Overview of Hands-On Example & XGBoost

Nick Henscheid
Sr. Quantitative Medicine Scientist

QUANTITATIVE MEDICINE

CRITICAL PATH INSTITUTE

Agenda

- Overview of hands-on portion of the webinar (Thurs)
- Code installation tips
- Introduction to the practicum dataset: predicting breast cancer recurrence
- Conceptual & practical explanation of XGBoost for classification

Hands-on Practicum: Overview

- This Thursday from 11AM – 1PM EDT there is a hands-on coached tutorial session
- We will have two coaches per breakout room
- Instructions to download and install the code are in your inbox
- The code is written in R
- The GitHub Repository is:

https://github.com/metrumresearchgroup/ascpt-ml-webinar-2023

Please allow approximately one hour to install the necessary packages and test the code

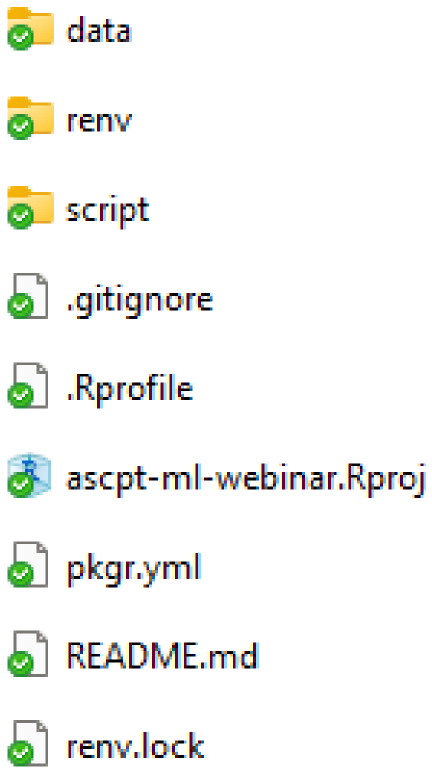
If you have issues, please email one of the coaches

Start after today's webinar to allow enough time to resolve issues before Thursday

Hands-on Practicum: Installation Tips

- Make sure you have R 4.1.3 or later and appropriate version of Rtools installed
- The .Rproj file provides an isolated environment to install packages
- Packages required:

  knitr, tidyverse, tidymodels, xgboost, here
- Use RStudio package installer or install.packages()
- The dataset is contained in the /data folder
- Test package installation with /script/test-script.R
- The workbook and answer key are in /script



data

renv

script

.gitignore

.Rprofile

ascpt-ml-webinar.Rproj

pkgr.yml

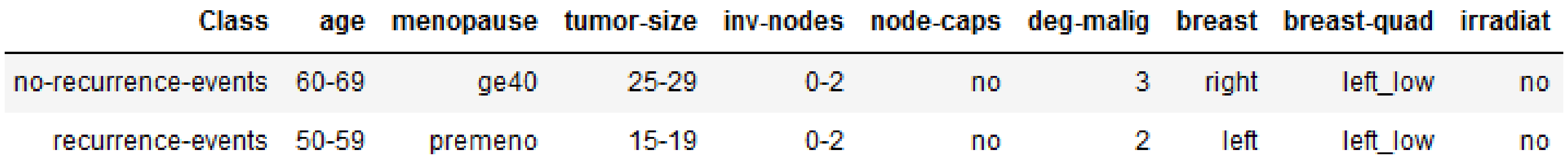
README.md

renv.lock

Example Dataset: UCI Breast Cancer

https://archive.ics.uci.edu/dataset/14/breast+cancer

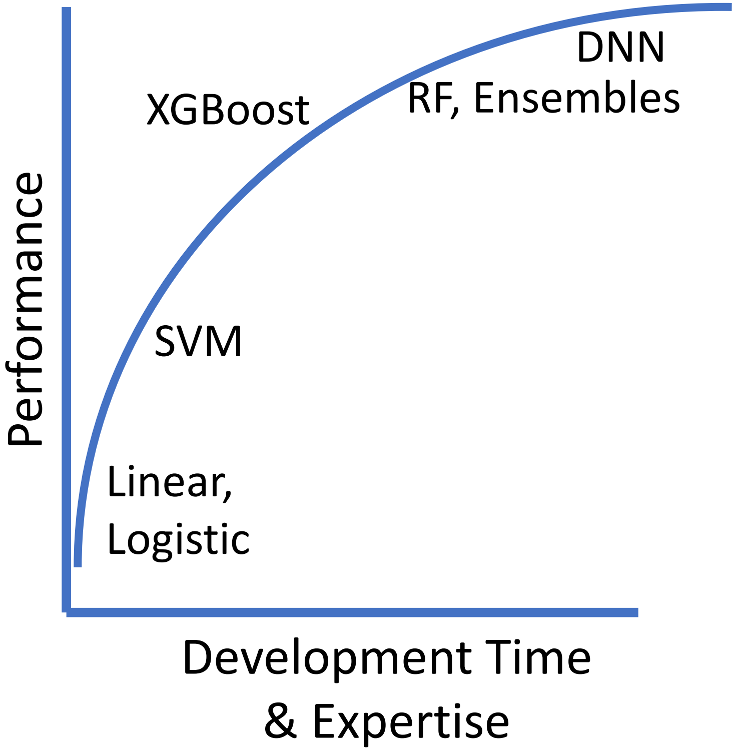
Variable Name	Role	Type	Values	Notes
Class	Target	Binary	no-recurrence-events, recurrence-events	
age	Feature	Categorical	10-19, 20-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89, 90-99	
menopause	Feature	Categorical	lt40, ge40, premeno	
tumor-size	Feature	Categorical	0-4, 5-9, 10-14, 15-19, 20-24, 25-29, 30-34, 35-39, 40-44, 45-49, 50-54, 55-59	
inv-nodes	Feature	Categorical	0-2, 3-5, 6-8, 9-11, 12-14, 15-17, 18-20, 21-23, 24-26, 27-29, 30-32, 33-35, 36-39	Number of lymph nodes showing metastases
node-caps	Feature	Binary	yes, no	Metastasis contained to lymph cap
deg-malig	Feature	Integer	1, 2, 3	Histological grade
breast	Feature	Binary	left, right	
breast-quad	Feature	Categorical	left-up, left-low, right-up, right-low, central	
irradiat	Feature	Binary	yes, no	Patient received rad therapy



Class	age	menopause	tumor-size	inv-nodes	node-caps	deg-malig	breast	breast-quad	irradiat
no-recurrence-events	60-69	ge40	25-29	0-2	no	3	right	left_low	no
recurrence-events	50-59	premeno	15-19	0-2	no	2	left	left_low	no

XGBoost: Overview

- XGBoost is a powerful *method* and a *software framework*
- It is ideal for large tabular datasets
- XGBoost can perform both classification and regression
- The algorithm is based on regularized gradient boosting but has added features to improve performance, reliability and usability
- XGBoost has many hyperparameters…
- …but tidymodels can help
- In terms of performance versus labor, XGBoost ranks highly: competitive performance across many tasks and datasets with minimal development

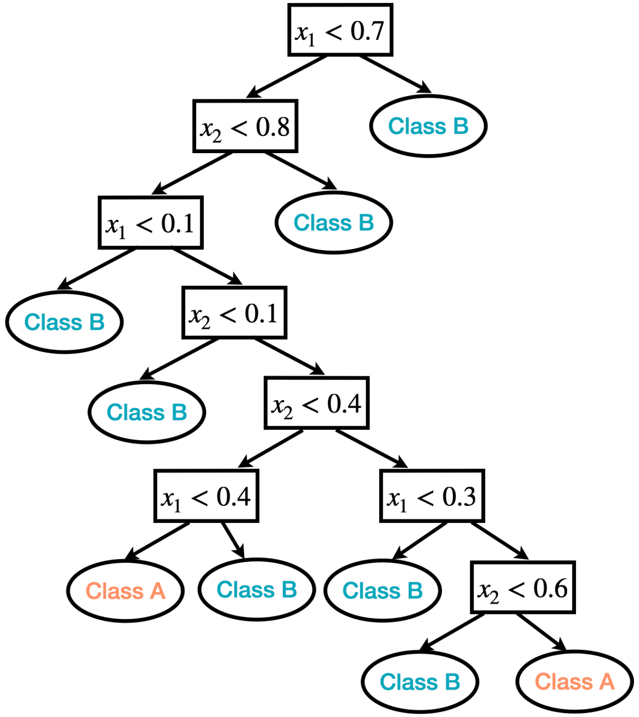


XGBoost: Algorithm Architecture

- XGBoost is based on the idea of an *ensemble*:

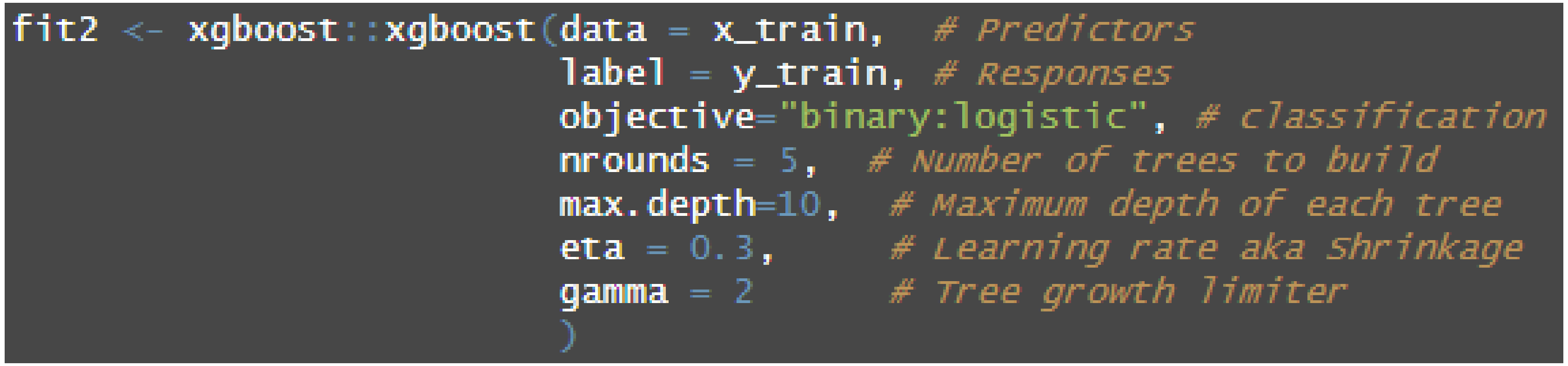
$$f(x) = \sum f_j(x)$$

- Each individual model ($f_j(x)$) is 'weak', but combining them results in a model with good performance
- The ensemble models in XGBoost are trees (by default the first model predicts 0.5 for everything).
- XGBoost trees are built iteratively by solving a regularized optimization problem designed to reduce errors made by previous model
- Trees are also pruned to reduce overfitting
- XGBoost has native parallel & distributed capabilities



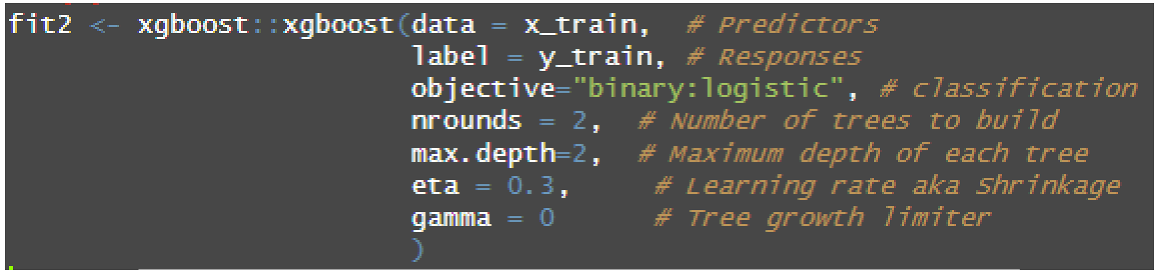
XGBoost: Hyperparameters

- XGBoost has an array of hyperparameters that influence how many trees are built, how deep the trees are, and how aggressively to regularize
- The full list of hyperparameters is found in the documentation; here are a few:
  - nrounds: controls the number of trees to build
  - max.depth: controls the maximum depth of each tree
  - eta: controls the weight that each tree is given in the final prediction (aka the learning rate)
  - gamma: limits the growth of each tree by eliminating branches that might overfit

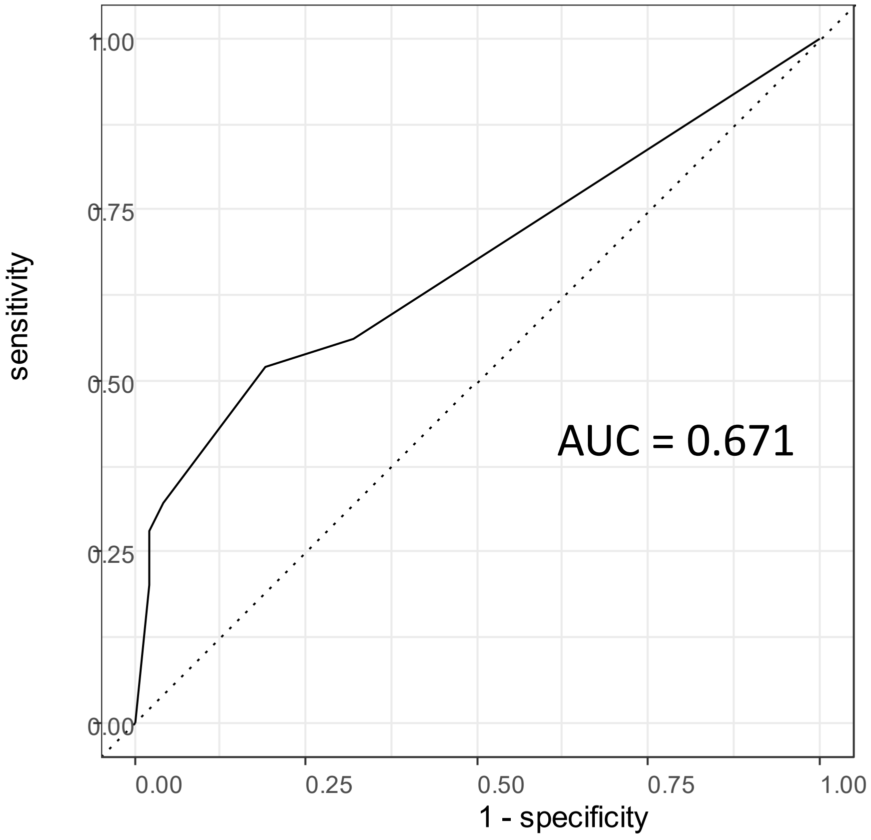


```
fit2 <- xgboost::xgboost(data = x_train,   # Predictors
                         label = y_train, # Responses
                         objective="binary:logistic", # classification
                         nrounds = 5,   # Number of trees to build
                         max.depth=10,   # Maximum depth of each tree
                         eta = 0.3,       # Learning rate aka Shrinkage
                         gamma = 2        # Tree growth limiter
                         )
```

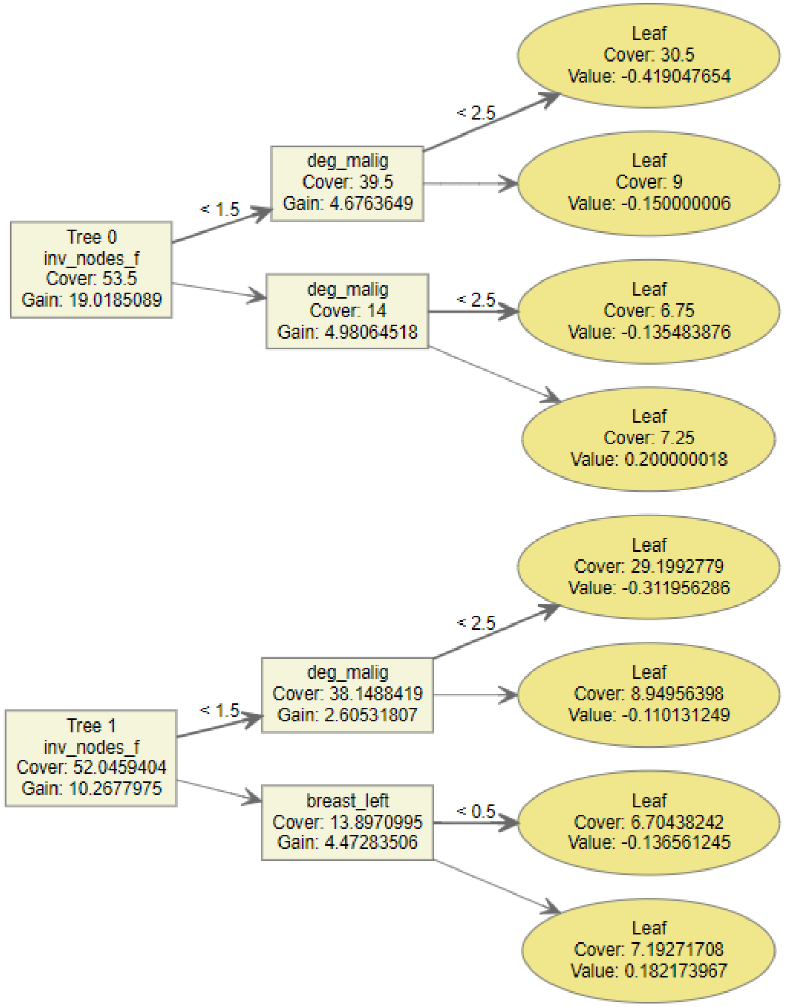
XGBoost: Hyperparameter examples



```
fit2 <- xgboost::xgboost(data = x_train,  # Predictors
                         label = y_train, # Responses
                         objective="binary:logistic", # classification
                         nrounds = 2,  # Number of trees to build
                         max.depth=2,  # Maximum depth of each tree
                         eta = 0.3,     # Learning rate aka Shrinkage
                         gamma = 0      # Tree growth limiter
                         )
```



AUC = 0.671



Tree 0
inv_nodes_f
Cover: 53.5
Gain: 19.0185089

< 1.5

deg_malig
Cover: 39.5
Gain: 4.6763649

< 2.5

Leaf
Cover: 30.5
Value: -0.419047654

Leaf
Cover: 9
Value: -0.150000006

deg_malig
Cover: 14
Gain: 4.98064518

< 2.5

Leaf
Cover: 6.75
Value: -0.135483876

Leaf
Cover: 7.25
Value: 0.200000018

Tree 1
inv_nodes_f
Cover: 52.0459404
Gain: 10.2677975

< 1.5

deg_malig
Cover: 38.1488419
Gain: 2.60531807

< 2.5

Leaf
Cover: 29.1992779
Value: -0.311956286

Leaf
Cover: 8.94956398
Value: -0.110131249

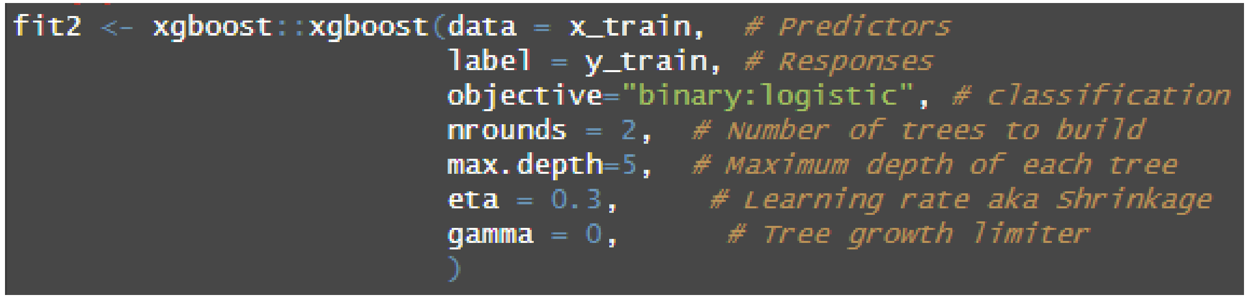
breast_left
Cover: 13.8970995
Gain: 4.47283506

< 0.5

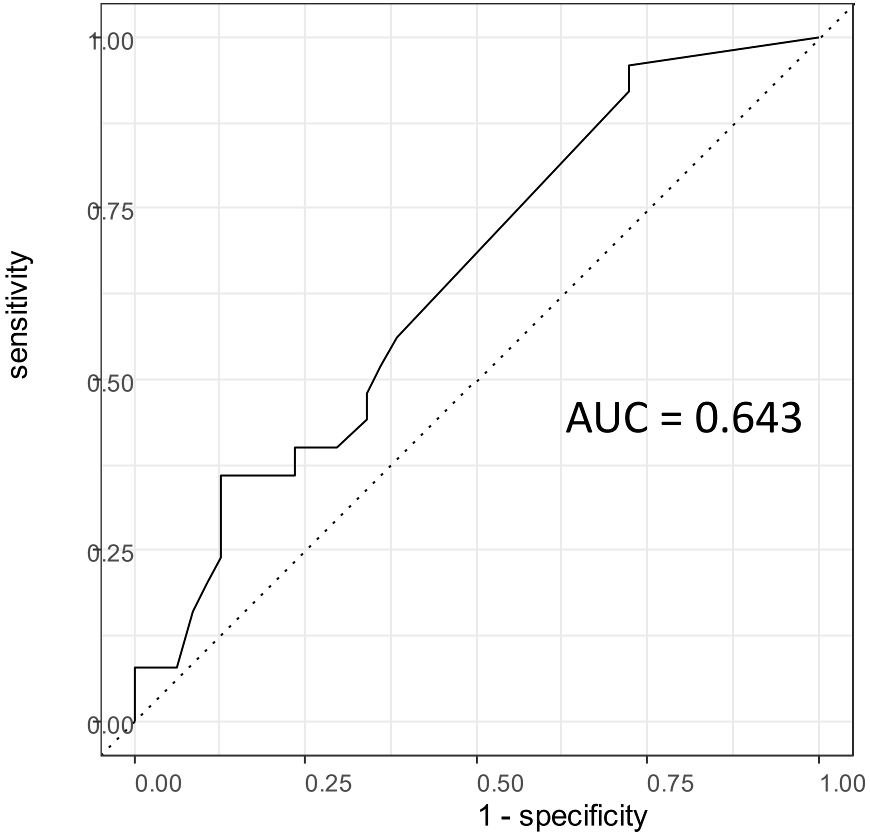
Leaf
Cover: 6.70438242
Value: -0.136561245

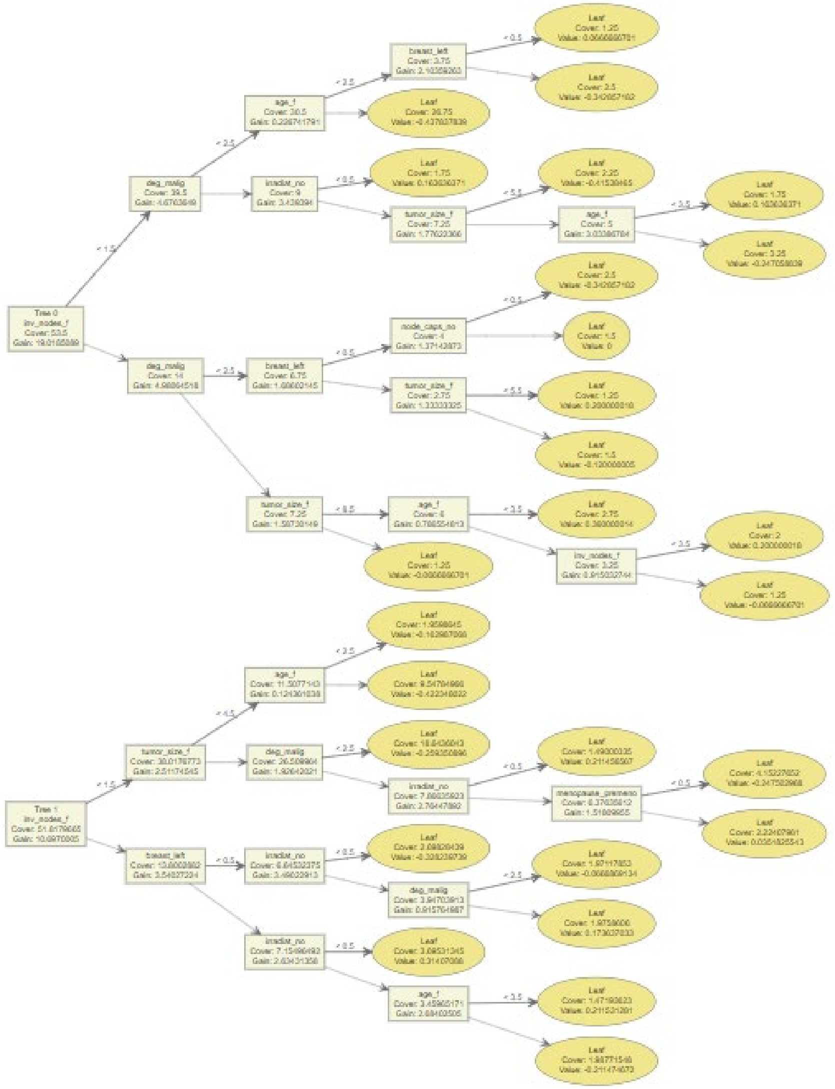
Leaf
Cover: 7.19271708
Value: 0.182173967

XGBoost: Hyperparameter examples

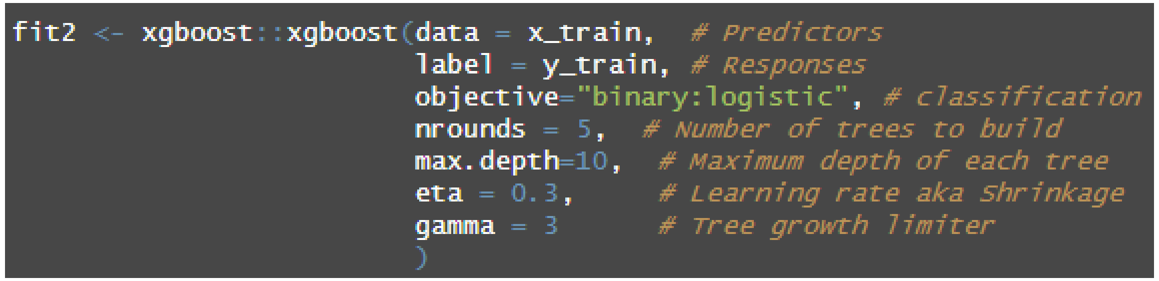


```
fit2 <- xgboost::xgboost(data = x_train,  # Predictors
                         label = y_train, # Responses
                         objective="binary:logistic", # classification
                         nrounds = 2,  # Number of trees to build
                         max.depth=5,  # Maximum depth of each tree
                         eta = 0.3,     # Learning rate aka Shrinkage
                         gamma = 0,      # Tree growth limiter
                         )
```

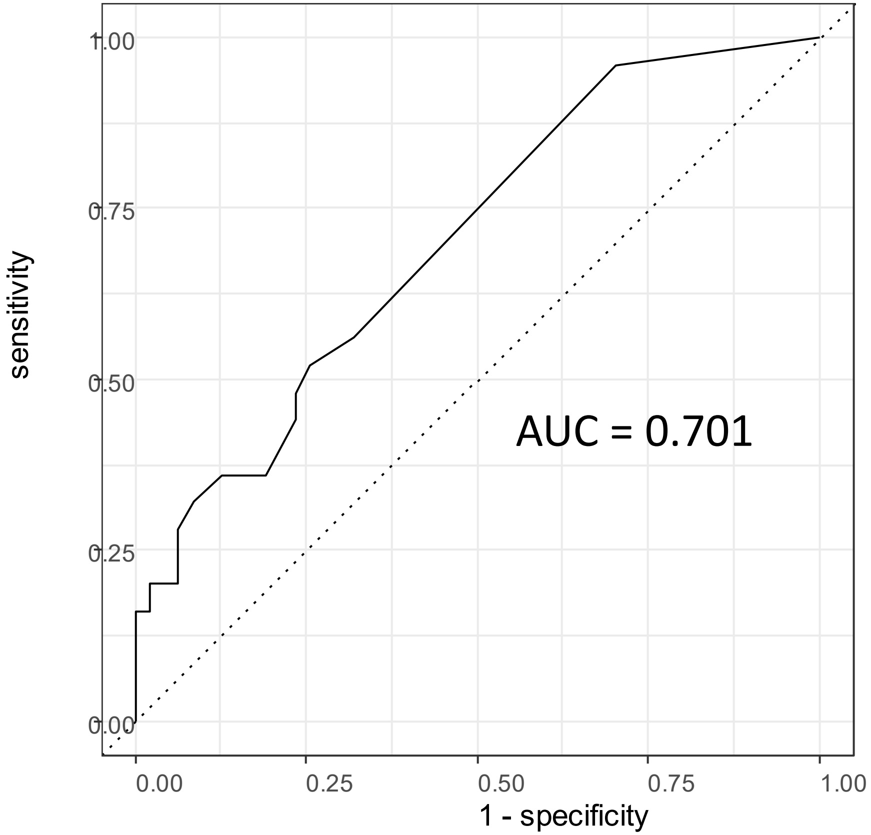


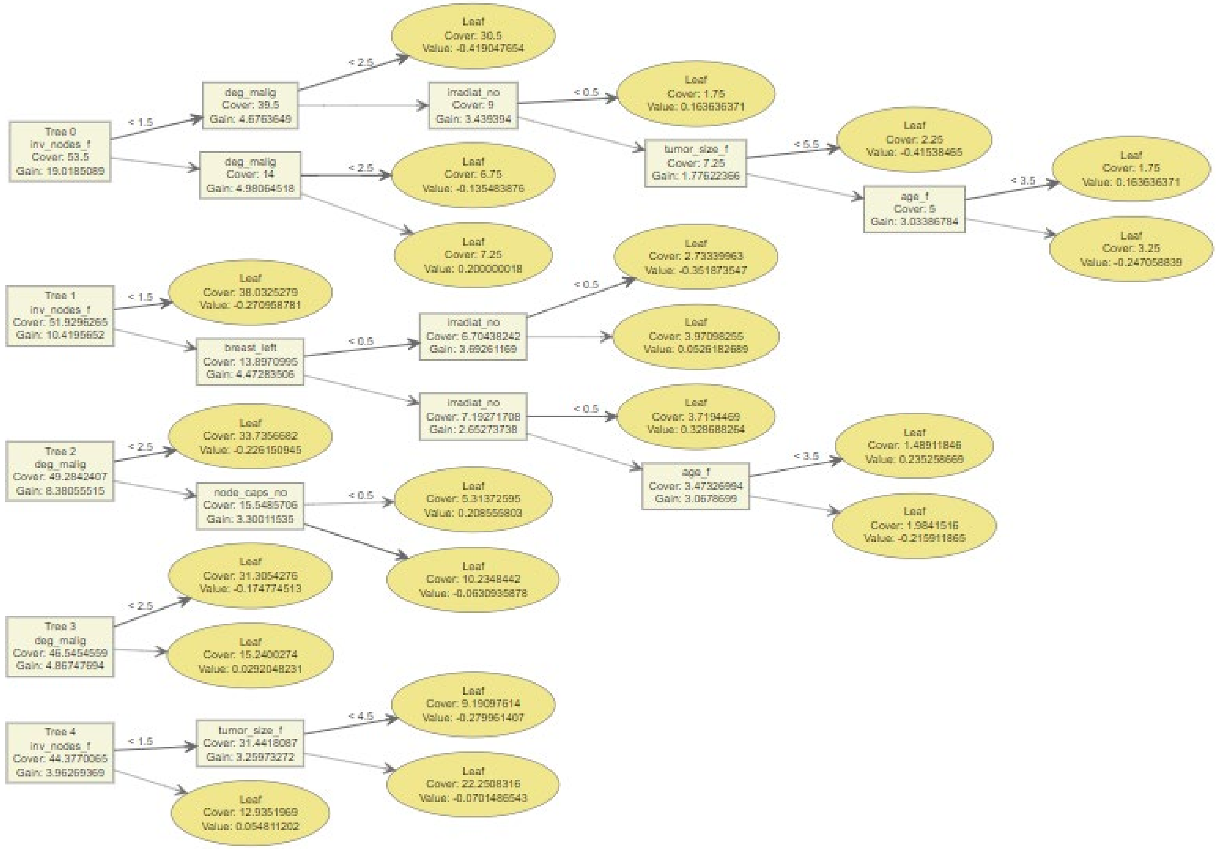


XGBoost: Hyperparameter examples



```
fit2 <- xgboost::xgboost(data = x_train,   # Predictors
                         label = y_train, # Responses
                         objective="binary:logistic", # classification
                         nrounds = 5,  # Number of trees to build
                         max.depth=10,  # Maximum depth of each tree
                         eta = 0.3,     # Learning rate aka Shrinkage
                         gamma = 3      # Tree growth limiter
                         )
```

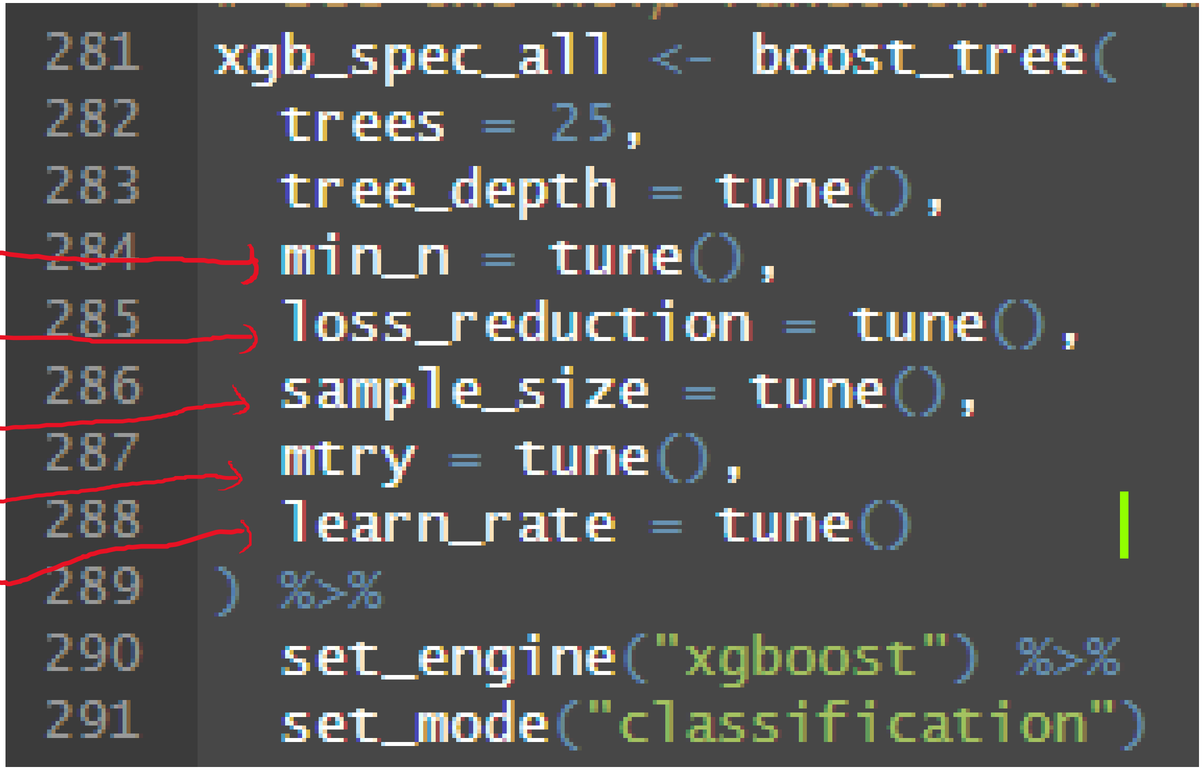



Tree 0 inv_nodes_f Cover: 53.5 Gain: 19.0185089
< 1.5
deg_malig Cover: 39.5 Gain: 4.6763649
< 2.5
Leaf Cover: 30.5 Value: -0.419047654
irradiat_no Cover: 9 Gain: 3.439394
< 0.5
Leaf Cover: 1.75 Value: 0.163636371
tumor_size_f Cover: 7.25 Gain: 1.77622366
< 5.5
Leaf Cover: 2.25 Value: -0.41538465
age_f Cover: 5 Gain: 3.03386784
< 3.5
Leaf Cover: 1.75 Value: 0.163636371
Leaf Cover: 3.25 Value: -0.247058839
deg_malig Cover: 14 Gain: 4.98064518
< 2.5
Leaf Cover: 6.75 Value: -0.135483876
Leaf Cover: 7.25 Value: 0.200000018
Tree 1 inv_nodes_f Cover: 51.9296265 Gain: 10.4195652
< 1.5
Leaf Cover: 38.0325279 Value: -0.270958781
breast_left Cover: 13.8970995 Gain: 4.47283506
< 0.5
irradiat_no Cover: 6.70438242 Gain: 3.69261169
< 0.5
Leaf Cover: 2.73339963 Value: -0.351873547
Leaf Cover: 3.97098255 Value: 0.0526182689
irradiat_no Cover: 7.19271708 Gain: 2.65273738
< 0.5
Leaf Cover: 3.7194469 Value: 0.328688264
age_f Cover: 3.47326994 Gain: 3.0678699
< 3.5
Leaf Cover: 1.48911846 Value: 0.235258669
Leaf Cover: 1.9841516 Value: -0.215911865
Tree 2 deg_malig Cover: 49.2842407 Gain: 8.38055515
< 2.5
Leaf Cover: 33.7356682 Value: -0.226150945
node_caps_no Cover: 15.5485706 Gain: 3.30011535
< 0.5
Leaf Cover: 5.31372595 Value: 0.208556803
Leaf Cover: 10.2348442 Value: -0.0630935878
Tree 3 deg_malig Cover: 46.5454559 Gain: 4.86747694
< 2.5
Leaf Cover: 31.3054276 Value: -0.174774513
Leaf Cover: 15.2400274 Value: 0.0292048231
Tree 4 inv_nodes_f Cover: 44.3770065 Gain: 3.96269369
< 1.5
tumor_size_f Cover: 31.4418087 Gain: 3.25973272
< 4.5
Leaf Cover: 9.19097614 Value: -0.279961407
Leaf Cover: 22.2508316 Value: -0.0701486543
Leaf Cover: 12.9351969 Value: 0.054811202

XGBoost: Parameter Tuning with tidymodels

XGBoost can be challenging to tune, but tidymodels makes this easy.
See Julia Silge’s video https://www.youtube.com/watch?v=hpudxAmxHSM
and blog post https://juliasilge.com/blog/xgboost-tune-volleyball/

- Number of trees
- Maximum tree depth
- Minimum # of data points to split
- Minimum loss reduction to split
- % of samples for training
- % of predictors sampled
- Learning rate



```r
xgb_spec_all <- boost_tree(
  trees = 25,
  tree_depth = tune(),
  min_n = tune(),
  loss_reduction = tune(),
  sample_size = tune(),
  mtry = tune(),
  learn_rate = tune()
) %>%
  set_engine("xgboost") %>%
  set_mode("classification")
```

Additional Resources

- https://xgboost.readthedocs.io/en/stable/index.html
- XGBoost on StatQuest
- hackerearth tutorial
- XGBoost Presentation by Tianqi Chen (one of the XGBoost developers)
- List of examples and challenge wins with XGBoost
- In-depth explanation of XGBoost (with math)