COMP4131: Data Modelling and Analysis

Lecture 9: Decision Tree and Random Forest

Daokun Zhang

University of Nottingham Ningbo China

daokun.zhang@nottingham.edu.cn

April 14, 2025

Overview

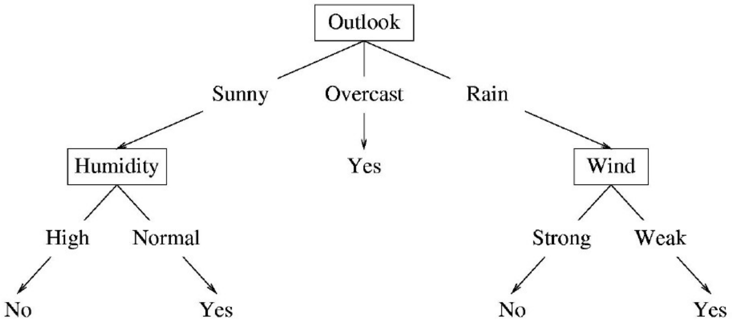
1 Decision Tree

2 Random Forest

Decision Tree

Classification with Decision Tree: An Overview

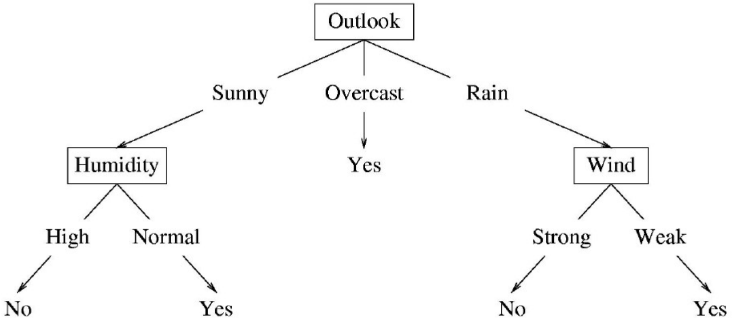
- A possible decision tree for the data:



- What prediction would we make for the new data (Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong)?

Classification with Decision Tree: An Overview

- A possible decision tree for the data:



- Each internal node: test one attribute, like "Outlook".
- Each branch from a node: select one value for the attribute, like "Outlook = Humidity".
- Each leaf node: predict the class label, like "PlayTennis = No".

The History of Decision Tree

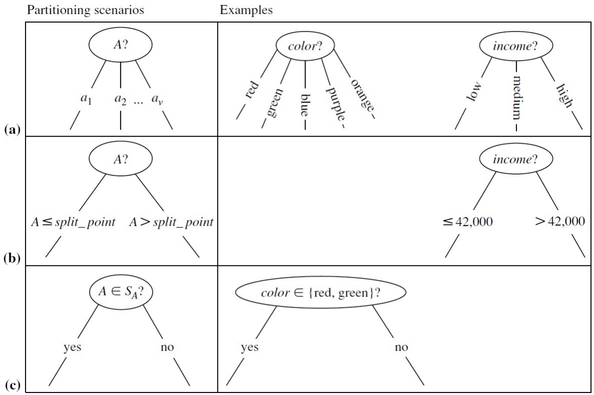
John Ross Quinlan

- ID3
  - During the late 1970s and early 1980s, John Ross Quinlan, developed a decision tree algorithm known as ID3 (Iterative Dichotomiser).
- C4.5
  - Quinlan later presented C4.5 (as successor of ID3), which then served as a competitive baseline to benchmark new machine learning algorithms.
- CART
  - In 1984, a group of statisticians published the book Classification and Regression Trees (CART), which described the generation of binary decision trees.

Decision Tree: Core Idea

- In ID3, C4.5, and CART, decision trees are constructed in a top-down recursive divide-and-conquer manner.
- The decision tree algorithm starts with a set of training samples including attributes and associated labels.
- The training set is recursively partitioned into smaller subsets as the tree is being built.
- Each partition requires to select an attribute and define a partition rule according to the choice of attribute values.
- Greedy algorithms are generally used to do iterative attribute selection and training set partition.

Decision Tree: Data Partition Rules



(a): Branch according to the value of the categorical-valued attribute A.
(b): Branch using a "split_point" for the continuous-valued attribute A.
(c): Branch with a value subset for the categorical-valued attribute A.
Different from scenario (a), scenarios (b) and (c) construct a binary tree.

Decision Tree Algorithm for Categorical-valued Data

```
1:  **Input**: training dataset $\mathcal{D}$, $attribute\_list$;
2:  create a node $N$;
3:  if all samples in $\mathcal{D}$ have the same class $C$ then
4:      return $N$ as a leaf node labeled with the class $C$;
5:  if $attribute\_list$ is empty then
6:      return $N$ as a leaf node labeled with the majority class in $\mathcal{D}$;
7:  $splitting\_attribute \leftarrow$ select an attribute from $attribute\_list$;
8:  $attribute\_list \leftarrow attribute\_list - splitting\_attribute$;
9:  $splitting\_criterion \leftarrow$ split $\mathcal{D}$ according to samples' categorical values at
    $splitting\_attribute$;
10: for each outcome $j$ of $splitting\_criterion$ do
11:     let $\mathcal{D}_j$ be the subset of samples in $\mathcal{D}$ satisfying outcome $j$;
12:     if $\mathcal{D}_j$ is empty then
13:         attach a leaf labeled with the majority class in $\mathcal{D}$ to node $N$;
14:     else
15:         go to step 1 to generate a decision tree with input ($\mathcal{D}_j$, $attribute\_list$);
16:         attach the generated decision tree to node $N$;
17: return $N$.
```

Attribute Selection Measure: Information Gain

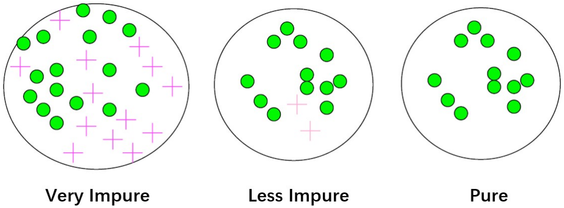
- ID3 uses information gain as its attribute selection measure.
- The attribute with the highest information gain is chosen as the splitting attribute for current node N.
- The highest information gain means that after partition, the information required to classify samples is minimized.
- The expected information needed to classify a sample in $\mathcal{D}$ is given by

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

  where m is the number of class, p_i is the probability that an arbitrary sample in $\mathcal{D}$ belongs to class C_i and is estimated by $|\mathcal{C}_{i,\mathcal{D}}|/|\mathcal{D}|$, with $\mathcal{C}_{i,\mathcal{D}}$ being the set of samples in $\mathcal{D}$ having class C_i.
- $\text{Info}(\mathcal{D})$ is just the average amount of information needed to identify the class label of a sample in $\mathcal{D}$, which is encoded in bits with a log function with base 2.

Attribute Selection Measure: Information Gain

- Suppose we are to partition the tuples in $\mathcal{D}$ on some categorical attribute A having v distinct values, $(a_1, a_2, \cdots, a_v)$, as observed from the training data.
- Then we can split $\mathcal{D}$ into v subsets $\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_v\}$ where $\mathcal{D}_j$ contains those samples in $\mathcal{D}$ that have the attribute value $A = a_j$.
- These subsets would correspond to the branches grown from node N.
- Ideally, we would like this partitioning to produce an exact classification of the samples in $\mathcal{D}$. That is, we would like for each subset to be pure.



Attribute Selection Measure: Information Gain

- How much information would we still need (after the partitioning) to arrive at an exact classification? This amount is measured by

$$\text{Info}_A(\mathcal{D}) = \sum_{j=1}^{v} \frac{|\mathcal{D}_j|}{|\mathcal{D}|} \times \text{Info}\left(\mathcal{D}_j\right).$$

- The term $|\mathcal{D}_j|/|\mathcal{D}|$ acts as the weight of the jth partition.
- $\text{Info}_A(\mathcal{D})$ is the expected information required to classify a sample from $\mathcal{D}$ after the partitioning by attribute A.
- The smaller the expected information (still) required, the greater the purity of the subsets.
- Information gain is defined as the difference between the original information requirement and the new requirement (i.e., obtained after partitioning on A). That is,

$$\text{Gain}(A) = \text{Info}(\mathcal{D}) - \text{Info}_A(\mathcal{D}).$$

Attribute Selection Measure: Information Gain

- Gain(A) tells us how much would be gained by branching on A.
- It is the expected reduction in the information requirement caused by knowing the value of A.
- The attribute A with the highest information gain, Gain(A), is chosen as the splitting attribute at node N.
- This is equivalent to saying that we want to partition on the attribute A that would do the "best classification", so that the amount of information still required to finish classifying the samples is minimal (i.e., minimum $\mathrm{Info}_A(\mathcal{D})$).

Attribute Selection Measure: Information Gain

Class-Labeled Training Sample from the *AllElectronics* Customer Database

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Attribute Selection Measure: Information Gain

- We first compute the expected information needed to classify a sample in $\mathcal{D}$:

$$\text{Info}(\mathcal{D}) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits.}$$

- The expected information needed to classify a sample in $\mathcal{D}$ if the samples are partitioned according to *age* is

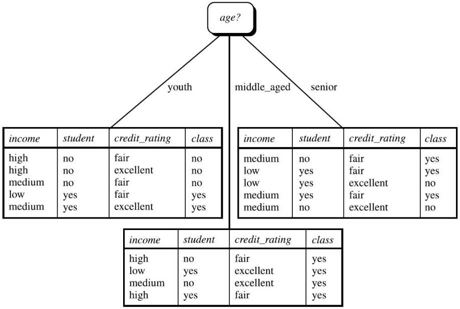
$$\text{Info}_{age}(D) = \frac{5}{14}\times\left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) + \frac{4}{14}\times\left(-\frac{4}{4}\log_2\frac{4}{4}\right) + \frac{5}{14}\times\left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) = 0.694 \text{ bits.}$$

- Hence, the gain in information from such a partitioning would be

$$\text{Gain}(age) = \text{Info}(\mathcal{D}) - \text{Info}_{age}(\mathcal{D}) = 0.940 - 0.694 = 0.246 \text{ bits.}$$

Attribute Selection Measure: Information Gain

- Similarly, we can calculate the information gain for other attributes as
  - Gain(*income*) = 0.029 bits.
  - Gain(*student*) = 0.151 bits.
  - Gain(*credit_rating*) = 0.048 bits.
- Because age has the highest information gain among the attributes, it is selected as the splitting attribute.
- Node N is labeled with *age*, and branches are grown for each of the attribute's values.



youth:

income	*student*	*credit_rating*	*class*
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

middle_aged:

income	*student*	*credit_rating*	*class*
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

senior:

income	*student*	*credit_rating*	*class*
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

Attribute Selection Measure: Information Gain

- But how can we compute the information gain of an attribute that is continuous-valued, unlike in the example?
- For such a scenario, we must determine the "best" *split_point* for the continuous-valued attribute A, where the *split_point* is a threshold on A.
- We first sort the values of A observed in the training set in increasing order, also denoted as $(a_1, a_2, \cdots, a_v)$.
- Typically, the midpoint between each pair of adjacent values, a_i and a_{i+1}, is considered as a possible *split_point*, $\frac{a_i + a_{i+1}}{2}$.
- For each possible *split_point* for A, we evaluate $\text{Info}_A(\mathcal{D})$, where the number of split subsets is two.
- We can denote $\mathcal{D}_1$ as the set of samples in $\mathcal{D}$ satisfying $A \leq \textit{split\_point}$, $\mathcal{D}_2$ as the set of samples in $\mathcal{D}$ satisfying $A > \textit{split\_point}$.

Attribute Selection Measure: Gain Ratio

- The information gain measure is biased toward splits with many outcomes. That is, it prefers to select attributes having a large number of values.
- For the *AllElectronics* Customer Database, if we choose the record Id (*RID*) as an attribute, the expected information needed to classify a sample in $\mathcal{D}$ if the samples are partitioned according to *RID* is

$$\text{Info}_{RID}(\mathcal{D}) = \frac{1}{14} \times \left(-\frac{1}{1}\log_2\frac{1}{1}\right) + \cdots + \frac{1}{14} \times \left(-\frac{1}{1}\log_2\frac{1}{1}\right) = 0 \text{ bits.}$$

- The information gain for the partition at *RID* is

$$\text{Gain}(RID) = \text{Info}(\mathcal{D}) - \text{Info}_{RID}(\mathcal{D}) = 0.940 - 0 = 0.940 \text{ bits} > \text{Gain}(age).$$

- The *RID* attribute would be selected to partition $\mathcal{D}$. However, this partition is meaningless, as test samples have different record IDs, making the trained decision tree unable to generalize.

Attribute Selection Measure: Gain Ratio

- C4.5, a successor of ID3, uses an extension to information gain known as gain ratio, which attempts to overcome this bias, by normalizing information gain with a "split information" value defined analogously with $\text{Info}(\mathcal{D})$ as

$$\text{SplitInfo}_A(\mathcal{D}) = -\sum_{j=1}^{v} \frac{|\mathcal{D}_j|}{|\mathcal{D}|} \times \log_2\left(\frac{|\mathcal{D}_j|}{|\mathcal{D}|}\right),$$

  which would be large if the number of split subsets v is large.

- The gain ratio is defined as

$$\text{GainRatio}(A) = \frac{\text{Gain}(A)}{\text{SplitInfo}_A(D)}.$$

- The attribute with the maximum gain ratio is selected as the splitting attribute.

Attribute Selection Measure: Gini Index

- CART uses Gini index to measures the impurity of sample set $\mathcal{D}$,

$$\mathrm{Gini}(\mathcal{D}) = 1 - \sum_{i=1}^{m} p_i^2,$$

  where p_i is the probability that a sample in $\mathcal{D}$ belongs to class C_i and is estimated by $|\mathcal{C}_{i,\mathcal{D}}|/|\mathcal{D}|$, with $\mathcal{C}_{i,\mathcal{D}}$ being the set of samples in $\mathcal{D}$ having class C_i. The sum is computed over m classes.
- The Gini index considers a binary split for each attribute.
  - For the case A is a categorical-valued attribute having v distinct values, $(a_1, a_2, \cdots, a_v)$, each subset, $\mathcal{S}_A$, can be considered as a binary test for attribute A of the form "$A \in \mathcal{S}_A$?"
  - For continuous-valued attributes, each possible split-point must be considered to construct a binary split "$A \leq split\_point$" and "$A > split\_point$".

Attribute Selection Measure: Gini Index

- When considering a binary split, we compute a weighted sum of the impurity of each resulting partition.
- For example, if a binary split on A partitions $\mathcal{D}$ into $\mathcal{D}_1$ and $\mathcal{D}_2$, the Gini index of $\mathcal{D}$ given that partitioning is

$$\mathrm{Gini}_A(\mathcal{D}) = \frac{|\mathcal{D}_1|}{|\mathcal{D}|}\,\mathrm{Gini}\left(\mathcal{D}_1\right) + \frac{|\mathcal{D}_2|}{|\mathcal{D}|}\,\mathrm{Gini}\left(\mathcal{D}_2\right).$$

- The reduction in impurity/Gini index that would be incurred by a binary split on a categorical- or continuous-valued attribute A is

$$\Delta\,\mathrm{Gini}(A) = \mathrm{Gini}(\mathcal{D}) - \mathrm{Gini}_A(\mathcal{D}).$$

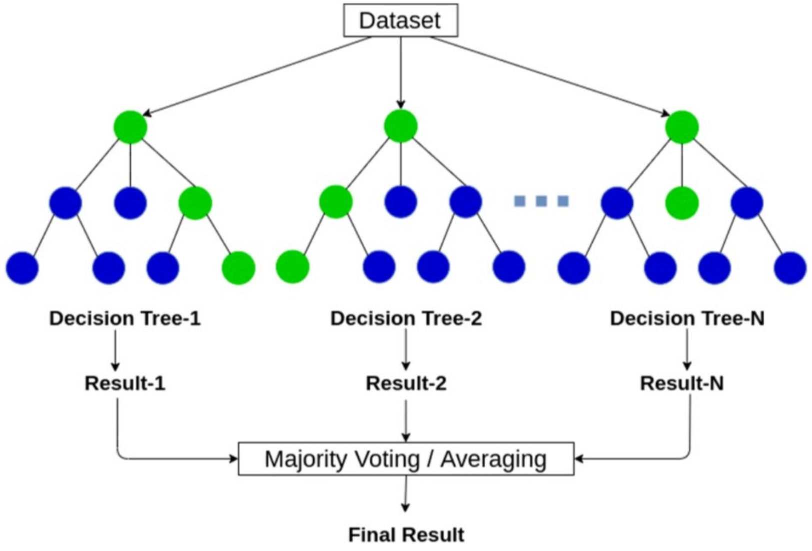
- The attribute that maximizes the reduction in impurity/Gini index is selected as the splitting attribute.
- This attribute and either its splitting subset (for a categorical-valued splitting attribute) or split-point (for a continuous-valued splitting attribute) together form the splitting criterion.

Decision Tree Classification and Regression

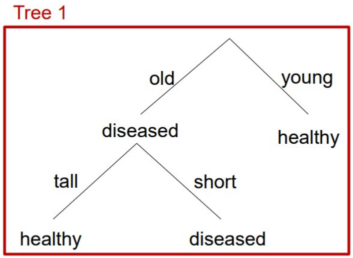
- Through iteratively selecting an attribute to split the set of training samples at each node, we can construct a complete tree.
- The Information Gain, Gain Ratio and Gini Index measures are designed to select splitting attributes for the classification case, and finally the majority class of the samples in each leaf node is assigned as the class of the leaf node.
- For the regression case, the reduction in the mean squared error of average prediction can be used to select splitting attributes, and finally the average target value of the samples in each leaf node is assigned as the target value of the leaf node.

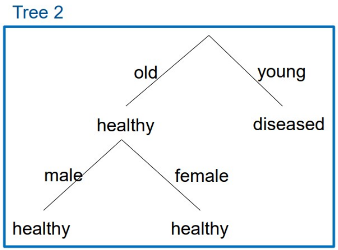
Random Forest

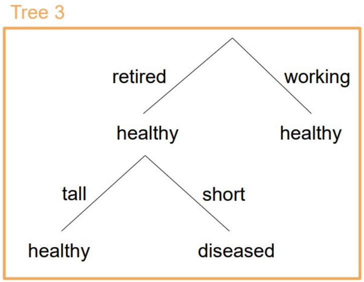
Intuition of Random Forest



Random Forest Classification







- New sample: (old, retired, male, short)
- Tree predictions:
  - Tree 1: diseased
  - Tree 2: healthy
  - Tree 3: diseased
- Majority rule: diseased

Bagging

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set $X = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$ with responses $Y = (y_1, ..., y_n)$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For $i = 1, \cdots, B$:

1. Sample, with replacement, n training examples from X, Y; call these X_i, Y_i.
2. Train a classification or regression tree f_i on X_i, Y_i.

After training, predictions for unseen samples $\boldsymbol{x}'$ can be made by averaging the predictions from all the individual regression trees on $\boldsymbol{x}'$:

$$\hat{f}(\boldsymbol{x}') = \frac{1}{B} \sum_{i=1}^{B} f_i\left(\boldsymbol{x}'\right),$$

or by taking the majority vote in the case of classification trees.

Bagging Effects on Variance Reduction

For the regression case, recall the decomposition of the mean squared error:

$$\text{Mean Squared Error} = \text{Bias}^2 + \text{Variance} + \text{Noise}.$$

If trees are sufficiently deep, they have very small bias.

Bagging can reduce the variance of of a single tree:

- For a given test example $\boldsymbol{x}'$, we can compare the variance of the bagging prediction $\hat{f}(\boldsymbol{x}')$ and a single tree's prediction $f_i(\boldsymbol{x}')$.
- Suppose under the randomness of the training set sampling,
  - the variance of $f_i(\boldsymbol{x}')$ is
    $$\text{Var}\left[f_i(\boldsymbol{x}')\right] = \sigma^2,$$
  - the co-variance between $f_i(\boldsymbol{x}')$ and $f_j(\boldsymbol{x}')$ with $i \neq j$ is
    $$\text{Cov}\left[f_i(\boldsymbol{x}'), f_j(\boldsymbol{x}')\right] = \rho\sigma^2,$$
    where ρ is the Pearson correlation coefficient between $f_i(\boldsymbol{x}')$ and $f_j(\boldsymbol{x}')$.

Bagging Effects on Variance Reduction

- We can calculate the variance of the bagging prediction $\hat{f}(\boldsymbol{x}')$ as

$$\begin{aligned}
\mathrm{Var}\left[\hat{f}(\boldsymbol{x}')\right] &= \mathrm{Var}\left[\frac{1}{B}\sum_{i=1}^{B} f_i\left(\boldsymbol{x}'\right)\right] \\
&= \frac{1}{B^2}\sum_{i=1}^{B}\sum_{j=1}^{B}\mathrm{Cov}\left[f_i(\boldsymbol{x}'), f_j(\boldsymbol{x}')\right] \\
&= \frac{1}{B^2}\sum_{i=1}^{B}\left\{\sum_{j=1,j\neq i}^{B}\mathrm{Cov}\left[f_i(\boldsymbol{x}'), f_j(\boldsymbol{x}')\right] + \mathrm{Var}\left[f_i(\boldsymbol{x}')\right]\right\} \\
&= \frac{1}{B^2}\sum_{i=1}^{B}\left[(B-1)\rho\sigma^2 + \sigma^2\right] \\
&= \frac{(B-1)\rho\sigma^2}{B} + \frac{\sigma^2}{B} \\
&= \rho\sigma^2 + \sigma^2\frac{1-\rho}{B}.
\end{aligned}$$

Bagging Effects on Variance Reduction

- From the identity $\mathrm{Var}\left[\hat{f}(\boldsymbol{x}')\right] = \rho\sigma^2 + \sigma^2\frac{1-\rho}{B}$, we can see that if $B \geq 2$, $\mathrm{Var}\left[\hat{f}(\boldsymbol{x}')\right] < \sigma^2 = \mathrm{Var}\left[f_i(\boldsymbol{x}')\right]$.
- However, the reduction of $\mathrm{Var}\left[\hat{f}(\boldsymbol{x}')\right]$ would be very small, if ρ is relatively large.
- From the identity $\mathrm{Var}\left[\hat{f}(\boldsymbol{x}')\right] = \frac{(B-1)\rho\sigma^2}{B} + \frac{\sigma^2}{B}$, we can find that if we want to reduce $\mathrm{Var}\left[\hat{f}(\boldsymbol{x}')\right]$ (irrespective of B), we need reduce the Pearson correlation coefficient ρ.
- Random Forest uses random feature selection (feature bagging) to de-correlate individual trees.

Random Forest

For $i = 1, \cdots, B$:

1. Sample, with replacement, n training examples from X, Y; call these X_i, Y_i.
2. Grow a classification or regression tree f_i on X_i, Y_i with feature bagging:
   2.1 At each partitioning, randomly select a subset of attributes from all attributes;
   2.2 Find the best attribute/split point from the attribute subset to do the split;

After training, predictions for unseen samples $\boldsymbol{x}'$ can be made by averaging the predictions from all the individual regression trees on $\boldsymbol{x}'$:

$$\hat{f}(\boldsymbol{x}') = \frac{1}{B}\sum_{i=1}^{B} f_i\left(\boldsymbol{x}'\right),$$

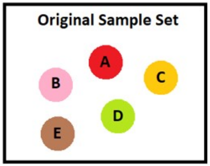
or by taking the majority vote in the case of classification trees.

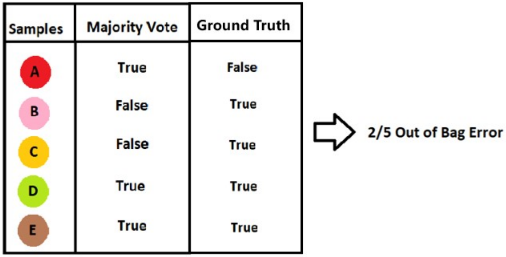
Random Forest Evaluation: Out-Of-Bag (OOB) Error

Random Forest use a bag (random subset) of samples to train each individual tree. The samples not in the set are called Out-Of-Bag (OOB) samples. As an evaluation metric of random forest, we can calculate the OOB error in the following procedure:

1. For each OOB sample, find all trees that are not trained by the OOB sample.
2. Take the majority vote of these trees' prediction for the OOB sample, or average these trees' prediction for regression case, compare the ensemble prediction to the true value of the OOB sample.
3. Calculate the OOB error as the averaged prediction error of all OOB samples.

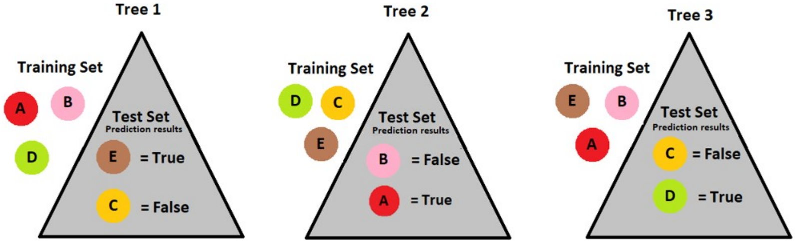
Random Forest Evaluation: OOB Error





Samples	Majority Vote	Ground Truth
A	True	False
B	False	True
C	False	True
D	True	True
E	True	True

2/5 Out of Bag Error



Random Forest Property: Attribute Importance Evaluation

With Random Forest, the importance of an attribute can be evaluated by the permutation importance. Taking the classification case as an example, the permutation importance for an attribute is calculated as

1. Record the prediction accuracy on the OOB samples for each tree.
2. Randomly permute the data for attribute a_j in the OOB samples, then record the accuracy again.
3. The decrease in accuracy as a result of this permuting is averaged over all trees, and is used as a measure of the importance of attribute a_j in the random forest.

The attribute importance evaluation can be used to do feature selection, remove irrelevant/noisy features, increase the generalization ability of the trained machine learning models.

References

- Decision Tree:
  - Jiawei Han, Micheline Kamber, and Jian Pei. "**Data Mining: Concepts and Techniques**." Third Edition. (2013) (Chapter 8.2 Decision Tree Induction)
- Random Forest:
  - `https://en.wikipedia.org/wiki/Random_forest`
  - `https://en.wikipedia.org/wiki/Out-of-bag_error`

The End