Artificial Intelligence

Classification 3

Extended from Kyuseok Shim's slides

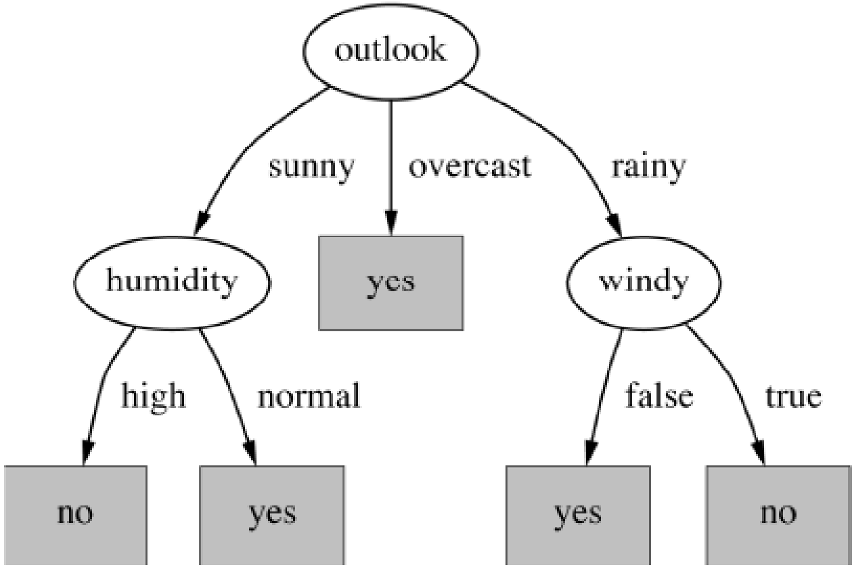
한양대학교 ERICA 소프트웨어융합대학 COLLEGE OF COMPUTING | 인공지능학과 Department of Artificial Intelligence

정 우 환 (whjung@hanyang.ac.kr)

Fall 2021

DECISION TREE CLASSIFIER

Decision Tree Induction: An Example



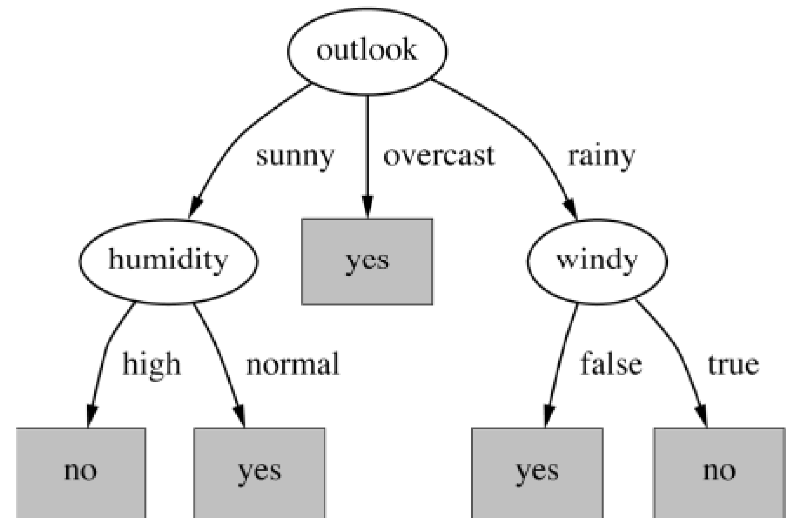
Outlook	Temp	Humidity	Wind	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny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

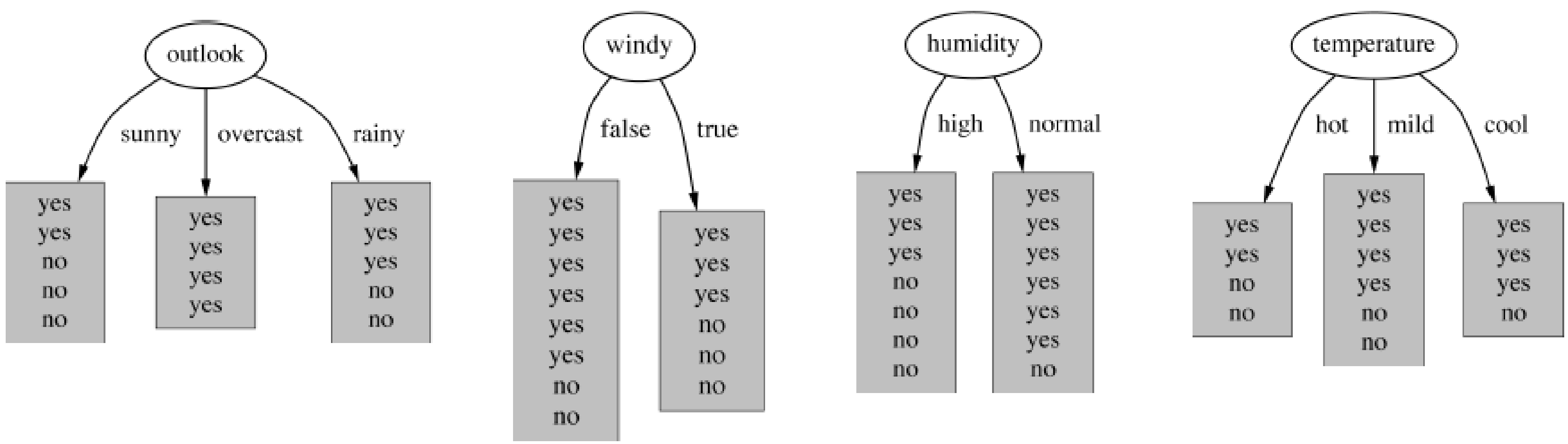
Decision trees

- 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



Decision Trees

Which attribute to select?



Ian H. Witten's slide

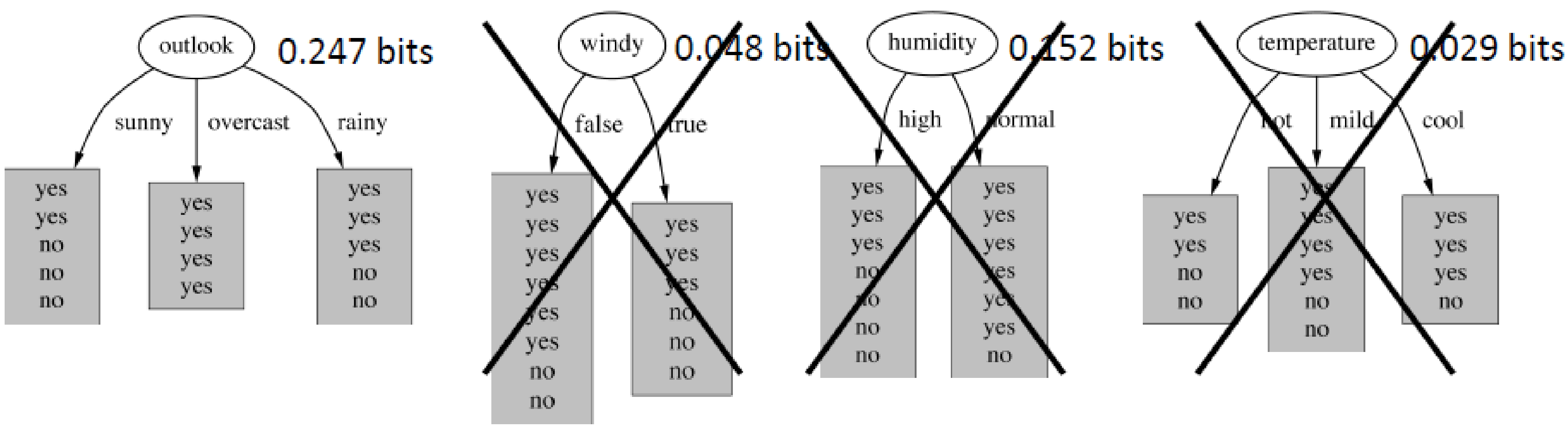
Decision Trees

- 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
  - 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

Decision Trees

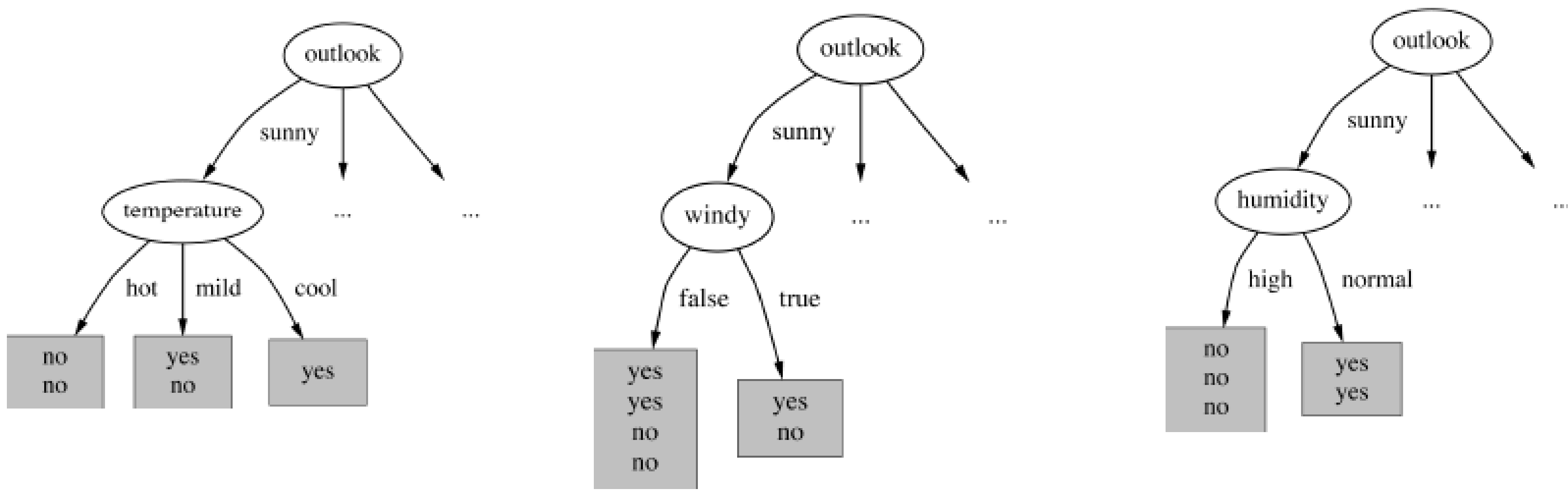
Which attribute to select?



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

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

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

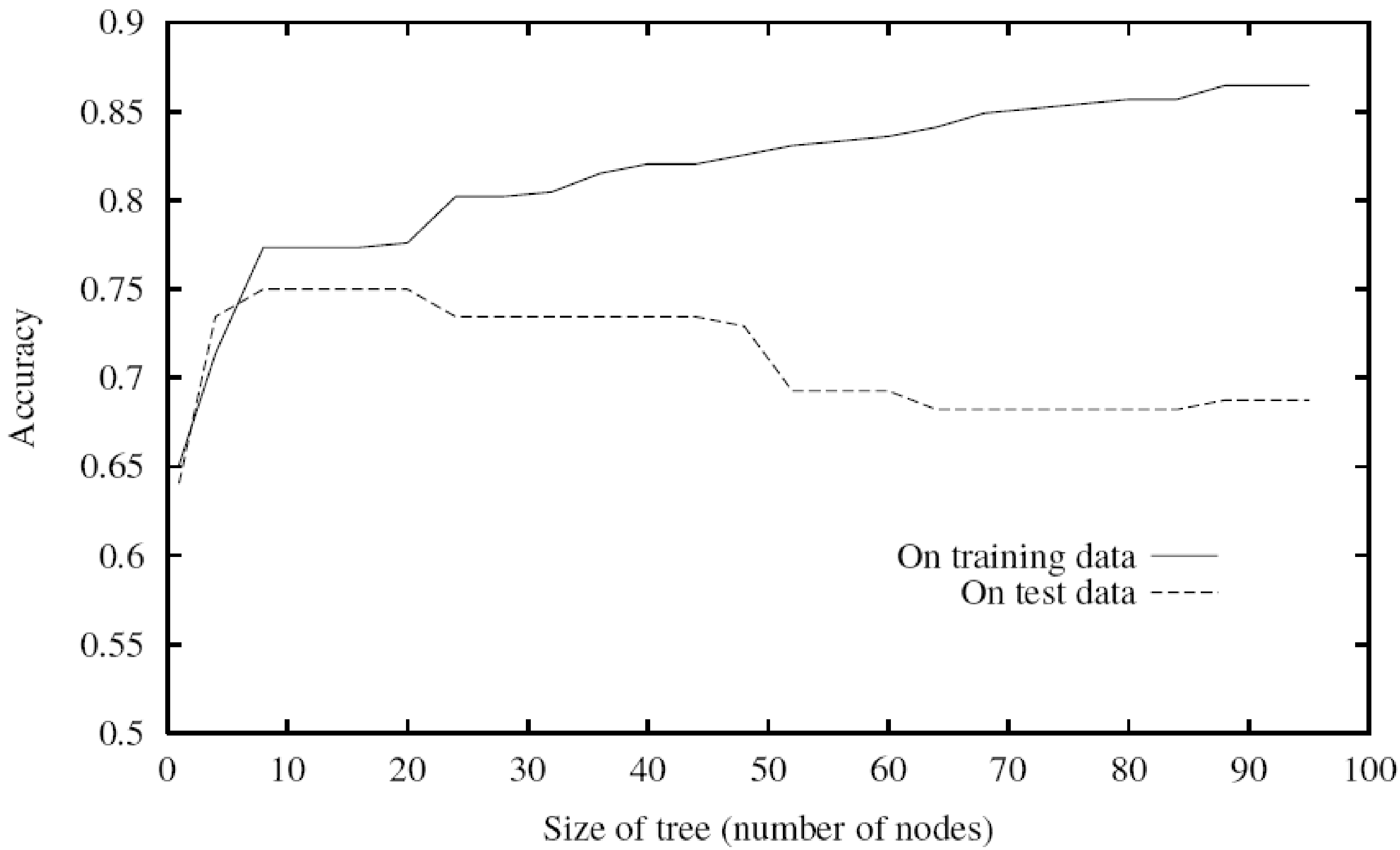
Similarly,

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

Overfitting in Decision Tree Learning

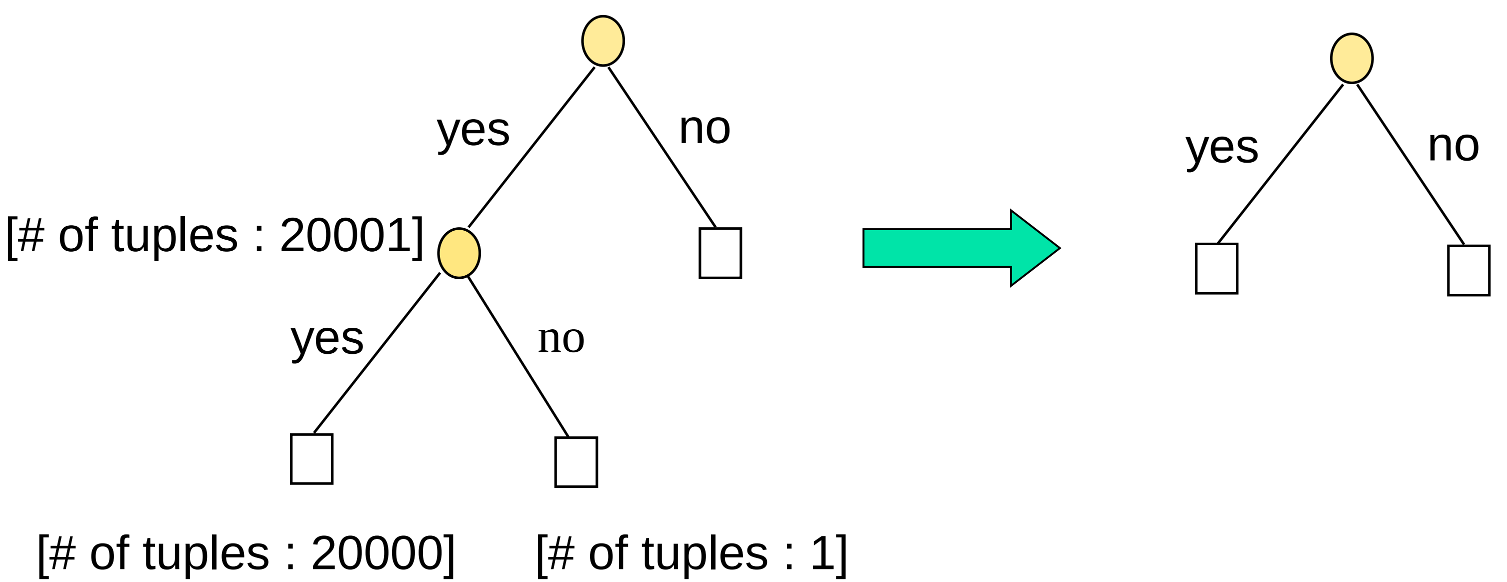


Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - 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”

Pruning Phase

- Smaller imperfect decision tree generally achieves better accuracy
- Prune leaf nodes recursively to prevent over-fitting



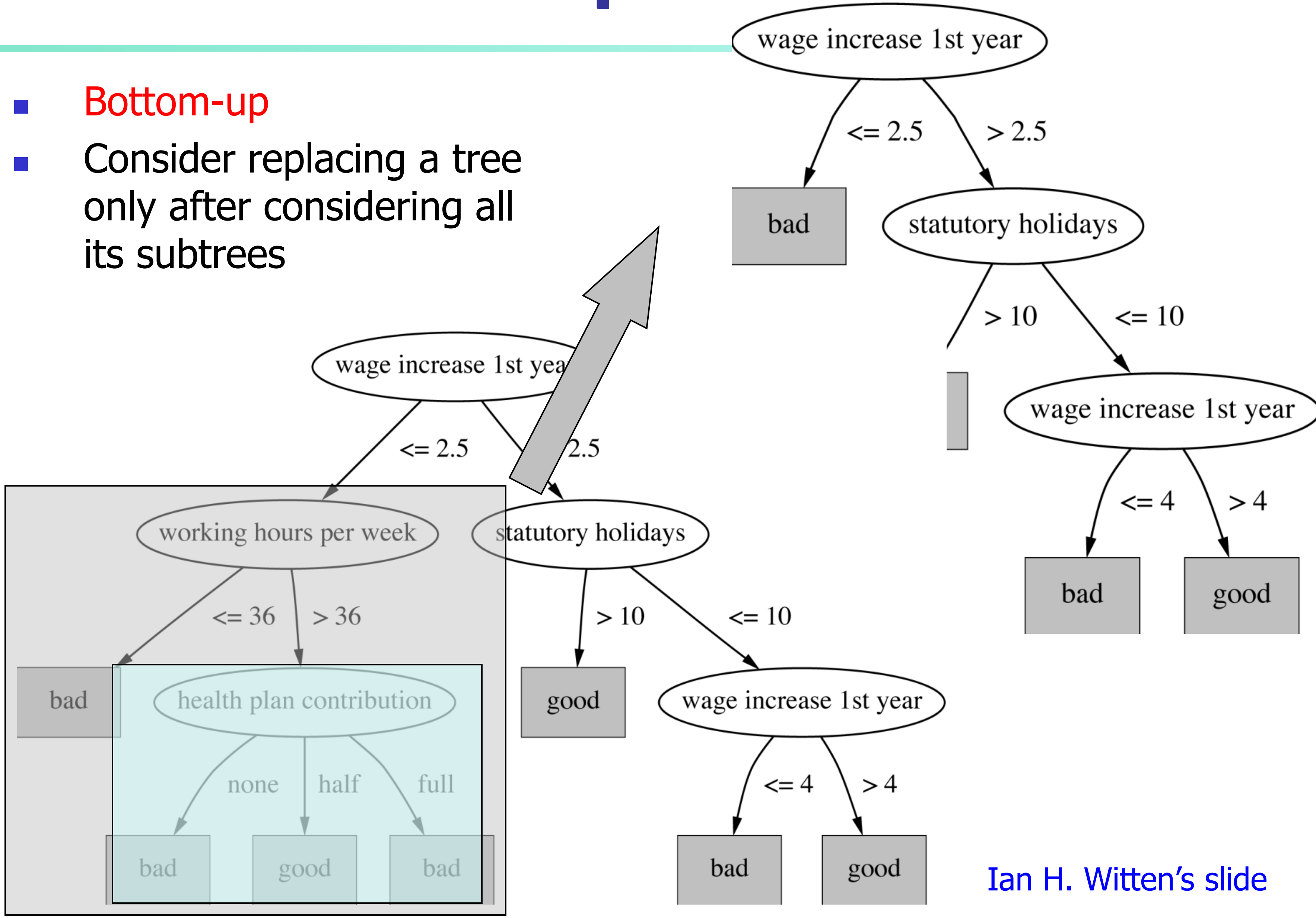
Post-pruning

- Split data into training and validation set
- Build full tree using training dataset
- Do until further pruning is harmful:
  - 1. Evaluate impact on validation set of pruning each possible node (plus those below it)
  - 2. Greedily remove the one whose removal most increases validation set accuracy
    - To possible approaches
      - Subtree replacement
      - Subtree raising
- Produces smallest version of the most accurate subtrees

Ian H. Witten's slide

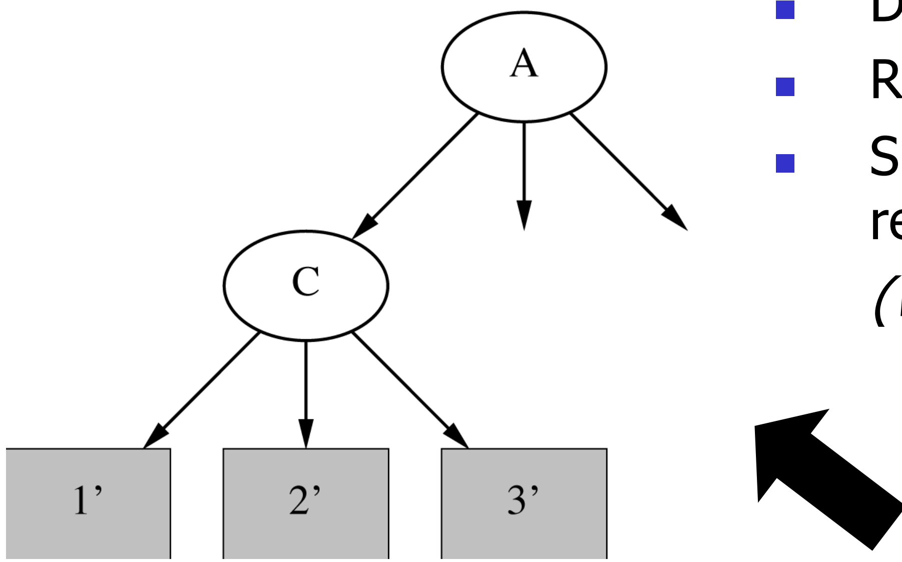
Subtree Replacement

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



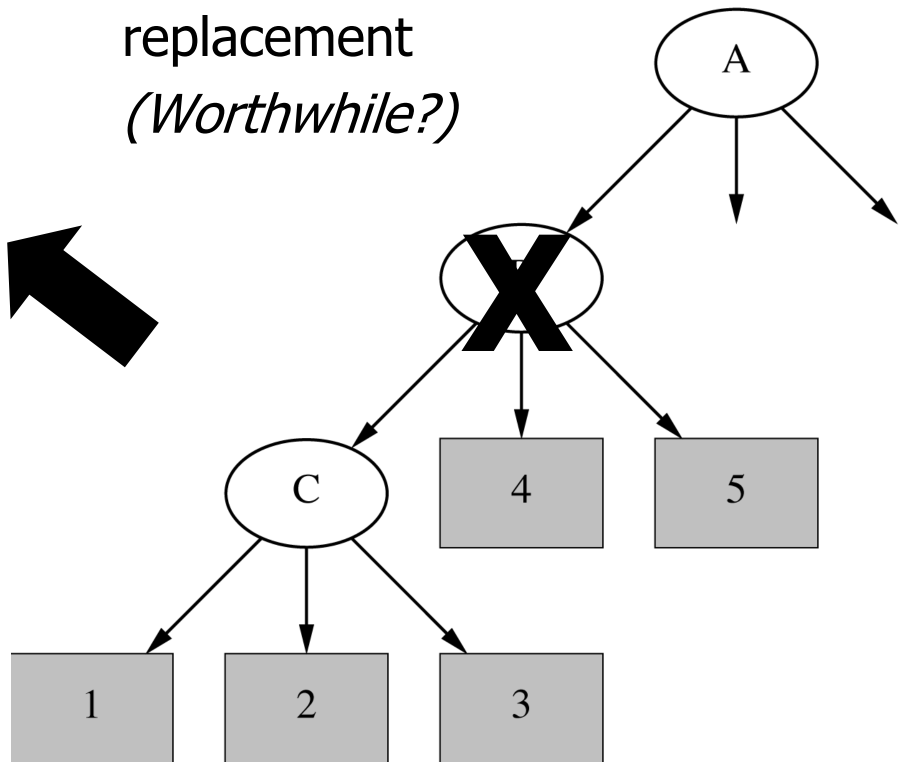
Ian H. Witten's slide

Subtree Raising



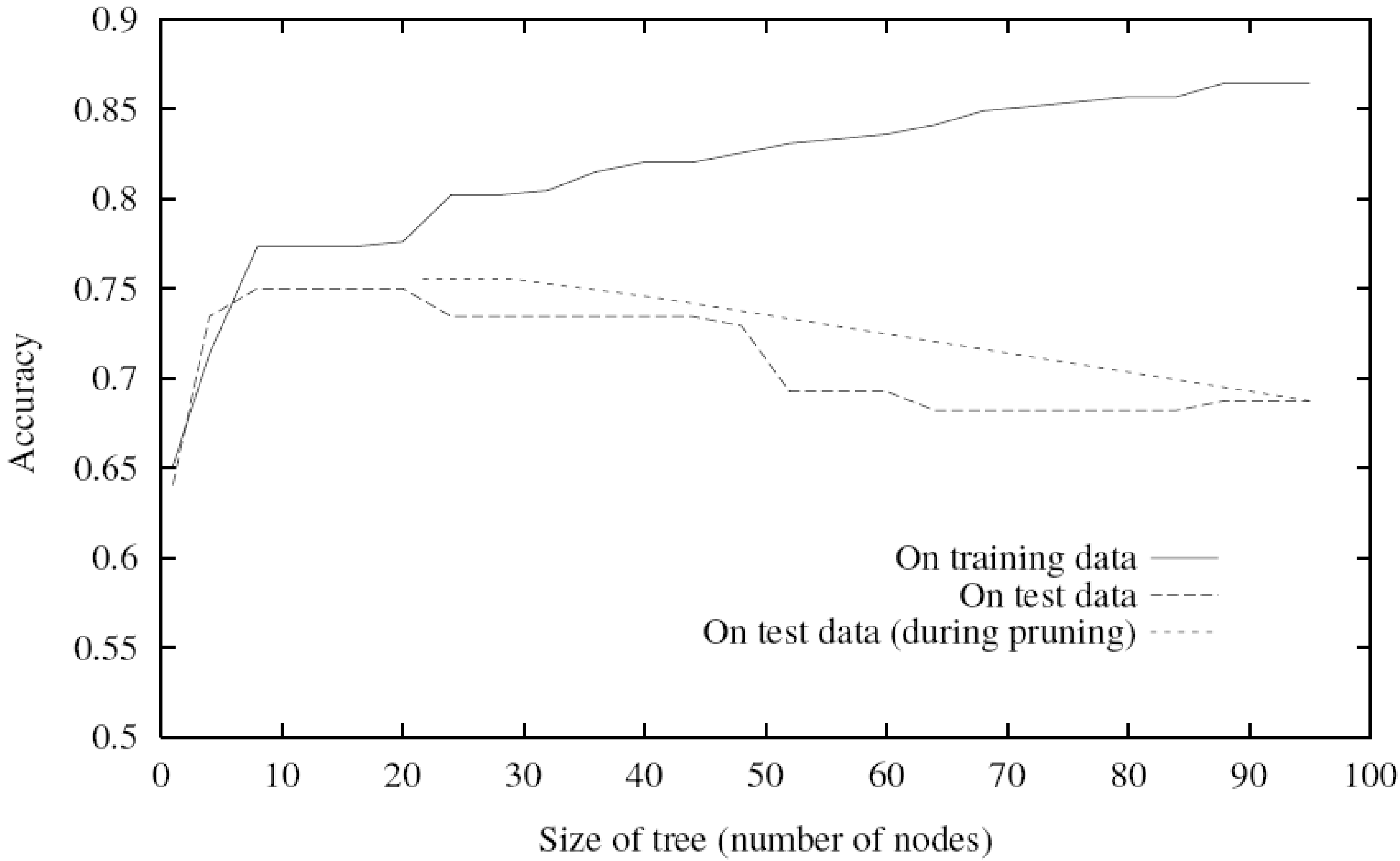
- Delete node
- Redistribute instances
- Slower than subtree replacement

(Worthwhile?)

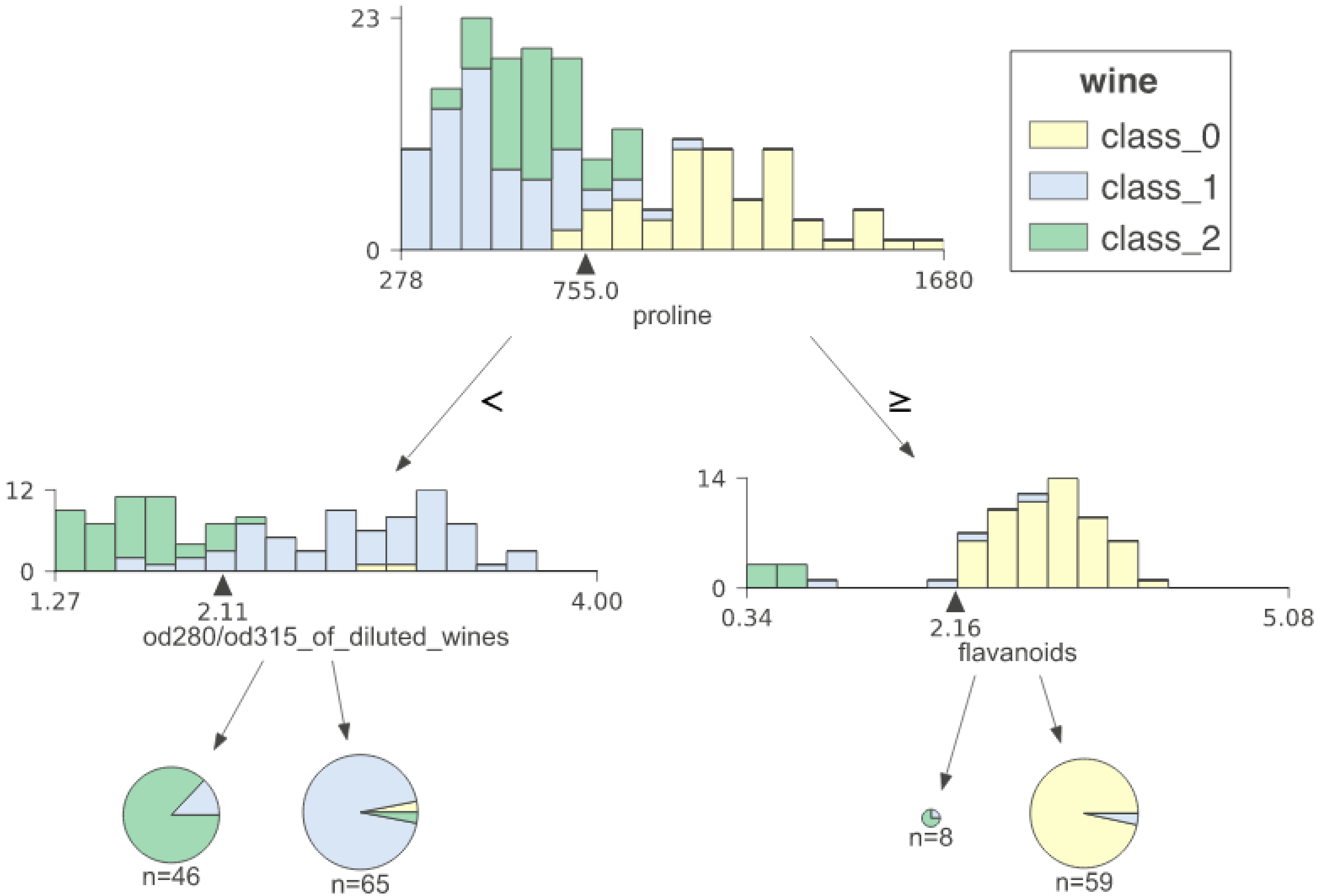


Ian H. Witten's slide

Effect of Reduced-Error Pruning

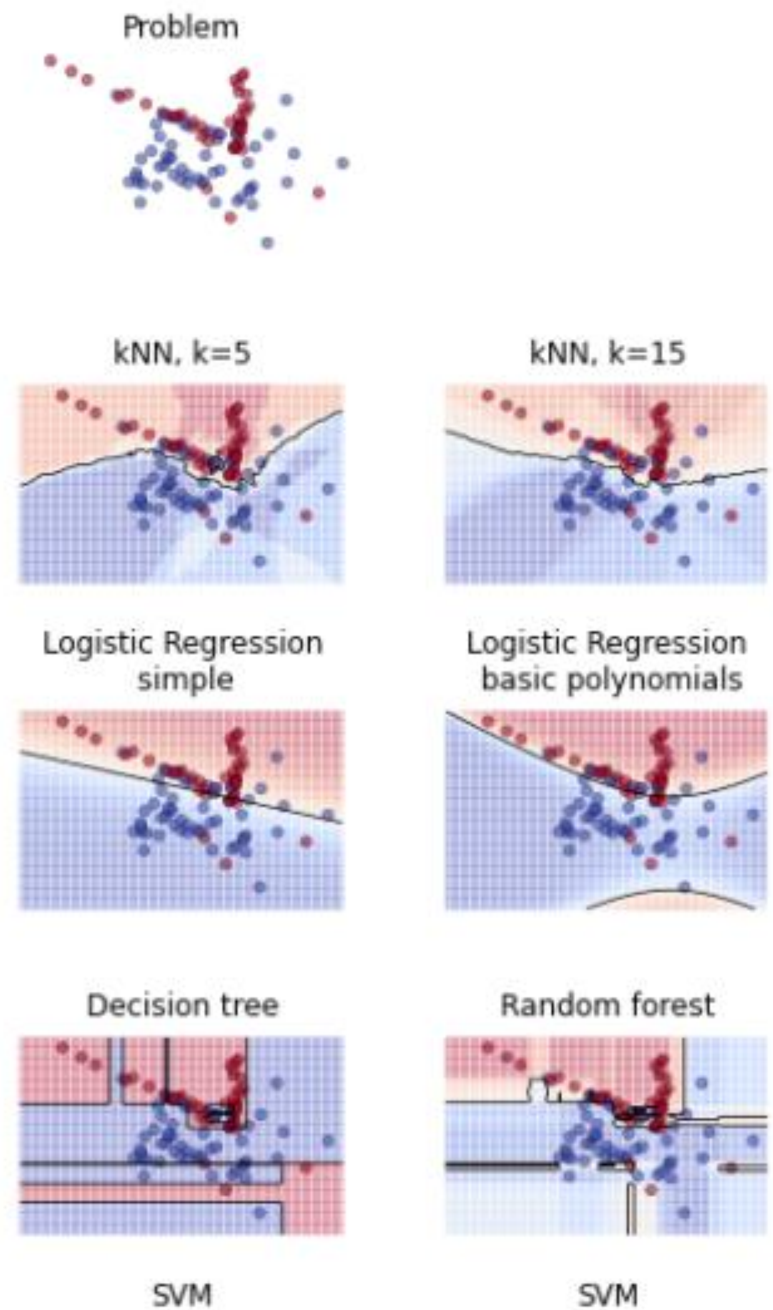


Decision Tree Visualization



https://explained.ai/decision-tree-viz/

Decision Boundaries



Python – Decision Tree Classifier

Import Libraries

```
from sklearn import tree
import graphviz
from sklearn.preprocessing import OneHotEncoder
import numpy as np
import pandas as pd
```

Open the Dataset

```
df = pd.read_csv('weather.nominal.csv')
df
```

	outlook	temperature	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes
5	rainy	cool	normal	True	no
6	overcast	cool	normal	True	yes
7	sunny	mild	high	False	no
8	sunny	cool	normal	False	yes
9	rainy	mild	normal	False	yes
10	sunny	mild	normal	True	yes

Data Preprocessing

- Sklearn does not provide the decision tree with categorical data
  - Should convert the categorical data to numerical data
  - We will use one hot encoding in sklearn

	Weather
1	Sunny
2	Overcast
3	Rainy

	Sunny	Overcast	Rainy
1	1	0	0
2	0	1	0
3	0	0	1

Data Preprocessing

Features   Class label

	outlook	temperature	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes

Ignore the last element

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

Only the last element

```
print(X)
```

```
[['sunny' 'hot' 'high' False]
 ['sunny' 'hot' 'high' True]
 ['overcast' 'hot' 'high' False]
 ['rainy' 'mild' 'high' False]
 ['rainy' 'cool' 'normal' False]
 ['rainy' 'cool' 'normal' True]
 ['overcast' 'cool' 'normal' True]
 ['sunny' 'mild' 'high' False]
 ['sunny' 'cool' 'normal' False]
 ['rainy' 'mild' 'normal' False]
 ['sunny' 'mild' 'normal' True]
 ['overcast' 'mild' 'high' True]
 ['overcast' 'hot' 'normal' False]
 ['rainy' 'mild' 'high' True]]
```

```
print(y)
```

```
['no' 'no' 'yes' 'yes' 'yes' 'no' 'yes' 'no' 'yes' 'yes' 'yes' 'yes' 'yes'
 'no']
```

Data Preprocessing

- X is transformed by one hot encoding

```
enc = OneHotEncoder(handle_unknown='ignore')
enc.fit(X)
en_X = enc.transform(X).toarray()
en_X
```

```
print(X)
```

```
[['sunny' 'hot' 'high' False]
 ['sunny' 'hot' 'high' True]
 ['overcast' 'hot' 'high' False]
 ['rainy' 'mild' 'high' False]
 ['rainy' 'cool' 'normal' False]
 ['rainy' 'cool' 'normal' True]
 ['overcast' 'cool' 'normal' True]
 ['sunny' 'mild' 'high' False]
 ['sunny' 'cool' 'normal' False]
 ['rainy' 'mild' 'normal' False]
 ['sunny' 'mild' 'normal' True]
 ['overcast' 'mild' 'high' True]
 ['overcast' 'hot' 'normal' False]
 ['rainy' 'mild' 'high' True]]
```

```
print(en_X)
```

```
[[0. 0. 1. 0. 1. 0. 1. 0. 1. 0.]
 [0. 0. 1. 0. 1. 0. 1. 0. 0. 1.]
 [1. 0. 0. 0. 1. 0. 1. 0. 1. 0.]
 [0. 1. 0. 0. 0. 1. 1. 0. 1. 0.]
 [0. 1. 0. 1. 0. 0. 0. 1. 1. 0.]
 [0. 1. 0. 1. 0. 0. 0. 1. 0. 1.]
 [1. 0. 0. 1. 0. 0. 0. 1. 0. 1.]
 [0. 0. 1. 0. 0. 1. 1. 0. 1. 0.]
 [0. 0. 1. 1. 0. 0. 0. 1. 1. 0.]
 [0. 1. 0. 0. 0. 1. 0. 1. 1. 0.]
 [0. 0. 1. 0. 0. 1. 0. 1. 0. 1.]
 [1. 0. 0. 0. 0. 1. 1. 0. 0. 1.]
 [1. 0. 0. 0. 1. 0. 0. 1. 1. 0.]
 [0. 1. 0. 0. 0. 1. 1. 0. 0. 1.]]
```

Data Preprocessing

- Check the encoded result

```
categories = []
for x in enc.categories_:
    categories += list(x)
print(categories)
```

```
['overcast', 'rainy', 'sunny', 'cool', 'hot', 'mild', 'high', 'normal', False, True]
```

0 1 0 0 0 1 1 0 0 1

['rainy', 'mild', 'high', True]

Building a Tree and Testing

Train a decision tree

1
```
clf = tree.DecisionTreeClassifier(criterion = "entropy")
clf = clf.fit(en_X,y)
print(clf.classes_)
```

```
['no' 'yes']
```

Labels of the model

2
```
test = [['overcast','mild','high', True]]
en_test = enc.transform(test).toarray()
print(en_test)
pred_y = clf.predict(en_test)
print(pred_y)
```

```
[[1. 0. 0. 0. 0. 1. 1. 0. 0. 1.]]
['yes']
```

Encode the test data

Note: test data should be a 2-D array

Output represents the predicted label of each data

3
```
pred_prob = clf.predict_proba(en_test)
print(pred_prob)
```

```
[[0. 1.]]
```

The probability of each label can also be shown

Cross-validation

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

print(cv_results.mean())
```

X → en_X

```
0.55
```

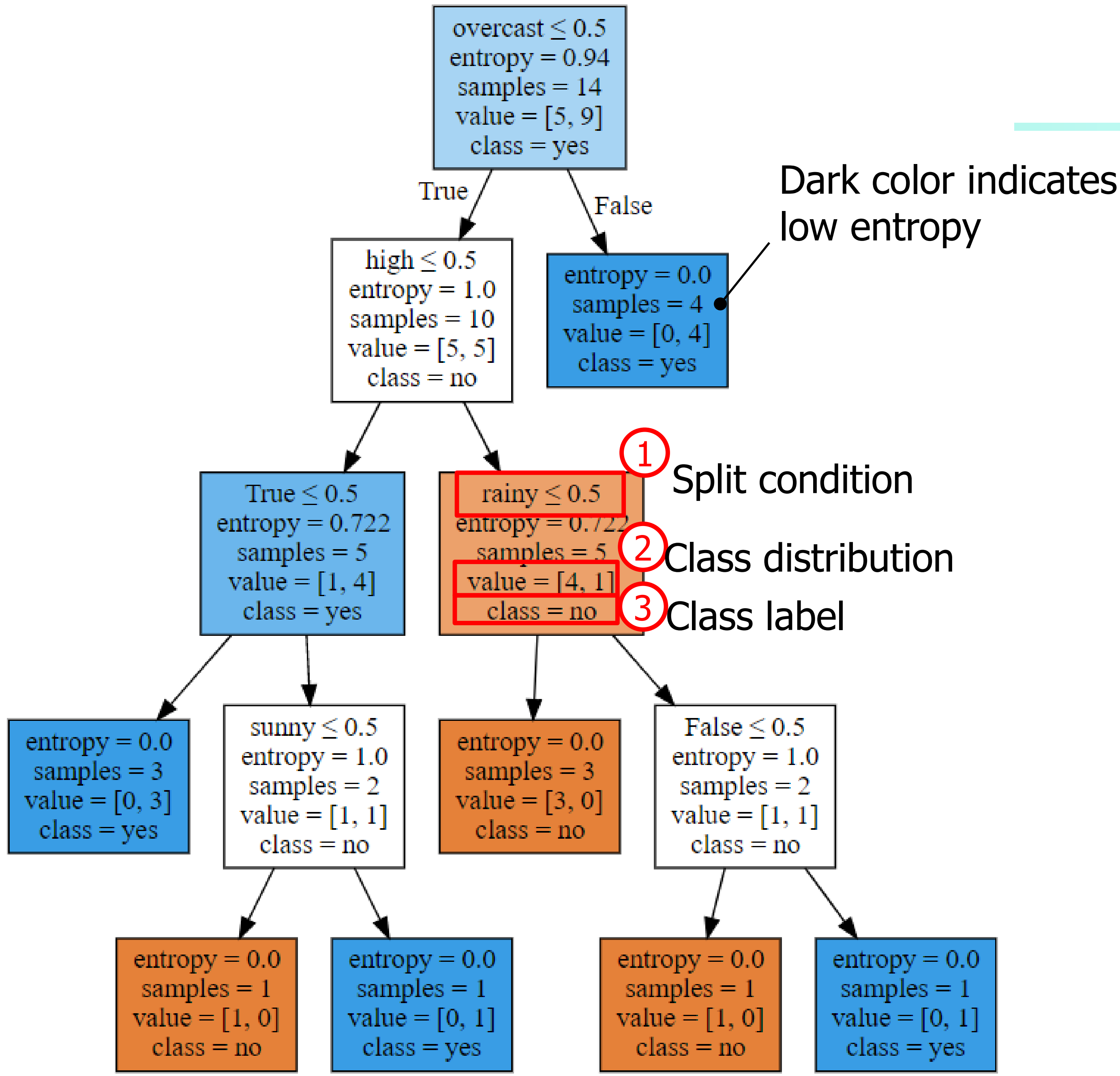
You can reuse the code used for KNN classifier
since the cross-validation code is almost the same

Visualize Tree

- Tree class includes the function *export_graphviz*

```
dot_data = tree.export_graphviz(clf, out_file=None,
                                feature_names=categories,
                                class_names=clf.classes_,
                                filled=True,
                                special_characters = True)
graph = graphviz.Source(dot_data)
graph
```

feature_names : the list of attributes of data
class_names : the list of labels of data
filled : graphviz colors the node if set True
special_characters : shows the special characters (i.e., ≤ or <=)


overcast ≤ 0.5
entropy = 0.94
samples = 14
value = [5, 9]
class = yes
True
False
high ≤ 0.5
entropy = 1.0
samples = 10
value = [5, 5]
class = no
entropy = 0.0
samples = 4
value = [0, 4]
class = yes
Dark color indicates low entropy
True ≤ 0.5
entropy = 0.722
samples = 5
value = [1, 4]
class = yes
rainy ≤ 0.5
entropy = 0.722
samples = 5
value = [4, 1]
class = no
1 Split condition
2 Class distribution
3 Class label
entropy = 0.0
samples = 3
value = [0, 3]
class = yes
sunny ≤ 0.5
entropy = 1.0
samples = 2
value = [1, 1]
class = no
entropy = 0.0
samples = 3
value = [3, 0]
class = no
False ≤ 0.5
entropy = 1.0
samples = 2
value = [1, 1]
class = no
entropy = 0.0
samples = 1
value = [1, 0]
class = no
entropy = 0.0
samples = 1
value = [0, 1]
class = yes
entropy = 0.0
samples = 1
value = [1, 0]
class = no
entropy = 0.0
samples = 1
value = [0, 1]
class = yes

Changing Parameters

```
clf = tree.DecisionTreeClassifier(criterion = "entropy",
                                  max_depth = 3,
                                  min_samples_split = 3,
                                  min_samples_leaf = 2)
```

criterion : measures used for decision tree (“entropy” or “gini”)
max_depth : maximum depth of the decision tree
min_samples_split : the minimum number of samples required to split an internal node
min_samples_leaf : the minimum number of samples in a leaf

Effect of Tree Size

```
clf = tree.DecisionTreeClassifier(criterion='entropy')
cv = KFold(n_splits = 10,
           shuffle=True,
           random_state=0)
cv_results = cross_val_score(clf, en_X, y, cv=cv)

print(cv_results.mean())
```

Unlimited tree size

```
0.6294334975369458
```

```
clf = tree.DecisionTreeClassifier(criterion='entropy',
                                  max_depth=3)
cv = KFold(n_splits = 10,
           shuffle=True,
           random_state=0)
cv_results = cross_val_score(clf, en_X, y, cv=cv)

print(cv_results.mean())
```

Tree depth is limited to 3

```
0.7201970443349753
```

MODEL EVALUATION

Classifier Evaluation Metrics: Confusion Matrix

Confusion Matrix:

Actual class\Predicted class	C_1	$\neg C_1$
C_1	**True Positives (TP)**	**False Negatives (FN)**
$\neg C_1$	**False Positives (FP)**	**True Negatives (TN)**

Example of Confusion Matrix:

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	**6954**	**46**	7000
buy_computer = no	**412**	**2588**	3000
Total	7366	2634	10000

- Given *m* classes, an entry, $\boldsymbol{CM_{i,j}}$ in a **confusion matrix** indicates # of tuples in class *i* that were labeled by the classifier as class *j*
- May have extra rows/columns to provide totals

Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

- **Classifier Accuracy,** or recognition rate: percentage of test set tuples that are correctly classified

  Accuracy = (TP + TN)/All

- **Error rate:** *1 – accuracy*, or

  Error rate = (FP + FN)/All

- **Class Imbalance Problem**:
  - One class may be *rare*, e.g. fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class
  - **Sensitivity**: True Positive recognition rate
    - **Sensitivity = TP/P**
  - **Specificity**: True Negative recognition rate
    - **Specificity = TN/N**

Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positive

$$precision = \frac{TP}{TP + FP}$$

- **Recall:** completeness – what % of positive tuples did the classifier label as positive?

$$recall = \frac{TP}{TP + FN}$$

- Perfect score is 1.0
- Inverse relationship between precision & recall
- ***F1* measure (*F_1* or *F1*-score)**: harmonic mean of precision and recall,

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

Classifier Evaluation Metrics: Example

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity*
cancer = no	**140**	**9560**	9700	98.56 (*specificity)*
Total	230	9770	10000	96.40 (*accuracy*)

- *Precision* = 90/230 = 39.13%   *Recall* = 90/300 = 30.00%

References (1)

- C. Apte and S. Weiss. **Data mining with decision trees and decision rules**. Future Generation Computer Systems, 13, 1997
- C. M. Bishop, **Neural Networks for Pattern Recognition**. Oxford University Press, 1995
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. **Classification and Regression Trees**. Wadsworth International Group, 1984
- C. J. C. Burges. **A Tutorial on Support Vector Machines for Pattern Recognition**. *Data Mining and Knowledge Discovery*, 2(2): 121-168, 1998
- P. K. Chan and S. J. Stolfo. **Learning arbiter and combiner trees from partitioned data for scaling machine learning**. KDD'95
- H. Cheng, X. Yan, J. Han, and C.-W. Hsu, **Discriminative Frequent Pattern Analysis for Effective Classification**, ICDE'07
- H. Cheng, X. Yan, J. Han, and P. S. Yu, **Direct Discriminative Pattern Mining for Effective Classification**, ICDE'08
- W. Cohen. **Fast effective rule induction**. ICML'95
- G. Cong, K.-L. Tan, A. K. H. Tung, and X. Xu. **Mining top-k covering rule groups for gene expression data**. SIGMOD'05

References (2)

- A. J. Dobson. **An Introduction to Generalized Linear Models**. Chapman & Hall, 1990.
- G. Dong and J. Li. **Efficient mining of emerging patterns: Discovering trends and differences**. KDD'99.
- R. O. Duda, P. E. Hart, and D. G. Stork. **Pattern Classification**, 2ed. John Wiley, 2001
- U. M. Fayyad. **Branching on attribute values in decision tree generation**. AAAI’94.
- Y. Freund and R. E. Schapire. **A decision-theoretic generalization of on-line learning and an application to boosting**. J. Computer and System Sciences, 1997.
- J. Gehrke, R. Ramakrishnan, and V. Ganti. **Rainforest: A framework for fast decision tree construction of large datasets**. VLDB’98.
- J. Gehrke, V. Gant, R. Ramakrishnan, and W.-Y. Loh, **BOAT -- Optimistic Decision Tree Construction**. SIGMOD'99.
- T. Hastie, R. Tibshirani, and J. Friedman. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction.** Springer-Verlag, 2001.
- D. Heckerman, D. Geiger, and D. M. Chickering. **Learning Bayesian networks: The combination of knowledge and statistical data**. Machine Learning, 1995.
- W. Li, J. Han, and J. Pei, **CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules**, ICDM'01.

References (3)

- T.-S. Lim, W.-Y. Loh, and Y.-S. Shih. **A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms.** Machine Learning, 2000.
- J. Magidson. **The Chaid approach to segmentation modeling: Chi-squared automatic interaction detection**. In R. P. Bagozzi, editor, Advanced Methods of Marketing Research, Blackwell Business, 1994.
- M. Mehta, R. Agrawal, and J. Rissanen. **SLIQ : A fast scalable classifier for data mining**. EDBT'96.
- T. M. Mitchell. **Machine Learning**. McGraw Hill, 1997.
- S. K. Murthy, **Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey**, Data Mining and Knowledge Discovery 2(4): 345-389, 1998
- J. R. Quinlan. **Induction of decision trees**. *Machine Learning*, 1:81-106, 1986.
- J. R. Quinlan and R. M. Cameron-Jones. **FOIL: A midterm report**. ECML’93.
- J. R. Quinlan. **C4.5: Programs for Machine Learning**. Morgan Kaufmann, 1993.
- J. R. Quinlan. **Bagging, boosting, and c4.5**. AAAI'96.

References (4)

- R. Rastogi and K. Shim. **Public: A decision tree classifier that integrates building and pruning**. VLDB'98.
- J. Shafer, R. Agrawal, and M. Mehta. **SPRINT : A scalable parallel classifier for data mining**. VLDB'96.
- J. W. Shavlik and T. G. Dietterich. **Readings in Machine Learning**. Morgan Kaufmann, 1990.
- P. Tan, M. Steinbach, and V. Kumar. **Introduction to Data Mining**. Addison Wesley, 2005.
- S. M. Weiss and C. A. Kulikowski. **Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems**. Morgan Kaufman, 1991.
- S. M. Weiss and N. Indurkhya. **Predictive Data Mining**. Morgan Kaufmann, 1997.
- I. H. Witten and E. Frank. **Data Mining: Practical Machine Learning Tools and Techniques**, 2ed. Morgan Kaufmann, 2005.
- X. Yin and J. Han. **CPAR: Classification based on predictive association rules**. SDM'03
- H. Yu, J. Yang, and J. Han. **Classifying large data sets using SVM with hierarchical clusters**. KDD'03.