Tree-based 分類技術

Decision Tree決策樹

Ensemble技術

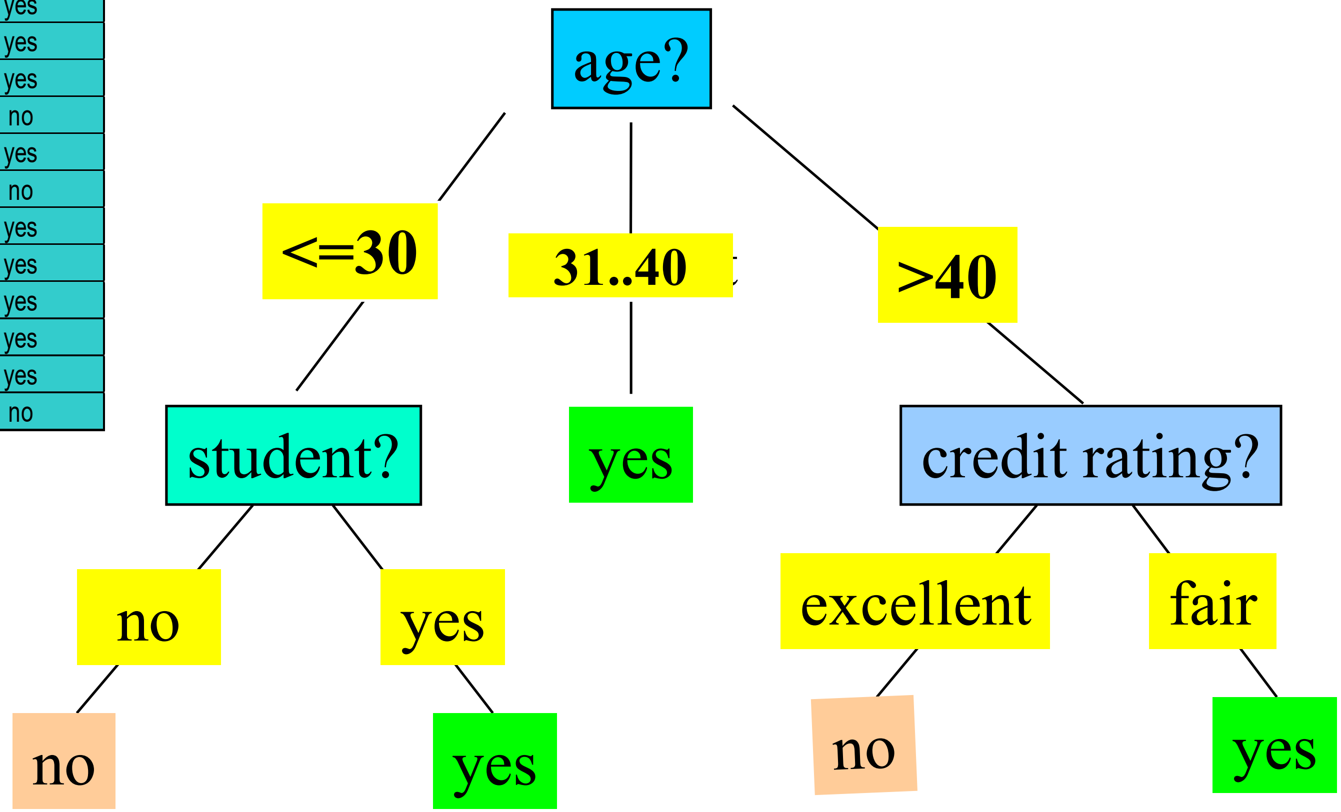
Decision Tree : An Example

- Training data set: Buys_computer
- The data set follows an example of Quinlan's ID3 (Playing Tennis)

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

Decision Tree : An Example

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



Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

三大主題

1. 特徵選擇
2. 決策樹的生成
3. 決策樹的剪枝

C5.0

	ID3	C4.5	CART
特征選擇	Information gain	Gain ratio	Gini index

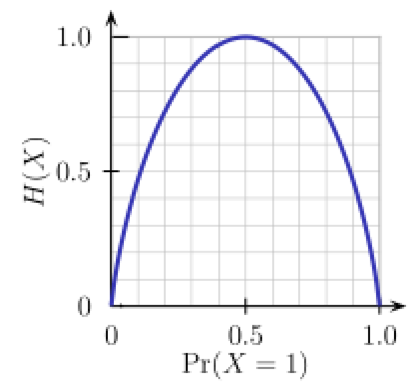
John Ross Quinlan
Iterative Dichotomiser 3 疊代二叉樹3代
https://en.wikipedia.org/wiki/ID3_algorithm

C4.5 https://en.wikipedia.org/wiki/C4.5_algorithm

CART (Classification And Regression Tree)
https://en.wikipedia.org/wiki/Decision_tree_learning

Brief Review of Entropy

- Entropy (Information Theory)
  - A measure of uncertainty associated with a random variable
  - Calculation: For a discrete random variable *Y* taking *m* distinct values $\{y_1, \ldots, y_m\}$,
    - $H(Y) = -\sum_{i=1}^{m} p_i \log(p_i)$, where $p_i = P(Y = y_i)$
  - Interpretation:
    - Higher entropy => higher uncertainty
    - Lower entropy => lower uncertainty
- Conditional Entropy
  - $H(Y|X) = \sum_x p(x) H(Y|X = x)$



m = 2

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"

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

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

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

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

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

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

Similarly,

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

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

Computing Information-Gain for Continuous-Valued Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the *best split point* for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
    - $(a_i+a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
  - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying A ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses **gain ratio** to overcome the problem (normalization to information gain)

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

  - GainRatio(A) = Gain(A)/SplitInfo(A)
- Ex.

$$SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557$$

  - gain_ratio(income) = 0.029/1.557 = 0.019
- The attribute with the **maximum gain ratio** is selected as the splitting attribute

$$\text{Gini}(p) = \sum_{k=1}^{K} p_k \cdot (1 - p_k) = 1 - \sum_{i=1}^{K} p_k^2$$

$$\text{Gini}(D) = 1 - \sum_{k=1}^{K} \left(\frac{|c_k|}{|D|} \right)^2$$

Gini Index (CART, IBM IntelligentMiner)

- If a data set D contains examples from n classes, gini index, $gini(D)$ is defined as

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

where p_j is the relative frequency of class j in D

- If a data set D is split on A into two subsets D_1 and D_2, the *gini* index $gini(D)$ is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)

Computation of Gini Index

- Ex. D has 9 tuples in buys_computer = "yes" and 5 in "no"

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute income partitions D into 10 in D_1: {low, medium} and 4 in D_2

$$gini_{income \in \{low, medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2)$$

$$\begin{aligned} &= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right) \\ &= 0.443 \\ &= Gini_{income \in \{high\}}(D). \end{aligned}$$

$Gini_{\{low,high\}}$ is 0.458; $Gini_{\{medium,high\}}$ is 0.450. Thus, split on the {low,medium} (and {high}) since it has the **lowest Gini index**

- All attributes are assumed continuous-valued
- May need other tools, e.g., clustering, to get the possible split values
- Can be modified for categorical attributes

Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - **Information gain**:
    - biased towards multivalued attributes
  - **Gain ratio**:
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - **Gini index**:
    - biased to multivalued attributes
    - has difficulty when # of classes is large
    - tends to favor tests that result in equal-sized partitions and purity in both partitions

Other Attribute Selection Measures

- CHAID: a popular decision tree algorithm, measure based on χ^2 test for independence
- C-SEP: performs better than info. gain and gini index in certain cases
- G-statistic: has a close approximation to χ^2 distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear comb. of attrs.
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others

ID3算法

1. Calculate the entropy of every attribute of the data set.
2. Partition ("split") the set into subsets using the attribute for which the resulting entropy after splitting is minimized; or, equivalently, **information gain is maximum**
3. Make a decision tree node containing that attribute.
4. Recurse on subsets using the remaining attributes.

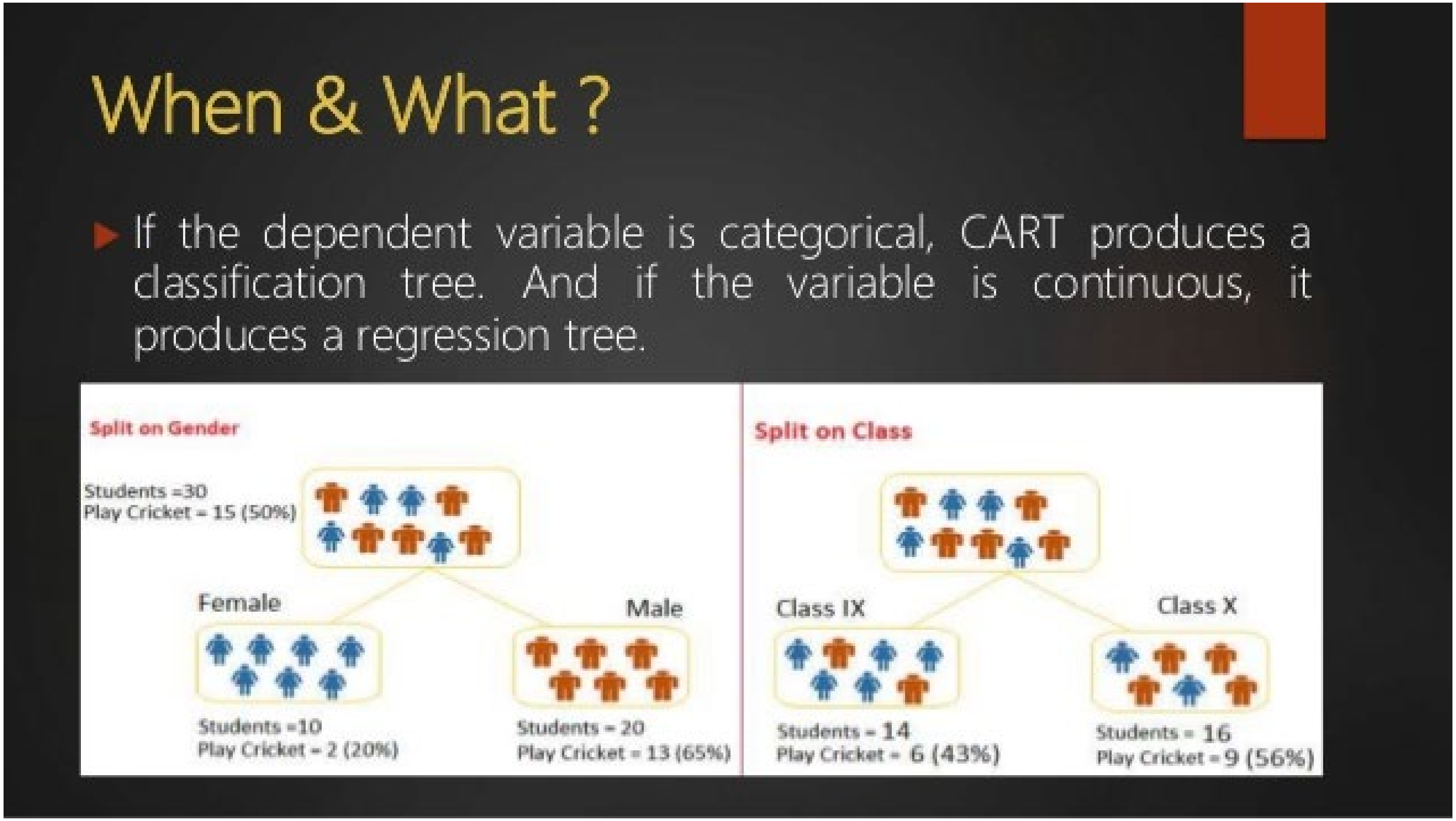
越是小型的決策樹越優於大的決策樹（簡單理論）。
儘管如此，該算法也不是總是生成最小的樹形結構。
是一個啟發式算法

C4.5算法

1. 檢查上述基本情況
2. 對於每個特徵a，計算劃分a的信息增益
3. 記a_best為最高信息增益的特徵
4. 創建一個在a_best上劃分的決策節點
5. 使用劃分後的樣本創建作為當前決策節點的子節點，並在這些子節點上遞歸地處理

CART – Classification & Regression Trees

https://www.slideshare.net/hemantchetwani/cart-classification-regression-trees



CART (Classification And Regression Tree)

Breiman, Leo; Friedman, J. H., Olshen, R. A., & Stone, C. J. Classification and regression trees. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software. 1984. ISBN 978-0-412-04841-8.

Decision trees are formed by a collection of rules based on variables in the modeling data set:

1. Rules based on variables' values are selected to get the best split to differentiate observations based on the dependent variable
2. Once a rule is selected and splits a node into two, the same process is applied to each "child" node (i.e. it is a recursive procedure)
3. Splitting stops when CART detects no further gain can be made, or some pre-set stopping rules are met. (Alternatively, the data are split as much as possible and then the tree is later pruned.)

Each branch of the tree ends in a terminal node.
Each observation falls into one and exactly one terminal node, and each terminal node is uniquely defined by a set of rules.

Overfitting and Tree Pruning(樹支修剪)

- 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 樹支修剪的兩種方法
  - Prepruning: *Halt tree construction early*-do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an **appropriate threshold**
  - Postpruning: *Remove branches* from a “fully grown” tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the “best pruned tree”

Enhancements to Basic Decision Tree Induction 基本決策樹的加強版

- Allow for **continuous-valued attributes**
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle **missing attribute values**
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- **Attribute construction**
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication

Sklearn
Decision Tree

sklearn.tree.DecisionTreeClassifier

Parameters 參數

class sklearn.tree. **DecisionTreeClassifier** (*criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False*) [source]

Attributes 屬性

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Methods 方法

apply (self, X[, check_input])	Returns the index of the leaf that each sample is predicted as.
decision_path (self, X[, check_input])	Return the decision path in the tree
fit (self, X, y[, sample_weight, ...])	Build a decision tree classifier from the training set (X, y).
get_depth (self)	Returns the depth of the decision tree.
get_n_leaves (self)	Returns the number of leaves of the decision tree.
get_params (self[, deep])	Get parameters for this estimator.
predict (self, X[, check_input])	Predict class or regression value for X.
predict_log_proba (self, X)	Predict class log-probabilities of the input samples X.
predict_proba (self, X[, check_input])	Predict class probabilities of the input samples X.
score (self, X, y[, sample_weight])	Returns the mean accuracy on the given test data and labels.
set_params (self, **params)	Set the parameters of this estimator.

L. Breiman, J. Friedman, R. Olshen, and C. Stone, “Classification and Regression Trees”, Wadsworth, Belmont, CA, 1984.

class sklearn.tree.DecisionTreeClassifier(
criterion=’**gini**’,
splitter=’**best**’,
max_depth=None,
min_samples_split=2,
min_samples_leaf=1,
min_weight_fraction_leaf=0.0,
max_features=None,
random_state=None,
max_leaf_nodes=None,
min_impurity_decrease=0.0,
min_impurity_split=None,
class_weight=None,
presort=False)

criterion : string, optional (default=”gini”)
The function to measure the quality of a split. Supported criteria are “**gini**” for the Gini impurity and “**entropy**” for the information gain.

splitter : string, optional (default=”best”)
The strategy used to choose the split at each node. Supported strategies are “best” to choose the best split and “random” to choose the best random split.

Sklearn
Decision Tree
DEMO

```
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier

# 導入數據
filename = 'pima_data.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

data = read_csv(filename, names=names)

# 將資料分為輸入資料和輸出結果
array = data.values

X = array[:, 0:8]
Y = array[:, 8]
```

```
num_folds = 10
seed = 7
kfold = KFold(n_splits=num_folds,
random_state=seed)

model = DecisionTreeClassifier()

result = cross_val_score(model, X, Y, cv=kfold)

print(result.mean())
```

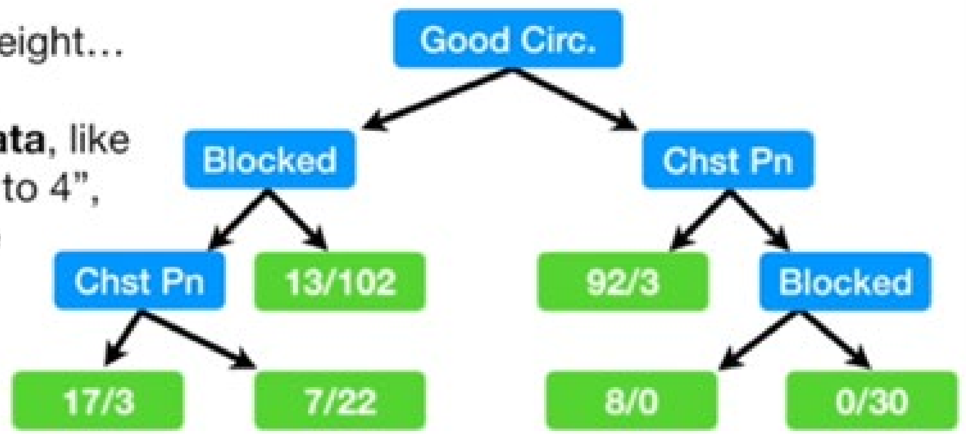
0.6886876281613123

Now we've seen how to build a tree with…

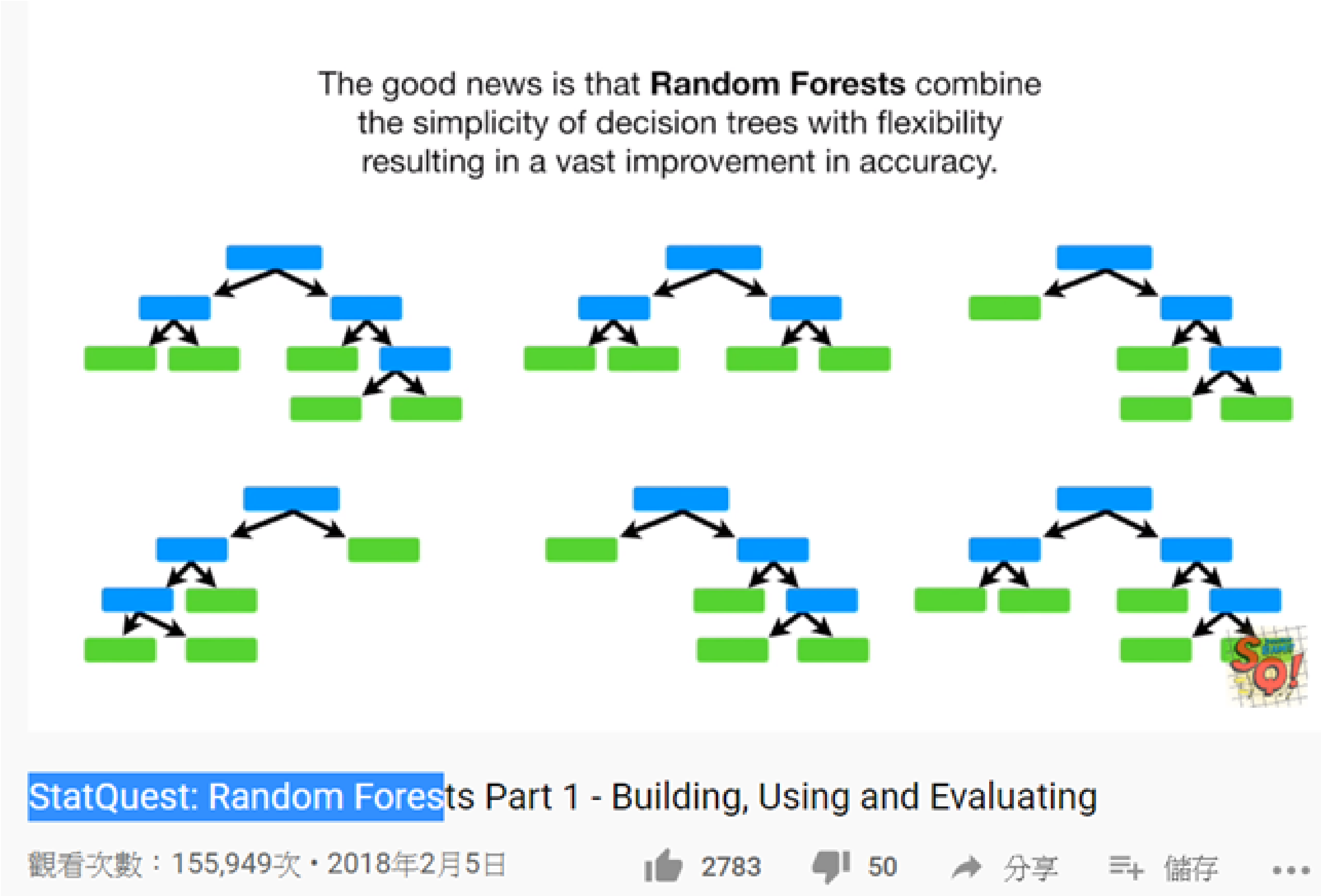
1) "yes/no" questions at each step…

2) Numeric data, like patient weight…

Now let's talk about **ranked data**, like "rank my jokes on a scale of 1 to 4", and **multiple choice data**, like "which color do you like, red, blue or green?"



StatQuest: Decision Trees
https://www.youtube.com/watch?v=7VeUPuFGJHk



StatQuest: Random Forests Part 1 - Building, Using and Evaluating
https://www.youtube.com/watch?v=J4Wdy0Wc_xQ

StatQuest: Random Forests Part 2: Missing data and clustering
https://www.youtube.com/watch?v=nyxTdL_4Q-Q

https://www.cs.ubc.ca/~nando/540-2013/lectures.html

Machine Learning — Lectures | Assignments | Project | Python

Lectures

Videos

After each lecture, you can download the videos **here** or watch them in youtube: **machine learning**.

The lectures for 340, the undergrad version of this course, are in youtube: **undergraduate machine learning**. I recommend you watch these prior to the 540 class.

Schedule

Tue Jan 8. Introduction to machine learning.

Th Jan 10. Linear prediction.

Tue Jan 15. Maximum likelihood and linear prediction.

Th Jan 17. Ridge, nonlinear regression with basis functions and Cross-validation.

Tue Jan 22. Ridge, nonlinear regression with basis functions and Cross-validation (continued).