PSS718 - Data Mining

Lecture 10 - Boosting

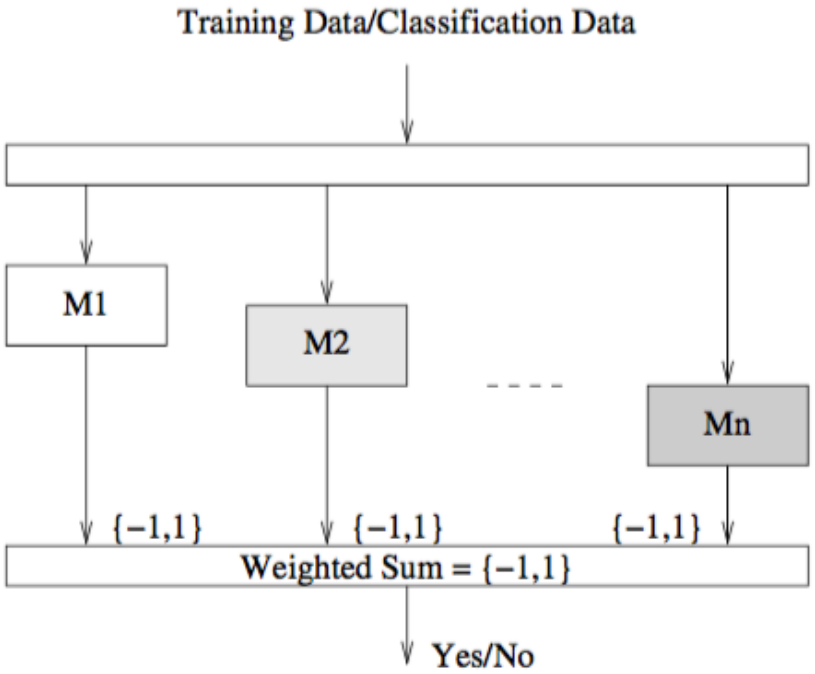
Asst.Prof.Dr. Burkay Genç

Hacettepe University

December 4, 2016

What is it?

- Boosting is a meta algorithm
- Efficient, simple, easy to use
- AdaBoost (adaptive boosting) is known as the "best off-the-shelf classifier in the world"



How does it work?

- Build multiple models using a "weak" learner
- Associate weights to observations
  - Increase weight for incorrectly classified observations
- Iteratively build new models and change weights
- Boosting may fail if not enough data exists
  - Or, if the weak models are too complex
  - And, also it is susceptible to noise

Representation

- Commonly presented in the form of decision trees
- Key extension is the way we merge the decisions
- A weighted score is used based on the quality of each individual model

Meta-learner

- Relies on a weak learner
- A weak leaner is slightly better than a random one
  - A good example is a single depth decision tree

Steps

1. Construct an initial "weak" model
2. Identify incorrect classifications and boost their weight
3. Build a new model from this boosted dataset
4. If the new model is acting better than random
   - TRUE: Go to Step 2
   - FALSE: Go to Step 5
5. Build an ensemble based on the accuracy of each model

Example

- Assume 10 observations: 1, 2 ... 10
- Initial weights are 0.1 each
- Assume first model is build and incorrectly classifies 7,8,9 and 10
- Let inaccuracy be the sum of the weights of the incorrectly classified observations:

$$\epsilon = 0.1 + 0.1 + 0.1 + 0.1 = 0.4$$

- And compute the weight of the model as,

$$\alpha = 0.5 * \log(\frac{1-\epsilon}{\epsilon}) = 0.5 * \log(\frac{1-0.4}{0.4}) = 0.2027$$

- Update weights of 7,8,9 and 10:

$$0.1 * e^{\alpha} = 0.1 * e^{0.2027} = 0.1225$$

Example

- Now build a new model with the updated weights
- The new model will give more importance to 7,8,9 and 10
- Assume that, this time 1 and 8 are incorrectly classified
  - Their weights are 0.1 and 0.1225 at the moment, so

$$\epsilon = 0.1 + 0.1225 = 0.2225$$

- Hence, compute α as,

$$\alpha = 0.5 * \log(\frac{1-\epsilon}{\epsilon}) = 0.6257$$

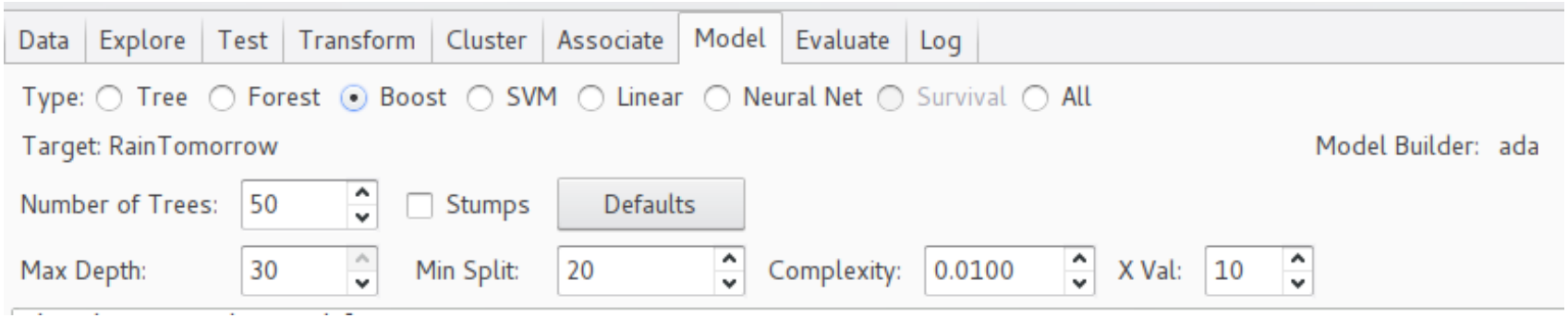
- Update weights of 1 and 8:
  - $0.1 * e^{0.6257} = 0.1869$
  - $0.1225 * e^{0.6257} = 0.229$

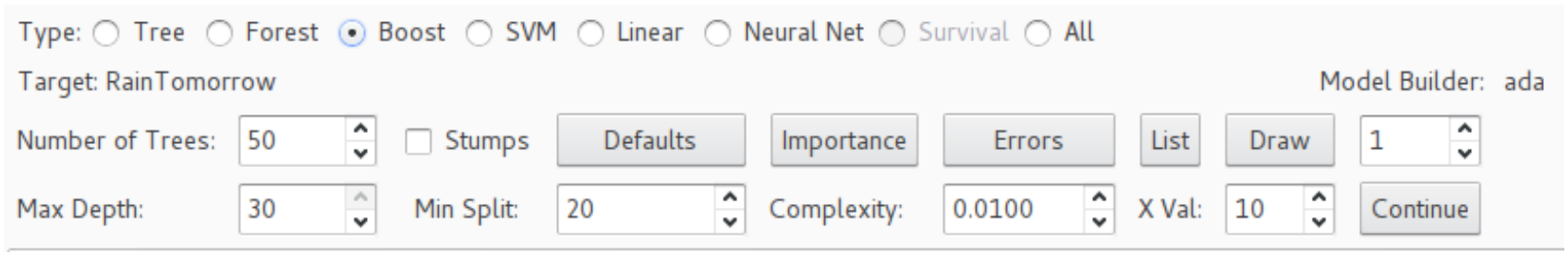
Example

- Each tree has a weight: α
- When making a decision, each tree has an individual decision
  - Multiply the decision by the tree's α
  - Compute the weighted average of the decision of all trees

$$D = \frac{\sum \alpha_i * d_i}{\sum \alpha_i}$$

User Interface





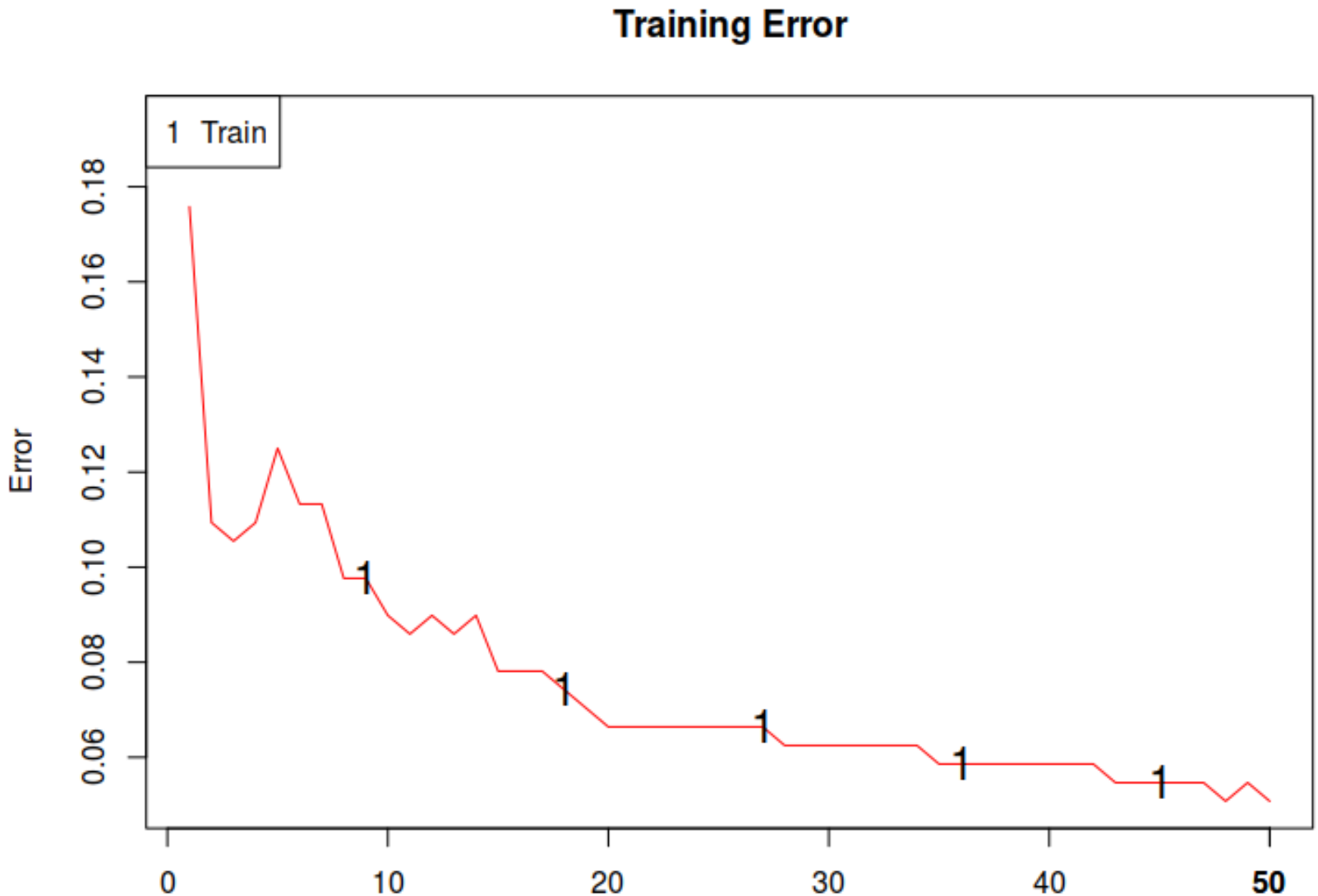
Result View

```
Summary of the Ada Boost model:

Call:
ada(RainTomorrow ~ ., data = crs$dataset[crs$train, c(crs$input,
    crs$target)], control = rpart::rpart.control(maxdepth = 30,
    cp = 0.01, minsplit = 20, xval = 10), iter = 50)

Loss: exponential Method: discrete   Iteration: 50

Final Confusion Matrix for Data:
          Final Prediction
True value  No Yes
       No  214   1
       Yes  12  29

Train Error: 0.051

Out-Of-Bag Error:  0.066  iteration= 39
```

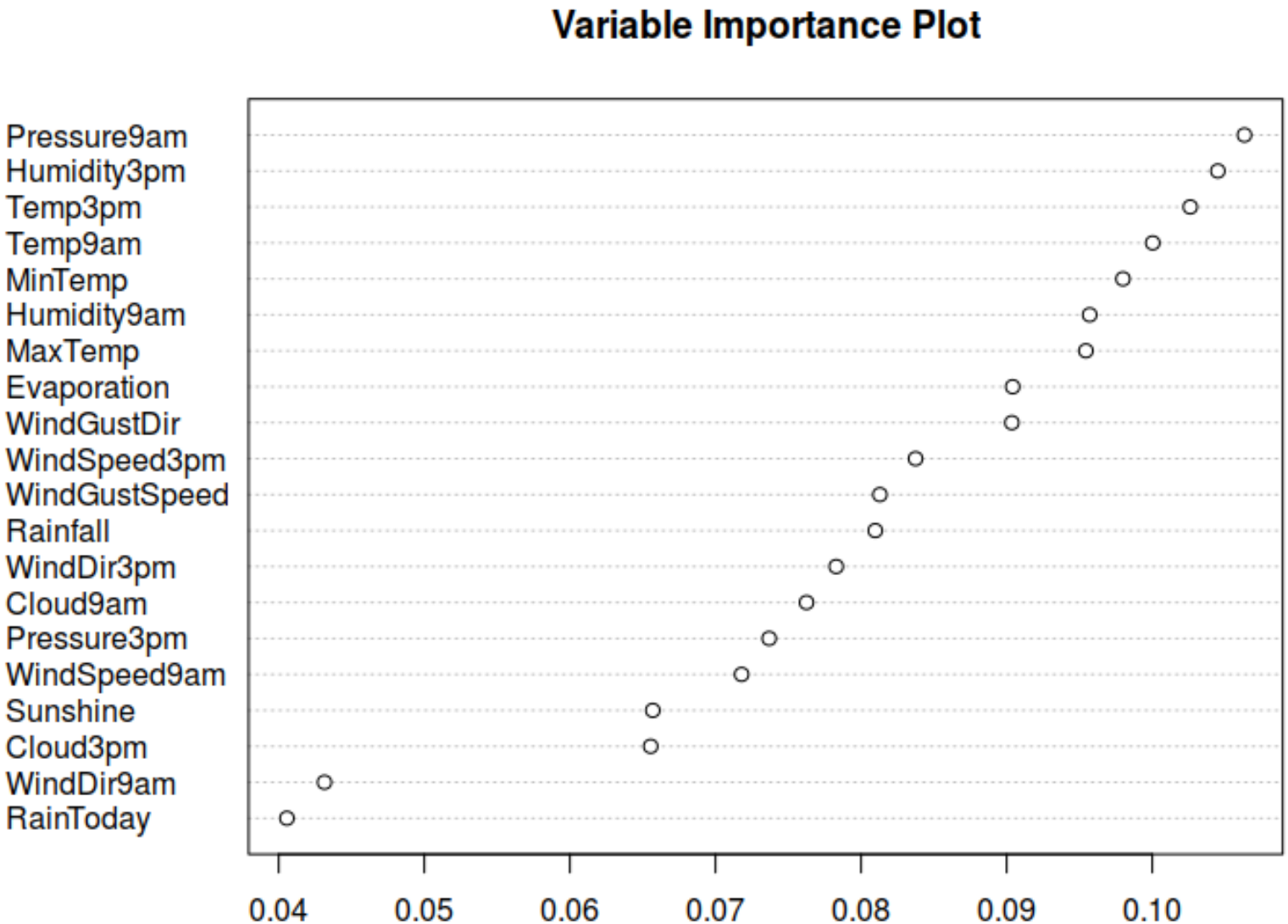
Result View

```
Additional Estimates of number of iterations:

train.err1 train.kap1
        48         48

Variables actually used in tree construction:
 [1] "Cloud3pm"      "Cloud9am"      "Evaporation"   "Humidity3pm"   "Humidity9am"   "MaxTemp"       "MinTemp"       "Pressure3pm"   "Pressure9am"
[10] "Rainfall"      "Sunshine"      "Temp3pm"       "Temp9am"       "WindDir3pm"    "WindDir9am"    "WindGustDir"   "WindGustSpeed" "WindSpeed3pm"
[19] "WindSpeed9am"

Frequency of variables actually used:

   WindDir9am   WindGustDir      Sunshine    WindDir3pm   Pressure3pm      Cloud3pm       MaxTemp       MinTemp       Temp9am  WindSpeed3pm
           36            26            25            25            23            12             8             6             6             6
  Evaporation WindGustSpeed      Cloud9am   Humidity3pm   Humidity9am   Pressure9am      Rainfall       Temp3pm  WindSpeed9am
            5             5             3             3             2             2             2             2             1

Time taken: 0.85 secs
```

Error Plot


Training Error
1 Train
Error
0.06
0.08
0.10
0.12
0.14
0.16
0.18
0
10
20
30
40
50

Importance Plot



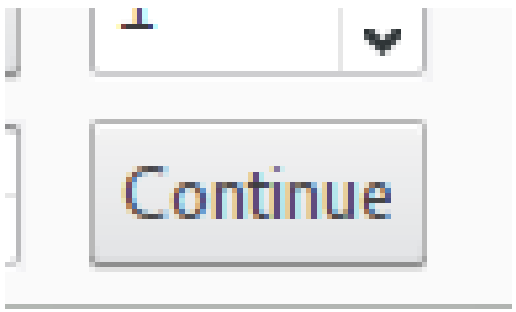
Tuning Parameters

- Number of Trees
- DT Parameters
  - MaxDepth
  - Min Split
  - Complexity

Adding Trees

Continue

You can use the Continue button to add more trees to the model without starting from scratch



Commands

control

```
control <- rpart.control(maxdepth=30, cp=0.010000,
minsplit=20, xval=10)
```

model

```
model <- ada(formula=form, data=data[train, vars],
control=control, iter=50)
```

Alternative Boosting Libraries

- caTools provides LogitBoost(). Simple to use and an efficient implementation for very large datasets
- gbm implements generalized boosted regression, providing a more widely applicable boosting algorithm
- mboost is another alternative offering model-based boosting