CLASSIFICATION

Slides by:
Shree Jaswal

TOPICS TO BE COVERED

- Basic Concepts;
- **Classification methods**:

1. Decision Tree Induction: Attribute Selection Measures, Tree pruning.

2. Bayesian Classification: Naïve Bayes' classifier

- **Prediction**: Structure of regression models; Simple linear regression, Multiple linear regression.
- **Model Evaluation & Selection: A**ccuracy and Error measures, Holdout, Random Sampling, Cross Validation, Bootstrap; Comparing Classifier performance using ROC Curves.
- **Combining Classifiers**: Bagging, Boosting, Random Forests.

CLASSIFICATION: DEFINITION

- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.
- Goal: previously unseen records should be assigned a class as accurately as possible.
  - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

CLASSIFICATION

- Maps data into predefined groups or classes.
- Two step process
  - Training set
    - A model built describing a predetermined set of data classes
    - Supervised learning
  - Use model for classification
    - Accuracy of the model is first estimated.
    - Then classify/ predict the data.

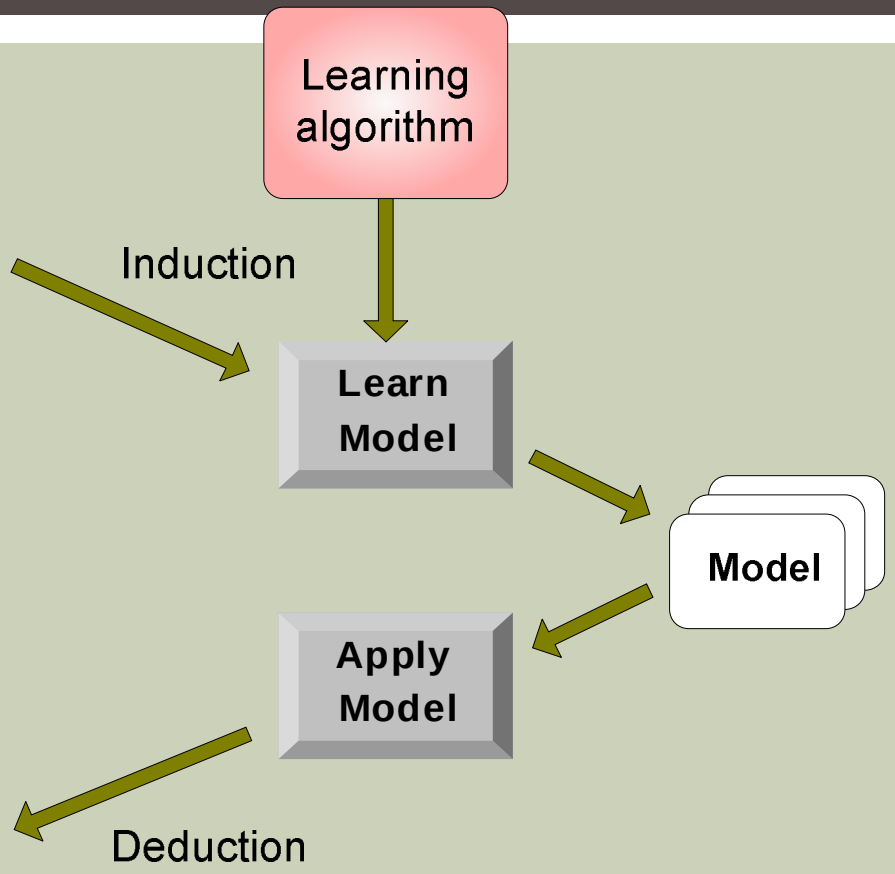
ILLUSTRATING CLASSIFICATION TASK

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

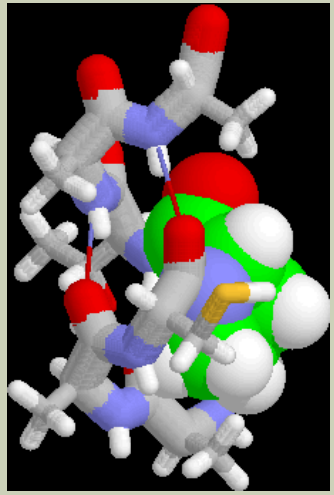
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



EXAMPLES OF CLASSIFICATION TASK

- Predicting tumor cells as benign or malignant
- Classifying credit card transactions as legitimate or fraudulent
- Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil
- Categorizing news stories as finance, weather, entertainment, sports, etc



CLASSIFICATION V/S PREDICTION

- Prediction: assess the class of an unlabeled sample.
  - Classification: predict discrete or nominal values
  - Regression: predict continuous or ordered values.

  Commonly "*classification*" is used to predict class labels, while "*prediction*" is used to predict continuous values as in regression.
- Regression is a data mining function that predicts a number.
- A regression task begins with a data set in which the target values are known.
- Profit, sales, mortgage rates, house values, square footage, temperature, or distance could all be predicted using regression techniques.
- For example, a regression model could be used to predict the value of a house based on location, number of rooms, lot size, and other factors.

ISSUES IN CLASSIFICATION

- Missing data
  - Values missing in the training data
  - Attribute values missing in the samples.
  - Handling missing data
    - Ignore
    - Assume a mean or average value.
    - Assume a special value
- Measuring performance
  - Classifier accuracy

CLASSIFIER ACCURACY

- Estimate classifier accuracy
  - Holdout method
  - Cross-validation method
- Increase classifier accuracy
  - Tree pruning, in case of decision trees.
  - Bagging
  - Boosting

CLASSIFICATION TECHNIQUES

- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines

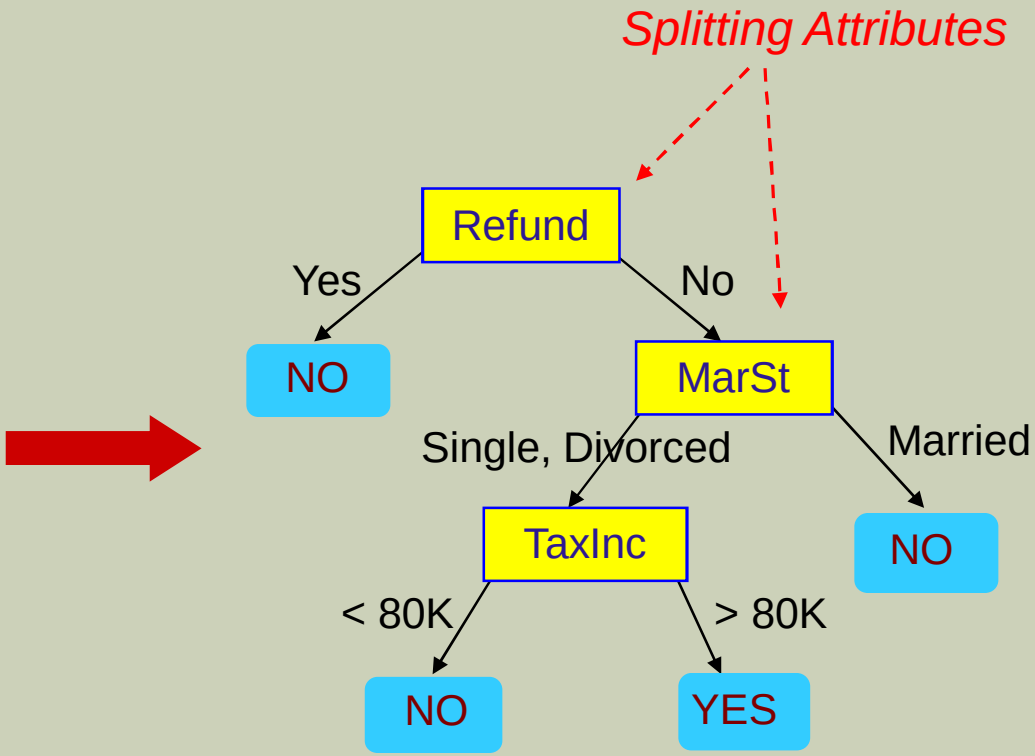
BY DECISION TREE

- A decision tree is a flow chart like tree structure, where
  - Each *internal node* denotes a *test* on an attribute
  - Each *branch* denotes an *outcome* of the test
  - Each *leaf node* represent a *class*
- In order to classify an unknown sample, the attribute values of the sample are tested against the decision tree.

EXAMPLE OF A DECISION TREE

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
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**

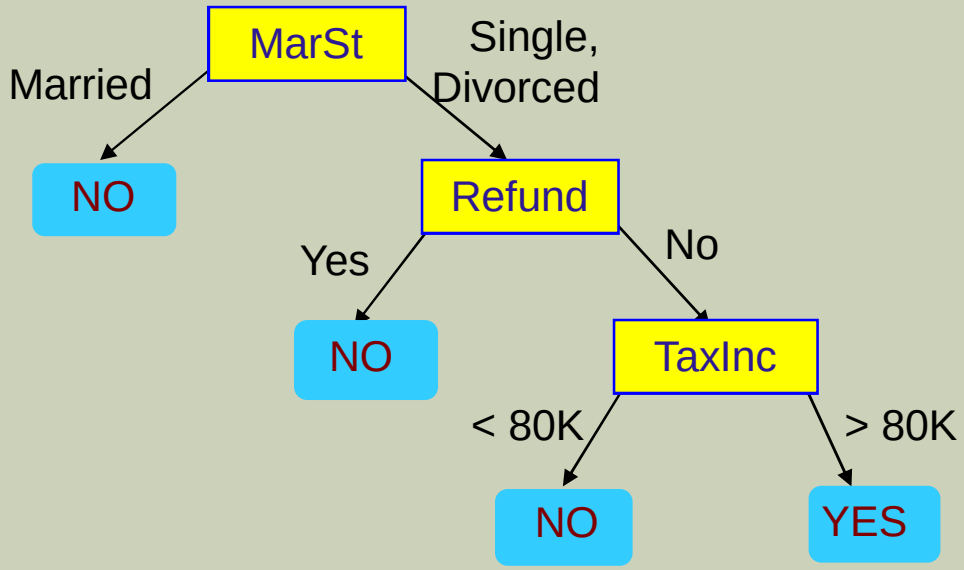
Training Data



Model: Decision Tree

ANOTHER EXAMPLE OF DECISION TREE

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
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**



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

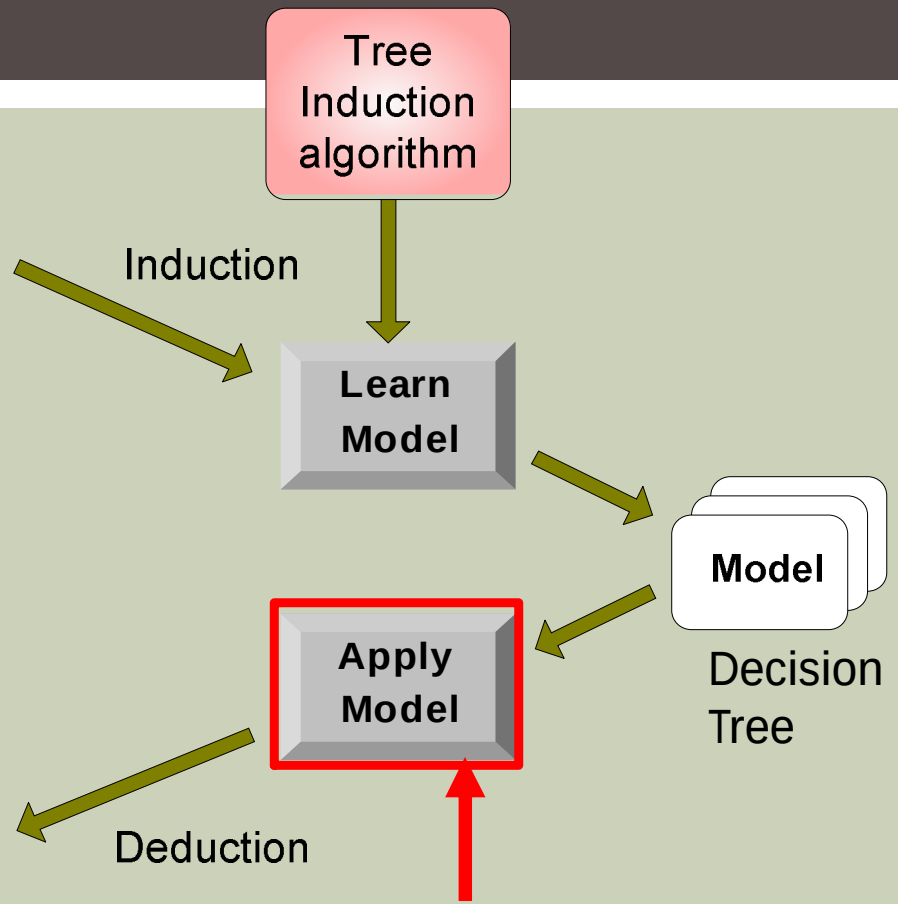
DECISION TREE CLASSIFICATION TASK

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

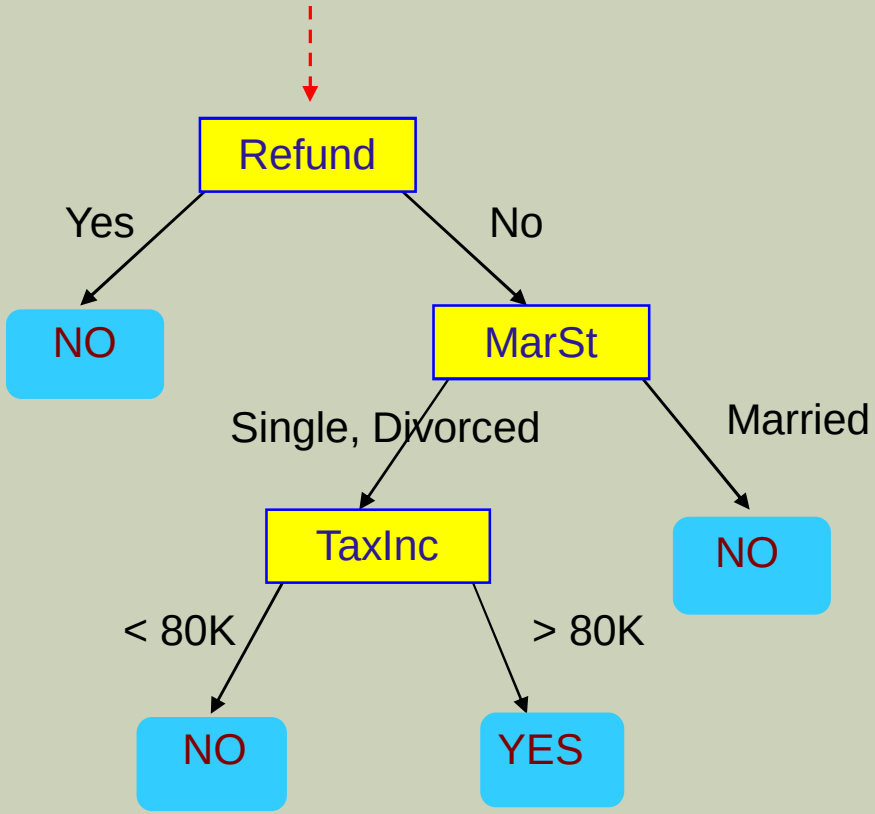
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



APPLY MODEL TO TEST DATA

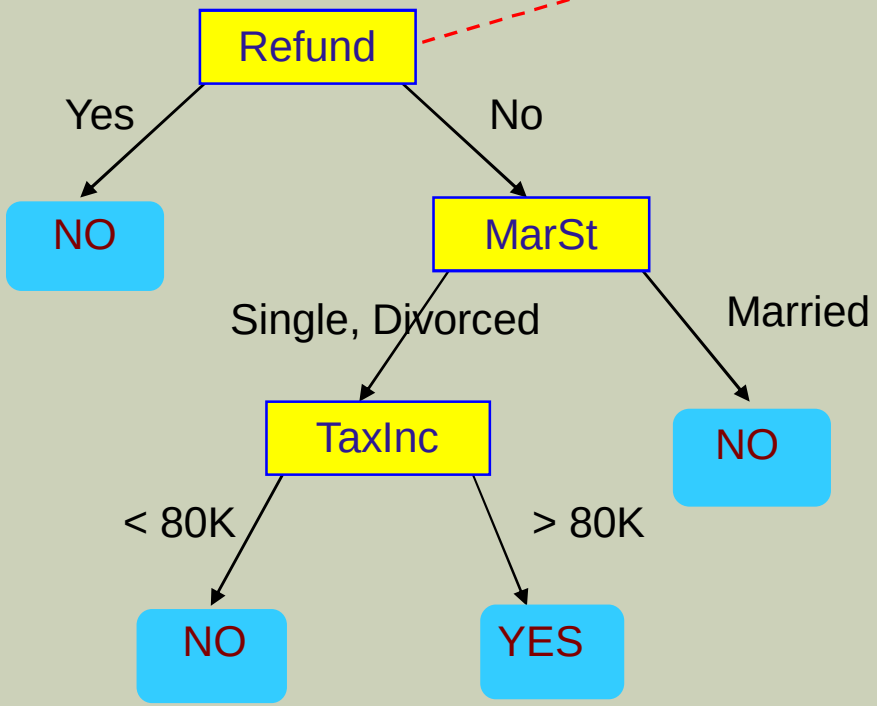
Start from the root of tree.



Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	**?**

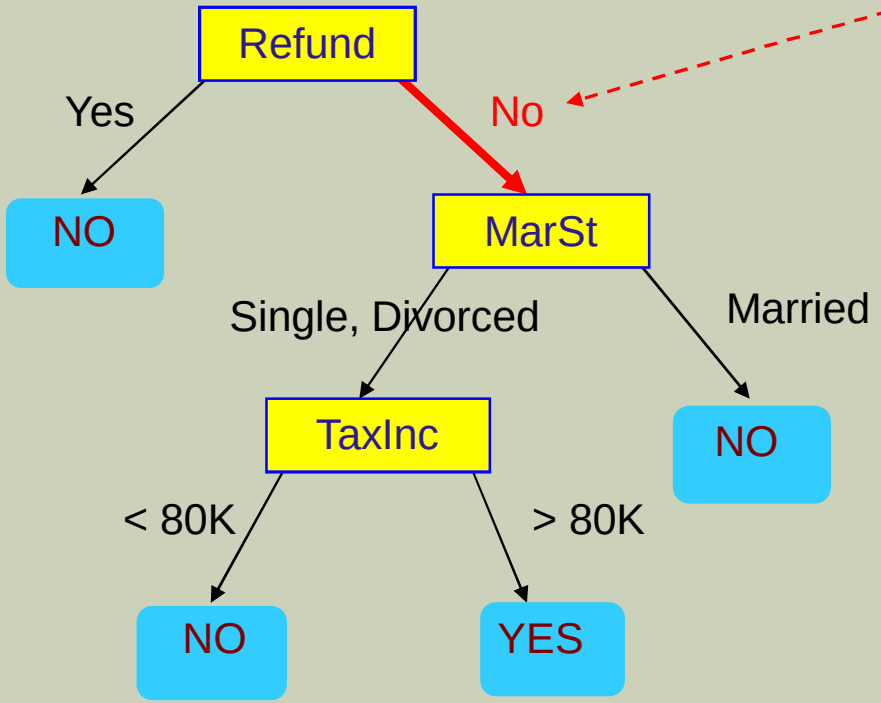
APPLY MODEL TO TEST DATA

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



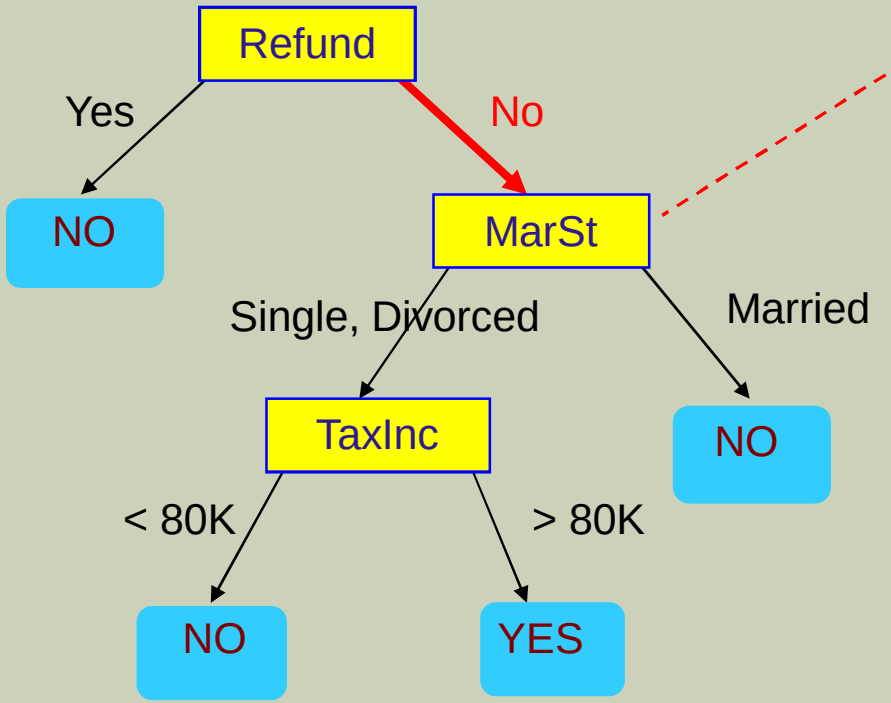
APPLY MODEL TO TEST DATA

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



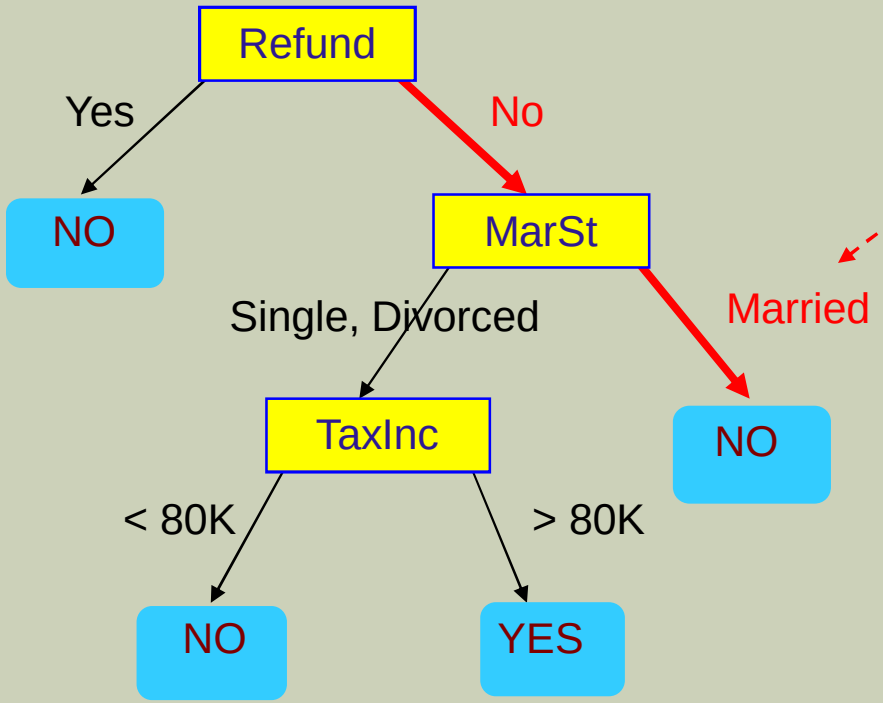
APPLY MODEL TO TEST DATA

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



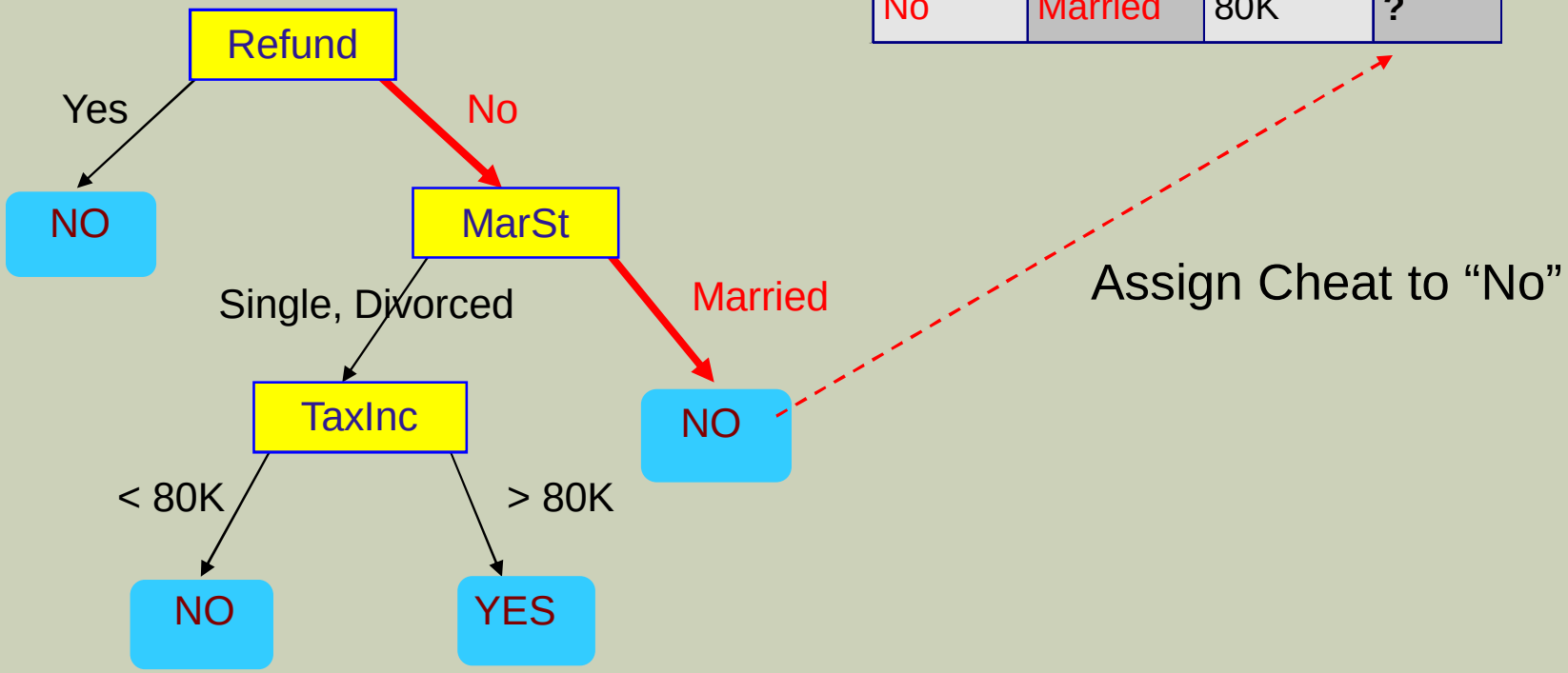
APPLY MODEL TO TEST DATA

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



APPLY MODEL TO TEST DATA

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



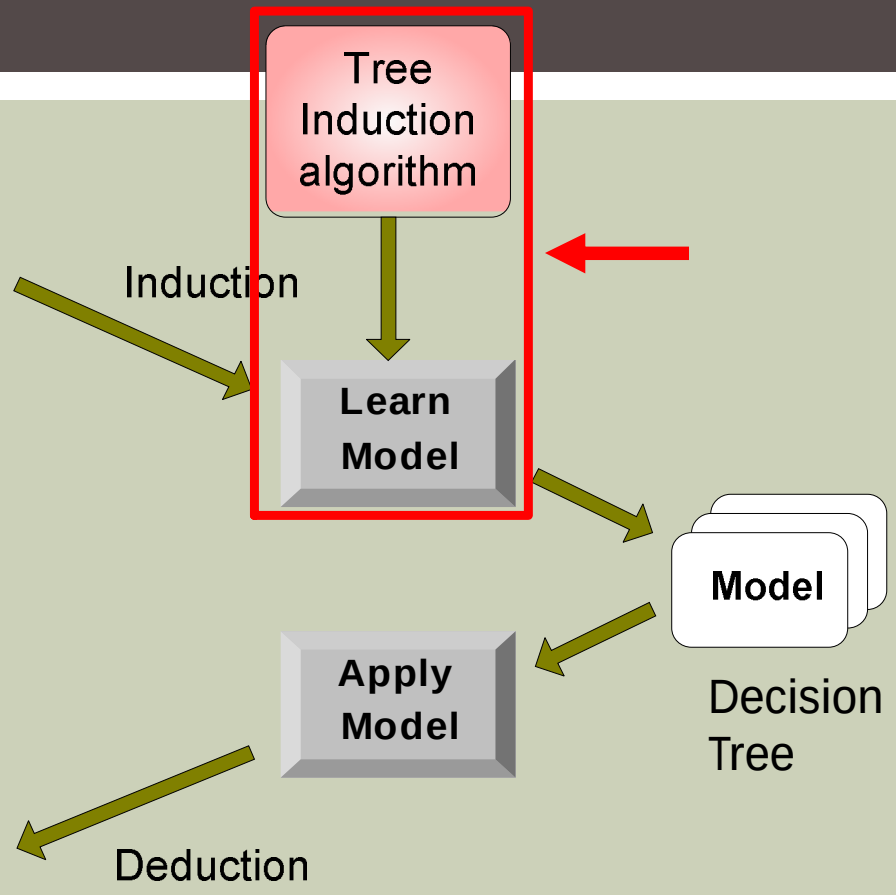
DECISION TREE CLASSIFICATION TASK

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 (Classification And Regression trees)
  - ID3 (Iterative dichotomiser), C4.5
  - SLIQ,SPRINT

TREE INDUCTION

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

TREE INDUCTION

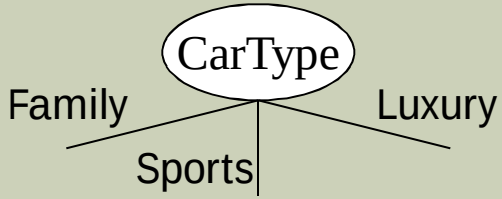
- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

HOW TO SPECIFY TEST CONDITION?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

SPLITTING BASED ON NOMINAL ATTRIBUTES

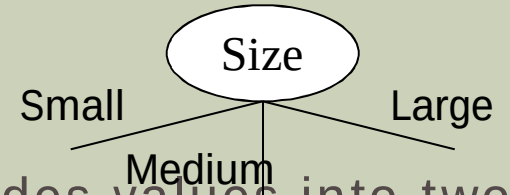
- Multi-way split: Use as many partitions as distinct values.



- Binary split: Divides values into two subsets.
  Need to find optimal partitioning.

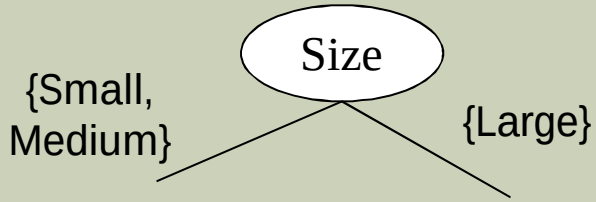
SPLITTING BASED ON ORDINAL ATTRIBUTES

- Multi-way split: Use as many partitions as distinct values.

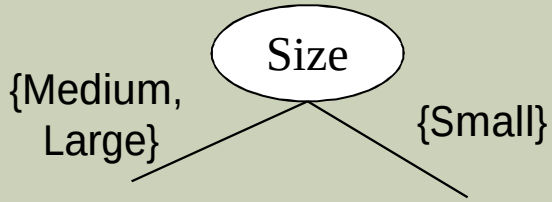


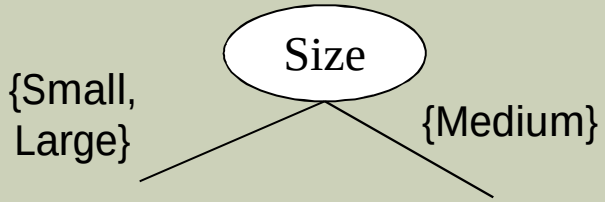
- Binary split: Divides values into two subsets.
  Need to find optimal partitioning.

- What about this split?



OR

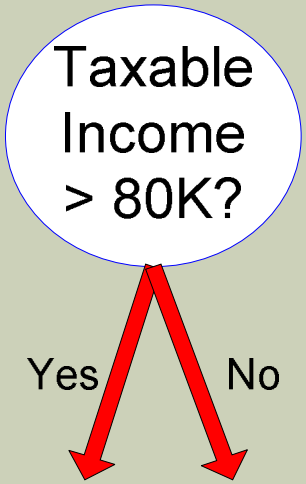




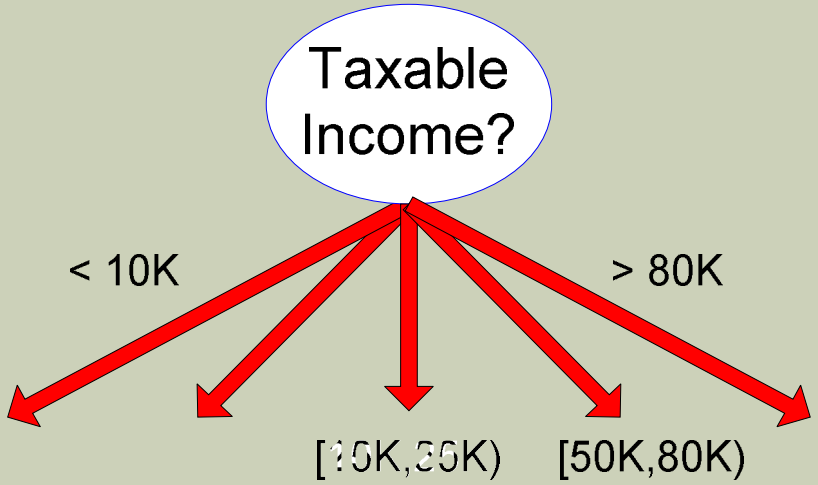
SPLITTING BASED ON CONTINUOUS ATTRIBUTES

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

SPLITTING BASED ON CONTINUOUS ATTRIBUTES



(i) Binary split



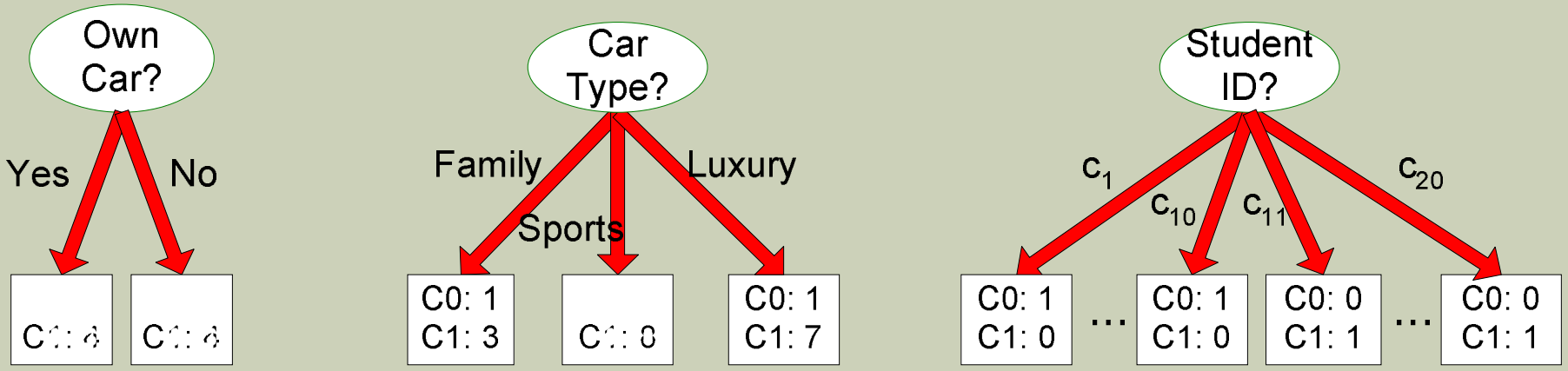
(ii) Multi-way split

TREE INDUCTION

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

HOW TO DETERMINE THE BEST SPLIT

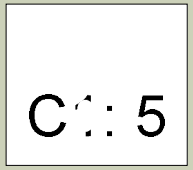
Before Splitting: 10 records of class 0,
10 records of class 1



Which test condition is the best?

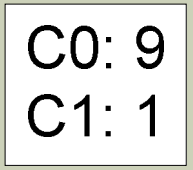
HOW TO DETERMINE THE BEST SPLIT

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



Non-homogeneous,

High degree of impurity



Homogeneous,

Low degree of impurity

CHOOSING ATTRIBUTES

- The order in which attributes are chosen determines how complicated the tree is.
- ID3 uses information theory to determine the most informative attribute.
- A measure of the information content of a message is the inverse of the probability of receiving the message:

  $$information(M) = 1/probability(M)$$

- Taking logs (base 2) makes information correspond to the number of bits required to encode a message:

  $$information(M) = -log_2(probability(M))$$

ENTROPY

- Different messages have different probabilities of arrival.
- Overall level of uncertainty (termed entropy) is:

$$-\Sigma_i P_i \log_2 P_i$$

- Frequency can be used as a probability estimate.
  - E.g. if there are 5 positive examples and 3 negative examples in a node the estimated probability of positive is 5/8 = 0.625.

INFORMATION AND LEARNING

- We can think of learning as building many-to-one mappings between input and output.
- Learning tries to reduce the information content of the inputs by mapping them to fewer outputs.
- Hence we try to minimise entropy.
- The simplest mapping is to map everything to one output.
- We seek a trade-off between accuracy and simplicity.

SPLITTING CRITERION

- Work out entropy based on distribution of classes.
- Trying splitting on each attribute.
- Work out expected information gain for each attribute.
- Choose best attribute.

ID3 ALGORITHM

- Constructs a decision tree by using top-down recursive approach.
- Main aim is to choose that splitting attribute which is having the highest information gain.

- The tree starts as a single node, representing all the training samples
- If all samples are of the same class, then the node becomes a leaf node and is labeled with that class.
- Else, an entropy-based algorithm, known as information gain, is used for selecting the attribute that will best separate the samples into individual classes. This attribute becomes the test attribute at the node.

- Branch is now constructed for each value of the test attribute and samples are partitioned accordingly.
- The algorithm then recursively applies the same process to form a decision tree for samples at each node.
- This recursive partitioning stops when either:
  - It is a leaf node
  - Splitting attributes is over

CALCULATE INFORMATION GAIN

- Entropy:
  - Given a collection S of c outcomes
  - Entropy(S) = S -p(I) log2 p(I)
  - Where p(I) is the proportion of S belonging to class I. S is over c. Log2 is log base 2

- Gain(S, A) is information gain of example set S on attribute A is defined as
  - Gain(S, A) = Entropy(S) - S ((|S_v| / |S|) * Entropy(S_v))
- Where:
  - S is each value v of all possible values of attribute A
  - S_v = subset of S for which attribute A has value v
  - |S_v| = number of elements in S_v |S| = number of elements in S

TRAINING SET

RID	Age	Income	Student	Credit	Buys
1	<30	High	No	Fair	No
2	<30	High	No	Excellent	No
3	31-40	High	No	Fair	Yes
4	>40	Medium	No	Fair	Yes
5	>40	Low	Yes	Fair	Yes
6	>40	Low	Yes	Excellent	No
7	31-40	Low	Yes	Excellent	Yes
8	<30	Medium	No	Fair	No
9	<30	Low	Yes	Fair	Yes
10	>40	Medium	Yes	Fair	Yes
11	<30	Medium	Yes	Excellent	Yes
12	31-40	Medium	No	Excellent	Yes
13	31-40	High	Yes	Fair	Yes
14	>40	Medium	No	Excellent	No

- S is a collection of 14 examples with 9 YES and 5 NO examples then

  Entropy(S) = - (9/14) Log_2 (9/14) - (5/14) Log_2 (5/14)

  = 0.940

- Let us consider Age as the splitting attribute

  Gain(S, Age) = Entropy(S) - (5/14)*Entropy($S_{<30}$)
  - (4/14)*Entropy($S_{31\text{-}40}$)
  - (5/14)*Entropy($S_{>40}$)

  = 0.940 - (5/14)*0.971 – (4/14)*0 – (5/14)*0.971

  = **0.246**

- Similarly consider Student as the splitting attribute

  Gain(S, Student) = Entropy(S) - (7/14)*Entropy(S_{YES})
  - (7/14)*Entropy(S_{NO})

  = **0.151**

- Similarly consider Credit as the splitting attribute

  Gain(S, Credit) = Entropy(S) - (8/14)*Entropy(S_{FAIR})
  - (6/14)*Entropy($S_{EXCELLENT}$)

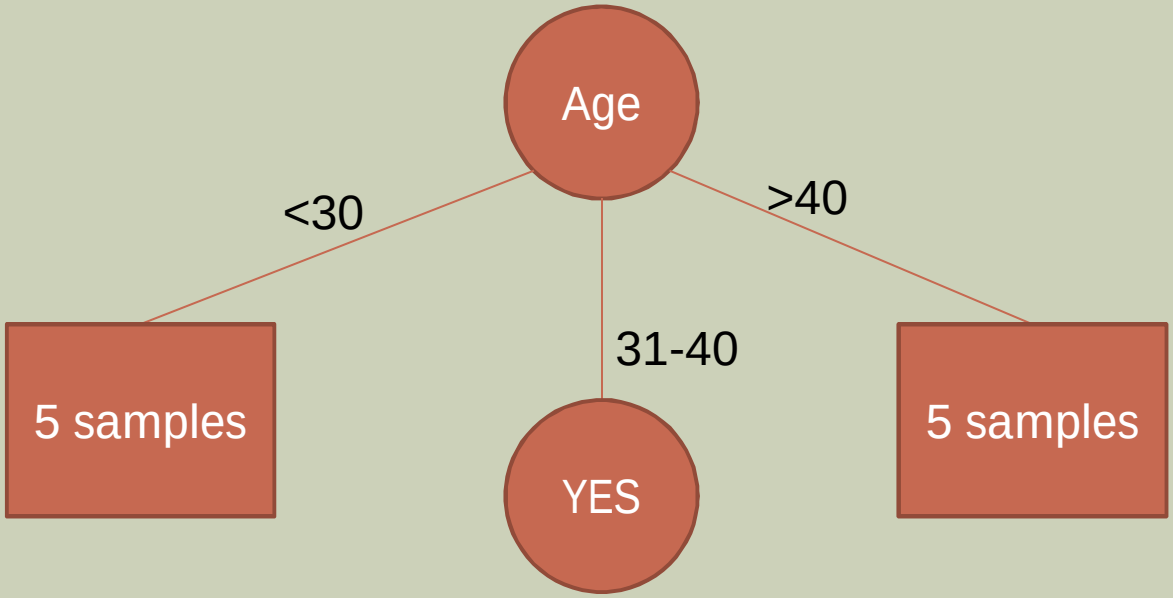
  = **0.046**

- Similarly consider Income as the splitting attribute

  Gain(S, Income) = Entropy(S) - (4/14)*Entropy(S_{HIGH})
  - (5/14)*Entropy(S_{MED})
  - (5/14)*Entropy(S_{LOW})

  = **0.029**

- We see that Gain (S, Age) is max with 0.246.
- Hence Age is chosen as the splitting attribute.



- Recursively we find the splitting attributes at the next level.
- Let us consider Student as the next splitting attribute

  Gain(Age$_{<30}$, Student) = Entropy(Age$_{<30}$) - (2/5)*Entropy(S_{YES}) - (3/5)*Entropy(S_{NO})

  = 0.971 - (2/5)*0 – (3/5)*0

  = **0.971**
- Similarly Credit as the next splitting attribute

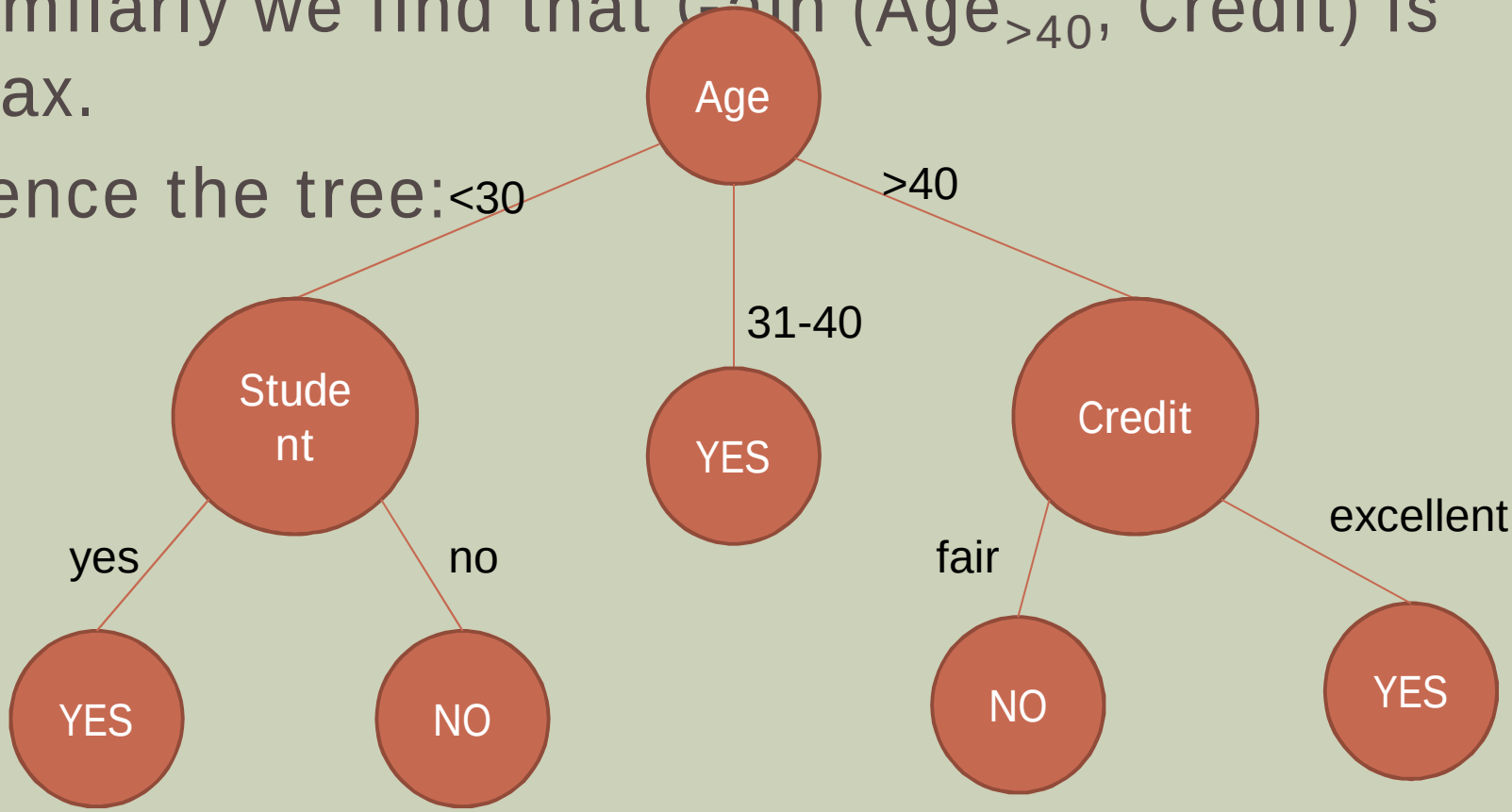
  Gain(Age$_{<30}$, Credit) = Entropy(Age$_{<30}$) - (3/5)*Entropy(S_{FAIR}) - (2/5)*Entropy($S_{EXCELLENT}$)

  = **0.57**
- Similarly Income as the next splitting attribute

  Gain(Age$_{<30}$, Income) = Entropy(Age$_{<30}$) - (2/5)*Entropy(S_{HIGH}) - (2/5)*Entropy(S_{MED}) - (1/5)*Entropy(S_{LOW})

  = **0.19**

- We see that Gain (Age$_{<30}$, Student) is max with 0.971.
- Hence Student is chosen as the next splitting attribute.
- Similarly we find that Gain (Age$_{>40}$, Credit) is max.
- Hence the tree:



EXTRACTING CLASSIFICATION RULES FROM TREES

- Represent the knowledge in the form of IF-THEN rules
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction
- The leaf node holds the class prediction
- Rules are easier for humans to understand
- Example

  IF *age* = "<=30" AND *student* = "*no*" THEN *buys_computer* = "*no*"

  IF *age* = "<=30" AND *student* = "*yes*" THEN *buys_computer* = "*yes*"

  IF *age* = "31…40" THEN *buys_computer* = "*yes*"

  IF *age* = ">40" AND *credit_rating* = "*excellent*" THEN *buys_computer* = "*yes*"

  IF *age* = ">40" AND *credit_rating* = "*fair*" THEN *buys_computer* = "*no*"

CONSTRUCT A DECISION TREE FOR THE BELOW EXAMPLE:

- Suppose we want ID3 to decide whether the weather is amenable to playing baseball. Over the course of 2 weeks, data is collected to help ID3 build a decision tree. The target classification is "should we play baseball?" which can be yes or no.
- The weather attributes can have the following values:
  - outlook = { sunny, overcast, rain }
  - temperature = {hot, mild, cool }
  - humidity = { high, normal }
  - wind = {weak, strong }

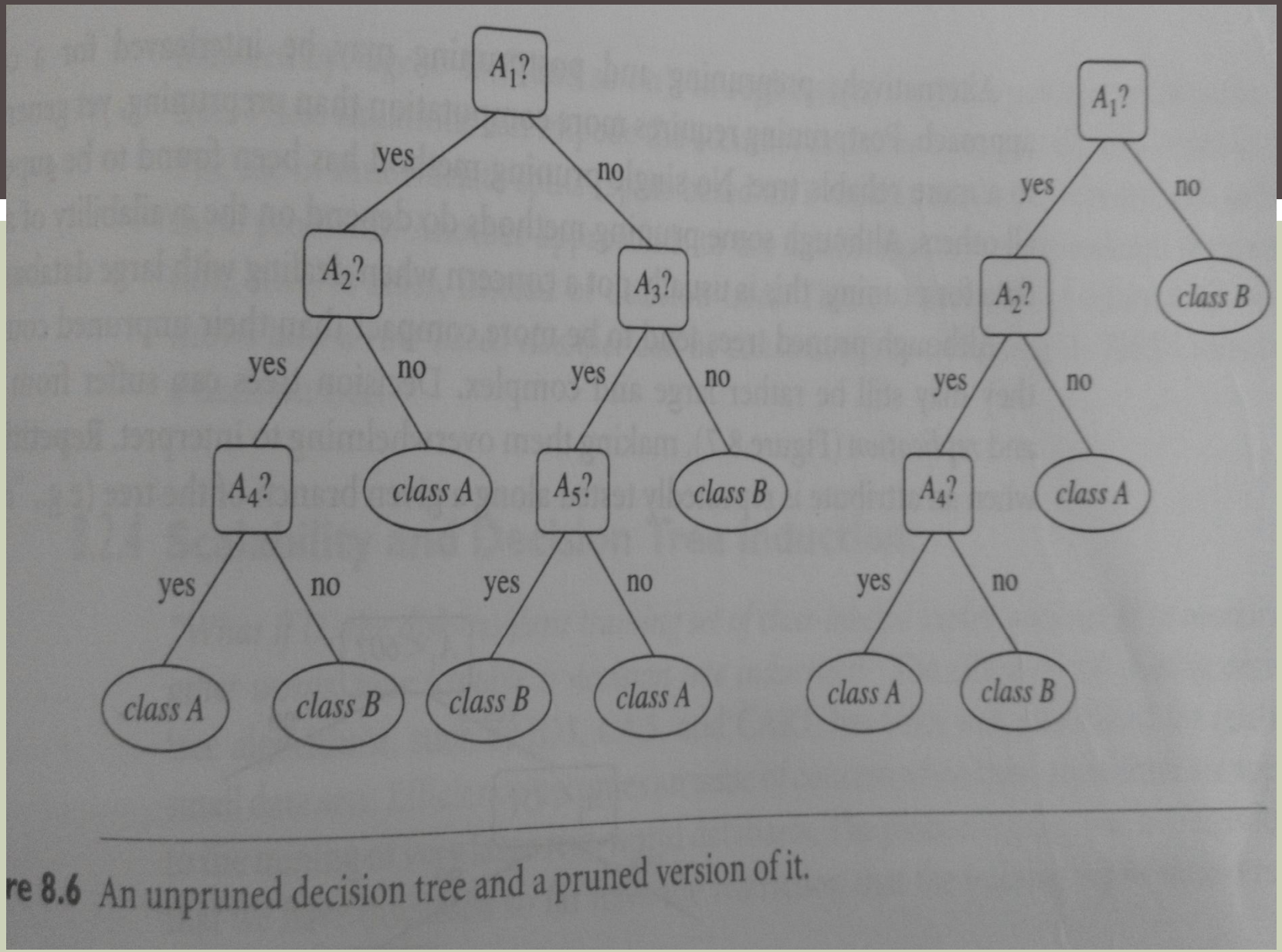
Day	Outlook	Temperature	Humidity	Wind	Play ball
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

TREE PRUNING

- Some branches of the decision tree may reflect anomalies due to noise/ outliers.
- Overfitting results in decision trees that are more complex than necessary
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records
- Tree pruning helps in faster classification with more accurate results
- Two methods
  - Pre-pruning: halting its construction during formation
  - Post-pruning: remove branches from a fully grown tree.

HOW TO ADDRESS OVERFITTING

- Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same
  - More restrictive conditions:
    - Stop if number of instances is less than some user-specified threshold
    - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
    - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).



re 8.6 An unpruned decision tree and a pruned version of it.

HOW TO ADDRESS OVERFITTING

- Post-pruning
  - Grow decision tree to its entirety
  - Trim the nodes of the decision tree in a bottom-up fashion
  - If generalization error improves after trimming, replace sub-tree by a leaf node.
  - Class label of leaf node is determined from majority class of instances in the sub-tree

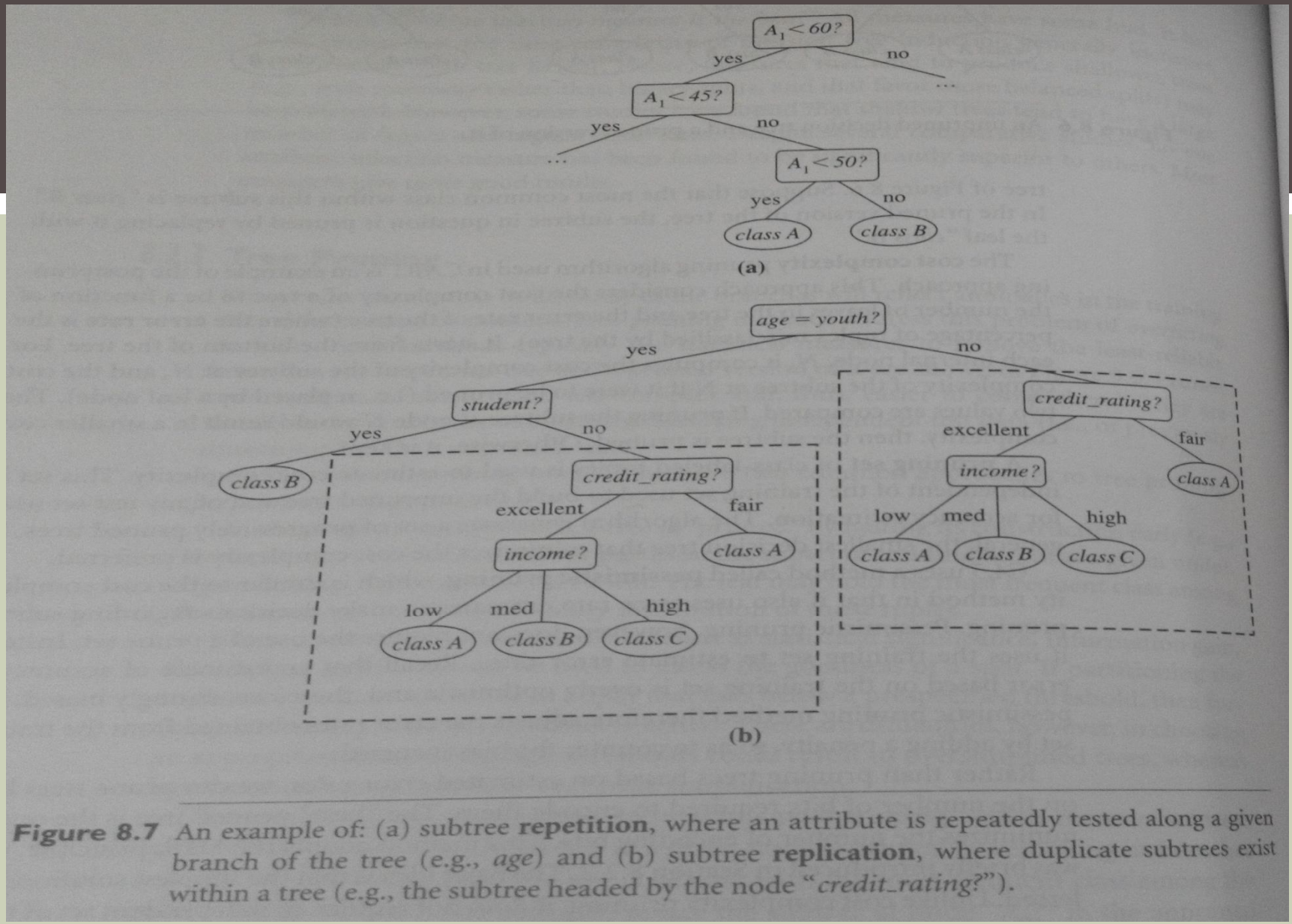


Figure 8.7 An example of: (a) subtree **repetition**, where an attribute is repeatedly tested along a given branch of the tree (e.g., *age*) and (b) subtree **replication**, where duplicate subtrees exist within a tree (e.g., the subtree headed by the node "*credit_rating*?").

BAYESIAN CLASSIFICATION: WHY?

- A statistical classifier: performs *probabilistic prediction, i.e.,* predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

BAYESIAN THEOREM: BASICS

- Let **X** be a data sample ("*evidence*"): class label is unknown
- Let H be a *hypothesis* that X belongs to class C
- Classification is to determine P(H|**X**), the probability that the hypothesis holds given the observed data sample **X**
- P(H) (*prior probability*), the initial probability
  - E.g., **X** will buy computer, regardless of age, income, …
- P(**X**): probability that sample data is observed
- P(**X**|H) (*posteriori probability*), the probability of observing the sample **X**, given that the hypothesis holds
  - E.g., Given that **X** will buy computer, the prob. that X is 31..40, medium income

BAYESIAN THEOREM

- Given training data **X**, *posteriori probability of a hypothesis* H, P(H|**X**), follows the Bayes theorem

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})}$$

- Informally, this can be written as

  posteriori = likelihood x prior/evidence

- Predicts **X** belongs to C_2 iff the probability $P(C_i|\mathbf{X})$ is the highest among all the $P(C_k|X)$ for all the *k* classes

- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

TOWARDS NAÏVE BAYESIAN CLASSIFIER

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector **X** = (x_1, x_2, …, x_n)
- Suppose there are *m* classes C_1, C_2, …, C_m.
- Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|\mathbf{X})$
- This can be derived from Bayes' theorem

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

- Since P(X) is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

needs to be maximized

NAÏVE BAYESIAN CLASSIFIER: TRAINING DATASET

Class:
C1:buys_computer = ‘yes’
C2:buys_computer = ‘no’

Data sample
X = (age <=30,
Income = medium,
Student = yes
Credit_rating = Fair)

age	income	student	credit_rating	_comp
<=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

NAÏVE BAYESIAN CLASSIFIER: AN EXAMPLE

- $P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643
  P(buys_computer = "no") = 5/14= 0.357

- Compute $P(X|C_i)$ for each class
  P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222
  P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6
  P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444
  P(income = "medium" | buys_computer = "no") = 2/5 = 0.4
  P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667
  P(student = "yes" | buys_computer = "no") = 1/5 = 0.2
  P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667
  P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4

- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(X|C_i)$: P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019
$P(X|C_i)*P(C_i)$: P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

Therefore, X belongs to class ("buys_computer = yes")

AVOIDING THE 0-PROBABILITY PROBLEM

- Naïve Bayesian prediction requires each conditional prob. be non-zero. Otherwise, the predicted prob. will be zero

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10),
- Use Laplacian correction (or Laplacian estimator)
  - Adding 1 to each case

    Prob(income = low) = 1/1003

    Prob(income = medium) = 991/1003

    Prob(income = high) = 11/1003
  - The "corrected" prob. estimates are close to their "uncorrected" counterparts

NAÏVE BAYESIAN CLASSIFIER: COMMENTS

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.

      Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- How to deal with these dependencies?
  - Bayesian Belief Networks

WHAT IS PREDICTION?

- (Numerical) prediction is similar to classification
  - construct a model
  - use model to predict continuous or ordered value for a given input
- Prediction is different from classification
  - Classification refers to predict categorical class label
  - Prediction models continuous-valued functions
- Major method for prediction: regression
  - model the relationship between one or more *independent* or **predictor** variables and a *dependent* or **response** variable
- Regression analysis
  - Linear and multiple regression
  - Non-linear regression
  - Other regression methods: generalized linear model, Poisson regression, log-linear models, regression trees

LINEAR REGRESSION

- Linear regression: involves a response variable y and a single predictor variable x

  $y = w_0 + w_1 x$

  where w_0 (y-intercept) and w_1 (slope) are regression coefficients
- Method of least squares: estimates the best-fitting straight line

$$w_1 = \frac{\sum_{i=1}^{|D|}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|}(x_i - \bar{x})^2} \qquad w_0 = \bar{y} - w_1 \bar{x}$$

- Multiple linear regression: involves more than one predictor variable
  - Training data is of the form $(\mathbf{X_1}, y_1), (\mathbf{X_2}, y_2), \ldots, (\mathbf{X_{|D|}}, y_{|D|})$
  - Ex. For 2-D data, we may have: $y = w_0 + w_1 x_1 + w_2 x_2$
  - Solvable by extension of least square method or using SAS, S-Plus
  - Many nonlinear functions can be transformed into the above

NONLINEAR REGRESSION

- Some nonlinear models can be modeled by a polynomial function
- A polynomial regression model can be transformed into linear regression model. For example,

  $y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$

  convertible to linear with new variables: $x_2 = x^2$, $x_3 = x^3$

  $y = w_0 + w_1 x + w_2 x_2 + w_3 x_3$
- Other functions, such as power function, can also be transformed to linear model
- Some models are intractable nonlinear (e.g., sum of exponential terms)
  - possible to obtain least square estimates through extensive calculation on more complex formulae

OTHER REGRESSION-BASED MODELS

- Generalized linear model:
  - Foundation on which linear regression can be applied to modeling categorical response variables
  - Variance of y is a function of the mean value of y, not a constant
  - Logistic regression: models the prob. of some event occurring as a linear function of a set of predictor variables
  - Poisson regression: models the data that exhibit a Poisson distribution
- Log-linear models: (for categorical data)
  - Approximate discrete multidimensional prob. distributions
  - Also useful for data compression and smoothing
- Regression trees and model trees
  - Trees to predict continuous values rather than class labels

REGRESSION TREES AND MODEL TREES

- Regression tree: proposed in CART system (Breiman et al. 1984)
  - CART: Classification And Regression Trees
  - Each leaf stores a *continuous-valued prediction*
  - It is the *average value of the predicted attribute* for the training tuples that reach the leaf
- Model tree: proposed by Quinlan (1992)
  - Each leaf holds a regression model—a multivariate linear equation for the predicted attribute
  - A more general case than regression tree
- Regression and model trees tend to be more accurate than linear regression when the data are not represented well by a simple linear model

METRICS FOR PERFORMANCE EVALUATION

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.
- Confusion Matrix:

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

METRICS FOR PERFORMANCE EVALUATION…

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

- Most widely-used metric:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

LIMITATION OF ACCURACY

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10

- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any class 1 example

CLASSIFIER ACCURACY MEASURES

	C_1	C_2
C_1	True positive	False negative
C_2	False positive	True negative

classes	buy_computer = yes	buy_computer = no	total	recognition(%)
buy_computer = yes	6954	46	7000	99.34
buy_computer = no	412	2588	3000	86.27
total	7366	2634	10000	95.52

- Accuracy of a classifier M, acc(M): percentage of test set tuples that are correctly classified by the model M
  - Error rate (misclassification rate) of M = 1 – acc(M)
  - Given *m* classes, $CM_{i,j}$, an entry in a **confusion matrix**, indicates # of tuples in class *i* that are labeled by the classifier as class *j*
- Alternative accuracy measures (e.g., for cancer diagnosis)

  sensitivity = t-pos/pos /* true positive recognition rate */

  specificity = t-neg/neg /* true negative recognition rate */

  precision = t-pos/(t-pos + f-pos)

  accuracy = sensitivity * pos/(pos + neg) + specificity * neg/(pos + neg)
  - This model can also be used for cost-benefit analysis

PREDICTOR ERROR MEASURES

- Measure predictor accuracy: measure how far off the predicted value is from the actual known value
- **Loss function**: measures the error betw. y_i and the predicted value y_i'
  - Absolute error: $| y_i - y_i' |$
  - Squared error: $(y_i - y_i')^2$
- Test error (generalization error): the average loss over the test set
  - Mean absolute error: $\frac{\sum_{i=1}^{d} | y_i - y_i' |}{d}$ Mean squared error: $\frac{\sum_{i=1}^{d} (y_i - y_i')^2}{d}$
  - Relative absolute error: $\frac{\sum_{i=1}^{d} | y_i - y_i' |}{\sum_{i=1}^{d} | y_i - \bar{y} |}$ Relative squared error: $\frac{\sum_{i=1}^{d} (y_i - y_i')^2}{\sum_{i=1}^{d} (y_i - \bar{y})^2}$

  The mean squared-error exaggerates the presence of outliers

  Popularly use (square) root mean-square error, similarly, root relative squared error

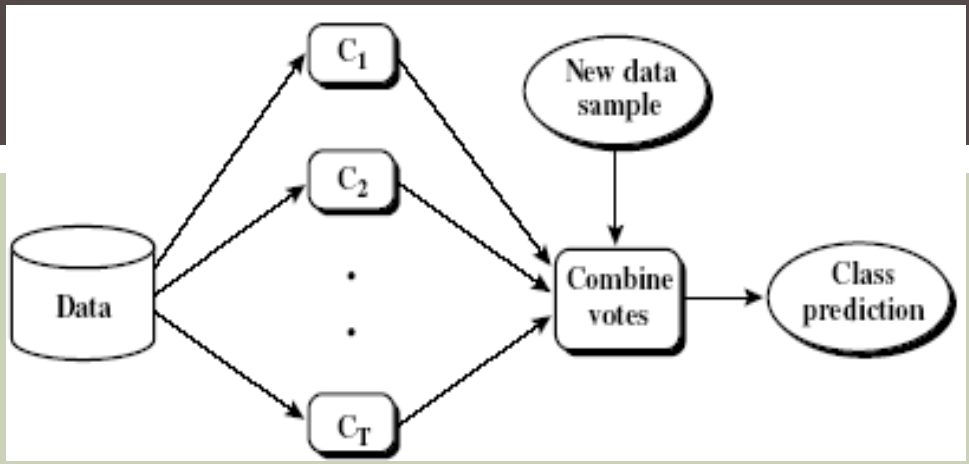
EVALUATING THE ACCURACY OF A CLASSIFIER OR PREDICTOR (I)

- Holdout method
  - Given data is randomly partitioned 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
    - Repeat holdout k times, accuracy = avg. of the accuracies obtained
- Cross-validation (*k*-fold, where k = 10 is most popular)
  - Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
  - At *i*-th iteration, use D_i as test set and others as training set
  - Leave-one-out: k folds where k = # of tuples, for small sized data
  - Stratified cross-validation: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data

EVALUATING THE ACCURACY OF A CLASSIFIER OR PREDICTOR (II)

- Bootstrap
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several boostrap methods, and a common one is **.632 boostrap**
  - Suppose we are given a data set of d tuples. The data set is sampled d times, with replacement, resulting in a training set of d samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data will end up in the bootstrap, and the remaining 36.8% will form the test set (since $(1 - 1/d)^d \approx e^{-1} = 0.368$)
  - Repeat the sampling procedue k times, overall accuracy of the model: $$acc(M) = \sum_{i=1}^{k} (0.632 \times acc(M_i)_{test\_set} + 0.368 \times acc(M_i)_{train\_set})$$

ENSEMBLE METHODS: INCREASING THE ACCURACY



- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, M_1, M_2, …, M_k, with the aim of creating an improved model M*
- Popular ensemble methods
  - Bagging: averaging the prediction over a collection of classifiers
  - Boosting: weighted vote with a collection of classifiers
  - Ensemble: combining a set of heterogeneous classifiers

BAGGING: BOOSTRAP AGGREGATION

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
  - Given a set D of *d* tuples, at each iteration *i*, a training set D_i of *d* tuples is sampled with replacement from D (i.e., boostrap)
  - A classifier model M_i is learned for each training set D_i
- Classification: classify an unknown sample **X**
  - Each classifier M_i returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to **X**
- Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy
  - Often significant better than a single classifier derived from D
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction

BOOSTING

- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy
- How boosting works?
  - Weights are assigned to each training tuple
  - A series of k classifiers is iteratively learned
  - After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to pay more attention to the training tuples that were misclassified by M_i
  - The final M* combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy
- The boosting algorithm can be extended for the prediction of continuous values
- Comparing with bagging: boosting tends to achieve greater accuracy, but it also risks overfitting the model to misclassified data

ADABOOST (FREUND AND SCHAPIRE, 1997)

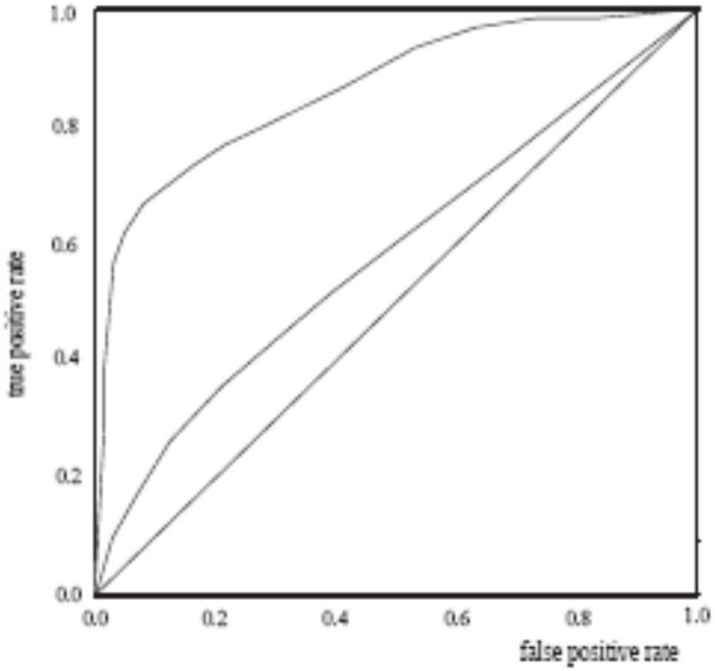
- Given a set of *d* class-labeled tuples, ($\mathbf{X_1}$, y_1), …, ($\mathbf{X_d}$, y_d)
- Initially, all the weights of tuples are set the same (1/d)
- Generate k classifiers in k rounds. At round i,
  - Tuples from D are sampled (with replacement) to form a training set D_i of the same size
  - Each tuple's chance of being selected is based on its weight
  - A classification model M_i is derived from D_i
  - Its error rate is calculated using D_i as a test set
  - If a tuple is misclssified, its weight is increased, o.w. it is decreased
- Error rate: err($\mathbf{X_j}$) is the misclassification error of tuple $\mathbf{X_j}$. Classifier M_i error rate is the sum of the weights of the misclassified tuples:

$$error(M_i) = \sum_{j}^{d} w_j \times err(\mathbf{X_j})$$

- The weight of classifier M_i's vote is $\log \frac{1-error(M_i)}{error(M_i)}$

MODEL SELECTION: ROC CURVES

- ROC (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Originated from signal detection theory
- Shows the trade-off between the true positive rate and the false positive rate
- The area under the ROC curve is a measure of the accuracy of the model
- Rank the test tuples in decreasing order: the one that is most likely to belong to the positive class appears at the top of the list
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model



- Vertical axis represents the true positive rate
- Horizontal axis rep. the false positive rate
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0