Classification

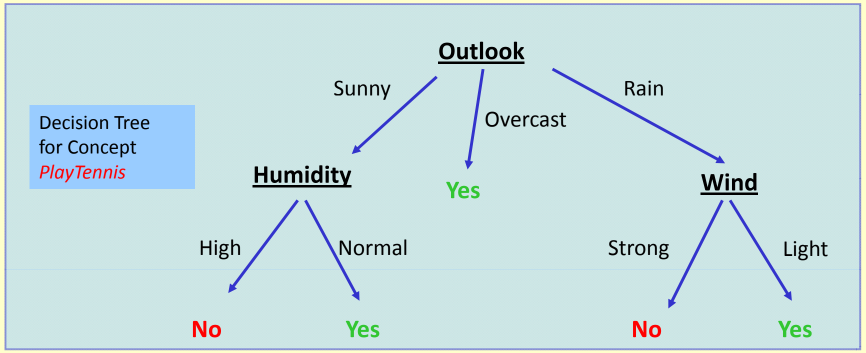
Classification

- What is classification
- Simple methods for classification
- **Classification by decision tree induction**
- Classification evaluation
- Classification in Large Databases

DECISION TREE INDUCTION

Decision trees

- **Internal node** denotes a test on an attribute
- **Branch** corresponds to an attribute value and represents the outcome of a test
- **Leaf node** represents a class label or class distribution
- Each **path** is a conjunction of attribute values



Why decision trees?

Decision trees are especially attractive for a data mining environment for three reasons.

- Due to their intuitive representation, they are easy to assimilate by humans.
- They can be constructed relatively fast compared to other methods.
- The accuracy of decision tree classifiers is comparable or superior to other models.

Decision tree induction

- Decision tree generation consists of two phases
  - **Tree construction**
    - At start, all the training examples are at the root
    - Partition examples recursively based on selected attributes
  - **Tree pruning**
    - Identify and remove branches that reflect noise or outliers
- Use of decision tree: Classifying an unknown sample
  - Test the attribute values of the sample against the decision tree

Choosing good attributes

- **Very important**!

1. If crucial attribute is missing, decision tree won't learn the concept
2. If two training instances have the same representation but belong to different classes, decision trees are inadequate

Name	Cough	Fever	Pain	Diagnosis
Ernie	No	Yes	Throat	Flu
Bert	No	Yes	Throat	Appendicitis

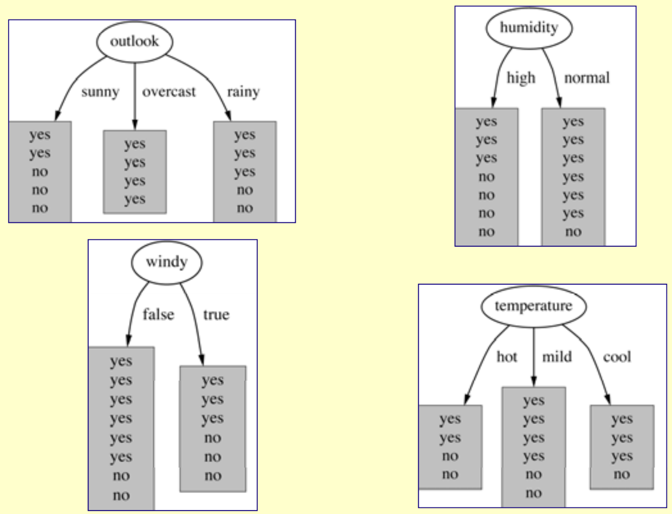
Multiple decision trees

- If attributes are adequate, you can construct a decision tree that correctly classifies all training instances
- Many correct decision trees
- Many algorithms prefer simplest tree (Occam's razor)
  - The principle states that one should not make more assumptions than the minimum needed
  - The simplest tree captures the most generalization and hopefully represents the most essential relationships
  - There are many more 500-node decision trees than 5-node decision trees. Given a set of 20 training examples, we might expect to be able to find many 500-node decision trees consistent with these, whereas we would be more surprised if a 5-nodedecision tree could perfectly fit this data.

Example for *play tennis* concept

Day	Outlook	Temperature	Humidity	Wind	*PlayTennis*?
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No

Which attribute to select?



Choosing the attribute split

- IDEA: evaluate attribute according to its power of separation between near instances
- Values of good attribute should distinguish between near instances from different class and have similar values for near instances from the same class
- Numerical values can be discretized

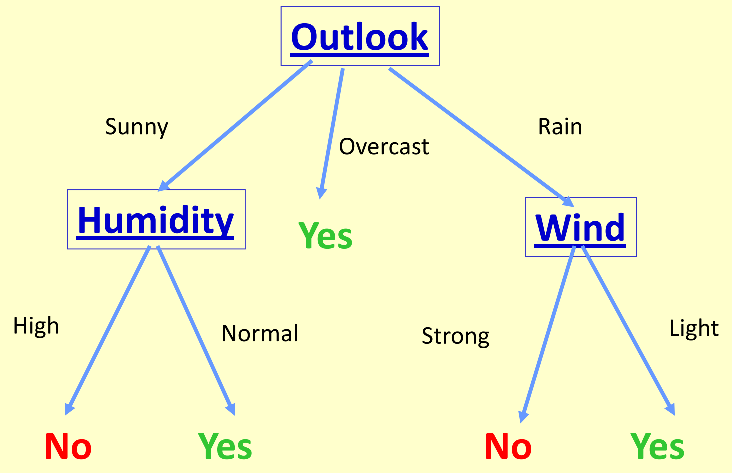
Choosing the attribute

- Many variants:
  - from machine learning: ID3 (Iterative Dichotomizer), C4.5 (Quinlan 86, 93)
  - from statistics: CART (Classification and Regression Trees) (Breiman et al 84)
  - from pattern recognition: CHAID (Chi-squared Automated Interaction Detection) (Magidson 94)
- Main difference: divide (split) criterion
  - Which attribute to test at each node in the tree ? The attribute that is most useful for classifying examples.

Split criterion

- Information gain
  - All attributes are assumed to be categorical (ID3)
  - Can be modified for continuous-valued attributes (C4.5)
- Gini index (CART, IBM IntelligentMiner)
  - All attributes are assumed continuous-valued
  - Assume there exist several possible split values for each attribute
  - Can be modified for categorical attributes

How was this tree built ?



ID3 / C4.5

Basic algorithm: Quinlan's ID3

- create a root node for the tree
- **if** all examples from S belong to the same class Cj
- **then** label the root with Cj
- **else**
  - select the 'most informative' attribute **A** with values **v1, v2,...,vn**
  - divide the training set **S** into **S1,...,Sn** according to **v1,...,vn**
- recursively build subtrees **T1, ... ,Tn** for **S1, ... ,Sn**
- generate decision tree T

Conditions for stopping partitioning

- All samples for a given node belong to the same class
- There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
- There are no samples left

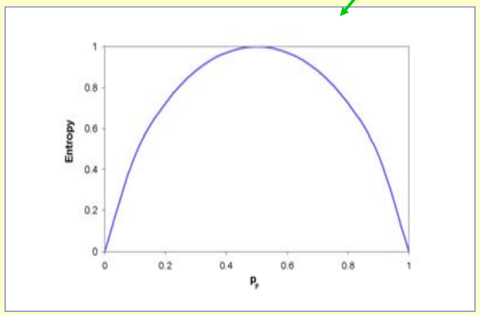
Information gain (ID3)

- Select the attribute with the highest ***information gain***
  - Assume there are two classes, ***P*** and ***N***
  - Let the set of examples *S* contain ***p*** elements of class ***P*** and ***n*** elements of class ***N***
  - The amount of information, needed to decide if an arbitrary example in *S* belongs to ***P*** or ***N*** is defined as

$$Info(p,n) = -\left(\frac{p}{p+n}\log_2\frac{p}{p+n} + \frac{n}{p+n}\log_2\frac{n}{p+n}\right)$$

Entropy

p	1-p	Ent
0.2	0.8	0.72
0.4	0.6	0.97
0.5	0.5	1
0.6	0.4	0.97
0.8	0.2	0.72



$\log_2(2)$

$$Ent = -\sum_{c=1}^{N} p_c \cdot \log_2 p_c$$

$\log_2(3)$

p1	p2	p3	Ent
0.1	0.1	0.8	0.92
0.2	0.2	0.6	1.37
0.1	0.45	0.45	1.37
0.2	0.4	0.4	1.52
0.3	0.3	0.4	1.57
0.33	0.33	0.33	1.58

Entropy

Entropy Ent(S) – **measures the impurity** of a training set S

where p_c is the relative frequency of C_c in S

$$Ent(s_1, \cdots, s_n) = -\sum_{c=1}^{N} p_c \cdot \log_2 p_c$$

PlayTennis?
No
No
Yes
Yes
Yes
No
Yes
No
Yes
Yes
Yes
Yes
Yes
No

$p_{NO} = 5/14$
$p_{YES} = 9/14$

Ent(PlayTennis) = Info(N, P) =

= - (5/14) $\log_2$(5/14) - (9/14) $\log_2$(9/14) = 0.94

Information gain

- An attribute A splits the dataset into subsets
- The entropy of the split is computed as follows

$$Info(A) = \frac{p_1+n_1}{p+n} Info(p_1,n_1) + \frac{p_2+n_2}{p+n} Info(p_2,n_2) + \frac{p_3+n_3}{p+n} Info(p_3,n_3)$$

- The encoding information that would be gained by branching on *A is*

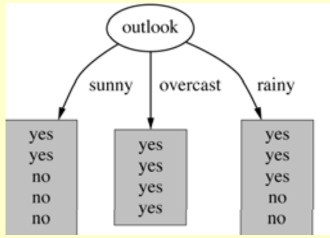
$$Gain(S,A) = Info(N,P) - Info(A)$$

- **Most informative** attribute: **max Gain(S,A)**

Gain for Outlook

PlayTennis?
No
No
Yes
Yes
Yes
No
Yes
No
Yes
Yes
Yes
Yes
Yes
No

E(9+,5-) = 0.940



[2+, 3-] E = 0.970

[4+, 0-] E = 0

[3+, 2-] E = 0.97

Gain(Outlook) = 0.94 - (5/14) x 0.970 - (4/14) x 0 - (5/14) x 0.970= **0.247**

Corresponds to the weighed mean of the entropy in each sub set

Entropy for each spit

- **$I(9^+,5^-)$ = -(9/14)** x **$\log_2$(9/14) - (5/14)** x **$\log_2$(5/14) = 0.940**
- Outlook?
  - Sunny: {D1,D2,D8,D9,D11} [2^+, 3^-] E = 0.970
  - Overcast: {D3,D7,D12,D13} [4^+, 0^-] E = 0
  - Rain: {D4,D5,D6,D10,D14} [3^+, 2^-] E = 0.970
- Humidity?
  - High: [3^+, 4^-] E = 0.985
  - Normal: [6^+, 1^-] E = 0.592
- Wind?
  - Light: [6^+, 2^-] E = 0.811
  - Strong: [3^+, 3^-] E = 1.00
- Temperature? …

Information gain

- S = [9^+,5^-], I(S) = 0.940
- Values(Wind) = { Light, Strong }
- S_{light}= [6^+,2^-] I(S_{light}) = 0.811
- S_{strong} = [3^+,3^-] I(S_{strong}) = 1.0
- **Gain(S,Wind) =** I(S) - (8/14) x I(S_{light}) - (6/14) x I(S_{strong}) =

  = 0.940 - (8/14) x 0.811 - (6/14) x 1.0 = **0.048**

Information gain

- $S = [9^+, 5^-]$, $I(S) = 0.940$
- **Gain(S,Outlook) =** 0.94 - (5/14)x0.970 - (4/14)x0 - (5/14)x0.970= **0.247**
- **Gain(S,Humidity) =** 0.94-(7/14)x0.985 -(7/14)x0.592 = **0.151**
- **Gain(S,Temperature) =** 0.94- (4/14)x1 - (6/14)x0.918 - (4/14)x0.811 = **0.029**

Information gain

Outlook?

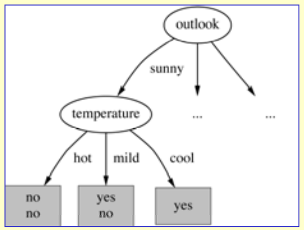
Rain {D4,D5,D6,D10,D14} $[3^+, 2^-]$ E > 0 **???**

Overcast {D3,D7,D12,D13} $[4^+, 0^-]$ E = 0 **OK - assign class Yes**

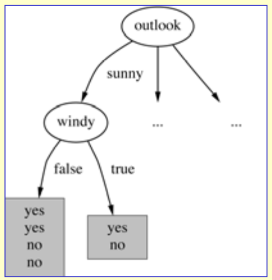
Sunny {D1,D2,D8,D9,D11} $[2^+, 3^-]$ E > 0 **???**

Which attribute should be tested here?

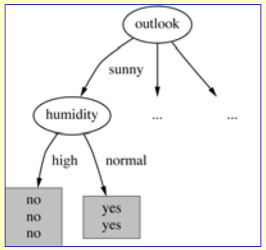
Continuing to split ...



Gain(Temperature) = 0.570



Gain(Wind) = 0.019



Gain(Humidity) = 0.970

Information gain

$E(S_{sunny}) = -(2/5)\log2(2/5) - (3/5)\log2(3/5) = 0.97$

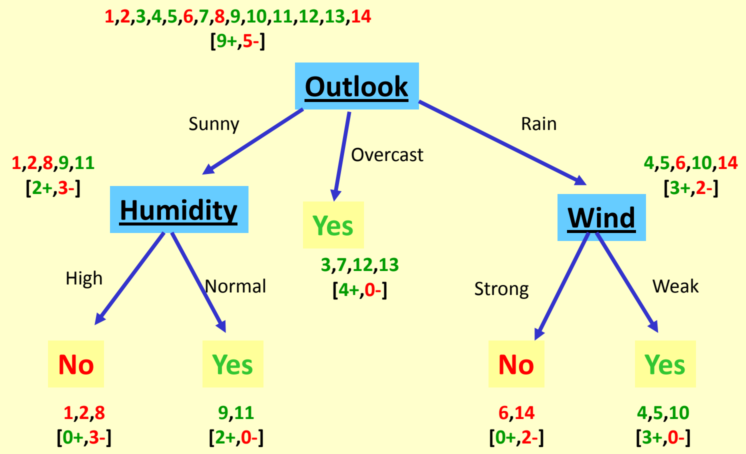
$Gain(S_{sunny}, Humidity) = 0.97-(3/5)0-(2/5)0 = 0.970$ **MAX !**

$Gain(S_{sunny}, Temperature) = 0.97-(2/5)0-(2/5)1-(1/5)0 = 0.570$

$Gain(S_{sunny}, Wind) = 0.97-(2/5)1-(3/5)0.918 = 0.019$

The same has to be done for the outlook(rain) branch.

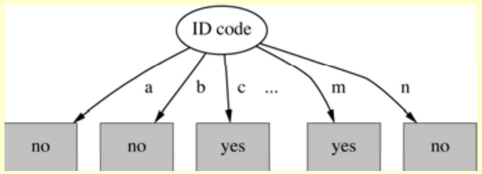
Decision tree for PlayTennis



Problems with information gain

- Problematic: attributes with a large number of values
  - (extreme case: ID code)
- Attributes which have a large number of possible values -> leads to many child nodes.
  - Information gain is 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)

Split for ID code attribute



- Extreme example: compute the information gain of the identification code
- **Gain(S,Day) =** 0.94 - (1/14) x info([0^+, 1^-]) - (1/14) x info([0^+, 1^-]) -... -(1/14) x info([1^+, 0^-]) - (1/14) x info([0^+, 1^-]) = 0.94
- The information gain measure tends to prefer attributes with large numbers of possible values

Gain Ratio

- **Gain ratio:** a modification of the information gain that **reduces its bias** on high-branch attributes
- Gain ratio should be
  - Large when data is evenly spread
  - Small when all data belong to one branch
- Gain ratio takes number and size of branches into account when choosing an attribute
  - It corrects the information gain by taking the *intrinsic information* of a split into account

$$Split(S,A) \equiv -\sum \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}.$$

Gain Ratio

$$\text{Gain Ratio}(S, Day) = \frac{Gain(S, Day)}{Split(S, Day)}$$

$$Split(S, Day) = 14 \times \left(-\frac{1}{14}\log\frac{1}{14}\right) = 3.807$$

$$\text{Gain Ratio}(S, Day) = \frac{0.94}{3.807} = 0.25$$

$$Split(S, Outlook) = \left(-\frac{5}{14}\log\frac{5}{14}\right) \times 2 + \left(-\frac{4}{14}\log\frac{4}{14}\right) = 1.577$$

Outlook?

Sunny: {D1,D2,D8,D9,D11} [2+, 3-] E = 0.970

Overcast: {D3,D7,D12,D13} [4+, 0-] E = 0

Rain: {D4,D5,D6,D10,D14} [3+, 2-] E = 0.970

$$\text{Gain Ratio}(S, Outlook) = \frac{0.247}{1.577} = 0.157$$

$$\text{Gain Ratio}(S, Humidity) = \frac{0.152}{1} = 0.152$$

$$\text{Gain Ratio}(S, Temperature) = \frac{0.029}{1.362} = 0.021$$

$$\text{Gain Ratio}(S, Wind) = \frac{0.048}{0.985} = 0.049$$

More on the gain ratio

- However: "ID code" still has greater gain ratio
  - Standard fix: *ad hoc* test to prevent splitting on that type of attribute
- Outlook still comes out top among the relevant attributes
- Problem with gain ratio: it may overcompensate
  - May choose an attribute just because its intrinsic information is very low
  - ***Standard fix***:
    - First, only consider attributes with greater than average information gain
    - Then, compare them on gain ratio

Another spit criterion

CART: GINI INDEX

- ID3 and CART were invented independently of one another at around the same time
- Both algorithms follow a similar approach for learning decision trees from training examples
  - Greedy, top-down recursive divide and conquer manner

Gini index

- If a data set T contains examples from n classes, the gini index gini(T) is defined as

$$gini(T) = 1 - \sum_{j=1}^{n} p_j^2$$

  - where p_j is the relative frequency of class j in T.
  - gini(T) is minimized if the classes in T are skewed.

Gini index

After splitting T into two subsets T1 and T2 with sizes N1 and N2, the ***gini index*** of the split data is defined as

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

- it corresponds to the weighted average of each branch index
- the attribute providing smallest $gini_{split}(T)$ is chosen to split the node.

Example of Gini split index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
	Taxable Income									
Sorted Values	60	70	75	85	90	95	100	120	125	220

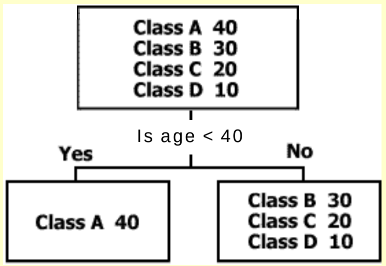
Split Positions	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		***0.300***		0.343		0.375		0.400		0.420	

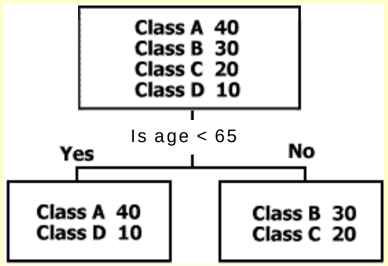
$$gini = \frac{0}{10} \cdot 0 + \frac{10}{10} \cdot \left(1 - \left(\frac{3}{10}\right)^2 - \left(\frac{7}{10}\right)^2\right) = 0.42$$

$$gini = \frac{6}{10} \cdot \left(1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2\right) + \frac{4}{10} \cdot \left(1 - \left(\frac{0}{4}\right)^2 - \left(\frac{4}{4}\right)^2\right) = 0.6 \cdot 0.5 + 0.4 \cdot 0 = 0.30$$

Entropy vs. Gini (on continuous attributes)

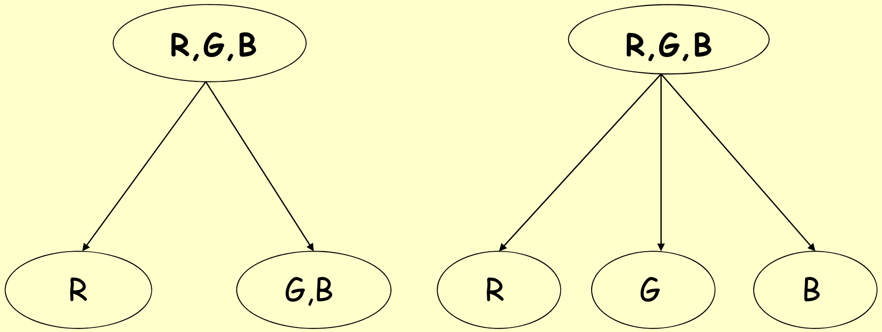
- Gini tends to isolate the largest class from all other classes
- Entropy tends to find groups of classes that add up to 50% of the data





CART

ID 3



C 4.5

c4.5

- It is a benchmark algorithm
- C4.5 innovations (Quinlan):
  - permit numeric attributes
  - deal sensibly with missing values
  - pruning to deal with for noisy data
- C4.5 - one of best-known and most widely-used learning algorithms
  - Last research version: C4.8, implemented in Weka as J4.8 (Java)
  - Commercial successor: C5.0 (available from Rulequest)

Numeric attributes

- Standard method: binary splits
  - E.g. temp < 45
- Unlike nominal attributes, every attribute has many possible split points
- Solution is straightforward extension (see slides on data pre-processing):
  - Evaluate info gain (or other measure) for every possible split point of attribute
  - Choose “best” split point
  - Info gain for best split point is info gain for attribute
- Computationally more demanding

Binary *vs.* multi-way splits

- Splitting (multi-way) on a nominal attribute exhausts all information in that attribute
  - Nominal attribute is tested (at most) once on any path in the tree
- Not so for binary splits on numeric attributes!
  - Numeric attribute may be tested several times along a path in the tree
- Disadvantage: tree is hard to read
- Remedy:
  - pre-discretize numeric attributes, *or*
  - use multi-way splits instead of binary ones

Missing as a separate value

- Missing value denoted as "?" in C4.X
- Simple idea: treat missing as a separate value
- Q: When this is not appropriate?
- A: When values are missing due to different reasons
  - Example : field **IsPregnant**=missing for a male patient should be treated differently (no) than for a female patient of age 25 (unknown)

Missing values - advanced

Split instances with missing values into pieces

- A piece going down a branch receives a weight proportional to the popularity of the branch
- weights sum to 1

- Info gain works with fractional instances
  - use sums of weights instead of counts
- During classification, split the instance into pieces in the same way
  - Merge probability distribution using weights

References

- Jiawei Han and Micheline Kamber, "Data Mining: Concepts and Techniques", 2000
- Ian H. Witten, Eibe Frank, "Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations", 1999
- Tom M. Mitchell, "Machine Learning", 1997
- J. Shafer, R. Agrawal, and M. Mehta. "SPRINT: A scalable parallel classifier for data mining". In VLDB'96, pp. 544-555,
- J. Gehrke, R. Ramakrishnan, V. Ganti. "RainForest: A framework for fast decision tree construction of large datasets." In VLDB'98, pp. 416-427
- Robert Holt "Cost-Sensitive Classifier Evaluation" (ppt slides)

Thank you !!!