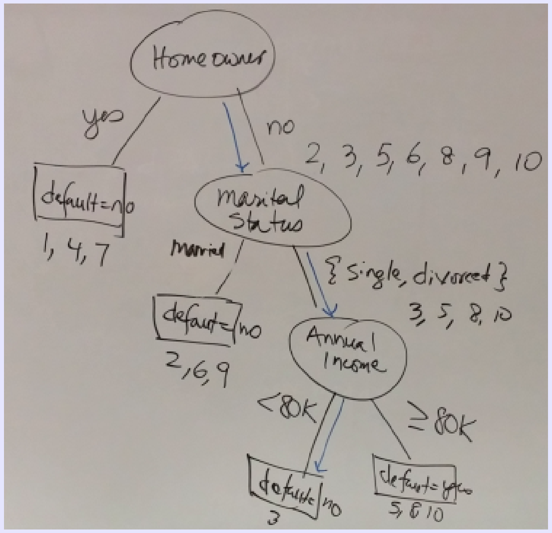
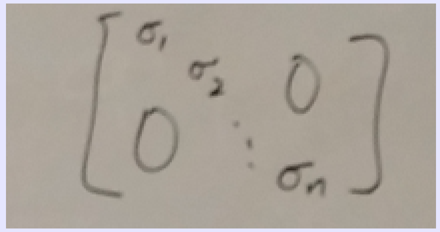
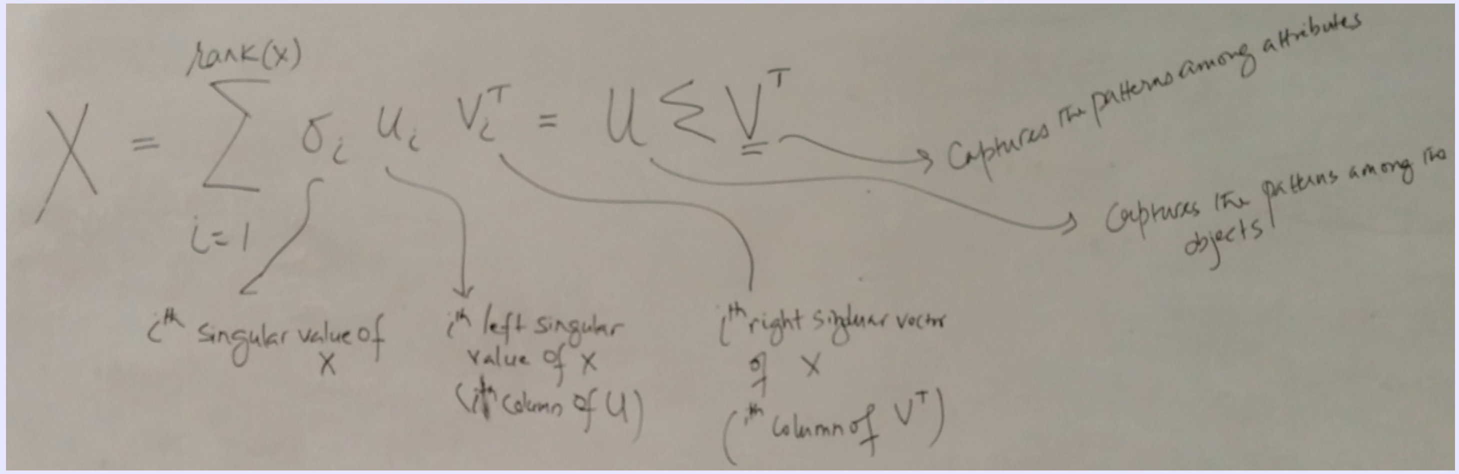
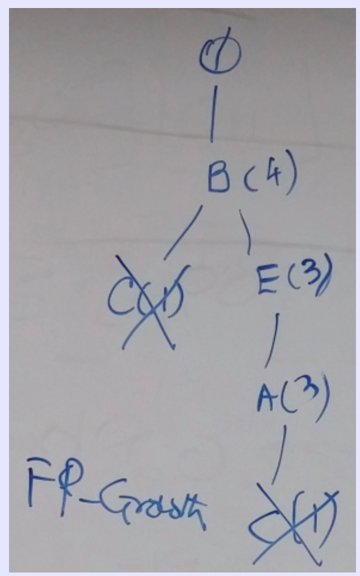
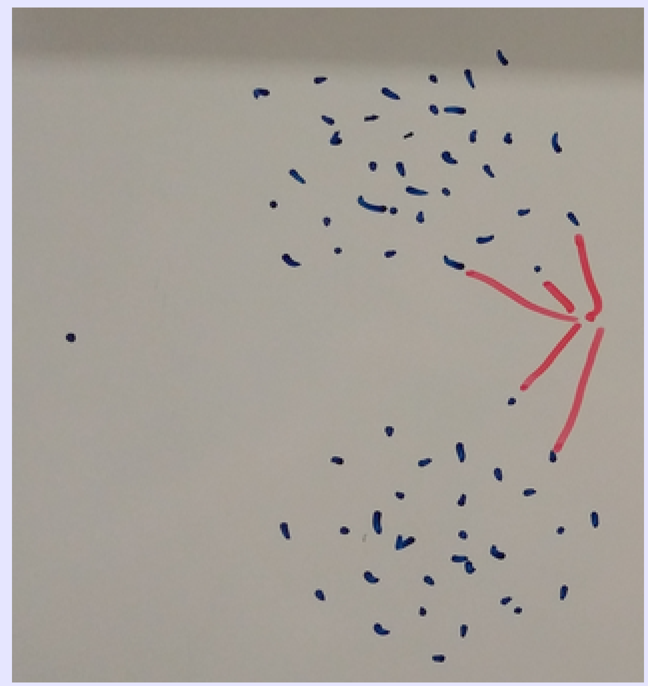










CS 422: Data Mining

Vijay K. Gurbani, Ph.D.,
Illinois Institute of Technology

Lecture 5: **Decision Trees (continued), Interpretation and evaluation of Decision Trees, Advanced Decision Trees**

Performance estimation

- Receiver Operation Characteristics (ROC) Curve
  - Developed in 1950s for signal detection theory to analyze noisy signals
    - Characterize the trade-off between positive hits and false alarms
- Performance of classifier represented as a point on the curve for some threshold.
- Let T: Value of a diagnostic test, and D: indicator variable for the presence of a disease, and c a threshold.
- The ROC curve plots the TPR as $\Pr(T>c|D = 1)$ against the FPR as $\Pr(T>c|D = 0)$ for all values of c (a threshold).
  - c is usually taken to be sort(unique(T)).

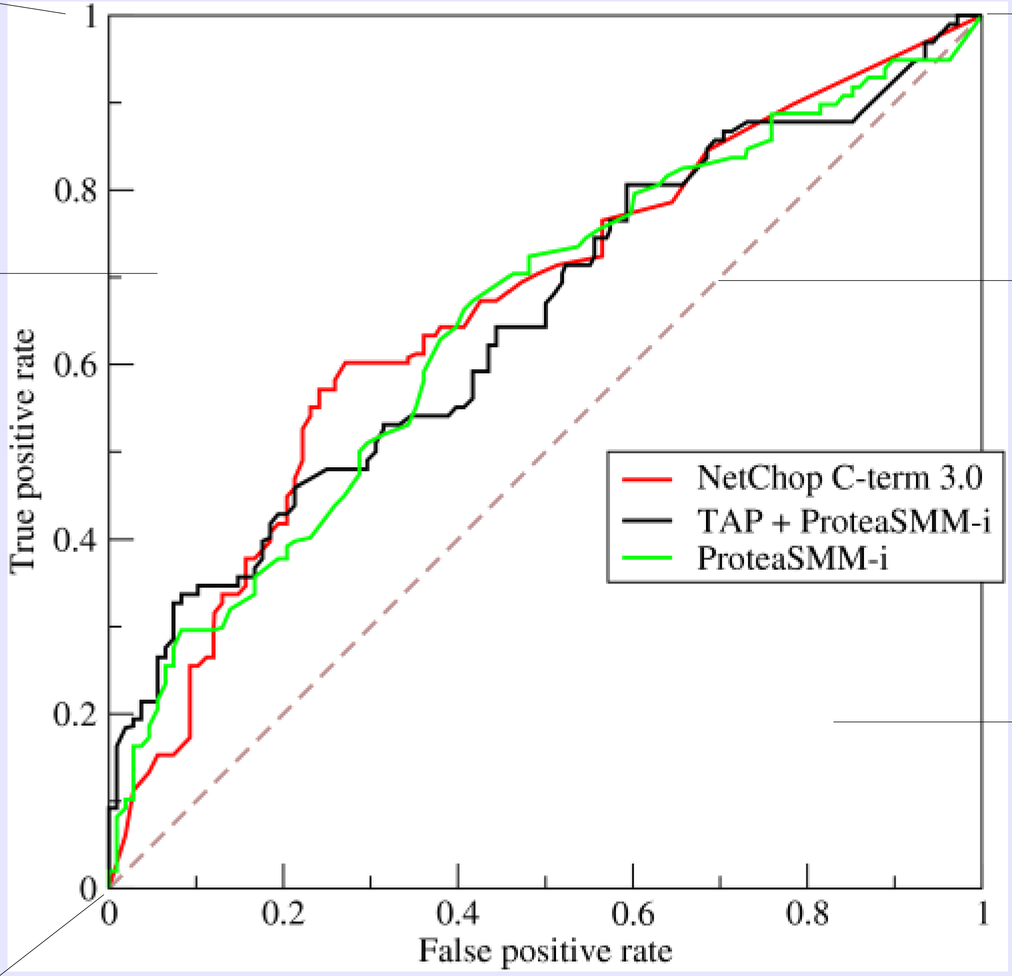
Performance estimation

(TPR=1,FPR=0): Ideal model

(TPR=1, FPR=1): model predicts every instance as positive class

Good area to be in for the model!

Random line (model should do better than this)



Bad area to be in for the model!

Which model is better? AUC.

Graphic source: Wikipedia

(TPR=0, FPR=0): model predicts every instance as negative class

Performance estimation

How to calculate a ROC curve? (1)

- Let *T*: Result of a diagnostic test, and *D*: indicator variable for the presence of a disease.
  - D is your true label.
  - T is the probability associated with a prediction of the observations.
- The ROC curve plots the TPR against the FPR (1-specificity) ∀*c* ∈ C (a threshold).
- *C* is usually taken to be sort(unique(T)).
- **Example**: Given some results:

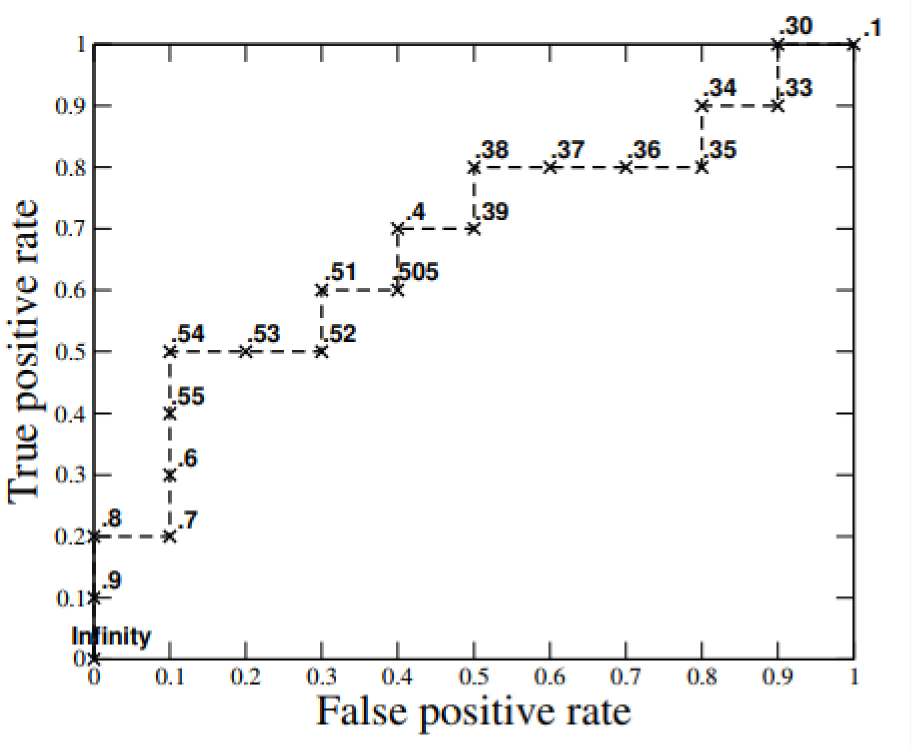
```
> head(df)
  label  prob
1     0 0.161
2     0 0.161
3     0 0.857
4     0 0.857
5     1 0.161
6     1 0.857
```

- Compute sensitivity as a conditional probability:
  Pr(df$prob > c | df$label = 1) => sensitivity
- Compute specificity as a conditional probability:
  Pr(df$prob <= c | df$label = 0) => specificity
- plot(1-specificity, sensitivity) ∀ c ∈{sort(unique(df$prob))}

Performance estimation

How to calculate a ROC curve? (2)

Inst#	Class	Score	Inst#	Class	Score
1	p	.9	11	p	.4
2	p	.8	12	n	.39
3	n	.7	13	p	.38
4	p	.6	14	n	.37
5	p	.55	15	n	.36
6	p	.54	16	n	.35
7	n	.53	17	p	.34
8	n	.52	18	n	.33
9	p	.51	19	p	.30
10	n	.505	20	n	.1



Source: Tom Fawcett, “ROC Graphs: Notes and Practical Considerations for Researchers (2004)”
Online at http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.9777

Selecting a final model

- Recap: What you have done so far is evaluated models through K-fold cross validation and estimated their performance using confusion matrices and RoCs.
- Next step: Choosing the final model.
- The model chosen is the model that performs **best**.
- How do we define *best*? (The best model is also called the *final* model.)
- The best model is the one that gives you the smallest prediction error (or minimizes the loss function) on the training set and generalizes well on the testing set.

Closing words on model selection

- (1) Overfitting and underfitting in decision tree models.

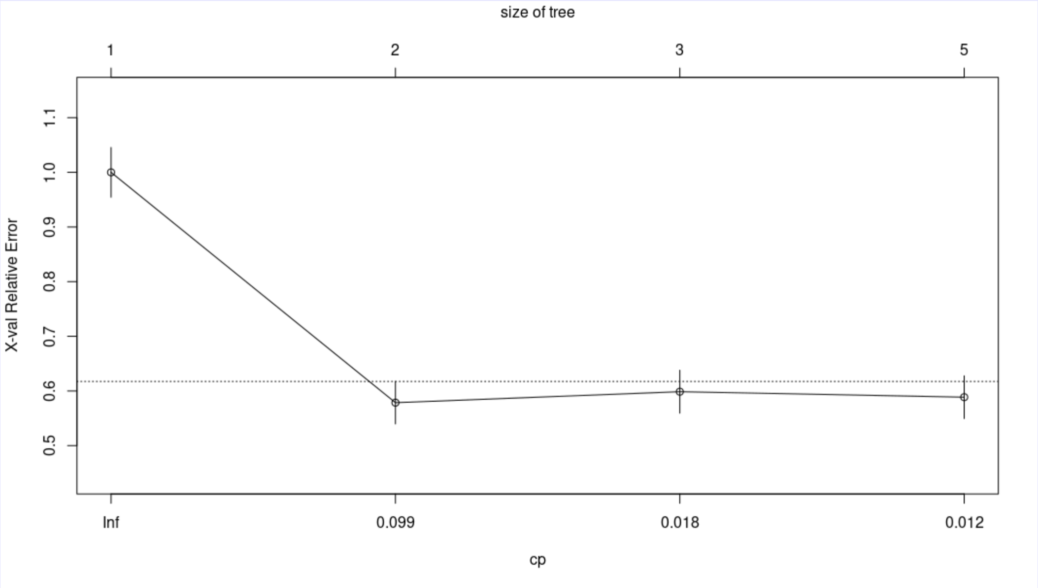
- If we allow the tree to grow (become deep), we run the risk of overfitting.
  - Overfitting can be mitigated by **pruning** the tree: Grow the tree to its entirety, then trim nodes in a bottom-up fashion. If generalization error improves, replace sub-tree by a leaf node. Class label of the leaf node is determined by majority class of the instances in the sub-tree.
- If we stop early, we may underfit (error on training data may be low).
- Strategies on when to stop splitting:
  - When the best candidate split at a node reduces the impurity by less than a threshold.
    - How to set this threshold?
      - Stop when node has a certain number of observations.
  - When all observations in a node belong to the same class.
- Tradeoff between tree complexity vs. test set accuracy.

- Pruning: Two approaches:
  - Prepruning: Halt growth of tree based on some constraint (e.g., gain in impurity < threshold).
    - + : Shorter trees.
    - - : When to stop?
  - Post-pruning: Grow tree to maximum size, then trim (e.g., replace subtree with new leaf node whose class label is determined from majority class of records affiliated with the subtree.)
    - + : Gives better results than prepruning since we have benefit of the fully grown tree.
    - - : Wasted compute cycles in constructing the subtree if we have to eventually prune it.

Closing words on model selection

- (1) Overfitting and underfitting in decision tree models.
- To prune: Focus on the complexity parameter (cp) corresponding to error and xerror. These two act as multiple multiple R^2 and adjusted R^2 in regression.
  - The cp parameter is defined in rpart as the threshold value for the split such that any split that does not decrease the overall lack of fit by a factor of cp is not attempted.
  - Any split which does not improve the fit by cp will likely be pruned off by cross-validation, and that hence the program need not pursue it
- Choose cp value with lowest xerror and prune the tree by: prune(model, cp=<chosen cp value>)

```
> printcp(model)

Classification tree:
rpart(formula = survived ~ pclass + sex + age, data = train,
    method = "class")

Variables actually used in tree construction:
[1] age    pclass sex

Root node error: 299/786 = 0.38

n= 786

      CP nsplit rel error xerror   xstd
1 0.4214      0     1.000  1.000 0.0455
2 0.0234      1     0.579  0.579 0.0388
3 0.0134      2     0.555  0.599 0.0393
4 0.0100      4     0.528  0.589 0.0391

> plotcp(model)
```

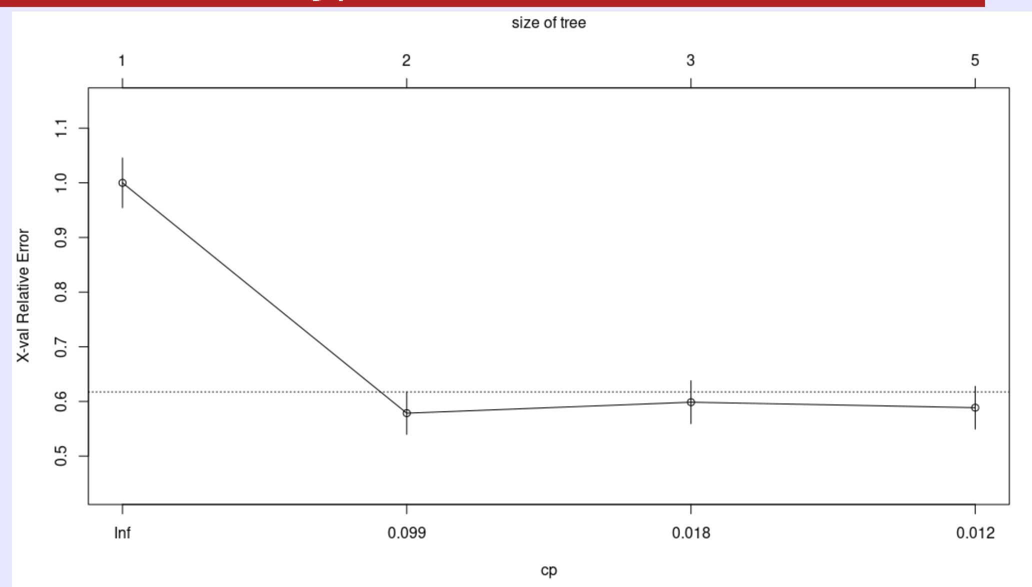


Closing words on model selection

- (1) Overfitting and underfitting in decision tree models.
- To prune: Focus on the complexity parameter (cp) corresponding to error and xerror. These two act as multiple multiple R^2 and adjusted R^2 in regression.
  - split
  - on,
- Ch alue>)

```
# Pruning the tree
printcp(model)
plotcp(model)
cpx <- model$cptable[which.min(model$cptable[,"xerror"]), "CP"]
pruned.model <- prune(model, cp=cpx)

# Run predictions on the pruned model
pred <- predict(pruned.model, test, type="class")
```

```
> printcp(model)

Classification tree:
rpart(formula = survived ~ pclass + sex + age, data = train,
    method = "class")

Variables actually used in tree construction:
[1] age    pclass sex

Root node error: 299/786 = 0.38

n= 786

      CP nsplit rel error xerror    xstd
1 0.4214      0     1.000  1.000 0.0455
2 0.0234      1     0.579  0.579 0.0388
3 0.0134      2     0.555  0.599 0.0393
4 0.0100      4     0.528  0.589 0.0391

> plotcp(model)
```



Closing words on model selection

- (2) Surrogate variables

```
Node number 1: 786 observations,    complexity param=0.4214047
  predicted class=0  expected loss=0.3804071  P(node) =1
    class counts:   487   299
   probabilities: 0.620 0.380
  left son=2 (518 obs) right son=3 (268 obs)
  Primary splits:
      sex    splits as  RL,           improve=102.305800, (0 missing)
      pclass < 1.5      to the right, improve= 30.798720, (0 missing)
      age    < 9.5      to the right, improve=  6.130452, (0 missing)
  Surrogate splits:
      age < 5.5     to the right, agree=0.662, adj=0.007, (0 split)
```

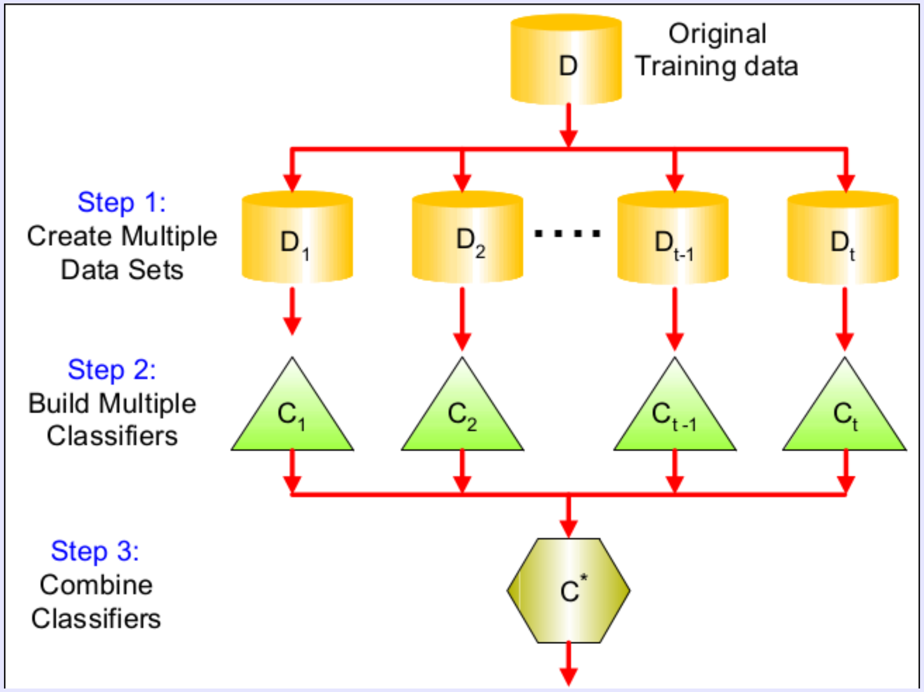
- (3) Variable importance

```
rpart(formula = survived ~ pclass + sex + age, data = train,
    method = "class")
  n= 786

        CP nsplit rel error  xerror     xstd
1 0.421405      0   1.00000 1.00000 0.045522
2 0.023411      1   0.57860 0.57860 0.038848
3 0.013378      2   0.55518 0.59866 0.039322
4 0.010000      4   0.52843 0.58863 0.039088

Variable importance
   sex pclass    age
    69     17     14
```

Sums to 100, but the most important variable may not always be the first split in the tree

Ensemble methods

- So far, we induce one classifier from training data.
- But, is *wisdom of the crowds* better?
- What if we created multiple classifiers and combined their prediction.
  - Will we get better results?



Ensemble methods

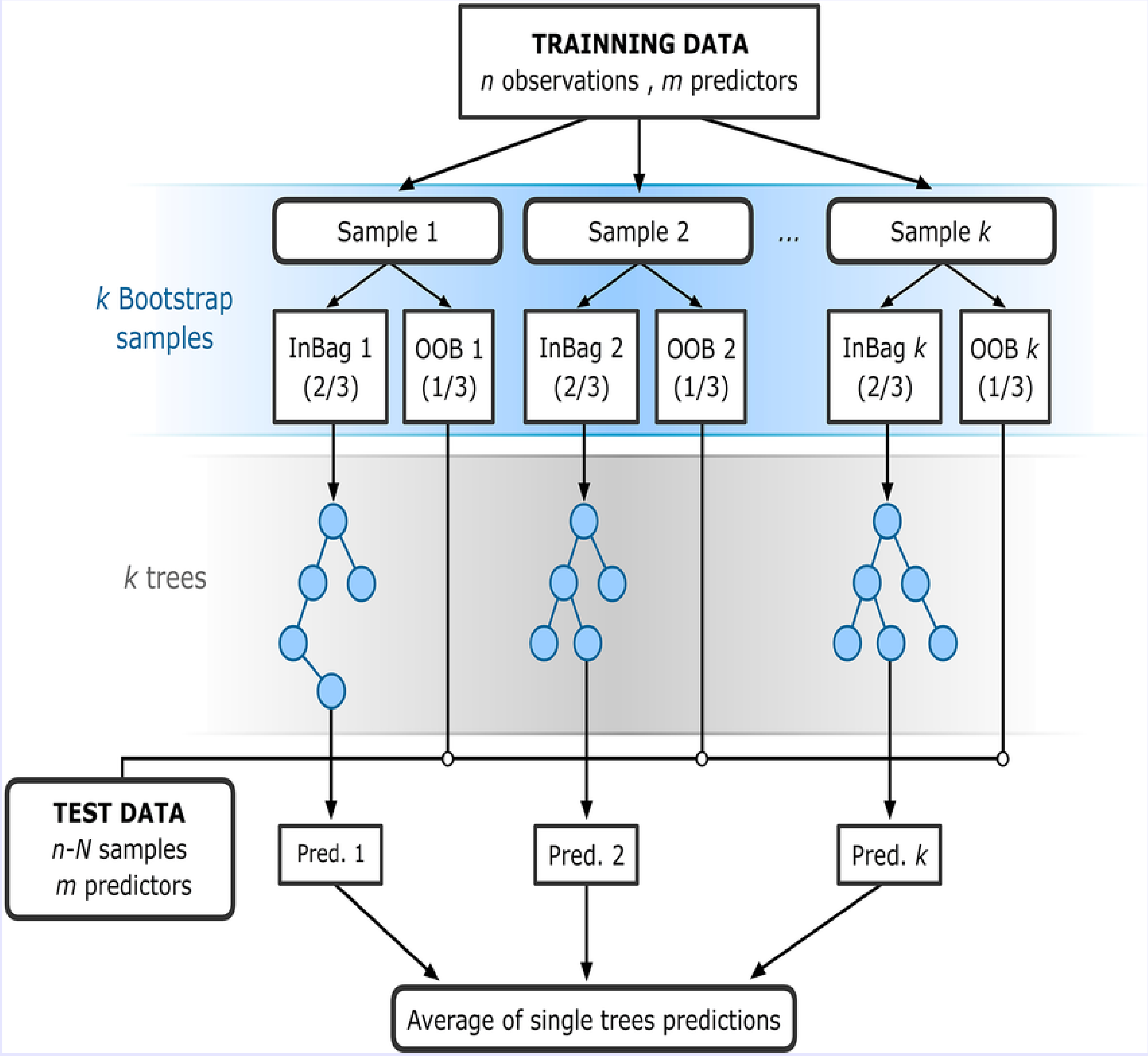
- “Wisdom of the crowds” => construct a set of base classifiers from training data and predict using a *combination* function.
  - Voting
  - Logistic regression
  - ...
- The ensemble can be created in multiple ways.
  - Manipulate the training dataset.
  - Manipulate input features.
  - Manipulate class labels.
  - Manipulate learning algorithm.
  - Use different learning algorithms (MCS).

Ensemble methods

- Can work effectively well (for some datasets).
  - Suppose we have 25 base classifiers, each of which has an error rate of $\varepsilon = 0.35$.
  - If base classifiers are identical, all commit the same mistake and error rate of ensemble remains 0.35.
  - On the other hand, if all classifiers are independent (their errors are not correlated), then ...
  - … Ensemble makes a wrong prediction only if > ½ of the base classifiers predict incorrectly, i.e. :

$$P(X \geq 13) = \sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

Ensemble methods: Random Forest



Modelling interannual variation in the spring and autumn land surface phenology of the European forest, Rodriguez-Galiano et al., 2016, Biogeosciences.

Ensemble methods: Random Forest

- Final word of wisdom: Ensemble methods are not universally better than their normal counterparts.
- For the ensemble method to be better, the individual classifiers should demonstrate some *instability* (their predictions should be independent).
  - Define *instability* as inappropriate sensitivity to input

Code: german-credit.Rmd

Class Imbalance

- Datasets with **balanced** class distribution are the exception as most (all) datasets have a imbalanced class distribution.
  - Medical
  - Manufacturing
- Correct classification of the rare (minority) class has greater value than correct classification of the majority class.
- Imbalanced classes present a number of problems to classification algorithms:
  - Accuracy is no longer a reasonable measurement.
  - Balanced accuracy is a better measure when the test (or training) datasets exhibit class imbalance.

Class Imbalance

- Other mitigation techniques:
  - Cost sensitive learning penalizes the model when it commits a false negative error.
  - Sampling techniques modify the class distribution such that the rare class is well represented in the training set.
    - Undersampling gathers **less** of the majority class observations for training.
      - Disadvantage: useful observations may not be part of the sample. (Can be overcome by sampling multiple times and using an ensemble method).
    - Oversampling gathers **more** of the minority class observations for training.
      - Disadvantage: If training data is noisy, oversampling may amplify the noise.
    - Hybrid approach uses both of the above techniques to arrive at a equivalent dataset.
  - Synthetic data may be generated, if possible. If so, the generation could ensure that the class distribution is equivalent.

Multi-class decision trees

- Classification extends to differentiating between multiple classes as well.
  - Code: multiclass.Rmd
- Evaluate multi-class regression models through overall accuracy, and per-class precision and recall.
- To create confusion matrices for each class in a multi-class classification, use a "one-vs-many" strategy.
  - Create per-class confusion matrices using the data from the overall multi-class confusion matrix (see next slide).
- To plot ROC curves, use "one-vs-many" strategy to plot different classes on the same ROC plot.
  - See the "add" parameter to ROCR::plot().

Multi-class decision trees

Predicted	Actual			
		Setosa	Versicolor	Virginica
	Setosa	10	0	0
	Versicolor	0	10	1
	Virginica	0	0	9

For class Setosa:

Predicted	Actual		
		Setosa	{Versicolor,Virginica}
	Setosa	10	
	{Versicolor,Virginica}		