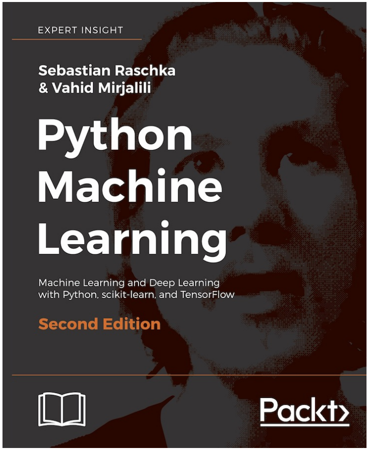
Machine Learning I (DATS 6202)
Decision Tree and Random Forest

Yuxiao Huang

Data Science, Columbian College of Arts & Sciences
George Washington University
yuxiaohuang@gwu.edu

October 22, 2018

Reference


EXPERT INSIGHT

Sebastian Raschka
& Vahid Mirjalili

Python
Machine
Learning

Machine Learning and Deep Learning
with Python, scikit-learn, and TensorFlow

Second Edition

Packt>

Picture courtesy of the website of the book code repository and info resource

Reference

- This set of slices is an excerpt of the book by Raschka and Mirjalili, with some trivial changes by the creator of the slides
- Please find the reference to and website of the book below:
  - *Raschka S. and Mirjalili V. (2017). Python Machine Learning. 2nd Edition.*
  - `https://sebastianraschka.com/books.html`
- Please find the website of the book code repository and info resource below:
  - `https://github.com/rasbt/python-machine-learning-book-2nd-edition`

Overview

1. Decision tree

2. Random forest

Interpretability

- An attractive property of decision tree classifier is interpretability
- We can simply read our decision from the tree by asking a sequence of questions
- Figure 1 (see next page) shows an example of a decision tree

Figure 1

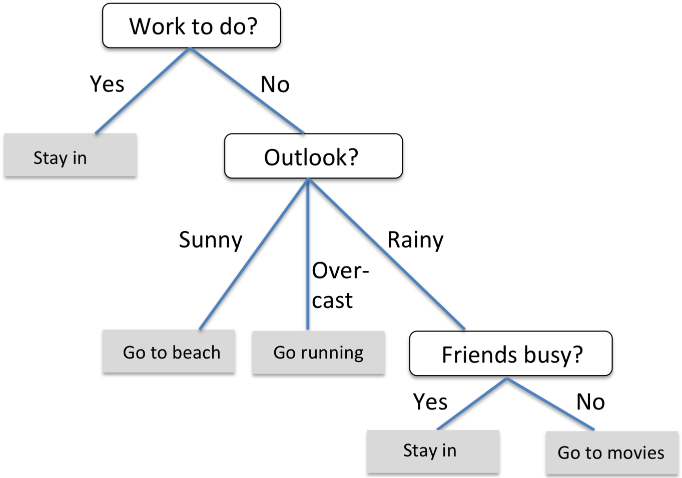
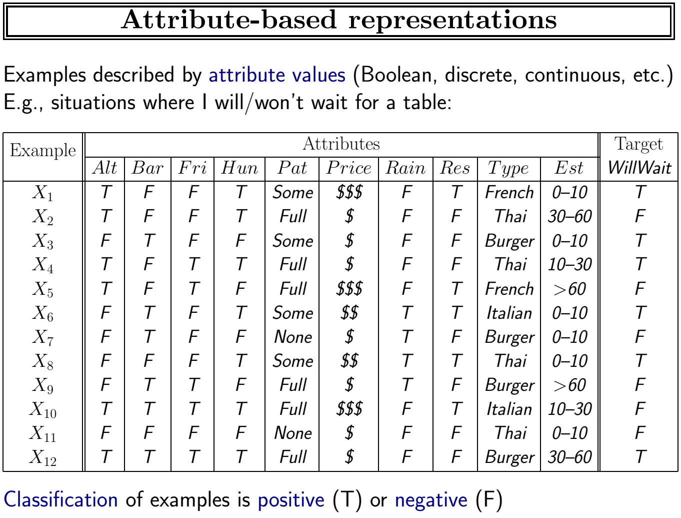

Work to do?
Yes
No
Stay in
Outlook?
Sunny
Over-cast
Rainy
Go to beach
Go running
Friends busy?
Yes
No
Stay in
Go to movies

Figure 2



Attribute-based representations

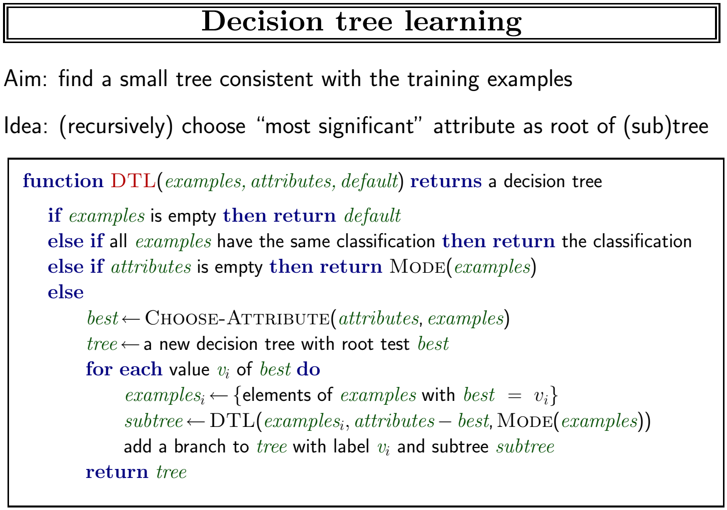
Examples described by attribute values (Boolean, discrete, continuous, etc.)
E.g., situations where I will/won't wait for a table:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*\$\$\$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*\$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*\$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*\$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*\$\$\$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*\$\$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*\$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*\$\$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*\$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*\$\$\$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*\$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*\$*	*F*	*F*	*Burger*	*30–60*	*T*

Classification of examples is positive (T) or negative (F)

Picture courtesy of the book *Artificial Intelligence: A Modern Approach (Third edition)*

Figure 3



Decision tree learning

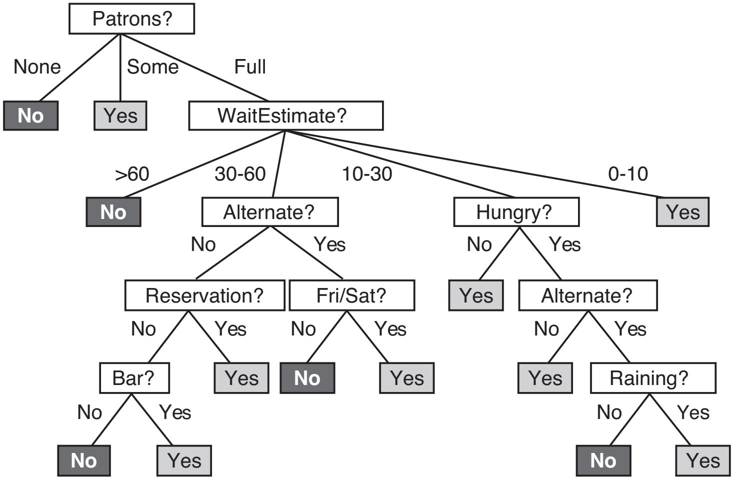
Aim: find a small tree consistent with the training examples

Idea: (recursively) choose "most significant" attribute as root of (sub)tree

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value $v_i$ of best do
            $examples_i$ ← {elements of examples with best = $v_i$}
            subtree ← DTL($examples_i$, attributes − best, MODE(examples))
            add a branch to tree with label $v_i$ and subtree subtree
        return tree
```

Picture courtesy of the book *Artificial Intelligence: A Modern Approach (Third edition)*

Figure 4



Picture courtesy of the book *Artificial Intelligence: A Modern Approach (Third edition)*

The objective function

- Here, our objective function at each split is the information gain:

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j) \quad \text{where}$$

  - f is the feature to perform the split
  - D_p and D_j are the dataset of the parent p and jth child node
  - I is our impurity measure
  - N_p and N_j are the number of samples at the parent and jth child node
- The information gain is simply the difference between the impurity of the parent node and the weighted sum of the child node impurities
  - the lower the child node impurities, the larger the information gain

The objective function

- Here, our objective function at each split is the information gain:

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$$

- **Q:** Shall we maximize or minimize it? Why?

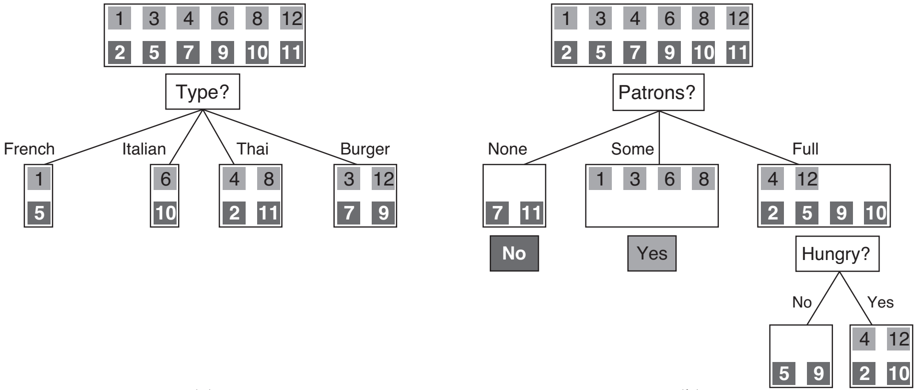
The objective function

- Here, our objective function at each split is the information gain:

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$$

- **Q:** Shall we maximize or minimize it? Why?
- **A:** Maximize it
  - generally the lower the impurity of the child nodes ($I(D_j)$), the better the classification
  - this is illustrated in Figure 5 (see next page)

Figure 5



(a)

(b)

Picture courtesy of the book *Artificial Intelligence: A Modern Approach (Third edition)*

The most informative feature

- So far, we have been using the concept of the most informative feature (a.k.a., the most significant attribute) without defining it
- Unsurprisingly, the most informative feature at each split is the one that maximizes the objective function

Binary decision trees

- A special case of decision tree is binary decision tree
- Here, each parent node is split into two child nodes, left and right
- Since the objective function for decision tree is

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$$

- The objective function for binary decision tree is?

Binary decision trees

- A special case of decision tree is binary decision tree
- Here, each parent node is split into two child nodes, left and right
- Since the objective function for decision tree is

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$$

- The objective function for binary decision tree is?

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

Measuring the impurity

- The three commonly used impurity measures
  - Entropy (I_H)
  - Gini impurity (I_G)
  - Classification error (I_E)

Entropy

- The entropy of node t, $I_H(t)$, is defined as follows

$$I_H(t) = -\sum_{i=1}^{c} p(i|t) \log_2 p(i|t)$$

- Here, $p(i|t)$ is the proportion of the samples belonging to class i for a particular node t
- The entropy is 0 if all samples at a node belong to the same class
- The entropy is maximal if we have a uniform class distribution
- For example, in a binary class setting
  - the entropy is 0 if $p(i = 1|t) = 1$ or $p(i = 0|t) = 0$
  - the entropy is 1 if the classes are distributed uniformly with $p(i = 1|t) = 0.5$ and $p(i = 0|t) = 0.5$

Gini impurity

- The gini impurity of node t, $I_G(t)$, is defined as follows

$$I_G(t) = \sum_{i=1}^{c} p(i|t)(1 - p(i|t)) = 1 - \sum_{i=1}^{c} p(i|t)^2$$

- The gini impurity is 0 if all samples at a node belong to the same class
- The gini impurity is maximal if we have a uniform class distribution
- For example, in a binary class setting:
  - the gini impurity is 0 if $p(i = 1|t) = 1$ or $p(i = 0|t) = 0$
  - the gini impurity is 0.5 if the classes are distributed uniformly with $p(i = 1|t) = 0.5$ and $p(i = 0|t) = 0.5$
- While gini is the default setting for parameter `criterion` in scikit learn `DecisionTreeClassifier`, it usually yields very similar results as entropy

Classification error

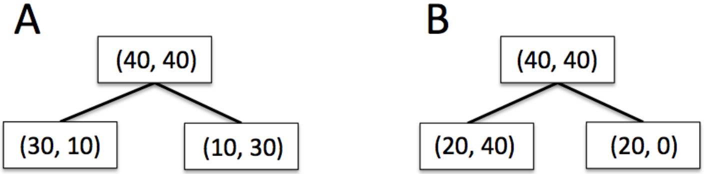
- The classification error of node t, $I_E(t)$, is defined as follows

$$I_E(t) = 1 - max\{p(i|t)\}$$

- Compared to entropy and gini impurity, classification error is less sensitive to changes in the class probabilities of the nodes, thus not recommended

Comparison between the three kinds of criterions

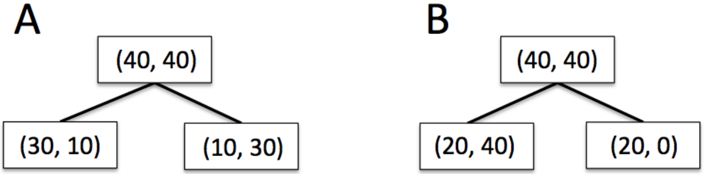
- Considering the following two splitting scenarios



- Which scenario is better?

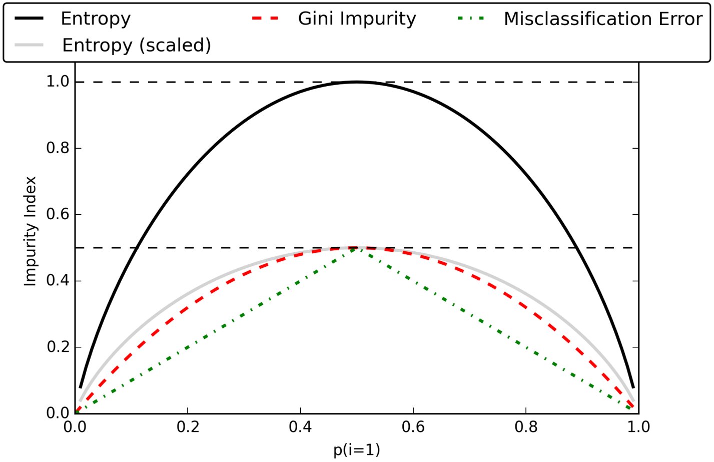
Comparison between the three kinds of criterions

- Considering the following two splitting scenarios



- Which scenario is better?
  - Entropy would favor scenario B ($IG_H = 0.31$) over A ($IG_H = 0.19$)
  - Gini would also favor scenario B ($IG_G = 0.16$) over A ($IG_H = 0.125$)
  - Classification error has no preference ($IG_E = 0.25$ for both scenarios)
- A visual comparison between the three is shown in Figure 6 (see next page)

Figure 6


Entropy
Gini Impurity
Misclassification Error
Entropy (scaled)
Impurity Index
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
p(i=1)

Problem of decision tree

- Decision tree tends to go too deep, resulting in overfitting
- Ways to limit the depth in scikit-learn `DecisionTreeClassifier`
    - `max_depth`: the maximum depth of the tree
    - `min_samples_split`: the minimum number of samples required to split an internal node
    - `min_samples_leaf`: the minimum number of samples required to be at a leaf node

Random forest

- Random forests have gained huge popularity in application of machine learning
- A random forest can be considered as an ensemble of decision trees
- The idea is to combine weak learners to build a more robust model

The random forest algorithm

- The random forest algorithm can be summarized in four simple steps
  1. Draw a random bootstrap sample of size n (randomly choose n samples from the training set with replacement)
  2. Grow a decision tree from the bootstrap sample. At each node:
     - 2.1 randomly select d features without replacement
     - 2.1 split the node using the feature that provides the best split according to the objective function (e.g., by maximizing the information gain)
  3. Repeat steps 1 and 2 k times
  4. Aggregate the prediction by each tree to assign the class label by majority vote

The hyperparameters

- In most implementations, the value of the hyperparameters, n (in step 1 of the algorithm), d (step 2.1), and k (step 3) are chosen as follows
    - $n =$ the number of samples in the training set
    - $d = \sqrt{m}$ where m is the number of features in the training set
    - the larger k, the better the performance of the random forest classifier, at the expense of an increased computational cost

Feature importance

- A unique property of random forest is that it can provide feature importance, which has been widely used in feature selection
- One application is screening the single nucleotide polymorphisms (SNPs)
- Figure 7 (see next page) shows a bar chart of feature importance in Iris dataset
- Figure 8 (see the last page) shows two scatter plots in the dataset

Figure 7

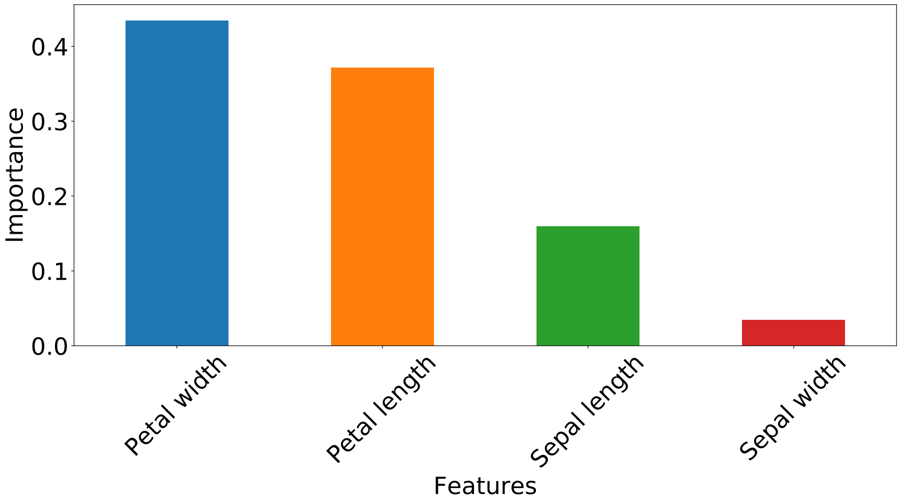
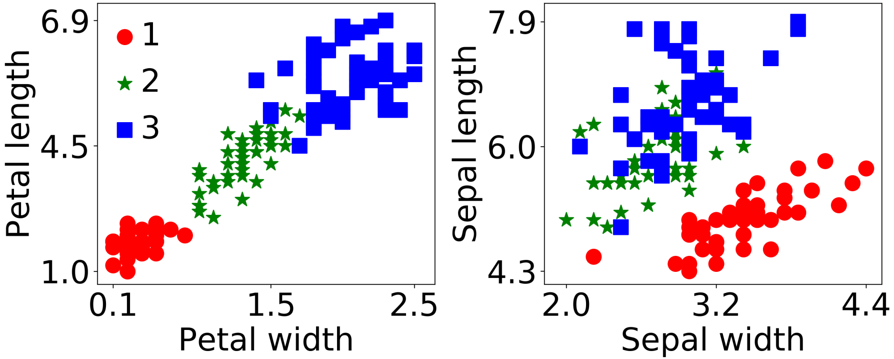

Importance
0.0
0.1
0.2
0.3
0.4
Petal width
Petal length
Sepal length
Sepal width
Features

Figure 8



1
2
3
Petal length
6.9
4.5
1.0
0.1
1.5
2.5
Petal width
Sepal length
7.9
6.0
4.3
2.0
3.2
4.4
Sepal width