Friedrich-Alexander-Universität
Technische Fakultät

7. Classification

Knowledge Discovery in Databases with Exercises

Dominik Probst, dominik.probst@fau.de
Computer Science 6 (Data Management), Friedrich-Alexander-Universität Erlangen-Nürnberg
Summer semester 2025

Outline

1. **Basic Concepts**
2. **Decision Tree Induction**
   - Information Gain (ID3)
   - Gain Ratio (C4.5)
   - Gini Index (CART)
3. **Rule-Based Classification**
4. **Bayes Classification Methods**
5. **Model Evaluation**
   - Evaluation Metrics
   - Evaluation Strategies
6. **Ensemble Methods**
7. **Summary**

Basic Concepts

Supervised vs. Unsupervised Learning

Supervised Learning

- The **training data** (observations, measurements, etc.) are accompanied by **labels** indicating the **class** of the observations.
- New data is classified based on a **model** created from the training data.

Unsupervised Learning

- Class labels of training data are unknown. Or rather, there are no training data.
- Given a set of measurements, observations, etc., the goal is to find classes or clusters in the data.
  $\longrightarrow$ Clustering.

Classification vs. Numerical Prediction

Friedrich-Alexander-Universität
Technische Fakultät

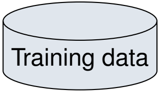
- **Classification:**
  - Predicts **categorical class labels** (discrete, nominal).
  - Constructs a model based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data.
- **Numerical prediction:**
  - Models **continuous-valued functions**.
  - I.e. predicts missing or unknown (future) values.
- **Typical applications of classification:**
  - Credit/loan approval: Will it be paid back?
  - Medical diagnosis: Is a tumor cancerous or benign?
  - Fraud detection: Is a transaction fraudulent or not?
  - Web-page categorization: Which category is it such as to categorize it by topic.

Classification – A Two-step Process

Friedrich-Alexander-Universität
Technische Fakultät

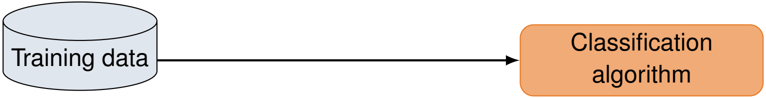
1. **Model construction: describing a set of predetermined classes:**
    - Each tuple/sample is assumed to belong to a predefined class, as determined by the **class-label attribute**.
    - The set of tuples used for model construction is the **training set**.
    - The **model** is represented as classification rules, decision trees, or mathematical formulae.
2. **Model usage, for classifying future or unknown objects:**
    - Estimate **accuracy** of the model:
        - The known label of **test samples** is compared with the result from the model.
        - **Accuracy rate** is the percentage of test-set samples that are correctly classified by the model.
        - Test set is independent of training set (otherwise overfitting).
        - More on evaluation metrics later.
    - If the accuracy is acceptable, **use the model** to classify data tuples whose class labels are not known.

Classification – 1. Model Construction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Classification – 1. Model Construction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Classification – 1. Model Construction

Training data → Classification algorithm → Classification rules

Training data:

NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Classification algorithm

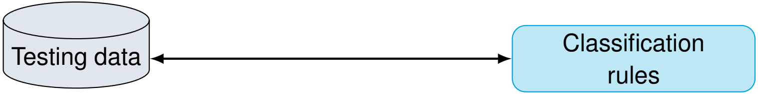
Classification rules:

```
if RANK ='Professor'
or YEARS >6
then TENURED ='yes'
```

Classification – 2. Model Usage

Classification rules

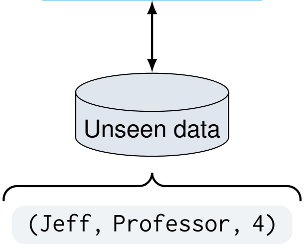
Classification – 2. Model Usage



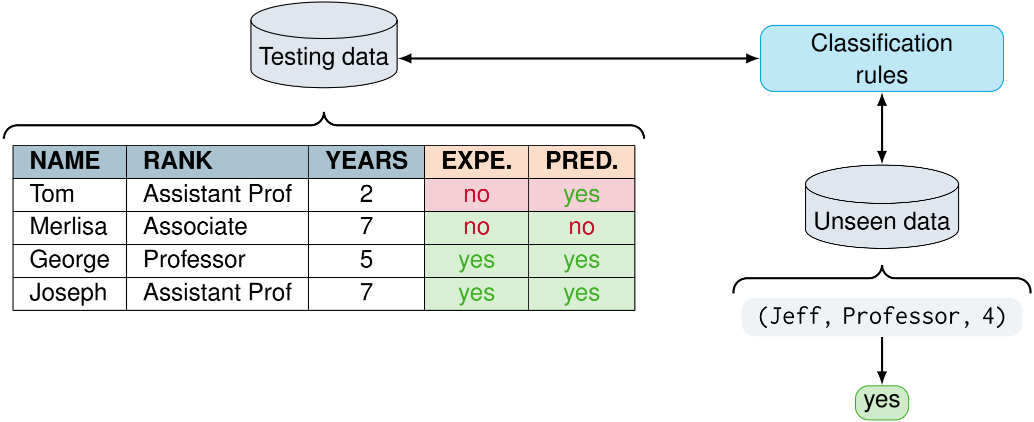
NAME	RANK	YEARS	EXPE.	PRED.
Tom	Assistant Prof	2	no	yes
Merlisa	Associate	7	no	no
George	Professor	5	yes	yes
Joseph	Assistant Prof	7	yes	yes

Classification – 2. Model Usage

NAME	RANK	YEARS	EXPE.	PRED.
Tom	Assistant Prof	2	no	yes
Merlisa	Associate	7	no	no
George	Professor	5	yes	yes
Joseph	Assistant Prof	7	yes	yes



Classification – 2. Model Usage

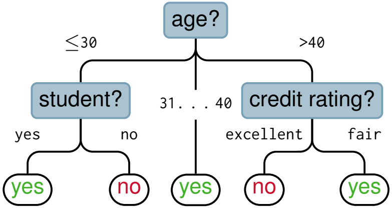


NAME	RANK	YEARS	EXPE.	PRED.
Tom	Assistant Prof	2	no	yes
Merlisa	Associate	7	no	no
George	Professor	5	yes	yes
Joseph	Assistant Prof	7	yes	yes

Decision Tree Induction

Decision Tree: An Example

- **Training dataset:** buys_computer.
- **Resulting tree:**



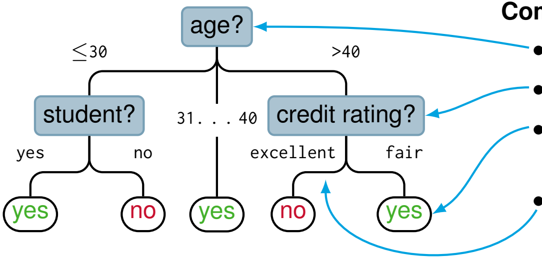
age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
$31 \ldots 40$	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
$31 \ldots 40$	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
$31 \ldots 40$	medium	no	excellent	yes
$31 \ldots 40$	high	yes	fair	yes
> 40	medium	no	excellent	no

Decision Tree Induction

Decision tree induction refers to the learning of a decision-tree based on labeled training data.

Decision Tree

A *decision tree* is a flowchart-like structure consisting of interconnected internal and leaf nodes.



Components of a Decision Tree

- **Root**: topmost node.
- **Internal node**: test on an attribute.
- **Leaf node**: holds a class label, also called *terminal node*.
- **Branch**: outcome of a leaf node's test coupled with a text. In this example: `excellent`.

Algorithm for Decision Tree Induction (I)

Construction in a *top-down recursive* and *divide-and-conquer* manner.

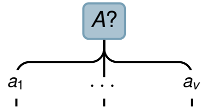
Input: data partition D, `attribute_list`, `attribute_selection_method`.

Algorithm Sketch `build_decision_tree`:

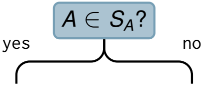
1. Create node N.
2. Determine splitting attribute A with `attribute_selection_method`.
3. Label N with splitting criterion.
4. If the splitting attribute has been fully utilized, remove it from `attribute_list`.
5. For each outcome of splitting criterion:
   - Partition D according to outcome of splitting criterion.
   - Grow branches on N for each partition.
6. Return node N

Attribute Types:

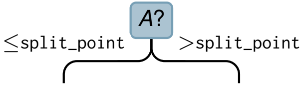
Discrete:



Discrete & Binary Tree:



Continuous:



Algorithm for Decision Tree Induction (II)

Stopping criteria:

- All samples in D belong to the same class:
  $\Rightarrow$ N becomes a leaf.
- Samples in D belong to multiple classes, but `attribute_list` is empty:
  $\Rightarrow$ Create leaf with majority class.
- Partition D is empty:
  $\Rightarrow$ Create leaf with majority class.

Decision Tree Algorithm

A detailed algorithm to construct a decision tree is covered in the appendix and in our reference book[1].

[1] J. Han et al., *Data Mining: Concepts and Techniques*, 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011, ISBN: 0123814790, pp. 332 – 335.

Attribute Selection Methods

An *attribute selection method* is a heuristic to determine the “best” splitting criterion to partition data.

- Also known as *splitting rules*.
- Provides ranking for each attribute.
- Partition data based on attribute with best score.
- Tree node is labeled with splitting criterion (attribute).

We will discuss **three** popular attribute selection methods[2]:

1. Information Gain
2. Gain Ratio
3. Gini Index

[2] Since this it not even close to an exhaustive list, we list some other popular methods in the appendix.

Decision Tree Induction

Information Gain (ID3)

Information Gain (ID3) (I)

- Partitions reflect least randomness, i. e. impurity.
- **Expected information** (entropy) needed to classify a tuple in D:

$$\text{Info}(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

  - p_i is the probability that tuple in D belongs to class C_i, estimated by $\frac{|C_i|}{|D|}$.
  - $\log_2$ because information is encoded in bits.

Information Gain (ID3) (II)

Calculate information for every attribute in `attribute_list` and data partition D:

Information Gain (ID3) (II)

Calculate information for every attribute in `attribute_list` and data partition D:

Discrete-valued Attribute

- Attribute A with v distinct values.
- Expected information required to classify tuple in D based on partitioning by A.
- D_A: dataset D partitioned by A, $D_{A,j}$: dataset D partitioned by A with distinct value j.

$$\text{Info}_A(D) = \sum_{j=1}^{v} \frac{|D_{A,j}|}{|D_A|} \text{Info}(D_{A,j})$$

Information Gain (ID3) (II)

Calculate information for every attribute in `attribute_list` and data partition D:

Discrete-valued Attribute

- Attribute A with v distinct values.
- Expected information required to classify tuple in D based on partitioning by A.
- D_A: dataset D partitioned by A, $D_{A,j}$: dataset D partitioned by A with distinct value j.

$$\text{Info}_A(D) = \sum_{j=1}^{v} \frac{|D_{A,j}|}{|D_A|} \text{Info}(D_{A,j})$$

Continuous-valued Attribute

- Attribute A with v distinct values.
- Order values in increasing order.
- Calculate midpoint of every neighbouring value.
- $v - 1$ possible split points $s_i = \frac{a_i + a_{i+1}}{2}$.
- Evaluate $\text{Info}_A(D)$ for every possible binary splitting ($A \leq s_i$ and $A > s_i$).

$$\text{Info}_A(D) = \frac{|D_{A \leq s_i}|}{|D_A|} \text{Info}(D_{A \leq s_i}) + \frac{|D_{A > s_i}|}{|D_A|} \text{Info}(D_{A > s_i})$$

Information Gain (ID3) (III)

Given $\text{Info}(D)$ and $\text{Info}_A(D)$, Information Gain of each attribute A is calculated as:

$$\text{Gain}(A) = \text{Info}(D) - \text{Info}_A(D)$$

Information Gain

The attribute with the **highest** Information Gain is selected as the splitting attribute.

Information Gain (ID3) - Example (I)

- **Target attribute:** buys_computer
- **Expected Information**:

$$\text{Info}(D) = I(9,5)$$
$$= -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right)$$
$$= 0.94$$

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
$31 \ldots 40$	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
$31 \ldots 40$	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
$31 \ldots 40$	medium	no	excellent	yes
$31 \ldots 40$	high	yes	fair	yes
> 40	medium	no	excellent	no

- **Attribute:** Age
- **Value distribution:**

Age	Yes	No	$I(\text{Yes}, \text{No})$
≤ 30	2	3	0.971
$31\dots40$	4	0	0
> 40	3	2	0.971

- **Calculation:**

$$\text{Info}_{\text{age}}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2)$$
$$= 0.694$$
$$\text{Gain}(\text{age}) = \text{Info}(D) - \text{Info}_{\text{age}}(D)$$
$$= 0.94 - 0.694$$
$$= 0.246$$

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
$31\dots40$	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
$31\dots40$	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
$31\dots40$	medium	no	excellent	yes
$31\dots40$	high	yes	fair	yes
> 40	medium	no	excellent	no

Information Gain (ID3) - Example (II)

- **Attribute:** Age
- **Value distribution:**

Age	Yes	No	$I(\text{Yes}, \text{No})$
≤ 30	2	3	0.971
$31 \ldots 40$	4	0	0
> 40	3	2	0.971

- **Calculation:**

$$\begin{aligned}\text{Info}_{\text{age}}(D) &= \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) \\ &= 0.694 \\ \text{Gain}(\text{age}) &= \text{Info}(D) - \text{Info}_{\text{age}}(D) \\ &= 0.94 - 0.694 \\ &= 0.246\end{aligned}$$

- **Gain of other attributes:**
  - $\text{Gain}(\texttt{income}) = 0.029$,
  - $\text{Gain}(\texttt{student}) = 0.151$,
  - $\text{Gain}(\texttt{credit\_rating}) = 0.048$.

Information Gain (ID3) - Example (II)

- **Attribute:** Age
- **Value distribution:**

Age	Yes	No	$I(\text{Yes}, \text{No})$
≤ 30	2	3	0.971
$31 \ldots 40$	4	0	0
> 40	3	2	0.971

- **Calculation:**

$$\begin{aligned} \text{Info}_{\text{age}}(D) &= \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) \\ &= 0.694 \\ \text{Gain}(\text{age}) &= \text{Info}(D) - \text{Info}_{\text{age}}(D) \\ &= 0.94 - 0.694 \\ &= 0.246 \end{aligned}$$

- **Gain of other attributes:**
  - $\text{Gain}(\texttt{income}) = 0.029$,
  - $\text{Gain}(\texttt{student}) = 0.151$,
  - $\text{Gain}(\texttt{credit\_rating}) = 0.048$.
- **Splitting attribute:**
  Age with $\text{Gain}(\texttt{age}) = 0.246$.

Information Gain (ID3) - Example (III)

age?

≤ 30

income	student	credit_rating	buys_computer
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

31. . . 40

income	student	credit_rating	buys_computer
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

>40

income	student	credit_rating	buys_computer
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

Decision Tree Induction

Gain Ratio (C4.5)

Gain Ratio (C4.5)

- **Problem of Information Gain:**
  - Tends to prefer attributes with large number of distinct values.
    - E. g. attribute `degree_program` with 278 values vs. `student_status` with 2 values.

Gain Ratio (C4.5)

- **Problem of Information Gain:**
  - Tends to prefer attributes with large number of distinct values.
    - E. g. attribute `degree_program` with 278 values vs. `student_status` with 2 values.
- **Solution:**
  - Normalize the Information Gain to get the **Gain Ratio (C4.5)**:

$$\text{SplitInfo}_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \log_2\left(\frac{|D_j|}{|D|}\right)$$

$$\text{GainRatio}(A) = \frac{\text{Gain}(A)}{\text{SplitInfo}_A(D)}$$

  - **Disadvantage:** Becomes unstable as $\text{SplitInfo}_A(D)$ approaches zero.

Gain Ratio

The attribute with the **highest** Gain Ratio is selected as the splitting attribute.

Gain Ratio (C4.5) - Example

- **Attribute:** Age
- **Calculation:**

$$\text{Gain}(\text{age}) = 0.246$$

$$\text{SplitInfo}_{age}(D) = -\frac{5}{14}\log_2\left(\frac{5}{14}\right) - \frac{4}{14}\log_2\left(\frac{4}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 1.577$$

$$\text{GainRatio}(\text{age}) = \frac{0.246}{1.577} = 0.156$$

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
31 . . . 40	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
31 . . . 40	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
31 . . . 40	medium	no	excellent	yes
31 . . . 40	high	yes	fair	yes
> 40	medium	no	excellent	no

Decision Tree Induction

Gini Index (CART)

Gini Index (CART) (I)

- **Problem of Information Gain and Gain Ratio:**
  - Use of logarithm is computationally expensive.
- **Solution:**
  - Use the **Gini Index (CART)** as an alternative to Information Gain and Gain Ratio.
  - Measures the **statistical dispersion** of a dataset.

Gini Index (CART) (II)

- **Measured impurity** of partition D is defined as the sum over n classes:

$$\text{Gini}(D) = 1 - \sum_{j=1}^{n} p_j^2$$

  - p_j is the non-zero probability that sample in D belongs to class C_j as estimated by $\frac{|C_{j,D}|}{|D|}$

Gini Index (CART) (II)

- **Measured impurity** of partition D is defined as the sum over n classes:

$$\text{Gini}(D) = 1 - \sum_{j=1}^{n} p_j^2$$

  - p_j is the non-zero probability that sample in D belongs to class C_j as estimated by $\frac{|C_{j,D}|}{|D|}$

Gini Index (CART) (II)

- **Measured impurity** of partition D is defined as the sum over n classes:

$$\text{Gini}(D) = 1 - \sum_{j=1}^{n} p_j^2$$

  - p_j is the non-zero probability that sample in D belongs to class C_j as estimated by $\frac{|C_{j,D}|}{|D|}$

Discrete-valued Attribute

- Attribute A with v distinct values.
- Compute all possible subsets of values.
- Compute weighted sum of each partition tuple (D_1 and D_2) as follows:

$$\text{Gini}_A(D) = \frac{|D_1|}{|D|}\text{Gini}(D_1) + \frac{|D_2|}{|D|}\text{Gini}(D_2)$$

Gini Index (CART) (II)

- **Measured impurity** of partition D is defined as the sum over n classes:

$$\text{Gini}(D) = 1 - \sum_{j=1}^{n} p_j^2$$

  - p_j is the non-zero probability that sample in D belongs to class C_j as estimated by $\frac{|C_{j,D}|}{|D|}$

Discrete-valued Attribute

- Attribute A with v distinct values.
- Compute all possible subsets of values.
- Compute weighted sum of each partition tuple (D_1 and D_2) as follows:

$$\text{Gini}_A(D) = \frac{|D_1|}{|D|}\text{Gini}(D_1) + \frac{|D_2|}{|D|}\text{Gini}(D_2)$$

Continuous-valued Attribute

- Order values in increasing order.
- Split the dataset at every midpoint.
- Evaluate $\text{Gini}_A(D)$ for every possible binary splitting:

$$\text{Gini}_A(D) = \frac{|D_1|}{|D|}\text{Gini}(D_1) + \frac{|D_2|}{|D|}\text{Gini}(D_2)$$

Gini Index (CART) (III)

- The **reduction in impurity** is defined as follows:

$$\Delta \text{Gini}_A(D) = \text{Gini}(D) - \text{Gini}_A(D).$$

Gini Index

The attribute (and partitioning) with the **highest** reduction in impurity is selected as the splitting attribute.

Gini Index (CART) - Example (I)

- **Target attribute:** buys_computer
- **Measured impurity**:

$$\text{Gini}(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2$$
$$= 0.459$$

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
$31 \ldots 40$	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
$31 \ldots 40$	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
$31 \ldots 40$	medium	no	excellent	yes
$31 \ldots 40$	high	yes	fair	yes
> 40	medium	no	excellent	no

Gini Index (CART) - Example (II)

- **Attribute:** Income
- **Subsets:**

$$D_1 : \{\texttt{low,medium}\}$$
$$D_2 : \{\texttt{high}\}$$

- **Calculation:**

$$\begin{aligned}
\text{Gini}(D|_{i=\{high\}}) &= \text{Gini}(D|_{i=\{medium,low\}}) \\
&= \frac{10}{14}\text{Gini}(D_1) + \frac{4}{14}\text{Gini}(D_2) \\
&= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) \\
&+ \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right) \\
&= 0.443
\end{aligned}$$

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
$31 \ldots 40$	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
$31 \ldots 40$	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
$31 \ldots 40$	medium	no	excellent	yes
$31 \ldots 40$	high	yes	fair	yes
> 40	medium	no	excellent	no

Gini Index (CART) - Example (II)

- **Attribute:** Income
- **Subsets:**

$$D_1 : \{\texttt{low}, \texttt{medium}\}$$
$$D_2 : \{\texttt{high}\}$$

- **Calculation:**

$$\begin{aligned}
\text{Gini}(D|_{\mathtt{i}=\{high\}}) =& \text{Gini}(D|_{\mathtt{i}=\{medium, low\}}) \\
=& \frac{10}{14}\text{Gini}(D_1) + \frac{4}{14}\text{Gini}(D_2) \\
=& \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) \\
&+ \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right) \\
=& 0.443
\end{aligned}$$

- **Reduction in impurity:**

$$\begin{aligned}
\Delta\text{Gini}_{\mathtt{i}=\{high\}}(D) =& \text{Gini}(D) - \text{Gini}(D|_{\mathtt{i}=\{high\}}) \\
=& 0.459 - 0.443 \\
=& 0.016
\end{aligned}$$

- **Other subsets:**
  - $\Delta\text{Gini}_{\mathtt{i}=\{medium\}}(D) = 0.001$
  - $\Delta\text{Gini}_{\mathtt{i}=\{low\}}(D) = 0.009$

Gini Index (CART) - Example (II)

- **Attribute:** Income
- **Subsets:**

$$D_1 : \{\texttt{low}, \texttt{medium}\}$$
$$D_2 : \{\texttt{high}\}$$

- **Calculation:**

$$\begin{aligned}
\text{Gini}(D|_{i=\{high\}}) &= \text{Gini}(D|_{i=\{medium,low\}}) \\
&= \frac{10}{14}\text{Gini}(D_1) + \frac{4}{14}\text{Gini}(D_2) \\
&= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) \\
&+ \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right) \\
&= 0.443
\end{aligned}$$

- **Reduction in impurity:**

$$\begin{aligned}
\Delta\text{Gini}_{i=\{high\}}(D) &= \text{Gini}(D) - \text{Gini}(D|_{i=\{high\}}) \\
&= 0.459 - 0.443 \\
&= 0.016
\end{aligned}$$

- **Other subsets:**
  - $\Delta\text{Gini}_{i=\{medium\}}(D) = 0.001$
  - $\Delta\text{Gini}_{i=\{low\}}(D) = 0.009$
- **Splitting subset:**
  - If income has the overall highest reduction of impurity, then the split is on {"low","medium"} and {"high"}.

Decision Tree Induction

Overview

Attribute Selection Methods Overview

The three methods, in general, return good results, but

- **Information Gain:**
  - Biased towards multi-valued attributes.
- **Gain Ratio:**
  - Tends to prefer unbalanced splits in which one partition is much smaller than the others.
- **Gini Index:**
  - Biased to multi-valued attributes.
  - Has difficulty when number of classes is large.
  - Tends to favor tests that result in equal-sized partitions and purity in both partitions.

Decision Tree Induction - Overfitting

- **Problem:**
  - Many branches may reflect anomalies due to noise or outliers.
  - Overall poor accuracy for unseen samples.

$\Rightarrow$ **Overfitting:** An induced tree may overfit the training data.

Decision Tree Induction - Overfitting

- **Problem:**
  - Many branches may reflect anomalies due to noise or outliers.
  - Overall poor accuracy for unseen samples.

$\Rightarrow$ **Overfitting:** An induced tree may overfit the training data.

- **Solution:**
  - **Pruning:** Avoid branches that have little importance.
  - **Two** types of pruning:
    1. **Prepruning:** Stop growing the tree early.
    2. **Postpruning:** Remove branches from a "fully grown" tree.

Decision Tree Induction - Enhancements

- A lot of research has been done to improve basic **decision tree induction** algorithms. E.g.:
  - **Allow for continuous-valued attributes.**
    - Dynamically define new discrete-valued attributes that partition the values of continuous-valued attributes into a discrete set of intervals.
  - **Handle missing attribute values.**
    - Assign the most common value of the attribute.
    - Assign probability to each of the possible values.
  - **Attribute construction.**
    - Create new attributes based on existing ones that are sparsely represented.
    - This reduces fragmentation, repetition, and replication.

Decision Tree Induction - Scalability

- **Problem:**
  - Basic decision tree algorithms (ID3, C4.5, and CART) are not scalable (They require the entire dataset to be in memory).
  - They are not designed to handle large datasets.
- **Solution:**
  - Extend the basic algorithms to handle large datasets.
  - We will take a look at **two** modifications/methods:
    1. RainForest
    2. BOAT

Scalability: RainForest

- **Basic Idea:**
  - Extract all data required for the attribute selection methods
    $\Rightarrow$ Store it in a **compact** data structure.
  - Apply the original algorithm to the compact data structure(s).
- **Data structure(s):**
  - **AVC-list:**
    - Attribute
    - Value
    - Class label
  - **AVC-set (of an attribute X):**
    - Aggregated projection of training dataset onto the attribute X with counts of each class label.
  - **AVC-group (of a node n):**
    - Set of AVC-sets of all predictor attributes at the node n.

Scalability: RainForest - AVC-set Example

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
$31 \ldots 40$	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
$31 \ldots 40$	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
$31 \ldots 40$	medium	no	excellent	yes
$31 \ldots 40$	high	yes	fair	yes
> 40	medium	no	excellent	no

Scalability: RainForest - AVC-set Example

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
$31 \ldots 40$	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
$31 \ldots 40$	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
$31 \ldots 40$	medium	no	excellent	yes
$31 \ldots 40$	high	yes	fair	yes
> 40	medium	no	excellent	no

AVC-set on age:

age	yes	no
≤ 30	2	3
$31 \ldots 40$	4	0
> 40	3	2

Scalability: RainForest - AVC-set Example

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
$31 \ldots 40$	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
$31 \ldots 40$	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
$31 \ldots 40$	medium	no	excellent	yes
$31 \ldots 40$	high	yes	fair	yes
> 40	medium	no	excellent	no

AVC-set on age:

age	yes	no
≤ 30	2	3
$31 \ldots 40$	4	0
> 40	3	2

AVC-set on income:

income	yes	no
high	2	2
medium	4	2
low	3	1

Scalability: RainForest - AVC-set Example

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
$31 \ldots 40$	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
$31 \ldots 40$	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
$31 \ldots 40$	medium	no	excellent	yes
$31 \ldots 40$	high	yes	fair	yes
> 40	medium	no	excellent	no

AVC-set on age:

age	yes	no
≤ 30	2	3
$31 \ldots 40$	4	0
> 40	3	2

AVC-set on income:

income	yes	no
high	2	2
medium	4	2
low	3	1

AVC-set on student:

student	yes	no
yes	6	1
no	3	4

Scalability: RainForest - AVC-set Example

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
$31 \ldots 40$	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
$31 \ldots 40$	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
$31 \ldots 40$	medium	no	excellent	yes
$31 \ldots 40$	high	yes	fair	yes
> 40	medium	no	excellent	no

AVC-set on age:

age	yes	no
≤ 30	2	3
$31 \ldots 40$	4	0
> 40	3	2

AVC-set on income:

income	yes	no
high	2	2
medium	4	2
low	3	1

AVC-set on student:

student	yes	no
yes	6	1
no	3	4

AVC-set on credit_rating:

credit_rating	yes	no
fair	6	2
excellent	3	3

Scalability: BOAT

- **Basic Idea:**
  - Use a statistical technique to create several smaller samples (subsets).
  - Every sample fits in the memory and is used to induce a decision tree.
  - All trees are combined to form a single tree T'.
- **Advantages:**
  - Resulting tree often very close to the tree induced from the entire dataset.
  - Requires only two scans of the DB.
  - An incremental algorithm:
    - Take insertions and deletions of training data and update the decision tree.

BOAT

The full title of BOAT is *Bootstrapped Optimistic Algorithm for Tree Construction*, which refers to the underlying statistical technique used: **Bootstrapping**.

Rule-Based Classification

Basic Concepts

- Rule-based classification is based on a set of **IF-THEN rules**.
- Each **IF-THEN rule** consists of two parts:
  - **IF** (antecedent/precondition): a condition or set of conditions that must be satisfied.
  - **THEN** (consequent): the conclusion or action that follows if the IF part is satisfied.

Basic Concepts

- Rule-based classification is based on a set of **IF-THEN rules**.
- Each **IF-THEN rule** consists of two parts:
  - **IF** (antecedent/precondition): a condition or set of conditions that must be satisfied.
  - **THEN** (consequent): the conclusion or action that follows if the IF part is satisfied.
- **Example:** (one rule)
  - **IF** $\texttt{age} \leq 30$ AND $\texttt{student}$ = "yes" **THEN** $\texttt{buys\_computer}$ = "yes".

- Rule-based classification is based on a set of **IF-THEN rules**.
- Each **IF-THEN rule** consists of two parts:
  - **IF** (antecedent/precondition): a condition or set of conditions that must be satisfied.
  - **THEN** (consequent): the conclusion or action that follows if the IF part is satisfied.
- **Example:** (one rule)
  - **IF** $\texttt{age} \leq 30$ AND $\texttt{student}$ = "yes" **THEN** $\texttt{buys\_computer}$ = "yes".
- Very **easy** to read and understand for humans.

Example

- Given is a **set of rules**:
  - **IF** `price` $<$ 1500 **THEN** `buy` = "yes".
  - **IF** `price` $\geq$ 1500 AND `color` = "red" **THEN** `buy` = "no".
  - **IF** `price` $\geq$ 1500 AND `location` = "Erlangen" **THEN** `buy` = "yes".

Example

- Given is a **set of rules**:
  - **IF** `price` $<$ 1500 **THEN** `buy` = "yes".
  - **IF** `price` $\geq$ 1500 AND `color` = "red" **THEN** `buy` = "no".
  - **IF** `price` $\geq$ 1500 AND `location` = "Erlangen" **THEN** `buy` = "yes".

- The set may be used to classify **new tuples**:

price	color	location	buy
1349	red	Nuremberg	?
2306	red	Erlangen	?
1995	green	Fuerth	?

Example

- Given is a **set of rules**:
  - **IF** `price` $<$ 1500 **THEN** `buy` = "yes".
  - **IF** `price` $\geq$ 1500 AND `color` = "red" **THEN** `buy` = "no".
  - **IF** `price` $\geq$ 1500 AND `location` = "Erlangen" **THEN** `buy` = "yes".

- The set may be used to classify **new tuples**:

price	color	location	buy
1349	red	Nuremberg	?
2306	red	Erlangen	?
1995	green	Fuerth	?

Example

- Given is a **set of rules**:
  - **IF** `price` $<$ 1500 **THEN** `buy` = "yes".
  - **IF** `price` $\geq$ 1500 AND `color` = "red" **THEN** `buy` = "no".
  - **IF** `price` $\geq$ 1500 AND `location` = "Erlangen" **THEN** `buy` = "yes".

- The set may be used to classify **new tuples**:

price	color	location	buy
1349	red	Nuremberg	yes
2306	red	Erlangen	?
1995	green	Fuerth	?

Example

- Given is a **set of rules**:
  - **IF** `price` $<$ 1500 **THEN** `buy` = "yes".
  - **IF** `price` $\geq$ 1500 AND `color` = "red" **THEN** `buy` = "no".
  - **IF** `price` $\geq$ 1500 AND `location` = "Erlangen" **THEN** `buy` = "yes".

- The set may be used to classify **new tuples**:

price	color	location	buy
1349	red	Nuremberg	yes
2306	red	Erlangen	?
1995	green	Fuerth	?

Example

- Given is a **set of rules**:
  - **IF** `price` $<$ 1500 **THEN** `buy` = "yes".
  - **IF** `price` $\geq$ 1500 AND `color` = "red" **THEN** `buy` = "no".
  - **IF** `price` $\geq$ 1500 AND `location` = "Erlangen" **THEN** `buy` = "yes".

- The set may be used to classify **new tuples**:

price	color	location	buy
1349	red	Nuremberg	yes
2306	red	Erlangen	?
1995	green	Fuerth	?

- Some scenarios might lead to **conflicts**:
  1. More than one rule is triggered.

Example

- Given is a **set of rules**:
  - **IF** `price` $<$ 1500 **THEN** buy = "yes".
  - **IF** `price` $\geq$ 1500 AND `color` = "red" **THEN** buy = "no".
  - **IF** `price` $\geq$ 1500 AND `location` = "Erlangen" **THEN** buy = "yes".

- The set may be used to classify **new tuples**:

price	color	location	buy
1349	red	Nuremberg	yes
2306	red	Erlangen	?
1995	green	Fuerth	?

- Some scenarios might lead to **conflicts**:
  1. More than one rule is triggered.

Example

- Given is a **set of rules**:
  - **IF** `price` $<$ 1500 **THEN** `buy` = "yes".
  - **IF** `price` $\geq$ 1500 AND `color` = "red" **THEN** `buy` = "no".
  - **IF** `price` $\geq$ 1500 AND `location` = "Erlangen" **THEN** `buy` = "yes".

- The set may be used to classify **new tuples**:

price	color	location	buy
1349	red	Nuremberg	yes
2306	red	Erlangen	?
1995	green	Fuerth	?

- Some scenarios might lead to **conflicts**:
  1. More than one rule is triggered.
  2. No rule is triggered.

Potential Solutions

1. **More than one rule is triggered: conflict resolution.**
    - **Size ordering:**
        - Assign the highest priority to the triggered rule that has the "toughest" requirement (i.e., rule with most used attribute in condition).
    - **Class-based ordering:**
        - Decreasing order of prevalence or misclassification cost per class.
        - No order of rules within class $\rightarrow$ disjunction (logical OR) between rules.
    - **Rule-based ordering** (decision list):
        - Rules are organized into one long priority list, according to some measure of rule quality, or by experts.
        - Rules must be applied in this particular order to avoid conflict.

Potential Solutions

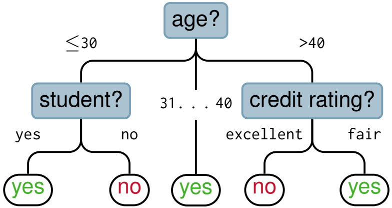
1. **More than one rule is triggered: conflict resolution.**
   - **Size ordering:**
     - Assign the highest priority to the triggered rule that has the "toughest" requirement (i.e., rule with most used attribute in condition).
   - **Class-based ordering:**
     - Decreasing order of prevalence or misclassification cost per class.
     - No order of rules within class $\rightarrow$ disjunction (logical `OR`) between rules.
   - **Rule-based ordering** (decision list):
     - Rules are organized into one long priority list, according to some measure of rule quality, or by experts.
     - Rules must be applied in this particular order to avoid conflict.
2. **No rule is triggered.**
   - Use a fallback/default rule.
   - Always evaluated as the last rule, if and only if other rules are not covered by some tuple, i. e. no rules have been triggered.

Rule Extraction from a Decision Tree

- **Basic idea:**
  - Rules are **easier to understand** than large trees.
  - A rule can be created for **each path from the root to a leaf.**
  - Each attribute-value pair along the path forms **a condition**:

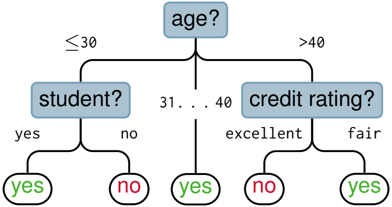
Rule Extraction from a Decision Tree

- **Basic idea:**
  - Rules are **easier to understand** than large trees.
  - A rule can be created for **each path from the root to a leaf.**
  - Each attribute-value pair along the path forms **a condition**:



Rule Extraction from a Decision Tree

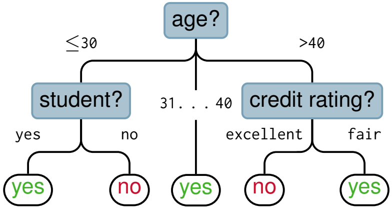
- **Basic idea:**
  - Rules are **easier to understand** than large trees.
  - A rule can be created for **each path from the root to a leaf.**
  - Each attribute-value pair along the path forms **a condition**:



1. **IF** age $\leq$ 30 AND student = "yes"
   THEN buys_computer = "yes".

Rule Extraction from a Decision Tree

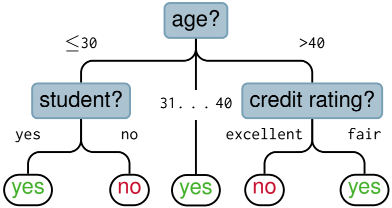
- **Basic idea:**
  - Rules are **easier to understand** than large trees.
  - A rule can be created for **each path from the root to a leaf.**
  - Each attribute-value pair along the path forms **a condition**:



1. **IF** age $\leq$ 30 AND student = "yes"
   THEN buys_computer = "yes".
2. **IF** age $\leq$ 30 AND student = "no"
   THEN buys_computer = "no".

Rule Extraction from a Decision Tree

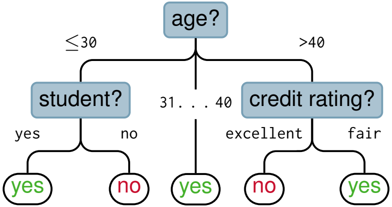
- **Basic idea:**
  - Rules are **easier to understand** than large trees.
  - A rule can be created for **each path from the root to a leaf.**
  - Each attribute-value pair along the path forms **a condition**:



1. **IF** age $\leq$ 30 AND student = "yes"
   THEN buys_computer = "yes".
2. **IF** age $\leq$ 30 AND student = "no"
   THEN buys_computer = "no".
3. **IF** age= 31 . . . 40
   THEN buys_computer = "yes".

Rule Extraction from a Decision Tree

- **Basic idea:**
  - Rules are **easier to understand** than large trees.
  - A rule can be created for **each path from the root to a leaf.**
  - Each attribute-value pair along the path forms **a condition**:



1. **IF** age $\leq$ 30 AND student = "yes"
   THEN buys_computer = "yes".
2. **IF** age $\leq$ 30 AND student = "no"
   THEN buys_computer = "no".
3. **IF** age= 31 . . . 40
   THEN buys_computer = "yes".
4. . . .

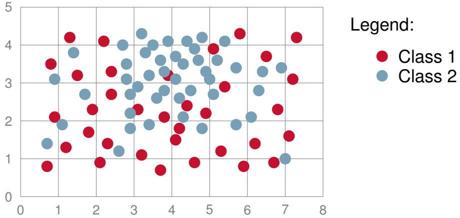
Rule Induction: Sequential Covering Method (I)

- Extracting rules from decision trees is not **the only** way to learn rules.
- Rules can be learned **directly** from **the training data**:
  - Rules are learned **sequentially**.
  - Each rule is optimized to cover **as many tuples of a given class** as possible while covering **as few tuples of other classes** as possible.

Rule Induction: Sequential Covering Method (I)

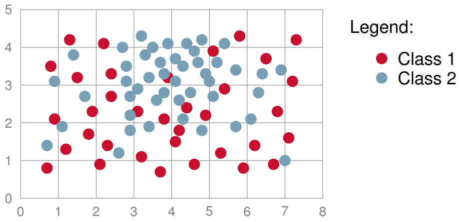
- Extracting rules from decision trees is not **the only** way to learn rules.
- Rules can be learned **directly** from **the training data**:
  - Rules are learned **sequentially**.
  - Each rule is optimized to cover **as many tuples of a given class** as possible while covering **as few tuples of other classes** as possible.

- **Steps of the method:**



- Extracting rules from decision trees is not **the only** way to learn rules.
- Rules can be learned **directly** from **the training data**:
  - Rules are learned **sequentially**.
  - Each rule is optimized to cover **as many tuples of a given class** as possible while covering **as few tuples of other classes** as possible.

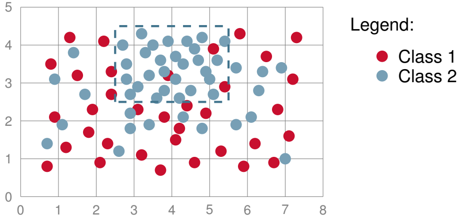
- **Steps of the method:**
  1. Start with an empty set of rules.



Rule Induction: Sequential Covering Method (I)

- Extracting rules from decision trees is not **the only** way to learn rules.
- Rules can be learned **directly** from **the training data**:
  - Rules are learned **sequentially**.
  - Each rule is optimized to cover **as many tuples of a given class** as possible while covering **as few tuples of other classes** as possible.

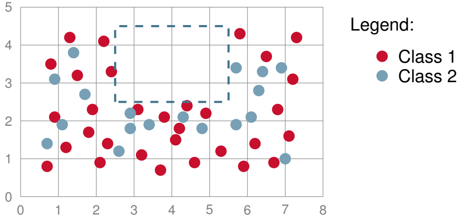
- **Steps of the method:**
  1. Start with an empty set of rules.
  2. Find the rule *r* with the best covering.



Rule Induction: Sequential Covering Method (I)

- Extracting rules from decision trees is not **the only** way to learn rules.
- Rules can be learned **directly** from **the training data**:
  - Rules are learned **sequentially**.
  - Each rule is optimized to cover **as many tuples of a given class** as possible while covering **as few tuples of other classes** as possible.

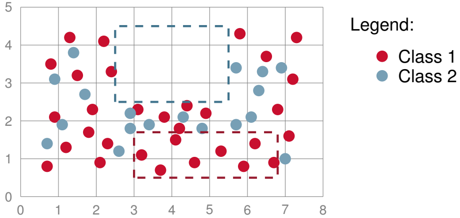
- **Steps of the method:**
  1. Start with an empty set of rules.
  2. Find the rule *r* with the best covering.
  3. Remove all tuples covered.



Rule Induction: Sequential Covering Method (I)

- Extracting rules from decision trees is not **the only** way to learn rules.
- Rules can be learned **directly** from **the training data**:
  - Rules are learned **sequentially**.
  - Each rule is optimized to cover **as many tuples of a given class** as possible while covering **as few tuples of other classes** as possible.

- **Steps of the method:**
  1. Start with an empty set of rules.
  2. Find the rule *r* with the best covering.
  3. Remove all tuples covered.
  4. Repeat with step 2 until:
     - No more tuples left.
     - The quality of a rule is below a threshold.



Rule Induction: Sequential Covering Method (II)

- **Typical sequential covering algorithms:**
  - FOIL[3], AQ[4], CN2[5], RIPPER[6].

[3] J. R. Quinlan, "Learning logical definitions from relations," *Mach. Learn.*, vol. 5, pp. 239–266, 1990. DOI: 10.1007/BF00117105. [Online]. Available: https://doi.org/10.1007/BF00117105
[4] R. S. Michalski et al., "The multi-purpose incremental learning system aq15 and its testing application to three medical domains," in *Proc. AAAI*, vol. 1986, 1986, pp. 1–041
[5] P. Clark and T. Niblett, "The cn2 induction algorithm," *Machine learning*, vol. 3, no. 4, pp. 261–283, 1989
[6] W. W. Cohen et al., "Fast effective rule induction," in *Proceedings of the twelfth international conference on machine learning*, 1995, pp. 115–123

Rule Induction: Sequential Covering Method (II)

- **Typical sequential covering algorithms:**
  - FOIL, AQ, CN2, RIPPER.
- **FOIL (First-Order Inductive Learner):**
  - Based on Information Gain
  - Suppose we have two rules:

$$R : \text{IF condition THEN class} = c$$
$$R' : \text{IF condition' THEN class} = c$$

  - *pos* / *neg* are # of positive/negative tuples covered by R, pos' / neg' respectively for R'.
  - FOIL assesses the information gained by extending *condition'* as

$$\text{FOIL\_Gain} = \text{pos}' \left(\log_2 \frac{\text{pos}'}{\text{pos}' + \text{neg}'} - \log_2 \frac{\text{pos}}{\text{pos} + \text{neg}} \right).$$

  - FOIL favors rules that have high accuracy and cover many positive tuples.

Bayes Classification Methods

Bayesian Classification: Concepts

- **Bayesian Classification:**
  - Statistical classification method.
  - Predict class membership probabilities.
  - Based on **Bayes' Theorem**.

[3] T. Bayes, "An essay towards solving a problem in the doctrine of chances," *Phil. Trans. of the Royal Soc. of London*, vol. 53, pp. 370–418, 1763

Bayesian Classification: Concepts

- **Bayesian Classification:**
  - Statistical classification method.
  - Predict class membership probabilities.
  - Based on **Bayes' Theorem**.

Bayes' Theorem[3]

Bayes' Theorem describes the probability of an event based on prior knowledge of conditions that might be related to the event.

[3] T. Bayes, "An essay towards solving a problem in the doctrine of chances," *Phil. Trans. of the Royal Soc. of London*, vol. 53, pp. 370–418, 1763

Bayesian Classification: Concepts

- **Bayesian Classification:**
  - Statistical classification method.
  - Predict class membership probabilities.
  - Based on **Bayes' Theorem**.
- **Our Focus:**
  - **Naïve Bayesian Classification**
    - Assumes conditional independence of attributes ("naïve").
    - Simplification of Bayesian classification.

Bayes' Theorem[3]

Bayes' Theorem describes the probability of an event based on prior knowledge of conditions that might be related to the event.

[3] T. Bayes, "An essay towards solving a problem in the doctrine of chances," *Phil. Trans. of the Royal Soc. of London*, vol. 53, pp. 370–418, 1763

Bayesian Classification: Basic Terms

- **Let ...**
  - **... X be a data sample ("evidence").**
    - The class label shall be unknown.
  - **... C_i be the hypothesis that X belongs to class i.**

Bayesian Classification: Basic Terms

- **Let ...**
  - **... X be a data sample ("evidence").**
    - The class label shall be unknown.
  - **... C_i be the hypothesis that X belongs to class i.**
- **Then our goal is to determine the Posteriori Probability $P(C_i|X)$:**
  - The probability that the hypothesis holds given the observed data sample X.

Bayesian Classification: Basic Terms

- **Let ...**
  - **... X be a data sample ("evidence").**
    - The class label shall be unknown.
  - **... C_i be the hypothesis that X belongs to class i.**
- **Then our goal is to determine the Posteriori Probability $P(C_i|X)$:**
  - The probability that the hypothesis holds given the observed data sample X.
- **To determine $P(C_i|X)$, we need to know:**
  - **The Prior Probability $P(C_i)$:**
    - The overall probability of the class i.
    - E.g., X will buy computer, regardless of age, income, ...
  - **The Likelihood $P(X|C_j)$:**
    - The probability of observing the sample X given that the hypothesis holds.
    - E.g., given that X buys computer, the probability that X is 31 . . . 40, medium income.
  - **The Probability of the sample data $P(X)$:**
    - The probability that sample data is observed.

Bayesian Classification: Posteriori Probability

- **The Posteriori Probability $P(C_i|X)$ follows from the Bayes' Theorem:**

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}.$$

Bayesian Classification: Posteriori Probability

- **The Posteriori Probability $P(C_i|X)$ follows from the Bayes' Theorem:**

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}.$$

The Maximum Posteriori Probability

Since the posteriori probability $P(C_i|X)$ is the basically the probability that X belongs to class C_i, we want to find the class C_i that **maximizes** this probability. X **is classified** as belonging to this class.

- **The Posteriori Probability $P(C_i|X)$ follows from the Bayes' Theorem:**

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}.$$

The Maximum Posteriori Probability

Since the posteriori probability $P(C_i|X)$ is the basically the probability that X belongs to class C_i, we want to find the class C_i that **maximizes** this probability. X **is classified** as belonging to this class.

- **Since $P(X)$ is constant for all classes, we only have to maximize:**

$$P(X|C_i)P(C_i).$$

- **Naïve Bayesian Classification:**
  - **Assumption:** All attributes are *conditionally independent*.
  - I.e. no dependence relation between attributes (which is "naïve").

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i)P(x_2|C_i) \cdots P(x_n|C_i).$$

  - Greatly reduces the computation cost.

Bayesian Classification: Likelihood

- Naïve **Bayesian Classification:**
  - **Assumption:** All attributes are *conditionally independent*.
  - I.e. no dependence relation between attributes (which is "naïve").

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i)P(x_2|C_i)\cdots P(x_n|C_i).$$

  - Greatly reduces the computation cost.

Bayesian Classification: Likelihood

- Naïve **Bayesian Classification:**
  - **Assumption:** All attributes are *conditionally independent*.
  - I.e. no dependence relation between attributes (which is "naïve").

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i)P(x_2|C_i)\cdots P(x_n|C_i).$$

  - Greatly reduces the computation cost.

Categorical Attribute

- $P(x_k|C_i)$ is the number of tuples in C_i having value x_k for A_k divided by $|C_{i,D}|$ (the number of tuples of C_i in D):

$$P(x_k|C_i) = \frac{|\{t \in C_i : t.A_k = x_k\}|}{|C_{i,D}|}$$

Bayesian Classification: Likelihood

- Naïve **Bayesian Classification:**
  - **Assumption:** All attributes are *conditionally independent*.
  - I.e. no dependence relation between attributes (which is "naïve").

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i)P(x_2|C_i)\cdots P(x_n|C_i).$$

  - Greatly reduces the computation cost.

Categorical Attribute

- $P(x_k|C_i)$ is the number of tuples in C_i having value x_k for A_k divided by $|C_{i,D}|$ (the number of tuples of C_i in D):

$$P(x_k|C_i) = \frac{|\{t \in C_i : t.A_k = x_k\}|}{|C_{i,D}|}$$

Continuous-valued Attribute

- $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ:

$$G(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$P(x_k|C_i) = G(x_k, \mu_{C_i}, \sigma_{C_i})$$

Bayesian Classification: Example (I)

- **Target attribute:** buys_computer
- **Data sample:**

$$X = (\text{age} \leq 30, \quad \text{income} = \text{"}\textit{medium}\text{"}, \quad \text{student} = \text{"}\textit{yes}\text{"}, \quad \text{credit\_rating} = \text{"}\textit{fair}\text{"})$$

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
$31 \ldots 40$	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
$31 \ldots 40$	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
$31 \ldots 40$	medium	no	excellent	yes
$31 \ldots 40$	high	yes	fair	yes
> 40	medium	no	excellent	no

Bayesian Classification: Example (I)

- **Target attribute:** buys_computer
- **Data sample:**

$$X = (\texttt{age} \leq 30,\ \texttt{income} = ''\textit{medium}'',\ \texttt{student} = ''\textit{yes}'',\ \texttt{credit\_rating} = ''\textit{fair}'')$$

- **Prior Probability $P(C_i)$:**

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
$31 \ldots 40$	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
$31 \ldots 40$	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
$31 \ldots 40$	medium	no	excellent	yes
$31 \ldots 40$	high	yes	fair	yes
> 40	medium	no	excellent	no

Bayesian Classification: Example (I)

- **Target attribute:** buys_computer
- **Data sample:**

$$X = (\text{age} \leq 30, \quad \texttt{income} = \text{"}\textit{medium}\text{"}, \quad \texttt{student} = \text{"}\textit{yes}\text{"}, \quad \texttt{credit\_rating} = \text{"}\textit{fair}\text{"})$$

- **Prior Probability $P(C_i)$:**

$$P(yes) =$$

$$P(no) =$$

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
31 . . . 40	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
31 . . . 40	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
31 . . . 40	medium	no	excellent	yes
31 . . . 40	high	yes	fair	yes
> 40	medium	no	excellent	no

Bayesian Classification: Example (I)

- **Target attribute:** buys_computer
- **Data sample:**

$$X = (\text{age} \leq 30, \\ \texttt{income} = "medium", \\ \texttt{student} = "yes", \\ \texttt{credit\_rating} = "fair")$$

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
$31 \ldots 40$	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
$31 \ldots 40$	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
$31 \ldots 40$	medium	no	excellent	yes
$31 \ldots 40$	high	yes	fair	yes
> 40	medium	no	excellent	no

- **Prior Probability $P(C_i)$:**

$$P(yes) = \frac{9}{14} \approx 0.643$$

$$P(no) =$$

Bayesian Classification: Example (I)

- **Target attribute:** buys_computer
- **Data sample:**

$$X = (\text{age} \leq 30, \quad \texttt{income} = \text{"}medium\text{"}, \quad \texttt{student} = \text{"}yes\text{"}, \quad \texttt{credit\_rating} = \text{"}fair\text{"})$$

- **Prior Probability $P(C_i)$:**

$$P(yes) = \frac{9}{14} \approx 0.643$$

$$P(no) = \frac{5}{14} \approx 0.357$$

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
$31 \ldots 40$	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
$31 \ldots 40$	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
$31 \ldots 40$	medium	no	excellent	yes
$31 \ldots 40$	high	yes	fair	yes
> 40	medium	no	excellent	no

Bayesian Classification: Example (II)

Friedrich-Alexander-Universität
Technische Fakultät

- **Likelihood(s)** $P(X_k | C_i)$**:**

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
31 . . . 40	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
31 . . . 40	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
31 . . . 40	medium	no	excellent	yes
31 . . . 40	high	yes	fair	yes
> 40	medium	no	excellent	no

Bayesian Classification: Example (II)

- **Likelihood(s)** $P(X_k|C_i)$:

$$P(\texttt{age} \leq 30|yes) =$$

$$P(\texttt{age} \leq 30|no) =$$

$$P(\texttt{income} = "medium"|yes) =$$

$$P(\texttt{income} = "medium"|no) =$$

$$P(\texttt{student} = "yes"|yes) =$$

$$P(\texttt{student} = "yes"|no) =$$

$$P(\texttt{credit\_rating} = "fair"|yes) =$$

$$P(\texttt{credit\_rating} = "fair"|no) =$$

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
31 . . . 40	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
31 . . . 40	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
31 . . . 40	medium	no	excellent	yes
31 . . . 40	high	yes	fair	yes
> 40	medium	no	excellent	no

Bayesian Classification: Example (II)

Friedrich-Alexander-Universität
Technische Fakultät

- **Likelihood(s)** $P(X_k|C_i)$**:**

$$P(\text{age} \leq 30|yes) = \frac{2}{9} \approx 0.222$$

$$P(\text{age} \leq 30|no) =$$

$$P(\texttt{income} = "medium" |yes) =$$

$$P(\texttt{income} = "medium" |no) =$$

$$P(\texttt{student} = "yes" |yes) =$$

$$P(\texttt{student} = "yes" |no) =$$

$$P(\texttt{credit\_rating} = "fair" |yes) =$$

$$P(\texttt{credit\_rating} = "fair" |no) =$$

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
$31\ldots40$	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
$31\ldots40$	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
$31\ldots40$	medium	no	excellent	yes
$31\ldots40$	high	yes	fair	yes
> 40	medium	no	excellent	no

Bayesian Classification: Example (II)

- **Likelihood(s) $P(X_k|C_i)$:**

$$P(\text{age} \leq 30|yes) = \frac{2}{9} \approx 0.222$$

$$P(\text{age} \leq 30|no) = \frac{3}{5} = 0.6$$

$$P(\texttt{income} = \text{"}medium\text{"}|yes) =$$

$$P(\texttt{income} = \text{"}medium\text{"}|no) =$$

$$P(\texttt{student} = \text{"}yes\text{"}|yes) =$$

$$P(\texttt{student} = \text{"}yes\text{"}|no) =$$

$$P(\texttt{credit\_rating} = \text{"}fair\text{"}|yes) =$$

$$P(\texttt{credit\_rating} = \text{"}fair\text{"}|no) =$$

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
$31 \ldots 40$	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
$31 \ldots 40$	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
$31 \ldots 40$	medium	no	excellent	yes
$31 \ldots 40$	high	yes	fair	yes
> 40	medium	no	excellent	no

Bayesian Classification: Example (II)

Friedrich-Alexander-Universität
Technische Fakultät

- **Likelihood(s)** $P(X_k|C_i)$**:**

$$P(\texttt{age} \leq 30|yes) = \frac{2}{9} \approx 0.222$$

$$P(\texttt{age} \leq 30|no) = \frac{3}{5} = 0.6$$

$$P(\texttt{income} = \text{"}medium\text{"}|yes) = \frac{4}{9} \approx 0.444$$

$$P(\texttt{income} = \text{"}medium\text{"}|no) =$$

$$P(\texttt{student} = \text{"}yes\text{"}|yes) =$$

$$P(\texttt{student} = \text{"}yes\text{"}|no) =$$

$$P(\texttt{credit\_rating} = \text{"}fair\text{"}|yes) =$$

$$P(\texttt{credit\_rating} = \text{"}fair\text{"}|no) =$$

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
31 ... 40	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
31 ... 40	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
31 ... 40	medium	no	excellent	yes
31 ... 40	high	yes	fair	yes
> 40	medium	no	excellent	no

Bayesian Classification: Example (II)

- **Likelihood(s)** $P(X_k|C_i)$:

$$P(\text{age} \leq 30|yes) = \frac{2}{9} \approx 0.222$$

$$P(\text{age} \leq 30|no) = \frac{3}{5} = 0.6$$

$$P(\texttt{income} = \text{"}medium\text{"}|yes) = \frac{4}{9} \approx 0.444$$

$$P(\texttt{income} = \text{"}medium\text{"}|no) = \frac{2}{5} = 0.4$$

$$P(\texttt{student} = \text{"}yes\text{"}|yes) =$$

$$P(\texttt{student} = \text{"}yes\text{"}|no) =$$

$$P(\texttt{credit\_rating} = \text{"}fair\text{"}|yes) =$$

$$P(\texttt{credit\_rating} = \text{"}fair\text{"}|no) =$$

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
$31 \ldots 40$	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
$31 \ldots 40$	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
$31 \ldots 40$	medium	no	excellent	yes
$31 \ldots 40$	high	yes	fair	yes
> 40	medium	no	excellent	no

Bayesian Classification: Example (II)

- **Likelihood(s)** $P(X_k|C_i)$:

$$P(\texttt{age} \leq 30|yes) = \frac{2}{9} \approx 0.222$$

$$P(\texttt{age} \leq 30|no) = \frac{3}{5} = 0.6$$

$$P(\texttt{income} = \text{"}medium\text{"}|yes) = \frac{4}{9} \approx 0.444$$

$$P(\texttt{income} = \text{"}medium\text{"}|no) = \frac{2}{5} = 0.4$$

$$P(\texttt{student} = \text{"}yes\text{"}|yes) = \frac{5}{9} \approx 0.556$$

$$P(\texttt{student} = \text{"}yes\text{"}|no) = \frac{1}{5} = 0.2$$

$$P(\texttt{credit\_rating} = \text{"}fair\text{"}|yes) = \frac{6}{9} \approx 0.667$$

$$P(\texttt{credit\_rating} = \text{"}fair\text{"}|no) = \frac{2}{5} = 0.4$$

age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
31 . . . 40	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
31 . . . 40	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
31 . . . 40	medium	no	excellent	yes
31 . . . 40	high	yes	fair	yes
> 40	medium	no	excellent	no

Bayesian Classification: Example (II)

- **Likelihood(s)** $P(X_k|C_i)$**:**

$$P(\text{age} \leq 30|\textit{yes}) = \frac{2}{9} \approx 0.222$$

$$P(\text{age} \leq 30|\textit{no}) = \frac{3}{5} = 0.6$$

$$P(\texttt{income} = \text{"}\textit{medium}\text{"}|\textit{yes}) = \frac{4}{9} \approx 0.444$$

$$P(\texttt{income} = \text{"}\textit{medium}\text{"}|\textit{no}) = \frac{2}{5} = 0.4$$

$$P(\texttt{student} = \text{"}\textit{yes}\text{"}|\textit{yes}) = \frac{5}{9} \approx 0.556$$

$$P(\texttt{student} = \text{"}\textit{yes}\text{"}|\textit{no}) = \frac{1}{5} = 0.2$$

$$P(\texttt{credit\_rating} = \text{"}\textit{fair}\text{"}|\textit{yes}) = \frac{6}{9} \approx 0.667$$

$$P(\texttt{credit\_rating} = \text{"}\textit{fair}\text{"}|\textit{no}) = \frac{2}{5} = 0.4$$

- **Likelihood(s)** $P(X|C_i)$**:**

$$P(X|\textit{yes}) =$$

$$P(X|\textit{no}) =$$

Bayesian Classification: Example (II)

- **Likelihood(s) $P(X_k|C_i)$:**

$$P(\texttt{age} \leq 30|yes) = \frac{2}{9} \approx 0.222$$

$$P(\texttt{age} \leq 30|no) = \frac{3}{5} = 0.6$$

$$P(\texttt{income} = "medium"|yes) = \frac{4}{9} \approx 0.444$$

$$P(\texttt{income} = "medium"|no) = \frac{2}{5} = 0.4$$

$$P(\texttt{student} = "yes"|yes) = \frac{5}{9} \approx 0.556$$

$$P(\texttt{student} = "yes"|no) = \frac{1}{5} = 0.2$$

$$P(\texttt{credit\_rating} = "fair"|yes) = \frac{6}{9} \approx 0.667$$

$$P(\texttt{credit\_rating} = "fair"|no) = \frac{2}{5} = 0.4$$

- **Likelihood(s) $P(X|C_i)$:**

$$P(X|yes) = 0.222 \cdot 0.444 \cdot 0.556 \cdot 0.667 \approx 0.037$$

$$P(X|no) =$$

Bayesian Classification: Example (II)

- **Likelihood(s)** $P(X_k|C_i)$**:**

$$P(\text{age} \leq 30|yes) = \frac{2}{9} \approx 0.222$$

$$P(\text{age} \leq 30|no) = \frac{3}{5} = 0.6$$

$$P(\texttt{income} = \text{"}medium\text{"}|yes) = \frac{4}{9} \approx 0.444$$

$$P(\texttt{income} = \text{"}medium\text{"}|no) = \frac{2}{5} = 0.4$$

$$P(\texttt{student} = \text{"}yes\text{"}|yes) = \frac{5}{9} \approx 0.556$$

$$P(\texttt{student} = \text{"}yes\text{"}|no) = \frac{1}{5} = 0.2$$

$$P(\texttt{credit\_rating} = \text{"}fair\text{"}|yes) = \frac{6}{9} \approx 0.667$$

$$P(\texttt{credit\_rating} = \text{"}fair\text{"}|no) = \frac{2}{5} = 0.4$$

- **Likelihood(s)** $P(X|C_i)$**:**

$$P(X|yes) = 0.222 \cdot 0.444 \cdot 0.556 \cdot 0.667 \approx 0.037$$

$$P(X|no) = 0.6 \cdot 0.4 \cdot 0.2 \cdot 0.4 = 0.019$$

Bayesian Classification: Example (II)

- **Likelihood(s) $P(X_k|C_i)$:**

$$P(\texttt{age} \leq 30|yes) = \frac{2}{9} \approx 0.222$$

$$P(\texttt{age} \leq 30|no) = \frac{3}{5} = 0.6$$

$$P(\texttt{income} = "medium"|yes) = \frac{4}{9} \approx 0.444$$

$$P(\texttt{income} = "medium"|no) = \frac{2}{5} = 0.4$$

$$P(\texttt{student} = "yes"|yes) = \frac{5}{9} \approx 0.556$$

$$P(\texttt{student} = "yes"|no) = \frac{1}{5} = 0.2$$

$$P(\texttt{credit\_rating} = "fair"|yes) = \frac{6}{9} \approx 0.667$$

$$P(\texttt{credit\_rating} = "fair"|no) = \frac{2}{5} = 0.4$$

- **Likelihood(s) $P(X|C_i)$:**

$$P(X|yes) = 0.222 \cdot 0.444 \cdot 0.556 \cdot 0.667 \approx 0.037$$

$$P(X|no) = 0.6 \cdot 0.4 \cdot 0.2 \cdot 0.4 = 0.019$$

- **Calculate $P(X|C_i) \cdot P(C_i)$[a]:**

[a] **Reminder:** Calculating the numerator of the posterior probability is sufficient to classify X

- **Likelihood(s) $P(X_k|C_i)$:**

$$P(\texttt{age} \leq 30|yes) = \frac{2}{9} \approx 0.222$$

$$P(\texttt{age} \leq 30|no) = \frac{3}{5} = 0.6$$

$$P(\texttt{income} = \text{"}medium\text{"}|yes) = \frac{4}{9} \approx 0.444$$

$$P(\texttt{income} = \text{"}medium\text{"}|no) = \frac{2}{5} = 0.4$$

$$P(\texttt{student} = \text{"}yes\text{"}|yes) = \frac{5}{9} \approx 0.556$$

$$P(\texttt{student} = \text{"}yes\text{"}|no) = \frac{1}{5} = 0.2$$

$$P(\texttt{credit\_rating} = \text{"}fair\text{"}|yes) = \frac{6}{9} \approx 0.667$$

$$P(\texttt{credit\_rating} = \text{"}fair\text{"}|no) = \frac{2}{5} = 0.4$$

- **Likelihood(s) $P(X|C_i)$:**

$$P(X|yes) = 0.222 \cdot 0.444 \cdot 0.556 \cdot 0.667 \approx 0.037$$

$$P(X|no) = 0.6 \cdot 0.4 \cdot 0.2 \cdot 0.4 = 0.019$$

- **Calculate $P(X|C_i) \cdot P(C_i)$[a]:**

$$P(X|yes) \cdot P(yes) = 0.024.$$

$$P(X|no) \cdot P(no) = 0.007.$$

[a] **Reminder:** Calculating the numerator of the posterior probability is sufficient to classify X

Bayesian Classification: Example (II)

- **Likelihood(s) $P(X_k|C_i)$:**

$$P(\text{age} \leq 30|yes) = \frac{2}{9} \approx 0.222$$
$$P(\text{age} \leq 30|no) = \frac{3}{5} = 0.6$$
$$P(\texttt{income} = "medium"|yes) = \frac{4}{9} \approx 0.444$$
$$P(\texttt{income} = "medium"|no) = \frac{2}{5} = 0.4$$
$$P(\texttt{student} = "yes"|yes) = \frac{5}{9} \approx 0.556$$
$$P(\texttt{student} = "yes"|no) = \frac{1}{5} = 0.2$$
$$P(\texttt{credit\_rating} = "fair"|yes) = \frac{6}{9} \approx 0.667$$
$$P(\texttt{credit\_rating} = "fair"|no) = \frac{2}{5} = 0.4$$

- **Likelihood(s) $P(X|C_i)$:**

$$P(X|yes) = 0.222 \cdot 0.444 \cdot 0.556 \cdot 0.667 \approx 0.037$$
$$P(X|no) = 0.6 \cdot 0.4 \cdot 0.2 \cdot 0.4 = 0.019$$

- **Calculate $P(X|C_i) \cdot P(C_i)^a$:**

$$P(X|yes) \cdot P(yes) = 0.024.$$
$$P(X|no) \cdot P(no) = 0.007.$$

- **Classification Result:**

[a] **Reminder:** Calculating the numerator of the posterior probability is sufficient to classify X

Bayesian Classification: Example (II)

- **Likelihood(s) $P(X_k|C_i)$:**

$$P(\text{age} \leq 30|yes) = \frac{2}{9} \approx 0.222$$
$$P(\text{age} \leq 30|no) = \frac{3}{5} = 0.6$$
$$P(\texttt{income} = \text{"}medium\text{"}|yes) = \frac{4}{9} \approx 0.444$$
$$P(\texttt{income} = \text{"}medium\text{"}|no) = \frac{2}{5} = 0.4$$
$$P(\texttt{student} = \text{"}yes\text{"}|yes) = \frac{5}{9} \approx 0.556$$
$$P(\texttt{student} = \text{"}yes\text{"}|no) = \frac{1}{5} = 0.2$$
$$P(\texttt{credit\_rating} = \text{"}fair\text{"}|yes) = \frac{6}{9} \approx 0.667$$
$$P(\texttt{credit\_rating} = \text{"}fair\text{"}|no) = \frac{2}{5} = 0.4$$

- **Likelihood(s) $P(X|C_i)$:**

$$P(X|yes) = 0.222 \cdot 0.444 \cdot 0.556 \cdot 0.667 \approx 0.037$$
$$P(X|no) = 0.6 \cdot 0.4 \cdot 0.2 \cdot 0.4 = 0.019$$

- **Calculate $P(X|C_i) \cdot P(C_i)^a$:**

$$P(X|yes) \cdot P(yes) = 0.024.$$
$$P(X|no) \cdot P(no) = 0.007.$$

- **Classification Result:**
  - X belongs to class C_1 (buys_computer = yes)

[a] **Reminder:** Calculating the numerator of the posterior probability is sufficient to classify X

Bayesian Classification: Problem

Zero-Probability Problem

Our naïve Bayes classifier performs poorly if any of the conditional probabilities **any** of the conditional probabilities $P(x_k | C_i)$ is zero, as this causes the **entire product** of probabilities to become zero.

Bayesian Classification: Problem

Zero-Probability Problem

Our naïve Bayes classifier performs poorly if any of the conditional probabilities **any** of the conditional probabilities $P(x_k | C_i)$ is zero, as this causes the **entire product** of probabilities to become zero.

- **Example:**
  - `income` = "low" (0 tuples), "medium" (990 tuples), "high" (10 tuples).

Bayesian Classification: Problem

Zero-Probability Problem

Our naïve Bayes classifier performs poorly if any of the conditional probabilities **any** of the conditional probabilities $P(x_k|C_i)$ is zero, as this causes the **entire product** of probabilities to become zero.

- **Example:**
  - `income` = "low" (0 tuples), "medium" (990 tuples), "high" (10 tuples).
- **Solution:**
  - **Use Laplacian correction (or Laplacian estimator):**
    - Add 1 to each case:

$$P(\texttt{income} = \text{"}low\text{"}) = \frac{1}{1003}$$
$$P(\texttt{income} = \text{"}medium\text{"}) = \frac{991}{1003}$$
$$P(\texttt{income} = \text{"}high\text{"}) = \frac{11}{1003}$$

Bayesian Classification: Summary

- **Advantages of the Naïve Bayes Classifier:**
  - Easy to implement.
  - Good results obtained in most of the cases.
- **Disadvantages of the Naïve Bayes Classifier:**
  - Assumption: class conditional independence, therefore loss of accuracy.
  - Practically, **dependencies** exist among most variables.
    - Cannot be modeled by Naïve Bayesian Classifier.
    - **How to deal with these dependencies?**
      $\longrightarrow$ Bayesian Belief Networks (not part of KDD[4]).

[4] **More info in the reference book (Chapter 9.1):** J. Han et al., *Data Mining: Concepts and Techniques*, 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011, ISBN: 0123814790

Model Evaluation

Model Evaluation

- **Classification models might perform differently depending on the use case.**
  $\Rightarrow$ **Model evaluation** is crucial to select the best model for a specific task.
- **Model evaluation can be split into two parts:**
  - **Evaluation metrics:**
    What metric is important for the task?
  - **Evaluation strategies:**
    How to tackle the evaluation? E.g. how to split the data into training and test sets?

Model Evaluation

Evaluation Metrics

Confusion Matrix

		Predicted Class		
		C_1	$\neg C_1$	*Total*
True Class	C_1	**TP**	**FN**	**P**
	$\neg C_1$	**FP**	**TN**	**N**
	Total	**P'**	**N'**	**P+N**

- **Confusion Matrix:**
  - Summarizes the results of a classification model.
  - Shows the number of correct and incorrect predictions for each class.
  - **Correctly classified tuples:**
    - **True Positives (TP)**: Positive tuples correctly classified as positive.
    - **True Negatives (TN)**: Negative tuples correctly classified as negative.
  - **Incorrectly classified tuples:**
    - **False Positives (FP)**: Negative tuples incorrectly classified as positive.
    - **False Negatives (FN)**: Positive tuples incorrectly classified as negative.

Evaluation Metrics: Accuracy and Error Rate

		Predicted Class		
		C_1	$\neg C_1$	*Total*
True Class	C_1	**TP**	**FN**	**P**
	$\neg C_1$	**FP**	**TN**	**N**
	Total	**P'**	**N'**	**P+N**

- **Accuracy:**
- **Error Rate:**

Evaluation Metrics: Accuracy and Error Rate

		Predicted Class		
		C_1	$\neg C_1$	*Total*
True Class	C_1	**TP**	**FN**	**P**
	$\neg C_1$	**FP**	**TN**	**N**
	Total	**P'**	**N'**	**P+N**

- **Accuracy:**
  - Percentage of correctly classified tuples.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}}$$

- **Error Rate:**

Evaluation Metrics: Accuracy and Error Rate

Friedrich-Alexander-Universität
Technische Fakultät

		Predicted Class		
		C_1	$\neg C_1$	*Total*
True Class	C_1	**TP**	**FN**	**P**
	$\neg C_1$	**FP**	**TN**	**N**
	Total	**P'**	**N'**	**P+N**

- **Accuracy:**
  - Percentage of correctly classified tuples.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}}$$

- **Error Rate:**
  - Inverse of accuracy, i.e. percentage of incorrectly classified tuples.

$$\text{Error Rate} = 1 - \text{Accuracy} = \frac{\text{FP} + \text{FN}}{\text{P} + \text{N}}$$

Predicted Class

		C_1	$\neg C_1$	*Total*
True Class	C_1	**TP**	**FN**	**P**
	$\neg C_1$	**FP**	**TN**	**N**
	Total	**P'**	**N'**	**P+N**

- **Sensitivity:**
- **Specificity:**

		Predicted Class		
		C_1	$\neg C_1$	*Total*
True Class	C_1	**TP**	**FN**	**P**
	$\neg C_1$	**FP**	**TN**	**N**
	Total	**P'**	**N'**	**P+N**

- **Sensitivity:**
  - True positive rate.

$$\text{Sensitivity} = \frac{\text{TP}}{\text{P}}$$

- **Specificity:**

Evaluation Metrics: Sensitivity and Specificity

		Predicted Class		
		C_1	$\neg C_1$	*Total*
True Class	C_1	**TP**	**FN**	**P**
	$\neg C_1$	**FP**	**TN**	**N**
	Total	**P'**	**N'**	**P+N**

- **Sensitivity:**
  - True positive rate.

$$\text{Sensitivity} = \frac{\text{TP}}{\text{P}}$$

- **Specificity:**
  - True negative rate.

$$\text{Specificity} = \frac{\text{TN}}{\text{N}}$$

Evaluation Metrics: Precision and Recall

Friedrich-Alexander-Universität
Technische Fakultät

		Predicted Class		
		C_1	$\neg C_1$	*Total*
True Class	C_1	**TP**	**FN**	**P**
	$\neg C_1$	**FP**	**TN**	**N**
	Total	**P'**	**N'**	**P+N**

- **Precision:**
- **Recall:**

Evaluation Metrics: Precision and Recall

		Predicted Class		
		C_1	$\neg C_1$	*Total*
True Class	C_1	**TP**	**FN**	**P**
	$\neg C_1$	**FP**	**TN**	**N**
	Total	**P'**	**N'**	**P+N**

- **Precision:**
  - Measure of exactness.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **Recall:**

Evaluation Metrics: Precision and Recall

		Predicted Class		
		C_1	$\neg C_1$	*Total*
True Class	C_1	**TP**	**FN**	**P**
	$\neg C_1$	**FP**	**TN**	**N**
	Total	**P'**	**N'**	**P+N**

- **Precision:**
  - Measure of exactness.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **Recall:**
  - Measure of completeness.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Evaluation Metrics: Precision and Recall

		Predicted Class		
		C_1	$\neg C_1$	*Total*
True Class	C_1	**TP**	**FN**	**P**
	$\neg C_1$	**FP**	**TN**	**N**
	Total	**P'**	**N'**	**P+N**

- **Precision:**
  - Measure of exactness.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **Recall:**
  - Measure of completeness.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \text{Sensitivity}$$

Evaluation Metrics: F_β and F_1 Measure

Predicted Class

True Class		C_1	$\neg C_1$	*Total*
	C_1	**TP**	**FN**	**P**
	$\neg C_1$	**FP**	**TN**	**N**
	Total	**P'**	**N'**	**P+N**

- **F_β Measure:**
- **F_1 Measure:**

Evaluation Metrics: F_β and F_1 Measure

Predicted Class

		C_1	$\neg C_1$	*Total*
True Class	C_1	**TP**	**FN**	**P**
	$\neg C_1$	**FP**	**TN**	**N**
	Total	**P'**	**N'**	**P+N**

- **F_β Measure:**
  - Combining precision and recall.
  - Gives β-times more weight to precision.
  - $\beta > 1$: Minimize false positives.
  - $\beta < 1$: Minimize false negatives.

$$F_\beta = \frac{(1+\beta^2) \times \text{Precision} \times \text{Recall}}{\beta^2 \times \text{Precision} + \text{Recall}}$$

- **F_1 Measure:**

Evaluation Metrics: F_β and F_1 Measure

Friedrich-Alexander-Universität
Technische Fakultät

		Predicted Class		
		C_1	$\neg C_1$	*Total*
True Class	C_1	**TP**	**FN**	**P**
	$\neg C_1$	**FP**	**TN**	**N**
	Total	**P'**	**N'**	**P+N**

- **F_β Measure:**
  - Combining precision and recall.
  - Gives β-times more weight to precision.
  - $\beta > 1$: Minimize false positives.
  - $\beta < 1$: Minimize false negatives.

$$F_\beta = \frac{(1+\beta^2) \times \text{Precision} \times \text{Recall}}{\beta^2 \times \text{Precision} + \text{Recall}}$$

- **F_1 Measure:**
  - Harmonic mean between the measures.
  - Equal weight to both measures.

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Evaluation Metrics: Example (I)

- **Evaluation results of a classification model:**

True Class	Cat	Dog	Fox	Cat	Cat	Hen	Cat	Cat	Dog	Fox
Predicted Class	Cat	Dog	Cat	Fox	Cat	Dog	Cat	Dog	Fox	Fox

Evaluation Metrics: Example (I)

- **Evaluation results of a classification model:**

True Class	Cat	Dog	Fox	Cat	Cat	Hen	Cat	Cat	Dog	Fox
Predicted Class	Cat	Dog	Cat	Fox	Cat	Dog	Cat	Dog	Fox	Fox

- **Resulting confusion matrix[5]:**

		Predicted Class Cat	¬Cat	*Total*
True Class	Cat	**TP**	**FN**	**P**
	¬Cat	**FP**	**TN**	**N**
	Total	**P'**	**N'**	**P+N**

[5] We want to evaluate the classification model with regard to the class *Cat*

Evaluation Metrics: Example (I)

- **Evaluation results of a classification model:**

True Class	Cat	Dog	Fox	Cat	Cat	Hen	Cat	Cat	Dog	Fox
Predicted Class	Cat	Dog	Cat	Fox	Cat	Dog	Cat	Dog	Fox	Fox

- **Resulting confusion matrix[5]:**

		Predicted Class Cat	¬Cat	*Total*
True Class	Cat	**3**	**FN**	**P**
	¬Cat	**FP**	**TN**	**N**
	Total	**P'**	**N'**	**P+N**

[5] We want to evaluate the classification model with regard to the class *Cat*

Evaluation Metrics: Example (I)

- **Evaluation results of a classification model:**

True Class	Cat	Dog	Fox	Cat	Cat	Hen	Cat	Cat	Dog	Fox
Predicted Class	Cat	Dog	Cat	Fox	Cat	Dog	Cat	Dog	Fox	Fox

- **Resulting confusion matrix[5]:**

		Predicted Class Cat	*Predicted Class* ¬Cat	*Total*
True Class	Cat	**3**	**FN**	**P**
True Class	¬Cat	**FP**	**TN**	**N**
	Total	**P'**	**N'**	**P+N**

[5] We want to evaluate the classification model with regard to the class *Cat*

- **Evaluation results of a classification model:**

True Class	Cat	Dog	Fox	Cat	Cat	Hen	Cat	Cat	Dog	Fox
Predicted Class	Cat	Dog	Cat	Fox	Cat	Dog	Cat	Dog	Fox	Fox

- **Resulting confusion matrix[5]:**

		Predicted Class		
		Cat	¬Cat	*Total*
True Class	Cat	**3**	**2**	**5**
	¬Cat	**FP**	**TN**	**N**
	Total	**P'**	**N'**	**P+N**

[5] We want to evaluate the classification model with regard to the class *Cat*

Evaluation Metrics: Example (I)

- **Evaluation results of a classification model:**

True Class	Cat	Dog	Fox	Cat	Cat	Hen	Cat	Cat	Dog	Fox
Predicted Class	Cat	Dog	Cat	Fox	Cat	Dog	Cat	Dog	Fox	Fox

- **Resulting confusion matrix[5]:**

		Predicted Class Cat	*Predicted Class* ¬Cat	*Total*
True Class	Cat	**3**	**2**	**5**
True Class	¬Cat	**FP**	**TN**	**N**
	Total	**P'**	**N'**	**P+N**

[5] We want to evaluate the classification model with regard to the class *Cat*

Evaluation Metrics: Example (I)

- **Evaluation results of a classification model:**

True Class	Cat	Dog	Fox	Cat	Cat	Hen	Cat	Cat	Dog	Fox
Predicted Class	Cat	Dog	Cat	Fox	Cat	Dog	Cat	Dog	Fox	Fox

- **Resulting confusion matrix[5]:**

		Predicted Class		
		Cat	¬Cat	*Total*
True Class	Cat	**3**	**2**	**5**
	¬Cat	**1**	**TN**	**N**
	Total	**4**	**N'**	**P+N**

[5] We want to evaluate the classification model with regard to the class *Cat*

Evaluation Metrics: Example (I)

- **Evaluation results of a classification model:**

True Class	Cat	Dog	Fox	Cat	Cat	Hen	Cat	Cat	Dog	Fox
Predicted Class	Cat	Dog	Cat	Fox	Cat	Dog	Cat	Dog	Fox	Fox

- **Resulting confusion matrix**[5]**:**

		Predicted Class		
		Cat	¬Cat	*Total*
True Class	Cat	**3**	**2**	**5**
	¬Cat	**1**	**TN**	**N**
	Total	**4**	**N'**	**P+N**

[5] We want to evaluate the classification model with regard to the class *Cat*

Evaluation Metrics: Example (I)

- **Evaluation results of a classification model:**

True Class	Cat	Dog	Fox	Cat	Cat	Hen	Cat	Cat	Dog	Fox
Predicted Class	Cat	Dog	Cat	Fox	Cat	Dog	Cat	Dog	Fox	Fox

- **Resulting confusion matrix[5]:**

		Predicted Class		
		Cat	¬Cat	*Total*
True Class	Cat	**3**	**2**	**5**
	¬Cat	**1**	**4**	**5**
	Total	**4**	**6**	**10**

[5] We want to evaluate the classification model with regard to the class *Cat*

Evaluation Metrics: Example (II)

		Predicted Class		
		Cat	¬Cat	*Total*
True Class	Cat	**3**	**2**	**5**
	¬Cat	**1**	**4**	**5**
	Total	**4**	**6**	**10**

Calculations:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}}$$

$$=$$

Results:

		Predicted Class		
		Cat	¬Cat	*Total*
True Class	Cat	**3** (TP)	**2**	**5** (P)
	¬Cat	**1**	**4** (TN)	**5** (N)
	Total	**4**	**6**	**10**

Calculations:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}} = \frac{3+4}{5+5} = \frac{7}{10} = 70\%$$

Results:

$$\text{Accuracy} = 70\%$$

Evaluation Metrics: Example (II)

		Predicted Class		
		Cat	¬Cat	*Total*
True Class	Cat	**3**	**2**	**5**
	¬Cat	**1**	**4**	**5**
	Total	**4**	**6**	**10**

Calculations:

$$\text{Error Rate} = \frac{\text{FP} + \text{FN}}{\text{P} + \text{N}}$$

$$=$$

Results:

$$\text{Accuracy} = 70\%$$

Evaluation Metrics: Example (II)

		Predicted Class		
		Cat	¬Cat	*Total*
True Class	Cat	**3**	**2** (FN)	**5** (P)
	¬Cat	**1** (FP)	**4**	**5** (N)
	Total	**4**	**6**	**10**

Calculations:

$$\text{Error Rate} = \frac{\text{FP} + \text{FN}}{\text{P} + \text{N}} = \frac{1+2}{5+5} = \frac{3}{10} = 30\%$$

Results:

$$\text{Accuracy} = 70\%$$
$$\text{Error Rate} = 30\%$$

Evaluation Metrics: Example (II)

Friedrich-Alexander-Universität
Technische Fakultät

		Predicted Class		
		Cat	¬Cat	*Total*
True Class	Cat	**3**	**2**	**5**
	¬Cat	**1**	**4**	**5**
	Total	**4**	**6**	**10**

Calculations:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{P}}$$

$$=$$

Results:

$$\text{Accuracy} = 70\%$$

$$\text{Error Rate} = 30\%$$

Evaluation Metrics: Example (II)

		Predicted Class		
		Cat	¬Cat	*Total*
True Class	Cat	**3** (TP)	**2**	**5** (P)
	¬Cat	**1**	**4**	**5**
	Total	**4**	**6**	**10**

Calculations:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{P}} = \frac{3}{5} = 60\%$$

Results:

$$\text{Accuracy} = 70\%$$
$$\text{Error Rate} = 30\%$$
$$\text{Sensitivity} = 60\%$$

Evaluation Metrics: Example (II)

Predicted Class

		Cat	¬Cat	*Total*
True Class	Cat	**3**	**2**	**5**
	¬Cat	**1**	**4**	**5**
	Total	**4**	**6**	**10**

Calculations:

$$\text{Specificity} = \frac{\text{TN}}{\text{N}}$$

$$=$$

Results:

$$\text{Accuracy} = 70\%$$
$$\text{Error Rate} = 30\%$$
$$\text{Sensitivity} = 60\%$$

Evaluation Metrics: Example (II)

Predicted Class

True Class		Cat	¬Cat	*Total*
	Cat	**3**	**2**	**5**
	¬Cat	**1**	**4** (TN)	**5** (N)
	Total	**4**	**6**	**10**

Calculations:

$$\text{Specificity} = \frac{\text{TN}}{\text{N}} = \frac{4}{5} = 80\%$$

Results:

$$\text{Accuracy} = 70\%$$
$$\text{Error Rate} = 30\%$$
$$\text{Sensitivity} = 60\%$$
$$\text{Specificity} = 80\%$$

		Predicted Class		
		Cat	¬Cat	*Total*
True Class	Cat	**3**	**2**	**5**
	¬Cat	**1**	**4**	**5**
	Total	**4**	**6**	**10**

Calculations:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$=$$

Results:

$$\text{Accuracy} = 70\%$$
$$\text{Error Rate} = 30\%$$
$$\text{Sensitivity} = 60\%$$
$$\text{Specificity} = 80\%$$

Evaluation Metrics: Example (II)

		Predicted Class		
		Cat	¬Cat	*Total*
True Class	Cat	**3** (TP)	**2**	**5**
	¬Cat	**1** (FP)	**4**	**5**
	Total	**4**	**6**	**10**

Calculations:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{3}{3+1} = \frac{3}{4} = 75\%$$

Results:

$$\text{Accuracy} = 70\%$$
$$\text{Error Rate} = 30\%$$
$$\text{Sensitivity} = 60\%$$
$$\text{Specificity} = 80\%$$

$$\text{Precision} = 75\%$$

Evaluation Metrics: Example (II)

		Predicted Class		
		Cat	¬Cat	*Total*
True Class	Cat	**3**	**2**	**5**
	¬Cat	**1**	**4**	**5**
	Total	**4**	**6**	**10**

Calculations:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$=$$

Results:

$$\text{Accuracy} = 70\%$$
$$\text{Error Rate} = 30\%$$
$$\text{Sensitivity} = 60\%$$
$$\text{Specificity} = 80\%$$

$$\text{Precision} = 75\%$$

Evaluation Metrics: Example (II)

		Predicted Class		
		Cat	¬Cat	*Total*
True Class	Cat	**3** (TP)	**2** (FN)	**5**
	¬Cat	**1**	**4**	**5**
	Total	**4**	**6**	**10**

Calculations:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
$$= \frac{3}{3+2} = \frac{3}{5} = 60\%$$

Results:

$$\text{Accuracy} = 70\%$$
$$\text{Error Rate} = 30\%$$
$$\text{Sensitivity} = 60\%$$
$$\text{Specificity} = 80\%$$

$$\text{Precision} = 75\%$$
$$\text{Recall} = 60\%$$

Evaluation Metrics: Example (II)

Predicted Class

True Class		Cat	¬Cat	*Total*
	Cat	**3**	**2**	**5**
	¬Cat	**1**	**4**	**5**
	Total	**4**	**6**	**10**

Calculations:

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$=$$

Results:

$$\text{Accuracy} = 70\%$$
$$\text{Error Rate} = 30\%$$
$$\text{Sensitivity} = 60\%$$
$$\text{Specificity} = 80\%$$

$$\text{Precision} = 75\%$$
$$\text{Recall} = 60\%$$

Evaluation Metrics: Example (II)

		Predicted Class		
		Cat	¬Cat	*Total*
True Class	Cat	**3**	**2**	**5**
	¬Cat	**1**	**4**	**5**
	Total	**4**	**6**	**10**

Calculations:

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot 0.75 \cdot 0.6}{0.75 + 0.6} \approx 0.6667 \approx 67\%$$

Results:

$$\text{Accuracy} = 70\%$$
$$\text{Error Rate} = 30\%$$
$$\text{Sensitivity} = 60\%$$
$$\text{Specificity} = 80\%$$

$$\text{Precision} = 75\%$$
$$\text{Recall} = 60\%$$
$$F_1 \approx 67\%$$

Friedrich-Alexander-Universität
Technische Fakultät

Model Evaluation

Evaluation Strategies

Evaluation Strategies

Friedrich-Alexander-Universität
Technische Fakultät

- **We will take a look at two types of evaluation strategies:**
  - **Methods to split data into training and test sets:**
    1. Holdout method
    2. Cross validation
  - **Methods to compare classification models and their settings:**
    1. Receiver Operating Characteristics (ROC) curve
- **Of course, there are many many more evaluation strategies**

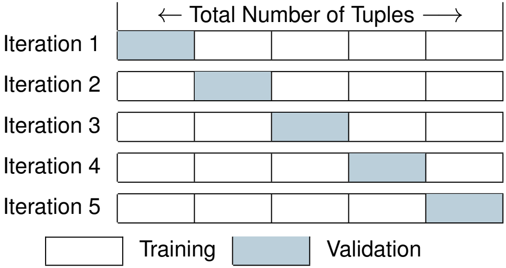
Evaluation Strategies: Holdout Method

- **Easy way to split a dataset: The Holdout Method**
  - Randomly assign tuples into two independent sets:
    - **Training set** (E.g., $2/3$) for model construction.
    - **Test set** (E.g., $1/3$) for accuracy estimation.
  - Random sampling: a variation of holdout that repeats holdout k times.
    - Create an average accuracy over all experiments.

Evaluation Strategies: Cross Validation

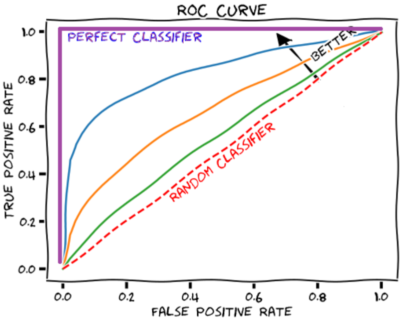
- **More robust than holdout method:** **Cross Validation**
- **In this case:** ***k*-fold cross validation**
  - Randomly partition the data into k mutually exclusive subsets (folds).
  - At each iteration, use one fold as test set and the others as training set.
  - Average accuracy of all iterations.

Example: k-fold cross validation with $k = 5$



Evaluation Strategies: ROC Curve

- **Receiver Operating Characteristics (ROC) curve:**
  - Visualizes the performance of a classification model:
    - Shows the performance of a model at different settings/thresholds.
    - Plots the True Positive Rate (TPR) against the False Positive Rate (FPR).
    - Shows the trade-off between sensitivity and specificity.
  - The closer the curve is to the top-left corner, the better the model.
    $\Rightarrow$ **The area under the ROC curve (AUC) is a measure of the model's accuracy.**



Ensemble Methods

Ensemble Method

An *ensemble method* creates a composite model that consists of several models to form one model.

- **Basic Idea:**
  - Use multiple models to improve classification accuracy.
  - The final prediction is made by combining the predictions of all models.
- **Popular methods:**
  - Bagging
  - Boosting
    - Popular algorithm: AdaBoost
      $\Rightarrow$ More on that in the appendix. [6]
  - Random Forest
    - Algorithm specialized on decision trees
      $\Rightarrow$ More on that in the appendix. [7]

[6] Not part of the exam.
[7] Not part of the exam.

Bagging

- **Basic Idea:**
  - Multiple models classify the same tuple.
  - The bagged classifier collects results.
  - The class with the **most votes** is returned as the final prediction.

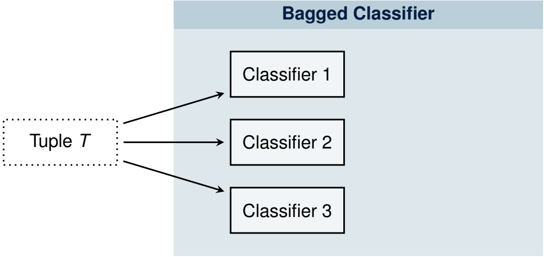
Bagging

- **Basic Idea:**
  - Multiple models classify the same tuple.
  - The bagged classifier collects results.
  - The class with the **most votes** is returned as the final prediction.

Tuple T

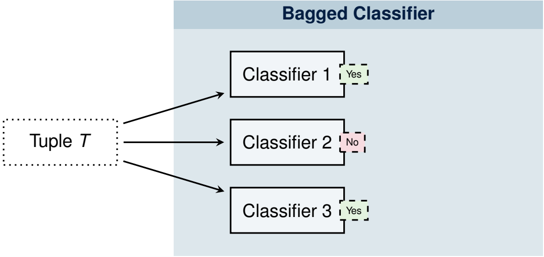
Bagging

- **Basic Idea:**
  - Multiple models classify the same tuple.
  - The bagged classifier collects results.
  - The class with the **most votes** is returned as the final prediction.



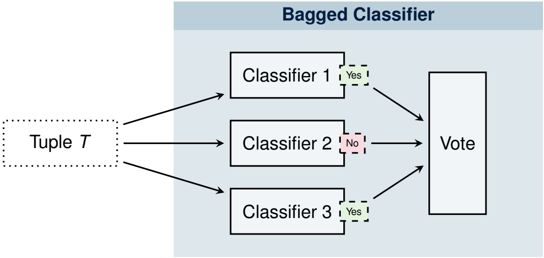
Bagging

- **Basic Idea:**
  - Multiple models classify the same tuple.
  - The bagged classifier collects results.
  - The class with the **most votes** is returned as the final prediction.



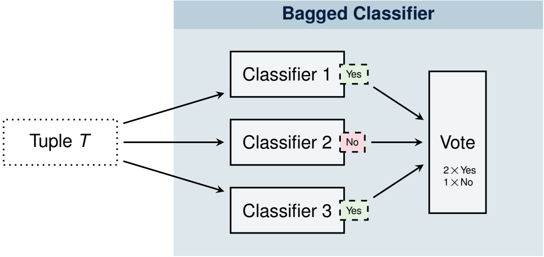
Bagging

- **Basic Idea:**
  - Multiple models classify the same tuple.
  - The bagged classifier collects results.
  - The class with the **most votes** is returned as the final prediction.



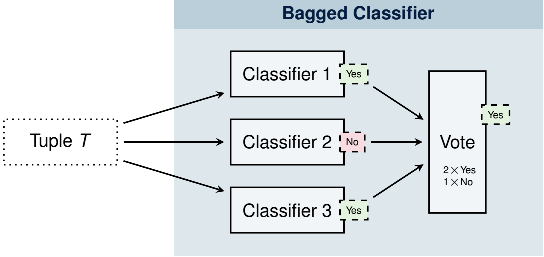
Bagging

- **Basic Idea:**
  - Multiple models classify the same tuple.
  - The bagged classifier collects results.
  - The class with the **most votes** is returned as the final prediction.



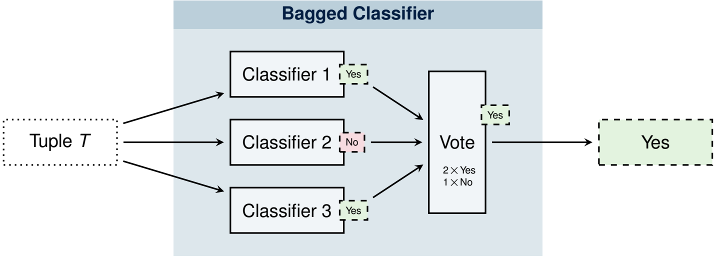
Bagging

- **Basic Idea:**
  - Multiple models classify the same tuple.
  - The bagged classifier collects results.
  - The class with the **most votes** is returned as the final prediction.



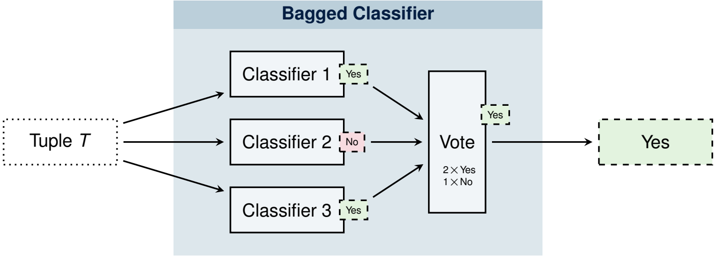
Bagging

- **Basic Idea:**
  - Multiple models classify the same tuple.
  - The bagged classifier collects results.
  - The class with the **most votes** is returned as the final prediction.



Bagging

- **Basic Idea:**
  - Multiple models classify the same tuple.
  - The bagged classifier collects results.
  - The class with the **most votes** is returned as the final prediction.
- **Continuous-valued Attributes:**
  - Multiple models predict the same tuple.
  - The bagged regressor collects results.
  - The final prediction is the **average** of all predictions.



Boosting

- **Training:**
  - Classifiers are iteratively learned.
  - After a classifier is learned, the weights of the training tuples are updated.
    $\Rightarrow$ The next classifier pays more attention to the misclassified tuples.

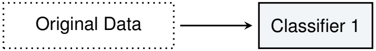
Boosting

- **Training:**
  - Classifiers are iteratively learned.
  - After a classifier is learned, the weights of the training tuples are updated.
    $\Rightarrow$ The next classifier pays more attention to the misclassified tuples.

Original Data

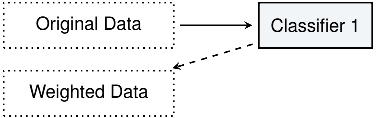
Boosting

- **Training:**
  - Classifiers are iteratively learned.
  - After a classifier is learned, the weights of the training tuples are updated.
    $\Rightarrow$ The next classifier pays more attention to the misclassified tuples.



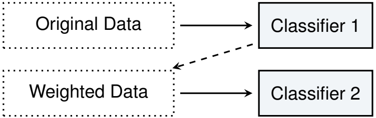
Boosting

- **Training:**
  - Classifiers are iteratively learned.
  - After a classifier is learned, the weights of the training tuples are updated.
    $\Rightarrow$ The next classifier pays more attention to the misclassified tuples.



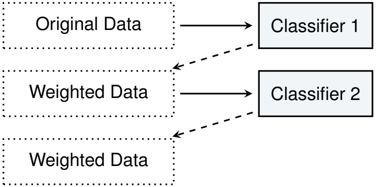
Boosting

- **Training:**
  - Classifiers are iteratively learned.
  - After a classifier is learned, the weights of the training tuples are updated.
    $\Rightarrow$ The next classifier pays more attention to the misclassified tuples.



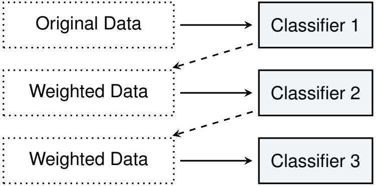
Boosting

- **Training:**
  - Classifiers are iteratively learned.
  - After a classifier is learned, the weights of the training tuples are updated.
    $\Rightarrow$ The next classifier pays more attention to the misclassified tuples.



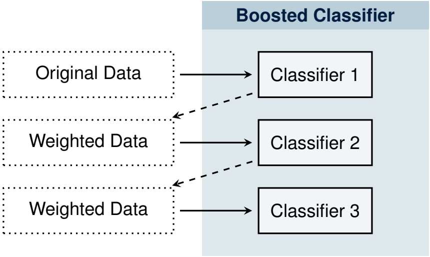
Boosting

- **Training:**
  - Classifiers are iteratively learned.
  - After a classifier is learned, the weights of the training tuples are updated.
    $\Rightarrow$ The next classifier pays more attention to the misclassified tuples.



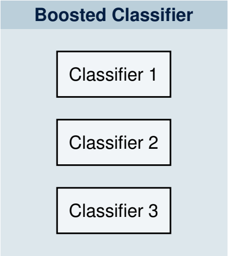
Boosting

- **Training:**
  - Classifiers are iteratively learned.
  - After a classifier is learned, the weights of the training tuples are updated.
    $\Rightarrow$ The next classifier pays more attention to the misclassified tuples.



Boosting

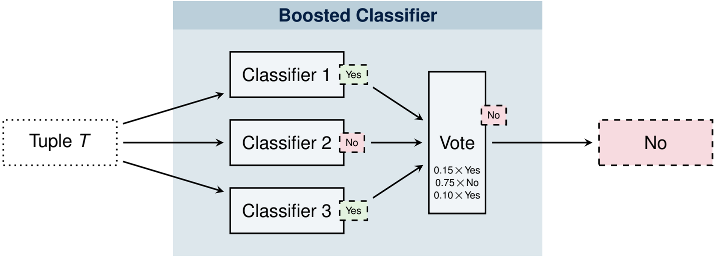
- **Training:**
  - Classifiers are iteratively learned.
  - After a classifier is learned, the weights of the training tuples are updated.
    $\Rightarrow$ The next classifier pays more attention to the misclassified tuples.
- **Predictions:**
  - All models classify the same tuple.
  - The boosted classifier collects results.
  - The weight of each classifier's vote is a function of its accuracy.



Boosting

- **Training:**
  - Classifiers are iteratively learned.
  - After a classifier is learned, the weights of the training tuples are updated.
    $\Rightarrow$ The next classifier pays more attention to the misclassified tuples.

- **Predictions:**
  - All models classify the same tuple.
  - The boosted classifier collects results.
  - The weight of each classifier's vote is a function of its accuracy.



Summary

Summary

- **Classification:**
  - A form of data analysis that extracts models describing important data classes.
- **Effective and scalable methods:**
  - Decision-tree induction, rule-based classification, and naïve Bayesian classification.
- **Evaluation metrics:**
  - Accuracy, sensitivity, specificity, precision, recall, and F_β-measure.
- **Evaluation strategies:**
  - Holdout, cross-validation, and ROC-curve analysis.
- **Ensemble methods:**
  - Boosting and Bagging.

Any questions about this chapter?

Ask them now or ask them later in our forum:

https://www.studon.fau.de/studon/goto.php?target=lcode_OLYeD79h

Appendix

Appendix

Decision Tree Induction

Basic Decision Tree Algorithm

Friedrich-Alexander-Universität
Technische Fakultät

Data:

- Training dataset *D* containing tuples with their associated class labels;
- `attribute_list`, the set of candidate attributes;
- `attribute_selection_method`, a method to determine the splitting criterion that "best" partitions the data tuples into individual classes. The criterion consists of a `splitting_attribute`, and possibly, either a `split_point` or `splitting_subset`.

Result: A decision tree.

```
create a node N;
if tuples in D are all of the same class C then
    return N as a leaf node labeled with the class C;
if attribute_list is empty then
    /* Majority voting */
    majority_class ← determine majority class in D;
    return N as a leaf node labeled with majority_class;
/* apply attribute_selection_method to find the
   "best" splitting_criterion */
splitting_criterion ← attribute_selection_method(D,
    attribute_list);
label node N with splitting_criterion;
if (splitting_attribute is discrete-valued and multiway splits
    allowed) or attribute value has only one unique value then
    // remove splitting_attribute
    attribute_list ← attribute_list - splitting_attribute;
foreach outcome j of splitting_criterion do
    /* partition the tuples and grow subtrees for
       each partition */
    D_j ← partition D to satisfy outcome j;
    if D_j is empty then
        attach a leaf labeled with the majority class in D to node N;
    else
        attach the node return by build_decision_tree(D_j,
            attribute_list) to node N;
return N;
```

Algorithm 1: `build_decision_tree`. Generate a decision tree from training tuples in data partition *D*.

Other Attribute Selection Methods

- **CHAID:**
  - A popular decision tree algorithm, measure based on χ^2 test for independence.
- **C-SEP:**
  - Performs better than Information Gain and Gini Index in certain cases.
- **G-statistic:**
  - Has a close approximation to χ^2 distribution.
- **MDL (Minimal Description Length) principle:**
  - I.e. the simplest solution is preferred.
  - The best tree is the one that requires the fewest number of bits to both (1) encode the tree and (2) encode the exceptions to the tree.
- **Multivariate splits:**
  - Partitioning based on multiple variable combinations.
  - CART: finds multivariate splits based on a linear combination of attributes.
- **Which Attribute Selection Method is the best?**
  - Most give good results, none is significantly superior to others.

Appendix

Evaluation: Bootstrap and Statistical Significance

Evaluation Strategy: Bootstrap

Bootstrap samples training data uniformly with replacement.

Several bootstrap methods exists, yet a common one is .632 **bootstrap.**

- Data set with d tuples is sampled d times - uniformly with replacement.
- Uniformly = every tuple has the same probability ($\frac{1}{d}$) for selection.
- With replacement = High change a tuple is selected more than once.
- Not selected tuples will form the test set.
- Probability of not being chosen is $1 - \frac{1}{d}$. Selecting d times: $(1 - \frac{1}{d})^d$.
  With a large data set it approaches $e^-1 = 0.368$.
- Thus, on average 63.2% of tuples are selected as the training set.
- Sampling procedure is repeated k times.
  Calculate accuracy in every iteration as follows:

$$\mathrm{Acc}(M) = \frac{1}{k}\sum_{i=1}^{k} 0.632 \cdot \mathrm{Acc}(M_i)_{\text{test\_set}} + 0.368 \cdot \mathrm{Acc}(M_i)_{\text{train\_set}}.$$

Evaluating Classifier Accuracy: Bootstrap (II)

- **Suppose we have 2 classifiers, M_1 and M_2, which one is better?**
- **Use 10-fold cross-validation to obtain $\overline{\text{err}}(M_1)$ and $\overline{\text{err}}(M_2)$.**
  - Recall: error rate is $1 - \text{accuracy}(M)$.
- **Mean error rates:**
  - Just estimates of error on the true population of future data cases.
- **What if the difference between the 2 error rates is just attributed to chance?**
  - Use a test of statistical significance.
  - Obtain confidence limits for our error estimates.

Evaluating Classifier Accuracy: Null Hypothesis

- **Perform k-fold cross-validation** with $k = 10$.
- **Assume samples follow a t-distribution with $k - 1$ degrees of freedom.**
- **Use t-test**
  - Student's t-test.
- **Null hypothesis:**
  - M_1 and M_2 are the same.
- **If we can reject the null hypothesis, then**
  - Conclude that difference between M_1 and M_2 is statistically significant.
  - Obtain confidence limits for our error estimates.

Estimating Confidence Intervals (I)

- **If only one test set available: pairwise comparison:**
  - For i-th round of 10-fold cross-validation, the same cross partitioning is used to obtain $\text{err}(M_1)_i$ and $\text{err}(M_2)_i$.
  - Average over 10 rounds to get $\overline{\text{err}}(M_1)$ and $\overline{\text{err}}(M_2)$.
  - t-test computes t-statistic with $k-1$ degrees of freedom:

$$t = \frac{\overline{\text{err}}(M_1) - \overline{\text{err}}(M_2)}{\frac{\text{var}(M_1 - M_2)}{\sqrt{k}}},$$

  - where

$$\text{var}(M_1 - M_2) = \frac{1}{k}\sum_{i=1}^{k}\left[\text{err}(M_1)_i - \text{err}(M_2)_i - (\overline{\text{err}}(M_1) - \overline{\text{err}}(M_2))\right]^2.$$

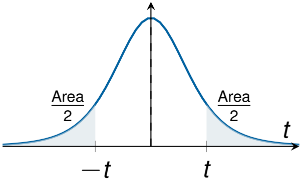
- **If two test sets available: use unpaired t-test:**

$$\text{var}(M_1 - M_2) = \sqrt{\frac{\text{var}(M_1)}{k_1} + \frac{\text{var}(M_2)}{k_2}},$$

where k_1 & k_2 are # of cross-validation samples used for M_1 & M_2, respectively.

Estimating Confidence Intervals (II)

- Symmetrical.
- **Significance level**:
  - E.g., sig $= 0.05$ or 5% means M_1 & M_2 are significantly different for 95% of population.
- Confidence limit: $z = \frac{\text{sig}}{2}$.



	Area in One Tail[1]		
	0.100	0.050	0.005
	Area in Two Tails[1]		
df/α	0.200	0.100	0.010
1	3.078	6.314	63.657
2	1.886	2.920	9.925
3	1.638	2.353	5.841
4	1.533	2.132	4.604
5	1.476	2.015	3.707
6	1.440	1.943	3.499
7	1.415	1.895	3.355
8	1.397	1.860	3.250
9	1.372	1.833	3.169

[1] Good link for a full table: `https://www.hawkeslearning.com/documents/statdatasets/stat_tables.pdf`

Estimating Confidence Intervals (III)

Friedrich-Alexander-Universität
Technische Fakultät

Are M_1 and M_2 significantly different?

- Compute t. Select significance level (E.g., sig = 5%).
- Consult table for t-distribution:
  - t-distribution is symmetrical:
    - Typically upper % points of distribution shown.
  - Find critical value c corresponding to $k - 1$ degrees of freedom (here, 9)
  - and for confidence limit $z = \frac{\text{sig}}{2}$ (here, 0.025).
  - $\implies$ Here, critical value $c = 2.262$
- If $t > c$ or $t < -c$, then t value lies in rejection region:
  - **Reject null hypothesis** that mean error rates of M_1 and M_2 are equal.
  - Conclude: **statistically significant difference** between M_1 and M_2.
- Otherwise, conclude that any difference is chance.

Appendix

Ensemble Methods: AdaBoost and Random Forests

AdaBoost ("Adaptive Boosting"[8]): Training

Friedrich-Alexander-Universität
Technische Fakultät

- **Given a data set D of d class-labeled tuples:** $(x_1, y_1), \ldots, (x_d, y_d)$ with $y_d \in Y = \{1, \ldots, c\}$.
- **Initialize empty lists to hold information per classifier: $\mathbf{w}$, $\boldsymbol{\beta}$, $\mathbf{M}$** $\leftarrow$ empty list.
- **Initialize weights for first classifier to hold same probability for each tuple:** $w_j^1 \leftarrow \frac{1}{d}$
- **Generate K classifiers in K iterations. At iteration k,**
  1. Calculate “normalized” weights: $\mathbf{p}^k = \frac{\mathbf{w}^k}{\sum_{j=1}^d w_j^i}$
  2. Sample dataset with replacement according to $\mathbf{p}^k$ to form training set D_k.
  3. Derive classification model M_k from D_k.
  4. Calculate error ε_k by using D_k as a test set as follows: $\varepsilon_k = \sum_{j=1}^d p_j^k \cdot \mathrm{err}(M_k, x_j, y_j)$, where the *misclassification error* $\mathrm{err}(M_k, x_j, y_j)$ returns 1 if $M_k(x_j) \neq y_j$, otherwise it returns 0.
  5. If $\mathrm{error}(M_k) > 0.5$: Abandon this classifier and go back to step 1.
  6. Calculate $\beta_k = \frac{\varepsilon_k}{1 - \varepsilon_k}$.
  7. Update weights for the next iteration: $w_j^{k+1} = w_j^k \beta_k^{1 - err(M_k, x_j, y_j)}$. *If a tuple is misclassified, its weight remains the same, otherwise it is decreased.* Misclassified tuple weights are increased relatively.
  8. Add $\mathbf{w}^{k+1}$, M_k, and β_k to their respective lists.

AdaBoost ("Adaptive Boosting"[10]): Prediction

- **Initialize weight of each class to zero.**
- **For each classifier i in k classifiers:**
  1. Calculate the weight of this classifier's vote: $w_i = \log(\frac{1}{\beta_i})$.
  2. Get class prediction c for (single) tuple x from current weak classifier M_i: $c = M_i(x)$.
  3. Add w_i to weight for class c.
- **Return predicted class with the largest weight.**
- Mathematically, this can be formulated as:

$$M(x) = \arg\max_{y \in Y} \sum_{i=1}^{k} (\log \frac{1}{\beta_i}) M_i(x)$$

[9] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997. DOI: 10.1006/jcss.1997.1504. [Online]. Available: https://doi.org/10.1006/jcss.1997.1504, Algorithm AdaBoost.M1 on p. 131.

Random Forest[12]

- Ensemble method consisting only of decision trees where each tree has been generated using random selection of attributes at each node.
- Classification: Each tree votes and the most popular class is returned.
- **Two methods to construct random forests:** (each builds *k* trees)
  1. Forest-RI (random input selection):
     - Random sampling with replacement to obtain training data from *D*.
     - Set *F* as the number of attributes to determine split at each node. *F* is smaller than the number of available attributes.
     - Construct decision tree M_i by randomly select candidates at each node. Use CART to grow tree to maximum size without pruning.
  2. Forest-RC: Similar to Forest-RI but new attributes (features) are generated by linear combinations of existing attributes to reduce correlation between individual classifiers. At each node, attributes are randomly selected.
- **Comparable in accuracy to AdaBoost, but more robust to errors and outliers.**
- **Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting.**

[11] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997. DOI: 10.1006/jcss.1997.1504. [Online]. Available: https://doi.org/10.1006/jcss.1997.1504, Algorithm AdaBoost.M1 on p. 131.

Class-Imbalanced Data

Class-Imbalanced Data refers to data where the main class of interest (positive labeled) is only represented by a small number of tuples. E.g., medical diagnosis and fraud detection.

- Problem because traditional methods assume *equality between classes*, i. e. a balanced distribution of classes and equal error costs.
- **Typical methods for imbalanced data in binary classification:**
  1. **Undersampling/Oversampling:** Changes distribution of tuples in training data.
     - *Undersampling:* Randomly eliminate tuples from negative class.
     - *Oversampling:* Re-samples data from positive class.
       For instance, method SMOTE generates synthetic data that is similar to existing data using nearest neighbor.
  2. **Threshold-moving:** Moves the decision threshold, t, so that the rare-class tuples are easier to classify, and hence, less chance of costly false-negative errors. Works when class returns a probability.
  3. **Ensemble techniques**.

  Threshold-moving and ensemble methods work well on extremely imbalanced data.
- **Still difficult on multi-class tasks.**

[13] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001. DOI: 10.1023/A:1010933404324. [Online]. Available: https://doi.org/10.1023/A:1010933404324.