Classification: Basic Concepts and Techniques

XX-161-A-21 – Big Data Analytics

Dr. Jim Scrofani
jwscrofa@nps.edu

Topics at a Glance

- Learning and the Classification Problem
- Examples of Classification Problems
- General Approaches
- Specific Approaches
  - Decision Tree
  - Zero Rule (Zero R)
  - One Rule (One R)

Learning Taxonomy

- Supervised Learning
  - Used to estimate an unknown (input, output) mapping from known (input, output) samples
  - "Supervised" – output values for training samples are known, i.e., provided by a "teacher"
  - Common approaches
    - Classification
    - Regression
- Unsupervised Learning
  - Used to estimate an unknown (input, output) mapping from input samples only
  - There is no teacher
  - Common approaches
    - Distribution estimation
    - Discover structure in the data ("clusters")

3

Classification: Definition

- Given a collection of records (training set)
  - Each record is characterized by a tuple ($\mathbf{x}$, y), where $\mathbf{x}$ is the attribute set and y is the class label
    - $\mathbf{x}$: attribute, feature, predictor, independent variable, input
    - y: class, response, dependent variable, output
- Task:
  - Learn a model that maps each attribute set $\mathbf{x}$ into one of the predefined class labels y

$$\mathbf{x} \mapsto y, \text{ where } y \in \{0, 1, \ldots, N\}$$

4

Classification Techniques

- Base Classifiers
  - Zero R
  - One R
  - *Decision Tree-based Methods*
  - Rule-based Methods
  - Nearest-neighbor
  - Neural Networks
  - Deep Learning
  - Naïve Bayes and Bayesian Belief Networks
  - Support Vector Machines
- Ensemble Classifiers
  - Boosting
  - Bagging
  - *Random Forests (details in Jupyter Guided Exercise)*

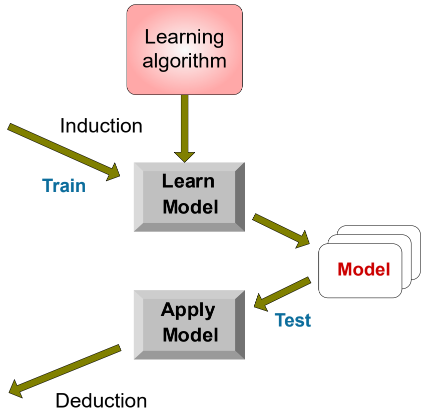
Examples of Classification Task

Task	Attribute set, x	Class label, y
Categorizing email messages	Features extracted from email message header and content	spam or non-spam
Identifying tumor cells	Features extracted from MRI scans	malignant or benign cells
Cataloging galaxies	Features extracted from telescope images	Elliptical, spiral, or irregular-shaped galaxies

General Approach for Building a Classification Model

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

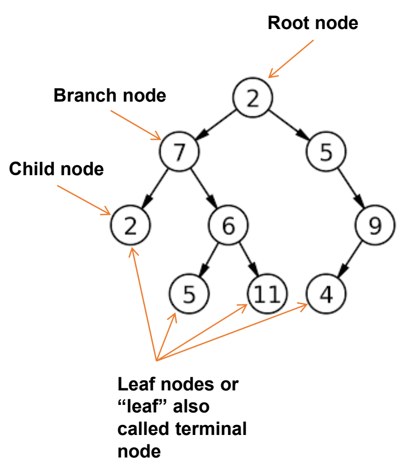


Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Decision Tree Classifier

- Predictive model that uses tree structure to distinguish observations from one another by class
- *Leaves* represent class labels
- *Branches* represent conjunctions of features that lead to class labels

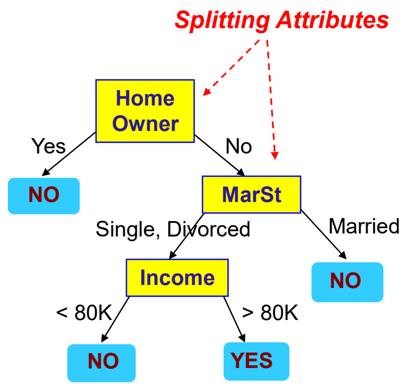


Example of a Decision Tree

categorical categorical continuous class

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data



Model: Decision Tree

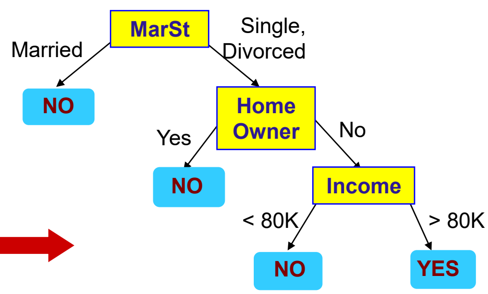
9

Example of a Decision Tree

categorical categorical continuous class

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data



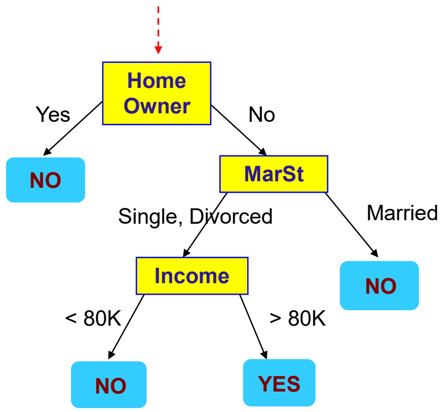
Model: Decision Tree

There can be more than one tree that fits the same data!

10

Apply Model to Test Data

Start from the root node of tree



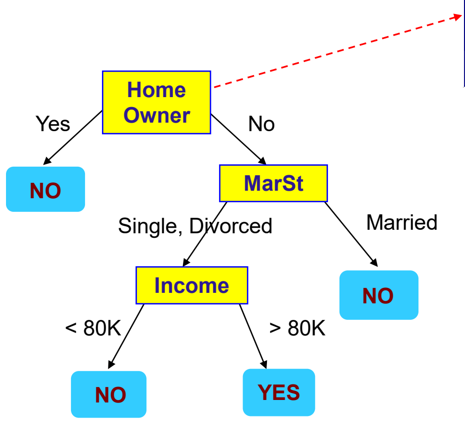
Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Apply Model to Test Data

Test Data

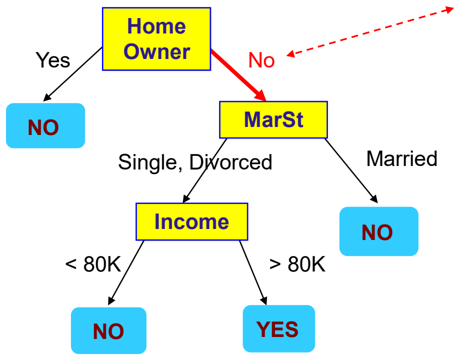
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data

Test Data

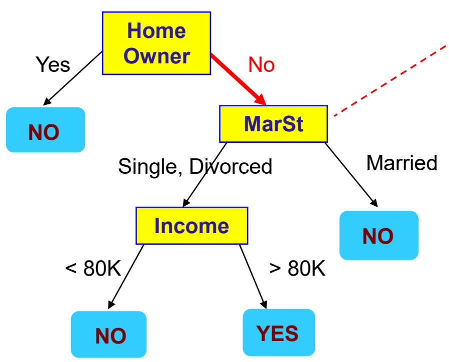
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data

Test Data

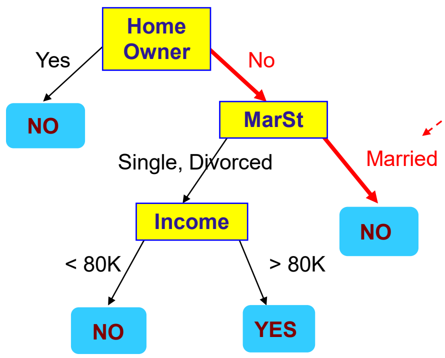
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data

Test Data

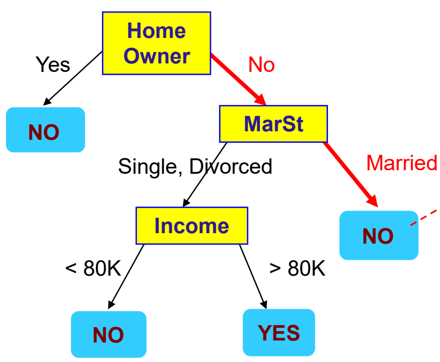
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data

Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



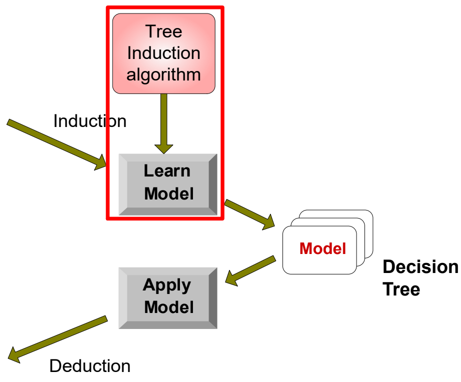
Assign Defaulted Borrower to "No"

Our model "classifies" this borrower as low risk

Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Decision Tree Induction

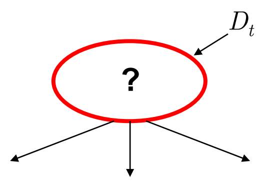
- Many Algorithms:
  - Hunt's Algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ, SPRINT

General Structure of Hunt's Algorithm

- Let D_t be the set of training records that reach a node t

- General Procedure:
  - If D_t contains records that all belong to the same class y_t, then t is a leaf node labeled as y_t
  - If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets
  - Recursively apply the procedure to each subset

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



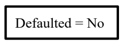
Hunt's Algorithm

Defaulted = No

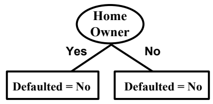
(a)

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Hunt’s Algorithm



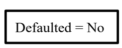
(a)



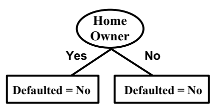
(b)

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

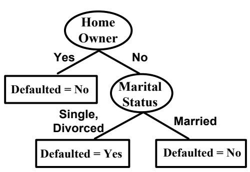
Hunt’s Algorithm



(a)



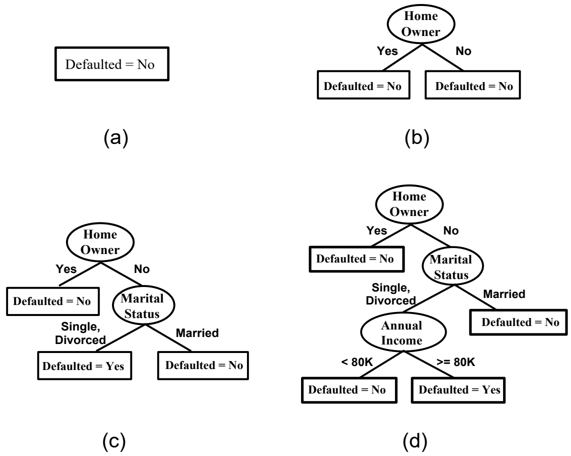
(b)



(c)

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Hunt’s Algorithm



ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Design Issues of Decision Tree Induction

- How should training records be split?
  - Method for specifying test condition
    - depending on attribute types
  - Measure for evaluating the goodness of a test condition

- How should the splitting procedure stop?
  - Stop splitting if all the records belong to the same class or have identical attribute values
  - Early termination

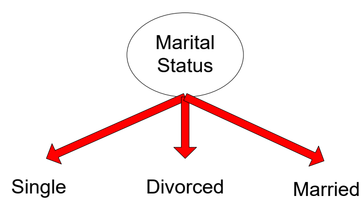
Methods for Expressing Test Conditions

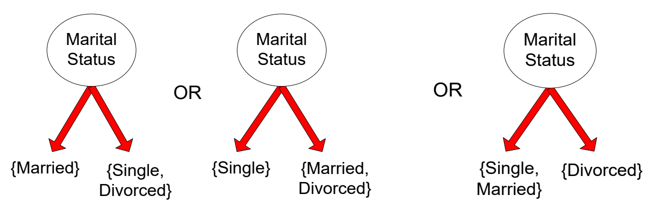
- Depends on attribute types
  - Binary
  - Nominal
  - Ordinal
  - Continuous

- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Test Condition for Nominal Attributes

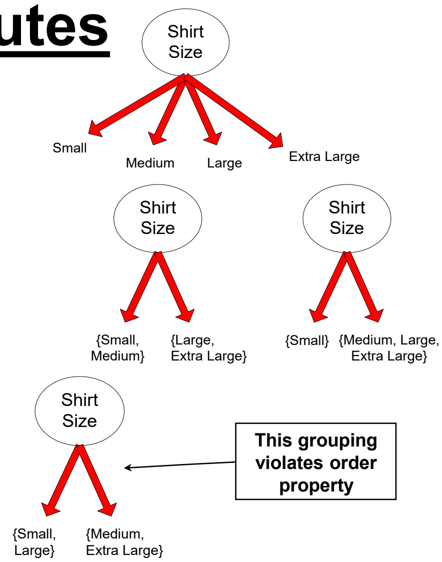
- Multi-way split:
  - Use as many partitions as distinct values
- Binary split:
  - Divides values into two subsets



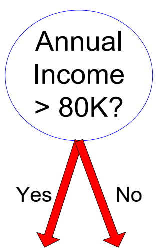


Test Condition for Ordinal Attributes

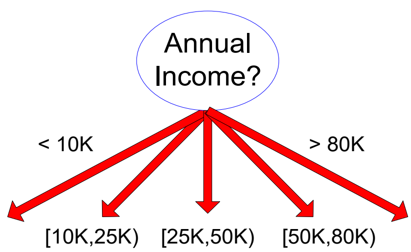
- Multi-way split:
  - Use as many partitions as distinct values
- Binary split:
  - Divides values into two subsets
  - Preserve order property among attribute values



Test Condition for Continuous Attributes



(i) Binary split



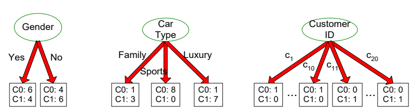
(ii) Multi-way split

Splitting Based on Continuous Attributes

- Different ways of handling
  - Discretization to form an ordinal categorical attribute
    - Ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering
    - Static – discretize once at the beginning
    - Dynamic – repeat at each node
  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - Consider all possible splits and finds the best cut
    - Can be more compute intensive

How to Determine the Best Split

**Before Splitting: 10 records of class 0,
10 records of class 1**



Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

Which test condition is the best?

How to Determine the Best Split

- Greedy approach:
  - Nodes with *purer* class distribution are preferred
- Need a measure of node *impurity*:

C0: 5 C1: 5	C0: 9 C1: 1
High degree of impurity	**Low degree of impurity**

Measures of Node Impurity

- Gini Index: $$\text{GINI}(t) = 1 - \sum_j [p(j|t)]^2$$
- Entropy: $$\text{Entropy}(t) = -\sum_j p(j|t) \log p(j|t)$$
- Misclassification error: $$\text{Error}(t) = 1 - \max_i p(i|t)$$

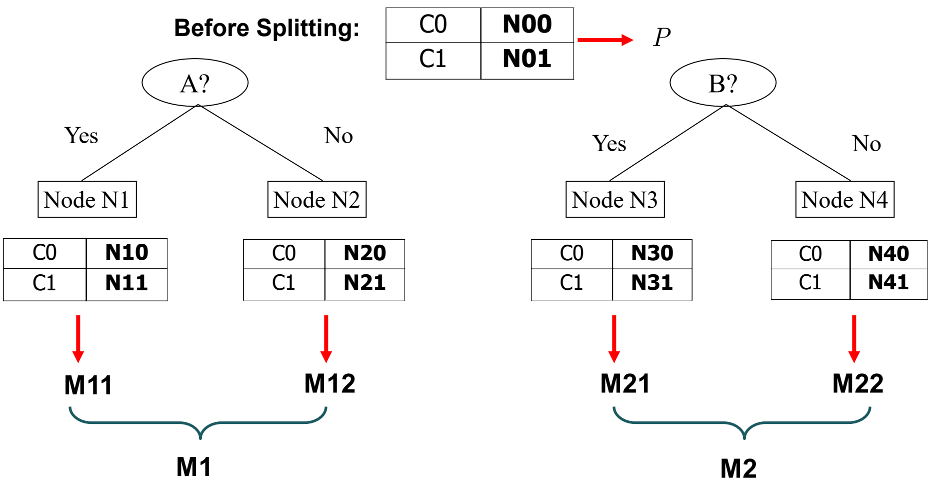
Finding the Best Split

1. Compute impurity measure P before splitting
2. Compute impurity measure M after splitting
   – Compute impurity measure of each child node
   – M is the weighted impurity of children
3. Choose the attribute test condition that produces the highest gain

$$\boxed{\text{Gain} = P - M}$$

or equivalently, lowest impurity measure after splitting M

Finding the Best Split



$$\boxed{\text{Gain} = P - M1 \text{ vs. } P - M2}$$

Measure of Impurity: GINI

- Gini Index for a given node t :

$$\text{GINI}(t) = 1 - \sum_j [p(j|t)]^2$$

(NOTE: $p(j|t)$ is the relative frequency of class j at node t).

  - Maximum of 1 - $1/n_c$ when records are equally distributed among all classes, implying least interesting information
  - Minimum of 0 when all records belong to one class, implying most interesting information

Measure of Impurity: GINI

- Gini Index for a given node t :

$$\text{GINI}(t) = 1 - \sum_j [p(j|t)]^2$$

- For 2-class problem $(p, 1-p)$:

$$\text{GINI} = 1 - p^2 - (1-p)^2 = 2p(1-p)$$

C1	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

Computing GINI Index of a Single Node

$$\text{GINI}(t) = 1 - \sum_j [p(j|t)]^2$$

C1	**0**
C2	**6**

$P(C1) = 0/6 = 0 \qquad P(C2) = 6/6 = 1$

$\text{GINI} = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$

C1	**1**
C2	**5**

$P(C1) = 1/6 \qquad P(C2) = 5/6$

$\text{GINI} = 1 - (1/6)^2 - (5/6)^2 = 0.278$

C1	**2**
C2	**4**

$P(C1) = 2/6 \qquad P(C2) = 4/6$

$\text{GINI} = 1 - (2/6)^2 - (4/6)^2 = 0.444$

Computing GINI Index for a Collection of Nodes

- When a node p is split into k partitions (children)

$$\boxed{\text{GINI}_{\text{split}} = \sum_{i=1}^{k} \frac{n_i}{n} \text{GINI}(i)}$$

where, n_i = number of records at child i,

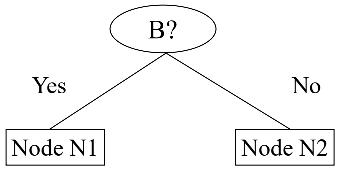
n = number of records at parent node p.

- Choose the attribute that minimizes weighted average GINI index of the children

- GINI index is used in decision tree algorithms such as CART, SLIQ, SPRINT

Binary Attributes: Computing GINI Index

- Splits into two partitions
- Effect of Weighing partitions:
  - Larger and Purer Partitions are sought



	Parent
C1	**7**
C2	**5**
GINI = 0.486	

GINI(N1)
$= 1 - (5/6)^2 - (1/6)^2$
$= 0.278$

GINI(N2)
$= 1 - (2/6)^2 - (4/6)^2$
$= 0.444$

	N1	N2
C1	**5**	**2**
C2	**1**	**4**
GINI=0.361		

Weighted GINI(N1, N2) $=$
$6/12 \times 0.278 +$
$6/12 \times 0.444 =$
0.361

Gain $= 0.486 - 0.361 = 0.125$

Categorical Attributes: Computing GINI Index

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7
GINI	**0.163**		

Two-way split
(find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3
GINI	**0.468**	

	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
GINI	**0.167**	

Which of these is the best?

Continuous Attributes: Computing GINI Index

- Use Binary Decisions based on one value
- Several Choices for the splitting value
  - Number of possible splitting values = Number of distinct values
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions,

    $\mathrm{A} < v$ and $\mathrm{A} \geq v$
- Simple method to choose best v
  - For each v, scan the database to gather count matrix and compute its GINI index
  - Computationally Inefficient! Repetition of work

ID	Home Owner	Marital Status	Annual Income	Defaulted
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Annual Income ?

	≤ 80	> 80
Defaulted Yes	0	3
Defaulted No	3	4

Continuous Attributes: Computing GINI Index...

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing GINI index
  - Choose the split position that has the least GINI index

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

Continuous Attributes: Computing GINI Index...

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing GINI index
  - Choose the split position that has the least GINI index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
Annual 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
	<= | >	<= | >	<= | >	<= | >	<= | >	<= | >	<= | >	<= | >	<= | >	<= | >	<= | >

Continuous Attributes: Computing GINI Index...

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing GINI index
  - Choose the split position that has the least GINI index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
Annual 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							
No				3 | 4							
Gini				0.343							

Continuous Attributes: Computing GINI Index...

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing GINI index
  - Choose the split position that has the least GINI index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
	Annual 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	1	2												
No							3	4	3	4												
Gini							0.343		0.417													

Continuous Attributes: Computing GINI Index...

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing GINI index
  - Choose the split position that has the least GINI index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
	Annual 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	

Continuous Attributes: Computing GINI Index...

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing GINI index
  - Choose the split position that has the least GINI index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
Annual Income	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	

Best split position corresponds to the one that produces the lowest Gini index, which occurs at $97,000.

Measure of Impurity: Entropy

- Entropy at a given node t:

$$\boxed{\text{Entropy}(t) = -\sum_j p(j|t)\log p(j|t)}$$

(NOTE: $p(j|t)$ is the relative frequency of class j at node t).

  - Maximum is $\log n_c$ when records are equally distributed among all classes implying least information
  - Minimum is 0.0 when all records belong to one class, implying most information

– Entropy based computations are similar to the GINI index computations

Computing Entropy of a Single Node

$$\text{Entropy}(t) = -\sum_j p(j|t) \log p(j|t)$$

C1	**0**
C2	**6**

$P(C1) = 0/6 = 0 \qquad P(C2) = 6/6 = 1$

$\text{Entropy} = -0 \log_2 0 - 1 \log_2 1 = -0 - 0 = 0$

C1	**1**
C2	**5**

$P(C1) = 1/6 \qquad P(C2) = 5/6$

$\text{Entropy} = -(1/6)\log_2(1/6) - (5/6)\log_2(1/6) = 0.65$

C1	**2**
C2	**4**

$P(C1) = 2/6 \qquad P(C2) = 4/6$

$\text{Entropy} = -(2/6)\log_2(2/6) - (4/6)\log_2(4/6) = 0.92$

Computing Information Gain After Splitting

- Information Gain:

$$\text{GAIN}_{\text{split}} = \text{Entropy}(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} \text{Entropy}(i)\right)$$

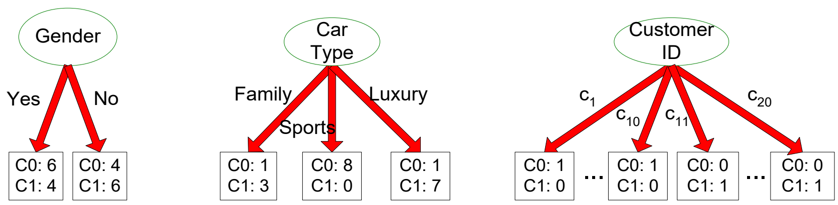
where parent node p is split into k partitions;
n_i = number of records in partition i.

  – Choose the split that achieves most reduction (maximizes GAIN)

  – Used in the ID3 and C4.5 decision tree algorithms

Problem with large number of partitions

- Node impurity measures tend to prefer splits that result in large number of partitions, each being small but pure



  – Customer ID has highest information gain because entropy for all the children is zero

Gain Ratio

- Gain Ratio:

$$\text{GainRATIO}_{\text{split}} = \frac{\text{GAIN}_{\text{split}}}{\text{SplitINFO}} \qquad \text{SplitINFO} = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

where parent node p is split into k partitions;
n_i = number of records in partition i.

  – Adjusts Information Gain by the entropy of the partitioning (SplitINFO).
    - Higher entropy partitioning (large number of small partitions) is penalized!
  – Used in C4.5 algorithm
  – Designed to overcome the disadvantage of Information Gain

Gain Ratio

- Gain Ratio:

$$\text{GainRATIO}_{\text{split}} = \frac{\text{GAIN}_{\text{split}}}{\text{SplitINFO}}$$

$$\text{SplitINFO} = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

	CarType		
	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7
Gini	0.163		

SplitINFO = 1.52

	CarType	
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3
Gini	0.468	

SplitINFO = 0.72

	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Gini	0.167	

SplitINFO = 0.97

Measure of Impurity: Classification Error

- Classification error at a node t :

$$\text{Error}(t) = 1 - \max_i p(i|t)$$

- Maximum is $1 - 1/n_c$ when records are equally distributed among all classes, implying least interesting information
- Minimum is 0 when all records belong to one class, implying most interesting information

Computing Error of a Single Node

$$\text{Error}(t) = 1 - \max_i p(i|t)$$

C1	**0**
C2	**6**

$P(C1) = 0/6 = 0 \qquad P(C2) = 6/6 = 1$

$\text{Error} = 1 - \max(0, 1) = 1 - 1 = 0$

C1	**1**
C2	**5**

$P(C1) = 1/6 \qquad P(C2) = 5/6$

$\text{Error} = 1 - \max(1/6, 5/6) = 1 - 5/6 = 1/6$

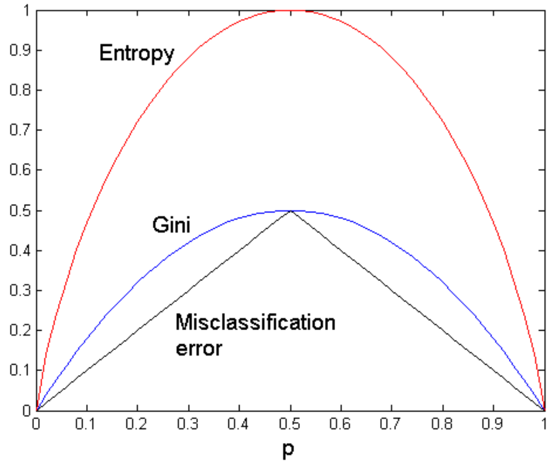
C1	**2**
C2	**4**

$P(C1) = 2/6 \qquad P(C2) = 4/6$

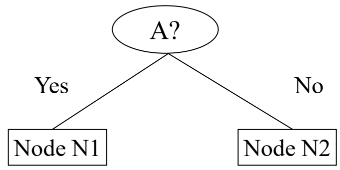
$\text{Error} = 1 - \max(2/6, 4/6) = 1 - 4/6 = 1/3$

Comparison among Impurity Measures

For a 2-class problem:



Misclassification Error vs Gini Index



	Parent
C1	**7**
C2	**3**
Gini = 0.42	

GINI(N1)

$= 1 - (3/3)^2 - (0/3)^2$

$= 0$

GINI(N2)

$= 1 - (4/7)^2 - (3/7)^2$

$= 0.489$

	N1	N2
C1	**3**	**4**
C2	**0**	**3**
Gini=0.342		

GINI(Children) $=$

$3/10 \times 0 +$

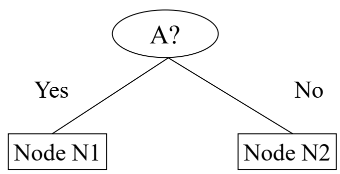
$7/10 \times 0.489 =$

0.342

GINI improves but

error remains the same !!

Misclassification Error vs Gini Index



	Parent
C1	**7**
C2	**3**
Gini = 0.42	

	N1	N2
C1	**3**	**4**
C2	**0**	**3**
Gini=0.342		

	N1	N2
C1	**3**	**4**
C2	**1**	**2**
Gini=0.416		

Misclassification error for all three cases $= 0.3!$

Decision Tree-Based Classification

- Advantages:
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Robust to noise (especially when methods to avoid overfitting are employed)
  - Can easily handle redundant or irrelevant attributes (unless the attributes are interacting)
- Disadvantages:
  - Space of possible decision trees is exponentially large. Greedy approaches are often unable to find the best tree
  - Does not take into account interactions between attributes
  - Each decision boundary involves only a single attribute

Zero Rule Classifier

- Zero Rule → Simplest classification method
- Relies <u>solely on target</u> (class)
- Ignores all predictors (features)
- Based on the <u>majority class</u> (use frequency table)
- Used as a *benchmark* – baseline performance assessment

Benchmark measure

Zero R Classifier

Observation → Zero R Classifier → "majority class"

Class

Outlook	Temp	Humidity	Windy	Play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Training Set

Majority class

Play?	
Yes	No
9	5

With a Zero R Classifier, the decision is always the majority class

Zero R Classifier

For new observations, the classifier always assigns (predicts) YES

TP

		Play?	
		Actual	
		Yes	No
Zero R Classifier (Predicted)	Yes	9	5
	No	0	0

TN

Confusion Matrix

- True positive (TP): classifier predicts YES, when actual is YES
- True negative (TN): classifier predicts NO, when actual is NO

__A confusion matrix__ allows visualization of the performance of a supervised learning algorithm.

Each row represents instances in a predicted class while each column represents instances in the actual class.

Name stems from the fact that it makes it easy to see whether the system is confusing two classes (commonly mislabeling one as another).

https://en.wikipedia.org/wiki/Confusion_matrix

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total}}$$
$$= \frac{9}{14} = 0.64$$

One Rule Classifier

- Another simple classification algorithm
- Generates one rule for each feature
- Then selects rule with smallest total error
  →This is the "One Rule"
- Based on frequency table for each feature
- Surprisingly accurate vs. state-of-the-art classifiers
- Easily interpreted

One R Classifier

Algorithm

- For each feature,
  - For each value of that feature, make a rule as follows;
    - Count how often each value of the Class appears
    - Find the most frequent Class
    - Make the rule assign that Class to this value of the feature
  - Calculate the total error of the rules of each feature
- Choose the feature with the smallest total error

If a feature is numerical → categorical

One R Classifier

Feature 1

		Play? Yes	Play? No
Outlook	Sunny	2	3
	Overcast	4	0
	Rainy	3	2

Feature 2

		Play? Yes	Play? No
Temp	Hot	2	2
	Mild	4	2
	Cool	3	1

Feature 3

		Play? Yes	Play? No
Humidity	High	3	4
	Normal	6	1

Feature 4

		Play? Yes	Play? No
Windy	FALSE	6	2
	TRUE	3	3

Outlook	Temp	Humidity	Windy	Play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

- For each feature,
  - For each value of that feature, make a rule as follows;
    - Count how often each value of the Class appears
    - Find the most frequent Class
    - Make the rule assign that Class to this value of the feature
  - Calculate the total error of the rules of each feature
- Choose the feature with the smallest total error

65

65

One R Classifier

Feature 1

		Play? Yes	Play? No
Outlook	Sunny	2	3
	Overcast	4	0
	Rainy	3	2

Frequency Table

Feature = Outlook

- For each feature,
  - For each value of that feature, make a rule as follows;
    - Count how often each value of the Class appears
    - Find the most frequent Class
    - Make the rule assign that Class to this value of the feature
  - Calculate the total error of the rules of each feature
- Choose the feature with the smallest total error

If Sunny → NO (3 NO, 2 YES)
If Overcast → YES (4 YES, 0 NO)
If Rainy → YES (3 YES, 2 NO)

66

One R Classifier

Feature = Temperature

- For each feature,
  - For each value of that feature, make a rule as follows;
    - Count how often each value of the Class appears
    - Find the most frequent Class
    - Make the rule assign that Class to this value of the feature
  - Calculate the total error of the rules of each feature
- Choose the feature with the smallest total error

Feature 2

		Play? Yes	Play? No
Temp	Hot	2	2
	Mild	4	2
	Cool	3	1

Frequency Table

If Hot → YES or NO (2 YES, 2 NO)
If Mild → YES (4 YES, 2 NO)
If Cool → YES (3 YES, 1 NO)

One R Classifier

Feature = Humidity

- For each feature,
  - For each value of that feature, make a rule as follows;
    - Count how often each value of the Class appears
    - Find the most frequent Class
    - Make the rule assign that Class to this value of the feature
  - Calculate the total error of the rules of each feature
- Choose the feature with the smallest total error

Feature 3

		Play? Yes	Play? No
Humidity	High	3	4
	Normal	6	1

Frequency Table

If High → NO (3 YES, 4 NO)
If Normal → YES (6 YES, 1 NO)

One R Classifier

Feature = Windy

- For each feature,
  - For each value of that feature, make a rule as follows;
    - Count how often each value of the Class appears
    - Find the most frequent Class
    - Make the rule assign that Class to this value of the feature
  - Calculate the total error of the rules of each feature
- Choose the feature with the smallest total error

If FALSE → YES (6 YES, 2 NO)
If TRUE → YES or NO (3 YES, 3 NO)

Feature 4

		Play?	
		Yes	No
Windy	FALSE	6	2
	TRUE	3	3

Frequency Table

One R Classifier

Feature = Outlook

If Sunny → NO
If Overcast → YES
If Rainy → YES

Outlook	Temp	Humidity	Windy	Play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

		Play?	
		Actual	
		Yes	No
Sunny (Predicted)	Yes	0	0
	No	2	3

Confusion Matrix

- For each feature,
  - For each value of that feature, make a rule as follows;
    - Count how often each value of the Class appears
    - Find the most frequent Class
    - Make the rule assign that Class to this value of the feature
  - Calculate the total error of the rules of each feature
- Choose the feature with the smallest total error

One R Classifier

Feature = Outlook

Outlook	Temp	Humidity	Windy	Play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

If Sunny → NO
If Overcast → YES
If Rainy → YES

		Play? Actual: Yes	Play? Actual: No
Sunny (Predicted)	Yes	0	0
	No	2	3

		Play? Actual: Yes	Play? Actual: No
Overcast (Predicted)	Yes	4	0
	No	0	0

One R Classifier

Feature = Outlook

Outlook	Temp	Humidity	Windy	Play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

If Sunny → NO
If Overcast → YES
If Rainy → YES

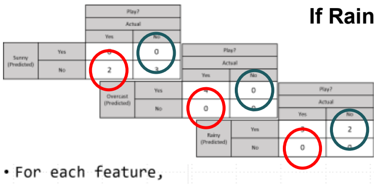
		Play? Actual: Yes	Play? Actual: No
Sunny (Predicted)	Yes	0	0
	No	2	3

		Play? Actual: Yes	Play? Actual: No
Overcast (Predicted)	Yes	4	0
	No	0	0

		Play? Actual: Yes	Play? Actual: No
Rainy (Predicted)	Yes	3	2
	No	0	0

One R Classifier

Feature = Outlook



If Sunny → NO
If Overcast → YES
If Rainy → YES

$$\text{Total Error} = \frac{FP + FN}{\text{Total}} = \frac{4}{14}$$

- False positive (FP): classifier predicts YES, when actual is NO
- False negative (FN): classifier predicts NO, when actual is TRUE

- For each feature,
  - For each value of that feature, make a rule as follows;
    - Count how often each value of the Class appears
    - Find the most frequent Class
    - Make the rule assign that Class to this value of the feature
  - Calculate the total error of the rules of each feature
- Choose the feature with the smallest total error

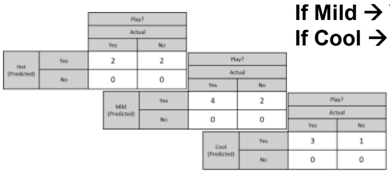
		Play?	
		Actual	
		Yes	No
Outlook (Predicted)	Yes	7	2
	No	2	3

One R Classifier

Feature = Temp

Outlook	Temp	Humidity	Windy	Play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

If Hot → YES or NO
If Mild → YES
If Cool → YES



- For each feature,
  - For each value of that feature, make a rule as follows;
    - Count how often each value of the Class appears
    - Find the most frequent Class
    - Make the rule assign that Class to this value of the feature
  - Calculate the total error of the rules of each feature
- Choose the feature with the smallest total error

		Play?	
		Actual	
		Yes	No
Temp (Predicted)	Yes	9	5
	No	0	0

$$\text{Total Error} = \frac{5}{14}$$

One R Classifier

Feature = Humidity

- For each feature,
  - For each value of that feature, make a rule as follows;
    - Count how often each value of the Class appears
    - Find the most frequent Class
    - Make the rule assign that Class to this value of the feature
  - Calculate the total error of the rules of each feature
- Choose the feature with the smallest total error

If High → NO
If Normal → YES

		Play? Actual Yes	Play? Actual No
High (Predicted)	Yes	0	0
	No	3	4

		Play? Actual Yes	Play? Actual No
Normal (Predicted)	Yes	6	1
	No	0	0

$$\text{Total Error} = \frac{4}{14}$$

		Play? Actual Yes	Play? Actual No
Humidity (Predicted)	Yes	6	1
	No	3	4

One R Classifier

Feature = Windy

- For each feature,
  - For each value of that feature, make a rule as follows;
    - Count how often each value of the Class appears
    - Find the most frequent Class
    - Make the rule assign that Class to this value of the feature
  - Calculate the total error of the rules of each feature
- Choose the feature with the smallest total error

If FALSE → YES
If TRUE → <u>YES</u> or NO

		Play? Actual Yes	Play? Actual No
FALSE (Predicted)	Yes	6	2
	No	0	0

		Play? Actual Yes	Play? Actual No
TRUE (Predicted)	Yes	3	3
	No	0	0

$$\text{Total Error} = \frac{5}{14}$$

		Play? Actual Yes	Play? Actual No
TRUE (Predicted)	Yes	9	5
	No	0	0

One R Classifier

- For each feature,
  - For each value of that feature, make a rule as follows;
    - Count how often each value of the Class appears
    - Find the most frequent Class
    - Make the rule assign that Class to this value of the feature
  - Calculate the total error of the rules of each feature
- Choose the feature with the smallest total error

Choose either the rule associated with the feature Outlook or the feature Humidity to classify new observations

Feature	Rule	Total Error
Outlook	If Sunny → NO If Overcast → YES If Rainy → YES	4/14
Temp	If Hot → YES If Mild→ YES If Cool → YES	5/14
Humidity	If High → NO If Normal → YES	4/14
Windy	If FALSE → YES If TRUE → YES	5/14

An implementation: https://learning.oreilly.com/library/view/learning-data-mining/9781787126787/dbbf3003-dcd4-46d0-b35e-253f12220d23.xhtml

Recap

- Learning and the Classification Problem
- Examples of Classification Problems
- General Approaches
- Specific Approaches
  - Decision Tree
  - Zero Rule (Zero R)
  - One Rule (One R)