UPC
Universidad Peruana
de Ciencias Aplicadas

Introduction to Supervised Learning

Basic Methods

Marcelo S. Lauretto

Escola de Artes, Ciências e Humanidades,
Universidade de São Paulo
marcelolauretto@usp.br

Lima - Peru

Algorithms - basic methods

- Inferring rudimentary rules
- Constructing decision trees
- Statistical modeling

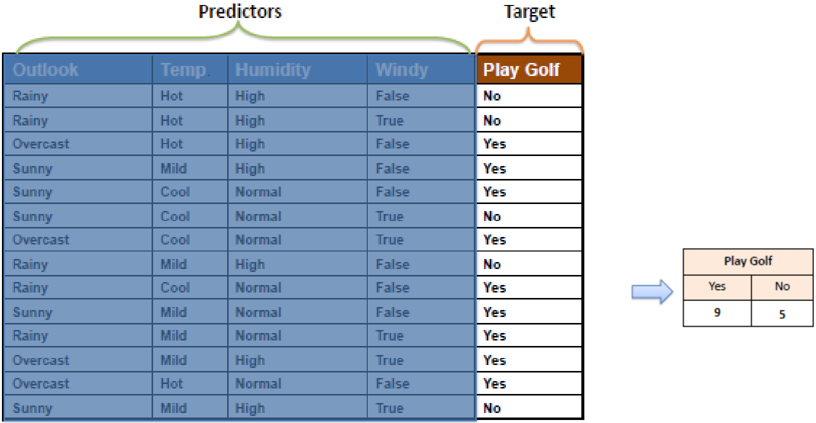
0-R – “Zero-Rule”

- The simplest classification method which relies on the target (class attribute) and ignores all predictors (attributes).
- Despite its lack of power, it is useful for determining a baseline performance as a benchmark for oher classification methods.
- **Algorithm**:
  - Construct a frequency table for the target and select its most frequent value.

0-R – "Zero-Rule"

- Example: The weather problem:

Plav = Yes



Predictors				Target
Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Play Golf	
Yes	No
9	5

1-R – “One-Rule”

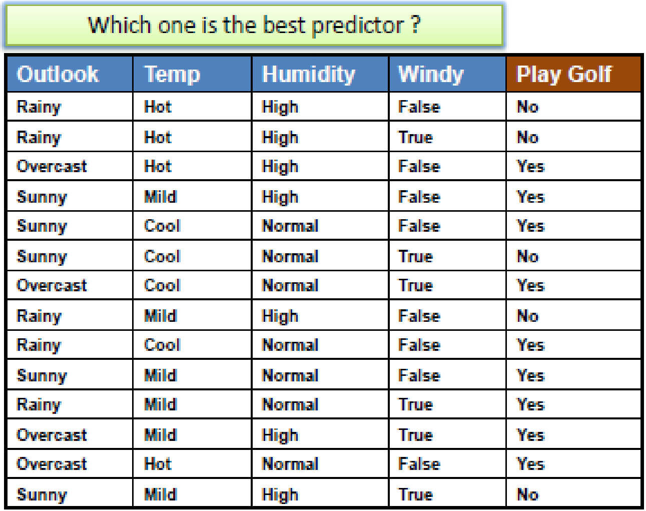
- Learns a 1-level decision tree
- Generates one rule for each predictor in the data, then selects the rule with the smallest total error as its "one rule".
- In some cases, OneR produces rules only slightly less accurate than state-of-the-art classification algorithms while producing rules that are simple for humans to interpret.
- **Algorithm**

```
For each attribute,
    For each value of the attribute, make a rule as follows:
        count how often each class appears
        find the most frequent class
        make the rule assign that class to this attribute-value
    Calculate the error rate of the rules
Choose the rules with the smallest error rate
```

- “Missing” is treated as a separate attribute value

1-R – “One-Rule”

- Example: The weather problem:

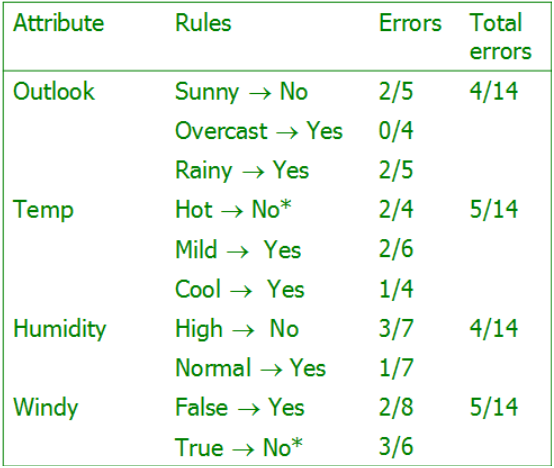


Which one is the best predictor ?

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

1-R – “One-Rule”

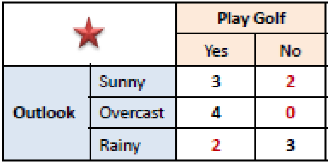
- Example: The weather problem:



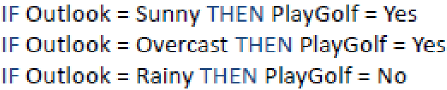
Attribute	Rules	Errors	Total errors
Outlook	Sunny → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	
Temp	Hot → No*	2/4	5/14
	Mild → Yes	2/6	
	Cool → Yes	1/4	
Humidity	High → No	3/7	4/14
	Normal → Yes	1/7	
Windy	False → Yes	2/8	5/14
	True → No*	3/6	

1-R – “One-Rule”

- Example: The weather problem:
- The best predictor:



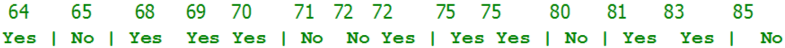
★		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3



IF Outlook = Sunny THEN PlayGolf = Yes
IF Outlook = Overcast THEN PlayGolf = Yes
IF Outlook = Rainy THEN PlayGolf = No

1-R – Numeric Attributes

- Discretize numeric attributes
- Divide each attribute's range into intervals
  - Sort instances according to attribute's values
  - Place breakpoints where class changes (majority class)
  - This minimizes the total error
- Example: *temperature* from weather data

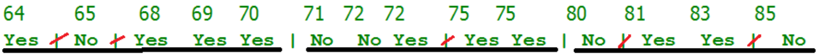


```
 64    65    68   69   70    71  72  72    75   75    80    81    83    85
Yes | No | Yes  Yes Yes | No  No Yes | Yes Yes | No | Yes  Yes |  No
```

- Missing values in numeric attributes:
  - An additional category is created for them
  - Discretization procedure is applied just for instances for which the attribute's value is defined

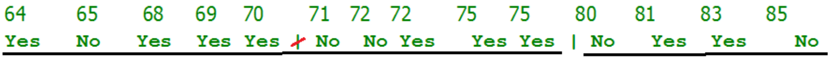
1-R – The Problem of Overfitting

- Procedure above tends to *overfit* (i.e. to produce rules with very low error rate for the training set but useless for classifying new instances)
    - Instances with incorrect class labels may produce separate intervals
    - Time stamp attribute will have zero errors (on the treining set)
- Simple solution: enforce minimum number of instances in majority class per interval (exhaustive search for the best combination)
- Example (with min=3):



```
64    65    68    69    70    71   72   72    75    75    80    81    83    85
Yes   No    Yes   Yes   Yes | No   No   Yes   Yes   Yes | No    Yes   Yes   No
```

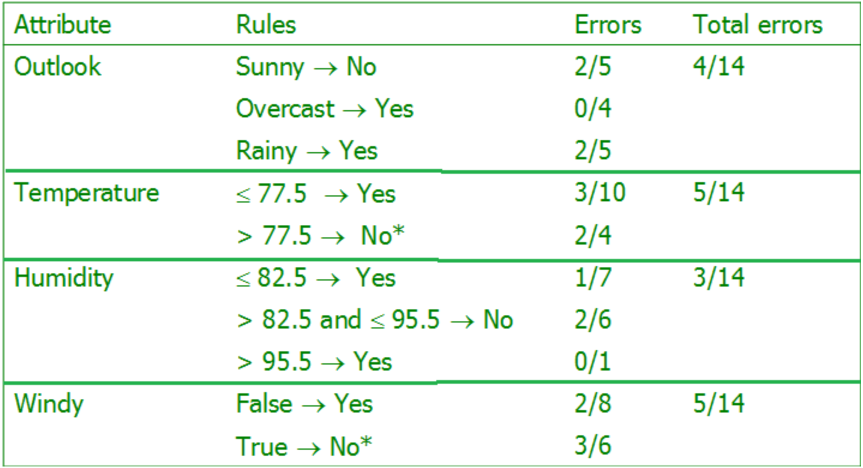
Merging adjacent intervals labelled with the same class:



```
64    65    68    69    70    71   72   72    75    75    80    81    83    85
Yes   No    Yes   Yes   Yes   No   No   Yes   Yes   Yes | No    Yes   Yes   No
```

1-R – With Overfitting Avoidance

- Resulting rule set:

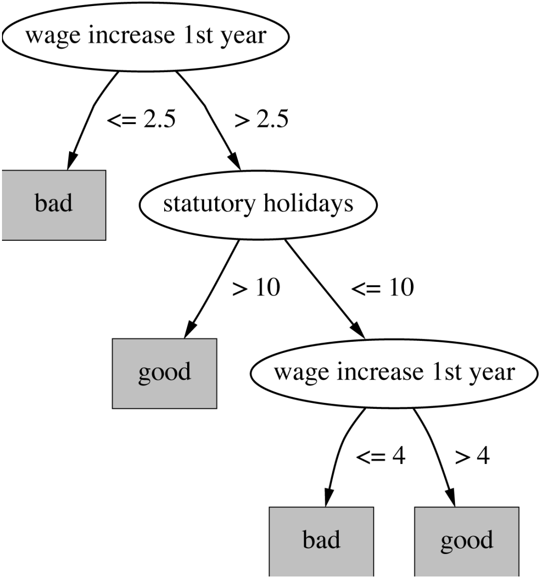


Attribute	Rules	Errors	Total errors
Outlook	Sunny → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	
Temperature	≤ 77.5 → Yes	3/10	5/14
	> 77.5 → No*	2/4	
Humidity	≤ 82.5 → Yes	1/7	3/14
	> 82.5 and ≤ 95.5 → No	2/6	
	> 95.5 → Yes	0/1	
Windy	False → Yes	2/8	5/14
	True → No*	3/6	

Classification Trees

- A *Classification Tree* is a recursive structure where:
  - Inner (non-terminal) node are *decision nodes*, labeled with attributes (or binary conditions) ;
  - Leaves (terminal nodes) are *response node*, labeled with classes (or estimated probabilities for all classes);
  - Edges linking inner nodes their children are labeled with values, intervals or subsets of the respective attributes.
  - Paths from the root to leaves correspond to decision rules.
- Variant: *Regression Trees*, for continuous target variables

A Classification Tree for the Labor Data



Induction of Classification Trees

- TDIDT: top-down induction of decision trees
- Basic principles:
  - Starting in the root node, recursively partitioning the training set using one-rule scheme variants
  - Each subset of training examples generated by the split rule corresponds to a new child node in the tree
  - When all examples in a subset have the same class label or when the subset achieves a stop rule:
    - the corresponding node is declared as a terminal node.
    - a class label (or a vector with class probabilities) is assigned
- Variant: *Regression Trees*, for continuous target variables

Induction of Classification Trees

- TDIDT: top-down induction of decision trees
- Recursive *divide-and-conquer* fashion:
  1. First: Select the best attribute for root node
     Create a branch for each possible attribute value (or according to the split scheme of the algorithm)
  2. Then: split instances into subsets
     One for each branch extending from the node
  3. Finally: repeat recursively for each branch, using only instances that reach the branch
- When all examples have the same class label or when the subset achieves a stop rule:
  - the corresponding node is declared as a terminal node.
  - a class label (or a vector with class probabilities) is assigned

General Algorithm for Induction of Classification Trees

- Main components:
    - Stop rule for tree expansion of a note t: Stp(t)
    - Class labeling criterion: Label(t)
    - Score function for evaluation of a split s^j of attribute a_j for the subset $\mathcal{L}$: score($\mathcal{L}, j, s^j$)

General Algorithm for Induction of Classification Trees

Build-Tree(t, $\mathcal{L}$, Stp, Label, score)

If training set $\mathcal{L}$ satisfies the stop rule Stp(t),

- **then** label t according to rule Label(t)
- **otherwise**
  a) For each attribute $a_j,\ j = 1 \dots M$:
     - for each possible valid split $s_1^j, s_2^j, \dots s_q^j$ of attribute a_j, evaluate score$(\mathcal{L}, j, s_q^j)$
     - choose the partition s^* with maximum score

  b) Choose the attribute a_j^* which yields the maximum score
  c) Label t with attribute a_j^*
  d) Split the training set $\mathcal{L}$ into the subsets $\mathcal{L}_1, \dots, \mathcal{L}_z$ induced by s^*
  e) Create new children nodes $t_1, \dots, t_z$ corresponding to subsets $\mathcal{L}_1, \dots, \mathcal{L}_z$ and apply the algorithm recursively

Algorithm for Classification Trees – Notation

- $n_{\bullet,t}$: number of instances of $\mathcal{L}$ incident on node t
- $n_{k,t}$: number of instances of class k incident on t
- $\pi_{k,t}$: (unknown) probability of an instance $\boldsymbol{x} \in \mathcal{X}$ incident on node t belonging to class k.
    - Estimator for $\pi_{k,t}$: proportion of class k in node t:

$$\hat{\pi}_{k,t} = \frac{n_{k,t}}{n_{\bullet,t}}$$

Terminal Node Labeling – Minimum Error

- Basic Idea: Label the node with the majority class.
- Formalization:
    - $\widehat{err}_t(k)$: denotes the estimated probability of an instance incident on t belonging to a class different from k, given that t is labeled with class k

      $$\widehat{err}_t(k) = \sum_{l \neq k} \hat{\pi}_{l,t} = 1 - \hat{\pi}_{k,t}$$

    - Hence, the minimum classification error, denoted $r(t)$, is given by

      $$r(t) = \min_{k=1 \ldots K} \widehat{err}_t(k) = 1 - \max_{k=1 \ldots K} \hat{\pi}_{k,t},$$

    - Therefore, the majority labeling criterion minimizes the classification error:

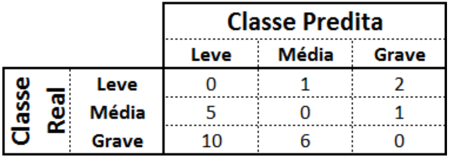
      $$k^* = \arg \min_{k=1 \ldots K} \widehat{err}_t(k) = \arg \max_{k=1 \ldots K} \hat{\pi}_{k,t}.$$

Terminal Node Labeling – Minimum Cost

- Not all misclassification types have the same consequences
- Which is worse?
  - To diagnose a healthy patient as ill, or to diagnose an ill patient as healthy?
  - To deny credit to a good customer, or to give loan to a bad customer?
- $C(l, k)$: cost of assigning class k to an instance which true class is l

$$C(l,k) = \begin{cases} 0 & \text{if } k = l \\ \geq 0 & \text{if } k \neq l. \end{cases}$$

- Example: severity of a disease



		Classe Predita		
		Leve	Média	Grave
Classe Real	Leve	0	1	2
	Média	5	0	1
	Grave	10	6	0

Terminal Node Labeling – Minimum Cost

- If t is labeled with class k, the expected misclassification cost is given by

$$\widehat{Cerr}_t(k) = \sum_{l \neq k} \hat{\pi}_{l,t} C(l, k)$$

- Under this criterion, $r(t)$ denotes the minimum expected misclassification cost for an instance incident on node t:

$$r(t) = \min_k \widehat{Cerr}_t(k).$$

- The class k^* chosen for labeling t is therefore

$$k^* = \arg \min_{k=1\ldots K} \widehat{Cerr}_t(k)$$

- *Note:* The minimum error criterion is a particular case of minimum cost, with

$$C(l, k) = \begin{cases} 0 & \text{if } k = l \\ 1 & \text{if } k \neq l. \end{cases}$$

Attribute Selection

- In the above generic algorithm, score function evaluates attribute splits
- Two classical split criteria based on *impurity*:
    - Entropy
    - Gini Index
- Impurity function requirements:
    1. Impurity is null if all instances are of same class
    2. Impurity is maximum if all instances have the same frequency (uniform distribution)
    3. Impurity is symmetrical on the classes

Attribute Selection – Gini index

- Simplified version of Gini coefficient, developed by Corrado Gini in 1912
    - One of the most commonly used measure to represent the income distribution of a nation's residents (measure of inequality)
- Gini index formulation:

$$
\begin{aligned}
\mathcal{G}(t) &= \sum_{l \neq k} \hat{\pi}_{k,t} \hat{\pi}_{l,t} = \sum_{k=1}^{K} \hat{\pi}_{k,t} \cdot \sum_{l \neq k} \hat{\pi}_{l,t} \\
&= \sum_{k=1}^{K} \hat{\pi}_{k,t} \cdot (1 - \hat{\pi}_{k,t}) = \sum_{k=1}^{K} \hat{\pi}_{k,t} - \sum_{k=1}^{K} \hat{\pi}_{k,t}^{2} \\
&= 1 - \sum_{k=1}^{K} \hat{\pi}_{k,t}^{2}
\end{aligned}
$$

Attribute Selection – Gini index

- Purity gain:
  - impurity before splitting - impurity after splitting
  - given a candidate split s of a node t yielding the children nodes $t_1, t_2, \ldots, t_Z$ the purity gain is given by:

$$\Delta\mathcal{G}(t) = \mathcal{G}(t) - \frac{N_{t_1}}{N_t}\mathcal{G}(t_1) - \ldots - \frac{N_{t_z}}{N_t}\mathcal{G}(t_z)$$

  where N_t is the number of instances in node t and $N_{t_1}, N_{t_2}, \ldots, N_{t_Z}$ are the number of instances distributed among $t_1, t_2, \ldots, T_Z$.

- Attribute selection: maximum purity gain

Attribute Selection – Shannon's Entropy

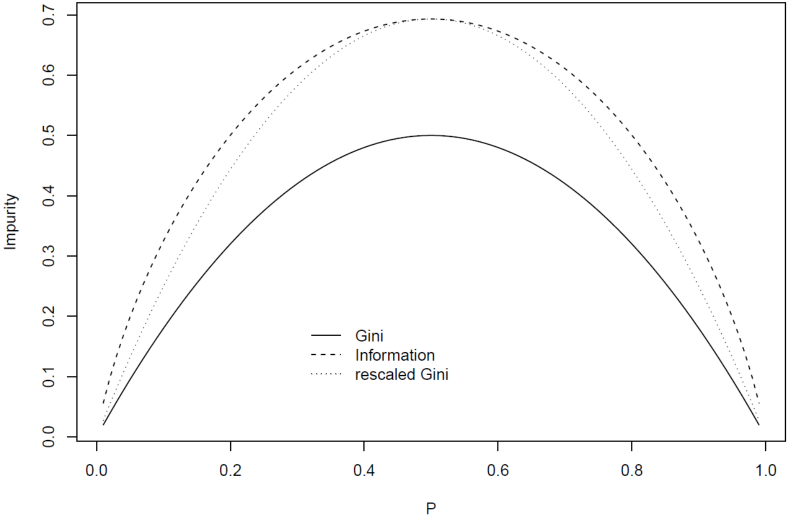
- Entropy:
  - Measure of disorder or uncertainty
  - Raised in thermodynamics and statistical mechanics
- Shannon's entropy:
  - Introduced by Claude E. Shannon in 1948
  - Provides an absolute limit on the best possible average length of lossless encoding or compression of any communication
- Entropy computation for a node t:

$$\mathcal{E}(t) = -\sum_{k=1}^{K} \hat{\pi}_{k,t} \cdot \log_2[\hat{\pi}_{k,t}]$$

- Information gain:

$$\Delta\mathcal{E}(t) = \mathcal{E}(t) - \frac{N_{t_1}}{N_t}\mathcal{E}(t_1) - \ldots - \frac{N_{t_Z}}{N_t}\mathcal{E}(t_Z)$$

Attribute Selection – Gini vs Entropy



Attribute Selection – Gain Ratio

- Impurity functions issue: bias towards multi-valued attributes
  - Subsets are more likely to be pure if there is a large number of values
    - ⇒ For non-binary trees, impurity functions (like Entropy and Gini index) are biased towards choosing attributes with a large number of values
    - ⇒ This may result in overfitting (selection of an attribute that is non-optimal for prediction)
- Gain ratio: a modification of the information gain that reduces its bias
  - It corrects the information gain by taking the intrinsic information of a split into account

Attribute Selection – Gain Ratio

- Intrinsic information: entropy of distribution of instances into branches (a measure of the "spreading" of instances along branches):

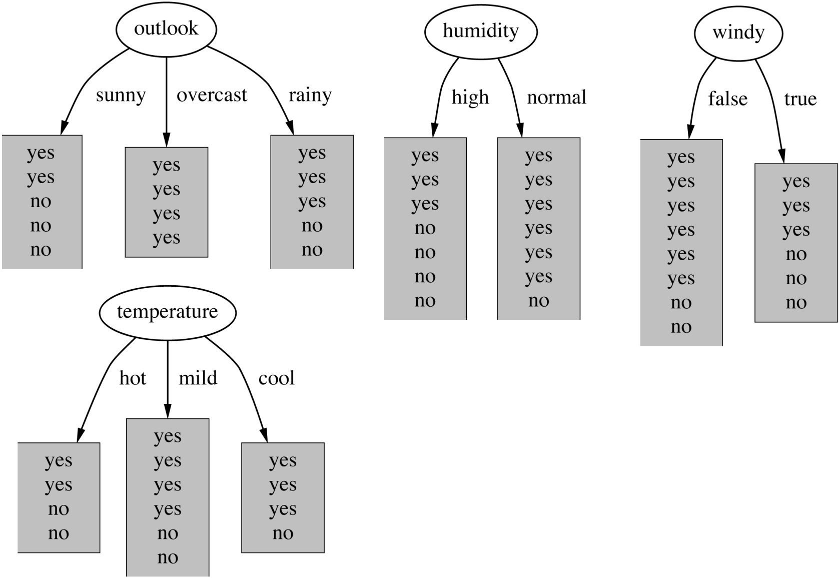
$$SP(t) = -\sum_{z=1}^{Z} \frac{N(t_z)}{N_t} \times \log_2 \left(\frac{N(t_z)}{N_t} \right)$$

- The gain ratio is given by:

$$GR(t) = \frac{\Delta\mathcal{E}(t)}{SP(t)}$$

- Value of attribute decreases as intrinsic information gets larger
- Problem with gain ratio: it may overcompensate
    - May choose an attribute just because its intrinsic information is very low
    - Standard fix: only consider attributes with greater than average information gain

Attribute Selection – Weather Data



outlook

sunny: yes yes no no no

overcast: yes yes yes yes

rainy: yes yes yes no no

humidity

high: yes yes yes no no no no

normal: yes yes yes yes yes yes no

windy

false: yes yes yes yes yes yes no no

true: yes yes yes no no no

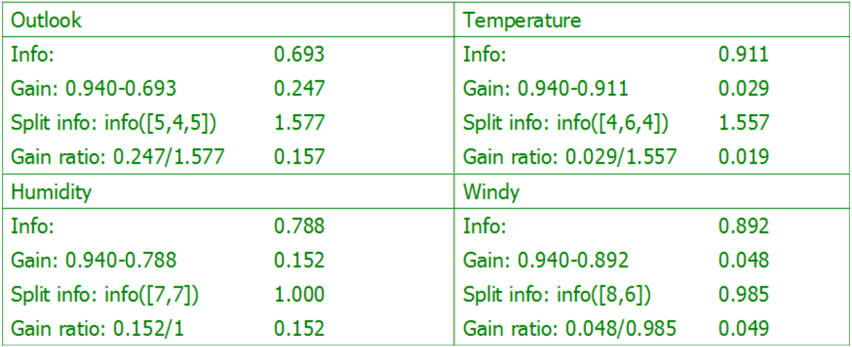
temperature

hot: yes yes no no

mild: yes yes yes yes no no

cool: yes yes yes no

Attribute Selection – Weather Data



Outlook		Temperature	
Info:	0.693	Info:	0.911
Gain: 0.940-0.693	0.247	Gain: 0.940-0.911	0.029
Split info: info([5,4,5])	1.577	Split info: info([4,6,4])	1.557
Gain ratio: 0.247/1.577	0.157	Gain ratio: 0.029/1.557	0.019
Humidity		**Windy**	
Info:	0.788	Info:	0.892
Gain: 0.940-0.788	0.152	Gain: 0.940-0.892	0.048
Split info: info([7,7])	1.000	Split info: info([8,6])	0.985
Gain ratio: 0.152/1	0.152	Gain ratio: 0.048/0.985	0.049

Split Strategies – Numeric Attributes

- Standard method: binary splits in the form

$$X < c_0$$

- How to choose split point c_0?

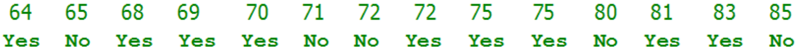
  Straightforward:

  - Evaluate score function for every possible split point of attribute
  - Choose “best” split point
  - Score for best split point is the optimum score for the attribute

- Computationally more demanding

Split Strategies – Numeric Attributes

- Example: Weather data – split on temperature attribute:



64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

  - E.g. temperature < 71.5: yes/4, no/2
    temperature $\geq$ 71.5: yes/5, no/3
  - Info([4,2],[5,3]) = 6/14 info([4,2]) + 8/14 info([5,3])
    = 0.939

- Place split points halfway between values

- Can evaluate all split points in one pass!

Split Strategies – Numeric Attributes

- Binary Split used by: C4.5, CART, LMT, etc
- Some algorithms generate splits with more than two intervals

  CAL5, FACT, REAL (Stern et al, 1998), etc

  Bottom-up approach:

  - Start with small “pure” intervals
  - Merge adjacent intervals if certain conditions hold

Split Strategies – Categorical Attributes

- Ordered categorical attributes may be treated similarly to numeric ones
- For nominal attributes, two approaches:
    - Start a new branch for each value
        - Use gain ratio or other method to avoid bias
    - Binary split of values:
        - For every possible subset S of the attribute, evaluate the score function for the rule in the form

$$X \in A$$

        - Choose the optimum subset
        - Score for best subset is the optimum score for the attribute

Algorithms for Classification Trees – Other refinements

- Stop rules
  - Minimum number of examples
  - Minimum score gain
- Tree Pruning to avoid overfitting
- Treatment of missing data
- Attribute importance

Naïve Bayes

- Concept description under a probabilistic point of view:
  - What is the probability of class y given the attribute vector $\boldsymbol{x}$?
- Classes: $y \in \{1...K\}$
- Bayes' Rule:

$$\Pr(y, \boldsymbol{x}) = \Pr(\boldsymbol{x}) \Pr(y|\boldsymbol{x})$$

$$\Pr(y, \boldsymbol{x}) = \Pr(y) \Pr(\boldsymbol{x}|y)$$

Equaling two Expressions:

$$\Pr(y|\boldsymbol{x}) = \frac{\Pr(y) \Pr(\boldsymbol{x}|y)}{\Pr(\boldsymbol{x})} = \frac{\Pr(y) \Pr(\boldsymbol{x}|y)}{\sum_{k=1}^{K} \Pr(k) \Pr(\boldsymbol{x}|k)}$$

Obs: On above equation, notice that:

$$Pr(\boldsymbol{x}) = \sum_{k=1}^{K} \Pr(k, \boldsymbol{x}) = \sum_{k=1}^{K} \Pr(k) \Pr(\boldsymbol{x}|k)$$

Naïve Bayes

- Bayes Rule:

$$\Pr(y|\boldsymbol{x}) = \frac{\Pr(y)\,\Pr(\boldsymbol{x}|y)}{\sum_{k=1}^{K}\Pr(k)\,\Pr(\boldsymbol{x}|k)}$$

- Interpretation:
  - $\Pr(y)$: *priori* probability (initial probability guess) for y
  - $\Pr(\boldsymbol{x}|y)$: *likelihood* of class y after observation $\boldsymbol{x}$
  - $\Pr(\boldsymbol{x}|y) \equiv \Pr(x_1, x_2, \ldots, x_M|y)$
    where x_j is the observed value of attribute a_j

- Naïve assumption: Attributes are
  - equally important
  - statistically independent (given the class value)
    - I.e., knowing the value of one attribute says nothing about the value of another (if the class is known)

These assumptions are expressed in the following equation:

$$\Pr(x_1, x_2, \ldots, x_M|y) = \Pr(x_1|y)\;\Pr(x_2|y)\;\ldots\;\Pr(x_M|y)$$

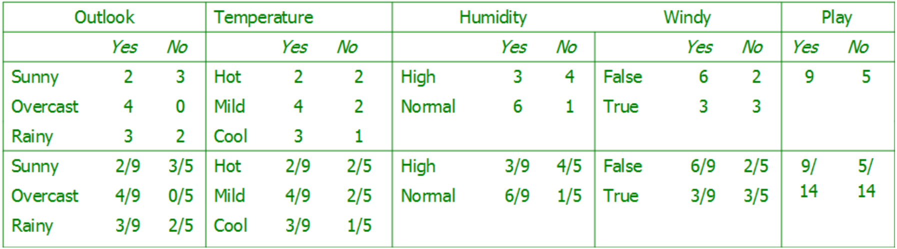
Naïve Bayes

- Bayes Rule:

$$\Pr(y|\boldsymbol{x}) = \frac{\Pr(y)\Pr(\boldsymbol{x}|y)}{\sum_{k=1}^{K}\Pr(k)\Pr(\boldsymbol{x}|k)}$$

- Computing $Pr(y)$:
    - Relative frequency of class y in training set $\mathcal{L}$
- Computing $\Pr(x_j|y)$ for categorical attributes:
    1. Count the absolute frequencies of each attribute value on class y
    2. Normalize the frequencies by the number of instances of class y

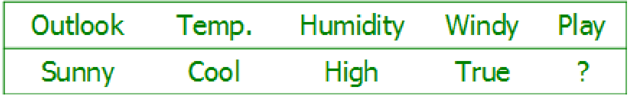
Naïve Bayes – Weather Data



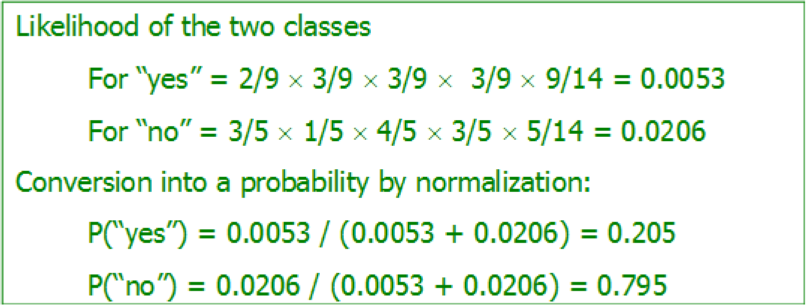
Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

Naïve Bayes – Weather Data

- A new day:



Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?



Likelihood of the two classes

For “yes” = 2/9 × 3/9 × 3/9 × 3/9 × 9/14 = 0.0053

For “no” = 3/5 × 1/5 × 4/5 × 3/5 × 5/14 = 0.0206

Conversion into a probability by normalization:

P(“yes”) = 0.0053 / (0.0053 + 0.0206) = 0.205

P(“no”) = 0.0206 / (0.0053 + 0.0206) = 0.795

Naïve Bayes – Extensions

- What if an attribute value doesn't occur with every class value?

  (e.g. "Humidity = high" for class "yes")

  - Probability will be zero!
  - *A posteriori* probability will also be zero!
    (No matter how likely the other values are!)

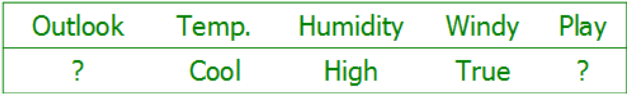
- Remedy: add 1 to the count for every attribute value-class combination (Laplace estimator)

- Result: probabilities will never be zero!

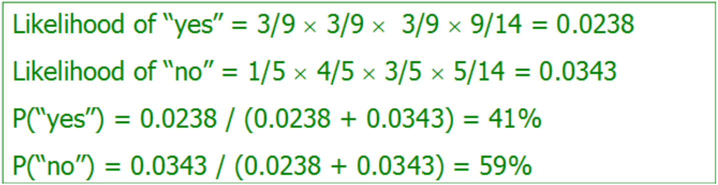
  (also: stabilizes probability estimates)

Naïve Bayes – Missing Values

- Missing data treatment is straightforward:
    - Training: instance is not included in frequency count for attribute value-class combination
    - Classification: attribute will be omitted from calculation
- Example:



Outlook	Temp.	Humidity	Windy	Play
?	Cool	High	True	?



Likelihood of "yes" = 3/9 × 3/9 × 3/9 × 9/14 = 0.0238

Likelihood of "no" = 1/5 × 4/5 × 3/5 × 5/14 = 0.0343

P("yes") = 0.0238 / (0.0238 + 0.0343) = 41%

P("no") = 0.0343 / (0.0238 + 0.0343) = 59%

Naïve Bayes – Numeric Attributes

- Usual assumption: numeric attributes have a normal probability distribution (given the class)

  Parameters for each class k: mean μ_k, standard deviation σ_k

  Probability density function (pdf):

$$f(x|\mu_k, \sigma_k) = \frac{1}{\sqrt{2\pi\sigma_k}} \exp\left(-\frac{(x-\mu_k)^2}{2\sigma_k^2}\right)$$

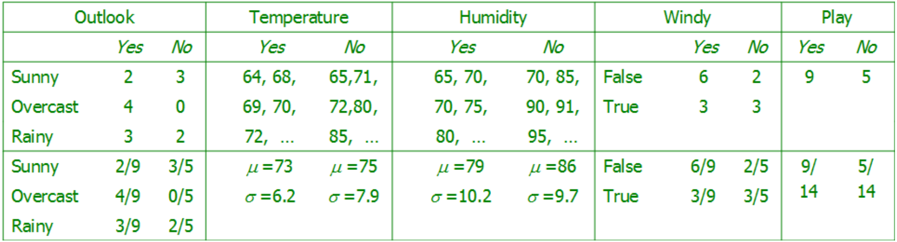
- Estimation of μ_k and σ_c are obtained from instances of class k

$$\mu_k = \frac{1}{N_k}\sum\nolimits_{x_i|y_i=k} x_i$$

$$\sigma_k = \sqrt{\frac{1}{N_k - 1}\sum\nolimits_{x_i|y_i=k}(x_i - \mu_k)^2}$$

N_k: number of instances of class k

Naïve Bayes – Weather Data



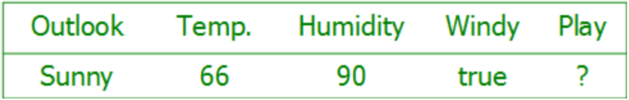
Outlook			Temperature		Humidity		Windy			Play	
	Yes	*No*	*Yes*	*No*	*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	64, 68,	65,71,	65, 70,	70, 85,	False	6	2	9	5
Overcast	4	0	69, 70,	72,80,	70, 75,	90, 91,	True	3	3		
Rainy	3	2	72, …	85, …	80, …	95, …					
Sunny	2/9	3/5	μ =73	μ =75	μ =79	μ =86	False	6/9	2/5	9/ 14	5/ 14
Overcast	4/9	0/5	σ =6.2	σ =7.9	σ =10.2	σ =9.7	True	3/9	3/5		
Rainy	3/9	2/5									

- Example density value:

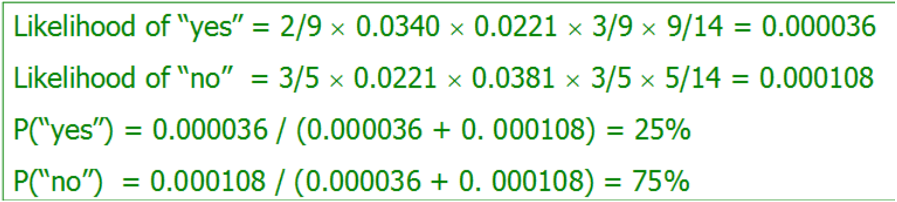
$$f(\text{temperature=66}|\text{yes}) = \frac{1}{\sqrt{2\pi 6.2}} \exp\left(-\frac{(66-73)^2}{26.2^2}\right) = 0.0340$$

Naïve Bayes – Weather Data

- A new day:



Outlook	Temp.	Humidity	Windy	Play
Sunny	66	90	true	?



Likelihood of "yes" = 2/9 × 0.0340 × 0.0221 × 3/9 × 9/14 = 0.000036

Likelihood of "no" = 3/5 × 0.0221 × 0.0381 × 3/5 × 5/14 = 0.000108

P("yes") = 0.000036 / (0.000036 + 0. 000108) = 25%

P("no") = 0.000108 / (0.000036 + 0. 000108) = 75%

- Missing values during training are not included in calculation of mean and standard deviation