More on Decision Tree Learning

Gini Index

- Gini index is calculated by subtracting the sum of squared probabilities of each class from one.
- It favors larger partitions and easy to implement whereas information gain favors smaller partitions with distinct values.

$$Gini\ Index = 1 - \sum_{i=1}^{n} p_i^2$$

- Classification and Regression Tree (CART) algorithm deploys the method of the Gini Index to originate binary splits.

An Example of Gini Index

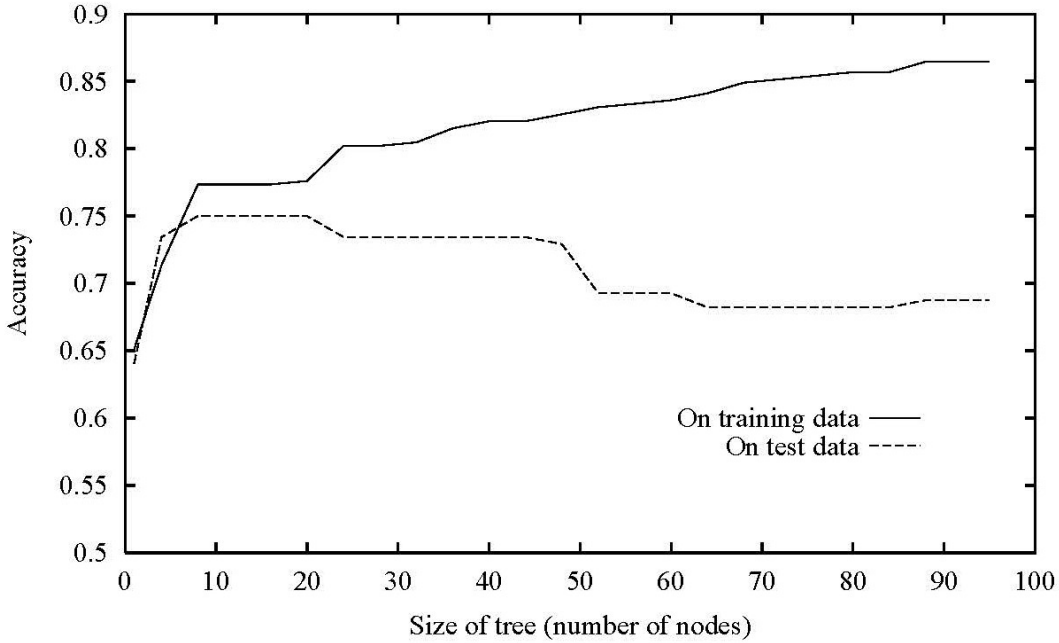
Past Trend	Open Interest	Trading Volume	Return
Positive	Low	High	Up
Negative	High	Low	Down
Positive	Low	High	Up
Positive	High	High	Up
Negative	Low	High	Down
Positive	Low	Low	Down
Negative	High	High	Down
Negative	Low	High	Down
Positive	Low	Low	Down
Positive	High	High	Up

An Example of Gini Index

- P(Past Trend=Positive): 6/10 P(Past Trend=Negative): 4/10
- If (Past Trend = Positive & Return = Up), probability = 4/6
- If (Past Trend = Positive & Return = Down), probability = 2/6
- Gini index = 1 - ((4/6)^2 + (2/6)^2) = 0.45
- If (Past Trend = Negative & Return = Up), probability = 0
- If (Past Trend = Negative & Return = Down), probability = 4/4
- Gini index = 1 - ((0)^2 + (4/4)^2) = 0
- Weighted sum of the Gini Indices can be calculated as follows:
- Gini Index for Past Trend = (6/10)0.45 + (4/10)0 = 0.27
- Similarly Gini Index for Open Interest = (4/10)0.5 + (6/10)0.45 = 0.47
- Gini Index for Trading Volume = (7/10)0.49 + (3/10)0 = 0.34
- 'Past Trend' has the lowest Gini Index and hence it will be chosen as the root node of the tree.

Overfitting in Decision Trees

- Overfitting occurs when the model fits too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably.
- Overfitting occurs when a model begins to "memorize" training data rather than "learning" to generalize from a trend.

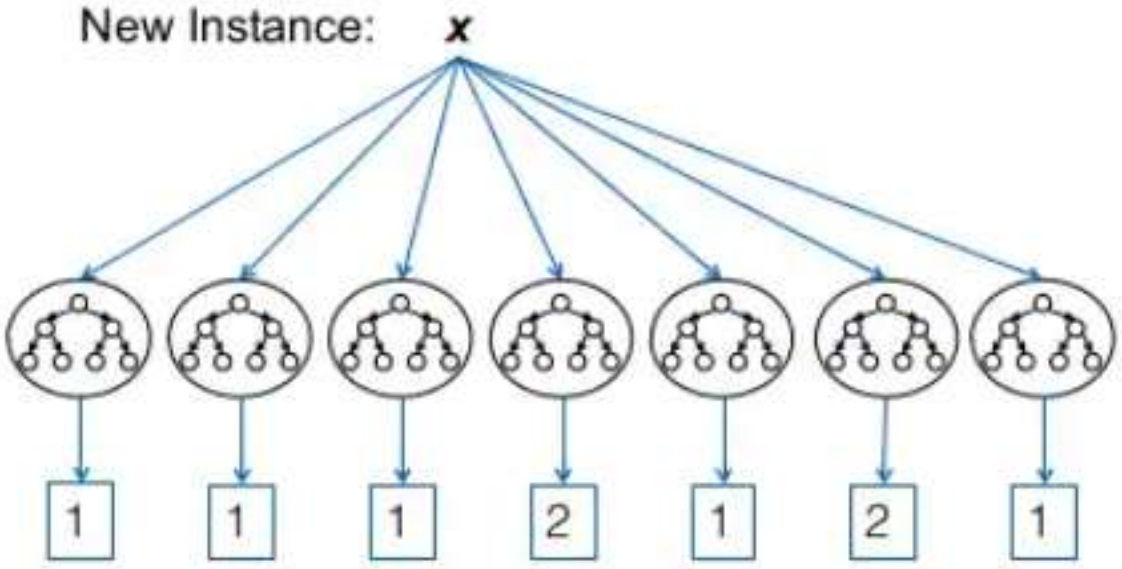


K-Fold Cross-Validation

- K-fold Cross-Validation is a method to evaluate the performance of a learned model.
- Steps:
  - Split training data into k equal parts
  - Fit the model on k-1 parts and calculate test error using the fitted model on the kth part
  - Repeat k times, using each data subset as the test set once. (usually k= 5~20)

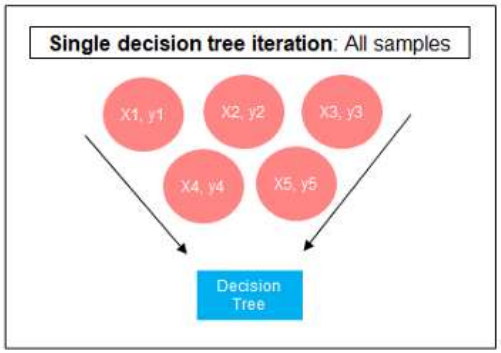
What is an Ensemble?

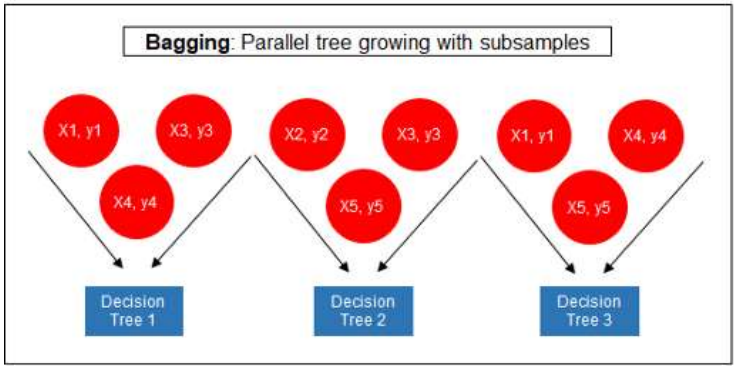
- An ensemble is a combination of classifiers/models to obtain better predictive performance.

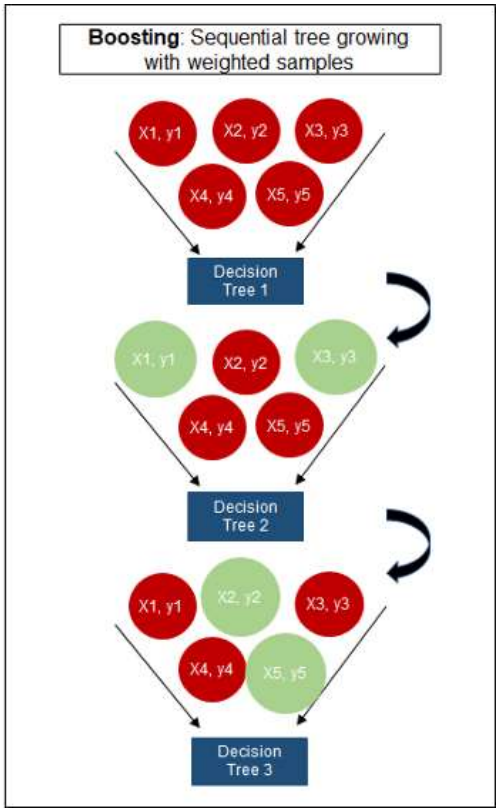


Methods to Learn Tree Ensembles

- Random Forest (Bagging)
- Gradient Tree Boosting
  - GBM
- Gradient Tree Boosting with Regularization (variant of GBM)
  - XGBoost







* Images from https://towardsdatascience.com/the-ultimate-guide-to-adaboost-random-forests-and-xgboost-7f9327061c4f

Bagging

- Weak learners are produced *parallelly* during the training phase. The performance of the model can be increased by parallelly training a number of weak learners on bootstrapped data sets.

Classifier generation:

- Let *N* be the size of the training set.
- for each of *t* iterations:
  - sample *N* instances with replacement from the original training set.
  - apply the learning algorithm to the sample.
  - store the resulting classifier.

Original training dataset: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Resampled training set 1: 2, 3, 3, 5, 6, 1, 8, 10, 9, 1

Resampled training set 2: 1, 1, 5, 6, 3, 8, 9, 10, 2, 7

Resampled training set 3: 1, 5, 8, 9, 2, 10, 9, 7, 5, 4

Classification:

- for each of the *t* classifiers:
  - predict class of instance using classifier.
- return the most popular class (or average prediction value in case of regression problems) as the final prediction.

Boosting

- The weak learners are *sequentially* produced during the training phase. The performance of the model is improved by assigning a higher weight to the previous, incorrectly classified samples.

Classifier generation:

- Initialize equal weights for all training examples
- For *T* rounds
  - Normalize the weights
  - Train a classifier and evaluate the training error
  - Update the weights of the training examples: increase if classified wrongly by this classifier, decrease if correctly

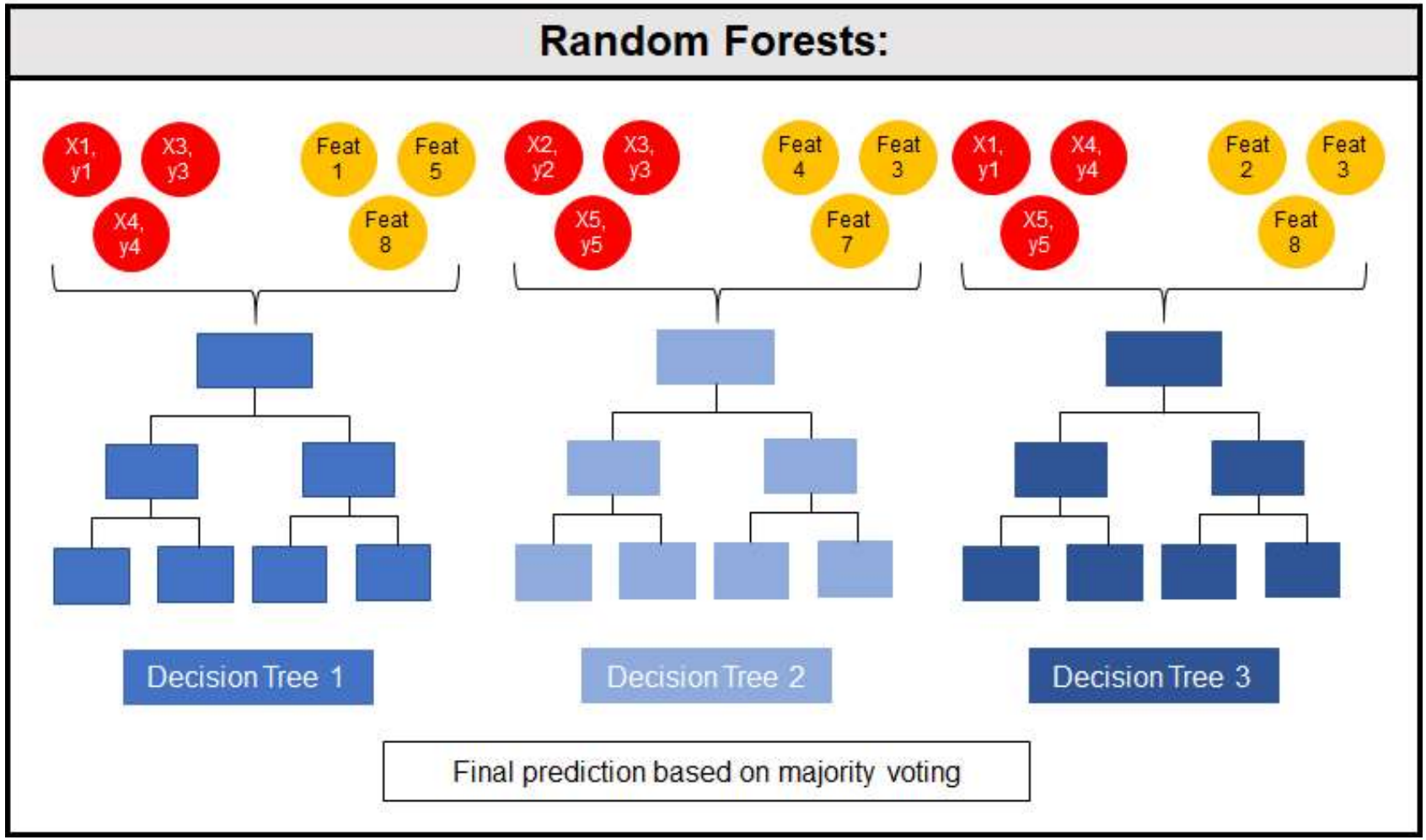
Classification:

- for each of the *t* classifiers:
  - predict class of instance using classifier.
- return the linear combination of the *T* classifiers (coefficient larger if training error is small)

Random Forests

- The random forests algorithm is based on the bagging approach.
- Random forests achieve a reduction in overfitting by combining many weak learners that underfit samples because they only utilize a subset of all training samples.
- Algorithm
  - For t in T rounds (with T being the number of trees grown):
    - Draw a random sample s with replacement from the training set
    - Repeat the following steps recursively until the tree's prediction does not further improve:
      - Randomly choose f number of features from all available features F
      - Choose the feature with the most information gain
      - This feature is used to split the current node of the tree on
  - Output: majority voting of all T trees decides on the final prediction results

Random Forests



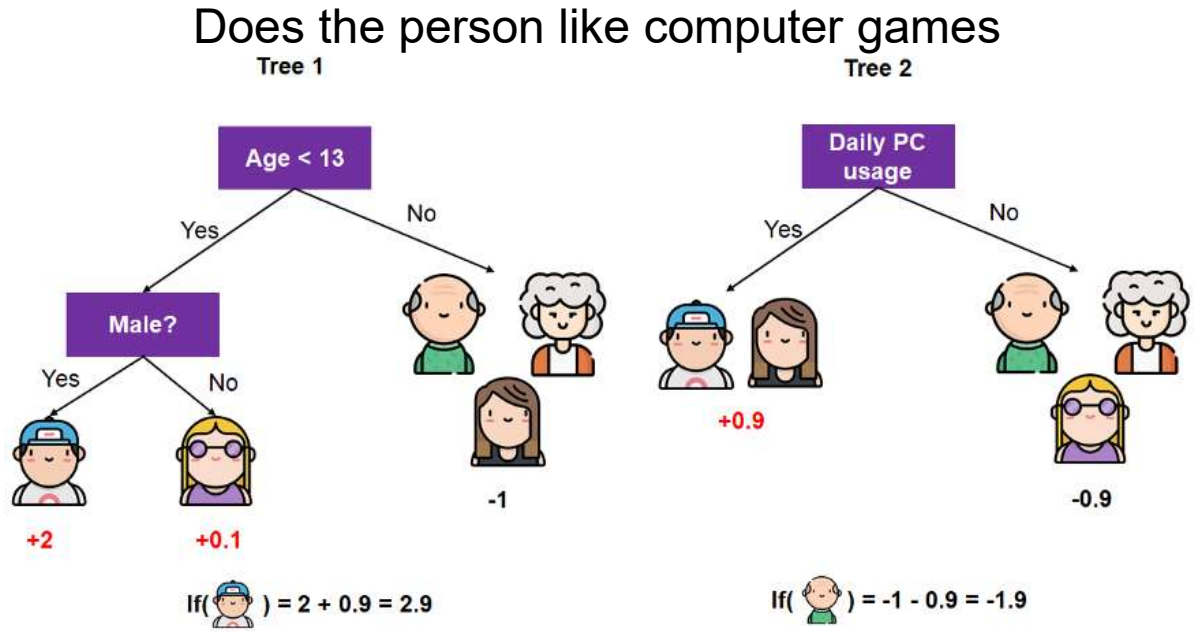
* Images from https://towardsdatascience.com/the-ultimate-guide-to-adaboost-random-forests-and-xgboost-7f9327061c4f

XGBoost (eXtreme Gradient Boosting)

- XGBoost uses the concept of gradient tree boosting, and was developed to increase speed and performance, while introducing regularization parameters to reduce overfitting.
- It uses Classification And Regression Trees (CART) in a sequential learning process as weak learners. These regression trees are similar to decision trees, however, they use a continuous score assigned to each leaf (i.e. the last node once the tree has finished growing) which is summed up and provides the final prediction.
- For each iteration i which grows a tree t, scores w are calculated which predict a certain outcome y. The learning process aims to minimize the overall score which is composed of the loss function at i-1 and the new tree structure of t.

XGBoost (eXtreme Gradient Boosting)

- The algorithm sequentially grows the trees and learns from previous iterations. Gradient descent is then used to compute the optimal values for each leaf and the overall score of tree t. The score is also called the impurity of the predictions of a tree.



Visualization of tree ensemble model using a continuous score to provide final prediction. Source: Julia Nikulski based on Chen & Guestrin (2016); icons made by Freepik from Flaticon.