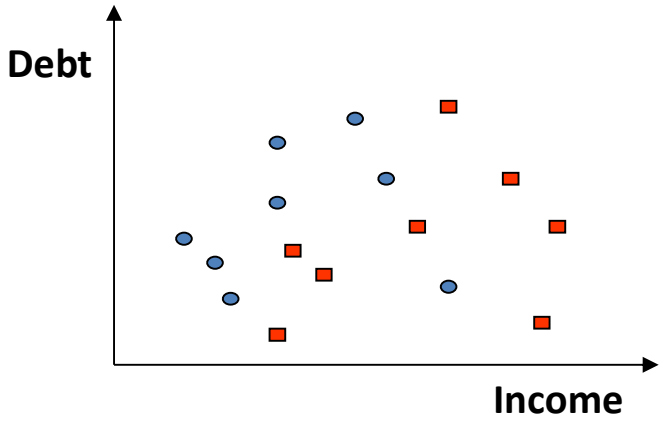
DECISION TREE

Classification example

ID	Income	Dept	대출 허용
1	13	2	Yes
2	6	12	No
3	3	3	Yes
4	4	10	No
new	7	5	?

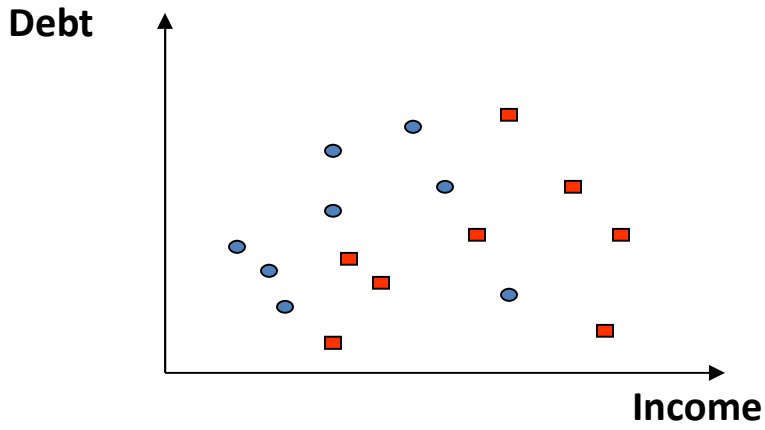


Understanding decision trees

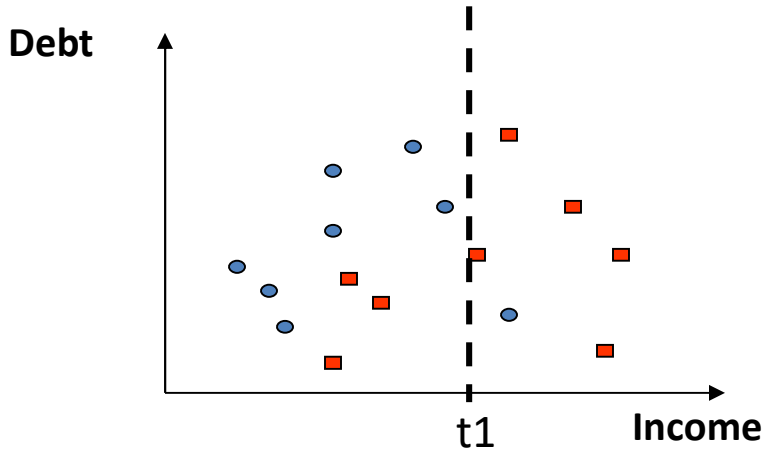
- A model in the form of a **tree structure**
- Nodes indicate a decision to be made on the attribute
- Branches: indicate the decision's choice
- Leaf nodes denote the result of a combination of decisions

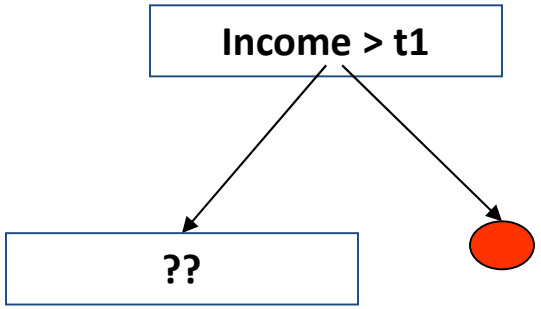
- Decision trees are built using a divide and conquer approach.

Decision Tree Example

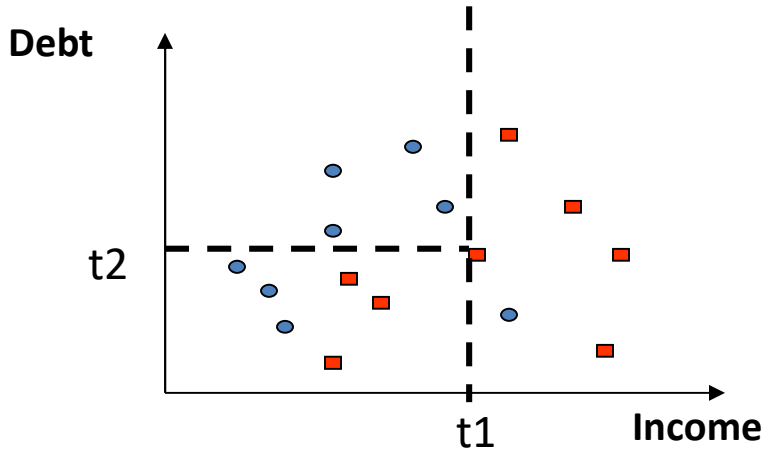


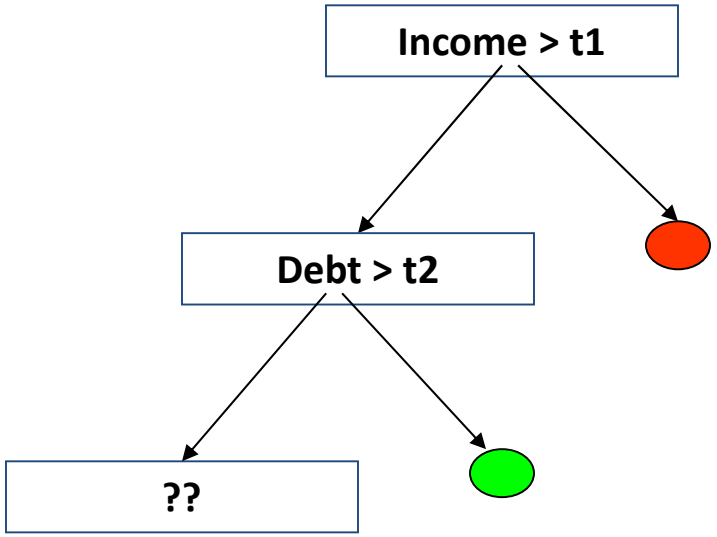
Decision Tree Example



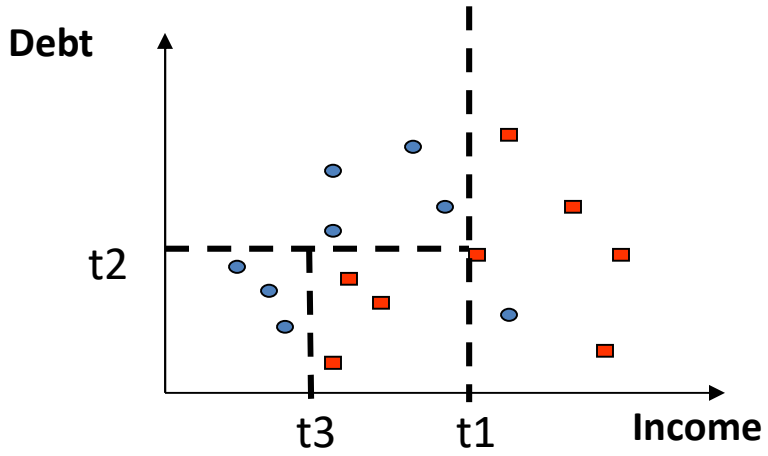


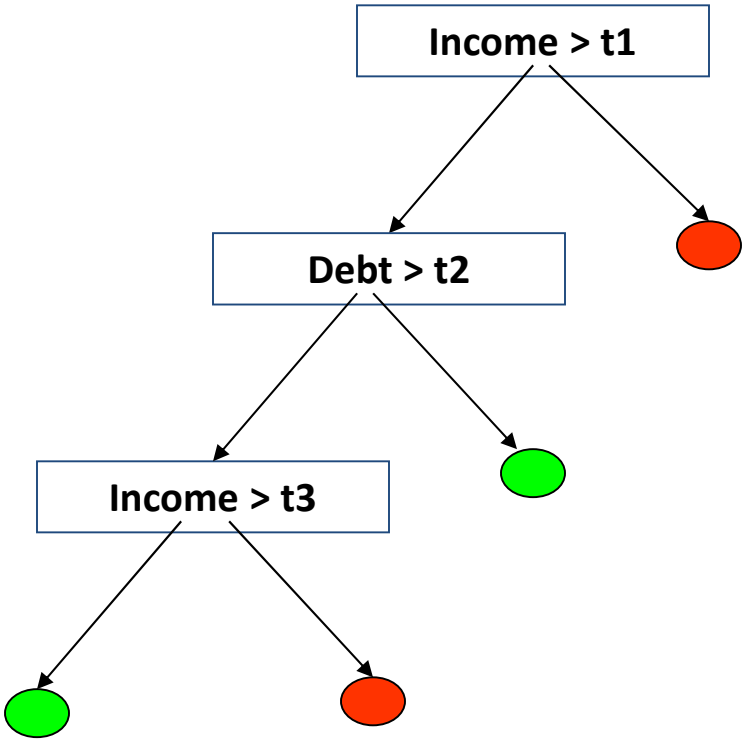
Decision Tree Example



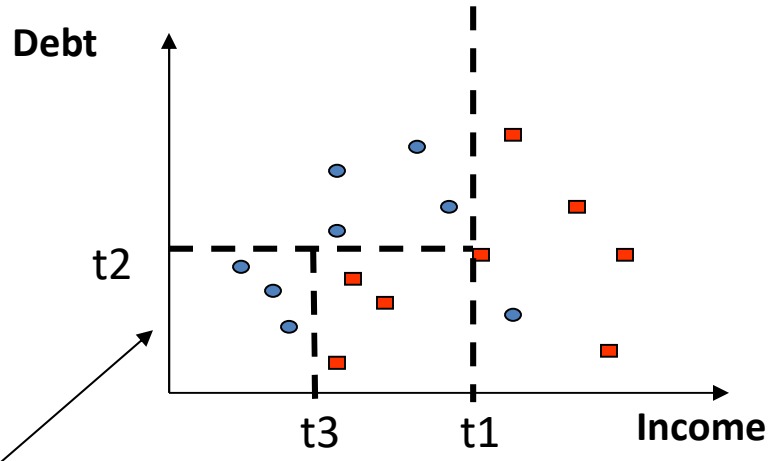


Decision Tree Example

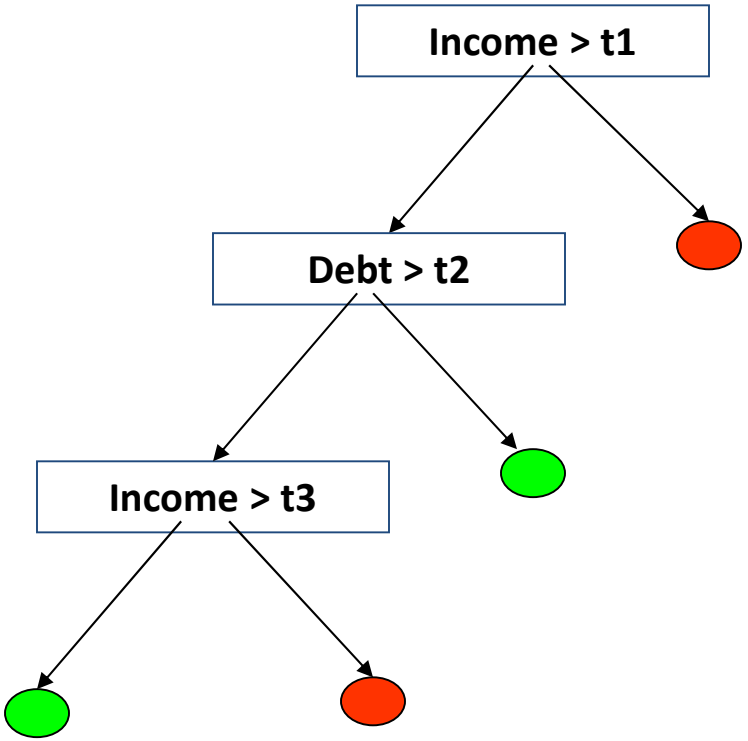




Decision Tree Example



Note: tree boundaries are piecewise linear and axis-parallel



Partitioning Up the Predictor Space

- One way to make predictions in a classification/regression problem is to divide the predictor space (i.e. all the possible values for for $X_1, X_2, \ldots, X_p$) into distinct regions, say $R_1, R_2, \ldots, R_k$

- Then for every X that falls in a particular region (say R_j) we make the same prediction

Some Natural Questions

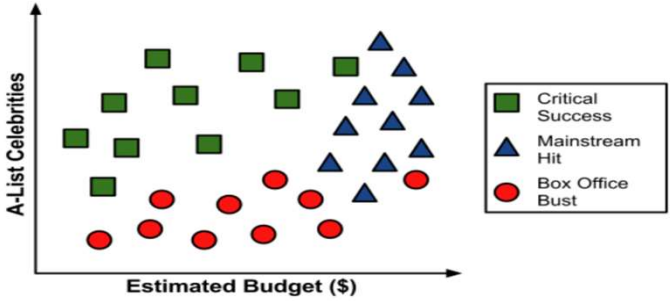
1. Where to split? i.e. how do we decide on what regions to use i.e. R_1, R_2,…,R_k or equivalently what tree structure should we use?

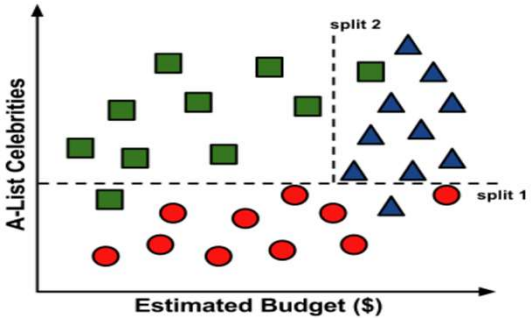
2. What values should we use for $\hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}_k$?

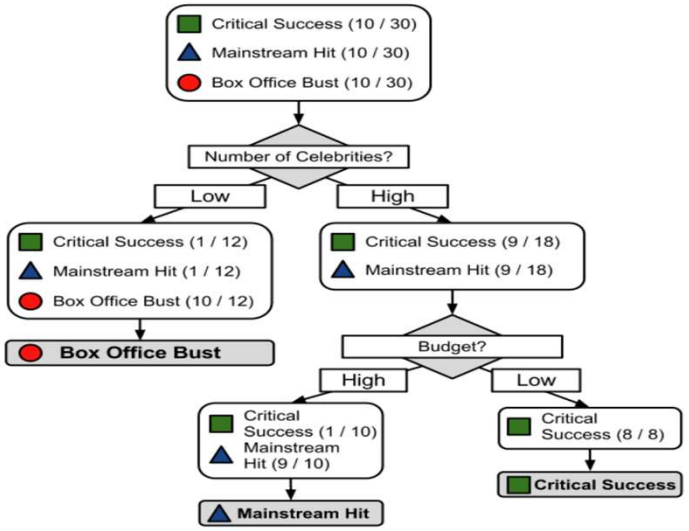
1. What values should we use for $\hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}_k$?

- Simple!
- For region R_j, the best prediction is simply
  - the most common category among the training data within that region (in case of classification)
  - the average of all the responses from our training data that fell in region R_j (in case of regression)

2. Where to Split?







look for feature values that split the data in such a way that partitions contain examples primarily of a single class.

C5.0 algorithm : 정보이론을 이용!

- C5.0 uses entropy for measuring purity

$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

- Entropy
  - '무질서한 정도', '규칙적이지 않은 정도', '불확실성'
  - 예1: 0과 1의 분포가 0.6:0.4인 경우

```
> -0.60 * log2(0.60) - 0.40 * log2(0.40)
[1] 0.9709506
```

  - 예2: 0과 1의 분포가 0.5:0.5인 경우

  - 예3: 0과 1의 분포가 1:0인 경우 (모두 0)

Where to split? : 정보이론을 이용!

- Which feature to split?

$$\text{InfoGain}(F) = \text{Entropy}(S_1) - \text{Entropy}(S_2)$$

the difference between the entropy before the split (*S1*) and after the split (*S2*):

$$\text{Entropy}(S) = \sum_{i=1}^{n} w_i \, \text{Entropy}(P_i)$$

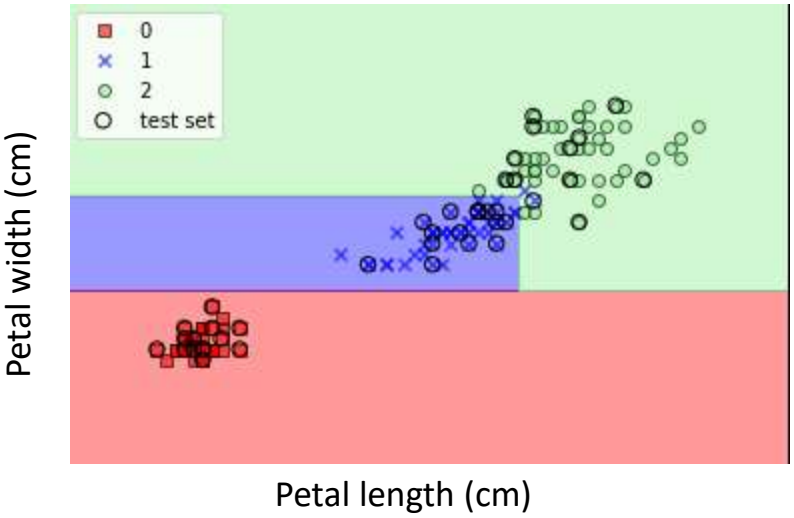
the sum of entropy of each of the n partitions weighted by its proportion (w_i).

  - The higher the information gain is, the better a feature is at creating homogeneous groups

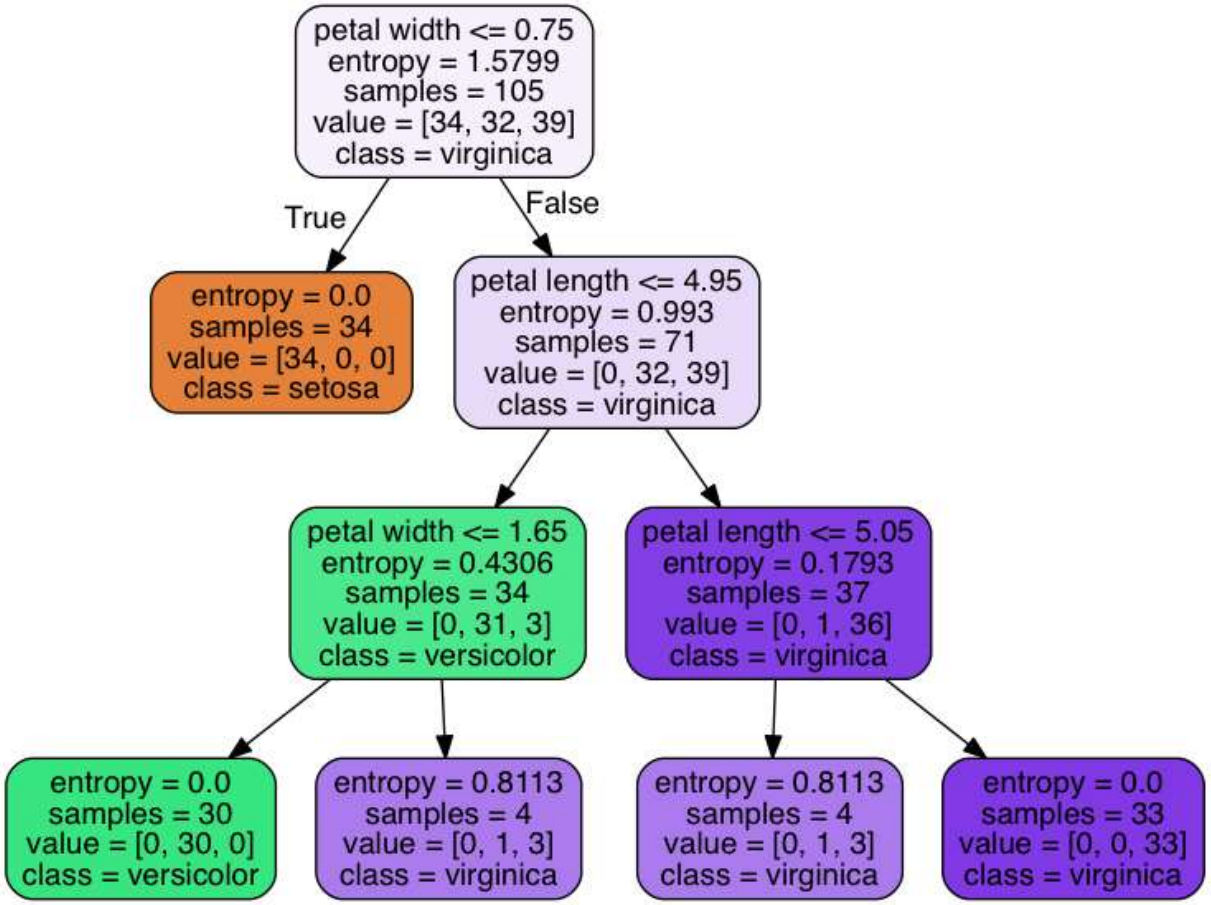
Example

- Building a decision tree

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
tree.fit(X_train, y_train)
```



Tree



https://github.com/rasbt/python-machine-learning-book/blob/master/code/ch03/ch03.ipynb

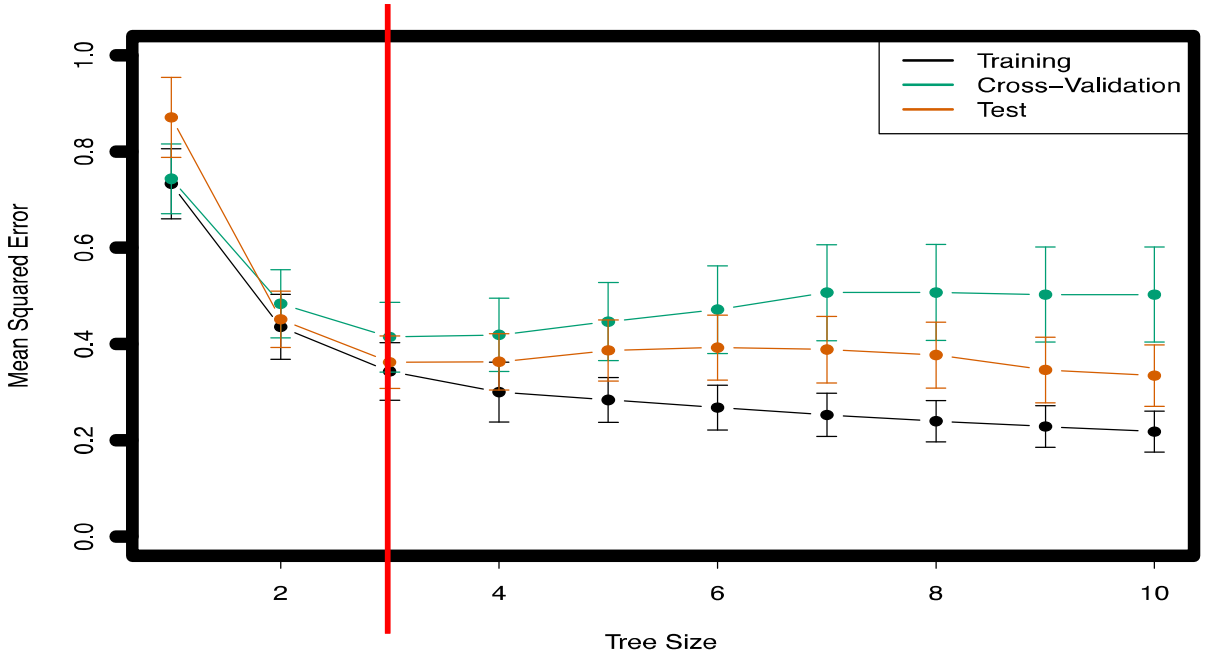
모델 학습

- 재귀적 분기 (recursive partitioning)

- 가지치기 (pruning)
  - A large tree (i.e. one with many terminal nodes) may tend to overfit the training data
  - Generally, we can improve accuracy by “pruning” the tree i.e. cutting off some of the terminal nodes
- How do we know how far back to prune the tree? We use **cross validation** to see which tree has the lowest error rate.

Example: Baseball Players' Salaries

- The minimum (cross validation) error occurs at a tree size of 3



Why trees are useful in Practice

- Can handle high dimensional data
  - But do not scale well to massive data sets (e.g. N in millions)
- Can handle any type of input variables
  - Most other methods require data of a single type
- Trees are (somewhat) interpretable

Pros and Cons of Decision Trees

- Pros:
  - Trees are very easy to explain to people (probably even easier than linear regression)
  - Trees can be plotted graphically, and are easily interpreted even by non-expert
  - They work fine on both classification and regression problems

- Cons:
  - Trees don’t have the same prediction accuracy as some of the more complicated approaches

Decision tree

- Tree-based
- Interpretable

- Training phase:
- Test phase:

- Pruning to get '*generalizable'* result
- Decision boundary:

RANDOM FOREST

Problem with a single best model

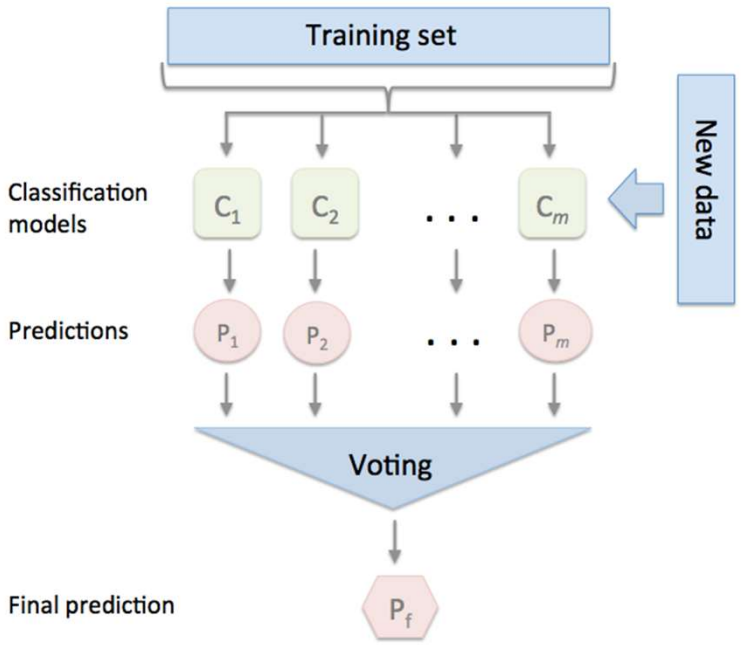
- Decision trees discussed earlier suffer from high variance!
  - If we randomly split the training data into 2 parts, and fit decision trees on both parts, the results could be quite different
- We would like to have models with *low variance* as well as low bias
- To solve this problem, we can use ensemble methods

General ensemble approach

- Use multiple learning algorithms to obtain better predictive performance than could be obtained from each learning algorithm alone.

- Start by m classifiers
  - Can be built from different algorithms
  - e.g. decision trees, SVM, etc.

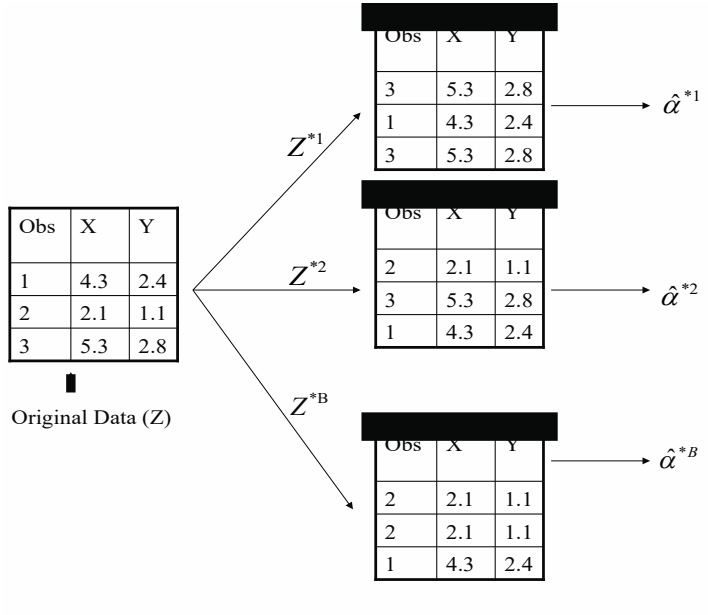
  - Or use the same base algorithm, fitting different subsets of training data
  - e.g. random forest (bootstrapping)



https://github.com/rasbt/python-machine-learning-book/tree/master/code/ch07/images

Bootstrapping

Resampling of the training dataset (and of equal size to the observed dataset), each of which is obtained by random sampling with replacement from the original dataset.



Produces plenty of training datasets

Bagging

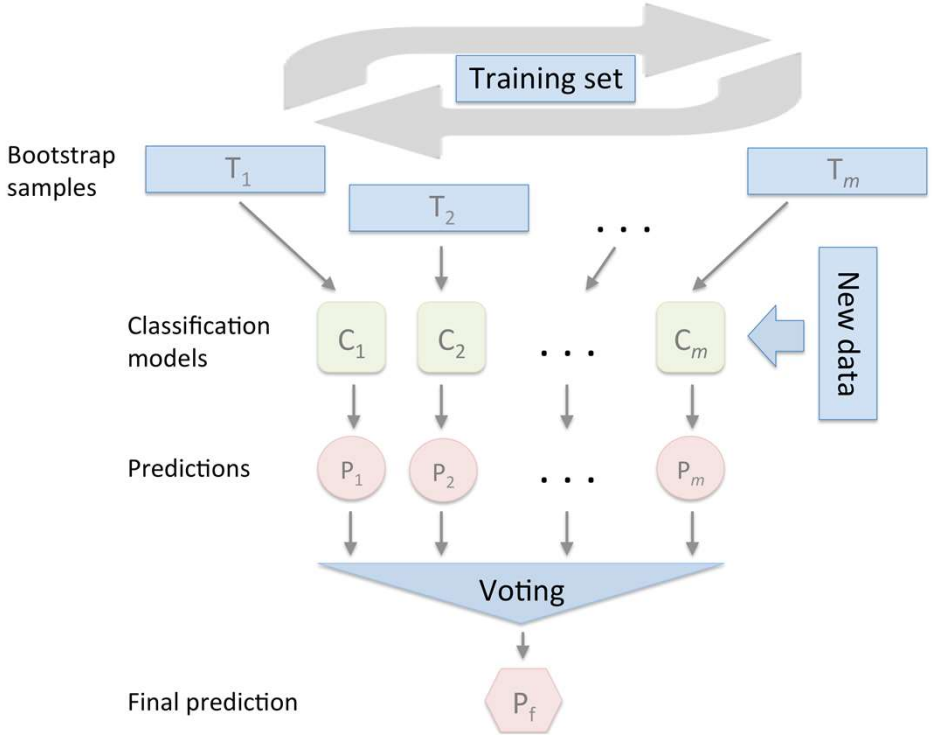
- Bagging (Bootstrap Aggregating) → 'Random Forest'
  - Bootstrap + voting (used for classification)
  - Bootstrap + Averaging (used for regression)

- Designed to improve the stability and accuracy of machine learning algorithms
- Also reduces variance and helps to avoid overfitting.

How does bagging work?

- Generate B different bootstrapped training datasets
- Train the learning method on each of the B training datasets, and obtain the prediction
- For prediction:
  - majority vote among all B models

Bagging



https://github.com/rasbt/python-machine-learning-book/tree/master/code/ch07/images

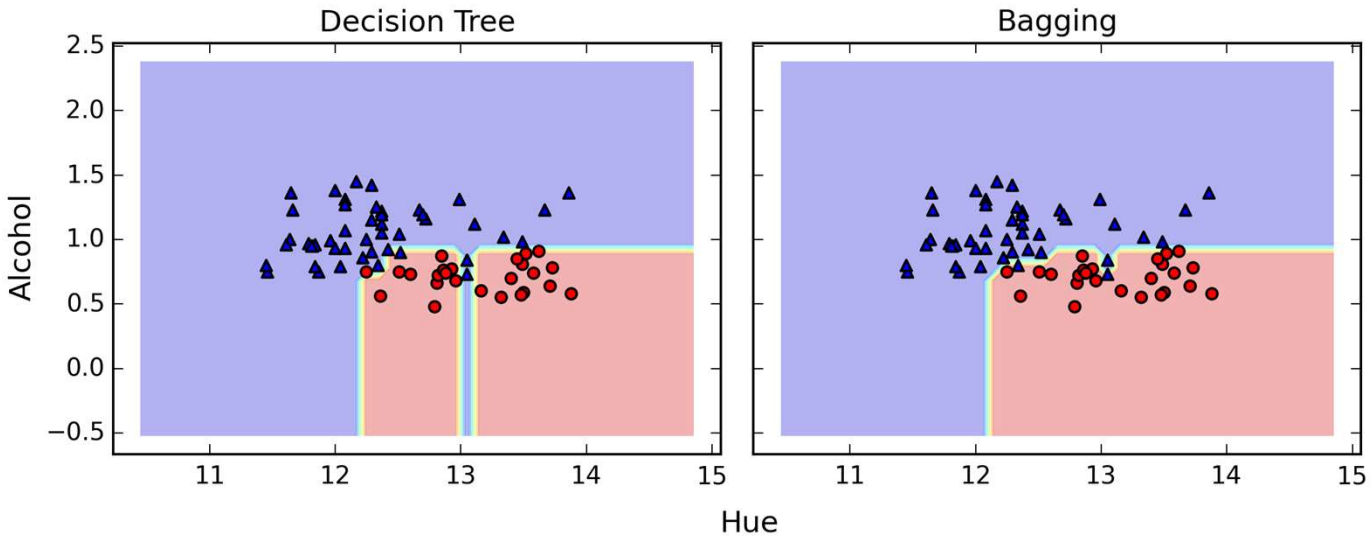
Bagging example

Sample indices	Bagging round 1	Bagging round 2	…
1	2	7	…
2	2	3	…
3	1	2	…
4	3	1	…
5	7	1	…
6	2	7	…
7	4	7	…
	C_1	C_2	C_m

- Each classifier receives a random subset of samples
- Each subset may contain duplicates
- Some samples don't appear in a resampled dataset

https://github.com/rasbt/python-machine-learning-book/tree/master/code/ch07/images

Decision boundary



Smoother decision boundary in the bagging ensemble

https://github.com/rasbt/python-machine-learning-book/tree/master/code/ch07/images

Bagging

- Bagging is an extremely powerful idea based on two things:
  - Averaging: reduces variance
  - Bootstrapping: plenty of training datasets

- Why does averaging reduces variance?
  - Averaging a set of observations reduces variance.

Random Forests (랜덤포레스트)

- Ensemble methods: by combining multiple weaker learners, a stronger learner is created

- Random forests
  - 같은 데이터로부터 의사결정나무를 여러 개 만들어 그 결과를 종합해 예측 성능을 높이는 기법 (Bagging)
  - 각 나무를 만들 때, m개의 feature들을 무작위로 선정하여 학습시킴 (보통 $m \approx \sqrt{p}$) → de-correlate trees!

Prediction for Random Forest

- Construct B classification trees using B bootstrapped training datasets
- For prediction, there are two approaches:
  1. Record the class that each bootstrapped data set predicts and provide an overall prediction to the most commonly occurring one (majority vote).
  2. If our classifier produces probability estimates we can just average the probabilities and then predict to the class with the highest probability.
- Both methods work well.

Example: combining class probabilities

- Assume the classifiers return class probabilities in a binary classification problem

$$C_1(x): [0.9, 0.1]$$
$$C_2(x): [0.8, 0.2]$$
$$C_3(x): [0.4, 0.6]$$

Prob(y_{pred} =0) =(0.9 + 0.8 + 0.4)/3 = 0.7

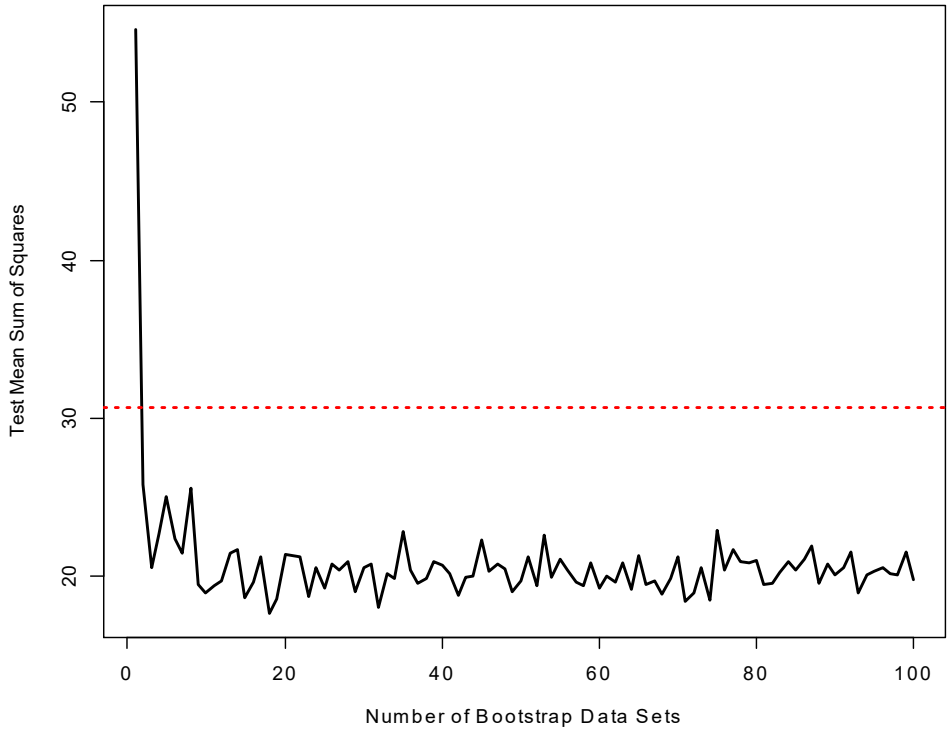
Prob(y_{pred} =1) = (0.1 + 0.2 + 0.6)/3 = 0.3

➔ y_{pred} = 0

- In contrast, in a simple majority voting
  - C1 and C2 predicts class 0, while C3 predicts class 1 ➔ mode{0,0,1} = 0

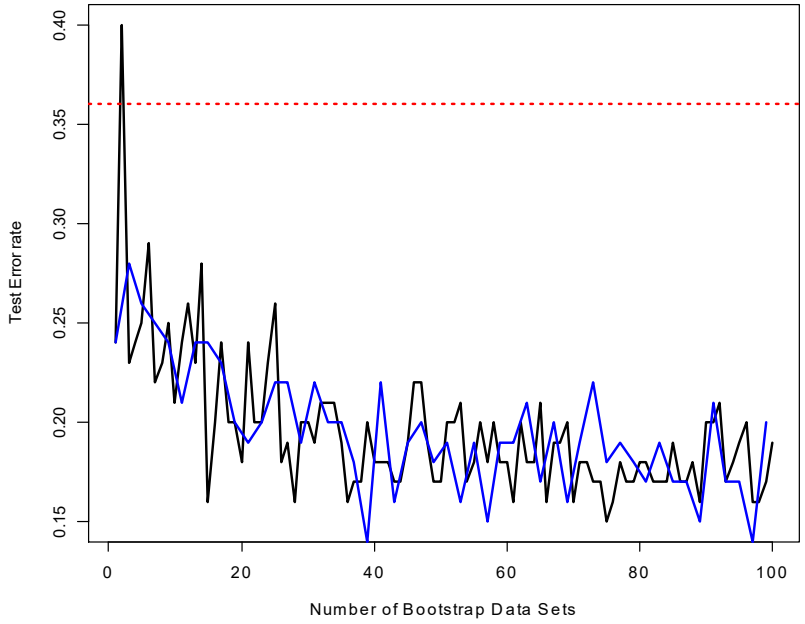
Example: single vs. bagged trees

- The red line represents the test mean sum of squares using a single tree.
- The black line corresponds to the bagging error rate



A Comparison of Error Rates

- The red line represents the test error rate using a single tree.
- Here the black line represents a simple majority vote approach
- The blue line corresponds to averaging the probability estimates.



Why are we considering a random sample of m predictors instead of all p predictors for splitting?

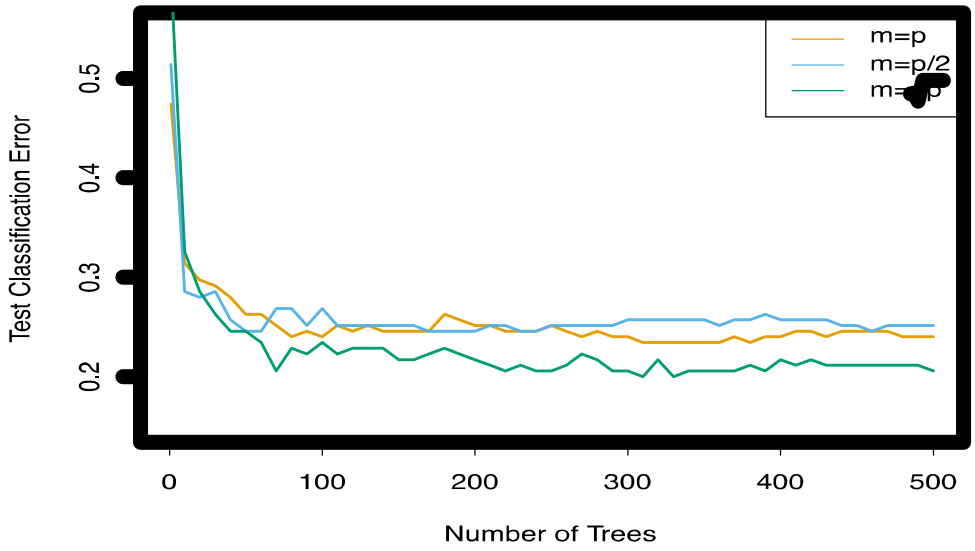
- Suppose that we have a very strong predictor in the data set along with a number of other predictors, then in the collection of bagged trees, most or all of them will use the very strong predictor for the first split

- All bagged trees will look similar. Hence all the predictions from the bagged trees will be highly correlated

- Averaging many highly correlated quantities does not lead to a large variance reduction, and thus random forests "de-correlates" the bagged trees leading to more reduction in variance

Random Forest with different values of “m”

- Notice when random forests are built using m = p, then this amounts simply to bagging.



Variable Importance Measure

- Bagging typically improves the accuracy over prediction using a single tree, but it is now hard to interpret the model
- We have hundreds of trees, and it is no longer clear which variables are most important to the procedure
- Thus bagging improves prediction accuracy at the expense of interpretability
- But, we can still get an overall summary of the importance of each predictor using Relative Influence Plots

Relative Influence Plots

- How do we decide which variables are most useful in predicting the response?
  - We can compute something called relative influence plots.
  - These plots give a score for each variable.
  - These scores represents the decrease in MSE when splitting on a particular variable
  - A number close to zero indicates the variable is not important and could be dropped.
  - The larger the score the more influence the variable has.

Example: Housing Data

- Median Income is by far the most important variable.

- Longitude, Latitude and Average occupancy are the next most important.