Artificial Intelligence

Classification 2

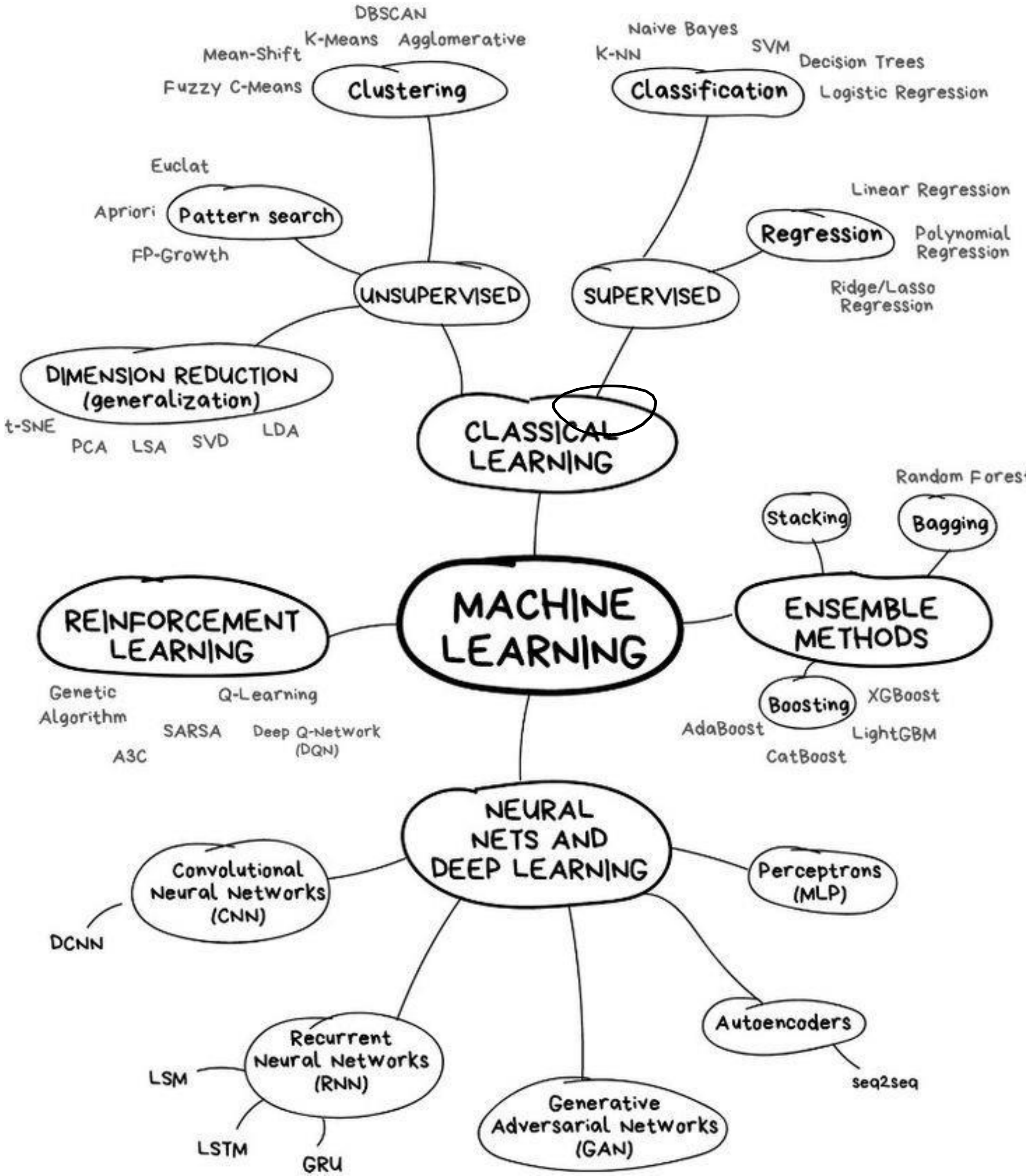
Extended from Kyuseok Shim's slides

한양대학교 ERICA
소프트웨어융합대학
COLLEGE OF COMPUTING

인공지능학과
Department of
Artificial Intelligence

정 우 환 (whjung@hanyang.ac.kr)

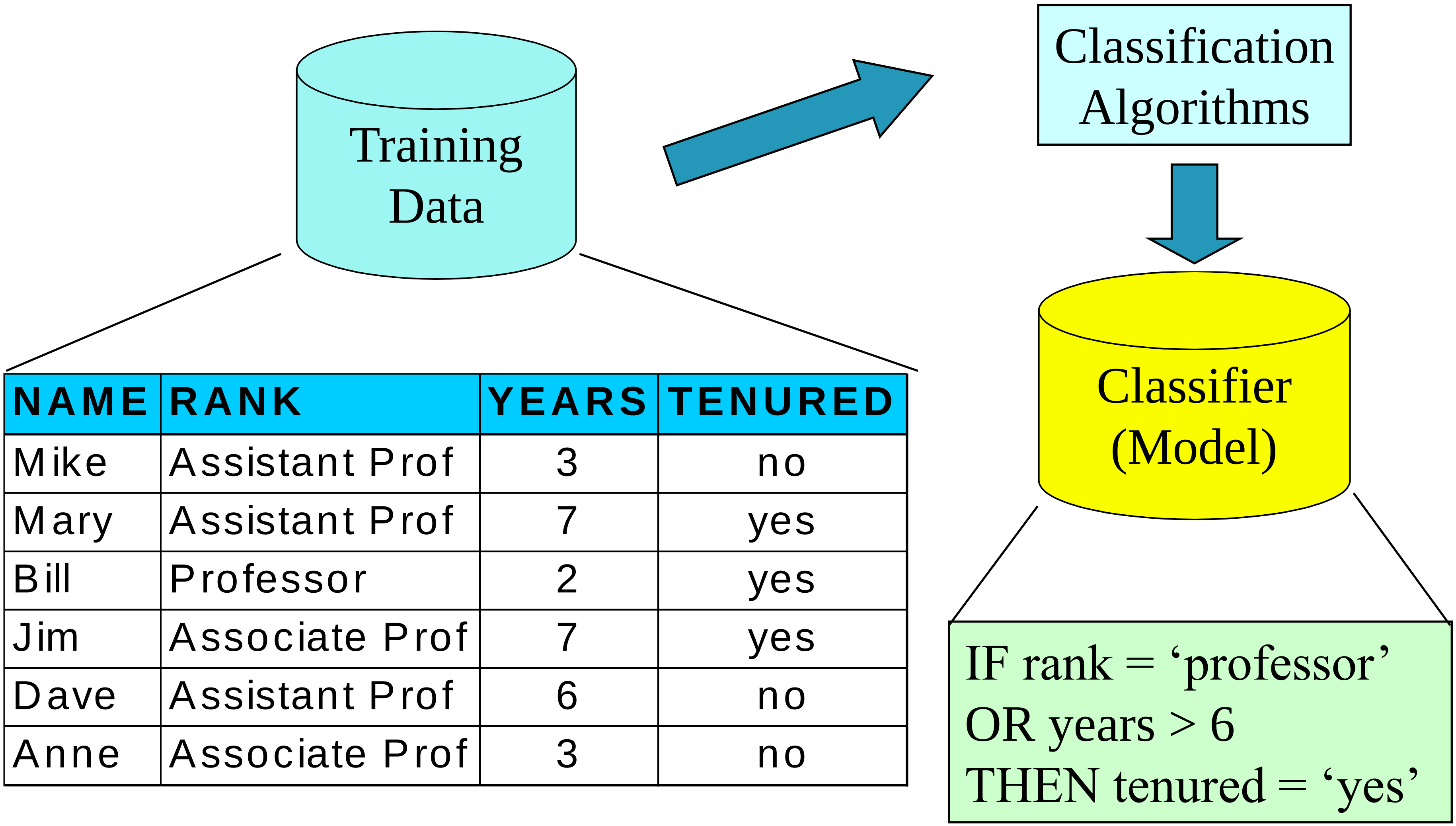
Fall 2021


MACHINE LEARNING
CLASSICAL LEARNING
UNSUPERVISED
SUPERVISED
Clustering
DBSCAN
K-Means
Agglomerative
Mean-Shift
Fuzzy C-Means
Pattern search
Euclat
Apriori
FP-Growth
DIMENSION REDUCTION (generalization)
t-SNE
PCA
LSA
SVD
LDA
Classification
Naive Bayes
SVM
K-NN
Decision Trees
Logistic Regression
Regression
Linear Regression
Polynomial Regression
Ridge/Lasso Regression
REINFORCEMENT LEARNING
Genetic Algorithm
Q-Learning
SARSA
Deep Q-Network (DQN)
A3C
ENSEMBLE METHODS
Stacking
Bagging
Random Forest
Boosting
XGBoost
AdaBoost
LightGBM
CatBoost
NEURAL NETS AND DEEP LEARNING
Convolutional Neural Networks (CNN)
DCNN
Perceptrons (MLP)
Recurrent Neural Networks (RNN)
LSM
LSTM
GRU
Generative Adversarial Networks (GAN)
Autoencoders
seq2seq

Classification—A Two-Step Process

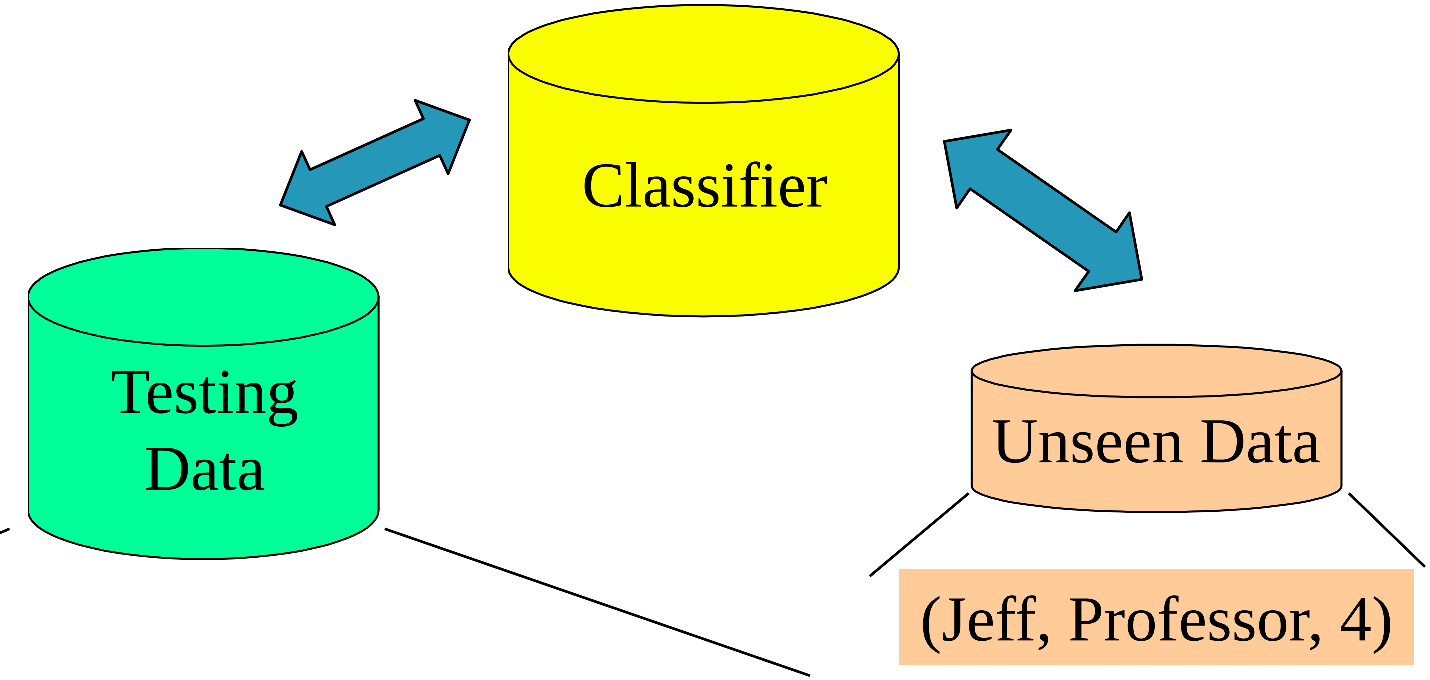
- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set (otherwise overfitting)
  - If the accuracy is acceptable, use the model to classify new data
- Note: If *the test set* is used to select models, it is called validation (test) set

Process (1): Model Construction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Process (2): Using the Model in Prediction



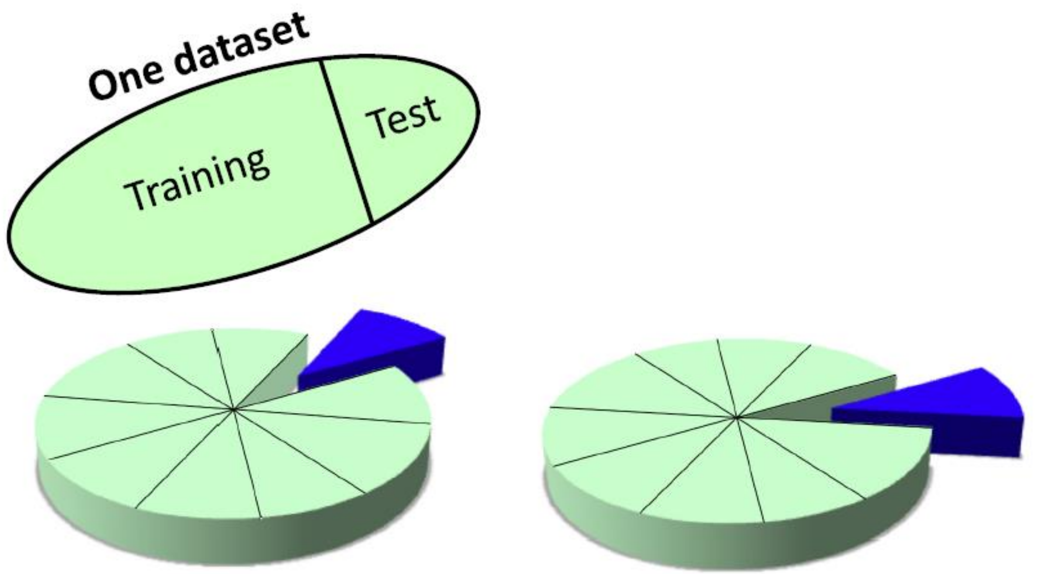
NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Tenured?

Yes

Cross-validation

- 10-fold cross-validation
  - Divide dataset into 10 parts (folds)
  - Hold out each part in turn
  - Average the results
  - Each data point used once for testing, 9 times for training



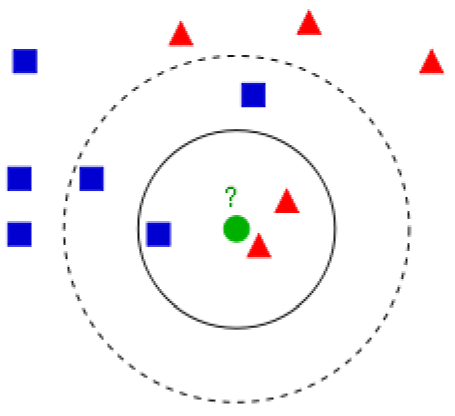
Ian H. Witten's slide

(repeat 10 times)

K-NEAREST NEIGHBOR CLASSIFIER

K-nearest Neighbor Classifier

- KNN classifier
  - Choose majority class among k-nearest neighbor neighbors



When k=3

When k=5

K-nearest Neighbor Classifier

- Assign to a point the label for majority of the k-nearest neighbors
- Often very accurate … but slow:
  - Scan entire training data to make each prediction?
    - Sophisticated data structures can make this faster
    - R-tree family works well up to 20 dimensions

K-nearest Neighbor Classifier

- Simplest form of learning
- To classify a new instance, search training set for one that's "most like" it
  - the instances themselves represent the "knowledge"
  - lazy learning: do nothing until you have to make predictions
- "Instance-based" learning = "nearest-neighbor" learning

Ian H. Witten's slide

Python – K-nearest Neighbor Classifier

Download the Dataset

- Download glass.csv from
  - https://hyu-my.sharepoint.com/:f:/g/personal/whjung_hanyang_ac_kr/Ev34n7L_Z0BErxWCad88rsAB6IdZCa0cUn7_Rd0ryYYYWQ?e=bqVe6k
  - PWD: ai202102
- Save the csv file in the same directory as the source file (.ipynb)

glass.csv

- Classify the type of glass
  - Motivated by criminological investigation
    - At the scene of the crime, the glass left can be used as evidence...if it is correctly identified!

Features:
RI: refractive index
Na: Sodium
Mg: Magnesium
...

Types of glass:
building_windows_float_processed
building_windows_non_float_processed
vehicle_windows_float_processed
...

Import Libraries

```
import pandas as pd
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.neighbors import KNeighborsClassifier
```

- `pandas`: a library for data analysis
- `cross_val_score`: a function for K-fold cross validation
- `KNeighborsClassifier`: a class for K-nearest neighbor classifier
- KFold : K-fold cross validation model

Open the Dataset

In [90]:
```
df = pd.read_csv('glass.csv')
df
```

Out [90]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.51793	12.79	3.50	1.12	73.03	0.64	8.77	0.00	0.00	'build wind float'
1	1.51643	12.16	3.52	1.35	72.89	0.57	8.53	0.00	0.00	'vehic wind float'
2	1.51793	13.21	3.48	1.41	72.64	0.59	8.43	0.00	0.00	'build wind float'
3	1.51299	14.40	1.74	1.54	74.55	0.00	7.59	0.00	0.00	tableware
4	1.53393	12.30	0.00	1.00	70.16	0.12	16.19	0.00	0.24	'build wind non-float'
5	1.51655	12.75	2.85	1.44	73.27	0.57	8.79	0.11	0.22	'build wind non-float'
6	1.51779	13.64	3.65	0.65	73.00	0.06	8.93	0.00	0.00	'vehic wind float'
7	1.51837	13.14	2.84	1.28	72.85	0.55	9.07	0.00	0.00	'build wind float'
8	1.51545	14.14	0.00	2.68	73.39	0.08	9.07	0.61	0.05	headlamps
9	1.51789	13.19	3.90	1.30	72.33	0.55	8.44	0.00	0.28	'build wind non-float'
10	1.51625	13.36	3.58	1.49	72.72	0.45	8.21	0.00	0.00	'build wind non-float'

Data Preprocessing

```
X = df.values[:, :-1]
y = df.values[:, -1]
```

```
print(X)
```

```
[[1.51793 12.79 3.5 ... 8.77 0.0 0.0]
 [1.51643 12.16 3.52 ... 8.53 0.0 0.0]
 [1.51793 13.21 3.48 ... 8.43 0.0 0.0]
 ...
 [1.51613 13.92 3.52 ... 7.94 0.0 0.14]
 [1.51689 12.67 2.88 ... 8.54 0.0 0.0]
 [1.51852 14.09 2.19 ... 9.32 0.0 0.0]]
```

```
print(y)
```

```
["'build wind float'" "'vehic wind float'" "'build wind float'"
 'tableware' "'build wind non-float'" "'build wind non-float'"
 "'vehic wind float'" "'build wind float'" 'headlamps'
 "'build wind non-float'" "'build wind non-float'"
 "'build wind non-float'" "'build wind float'" "'vehic wind float'"
 "'vehic wind float'" "'build wind non-float'" 'headlamps'
 "'build wind non-float'" 'containers' "'build wind non-float'"
 "'build wind float'" "'build wind non-float'" "'build wind non-float'"
 "'build wind float'" 'containers' "'build wind non-float'"
 "'build wind non-float'" 'headlamps' "'build wind non-float'"
 "'vehic wind float'" "'build wind non-float'" "'vehic wind float'"
 'tableware' "'build wind non-float'" "'build wind float'"
 "'build wind float'" "'build wind float'" "'build wind non-float'"
 "'build wind non-float'" "'build wind non-float'" "'build wind float'"
```

Changing Model Parameters

```
clf = KNeighborsClassifier(n_neighbors=10,
                           weights='uniform',
                           metric='euclidean')
```

n_neighbors : number of neighbors (k)
weights : weight function used in prediction.
	'uniform' : all neighbors have same weight
	'distance' : weights are given according to the distance
	* Note : user defined function can also be called
metric : the distance metric to use

K-fold Cross-validation

```
cv = KFold(
     n_splits=10,
     shuffle=True,
     random_state=0)
cv_results = cross_val_score(clf, X, y, cv=cv)

print(cv_results.mean())
```

The number of folds

Whether to shuffle the data before splitting into batches

The random seed

```
0.6155844155844156
```

K-fold Cross-validation

```
cv = KFold(
    n_splits=10,
    shuffle=True,
    random_state=0)
cv_results = cross_val_score(clf, X, y, cv=cv)

print(cv_results.mean())
```

Features

The classifier

Labels

0.6155844155844156

K-fold Cross-validation

```
cv = KFold(
    n_splits=10,
    shuffle=True,
    random_state=0)
cv_results = cross_val_score(clf, X, y, cv=cv)

print(cv_results.mean())
```

```
0.6155844155844156
```

Scores of 10-fold cross-validations

Print the average of scores

Prediction with KNN

Fit the model to the train data X

```
clf.fit(X,y)
pred_y = clf.predict(
    [[1.5, 13, 1.5, 1.5, 70, 0.5, 8.9, 0.1, 0.2]])
print(pred_y)
```

Test data should be a 2-D array

```
["'build wind float'"]
```

The prediction result is printed

Comparison with Varying k

```
clf = KNeighborsClassifier(n_neighbors=20, weights='uniform')
clf2 = KNeighborsClassifier(n_neighbors=5, weights='uniform')
clf3 = KNeighborsClassifier(n_neighbors=1, weights='uniform')

results = cross_val_score(clf, X, y, cv=cv)
results2 = cross_val_score(clf2, X, y, cv=cv)
results3 = cross_val_score(clf3, X, y, cv=cv)

print("20 neighbors: {}".format(
    results.mean()))
print("5 neighbors: {}".format(
    results2.mean()))
print("1 neighbors: {}".format(
    results3.mean()))
```

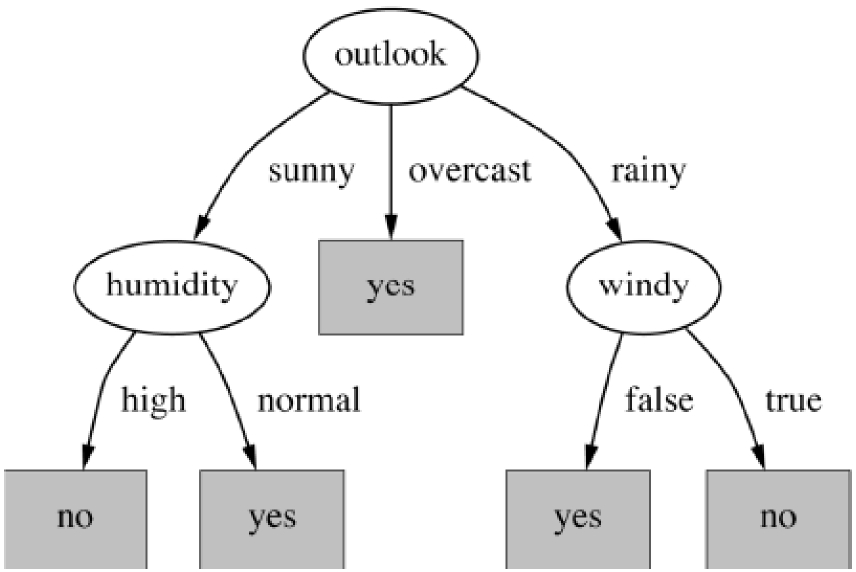
Varying the number of neighbors

```
20 neighbors: 0.6155844155844156
5 neighbors: 0.6480519480519480
1 neighbors: 0.7370129870129871
```

Note: It is not always a good idea to increase k

DECISION TREE CLASSIFIERE

Decision Tree Induction: An Example

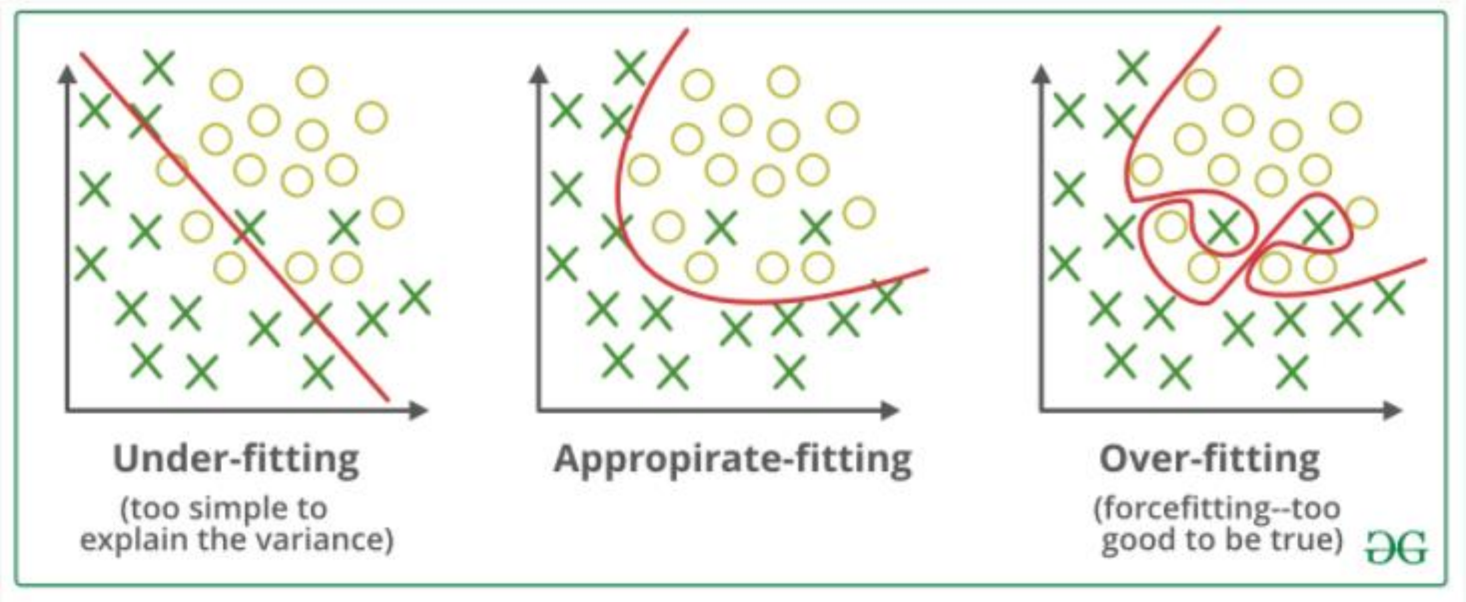


Outlook	Temp	Humidity	Wind	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Decision Tree Algorithm

- A decision tree is created in two phases:
  - Building Phase
    - Recursively split nodes using best splitting attribute for node until all the examples in each node belong to one class
  - Pruning Phase
    - Prune leaf nodes recursively to prevent over-fitting
    - Smaller imperfect decision tree generally achieves better accuracy

Underfitting and Overfitting



dewangNautiyal
Geekforgeeks

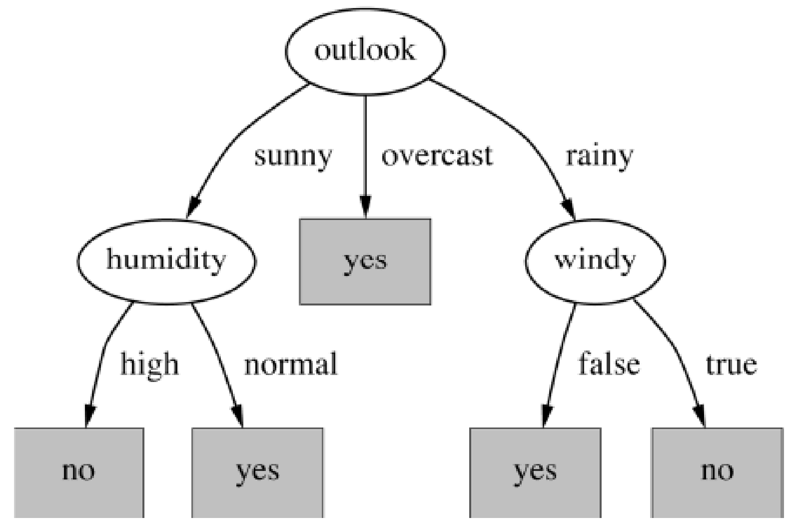
Building Phase

- General tree-growth algorithm (binary tree)

```
Partition(Data S)
    If (all points in S are of the same class) then return;
    for each attribute A do
        evaluate splits on attribute A;
    Use best split to partition S into S1 and S2;
    Partition(S1);
    Partition(S2);
```

Decision trees

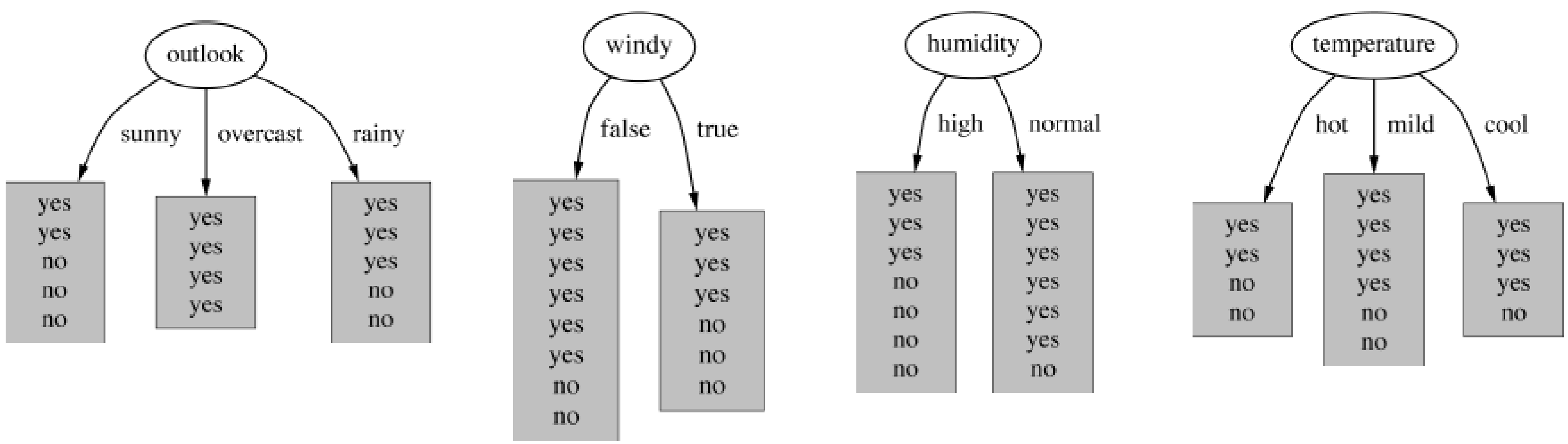
- Top-down: recursive divide-and-conquer
  - Select attribute for root node
    - Create branch for each possible attribute value
  - Split instances into subsets
    - One for each branch extending from the node
  - Repeat recursively for each branch
    - using only instances that reach the branch
  - Stop
    - if all instances have the same class



Ian H. Witten's slide

Decision Trees

Which attribute to select?



Ian H. Witten's slide

Decision Trees

- Which is the best attribute?
  - Aim: to get the smallest tree
  - Heuristic
    - choose the attribute that produces the "purest" nodes
    - i.e., the greatest information gain
- Q: How to measure the amount of information (gain)?

Information

- Quantity of information

1000 bits

0000000…000000000

Same quantity?

1000 bits

0010001…111001001

0 * 1000

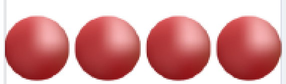
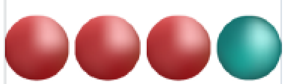
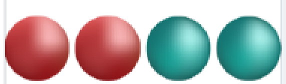
Same quantity?

0*2,1*1,0*3…1*3,0*2,1*1,0*2,1*1

(Self) Information $I(x)$

- Roughly speaking, the minimum number of bits to encode a signal x
- Definition
  - $I(x) = -\log P(x)$
- Intuition
  - If a pattern is frequent, it can be simply and efficiently encoded/compressed
  - Example: 0000000…000000000

Probability of Winning

	P(red)	P(blue)	P(winning)
	1	0	$1 \times 1 \times 1 \times 1 = \mathbf{1}$
	0.75	0.25	$0.75 \times 0.75 \times 0.75 \times 0.25 = \mathbf{0.105}$
	0.5	0.5	$0.5 \times 0.5 \times 0.5 \times 0.5 = \mathbf{0.0625}$

http://incredible.ai/machine-learning/2018/11/08/Entropy/

Decision Trees

- Which is the best attribute?
  - Aim: to get the smallest tree
  - Heuristic
    - choose the attribute that produces the "purest" nodes
    - i.e., the greatest information gain
  - Information theory: measure information in bits
    - entropy(p_1, p_2,..., p_n)= $-p_1 \log p_1 - p_2 \log p_2 \ldots -P_n \log p_n$
- Information gain
  - Amount of information gained by knowing the value of the attribute
  - (Entropy of distribution before the split) – (entropy of distribution after it)
  - Claude Shannon, American mathematician and scientist 1916–2001

Ian H. Witten's slide

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

- Information gained by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

Attribute Selection: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940$$

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

age	income	student	credit_rating	buys_computer
<=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

$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

$\frac{5}{14}I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

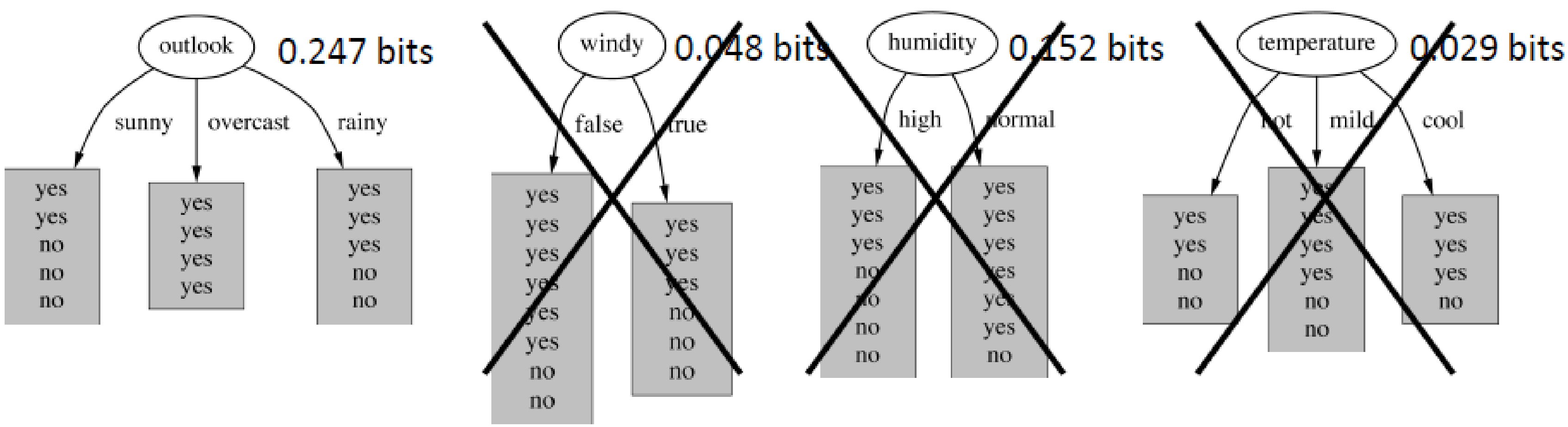
$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly,

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

Decision Trees

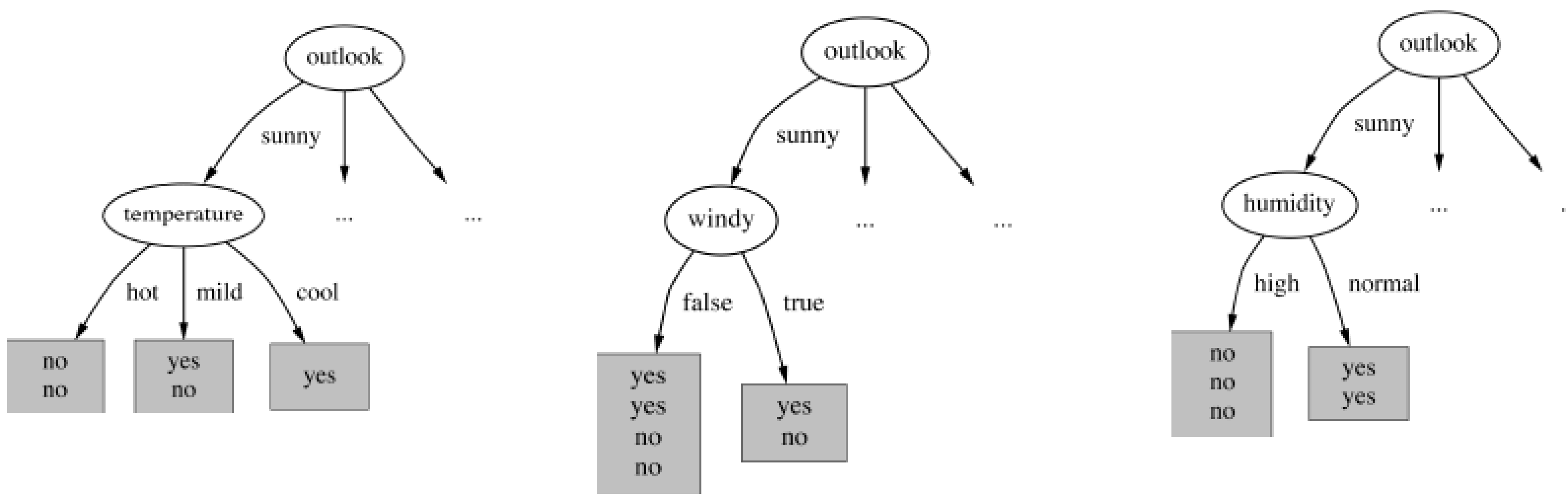
Which attribute to select?



Ian H. Witten's slide

Decision Trees

Continue to split …



gain(*temperature*) = 0.571 bits

gain(*windy*) = 0.020 bits

gain(*humidity*) = 0.971 bits

Ian H. Witten's slide

Splitting Numeric Attributes

- Split on temperature attribute:

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

  - E.g. temperature $<$ 71.5: yes/4, no/2
    temperature $\geq$ 71.5: yes/5, no/3

  - Info([4,2],[5,3])
    = 6/14 info([4,2]) + 8/14 info([5,3])
    = 0.939 bits

- Place split points halfway between values
- Can evaluate all split points in one pass!

Ian H. Witten's slide

Avoid repeated sorting!

- Sort instances by the values of the numeric attribute
  - Time complexity for sorting: $O(n \log n)$
- Q. Does this have to be repeated at each node of the tree?
- A: No! Sort order for children can be derived from sort order for parent
  - Time complexity of derivation: $O(n)$
  - Drawback: need to create and store an array of sorted indices for each numeric attribute

Ian H. Witten's slide

More speeding up

- Entropy only needs to be evaluated between points of different classes (Fayyad & Irani, 1992)

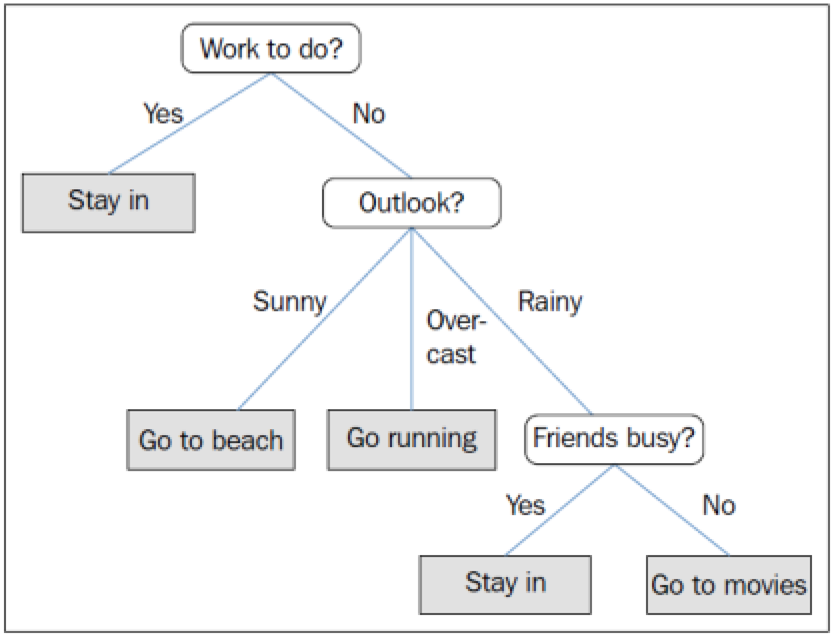
value	64	65	68	69	70	71	72	72	75	75	80	81	83	85
class	Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

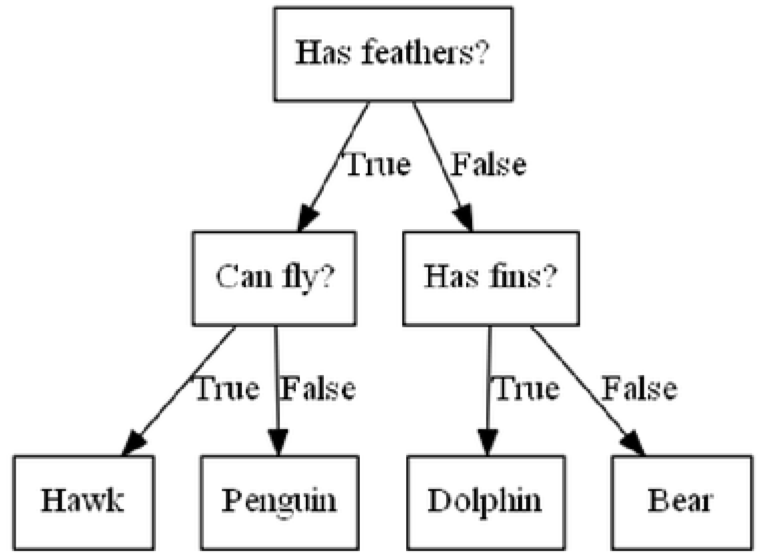
Potential optimal breakpoints

Breakpoints between values of the same class cannot be optimal

Ian H. Witten's slide

Decision trees for multi-class classification



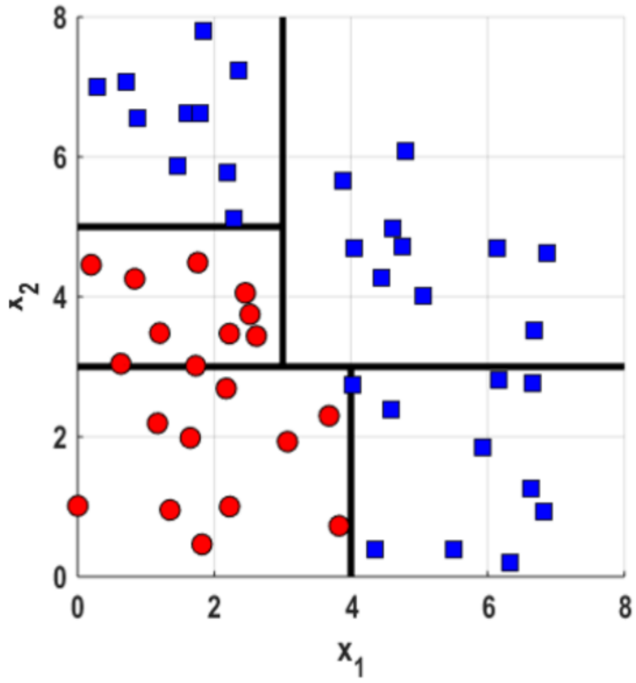


Visual Introduction to Decision Trees

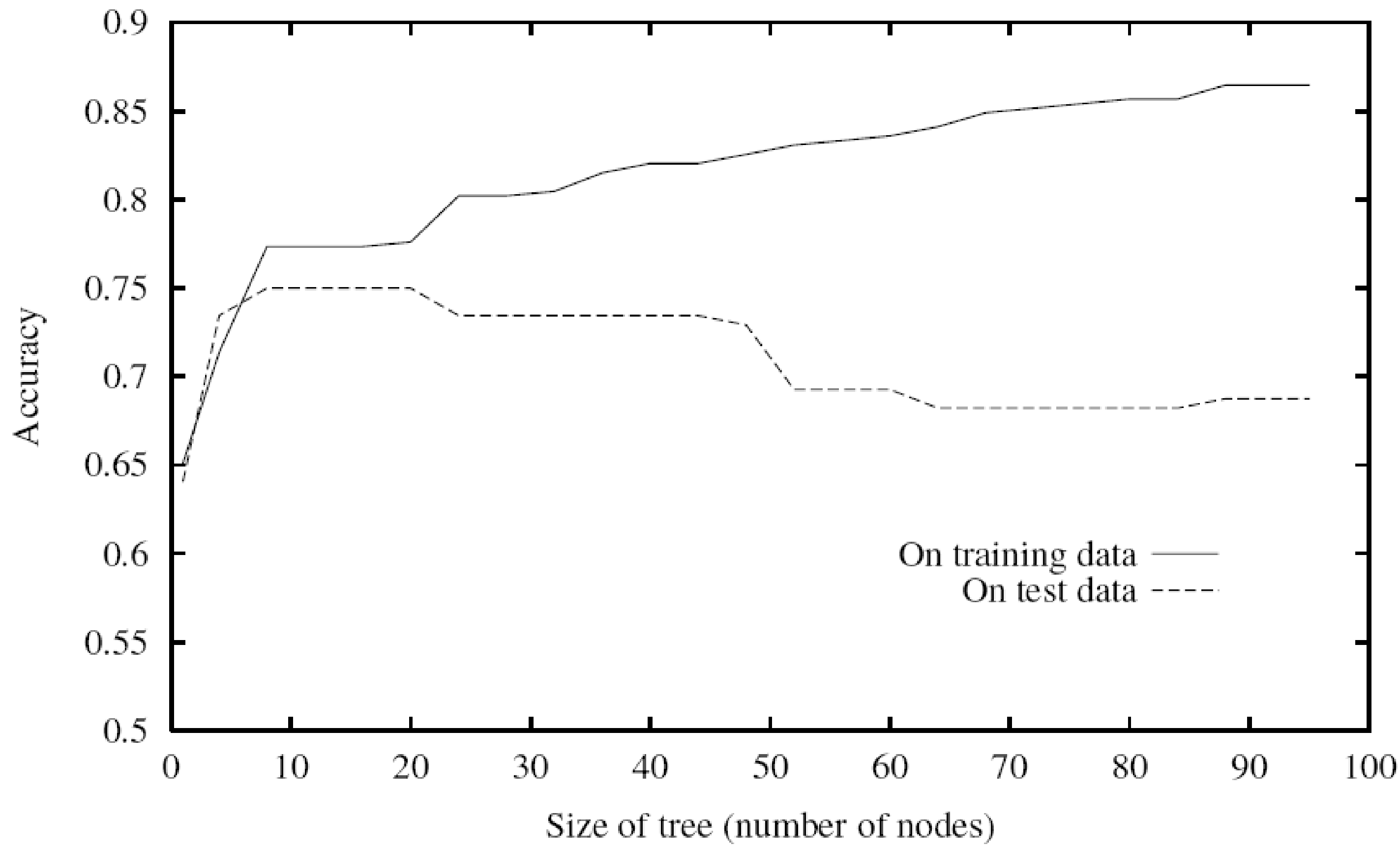
- http://www.r2d3.us/visual-intro-to-machine-learning-part-1/

Interpreting a Decision Tree

2. Interpreting a decision tree: Consider the decision boundary in Fig. and draw the equivalent decision tree. Red circle are Class +1 and blue squares are class -1.
   [10 Points]



Overfitting in Decision Tree Learning



Avoiding Overfitting

- How can we avoid overfitting?
  - Method 1: Stop growing when data split not statistically significant
  - Method 2: Grow full tree, then post-prune
- How to select the "best" tree:
  - Measure performance over training data
  - Measure performance over separate validation data set

Pruning

- Goal: Prevent overfitting to noise in the data
- Two strategies for "pruning" the decision tree:
  - Postpruning - take a fully-grown decision tree and discard unreliable parts
  - Prepruning - stop growing a branch when information becomes unreliable
- Postpruning preferred in practice—prepruning can "stop too early"

Ian H. Witten's slide

Prepruning

- Based on statistical significance test
  - Stop growing the tree when there is no *statistically significant* association between any attribute and the class at a particular node
- Most popular test: chi-squared test
- ID3 used chi-squared test in addition to information gain
  - Only statistically significant attributes were allowed to be selected by information gain procedure
- Pre-pruning may stop the growth process prematurely: early stopping
- Pre-pruning faster than post-pruning

Ian H. Witten's slide

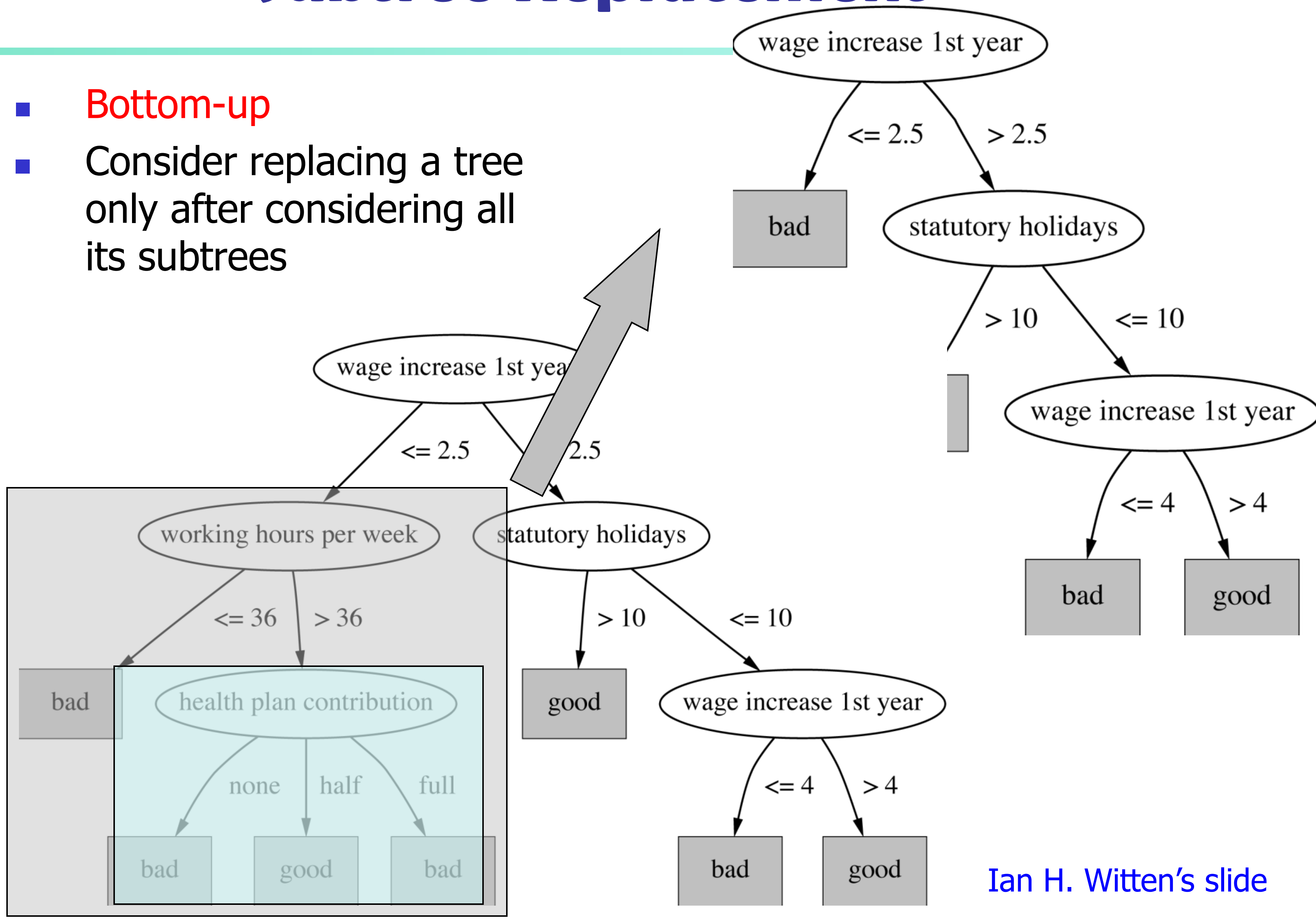
Post-pruning

- First, build full tree
- Then, prune it
  - Fully-grown tree shows all attribute interactions
- Two pruning operations:
  - Subtree replacement
  - Subtree raising
- Possible strategies:
  - Error estimation
  - Significance testing
  - MDL principle

Ian H. Witten's slide

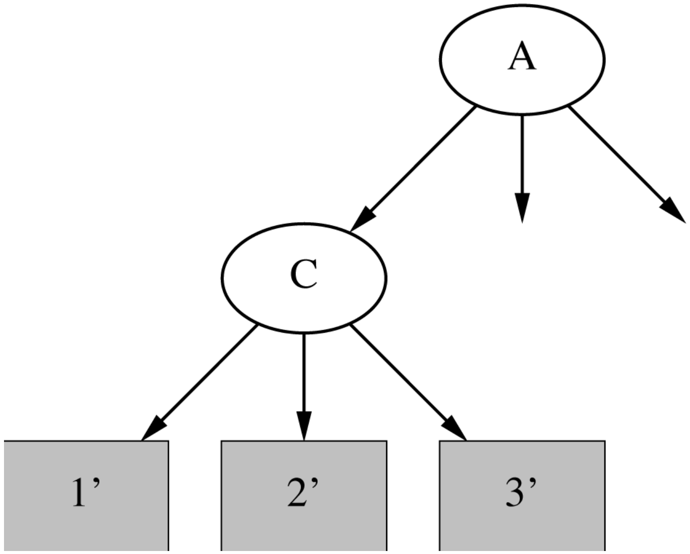
Subtree Replacement

- Bottom-up
- Consider replacing a tree only after considering all its subtrees

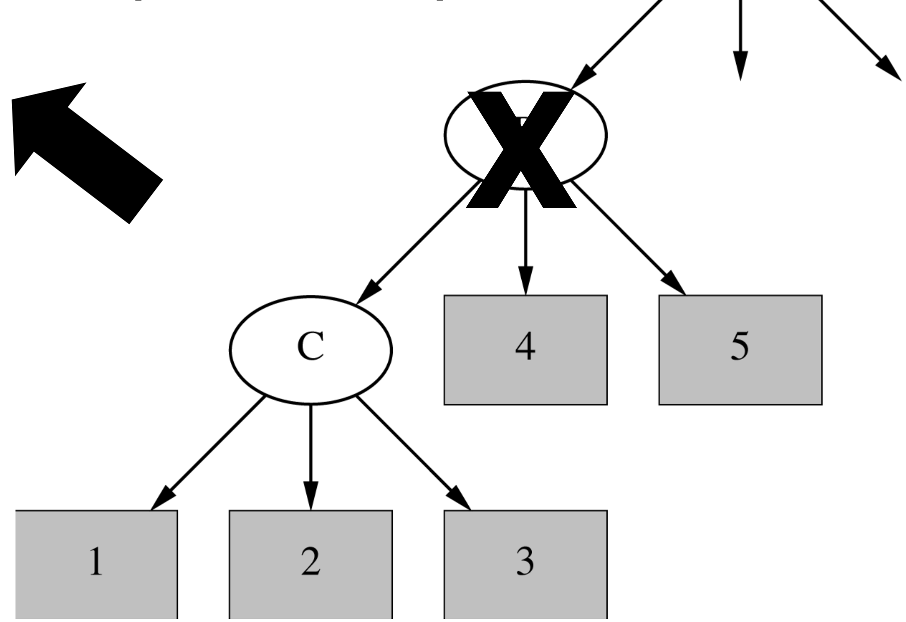


Ian H. Witten's slide

Subtree Raising



- Delete node
- Redistribute instances
- Slower than subtree replacement
  (Worthwhile?)



Ian H. Witten's slide