Plotting trees from Random Forest models with ggraph

- Tweet
-
-

16 March 2017

Today, I want to show how I use Thomas Lin Pedersen's awesome ggraph (https://github.com/thomasp85/ggraph) package to plot decision trees from Random Forest models.

I am very much a visual person, so I try to plot as much of my results as possible because it helps me get a better feel for what is going on with my data.

A nice aspect of using tree-based machine learning, like Random Forest models, is that that they are more easily interpreted than e.g. neural networks as they are based on decision trees. So, when I am using such models, I like to plot final decision trees (if they aren't too large) to get a sense of which decisions are underlying my predictions.

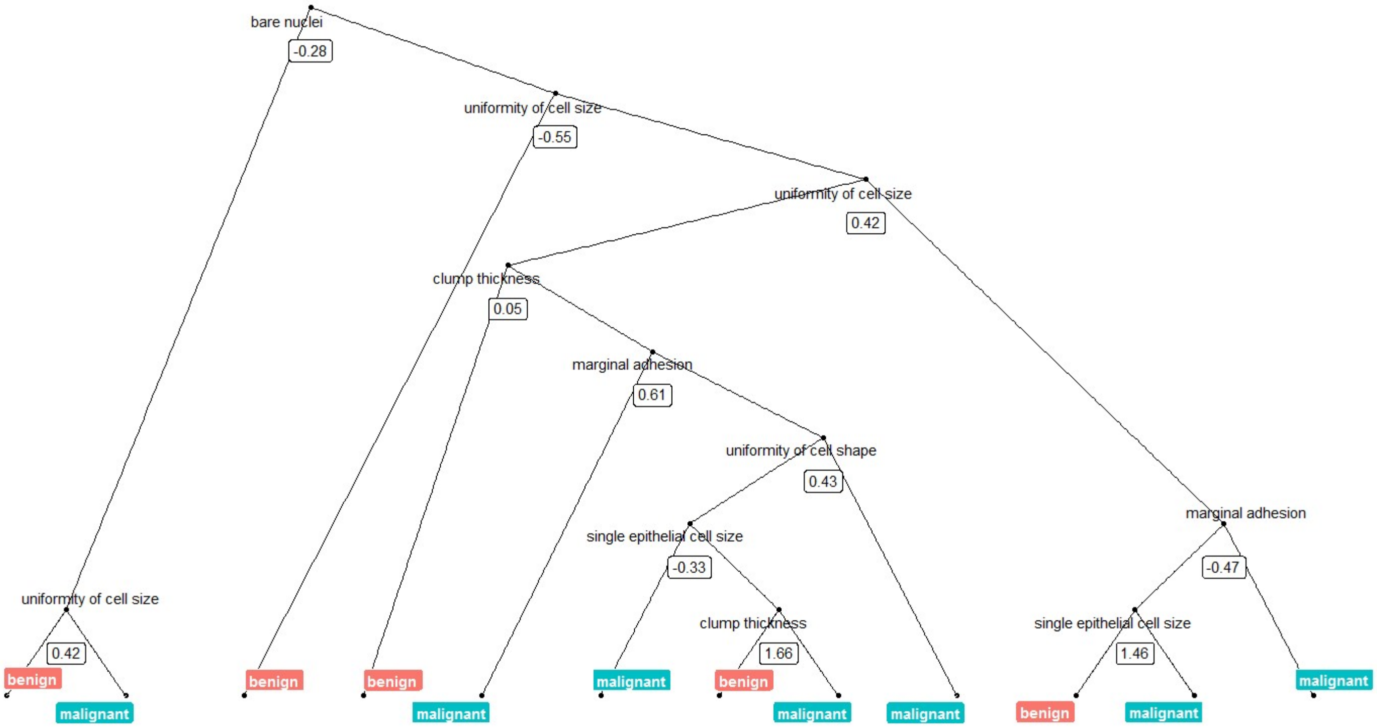
There are a few very convient ways to plot the outcome if you are using the `randomForest` package but I like to have as much control as possible about the layout, colors, labels, etc. And because I didn't find a solution I liked for `caret` models, I developed the following little function (below you may find information about how I built the model):

As input, it takes part of the output from `model_rf <- caret::train(... "rf" ...)` , that gives the trees of the final model: `model_rf$finalModel$forest` . From these trees, you can specify which one to plot by index.

```
library(dplyr)
library(ggraph)
library(igraph)

tree_func <- function(final_model, 
                      tree_num) {
  
  # get tree by index
  tree <- randomForest::getTree(final_model, 
                                k = tree_num, 
                                labelVar = TRUE) %>%
    tibble::rownames_to_column() %>%
    # make leaf split points to NA, so the 0s won't get plotted
    mutate(`split point` = ifelse(is.na(prediction), `split point`, NA))
  
  # prepare data frame for graph
  graph_frame <- data.frame(from = rep(tree$rowname, 2),
                            to = c(tree$`left daughter`, tree$`right daughter`))
  
  # convert to graph and delete the last node that we don't want to plot
  graph <- graph_from_data_frame(graph_frame) %>%
    delete_vertices("0")
  
  # set node labels
  V(graph)$node_label <- gsub("_", " ", as.character(tree$`split var`))
  V(graph)$leaf_label <- as.character(tree$prediction)
  V(graph)$split <- as.character(round(tree$`split point`, digits = 2))
  
  # plot
  plot <- ggraph(graph, 'dendrogram') + 
    theme_bw() +
    geom_edge_link() +
    geom_node_point() +
    geom_node_text(aes(label = node_label), na.rm = TRUE, repel = TRUE) +
    geom_node_label(aes(label = split), vjust = 2.5, na.rm = TRUE, fill = "white") +
    geom_node_label(aes(label = leaf_label, fill = leaf_label), na.rm = TRUE, 
                                        repel = TRUE, colour = "white", fontface = "bold", show.le
gend = FALSE) +
    theme(panel.grid.minor = element_blank(),
          panel.grid.major = element_blank(),
          panel.background = element_blank(),
          plot.background = element_rect(fill = "white"),
          panel.border = element_blank(),
          axis.line = element_blank(),
          axis.text.x = element_blank(),
          axis.text.y = element_blank(),
          axis.ticks = element_blank(),
          axis.title.x = element_blank(),
          axis.title.y = element_blank(),
          plot.title = element_text(size = 18))
  print(plot)
}
```

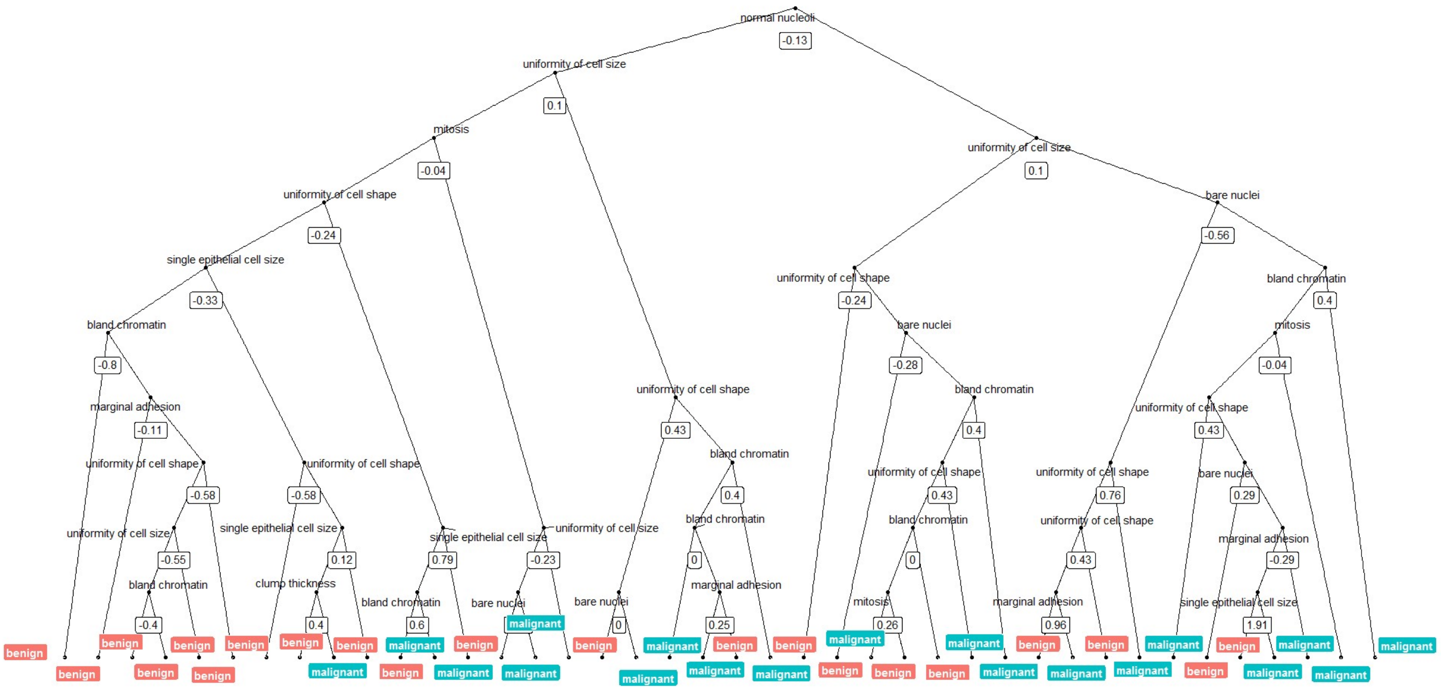
We can now plot, e.g. the tree with the smalles number of nodes:

```
tree_num <- which(model_rf$finalModel$forest$ndbigtree == min(model_rf$finalModel$forest$ndbigtre
e))

tree_func(final_model = model_rf$finalModel, tree_num)
```



Or we can plot the tree with the biggest number of nodes:

```
tree_num <- which(model_rf$finalModel$forest$ndbigtree == max(model_rf$finalModel$forest$ndbigtre
e))

tree_func(final_model = model_rf$finalModel, tree_num)
```



Preparing the data and modeling

The data set I am using in these example analyses, is the **Breast Cancer Wisconsin (Diagnostic) Dataset**. The data was downloaded from the UC Irvine Machine Learning Repository (http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29).

The first data set looks at the predictor classes:

- malignant or
- benign breast mass.

The features characterize cell nucleus properties and were generated from image analysis of fine needle aspirates (FNA) (https://en.wikipedia.org/wiki/Fine-needle_aspiration) of breast masses:

- Sample ID (code number)
- Clump thickness
- Uniformity of cell size
- Uniformity of cell shape
- Marginal adhesion
- Single epithelial cell size
- Number of bare nuclei
- Bland chromatin
- Number of normal nuclei
- Mitosis
- Classes, i.e. diagnosis

```
bc_data <- read.table("datasets/breast-cancer-wisconsin.data.txt", header = FALSE, sep = ",")
colnames(bc_data) <- c("sample_code_number",
                       "clump_thickness",
                       "uniformity_of_cell_size",
                       "uniformity_of_cell_shape",
                       "marginal_adhesion",
                       "single_epithelial_cell_size",
                       "bare_nuclei",
                       "bland_chromatin",
                       "normal_nucleoli",
                       "mitosis",
                       "classes")

bc_data$classes <- ifelse(bc_data$classes == "2", "benign",
                          ifelse(bc_data$classes == "4", "malignant", NA))

bc_data[bc_data == "?"] <- NA

# impute missing data
library(mice)

bc_data[,2:10] <- apply(bc_data[, 2:10], 2, function(x) as.numeric(as.character(x)))
dataset_impute <- mice(bc_data[, 2:10],  print = FALSE)
bc_data <- cbind(bc_data[, 11, drop = FALSE], mice::complete(dataset_impute, 1))

bc_data$classes <- as.factor(bc_data$classes)

# how many benign and malignant cases are there?
summary(bc_data$classes)

# separate into training and test data
library(caret)

set.seed(42)
index <- createDataPartition(bc_data$classes, p = 0.7, list = FALSE)
train_data <- bc_data[index, ]
test_data  <- bc_data[-index, ]

# run model
set.seed(42)
model_rf <- caret::train(classes ~ .,
                         data = train_data,
                         method = "rf",
                         preProcess = c("scale", "center"),
                         trControl = trainControl(method = "repeatedcv",
                                                  number = 10,
                                                  repeats = 10,
                                                  savePredictions = TRUE,
                                                  verboseIter = FALSE))
```

If you are interested in more machine learning posts, check out the category listing for **machine_learning** on my blog (https://shiring.github.io/categories.html#machine_learning-ref).

```
sessionInfo()
```

```
## R version 3.3.3 (2017-03-06)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 7 x64 (build 7601) Service Pack 1
## 
## locale:
## [1] LC_COLLATE=English_United States.1252 
## [2] LC_CTYPE=English_United States.1252   
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C                          
## [5] LC_TIME=English_United States.1252    
## 
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base     
## 
## other attached packages:
## [1] igraph_1.0.1       ggraph_1.0.0       ggplot2_2.2.1.9000
## [4] dplyr_0.5.0       
## 
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.9        nloptr_1.0.4       plyr_1.8.4        
##  [4] viridis_0.3.4      iterators_1.0.8    tools_3.3.3       
##  [7] digest_0.6.12      lme4_1.1-12        evaluate_0.10     
## [10] tibble_1.2         gtable_0.2.0       nlme_3.1-131      
## [13] lattice_0.20-34    mgcv_1.8-17        Matrix_1.2-8      
## [16] foreach_1.4.3      DBI_0.6            ggrepel_0.6.5     
## [19] yaml_2.1.14        parallel_3.3.3     SparseM_1.76      
## [22] gridExtra_2.2.1    stringr_1.2.0      knitr_1.15.1      
## [25] MatrixModels_0.4-1 stats4_3.3.3       rprojroot_1.2     
## [28] grid_3.3.3         caret_6.0-73       nnet_7.3-12       
## [31] R6_2.2.0           rmarkdown_1.3      minqa_1.2.4       
## [34] udunits2_0.13      tweenr_0.1.5       deldir_0.1-12     
## [37] reshape2_1.4.2     car_2.1-4          magrittr_1.5      
## [40] units_0.4-2        backports_1.0.5    scales_0.4.1      
## [43] codetools_0.2-15   ModelMetrics_1.1.0 htmltools_0.3.5   
## [46] MASS_7.3-45        splines_3.3.3      randomForest_4.6-12
## [49] assertthat_0.1     pbkrtest_0.4-6     ggforce_0.1.1     
## [52] colorspace_1.3-2   labeling_0.3       quantreg_5.29     
## [55] stringi_1.1.2      lazyeval_0.2.0     munsell_0.4.3
```

machine_learning 14 (/categories.html#machine_learning-ref) ggplot2 ^{32} (/tags.html#ggplot2-ref) machine_learning ^{13} (/tags.html#machine_learning-ref) caret ^{3} (/tags.html#caret-ref) random_forest ^{6} (/tags.html#random_forest-ref) ggraph ^{1} (/tags.html#ggraph-ref) igraph ^{3} (/tags.html#igraph-ref)

« Previous (/machine_learning/2017/03/07/grid_search) Archive (/archive.html)

Next » (/machine_learning/2017/03/31/webinar_code)

40 Comments **shirinsplayground** 🔒 **Disqus' Privacy Policy** 1 **Login**

♡ **Recommend** 7 **Tweet** **f Share** **Sort by Best**

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Pieter Vos • 4 years ago

Hi Shirin,

Thank you for sharing this. I have a question though. If you have 500 trees in your random forest, do you visualize each of them to understand your model? How would you know this single decision tree is representative for the random forest, because the final decision is made on all trees (ensemble)?

thanks

Pieter

2 ^ | ˅ • Reply • Share ›

Shirin Elsinghorst Mod ➦ Pieter Vos • 4 years ago

Hi Pieter.
Thanks! :-)
No, I don't look at all trees but I generally pick a few to plot so that I get an idea of which decisions were made by the model. Really understanding a reasonably complex model is by definition almost impossible but by looking at a few trees you can get a good sense of which features are similar and how they were split.

^ | ˅ • Reply • Share ›

AJT • 2 years ago

Brilliant post. Thanks & super cool!

1 ^ | ˅ • Reply • Share ›

รัก อย่างไร้เงื่อนไข • 3 years ago

Thank you so much, It great to share the random forest it showed in the decision trees.

1 ^ | ˅ • Reply • Share ›

Glorious1 • 4 years ago • edited

Thank you for this. I have huge trees, with about 15,000 nodes (which I guess become vertices?). So I'm very interested in how you used delete_vertices(). I don't quite follow the comment " # . . . delete the last node that we don't want to plot".

1. Does this mean it should plot everything after the vertex you specify, but not before?

2. This doesn't seem to match the help for delete_vertices, which indicates you specify a 'sequence'

and error when I try to plot (Error in seq.default(h[1], h[2], length.out = n) :
'to' must be a finite number).
Thanks for any help.

1 ^ | ˅ • Reply • Share ›

Shirin Elsinghorst Mod ➦ Glorious1 • 4 years ago

I am not plotting the last row of vertices ("0") because I plotted the labels at their stead. As for you error messages, I can't really say without having seen your data and models...

^ | ˅ • Reply • Share ›

This comment was deleted.

Shirin Elsinghorst Mod ➦ Guest • 4 years ago

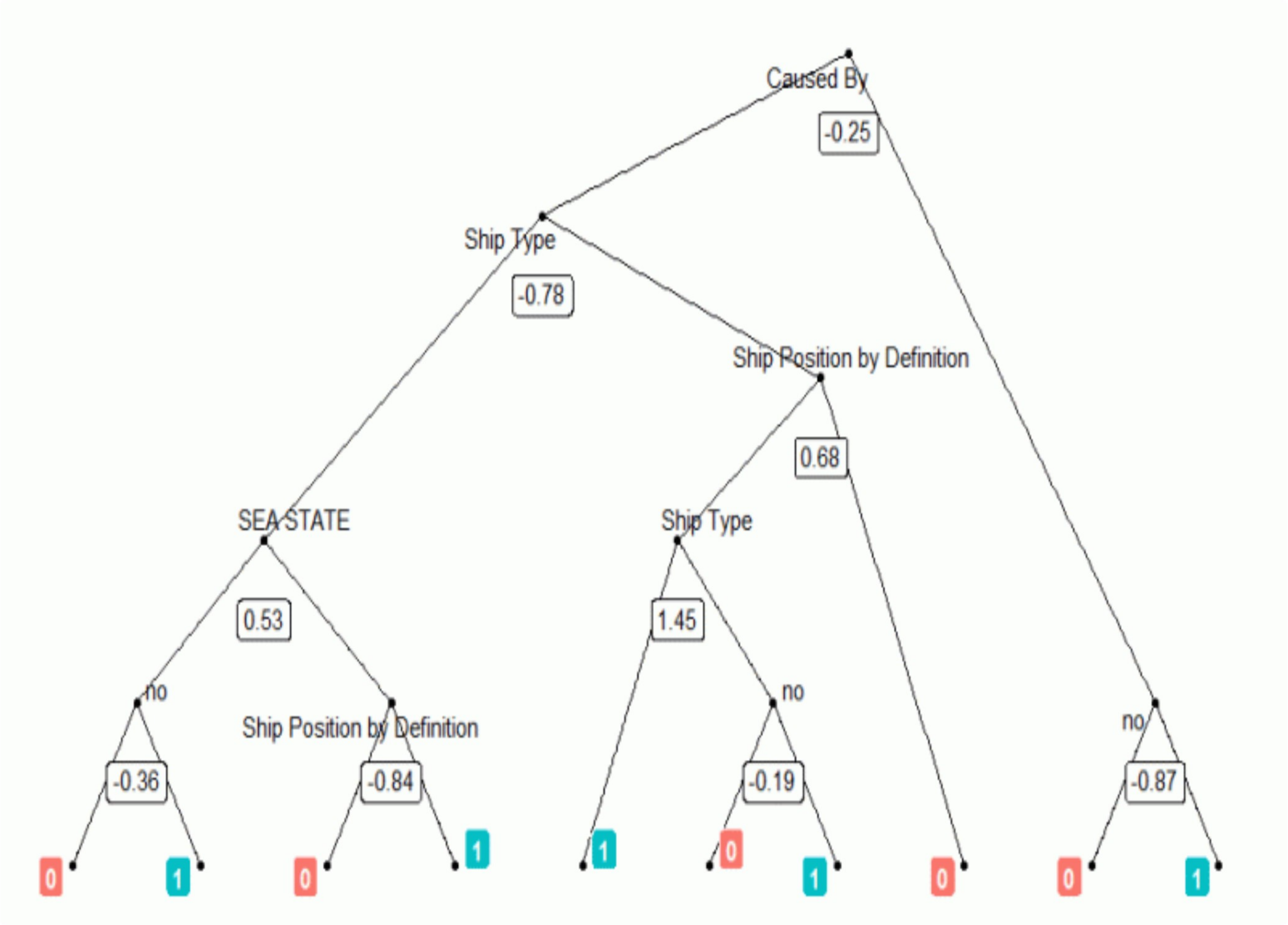
This is shown for the randomForest package...

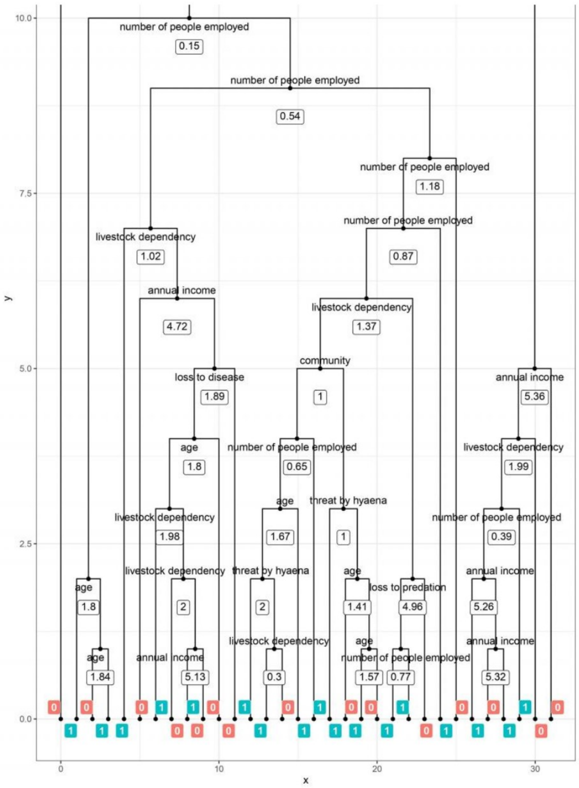
^ | ˅ • Reply • Share ›

GOKHAN CAMLIYURT • 2 months ago

Hi Shirin,

Thank oyu very much for sharing. Able to do a beatiful tree with little mistakes :)





^ | ⌄ • Reply • Share ›

Camilla Jensen • 2 years ago

Dear Shirin, I am using the code you have written for the min/max decision tree when using the Caret package in r and for random forest. It is really hard to find any other code and I have the problem (just a humble social scientist with very marginal coding skills) that I feel unsure about how to interpret the decision tree that I get from the code. Because in other decision trees we can see the math signs (>=< etc.) and also it is often written one way is yes, the other is no etc. But in the decision tree I get from your code - which is really excellent and aesthetic, there is the only problem that I am not 100% what the interpretations are. Could you possible rewrite the code so that we can see the signs on the decision-tree? Thank you so much for your attention! Many Christmas

⌃ | ⌄ • Reply • Share ›

This comment was marked as spam.

Shirin Elsinghorst **Mod** → JVD • 3 years ago

At which point in the function do you get this error?

⌃ | ⌄ • Reply • Share ›

JVD → Shirin Elsinghorst • 3 years ago • edited

EDIT:
I started everything again and somehow the error disappeared...

⌃ | ⌄ • Reply • Share ›

Shirin Elsinghorst **Mod** → JVD • 3 years ago

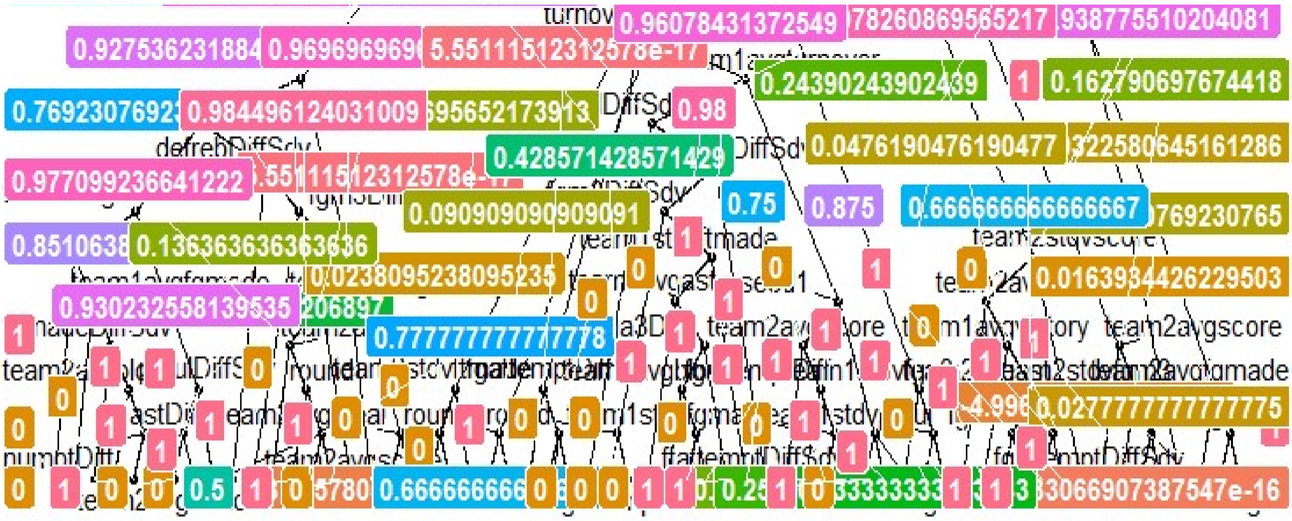
It's magic :-D

1 ⌃ | ⌄ • Reply • Share ›

Siladitya Sen • 4 years ago

Great! It worked!!!

⌃ | ⌄ • Reply • Share ›


turnov
0.96078431372549
78260869565217
938775510204081
0.927536231884
0.96969696969
5.55111512312578e-17
0.24390243902439
1
0.162790697674418
0.769230769
0.984496124031009
695652173913
DiffSd
0.98
defrebDiffSdv
0.428571428571429
DiffSd
0.0476190476190477
322580645161286
0.977099236641222
5.55111512312578e-17
0.0909090909090909
0.75
0.875
0.666666666666667
769230765
0.8510638
0.136363636363636
tmade
0.0238095238095235
0.016393442622950
0.930232558139535
206897
0.777777777777778
team2avgscore
team2avgscore
team2avgfgmade
4.996
0.0277777777777775
0.5
5780
0.666666666
0.25
33333333
3.33066907387547e-16
1
0

Thank you for the suggestion! I'll look into it. :-)

1 ^ | ⌄ • Reply • Share ›

Abhijit Dasgupta • 4 years ago

You might want to mention early in your code that you are using random forests via the caret train

/Shirin_Glander) (http://de.linkedin.com/in/shirin-glander-01120881) with help from Jekyll

Bootstrap (http://jekyllbootstrap.com) and Bootstrap (http://getbootstrap.com)