



Data Science: Algorithms-1

Won Kim
2022



Roadmap: End-to-End Process

- 1. Objective Setting
- 2. Data Curation
- 3. Data Inspection
- 4. Data Preparation
- 5. **Analysis Algorithms**
- 6. Evaluation
- 7. Deployment



Grouping Data Mining Algorithms: in terms of supervised or unsupervised

- Supervised Learning
 - Build a model by “learning” from a training dataset (dataset with correct labels/answers)
 - Use the model to “predict” the correct label/answer for new data
- Unsupervised Learning
 - There is no training dataset.
 - Just divide a given dataset into multiple groups based on some measure of similarity.



Supervised Learning Algorithms

- Decision Trees
- Random Forests
- k-nearest Neighbors
- Regression
- Logistic Regression
- Support Vector Machines
- Naive Bayes
- Neural Networks
- Similarity Learning



Unsupervised Learning Algorithms

- Clustering
- Topic Models
- Hidden Markov Models
- Neural Networks



Grouping Data Mining Algorithms: in terms of data mining tasks

- Regression
 - Discover the best-fit function for data ($Y = aX + b$)
- Classification
 - Predict the class to which a new data belongs
- Clustering
 - Divide a dataset into related subgroups
 - Outlier Detection
 - Discover exceptional data
- Association Rules Discovery
 - rules for discovering related data



Data Mining Algorithms

- Regression
 - linear regression, multiple linear regression
 - non-linear regression
- Classification
 - neural networks, decision trees, Bayesian networks, support vector machines, k-nearest neighbors
- Clustering
 - hierarchical, centroid, statistical distribution, density, subspace, group, graph-based cluster model
 - linkage, k-means, EM (expectation maximization), DBSCAN, OPTICS, clique, SOM
- Association Rules Discovery
 - apriori algorithm, FP (frequent pattern) tree



Artificial Neural Networks

- Lots of artificial neural networks (ANN)
- Some popular neural networks
 - feedforward NN (multi-layer perceptrons)
 - convolutional NN(CNN)
 - for image classification
 - recurrent NN(RNN) (LSTM)
 - for natural language processing and sequence data
 - generative adversarial networks (GAN) (StyleGAN)
 - for generation of images, audio, text



Artificial Neural Networks

- Some popular neural networks (cont'd)
 - autoencoders
 - for dimensionality reduction, generation of images, image inpainting
 - radial basis function(RBF) NN
 - time series prediction, classification, function approximation
 - Kohonen self-organizing map(SOM)
 - clustering
 - modular NN



How to Select the Right Data Mining Algorithm?

- Not easy; need to try several
- Many factors need to be considered
 - data mining purpose
 - amount of dataset
 - availability of a training dataset
 - bias and variance of the algorithm
 - dimensionality of the input dataset
 - noise in the training set
 - ...more



Data Mining SW

- Open Source

- Pandas, Scikit-Learn, TensorFlow, Pytorch, R
- Orange, UIMA, Weka, Carrot2, JHelpWork, ELKI, KNIME,...

- Products

- IBM/SPSS Modeler, Microsoft Analysis Services, Oracle Data Mining, SAS Enterprise Miner, (StatSoft) Statistica Data Miner, Angoss Knowledge Studio, (Actuate) BIRT Analytics, Clarabridge, E-NI, KXEN Modeler, LIONsolver,...

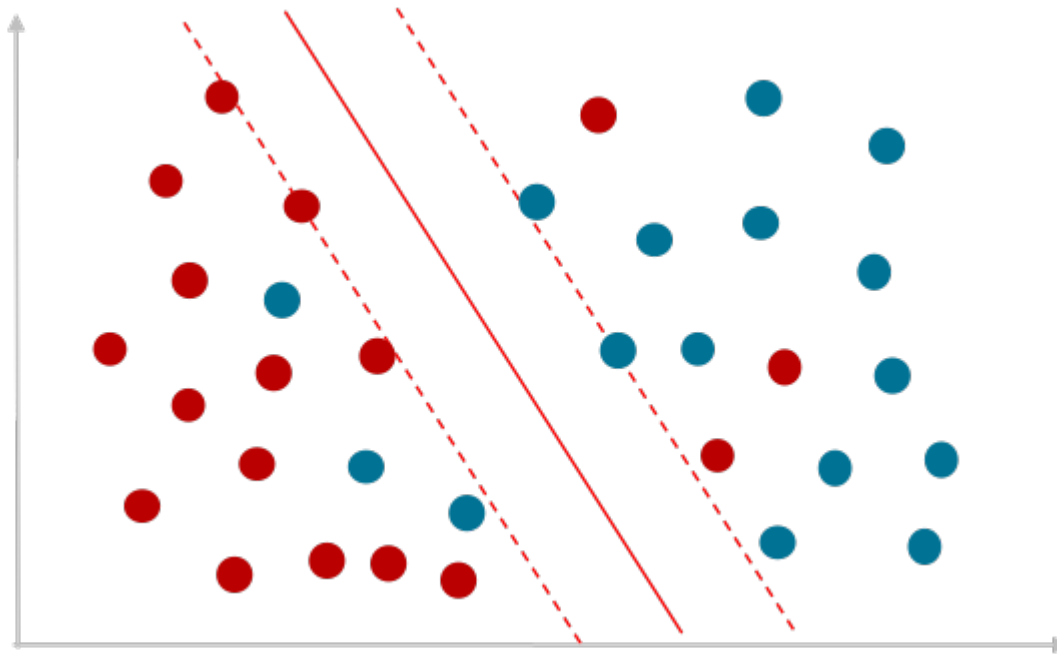


Data Mining Tasks

- Regression
- **Classification**
- Clustering
- Outlier Detection

Classification

- Among existing subgroups of data, predicting the subgroup (class) to which a new data belongs.





Classification and Estimation Problems

- Given (training data)
 - a finite set of (input) attributes (features)
 - a target attribute (the goal)
- Goal (predict)
 - Predict the value of the goal given the values of new input attributes
- If the goal is a discrete variable, the problem is a classification problem
- If the goal is a continuous variable, the problem is an estimation problem



Classification vs. Regression

- Classification
 - Build a model to predict the **class** (categorical data) of new data
- Regression
 - Build a model to predict the **value** (continuous data) of new data



Classification

- Often, data needs to be classified
 - categorical (e.g.: ages 10s, 20s, 30s,...)
 - ordinal (e.g.: clothes sizes S, M, L, XL)
 - integer-valued
 - real-valued



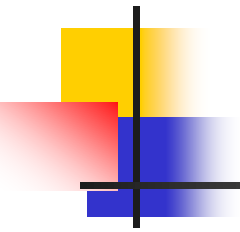
Classification Applications

- Computer vision
 - Medical imaging
 - Optical character recognition
 - Video tracking
- Pattern recognition
 - Speech, biometric, handwriting
- Internet search engines
- Natural language processing
- Drug discovery
- Document classification
- Biological classification
- Credit scoring
- ...



Classification Algorithms

- Decision Trees
- K-Nearest Neighbors
- Logistic Regression
- Support Vector Machines
- Neural Network
- Gaussian Mixture Model
- Naïve Bayes Classifier
- RBF (Radial Basis Function) Classifiers
- ...



Classifier (General)

Using Pandas and Scikit-Learn

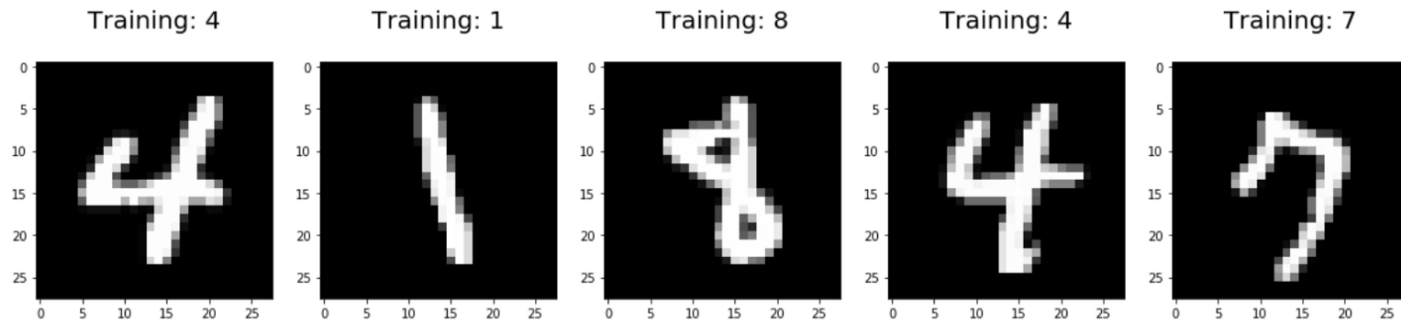


Note

- Common to all classifiers:
 - Load the dataset
 - Split the dataset (train & test sets)
 - Import the model
 - Create an instance of the model
 - Train the model (fit)
 - Test (predict using) the model
 - Evaluate the model

Logistic Regression Using Scikit-Learn

- <https://towardsdatascience.com/logistic-regression-using-python-sklearn-numpy-mnist-handwriting-recognition-matplotlib-a6b31e2b166a>
- Logistic Regression to predict digit labels based on images.



- The image shows training digits (observations) from the MNIST dataset whose category membership is known (labels 0–9).
- After training a model with logistic regression, it can be used to predict an image label (labels 0–9) given a new image.



Step 1: Set Up a (Toy) Dataset

- Load the dataset (need Anaconda)
from sklearn.datasets import load_digits
digits = load_digits()
 - Split the dataset into training and testing set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test =
train_test_split(digits.data,
digits.target, test_size=0.25, random_state=0)
- # test_size: proportion of dataset used for testing
random_state: state of pseudo random number generator



Step 2: Import the Model and Learn

- Import the classifier model
`from sklearn.linear_model import LogisticRegression`
- Make an instance of the model
`# all parameters not specified are set to their defaults`
`logisticRegr = LogisticRegression()`
- Train the model
`logisticRegr.fit(x_train, y_train)`



Step 3: Make Predictions

- Predict using the model
 - # Returns a NumPy Array
 - # Predict for One Observation (image)
`logisticRegr.predict(x_test[0].reshape(1,-1))`
- Predict multiple observations (images)
`logisticRegr.predict(x_test[0:10])`
- Predict on entire dataset
`predictions = logisticRegr.predict(x_test)`



Step 4: Evaluate

- Measure accuracy

Use score method to get accuracy of the model

```
score = logisticRegr.score(x_test, y_test)
```

```
print(score)
```



Roadmap: Classification

- Decision Trees
- K-Nearest Neighbors



Acknowledgments

- <http://www.cs.kent.edu/~jin/DM07/ClassificationDecisionTree.ppt>
- http://faculty.washington.edu/fxia/courses/LING572/decision_tree.ppt
- <https://www.dezyre.com/data-science-in-r-programming-tutorial/decision-tree-tutorial>



Example 1

- Given
 - Customer's age and salary
 - and rank
- Determine (Predict)
 - the rank of a new customer



How Is This Done?

- (Step 1) create a customer-ranking rule/model, using a training dataset (i.e., the given data)
 - age (30-39) && salary (high) → rank (good)
 - age (40-49) && salary (middle) → rank (average)
- (Step 2) using the model developed, predict the rank of a new customer
 - age (35) && salary (high) → rank (good)
 - age (25) && salary (middle) → rank (average)

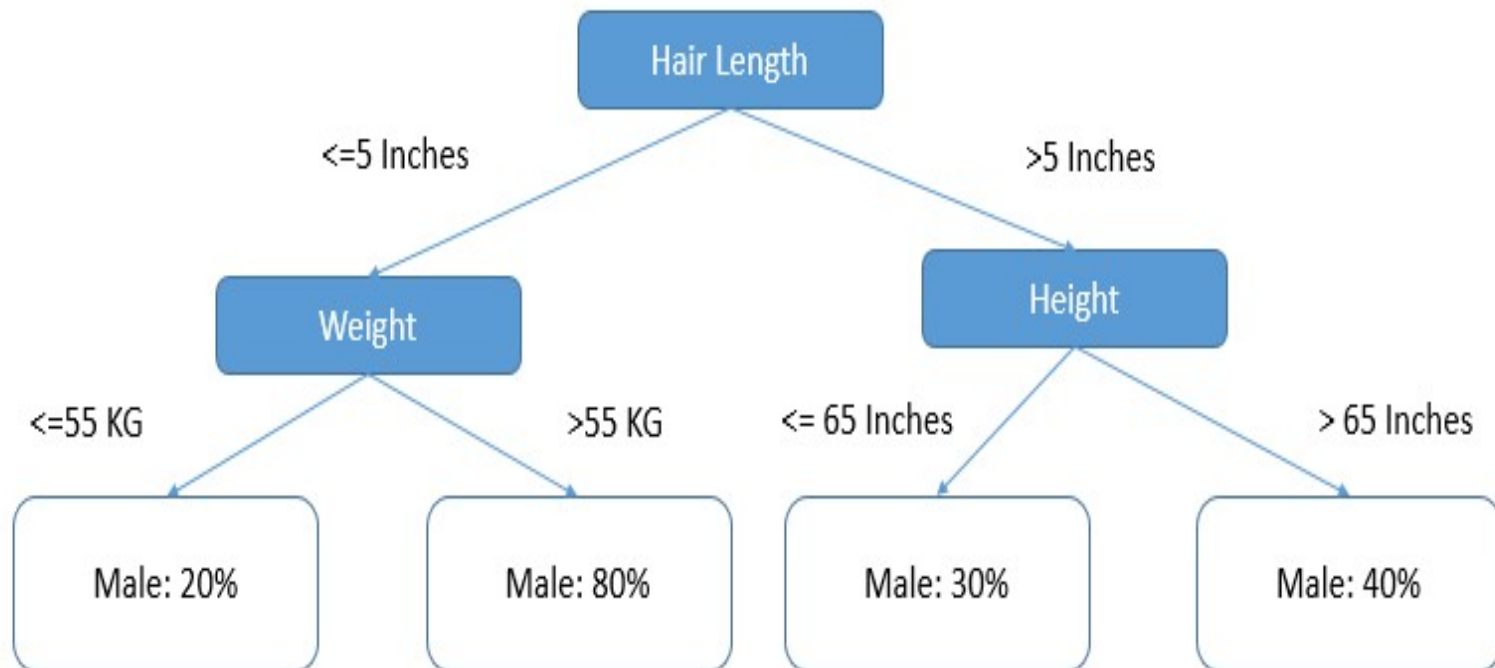


Example 2

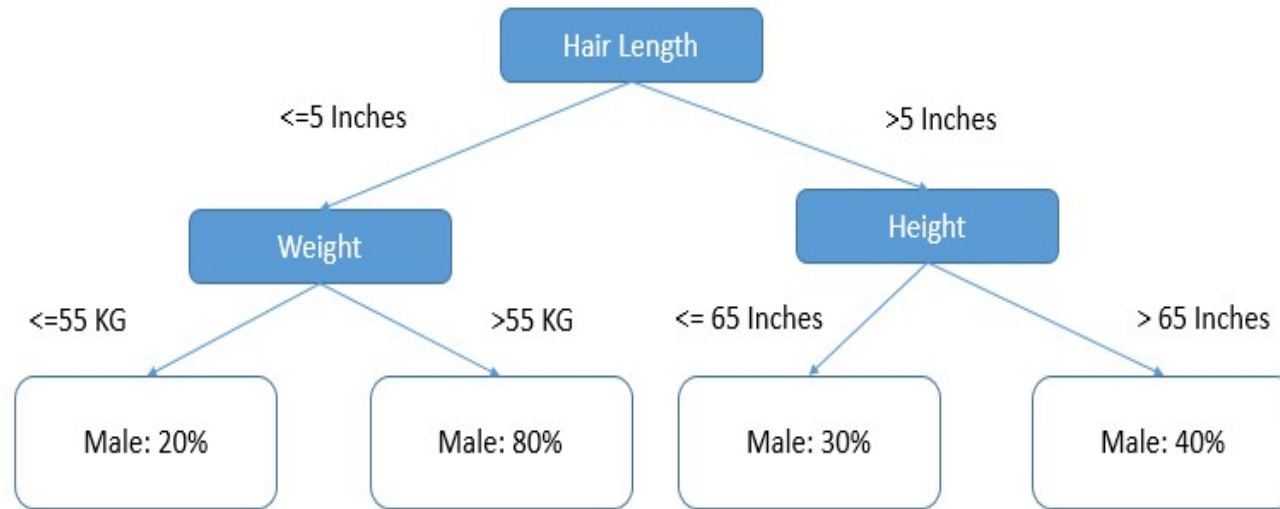
- Given
 - many person's hair length, weight, height
 - and gender
- Predict (determine)
 - the gender of a new person

How Is the Learning Model Created?

- (e.g.) A decision tree is built using data for, say, 1,000 persons
{hair length, weight, height: male/female}



How Is the Learning Model Used?



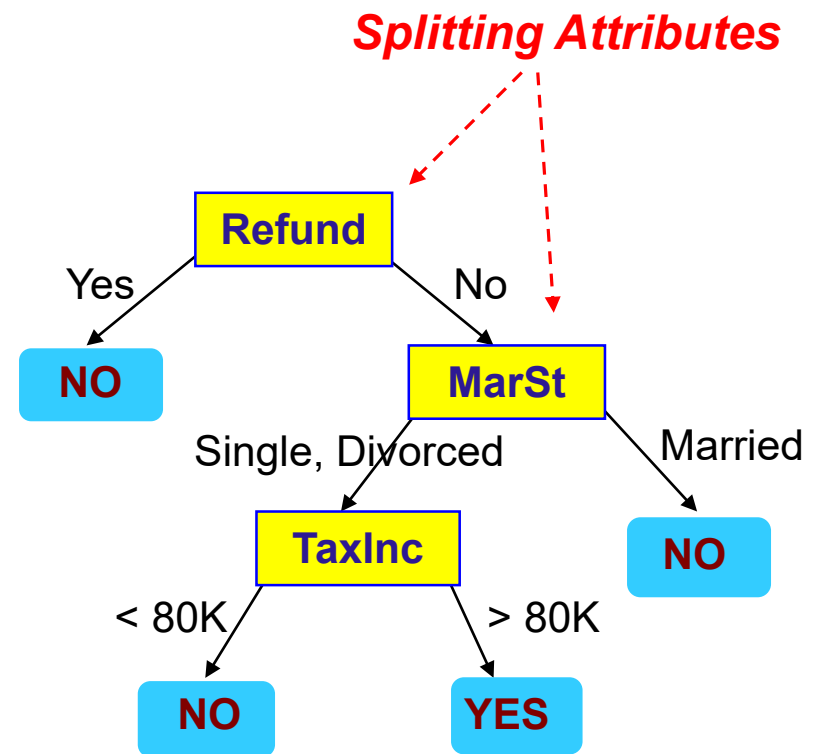
- (e.g.) Using the decision tree, it is determined that a person whose hair length is > 5 inches and height is > 65 inches is 40% probably male.
- a new person whose {hair length=5 inches, weight=63kg, height=66 inches) is 80% probably male (height ignored)

Example 3: Creating a Decision Tree

(* predicting tax cheaters *)

<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

Training Data

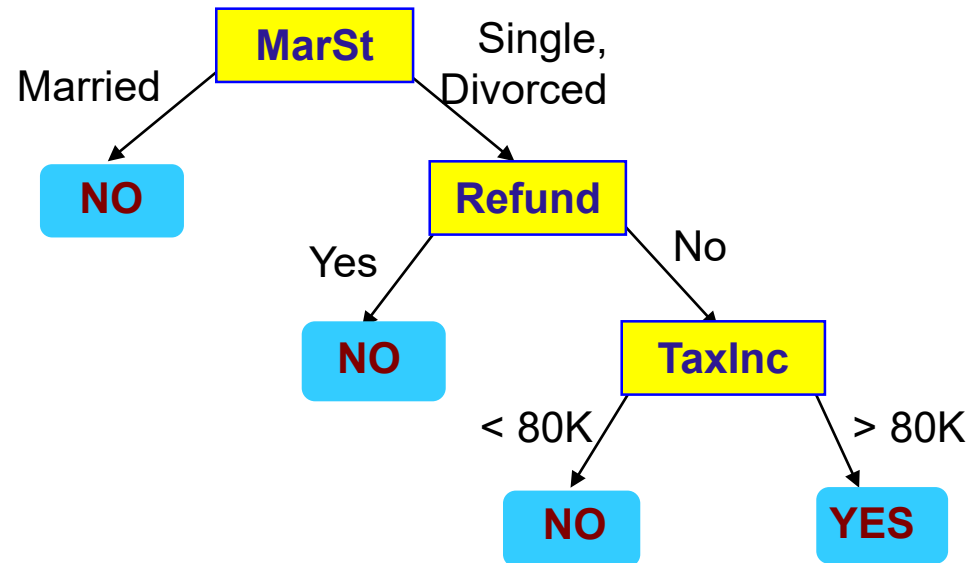


Model: Decision Tree

Example 3: Creating Another Decision Tree

categorical
categorical
continuous
target

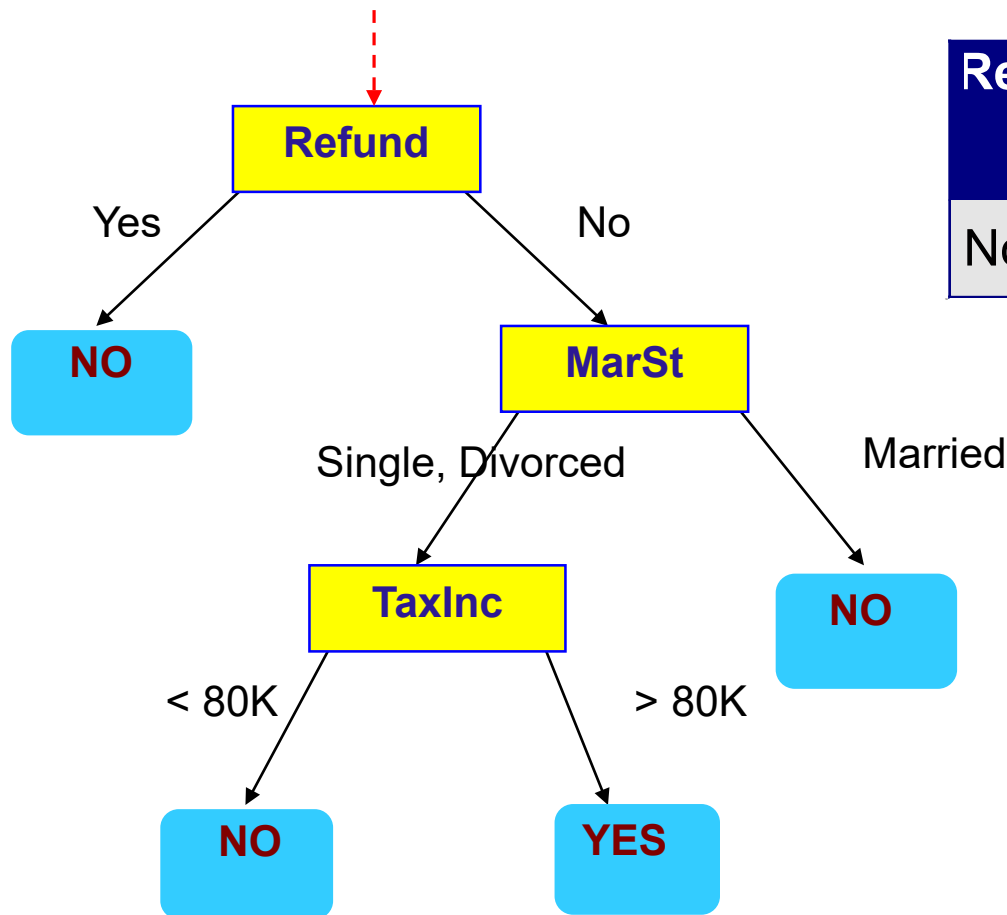
<i>Tid</i>	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



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

Applying the Model to Test Data (1/6)

Start from the root of tree.



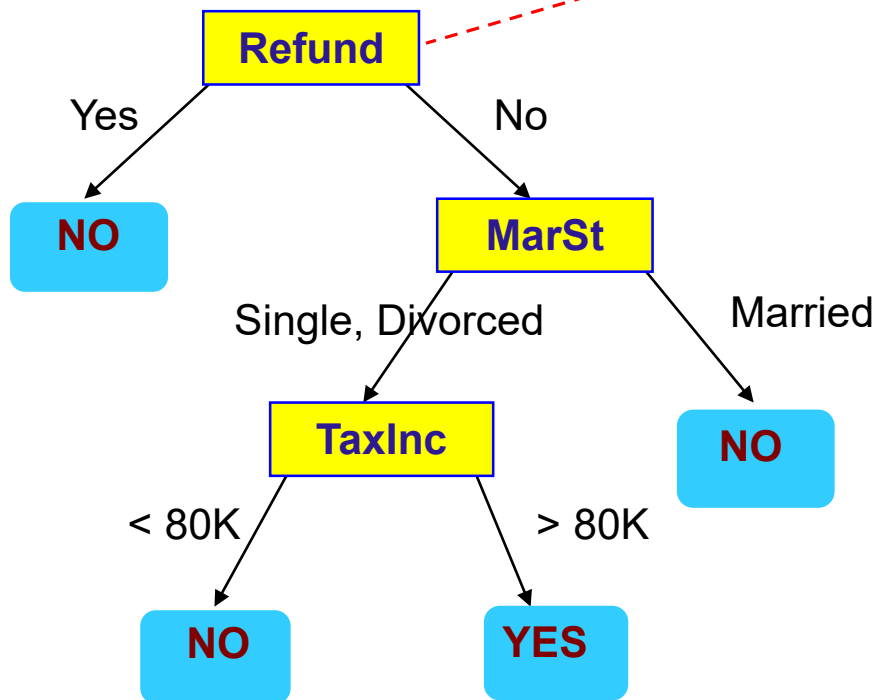
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Applying the Model to Test Data (2/6)

Test Data

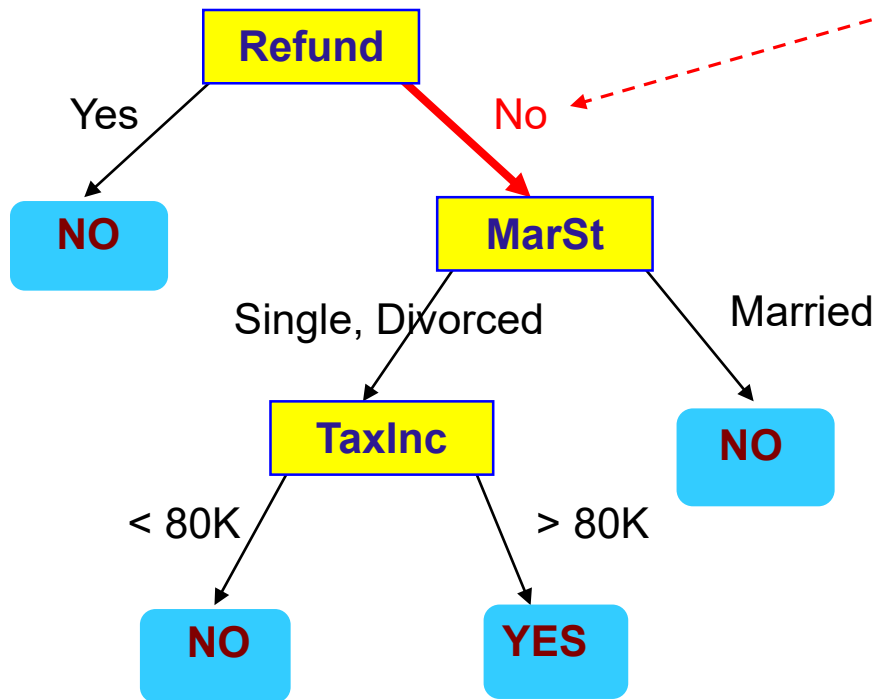
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Applying the Model to Test Data (3/6)

Test Data

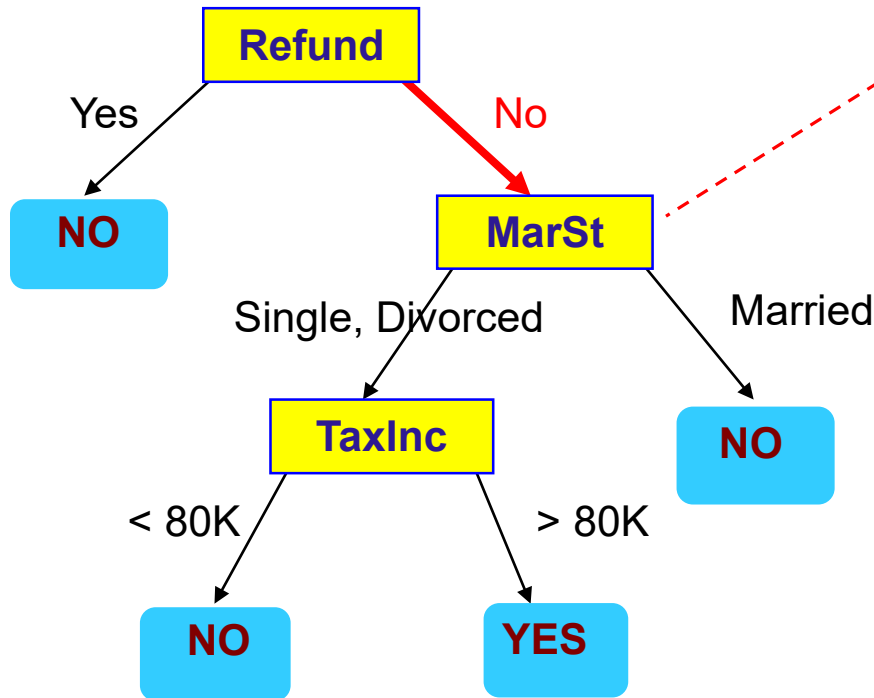
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data (4/6)

Test Data

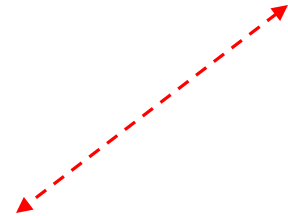
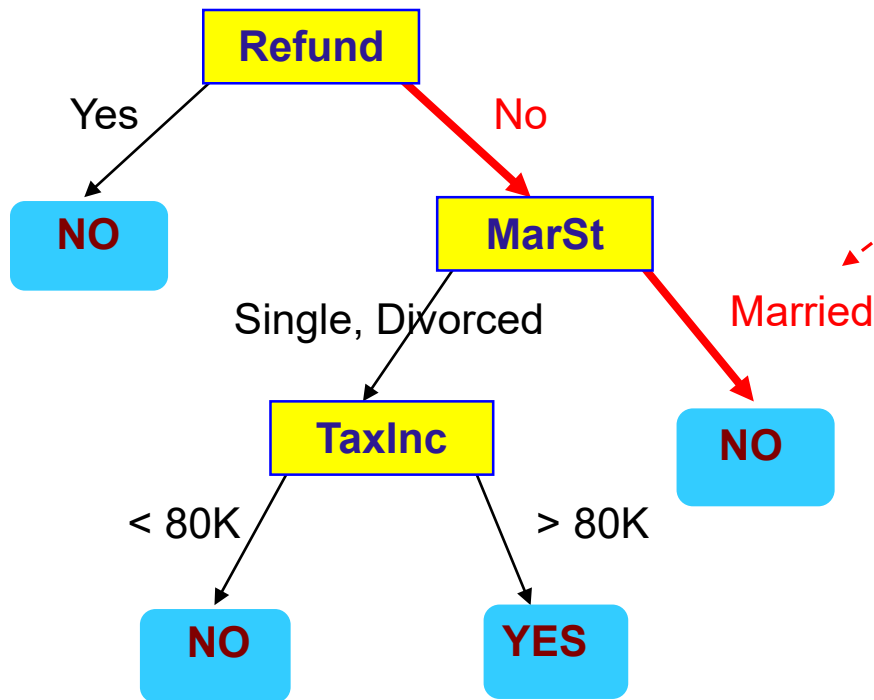
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Applying the Model to Test Data (5/6)

Test Data

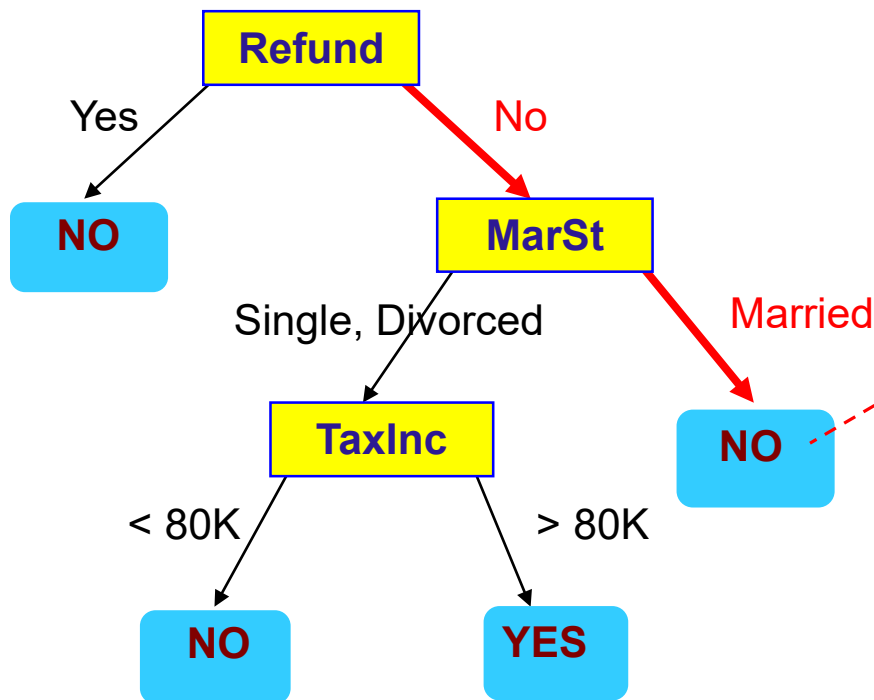
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



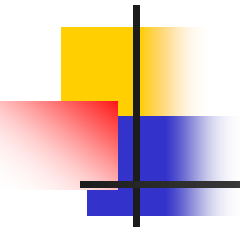
Applying the Model to Test Data (6/6)

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Assign "No" to Cheat



A Closer Look

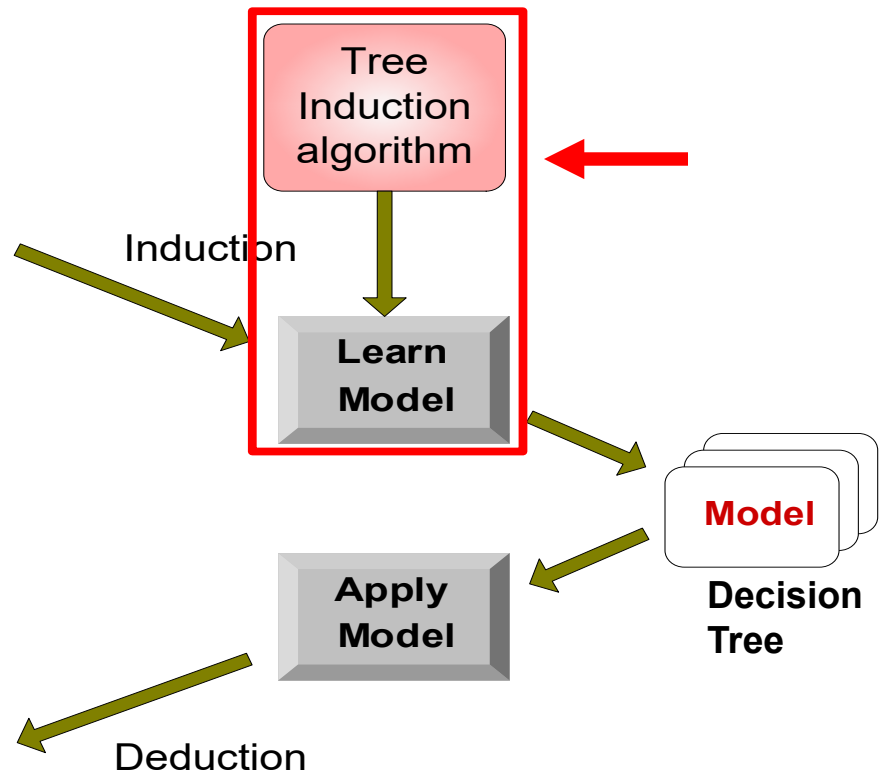
Decision Tree Classification Task

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





Tree Induction

- Find the “best” decision attribute A , and assign A as decision attribute for node.
 - The “best” attribute is selected based on a certain criterion (a measure of impurity or information gain (We will study this a little later))
- For each value of A , create a new branch, and divide up training examples.
- Repeat the process until the gain is small enough.



Key Issues in Tree Induction

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

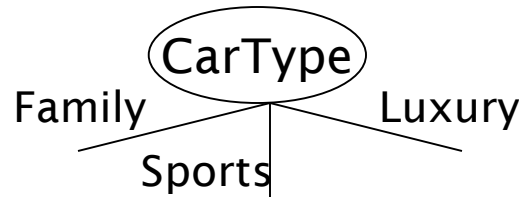


How to Specify the Test Condition?

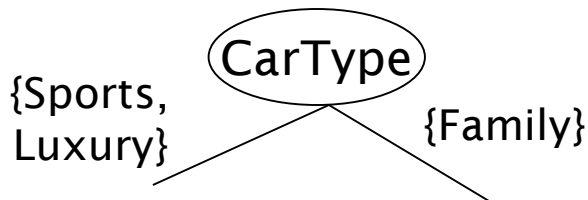
- Depends on attribute types
 - nominal (naming data without assigning order)
 - ordinal (assigning order: "S, M, L, XL")
 - continuous
- Depends on the number of ways to split
 - 2-way split
 - multi-way split

Splitting Nominal Attributes

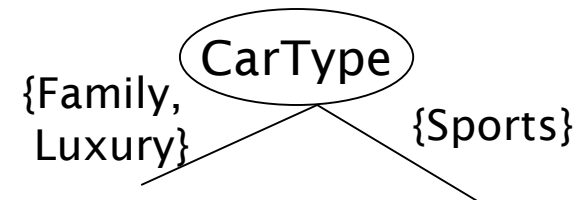
- **Multi-way split:** Use as many partitions as distinct values.



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

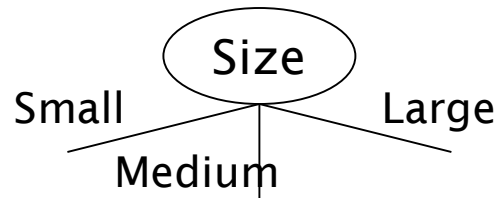


OR

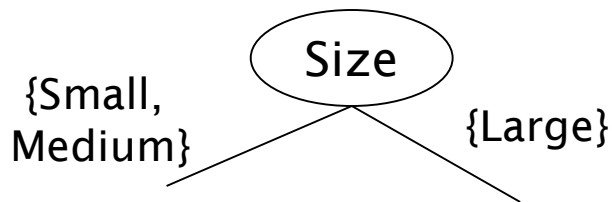


Splitting Ordinal Attributes

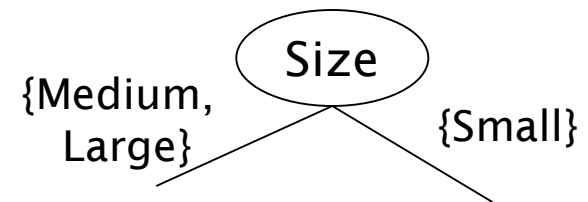
- **Multi-way split:** Use as many partitions as distinct values.



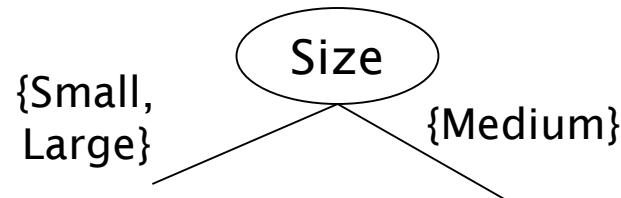
- **Binary split:** Divide values into two subsets.
Need to find optimal partitioning.



OR



- What about this split?

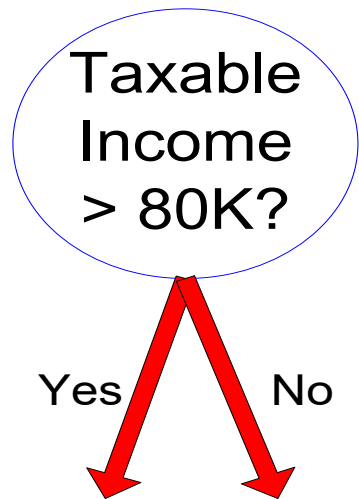




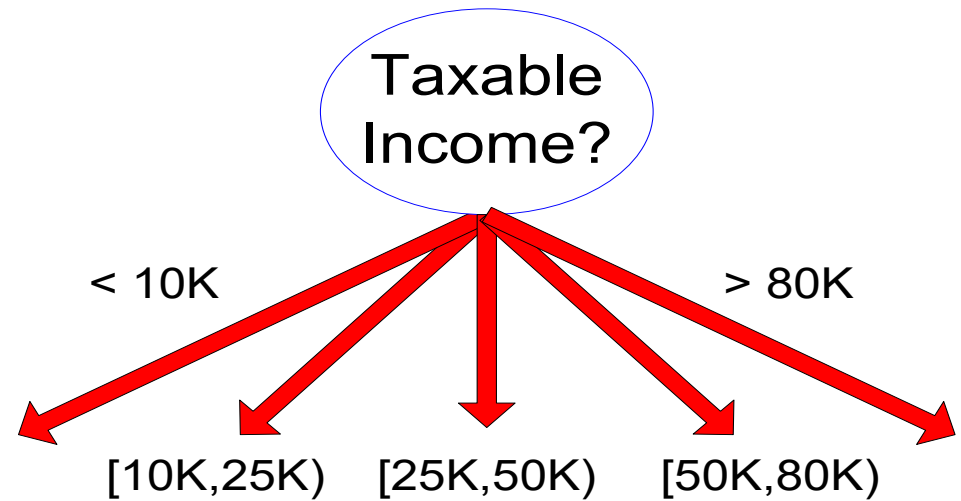
Splitting Continuous Attributes

- Two ways
 - **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 find the best cut
 - in general compute intensive

Example



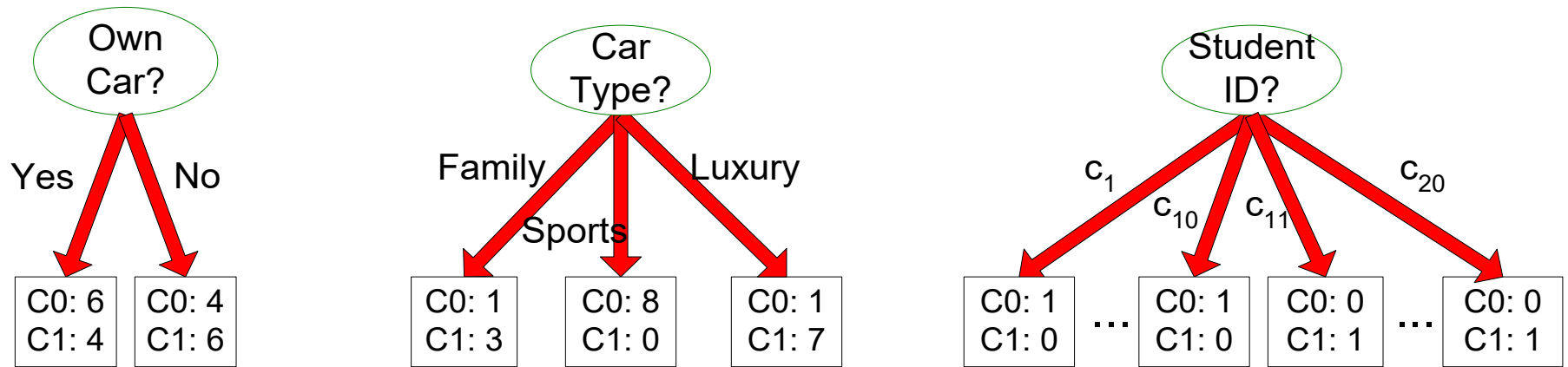
(i) Binary split



(ii) Multi-way split

Determining the Best Split (1/2)

Before Splitting: 10 records of class 0 (C0),
10 records of class 1 (C1)



Which test condition is the best?

Determining the Best Split (2/2)

- Greedy approach:
 - Nodes with **homogeneous** (i.e., pure) class distribution are preferred
- Need a measure of node impurity

C0: 5
C1: 5

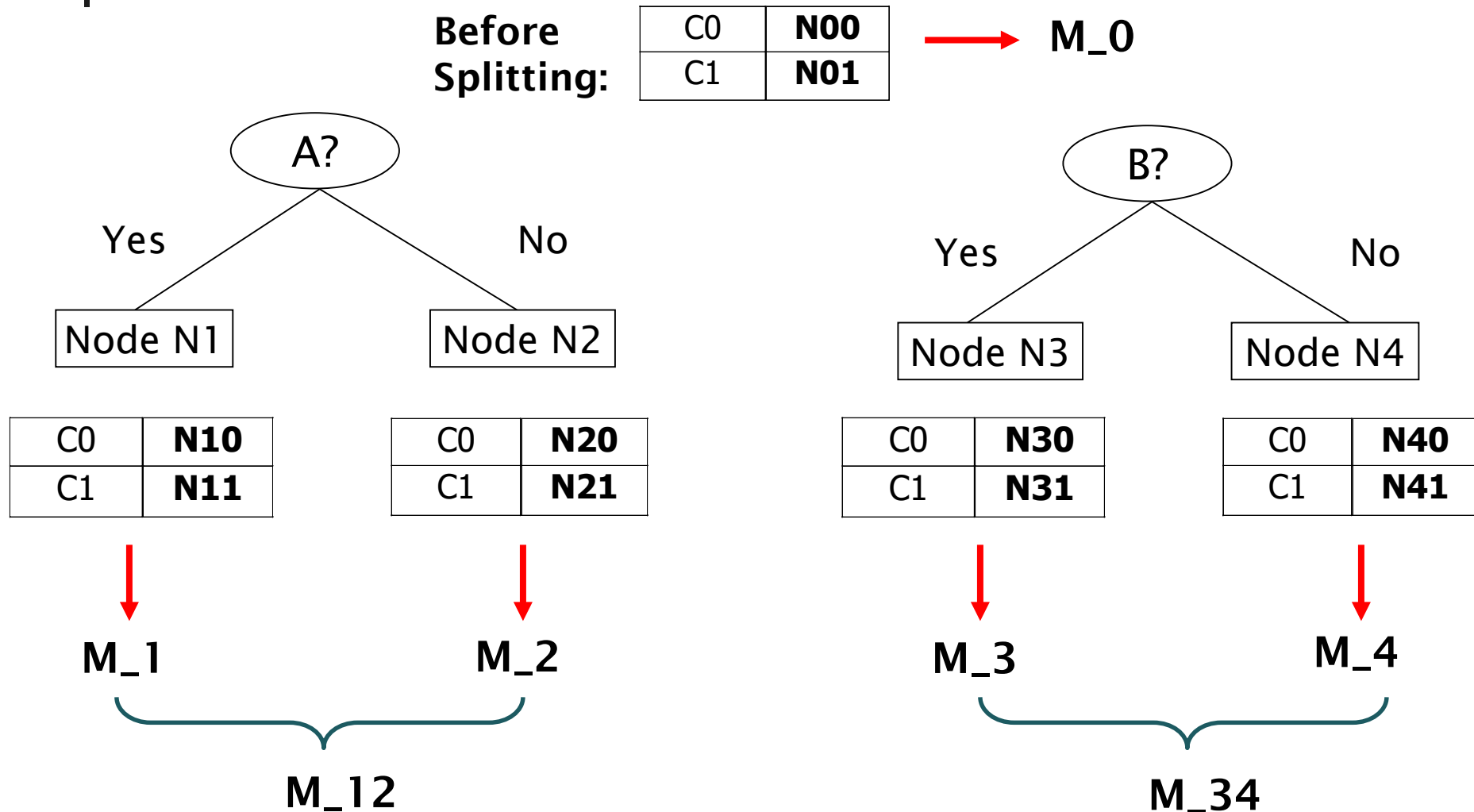
non-homogeneous,
high degree of impurity

C0: 9
C1: 1

homogeneous,
low degree of impurity

Splitting Criterion:

(Below, N_{xx} is a distribution, M_x is measure of node impurity)



$$\text{Gain} = M_0 - M_{12} \text{ vs. } M_0 - M_{34}$$



Measures (Metrics) of Node Impurity

- Entropy
 - Information Gain, Gain Ratio
- Gini Index
- Classification Error
- Variance Reduction



How to Build a Decision Tree: Walkthrough Example

- A publisher wants to know if a new novel manuscript will be a hit
- based on a past history of two authors

Novelist	Novel Genre	Hit (Yes/No)
Isaac Asimov	science fiction	yes
Isaac Asimov	fantasy	yes
Isaac Asimov	adventure	no
Isaac Asimov	science fiction	yes
Tom Clancy	science fiction	no
Tom Clancy	fantasy	no
Tom Clancy	adventure	yes

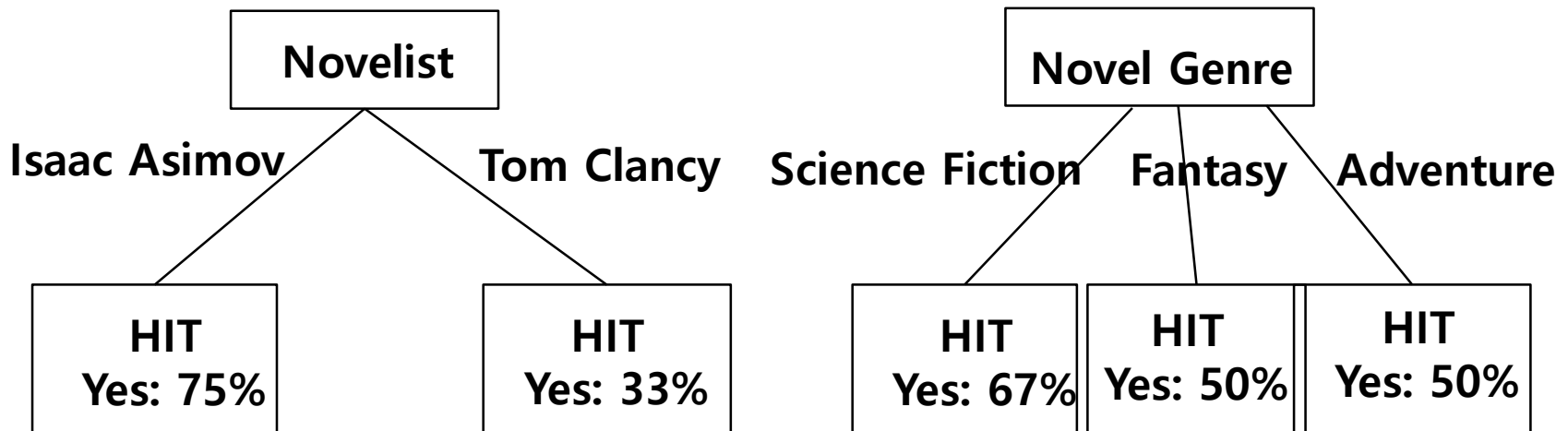


How to Build a Decision Tree

- There are 2 attributes (independent variables):
 - Novelist, Novel Genre
- Which should be chosen for the root node of the decision tree?
 - Should toss a coin to decide? **No**.
 - Must consider both.

Selecting the Root Node

- Select "Novelist"
 - Isaac Asimov: 3 out of 4 were hits (75%)
 - Tom Clancy: 1 out of 3 was a hit (33%)
- Select "Novel Genre"
 - Science Fiction: 2 out of 3 were hits (67%)
 - Fantasy or Adventure: 1 out of 2 was a hit (50%)





Choosing One

- For each root node choice,
 - 1. compute the “entropy” of the root node,
 - 2. compute the “entropy” of the next level nodes, and
 - 3. compute the difference between (1) and (2) above.
- Select the root node choice that gives the biggest “information gain” (or entropy reduction) ((3) above).



Entropy

- Entropy is a measure of “predictability” (opposite “randomness”), or “homogeneity” (of elements in a group)
- The highest randomness is the lowest predictability.
 - Entropy for tossing a coin once (50% chance of heads or tail) is 1 (using Claude Shannon’s formula). Entropy is 0 if predictability is 100%.
- (“information gain”)
 - If we have more information to consider, randomness becomes lower.



Computing Entropy: Claude Shannon (1948)

- Originally, “computing the minimum length of lossless compression encoding for data value”

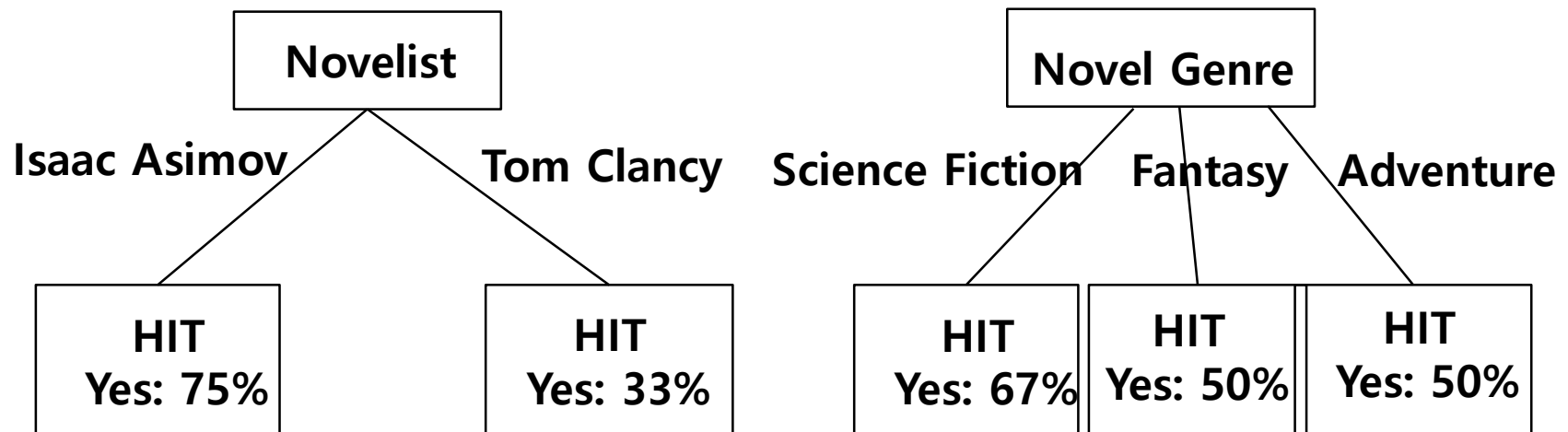
Shannon's entropy equation:

$$H(X) = - \sum_{i=0}^{N-1} p_i \log_2 p_i$$

- P_i is probability
- Entropy is the sum of $P_i * \log_2 P_i$ (i=1,n)

Illustration: Selecting the Root Node Attribute

- “Novelist” for the root node : info gain = 0.12
- “Novel Genre” for the root node : info gain = 0.01
- So, select “Novelist” for the root node.
- How do we get these numbers?





Computing the Entropy of the “Novelist” Root Node (1/3)

- Entropy of the root node = 0.98
- $entropy_{root} =$
 - *(probability of a hit-Yes among all novels*
** log(probability of a hit-Yes among all novels)*
+ probability of a hit-No
** log(probability of a hit-No among all novels))*
 - = - $(4/7 * \log(4/7) + 3/7 * \log(3/7))$
 - = - $(0.57 * \log(0.57) + 0.43 * \log(0.43))$
 - = 0.98



Computing the Entropy of the “Novelist” Root Node (2/3)

- The “Novelist” root node has two child nodes
 - The entropy of the left child node = 0.81
 - The entropy of the right child node = 0.92

$$entropy_{left-child} = -(.75 * \log(0.75) + 0.25 * \log(0.25)) = 0.81$$

$$entropy_{right-child} = -(.33 * \log(0.33) + 0.67 * \log(0.67)) = 0.92$$



Computing the Entropy of the “Novelist” Root Node (3/3)

- Information gain

- The child nodes represent additional information, and so there is information gain in splitting the root node.

$$entropy_{root} - average (\Sigma weight * child\ entropy)$$

- For the left child node, there are 4 novels, and for the right child node, there are 3 novels.

- So, the weights are 4 and 3, respectively.

- Computing the information gain

$$0.98 - (4 * entropy_{left-child} + 3 * entropy_{right-child}) / 7 \\ = 0.98 - (4 * 0.81 + 3 * 0.92) / 7 = 0.12$$

Computing the Entropy of the “Novel Genre” Root Node

- Entropy for the “Novel Genre” root node = 0.98
- Entropy for the child nodes

$$entropy_{left-child} = -(0.67 * \log(0.67) + 0.33 * \log(0.33)) = 0.92$$

$$entropy_{middle-child} = -(0.5 * \log(0.5) + 0.5 * \log(0.5)) = 1.0$$

$$entropy_{right-child} = -(0.5 * \log(0.5) + 0.5 * \log(0.5)) = 1.0$$

- information gain

$$0.98 - (3 * 0.92 + 2 * 1.0 + 2 * 1.0) / 7 = 0.01$$



Choosing the Node Attribute

- Information gain for the “Novelist” attribute as the root node = 0.12
- Information gain for the “Novel Genre” attribute as the root node = 0.01
- “Novelist” attribute has a bigger information gain, and is finally chosen.
- The choice of the nodes at each lower level of the decision tree is made similarly.



Notes

- The use of “information gain” (or entropy reduction) is very powerful, but has some problems.
- It tends to prefer splits that result in a large number of partitions, each being small but pure.
- Some decision-tree building algorithms use other criteria.
 - C4.5 uses gain ratio.
 - CART uses GINI index.



Gain Ratio

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO}$$

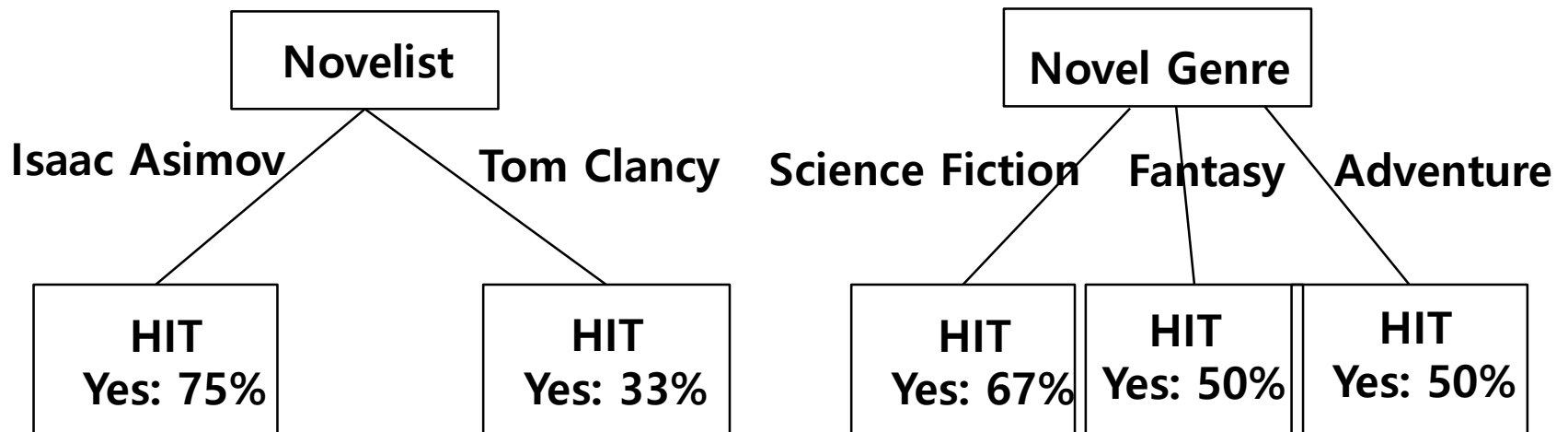
$$SplitINFO = -\sum_{i=1}^k \frac{n_i}{n} \log \frac{n_i}{n}$$

Parent node p is split into k partitions.
 n_i is the 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!

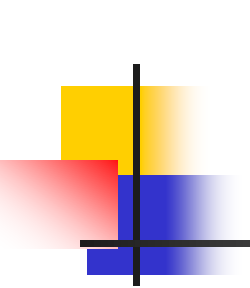
Decision Stump

- Decision tree with only the root node (and the leaf nodes)
- We saw these earlier.
- “weak” decision tree (classifier).
- But is very useful for “Ensemble Learning” (later).



Decision Tree vs. Decision Stump





For Homework: Example Dataset (* All attributes are categorical type *)

- <https://www.gyunam.com/post/decision-tree/>
- “Will a person respond to our marketing campaign?”
(based on {residential district, house type, income, previous customer, previous marketing outcome})



Decision Tree

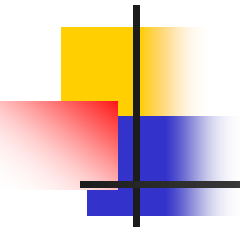
(total 14 samples)

District	House Type	Income	Previous Customer	Outcome
Suburban	Detached	High	No	Not responded
Suburban	Detached	High	Yes	Not responded
Rural	Detached	High	No	Responded
Urban	Semi-detached	High	No	Responded
Urban	Semi-detached	Low	No	Responded
Urban	Semi-detached	Low	Yes	Not responded
Rural	Semi-detached	Low	Yes	Responded
Suburban	Terrace	High	No	Not responded
Suburban	Semi-detached	Low	No	Responded
Urban	Terrace	Low	No	Responded
Suburban	Terrace	Low	Yes	Responded
Rural	Terrace	High	Yes	Responded
Rural	Detached	Low	No	Responded
Urban	Terrace	High	Yes	Not responded



Programming Homework

- Write a Python program for building a decision tree, using information gain for selecting a node.
- You may use Numpy and Pandas (but no other ML libraries).
- Use the dataset on the previous page to test your program.
- Be sure to comment the program (particularly about the node split decision, computations of entropy and information gain)



Another Look



Best Decision Tree?

- “Best”: You need a bias (e.g., prefer the “smallest” tree)
 - least depth? fewest nodes? which trees are the best predictors of unseen data?
 - Occam's Razor
 - prefer the simplest hypothesis that fits the data.
 - “Simpler solutions are more likely to be correct than complex solutions.”
- ➔ Find a decision tree that is as small as possible and fits the data



Finding a Smallest Decision Tree

- A decision tree can represent any discrete function of the inputs: $y=f(x_1, x_2, \dots, x_n)$
- The space of decision trees is too big for systemic search for a smallest decision tree.
- Solution: greedy algorithm



Strengths of Decision Trees

- Generate understandable rules
- Ease of calculation at classification time
- Can handle both continuous and categorical variables
- Can clearly indicate best attributes



Weaknesses of Decision Trees

- Greedy algorithm
 - no global optimization
- Error-prone with many attributes
 - number of training examples becomes smaller quickly in a tree with many levels/branches.
- Expensive to train
 - sorting, combination of attributes, calculating quality measures, etc.



Decision Tree Induction Algorithms

- CHAID (Chi-square Automatic Interaction Detection)
- CART (Classification And Regression Tree)
- ID3 (Iterative Dichotomizer)
- (successor to ID3) C4.5, C5.0
- MARS (Multivariate Adaptive Regression Spline)
- SLIQ, SPRINT



Key Inventors of Decision Trees

- ID3 (1975), C4.5 (1993), C5.0
 - J. Ross Quinlan
- CHAID (1980)
 - Gordon Kass
- CART (1984)
 - Leo Breiman, Jerome Friedman, Charles Stone, R. Olshen
- MARS (1991)
 - Jerome Friedman



Use of Decision Trees for Classification and Regression

- Each internal node is a test
 - Theoretically, a node can test multiple attributes
 - In most systems, a node tests exactly one attribute
- Each branch corresponds to test results
 - A branch corresponds to an attribute value or a range of attribute values
- Classification vs. regression
 - Classification: Each leaf node assigns a class
 - Regression: Each leaf node assigns a real value



Decision Tree Regression

- Decision tree can be used for regression, when the dependent variable (target class) is continuous (numbers).



Roadmap: Classification

- Decision Trees
- K-Nearest Neighbors

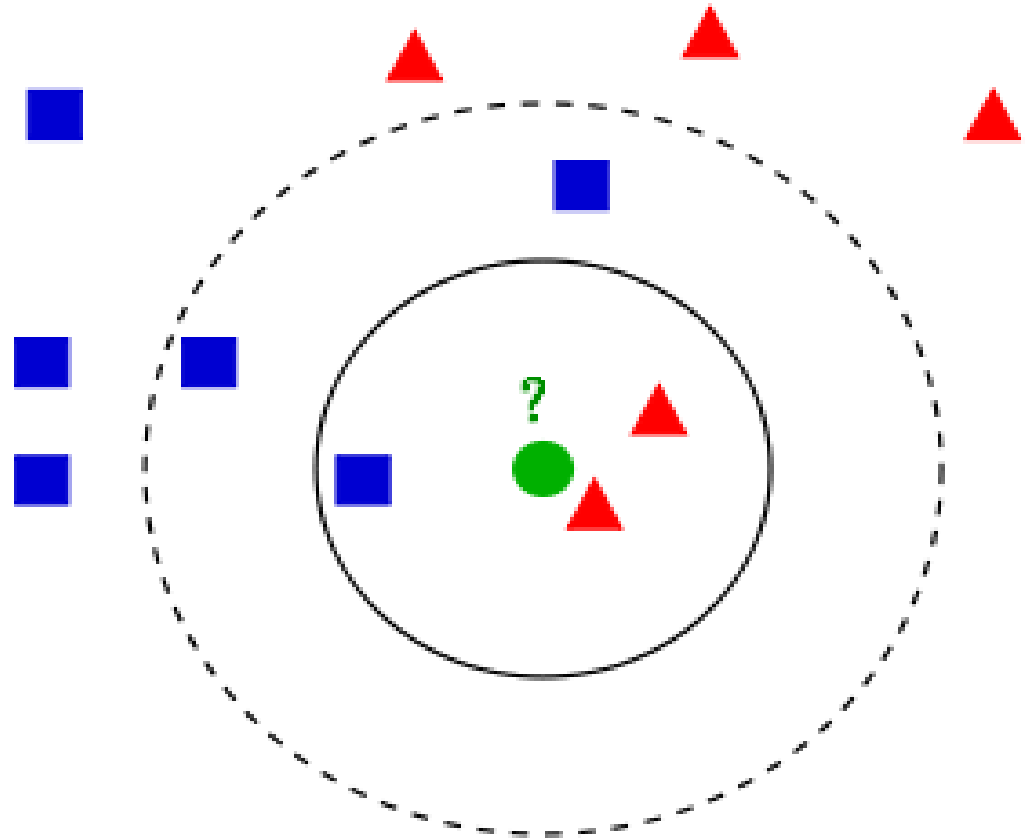


Acknowledgments

- <https://www.r-bloggers.com/k-nearest-neighbor-step-by-step-tutorial/>
- <https://bcourses.berkeley.edu/courses/1377158/files/61598112/download?verifier=5lpPFMM751cYXH0vVWOfWI2KzLhiU6pDNXeprQd5&wrap=1>

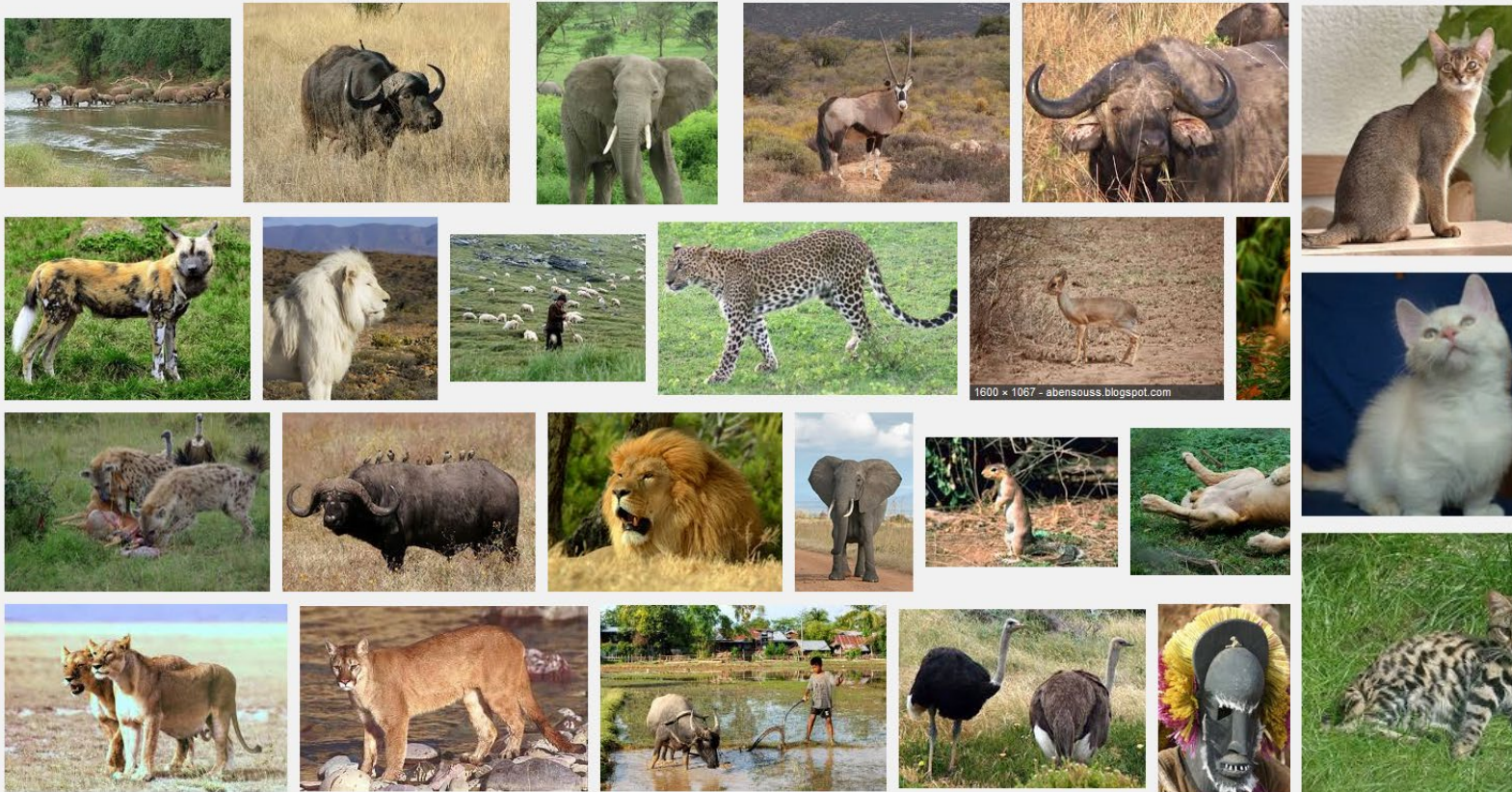
K-Nearest Neighbors Algorithm

- Sample data is classified by a majority vote of its k "nearest" neighbors.
 - $k=3$?
 - $k=5$?



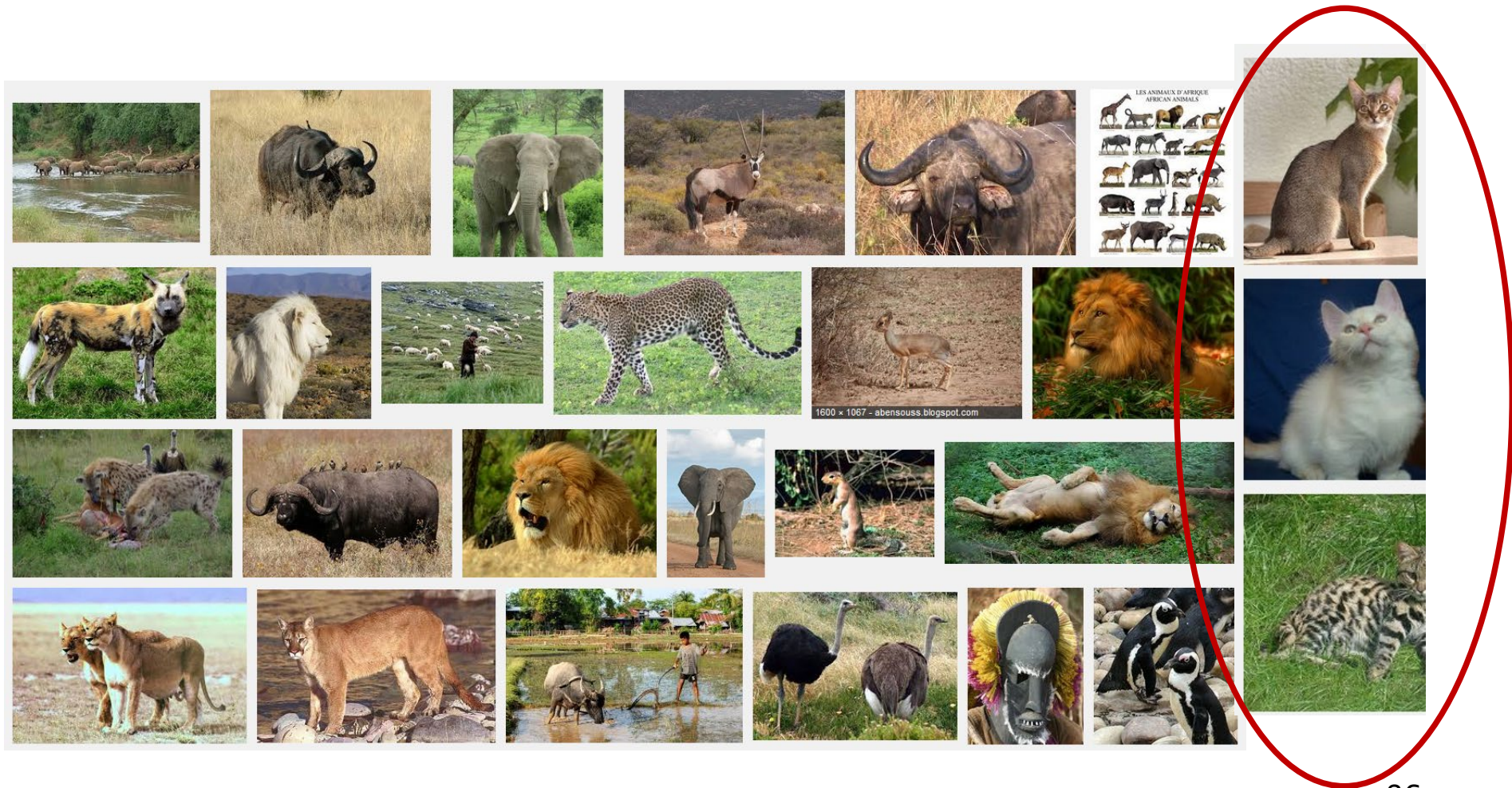
Analogy

Given a query item:
Find k closest matches
in a labeled dataset ↓



Returns the most frequent label

$k = 3$ votes for "cat"





Walkthrough Example

- Given {height, weight, T-shirt size} of some customers
- Predict the T-shirt size of a new customer (height=161cm, weight=61kg).
- {height, weight} are the features and T-shirt size is the class.



Training Dataset

HEIGHT(cm)	WEIGHT(kg)	T SHIRT SIZE
158	58	M
158	59	M
158	63	M
160	59	M
160	60	M
163	60	M
163	61	M
160	64	L
163	64	L
165	61	L
165	62	L
165	65	L
168	62	L
168	63	L
168	66	L
170	63	L
170	64	L
170	68	L



Computing Steps (1/2)

- For each record in the training dataset, compute the distance between it and the new customer data.
 - Let us use the Euclidean distance function to compute distance.

$$\begin{aligned}d(\mathbf{p}, \mathbf{q}) &= d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} \\&= \sqrt{\sum_{i=1}^n (q_i - p_i)^2}.\end{aligned}$$

- Rank each record in the training dataset
 - (1 for the closest, 2 for the second closest, etc.)



Computing Steps (2/2)

- Let us assume $k=5$.
- Search for the top 5 ranked records in the training dataset.
- Determine which class ("M", "L") is the majority.
- Assign the class to the new customer.

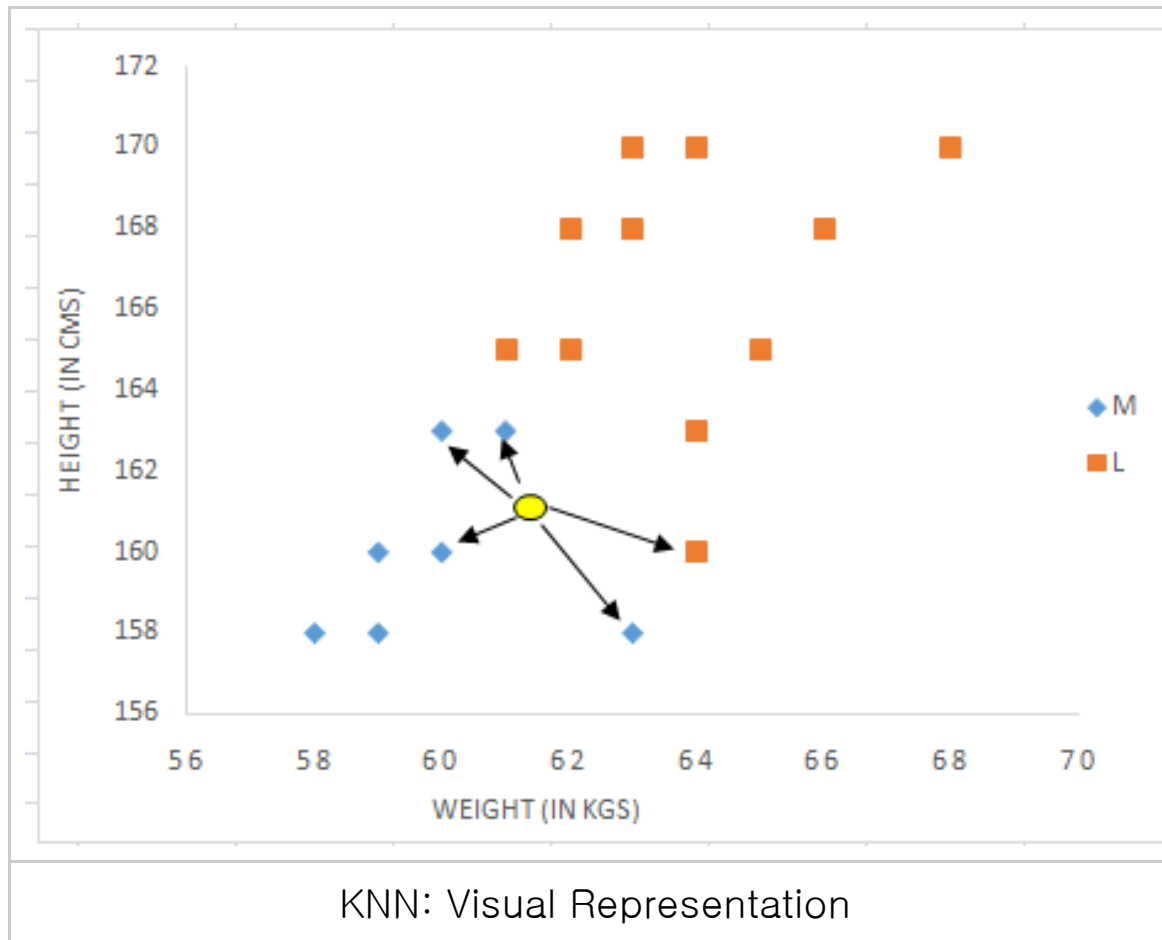
The Result

- In the screen capture, 4 of the closest records are "M".
- The predicted T-shirt size for the new customer is M.

fx		=SQRT(((\$A\$21-A6)^2+(\$B\$21-B6)^2)				
	A	B	C	D	E	
	Height (in cms)	Weight (in kgs)	T Shirt Size	Distance		
1						
2	158	58	M	4.2		
3	158	59	M	3.6		
4	158	63	M	3.6		
5	160	59	M	2.2	3	
6	160	60	M	1.4	1	
7	163	60	M	2.2	3	
8	163	61	M	2.0	2	
9	160	64	L	3.2	5	
10	163	64	L	3.6		
11	165	61	L	4.0		
12	165	62	L	4.1		
13	165	65	L	5.7		
14	168	62	L	7.1		
15	168	63	L	7.3		
16	168	66	L	8.6		
17	170	63	L	9.2		
18	170	64	L	9.5		
19	170	68	L	11.4		
20						
21	161	61				

The Result: Visual Representation

- Blue: T-shirt size M, Orange: T-shirt size L
- Yellow: new customer data





Actually, “Step 0” Was Skipped

- Before Step 1, we need to normalize the features (independent variables), using any of the normalization formulas.
- This is necessary because the two features are in different units.
 - (without normalization, the height is more dominant than the weight.)

Normalized Training Dataset

- There are two differences
 - row 5: rank changed from 3 to 4
 - row 11: now rank 5
 - row 9: not one of the top 5

	A	B	C	D	E
1	Height (in cms)	Weight (in kgs)	T Shirt Size	Distance	
2	-1.39	-1.64	M	1.3	
3	-1.39	-1.27	M	1.0	
4	-1.39	0.25	M	1.0	
5	-0.92	-1.27	M	0.8	4
6	-0.92	-0.89	M	0.4	1
7	-0.23	-0.89	M	0.6	3
8	-0.23	-0.51	M	0.5	2
9	-0.92	0.63	L	1.2	
10	-0.23	0.63	L	1.2	
11	0.23	-0.51	L	0.9	5
12	0.23	-0.13	L	1.0	
13	0.23	1.01	L	1.8	
14	0.92	-0.13	L	1.7	
15	0.92	0.25	L	1.8	
16	0.92	1.39	L	2.5	
17	1.39	0.25	L	2.2	
18	1.39	0.63	L	2.4	
19	1.39	2.15	L	3.4	
20					
21	-0.7	-0.5			



A Closer Look at the k-Nearest Neighbors Algorithm

- There really is no “training”, since training does not generate a ‘trained model’.
 - (“Testing” is done against the training dataset, not a trained model. In that sense, the dataset can be regarded as the model !)
- Accuracy generally improves with more data.
- Besides the “k”, two other choices are important:
 - weighting of neighbors (e.g. inverse distance – bigger weights for closer neighbors)
 - distance (and similarity) metric
- Testing should be done to discover the best values for the above.



K-Nearest Neighbors for Regression?

- It can be used for regression, when the dependent variable (target class) is continuous (numbers).
- In this case, the predicted value is the average of the k nearest neighbors' dependent variable values.



Summary

- Strengths

- easy to understand
- no assumptions about data

- Weaknesses

- Computing is expensive (computed against all the training dataset)
- sensitive to data scale
- not work well on skewed target variable
- not work well with a large number of independent variables (features)



The “k” in k-Nearest Algorithm

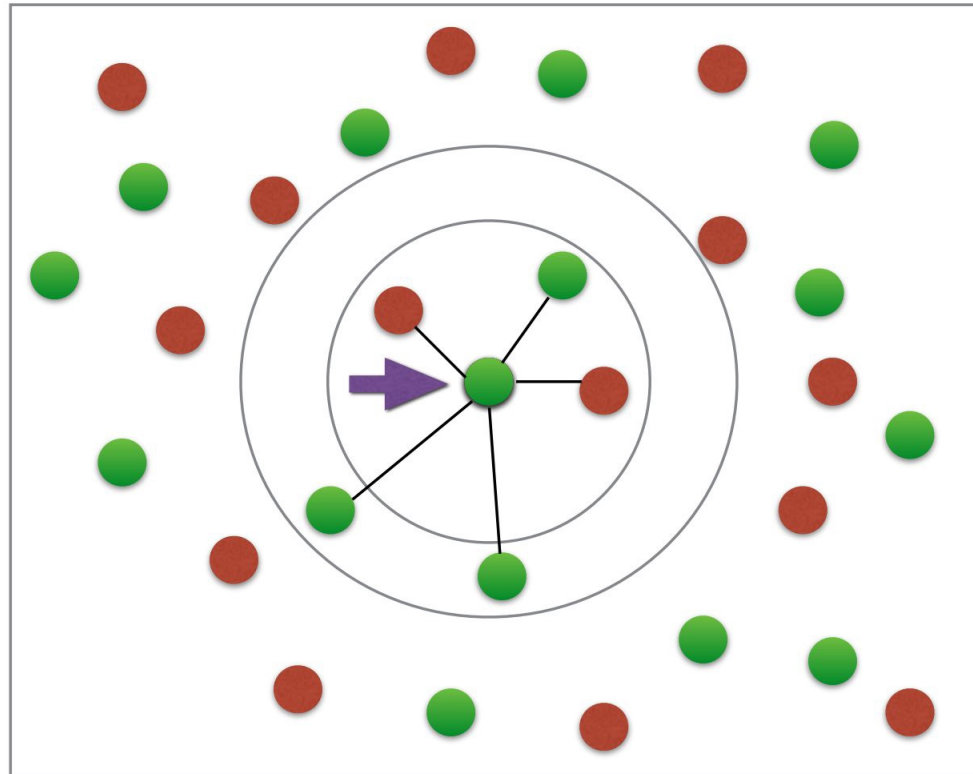
- It is called a “hyperparameter” of the algorithm.
- It must be supplied by the user (or software).
- The best “k” can only be determined through trial and error.
- It cannot be computed from the dataset.



Illustration of the Machine Learning Process and the Use of Pandas and Scikit-Learn Using K-Nearest Neighbors Classifier

Acknowledgments

- <https://towardsdatascience.com/building-a-k-nearest-neighbors-k-nn-model-with-scikit-learn-51209555453a>





Read Data

```
import pandas as pd
# read in the data using pandas
df = pd.read_csv('data/diabetes.csv')
# check if data has been read in properly
df.head()
```

	pregnancies	glucose	diastolic	triceps	insulin	bmi	dpf	age	diabetes
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1



Check Data

check number of rows and columns in the dataset
df.shape

(768, 9)

8 input features, 1 target class



Split the Data: Create Training Data

create a dataframe with all training data except the target column

```
X = df.drop(columns=['diabetes'])
```

#X contains all training data, with 9 features

#'diabetes' will be the target

check that the target variable has been removed

```
X.head()
```

	pregnancies	glucose	diastolic	triceps	insulin	bmi	dpf	age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33



Split Data: Create Target

```
# separate target values  
y = df['diabetes'].values  
# view only the first 5 target values  
y[0:5]
```

```
array([1, 0, 1, 0, 1])
```




Split Data into Training and Test Data

```
from sklearn.model_selection import  
train_test_split  
# split dataset into train and test data
```

```
X_train, X_test, y_train, y_test =  
    train_test_split(X, y,  
                    test_size=0.2,  
                    random_state=1,  
                    stratify=y)
```



Parameters of 'train_test_split'

- The first two parameters are the input and target data.
- Next, set '**test_size**' to **0.2**. This means that 20% of all the data will be used for testing, which leaves 80% of the data as training data.
- Setting '**random_state**' to **1** (random seed) ensures that we get the same split each time so we can reproduce our results.
- Setting '**stratify**' to **y** makes the training split represent the proportion of each value in the y variable. For example, in our dataset, if 25% of patients have diabetes and 75% don't have diabetes, setting 'stratify' to y will ensure that the random split has 25% of patients with diabetes and 75% of patients without diabetes.



Build the Model

```
from sklearn.neighbors import KNeighborsClassifier
```

```
# Create a KNN classifier
```

```
knn = KNeighborsClassifier(n_neighbors = 3)
```

```
# Train the KNN classifier
```

```
knn.fit(X_train,y_train)
```



Test (Predict Using) the Model

#show the first 5 model predictions on the test data

```
knn.predict(X_test[0:5])
```

```
array([0, 0, 0, 0, 1])
```

'no diabetes' for the first 4 patients



Check Model Accuracy

#check accuracy of the model on the test data

```
knn.score(X_test, y_test)
```

```
0.66883116883116878
```



Programming Homework

- Write a Python program that implements a k-nearest neighbors algorithm, and test it using the example dataset on page 86.
- You may use Numpy and Pandas (but not any other ML libraries).
- Post a single WORD file for both PHW 4-1 and PHW 4-2 containing your source code and the test result screen to CyberCampus.
- Due 9:00 p.m. the day before the next regular class.



End of Class
