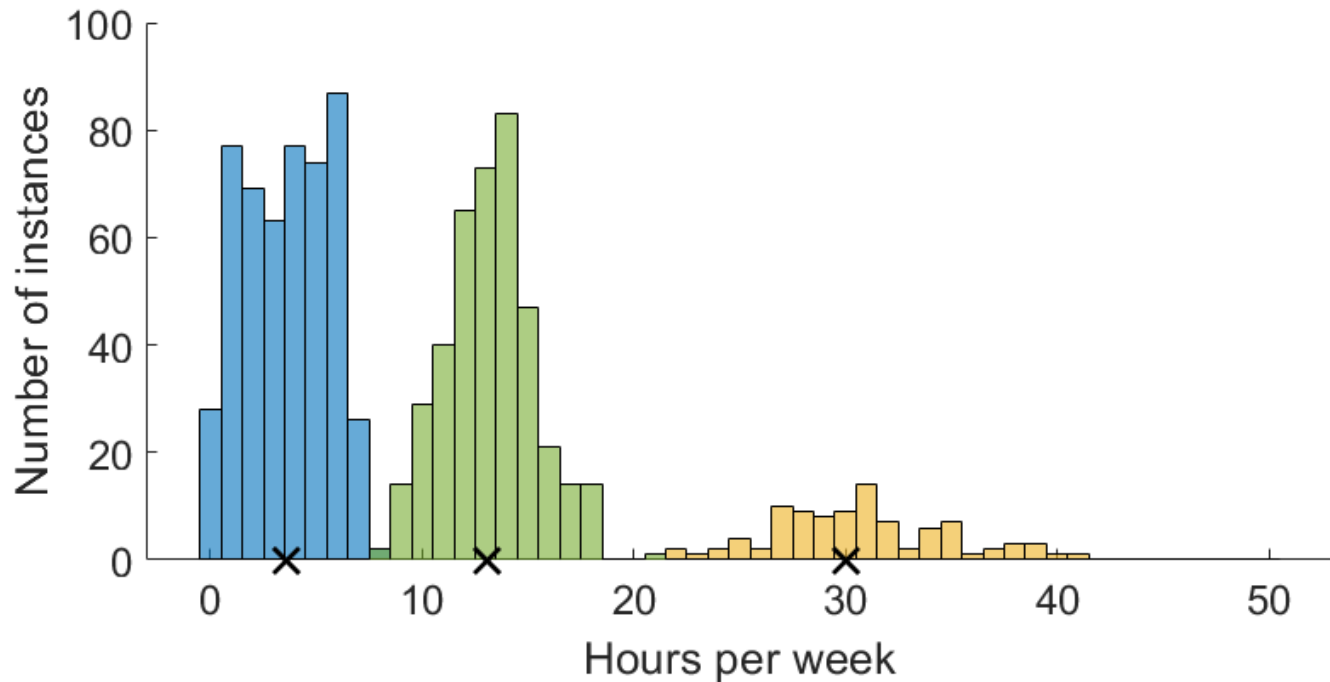
Discretisation: K-means clustering



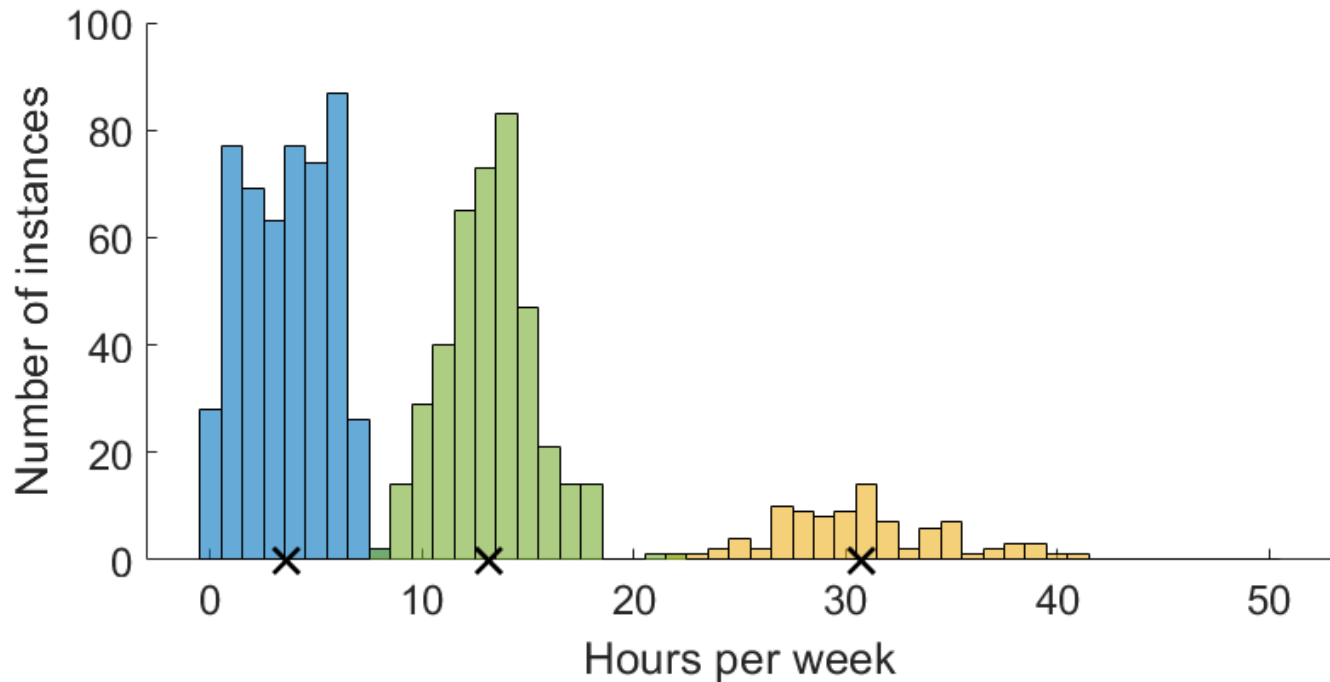
Discretisation: K-means clustering



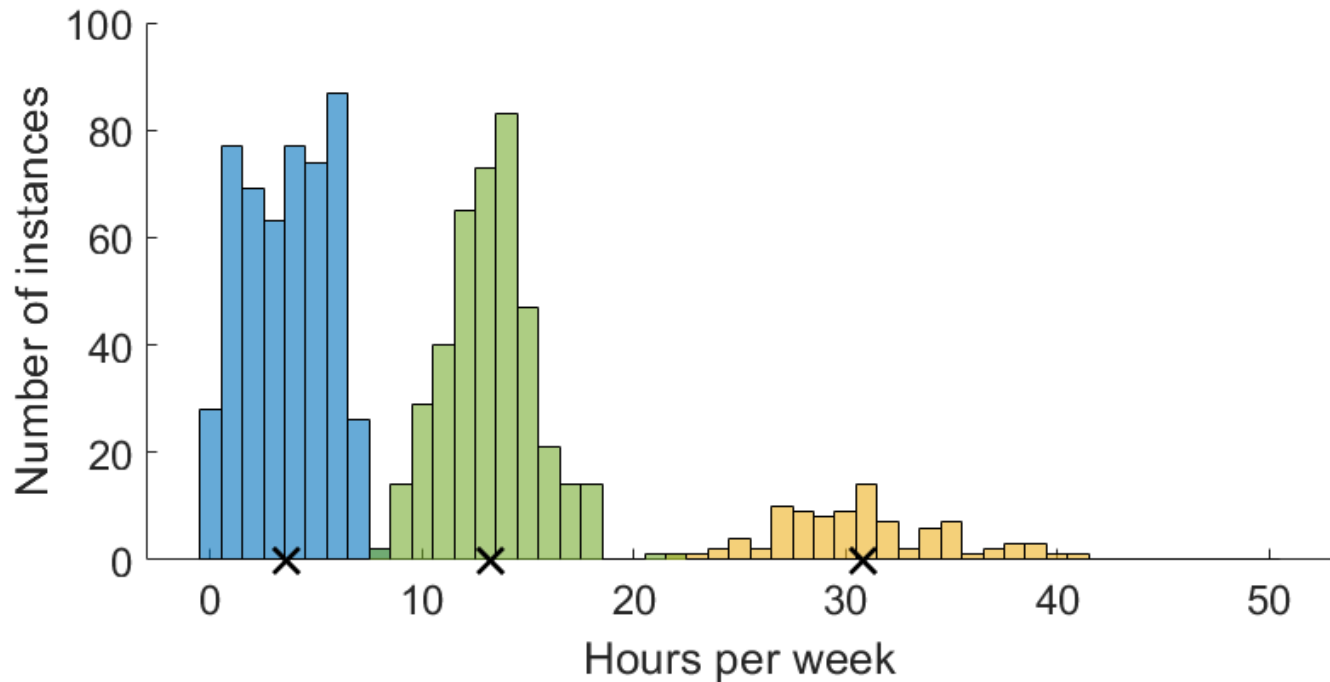
Discretisation: K-means clustering



Discretisation: K-means clustering



Discretisation: K-means clustering



Supervised discretisation

- Group values into class-contiguous intervals

- Sort values, and identify breakpoints in class membership

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

Supervised discretisation

- Supervised discretisation may help you find “groups” that are most relevant for your classification task
- 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:

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

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 distribution**

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 :

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

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 ...

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]

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]

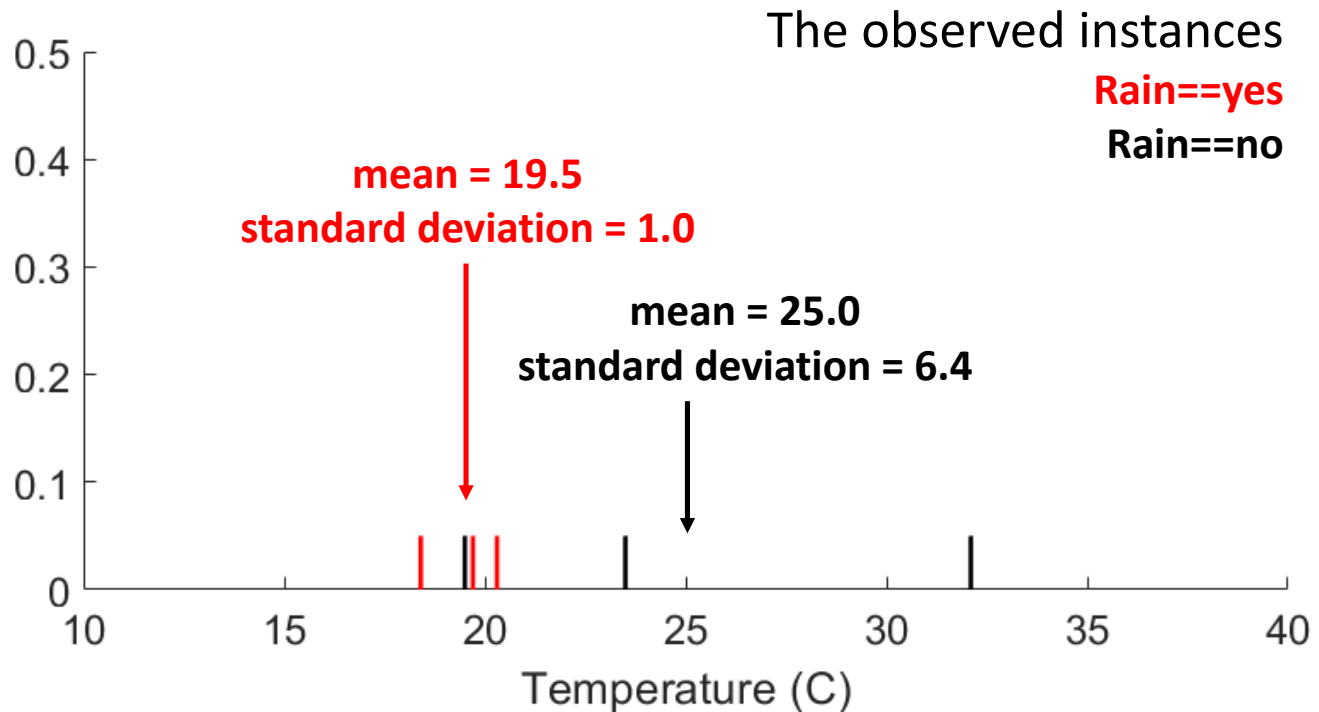
Mean:

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

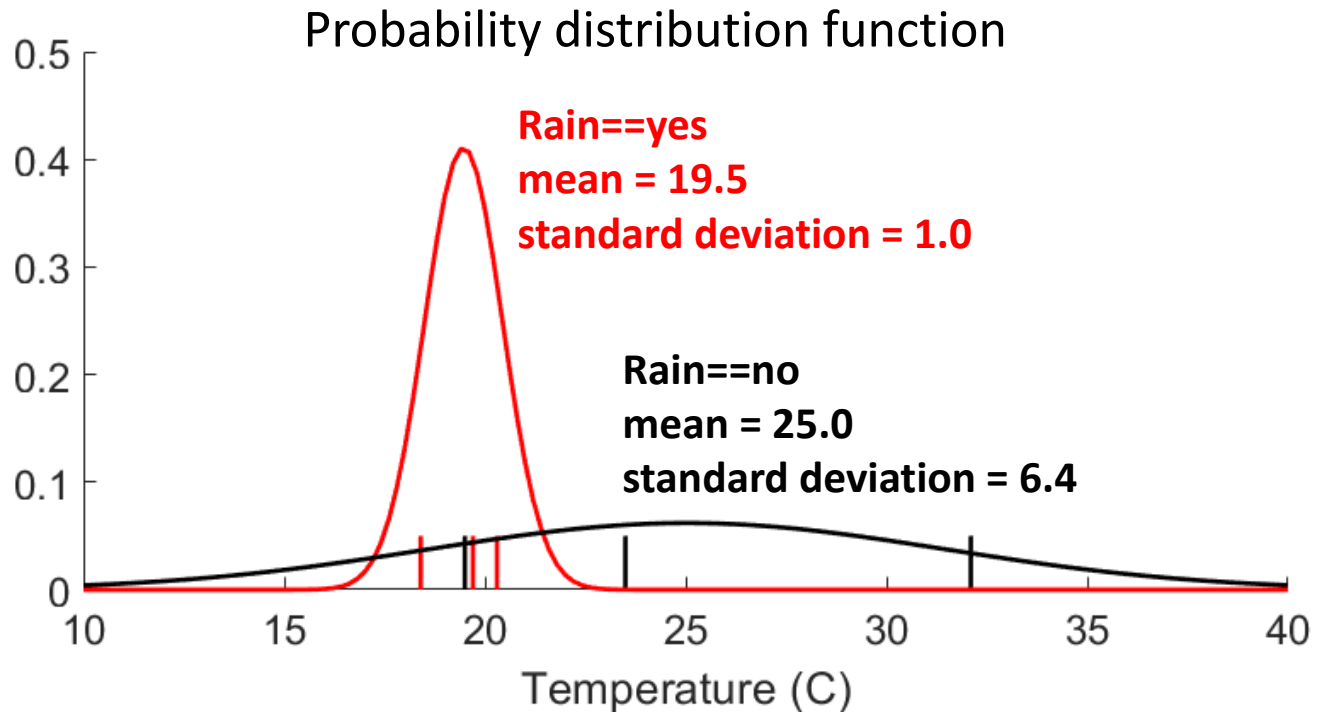
Standard deviation:

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

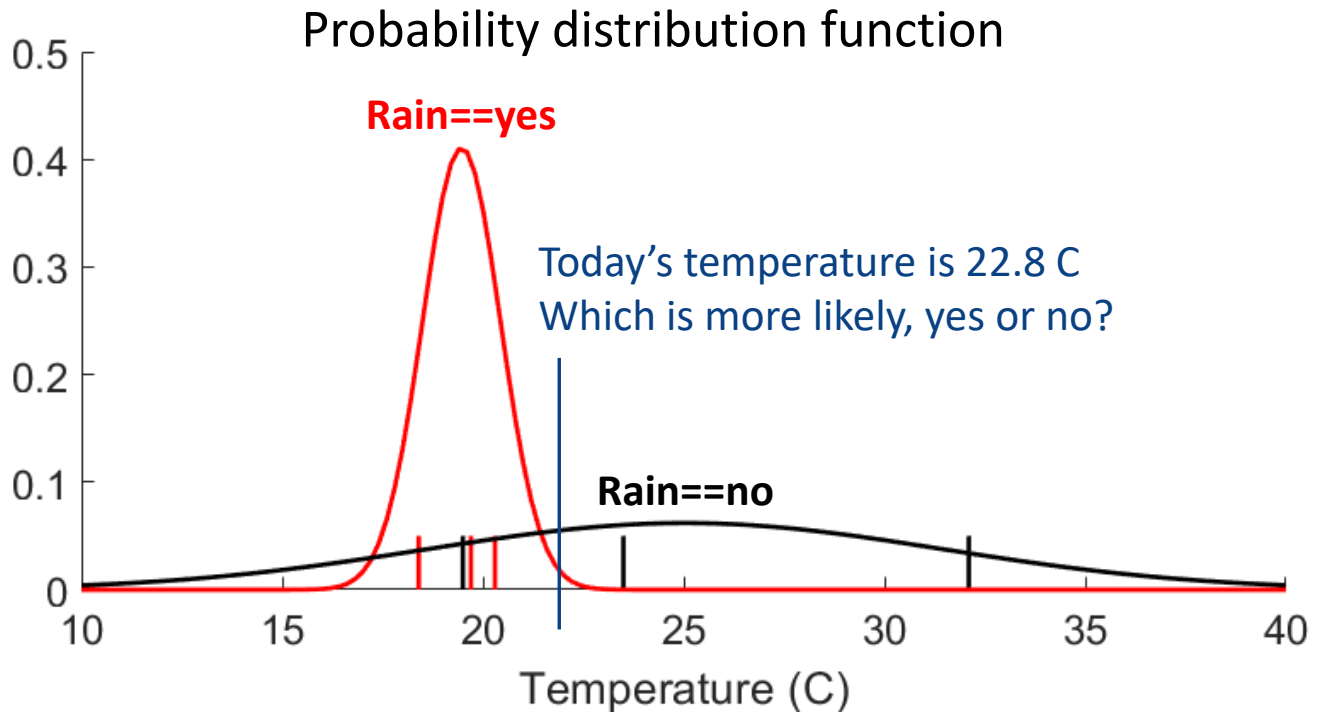
Example: Gaussian naïve Bayes



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:

$$\begin{aligned} P(temp = 22.8 | rain = no) &= P(rain == no) \phi_{\sigma_{no}}(22.8 - \mu_{no}) \\ &= (0.5) \frac{1}{6.4\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{22.8-25.0}{6.4}\right)^2} = \mathbf{0.0292} \end{aligned}$$

$$\begin{aligned} P(temp = 22.8 | rain = yes) &= P(rain == yes) \phi_{\sigma_{yes}}(22.8 - \mu_{yes}) \\ &= (0.5) \frac{1}{1.0\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{22.8-19.5}{1.0}\right)^2} = \mathbf{0.0006} \end{aligned}$$

Gaussian Naïve Bayes

- 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

- If x_i is a numeric attribute, $P(x_i | c_j)$ can be represented with a Gaussian distribution
- But not all real-world data follows a Gaussian distribution...

Example: Detect boundaries

- Problem: classify boundary type from elevation

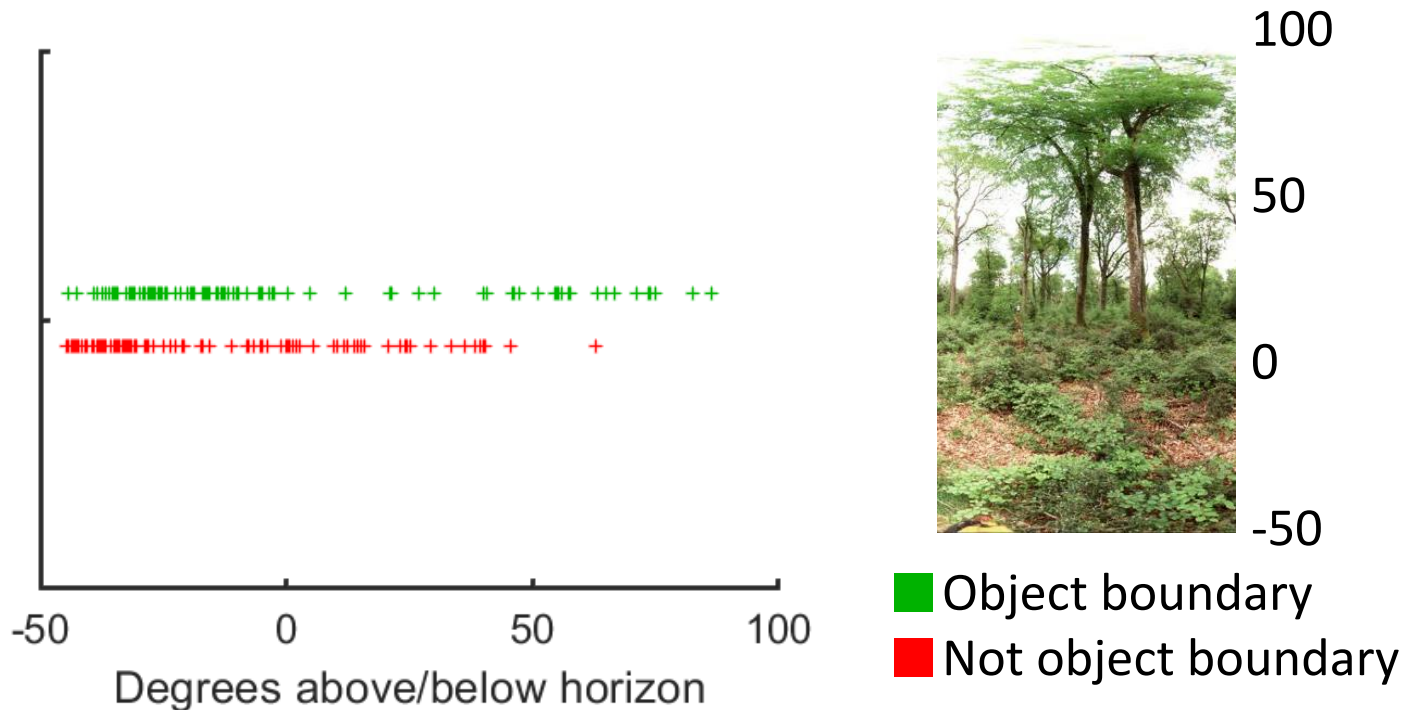


Image: SYNS dataset (<https://syms.soton.ac.uk/>)

Example: Detect boundaries

- Are these distributions Gaussian?

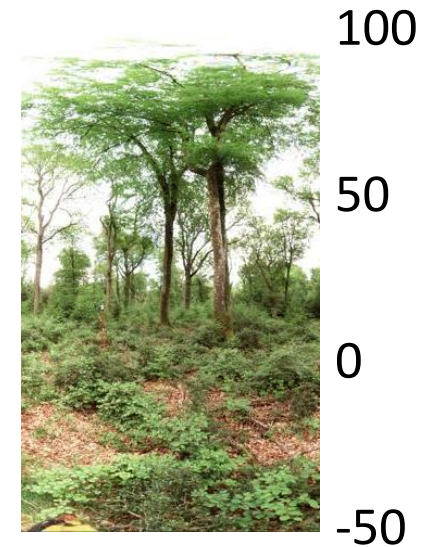
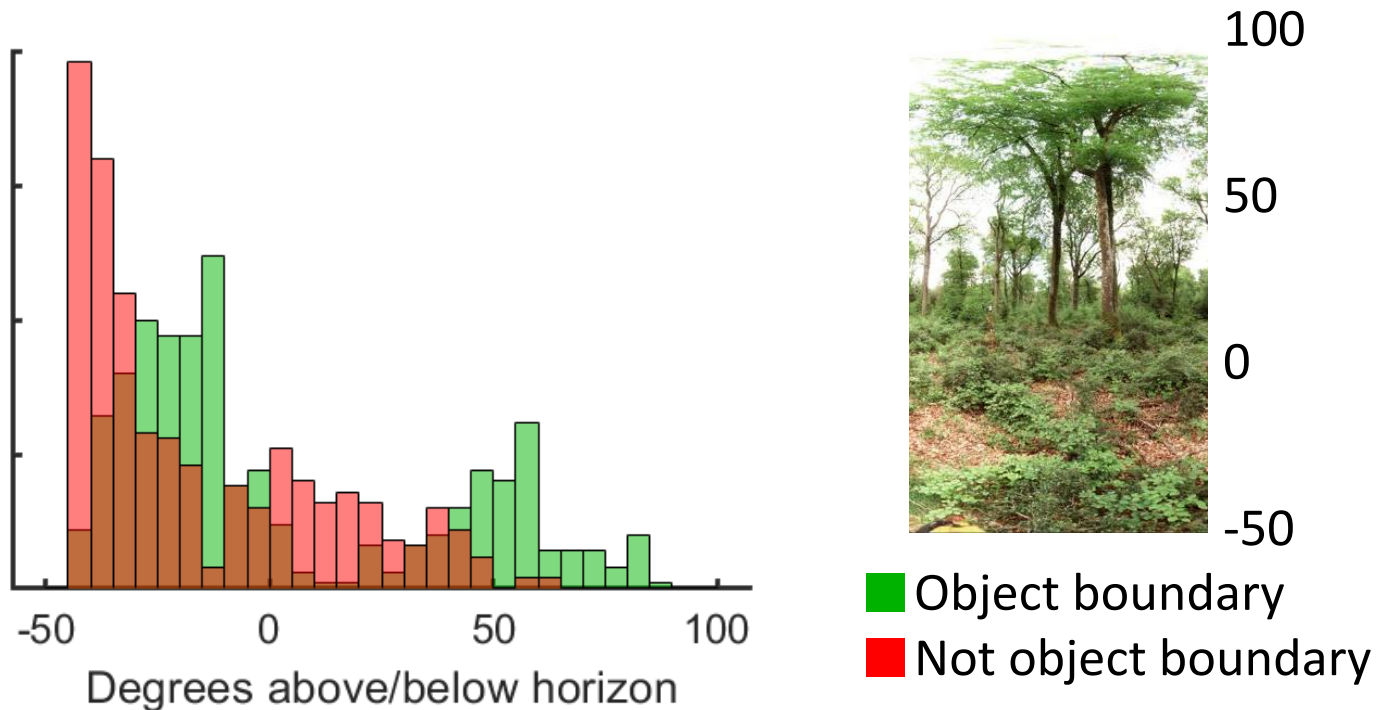


Image: SYNS dataset (<https://syms.soton.ac.uk/>)

Kernel density estimation (KDE)

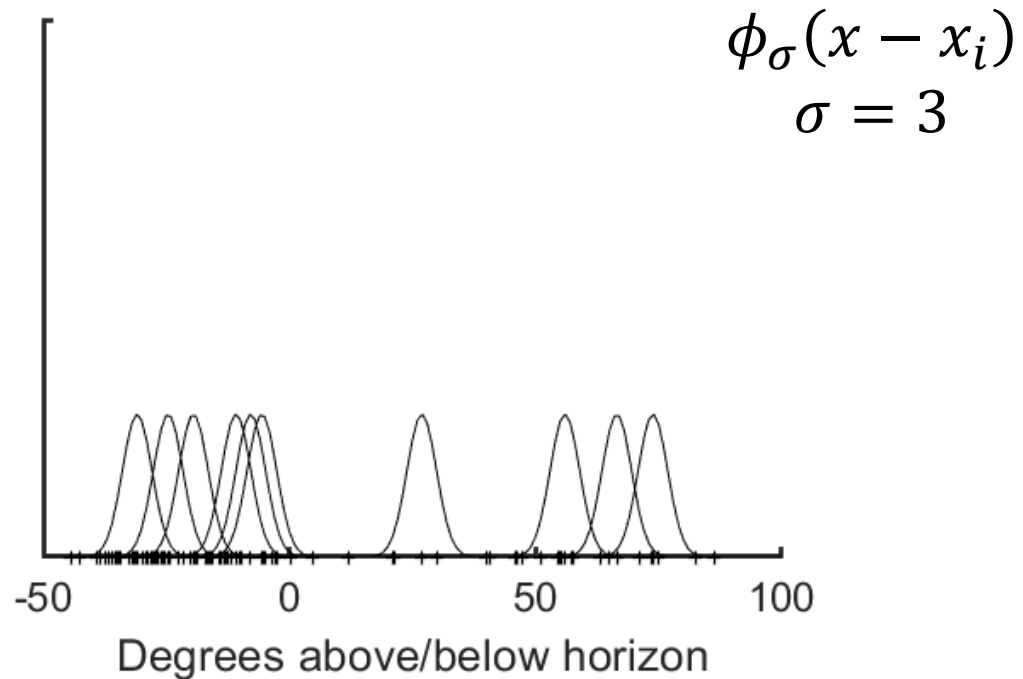
- Kernel density estimate of the distribution:

$$\hat{f}(x) = \frac{1}{N} \sum_{i=1}^N \phi_{\sigma}(x - x_i)$$

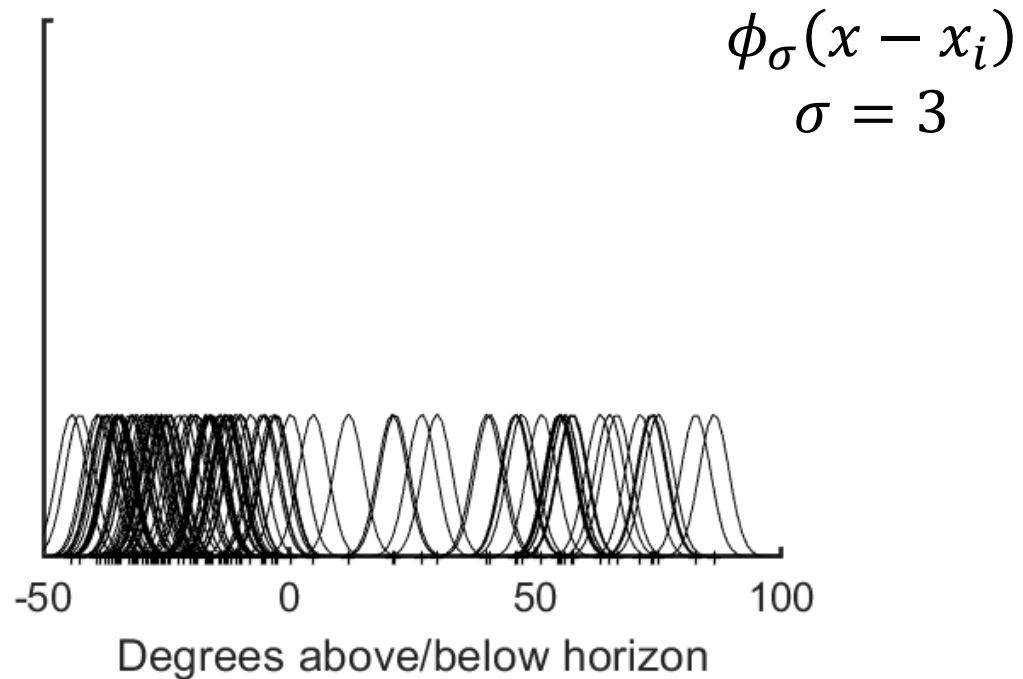
where ϕ_{σ} is the Gaussian distribution with mean of 0 and standard deviation of σ

- Learn the probability distribution from the data instead of assuming a specific distribution (e.g., Gaussian)

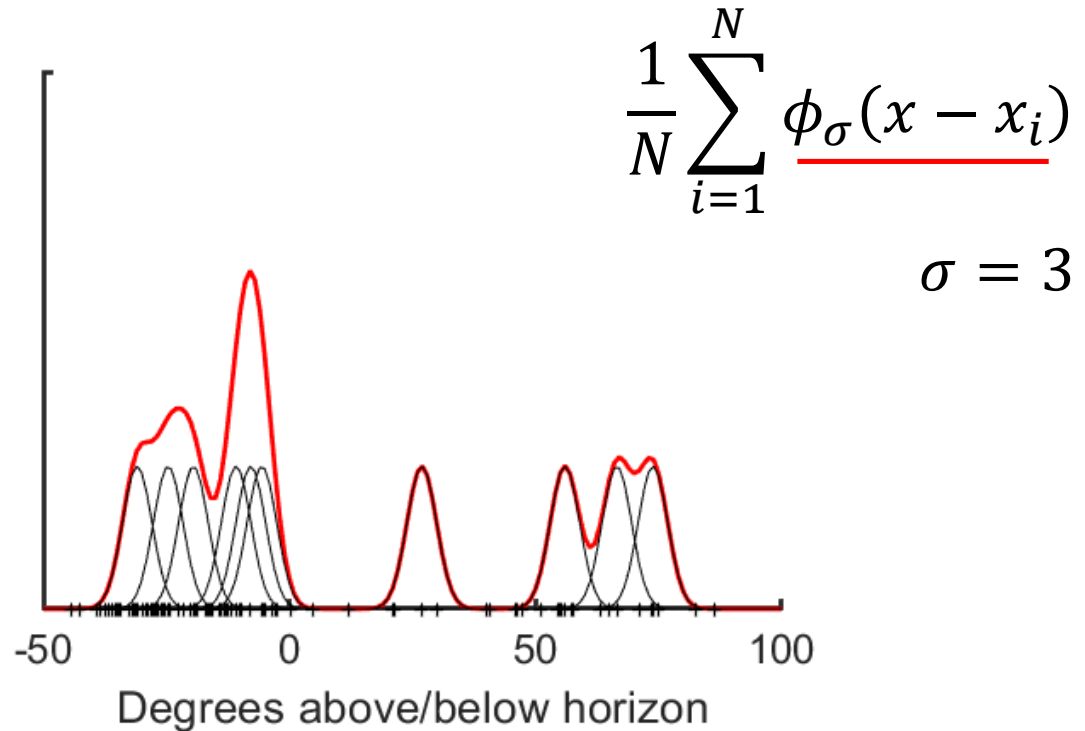
Kernel density estimation (KDE)



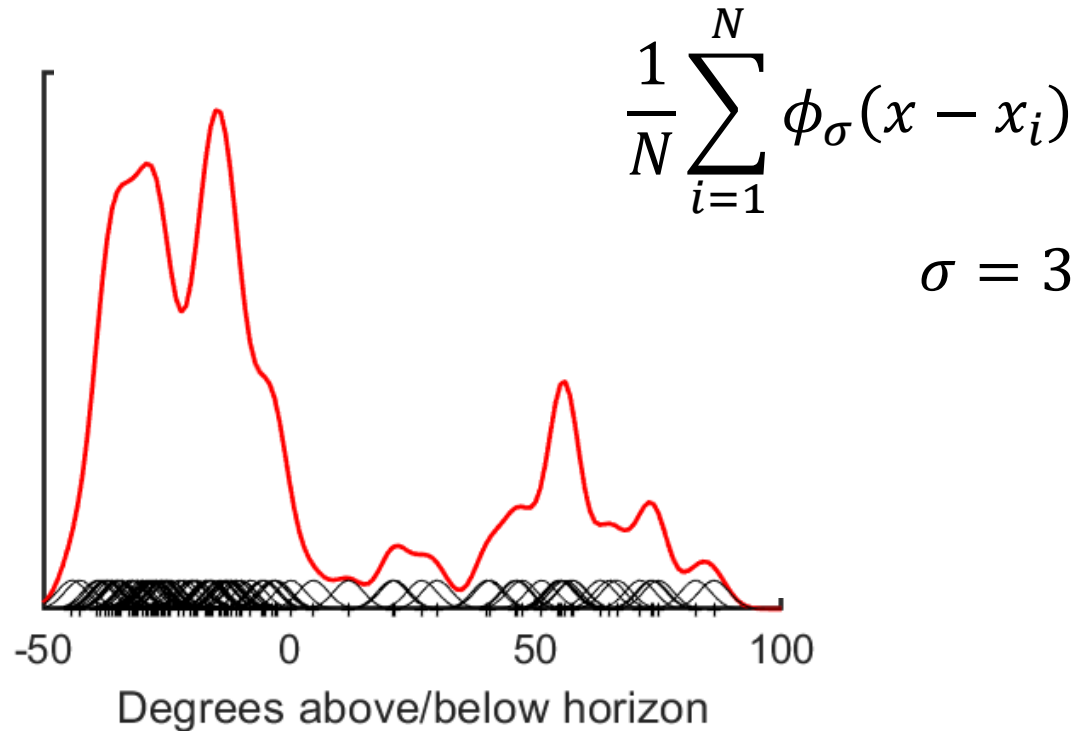
Kernel density estimation (KDE)



Kernel density estimation (KDE)

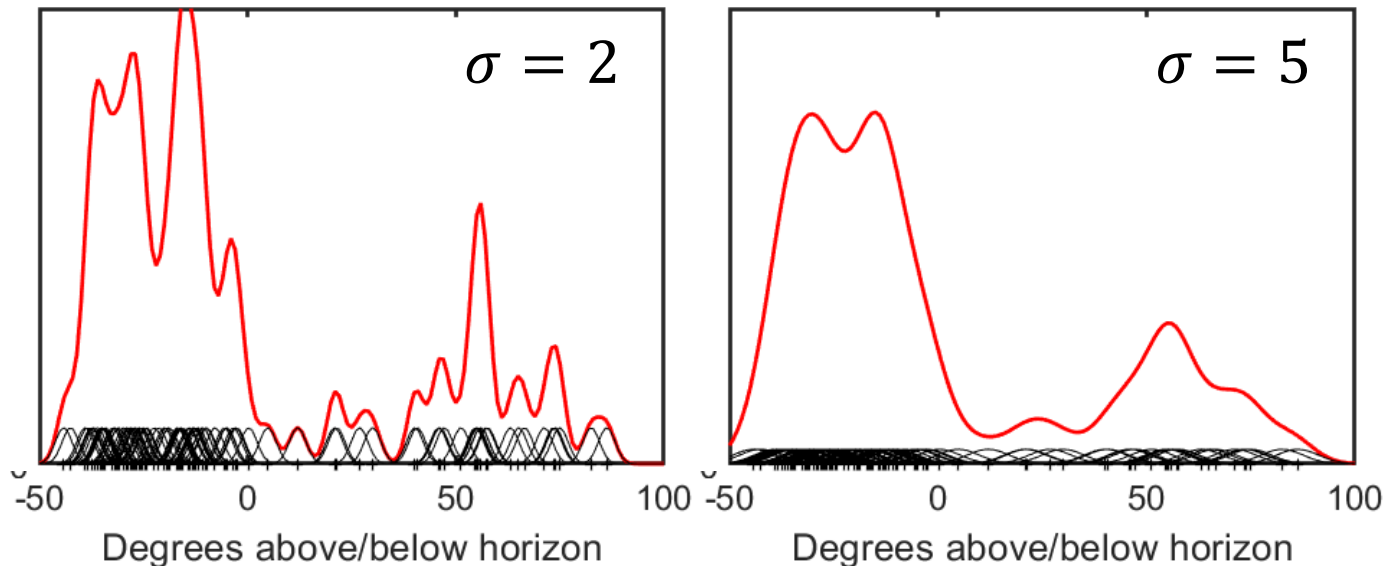


Kernel density estimation (KDE)



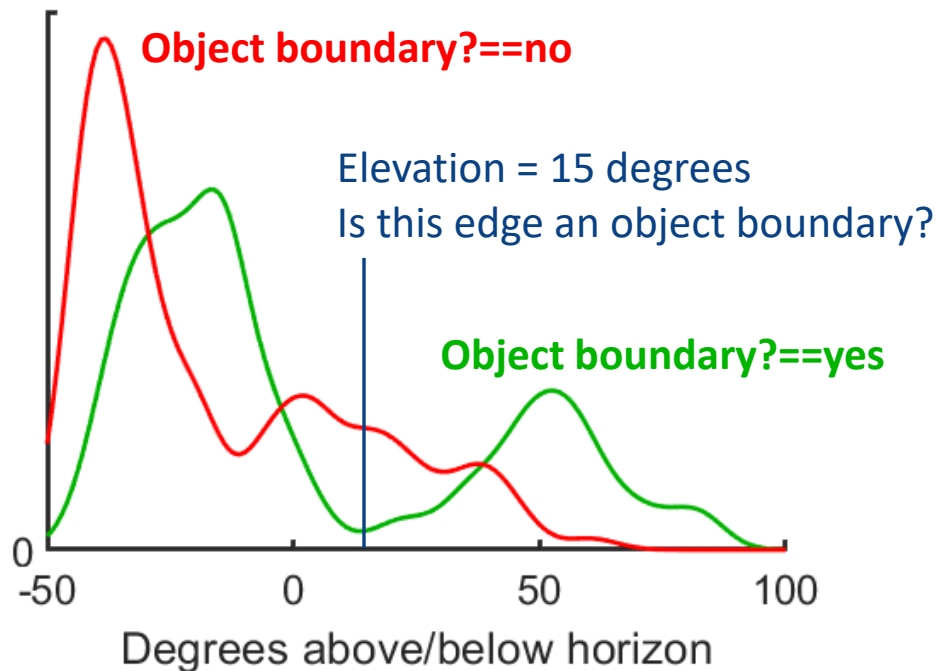
KDE: Kernel bandwidth

- KDE has one parameter: the standard deviation of the Gaussian σ (= “kernel bandwidth”)
- Parameter must be chosen by user



Example: KDE naïve Bayes

Probability distribution function



KDE Naïve Bayes

- Which class is more likely for test item x_{test} ?
- We only have one feature x (elevation), so choose the class c_j that maximizes $P(c_j)P(x_{test}|c_j)$
- Compute likelihood using KDE:

$$P(x_{test}|c_{yes}) = \frac{1}{N} \sum_{n=1}^N \phi_{\sigma_{yes}}(x_{test} - x_n)$$

$\arg \max (P(c_j) P(x_{test}|c_j))$
 $\arg \max [P(c_j) \cdot \prod_i P(x_i | c_j)]$
 $\log P(c_j) + \sum_i \log P(x_i | c_j)$
 $\downarrow \frac{1}{N} \sum_{n=1}^N \phi_{\sigma}(x_{test} - x_n)$

x_1, \dots, x_n = elevations of the N “yes” examples in the training set

$$P(x_{test}|c_{no}) = \frac{1}{M} \sum_{m=1}^M \phi_{\sigma_{no}}(x_{test} - x_m)$$

x_1, \dots, x_m = elevations of the M “no” examples in the training set

Example: KDE naïve Bayes

- Compute the probability of the test instance using KDE:

$$\begin{aligned} P(x = 15|yes) &= P(yes) \frac{1}{N} \sum_{n=1}^N \phi_{\sigma_{yes}}(x_{test} - x_n) \\ &= (0.266) \left(\frac{1}{266} \right) \sum_{n=1}^{266} \phi_5(15 - x_n) = \mathbf{0.00029} \end{aligned}$$

$$\begin{aligned} P(x = 15|no) &= P(no) \frac{1}{M} \sum_{m=1}^M \phi_{\sigma_{no}}(x_{test} - x_m) \\ &= (0.734) \left(\frac{1}{734} \right) \sum_{n=1}^{734} \phi_5(15 - x_m) = \mathbf{0.00497} \end{aligned}$$

KDE summary

- Advantages:
 - Model arbitrary probability distributions
 - No assumptions about the shape of the distribution (e.g., Gaussian)
- Disadvantages:
 - Need to choose a kernel bandwidth
 - Need many parameters to represent the PDF; slow to compute probability at new points

Naïve Bayes summary

- Naïve Bayes works with various data types
 - Different ways to compute $P(x_i | c_j)$ for different types of attributes
- 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

Types of naïve Bayes

- **Multivariate:** attributes are nominal and can take any of a fixed number of values
- Binomial (or Bernoulli): attributes are binary
 - Special case of multivariate
- Multinomial: attributes are natural numbers corresponding to frequency
 - Probability $P(a_k = m | c_j) \approx P(a_k = 1 | c_j)^m / (m!)$
- **Gaussian:** 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