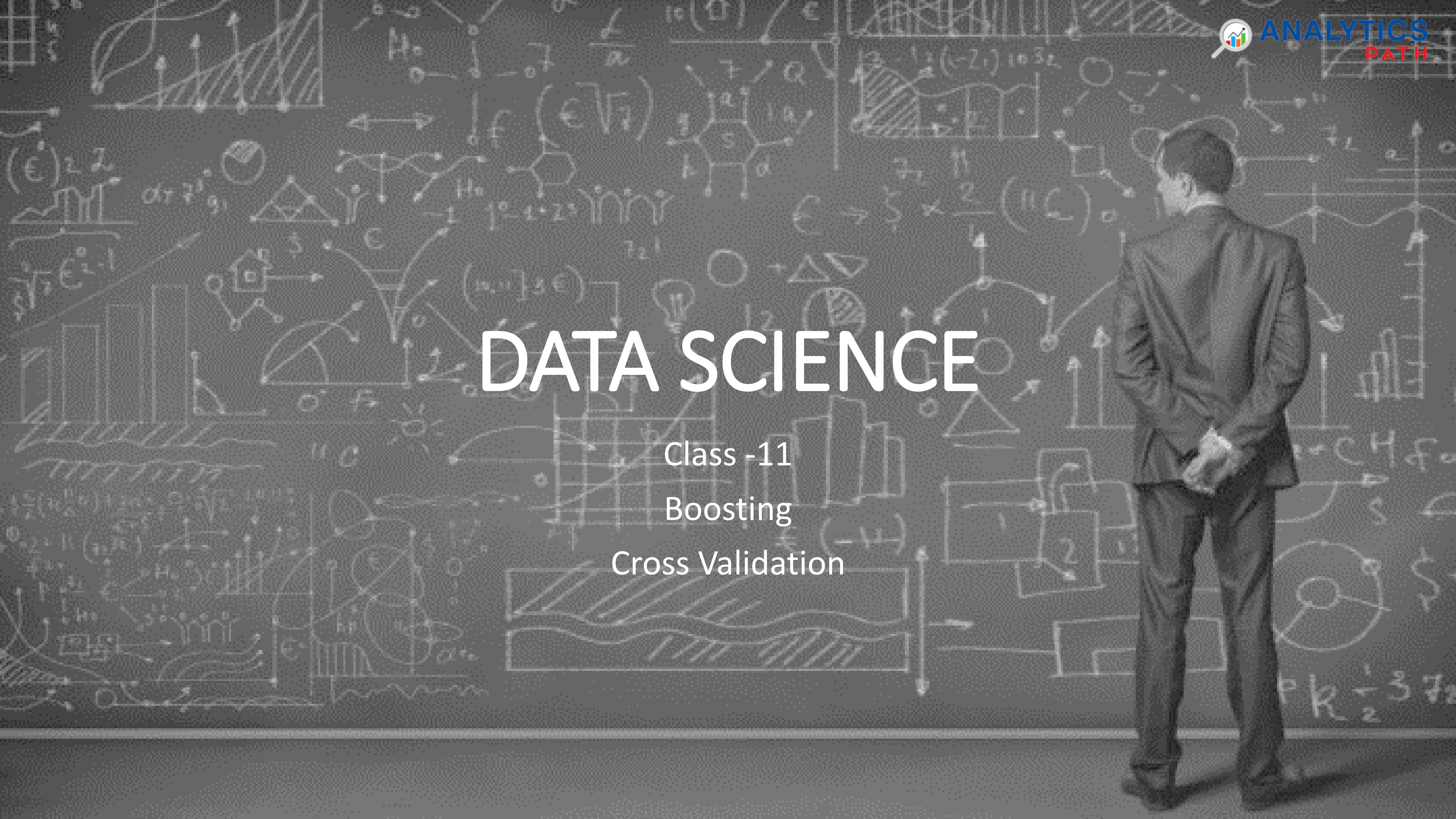
ANALYTICS PATH



DATA SCIENCE

Class -11

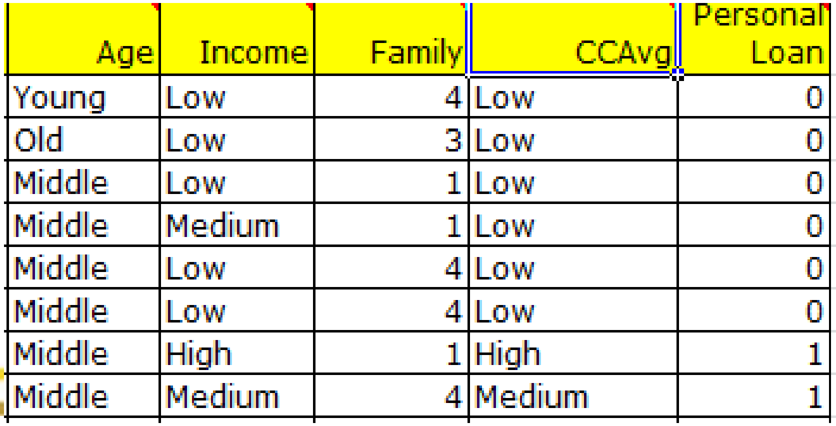
Boosting

Cross Validation

TEST – II

(17M)

- Specify any three differences between Linear Regression and Logistic Regression? (3 M)
- What is the difference between correlation and covariance? (1M)
- How is Linear SVM different from Logistic Regression? (2M)
- Cost complexity pruning and Pessimistic pruning are associated to which algorithms?(2M)
- Briefly explain the process of making a split in a decision tree. (3M)
- What affect does the learning rate (alpha) bear on the Linear Models?(2M)
- Compute the support confidence and Lift of the following rule: (3M)
  - If Age is middle and Family is 1 then Personal Loan is 1
- How do we deal with a bias condition while modeling?(1M)



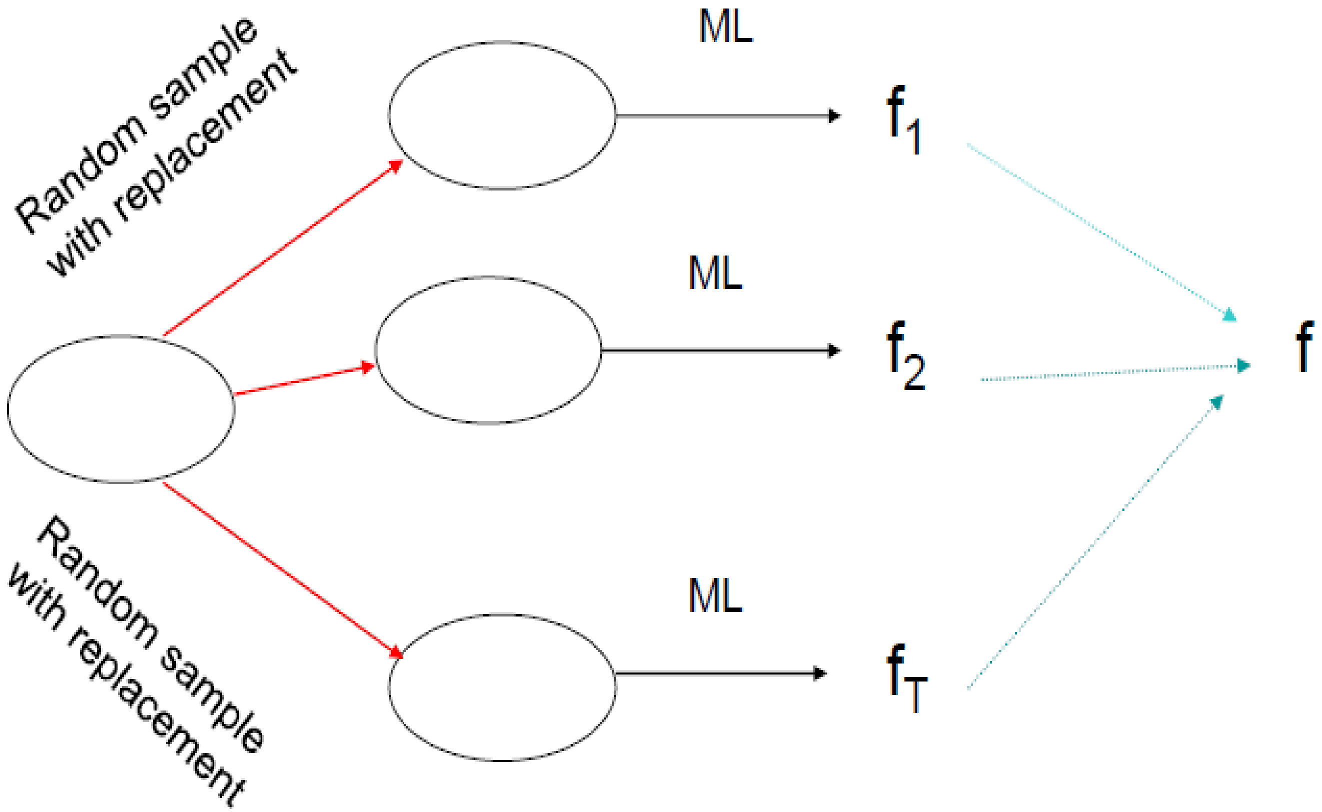
Age	Income	Family	CCAvg	Personal Loan
Young	Low	4	Low	0
Old	Low	3	Low	0
Middle	Low	1	Low	0
Middle	Medium	1	Low	0
Middle	Low	4	Low	0
Middle	Low	4	Low	0
Middle	High	1	High	1
Middle	Medium	4	Medium	1

Ensembles

- Combine multiple models (weak learners) to produce strong learners

- Advantages
  - Leverage the strengths of different kind of models
  - Reduce overfitting
  - Very small or Very Large data can be handled well
- Disadvantage
  - Speed and Explicability

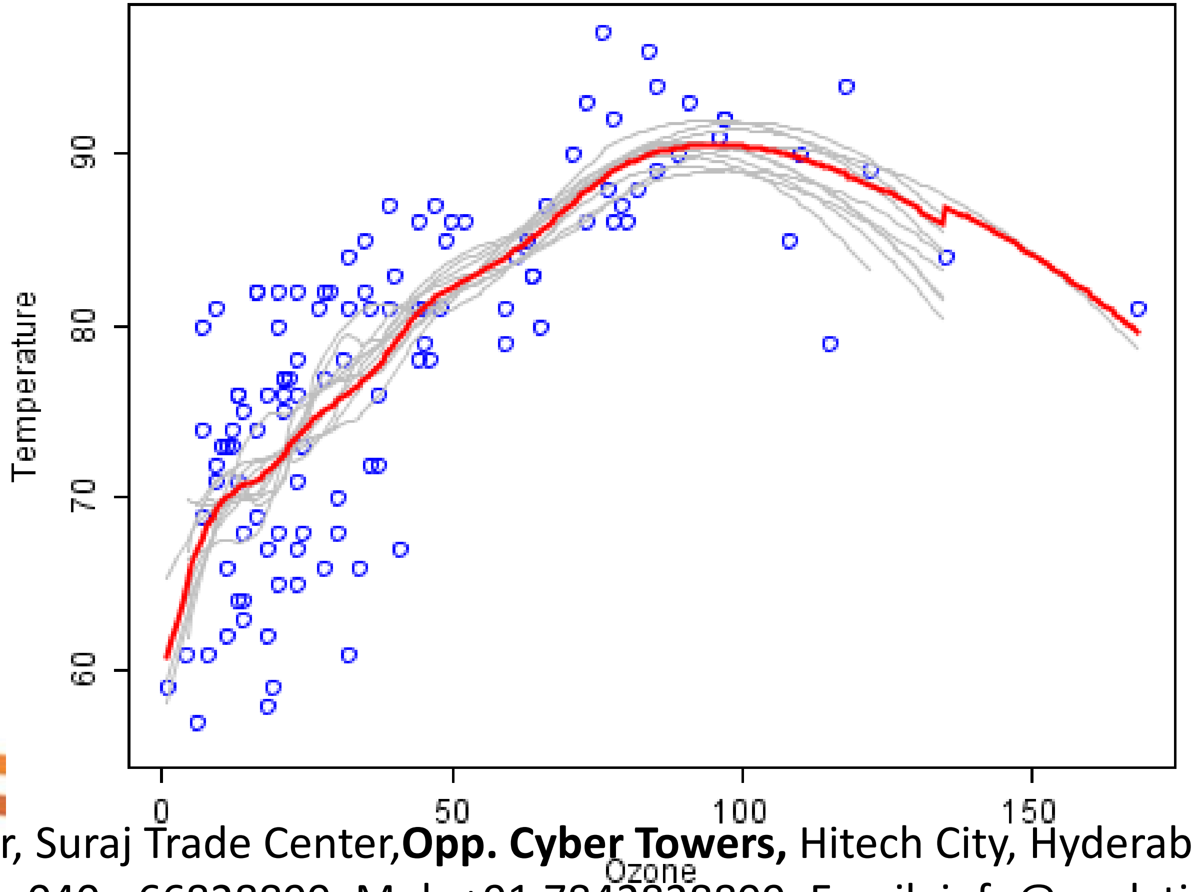
Bagging


Random sample with replacement
Random sample with replacement
ML
ML
ML
f_1
f_2
f_T
f

Bagging (Bootstrap Aggregating)

- Bootstrap the samples and create multiple sets
  - Pick randomly with replacement
- Run a base learner and generate one weak classifier for each set
- Vote on test samples

Bagging - Example

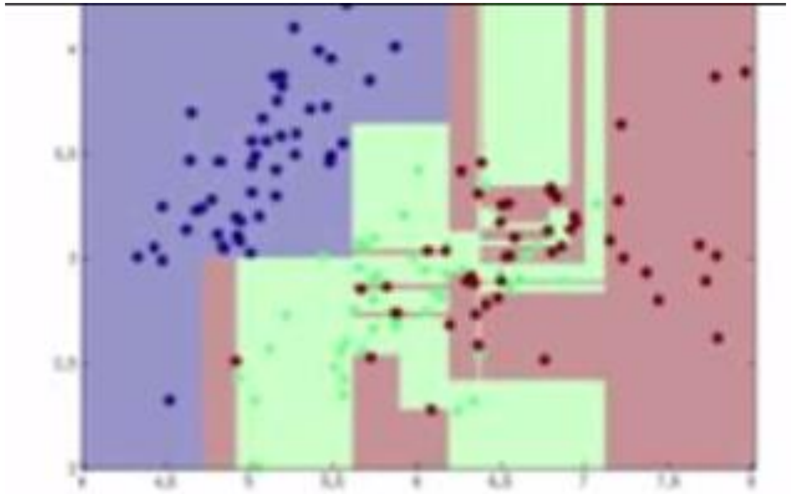


Bagging Contd…

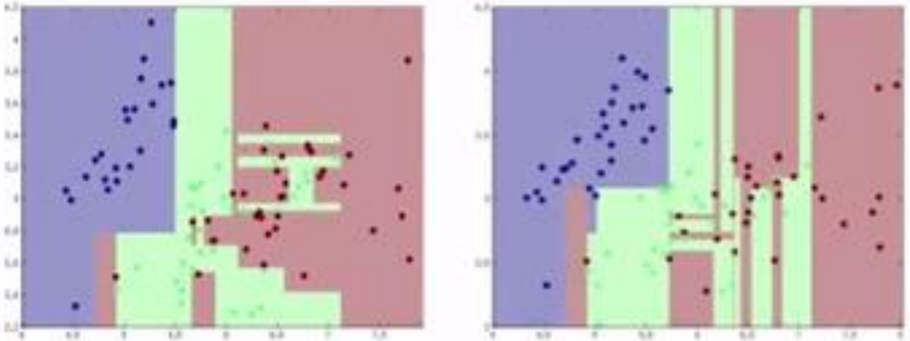
- Bagging is good for unstable learners as it reduces variance and overfitting
  - How do I generate a large number of unstable learners
    - Choose records randomly
- Rules, decision trees
  - should work wonders

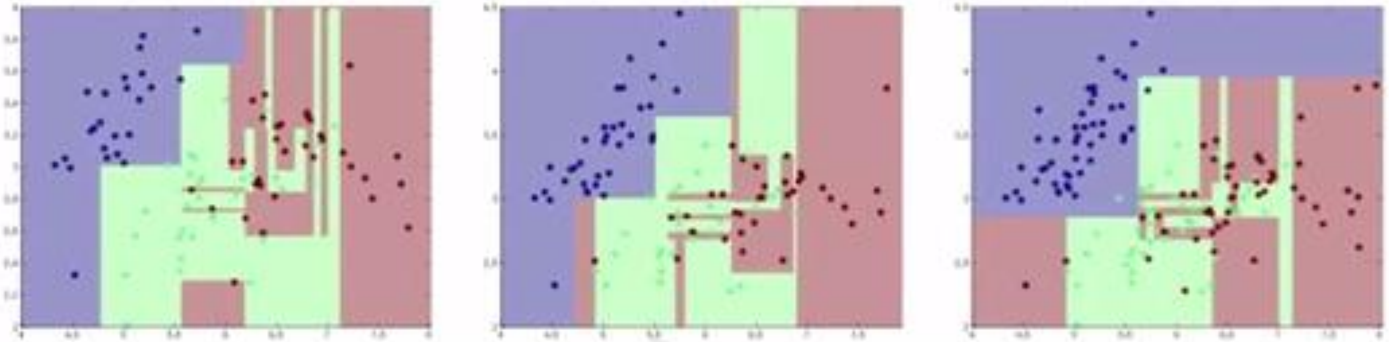
Bagging

Simulates "equally likely" data sets we could have observed instead, & their classifiers

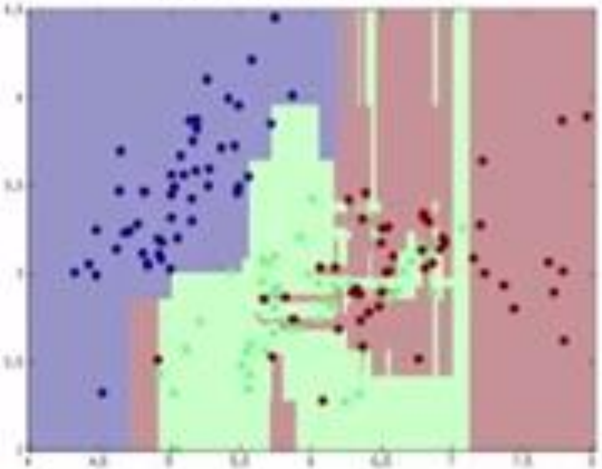


Full data set

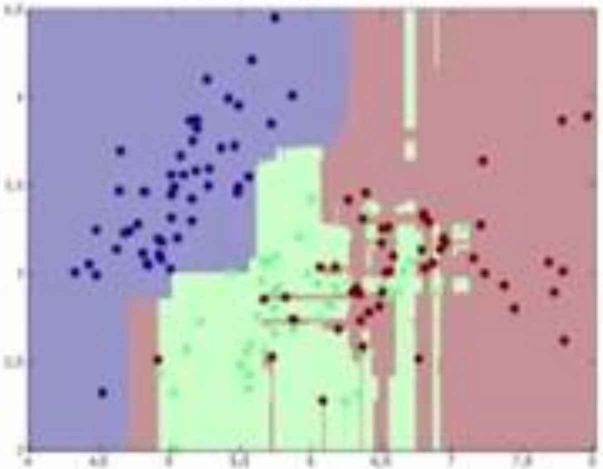




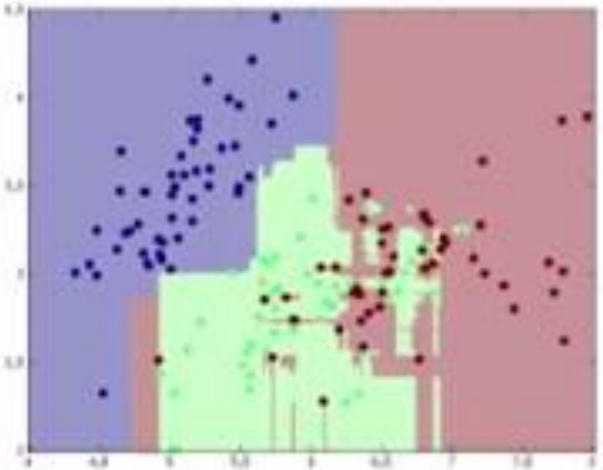
Avg of 5 trees



Avg of 25 trees



Avg of 100 trees



A lot of data??

- With a lot of data, we usually learn the same classifier most of the times. So how do we handle this?
- How do we model when we have a lot of dimensions?

Random Forest

A variant of the Bagging concept

Random Forest

- Select a large number of data sets through bagging (size is same in all)
- Use m input variables at each node of each tree. m should be much less than M (total attributes).
- Each tree is fully grown and not pruned
- Mode (for classification) or average for regression of all the trees is used as prediction.

Random Forests

- One of the best Machine Learning Algorithms
- Regression and Classification problems can be solved
- Right number of trees and right number of attributes to be used are to be selected

Random Forest - Imputation

TRAIN SET

- If x(m,n) is a missing continuous value, estimate its fill as an average over the non-missing values of the mth variables weighted by the proximities between the nth case and the non-missing value case.
- If it is a missing categorical variable, replace it by the most frequent non-missing value where frequency is weighted by proximity.
- Now iterate-construct a forest again using these newly filled in values, find new fills and iterate again. Our experience is that 4-6 iterations are enough.

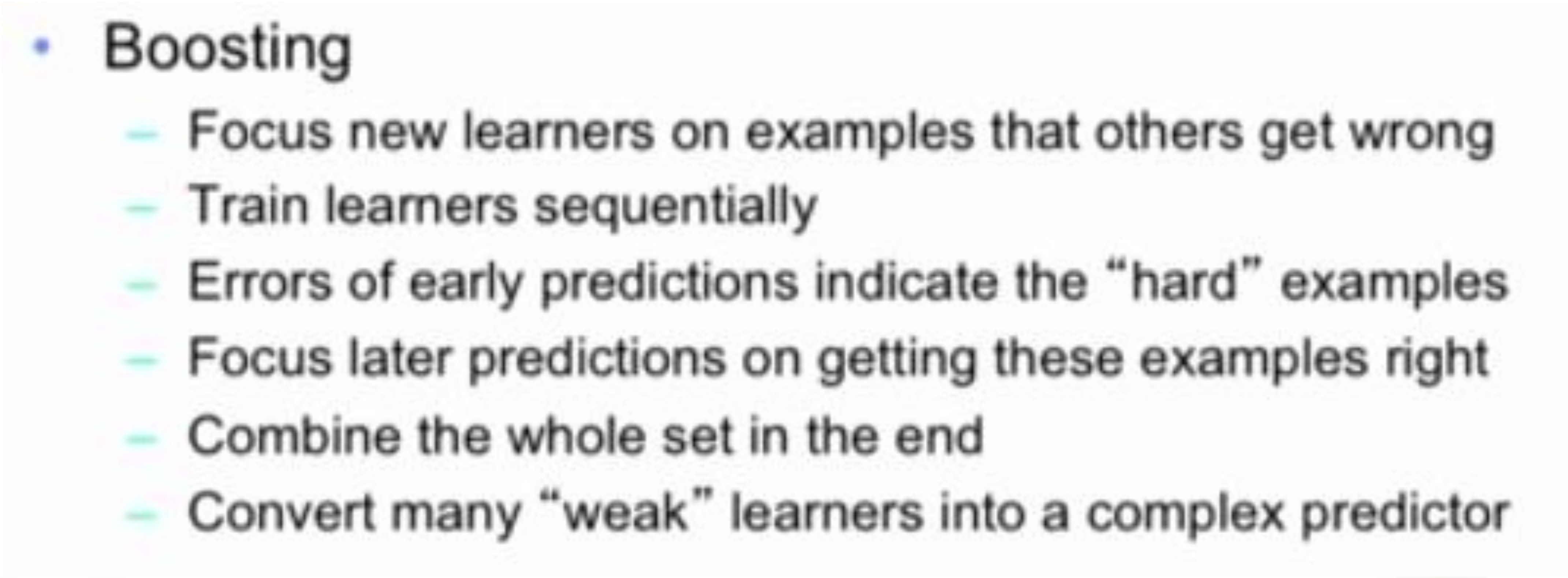
TEST SET

- When there is a test set, there are two different methods of replacement depending on whether labels exist for the test set.
- If they do, then the fills derived from the training set are used as replacements. If labels no not exist, then each case in the test set is replicated nclass times (nclass= number of classes). The first replicate of a case is assumed to be class 1 and the class one fills used to replace missing values. The 2nd replicate is assumed class 2 and the class 2 fills used on it.
- This augmented test set is run down the tree. In each set of replicates, the one receiving the most votes determines the class of the original case.

Source: http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#missing1

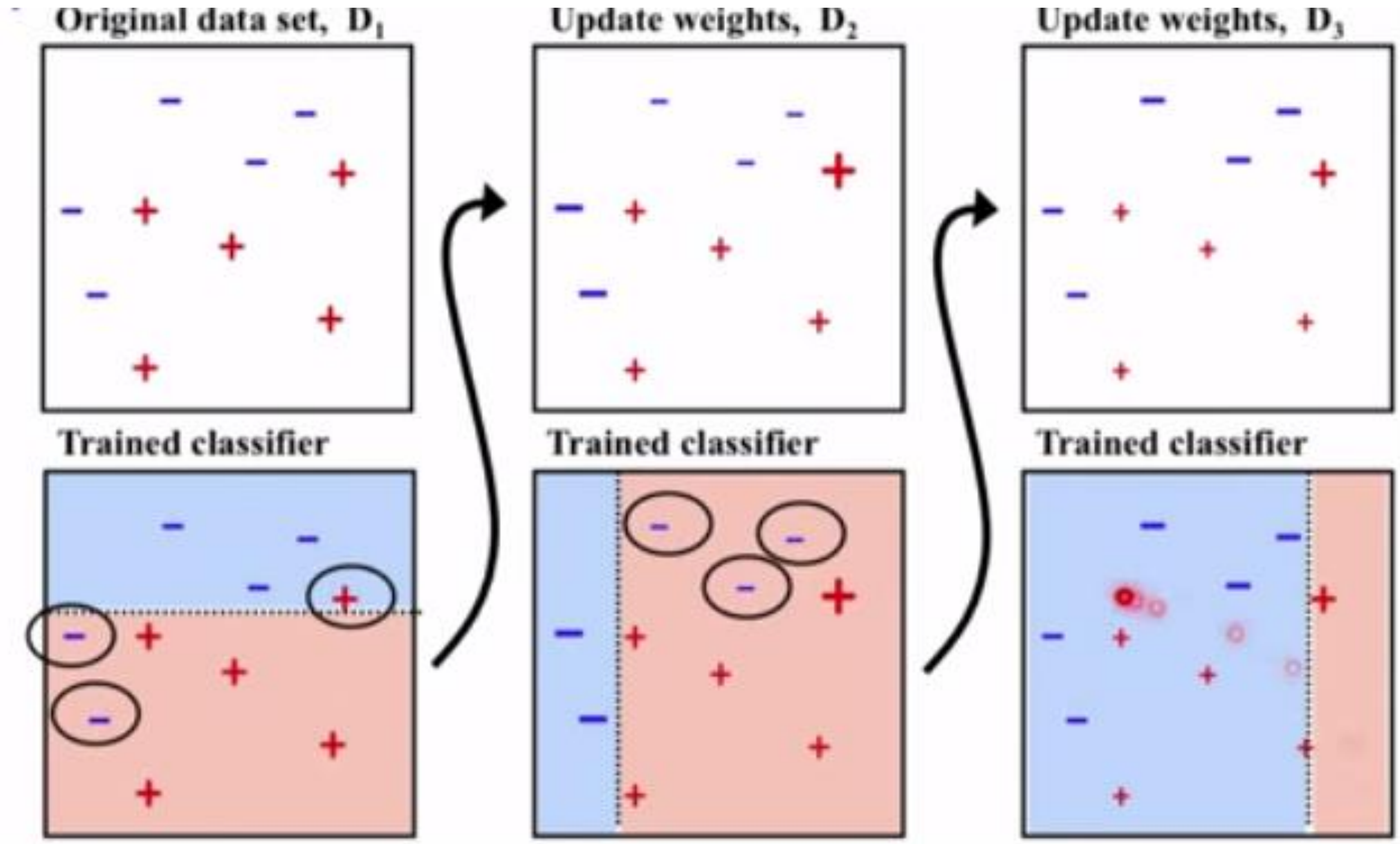
Boosting

Boosting

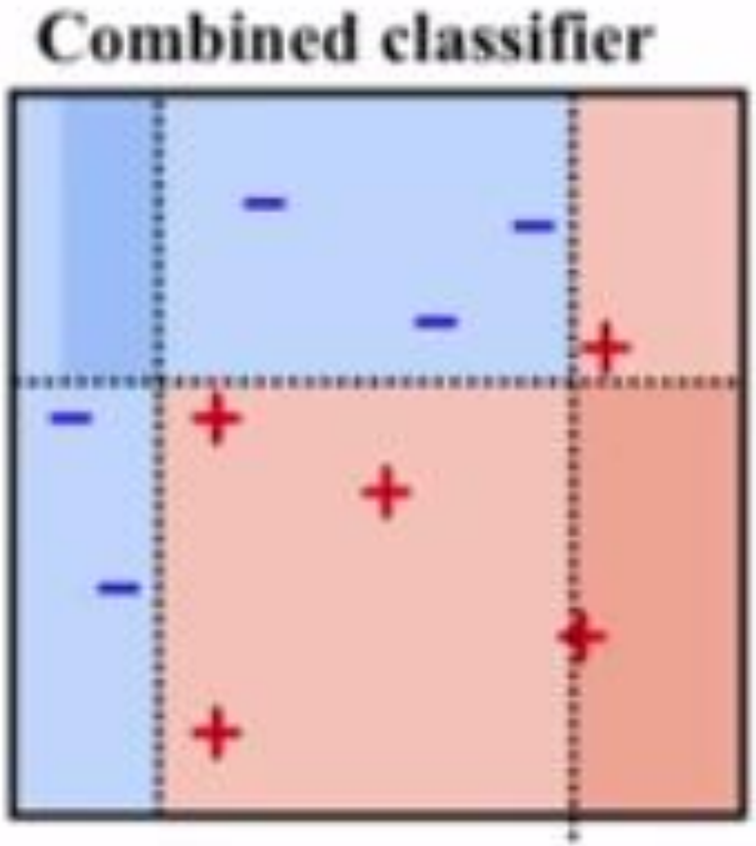


- Boosting
  - Focus new learners on examples that others get wrong
  - Train learners sequentially
  - Errors of early predictions indicate the “hard” examples
  - Focus later predictions on getting these examples right
  - Combine the whole set in the end
  - Convert many “weak” learners into a complex predictor

Ada- Boost


Original data set, D_1
Update weights, D_2
Update weights, D_3
Trained classifier
Trained classifier
Trained classifier

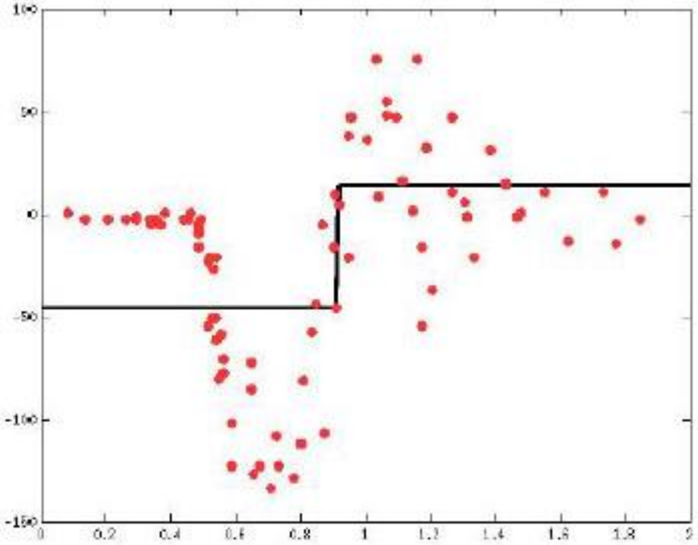
Ada-Boost



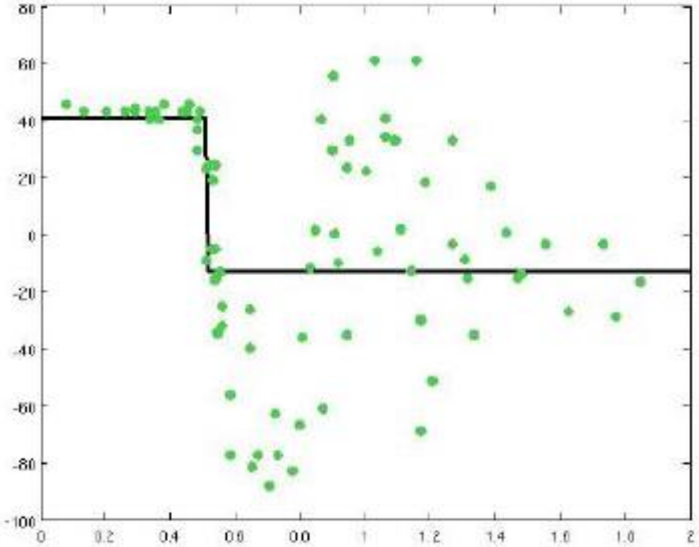
Gradient Boosting Machines

Gradient Boosting Machines

Learn a simple predictor…

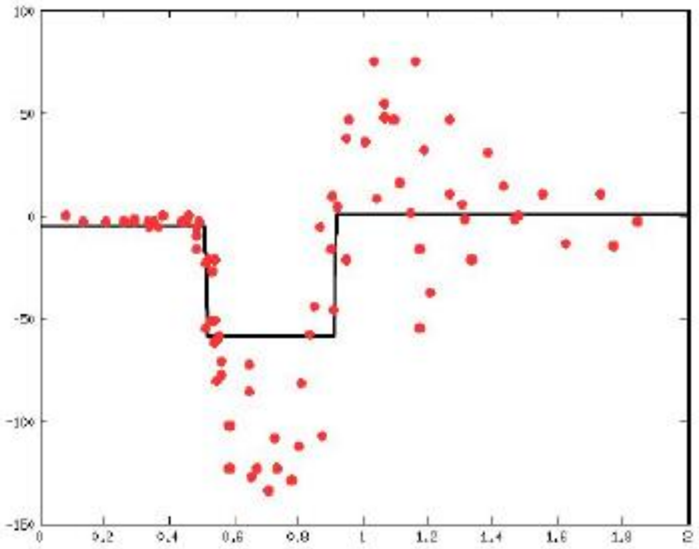


Then try to correct its errors

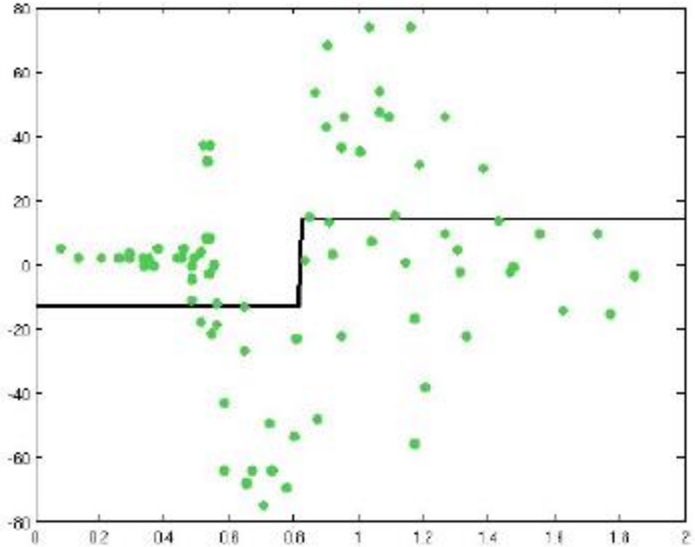


Gradient Boosting Machines

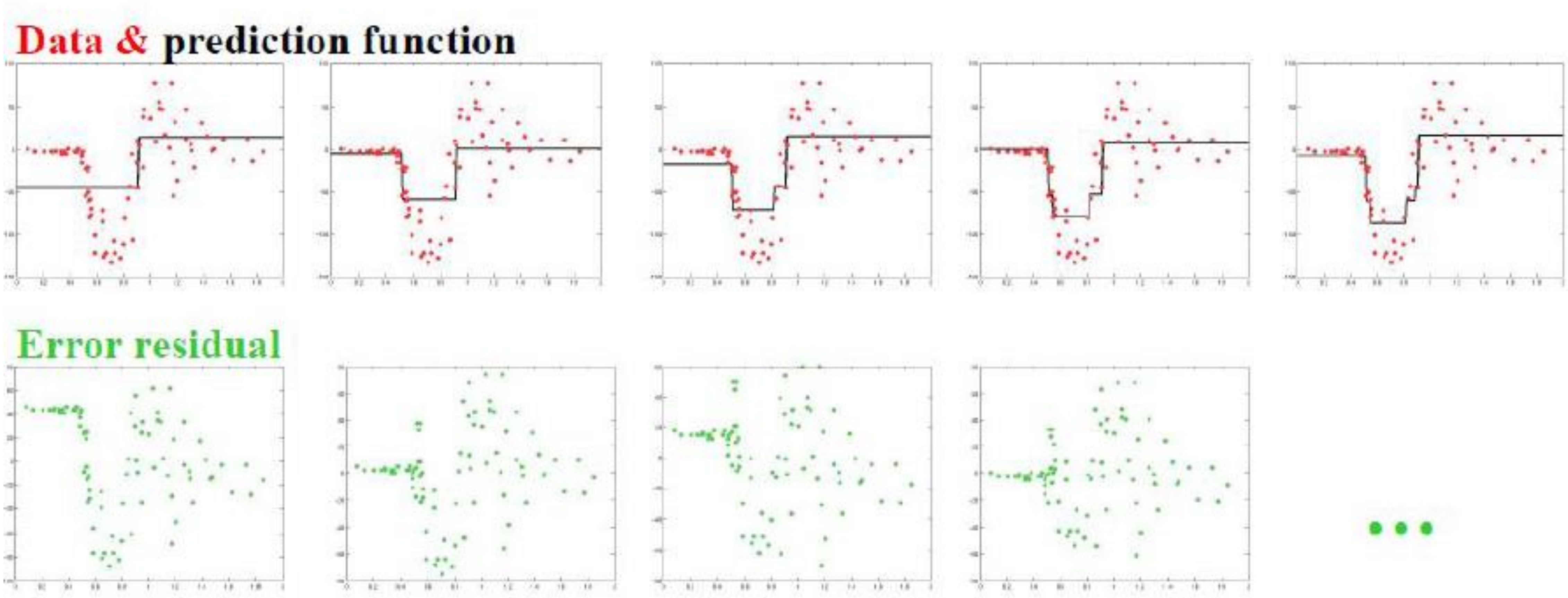
Combining gives a better predictor…



Can try to correct its errors also, & repeat



Gradient Boosting Machines



Gradient Boosting Machines

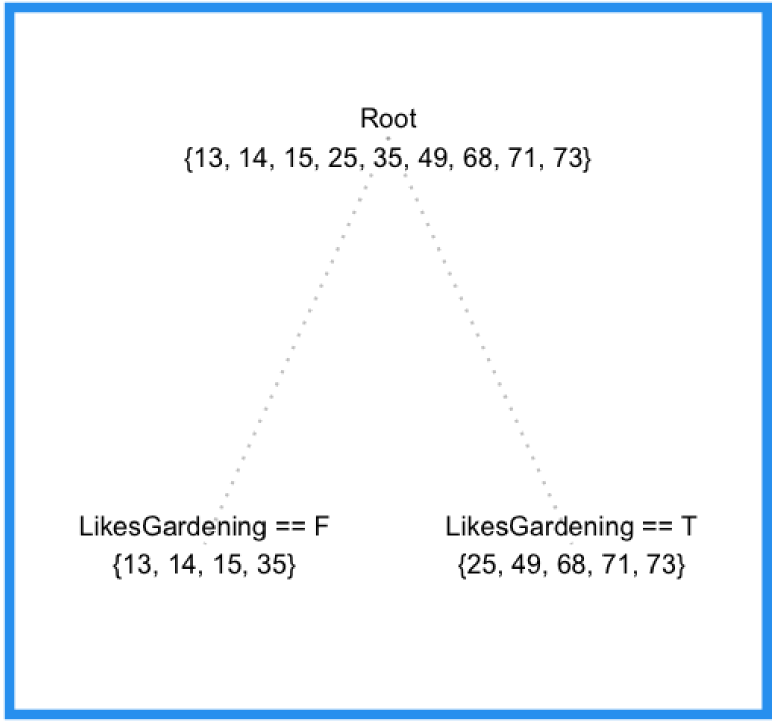
PersonID	Age	LikesGardening	PlaysVideoGames	LikesHats
1	13	FALSE	TRUE	TRUE
2	14	FALSE	TRUE	FALSE
3	15	FALSE	TRUE	FALSE
4	25	TRUE	TRUE	TRUE
5	35	FALSE	TRUE	TRUE
6	49	TRUE	FALSE	FALSE
7	68	TRUE	TRUE	TRUE
8	71	TRUE	FALSE	FALSE
9	73	TRUE	FALSE	TRUE

Source : http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/

Gradient Boosting Machines

PersonID	Age	LikesGardening	PlaysVideoGames	LikesHats
1	13	FALSE	TRUE	TRUE
2	14	FALSE	TRUE	FALSE
3	15	FALSE	TRUE	FALSE
4	25	TRUE	TRUE	TRUE
5	35	FALSE	TRUE	TRUE
6	49	TRUE	FALSE	FALSE
7	68	TRUE	TRUE	TRUE
8	71	TRUE	FALSE	FALSE
9	73	TRUE	FALSE	TRUE

Tree 1



Root
{13, 14, 15, 25, 35, 49, 68, 71, 73}

LikesGardening == F
{13, 14, 15, 35}

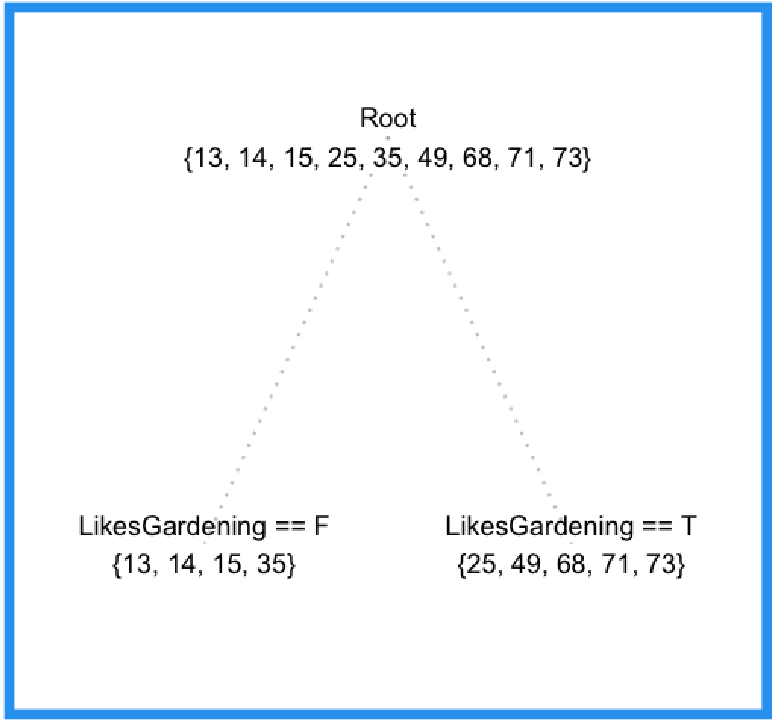
LikesGardening == T
{25, 49, 68, 71, 73}

Source : http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/

Gradient Boosting Machines

PersonID	Age	Tree1 Prediction	Tree1 Residual
1	13	19.25	-6.25
2	14	19.25	-5.25
3	15	19.25	-4.25
4	25	57.2	-32.2
5	35	19.25	15.75
6	49	57.2	-8.2
7	68	57.2	10.8
8	71	57.2	13.8
9	73	57.2	15.8

Tree 1



Root
{13, 14, 15, 25, 35, 49, 68, 71, 73}

LikesGardening == F
{13, 14, 15, 35}

LikesGardening == T
{25, 49, 68, 71, 73}

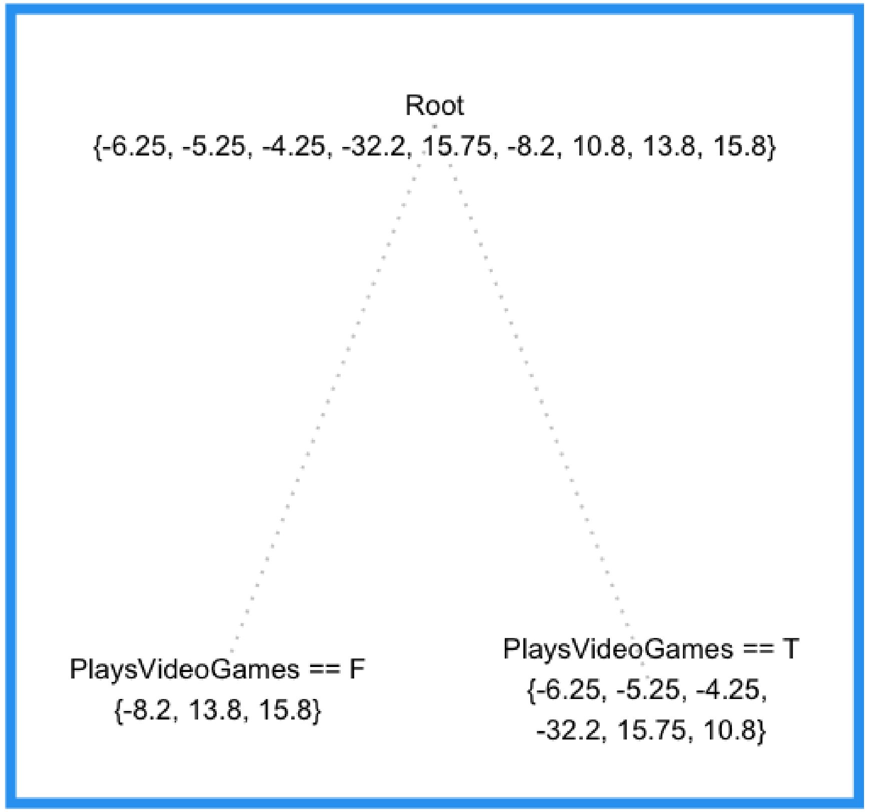
Now we can fit a second regression tree to the residuals of the first tree.

Source : http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/

Gradient Boosting Machines

PersonID	Age	Tree1 Prediction	Tree1 Residual
1	13	19.25	-6.25
2	14	19.25	-5.25
3	15	19.25	-4.25
4	25	57.2	-32.2
5	35	19.25	15.75
6	49	57.2	-8.2
7	68	57.2	10.8
8	71	57.2	13.8
9	73	57.2	15.8

Tree2



Root
{-6.25, -5.25, -4.25, -32.2, 15.75, -8.2, 10.8, 13.8, 15.8}

PlaysVideoGames == F
{-8.2, 13.8, 15.8}

PlaysVideoGames == T
{-6.25, -5.25, -4.25,
-32.2, 15.75, 10.8}

Now we can improve the predictions from our first tree by adding the "error-correcting" predictions from this tree.

Source : http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/

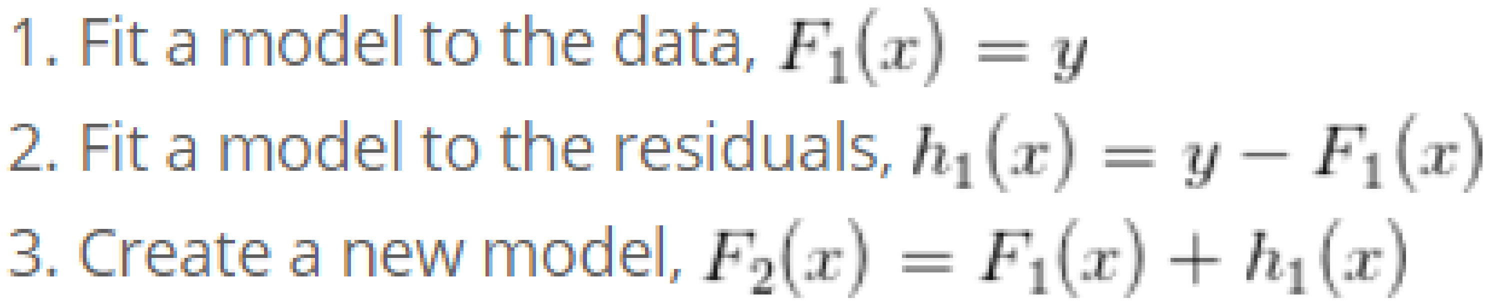
Gradient Boosting Machines

PersonID	Age	Tree1 Prediction	Tree1 Residual	Tree2 Prediction	Combined Prediction	Final Residual
1	13	19.25	-6.25	-3.567	15.68	2.683
2	14	19.25	-5.25	-3.567	15.68	1.683
3	15	19.25	-4.25	-3.567	15.68	0.6833
4	25	57.2	-32.2	-3.567	53.63	28.63
5	35	19.25	15.75	-3.567	15.68	-19.32
6	49	57.2	-8.2	7.133	64.33	15.33
7	68	57.2	10.8	-3.567	53.63	-14.37
8	71	57.2	13.8	7.133	64.33	-6.667
9	73	57.2	15.8	7.133	64.33	-8.667

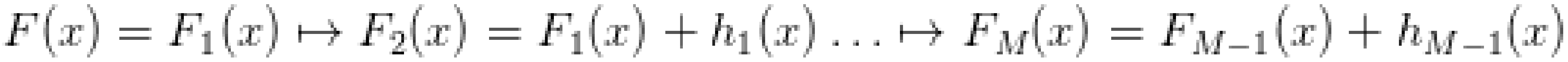
Tree1 SSE	Combined SSE
1994	1765

Source : http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/

GBM – Contd.



1. Fit a model to the data, $F_1(x) = y$
2. Fit a model to the residuals, $h_1(x) = y - F_1(x)$
3. Create a new model, $F_2(x) = F_1(x) + h_1(x)$



$$F(x) = F_1(x) \mapsto F_2(x) = F_1(x) + h_1(x) \ldots \mapsto F_M(x) = F_{M-1}(x) + h_{M-1}(x)$$

Why ‘Gradient’ Boosting Machine ?

- GBM makes use of Gradient Descent Algorithm at every step to find the best tree to be used to minimize the loss function

- Loss functions used in GBM for Regression are …
  - Squared Loss or MSE
  - Absolute Loss
  - Huber Loss etc…

Why ‘Gradient’ Boosting Machine ? Contd…

- GBM makes use of Gradient Descent Algorithm at every step to find the best tree to be used to minimize the loss function

- What about Classification?
  - The only change will be to use a differentiable loss function such as
    - Log Loss or Softmax

Cross- validation

Cross validation

- Cross validation is a model evaluation method that is better than residuals.
- The problem with residual evaluations is that they do not give an indication of how well the learner will do when it is asked to make new predictions for data it has not already seen.
- One way to overcome this problem is to not use the entire data set when training a model.
- Some of the data is removed before training begins. Then when training is done, the data that was removed can be used to test the performance of the learned model on ``new'' data.
- This is the basic idea for a whole class of model evaluation methods called *cross validation*.

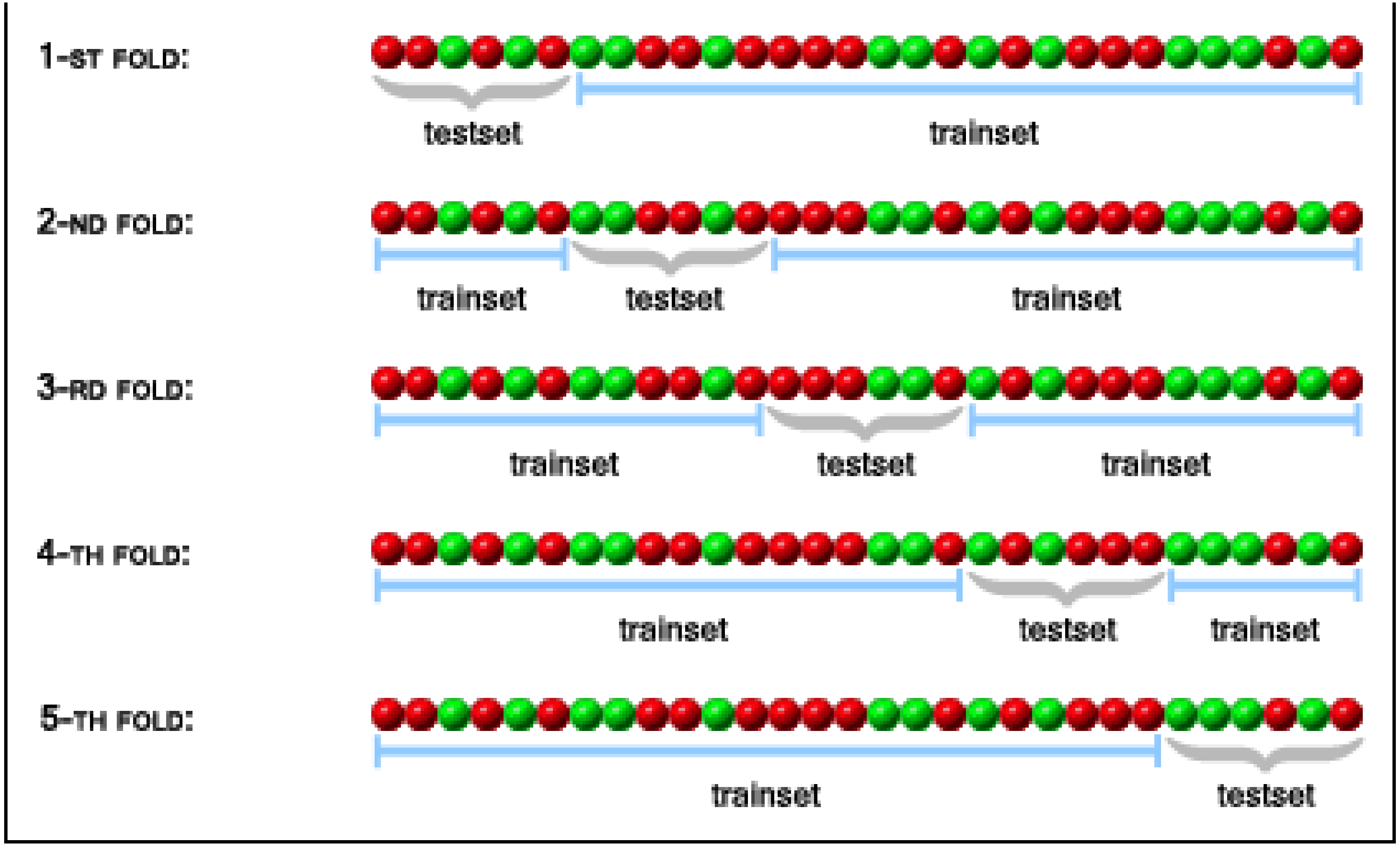
Cross Validation – Holdout Method |Train Test Split

- The **holdout method** is the simplest kind of cross validation. The data set is separated into two sets, called the training set and the testing set.
- The model is built using the training set only.

Dis-Advantages:

- The evaluation may depend heavily on which data points end up in the training set and which end up in the test set.
- Thus the evaluation may be significantly different depending on how the division is made.

Cross Validation – K Fold Cross Validation



5 fold CV

Cross Validation – Leave One Out Cross Validation

- When K = Number of records
- Done when the data is very less
- To get a realistic estimate of the errors

Let’s R