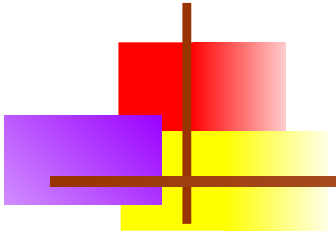
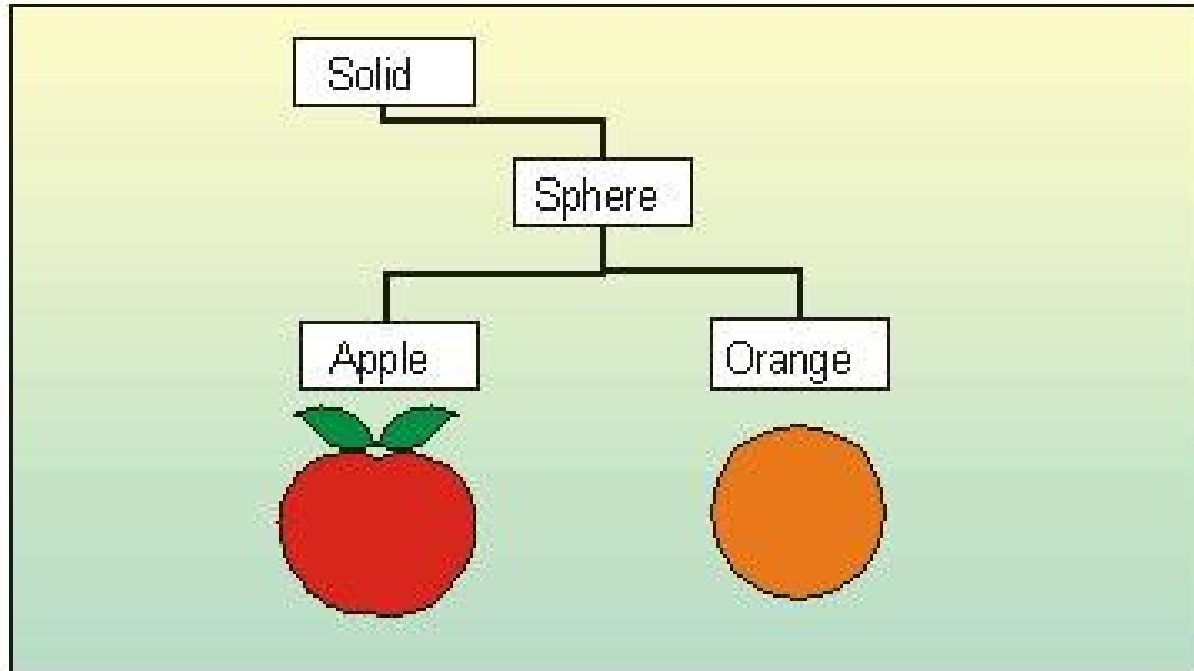


Classification



Fruit Categories



Yellow
Bananas















Golden
Pineapple



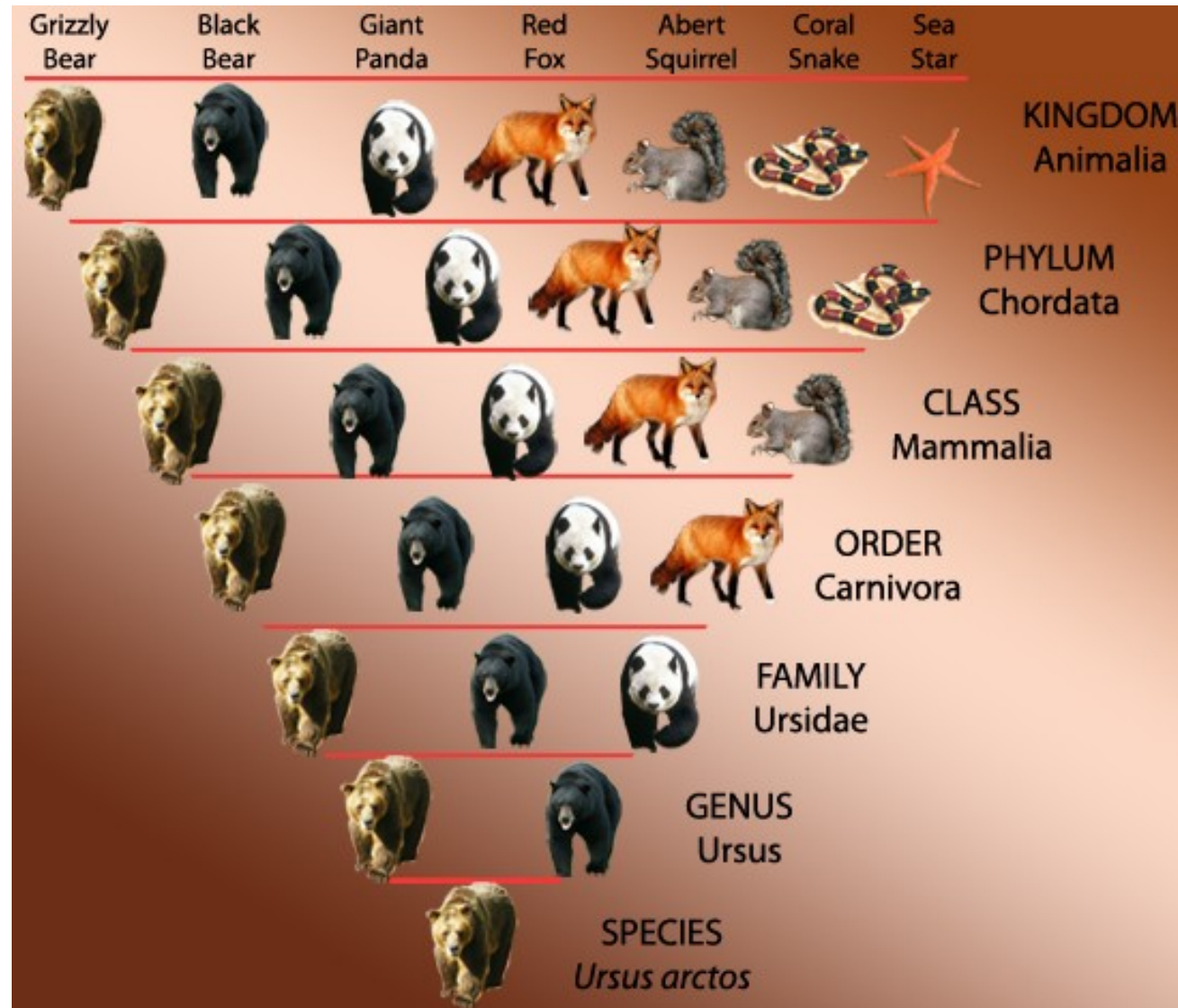
Passion
Fruits



Fruit Classification

<p>Yellow Bananas</p> 	<p>Golden Pineapple</p> 	<p>Black Grape</p> 
<p>Green Plantains</p> 	<p>Passion Fruits</p> 	<p>Black berry</p> 
<p>Tangerines</p> 	<p>Bosc Pears</p> 	<p>Blue berry</p> 
<p>Hass Avocado</p> 	<p>Anjou Pears</p> 	<p>Straw berry</p> 

Animal Classification/Categorization





Definition of Classification

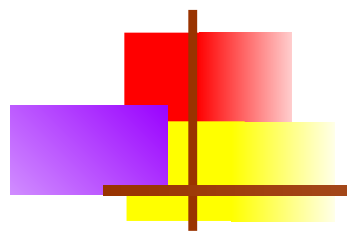
- Is assigning an object to a category of objects because of its similarity to them
- Like linear regression, Classification is another fundamental supervised learning method for **prediction**,
- In classification learning, a classifier is presented with a set of example (training set) that are already classified and, from these training set, the classifier learns to **predict** where to assign the unseen example
- Unlike clustering, classification uses predetermined labels to make classifications
- We will focus on one classification methods: **decision trees**



Example of Classification Application

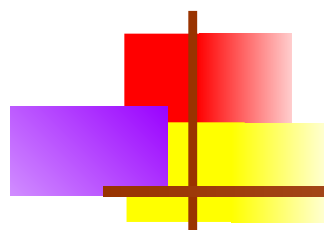
Following are the examples of cases where classification is used:

- A bank loan officer wants to analyze the data in order to know which customer (loan applicant) are risky or which are safe.
- A marketing manager at a company needs to analyze to guess a customer with a given profile will buy a new computer.
- In both of the above examples a model or classifier is constructed to predict categorical labels.
 - These labels are risky or safe for loan application data and yes or no for marketing data.



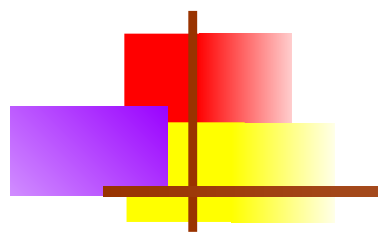
Classification: Application 1

- Direct Marketing
 - Goal: Reduce cost of mailing by *targeting* a set of consumers likely to buy a new cell-phone product.
 - Approach:
 - Use the data for a similar product introduced before.
 - We know which customers decided to buy and which decided otherwise. This *{buy, don't buy}* decision forms the *class attribute*.
 - Collect various demographic, lifestyle, and company-interaction related information about all such customers.
 - Type of business, where they stay, how much they earn, etc.
 - Use this information as input attributes to learn a classifier model.



Classification: Application 2

- Fraud Detection
 - Goal: Predict fraudulent cases in credit card transactions.
 - Approach:
 - Use credit card transactions and the information on its account-holder as attributes.
 - When does a customer buy, what does he buy, how often he pays on time, etc
 - Label past transactions as fraud or fair transactions. This forms the class attribute.
 - Learn a model for the class of the transactions.
 - Use this model to detect fraud by observing credit card transactions on an account.



Classification: Application 3

- Customer Attrition/Churn:
 - Goal: To predict whether a customer is likely to be lost to a competitor.
 - Approach:
 - Use detailed record of transactions with each of the past and present customers, to find attributes.
 - How often the customer calls, where he calls, what time-of-the day he calls most, his financial status, marital status, etc.
 - Label the customers as loyal or disloyal.
 - Find a model for loyalty.



How Does Classification Works?

The Data Classification process includes the two steps:

- Building the Classifier or Model
- Use the developed Classifier to start the process of Classification

Example of classification Process

- Given the training dataset about bank loan payments

<i>Tid</i>	<i>Refund</i>	<i>Marital Status</i>	<i>Taxable Income</i>	<i>Cheat</i>
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

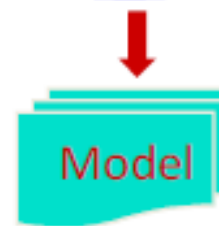
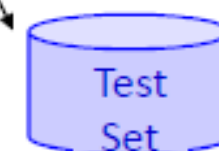
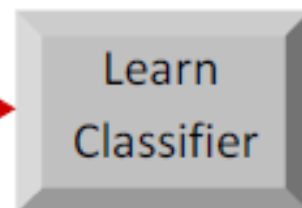
- Predict classification for the following customers

<i>Refund</i>	<i>Marital Status</i>	<i>Taxable Income</i>	<i>Cheat</i>
No	Single	75K	?
Yes	Married	50K	?
No	Married	150K	?
Yes	Divorced	90K	?
No	Single	40K	?
No	Married	80K	?

Classification Example

	categorical	categorical	continuous	class
<i>Tid</i>	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

Refund	Marital Status	Taxable Income	Cheat
No	Single	75K	?
Yes	Married	50K	?
No	Married	150K	?
Yes	Divorced	90K	?
No	Single	40K	?
No	Married	80K	?



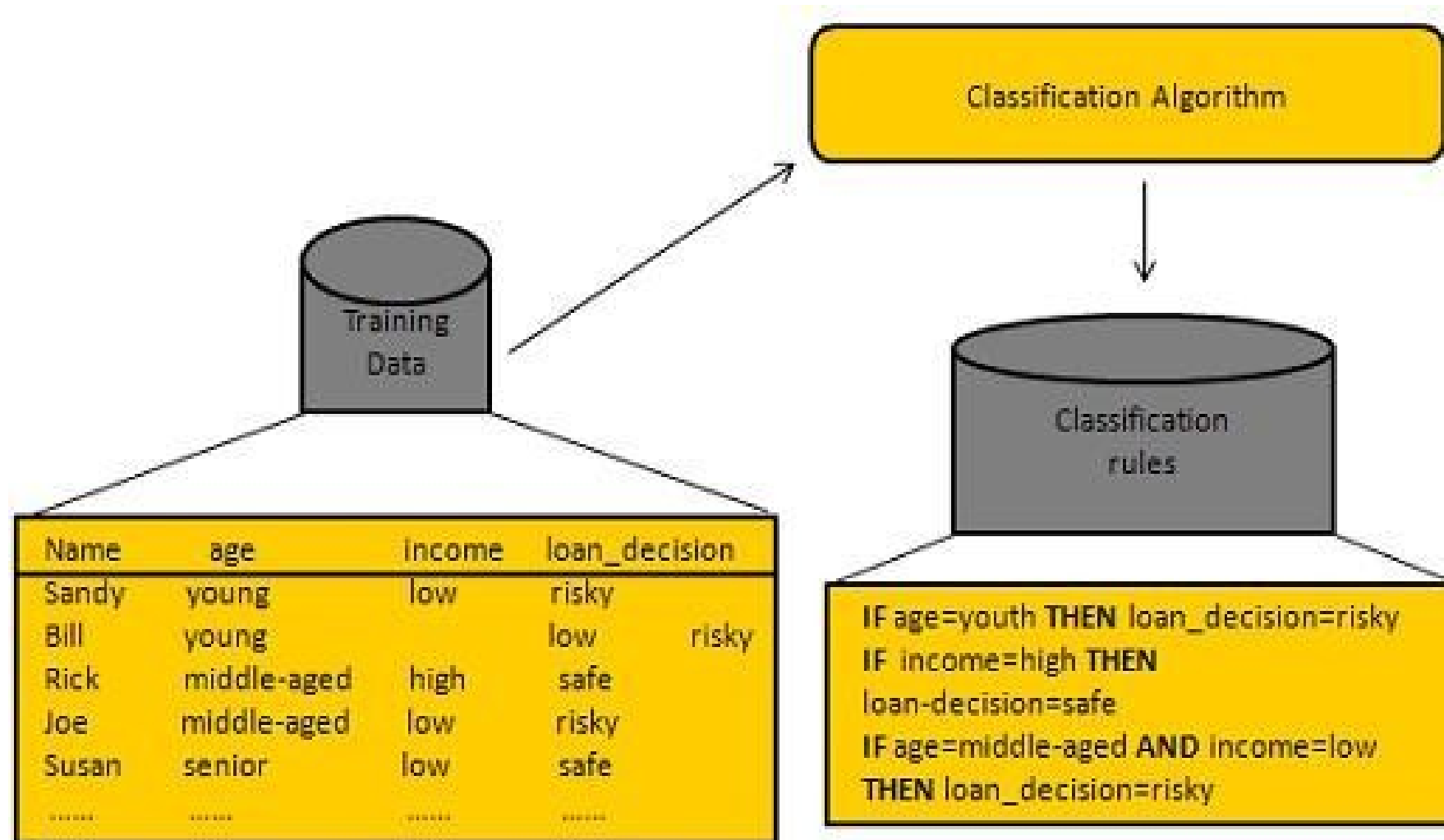


Building the Classifier or Model

This step is the learning step or the learning phase.

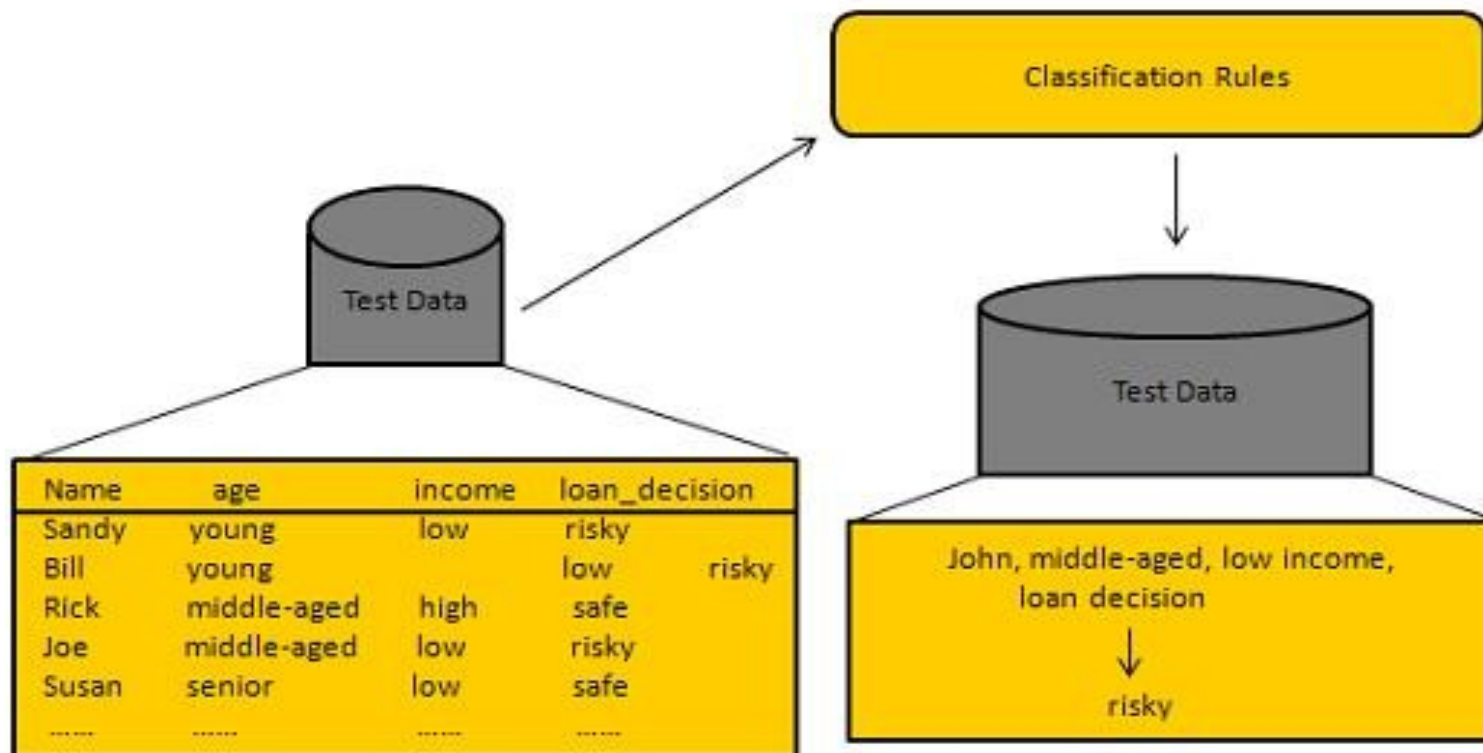
- In this step the classification algorithms build the classifier.
- The classifier is built from the training set made up of database tuples and their associated class labels.
- Each tuple that constitutes the training set is referred to as a category or class.
 - These tuples can also be referred to as sample, object or data points.

Building the Classifier



Using Classifier for Classification

- In this step the classifier is used for classification. Here the test data is used to estimate the accuracy of classification rules.
- The classification rules can be applied to the new data tuples if the accuracy is considered acceptable.





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.



Decision Trees

- A *decision tree* (also called *prediction tree*) uses a tree structure to specify sequences of decisions and consequences.
- Given in $X = \{x_1, x_2, \dots, x_n\}$, the goal is to predict a response or output variable Y .
- Each member of the set $X = \{x_1, x_2, \dots, x_n\}$ is called an *input variable*.
- The prediction can be achieved by constructing a decision tree with test points and branches.
- At each test point, a decision is made to pick a specific branch and traverse down the tree.
- Eventually, a final point is reached, and a prediction can be made.
- Each test point in a decision tree involves testing a particular input variable (or attribute), and each branch represents the decision being made. Due to its flexibility and easy visualization, decision trees are commonly deployed in data mining applications for classification purposes.



Decision Trees

- The input values of a decision tree can be categorical or continuous.
- A decision tree employs a structure of test points (called *nodes*) and branches, which represent the decision being made.
- A node without further branches is called a *leaf node*. The leaf nodes return class labels and, in some implementations, they return the probability scores. A decision tree can be converted into a set of decision rules.

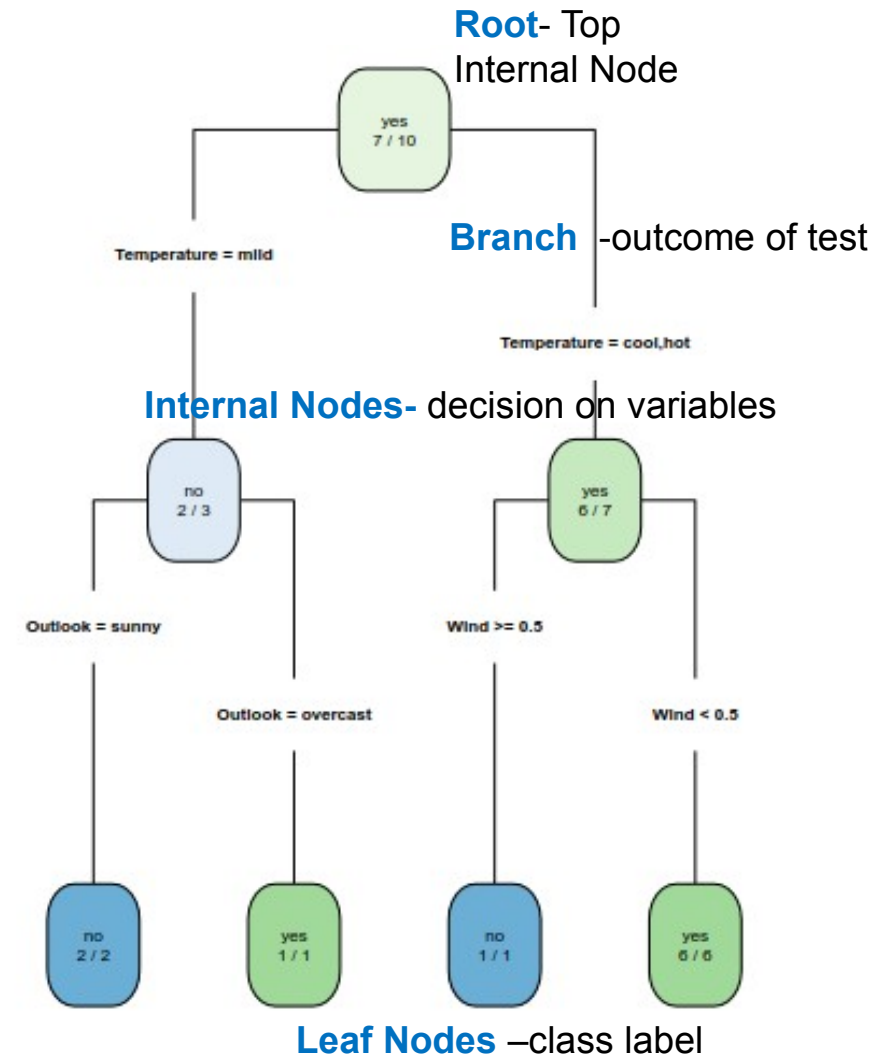


Decision Tree Classifier - What is it?

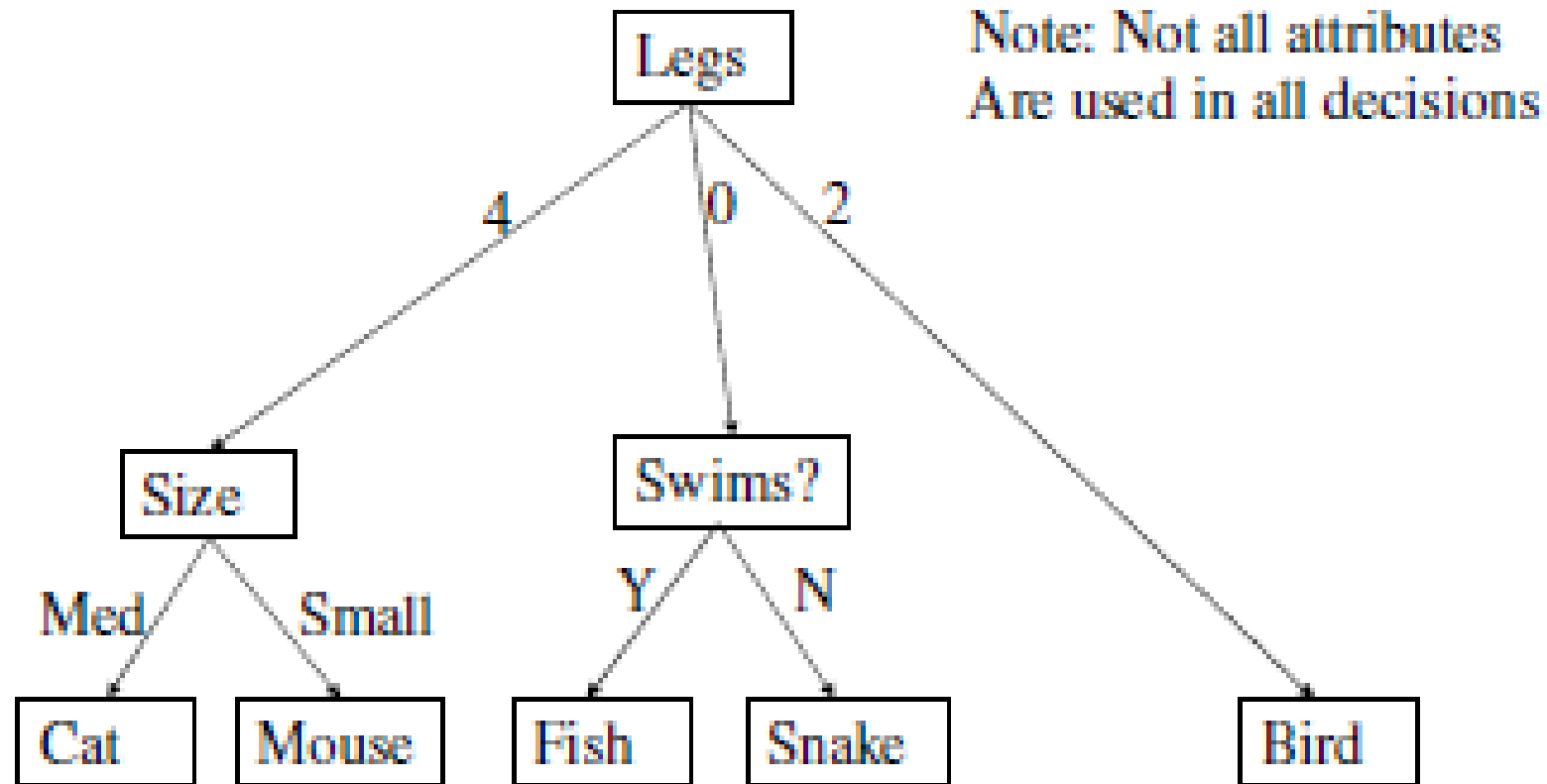
- Used for classification:
 - Returns probability scores of class membership
 - Assigns label based on highest scoring class
- **Input** variables can be continuous or discrete
- **Output:**
 - A tree that describes the decision flow.
 - Leaf nodes return either a probability score, or simply a classification.
 - Trees can be converted to a set of "decision rules"
 - "IF income < \$50,000 AND mortgage_amt > \$100K THEN default=T with 75% probability"

Example of Decision tree: Weather Data

Outlook	Temperature	Humidity	Wind	Play
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	hot	high	FALSE	yes
sunny	mild	high	FALSE	no
rainy	cool	normal	FALSE	yes
sunny	cool	normal	FALSE	yes
rainy	cool	normal	FALSE	yes
sunny	hot	normal	FALSE	yes
overcast	mild	high	TRUE	yes
sunny	mild	high	TRUE	no



A Decision Tree



Overview of a Decision Tree

- Figure 7-1 shows an example of using a decision tree to predict whether customers will buy a product.

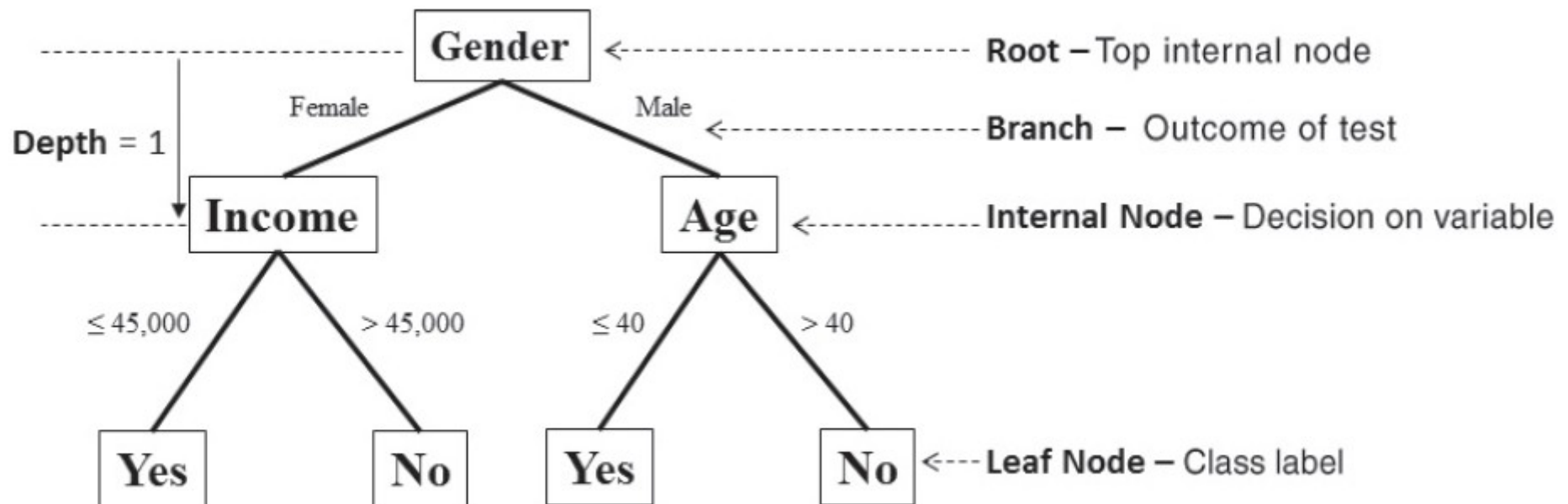


FIGURE 7-1 Example of a decision tree



Decision Trees in R

- In R, *rpart* is for modeling decision trees, and an optional package *rpart.plot* enables the plotting of a tree.
- You have an example of how to use decision trees in R with *rpart.plot* to predict whether to play golf given factors such as weather outlook, temperature, humidity, and wind.
- In R, first initialize the packages.
- `install.packages("rpart.plot")` # install package *rpart.plot*
- `library("rpart")` # load libraries
- `library("rpart.plot")`



Decision Trees in R

- The CSV file contains five attributes: *Play*, *Outlook*, *Temperature*, *Humidity*, and *Wind*. *Play* would be the output variable (or the predicted class), and *Outlook*, *Temperature*, *Humidity*, and *Wind* would be the input variables.
- In R, read the data from the CSV file in the working directory and display the content.
- `play_decision <- read.table("DTdata.csv",header=TRUE,sep=",")`
- `play_decision`
- Display a summary of *play_decision*.
`summary(play_decision)`



Decision Trees in R

- The following code snippet shows how to use the `rpart` function to construct a decision tree.

```
fit <- rpart(Play ~ Outlook + Temperature + Humidity + Wind, method="class", data=play_decision,  
control=rpart.control(minsplit=1), parms=list(split='information'))
```

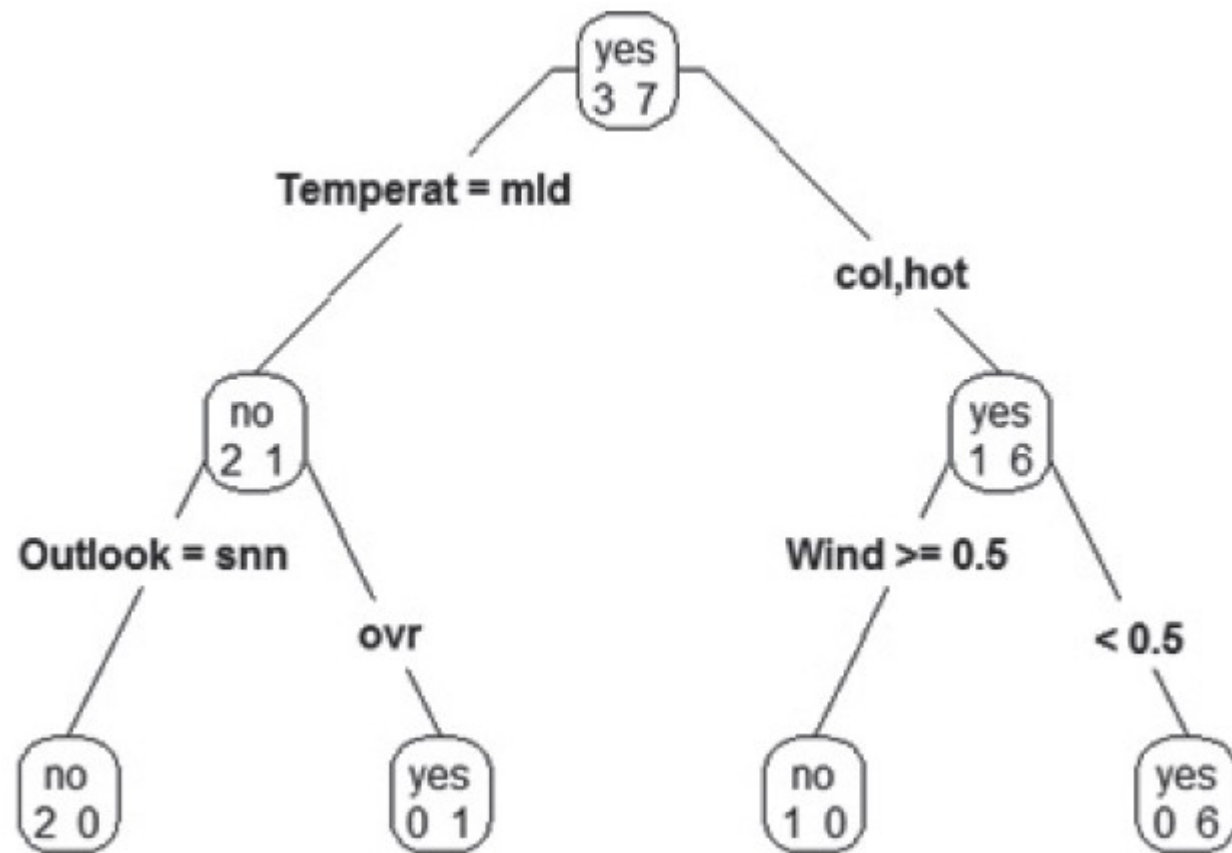
- The `rpart` function has four parameters. The first parameter, `Play ~ Outlook + Temperature + Humidity + Wind`, is the model indicating that attribute *Play* can be predicted based on attributes *Outlook*, *Temperature*, *Humidity*, and *Wind*.
- The second parameter, *method*, is set to “class,” telling R it is building a classification tree.
- The third parameter, *data*, specifies the dataframe containing those attributes mentioned in the formula.



Decision Trees in R

- Enter **summary(fit)** to produce a summary of the model built from rpart.
- The output produced by the summary is difficult to read and comprehend. The `rpart.plot()` function from the `rpart.plot` package can visually represent the output in a decision tree.
- Enter the following R code to plot the tree based on the model being built. The resulting tree is shown in Figure 7-9. Each node of the tree is labeled as either yes or no referring to the *Play* action of whether to play outside. Note that, by default, R has converted the values of *Wind* (True/False) into numbers.
- `rpart.plot(fit, type=4, extra=1)`

Decision Trees in R





Decision Trees in R

- The decisions in Figure 7-9 are abbreviated. Use the following command to spell out the full names and display the classification rate at each node.
- `rpart.plot(fit, type=4, extra=2, clip.right.labs=FALSE, varlen=0, faclen=0)`



Decision Trees in R

- The decision tree can be used to predict outcomes for new datasets. Consider a testing set that contains the following record.
- Outlook="rainy", Temperature="mild", Humidity="high", Wind=FALSE
- The goal is to predict the play decision of this record. The following code loads the data into R as a data frame *newdata*. Note that the training set does not contain this case.
- ```
newdata <- data.frame(Outlook="rainy", Temperature="mild",
Humidity="high", Wind=FALSE)
```
- ```
newdata
```



Decision Trees in R

- Next, use the *predict* function to generate predictions from a fitted *rpart* object. The format of the predict function follows.
- `predict(object, newdata = list(), type = c("vector", "prob", "class", "matrix"))`
- Parameter *type* is a character string denoting the type of the predicted value. Set it to either `prob` or `class` to predict using a decision tree model and receive the result as either the class probabilities or just the class. The output shows that one instance is classified as *Play=no*, and zero instances are classified as *Play=yes*. Therefore, in both cases, the decision tree predicts that the play decision of the testing set is not to play.
- `predict(fit,newdata=newdata,type="prob")`
- `predict(fit,newdata=newdata,type="class")`