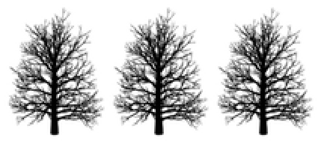

Completed • Knowledge • 1,694 teams

Forest Cover Type Prediction

Fri 16 May 2014 – Mon 11 May 2015 (4 months ago)

R - Caret - Using ROC instead of accuracy in model training

1

Hi my name is Abhi and I am using caret to build a gbm trees based model. However instead of accuracy I would like to use roc as my metric

Here is the code I have so far

myTuneGrid <- expand.grid(n.trees = 500,interaction.depth = 11,shrinkage = 0.1)

fitControl <- trainControl(method = "repeatedcv", number = 7,repeats = 1, verboseIter = FALSE,returnResamp = "all",classProbs = TRUE)

myModel <- train(Cover_Type ~ .,data = modelData,method = "gbm",trControl = fitControl,tuneGrid = myTuneGrid,metric='roc')

However when I run this code I get a warning

Warning message:
In train.default(x, y, weights = w, ...) :
The metric "roc" was not in the result set. Accuracy will be used instead.

How do I force my model to use roc instead of accuracy. What am I doing wrong here?

#1 | Posted 11 months ago

abhimanipal

3

Hi Abhi,

I think caret only supports the area under ROC metric for 2-class problems. This challenge has 7 classes.

Anyway, to use the ROC metric for 2-class problems:

You can specify a summaryFunction in trainControl. The summary function calculates metrics. The defaultSummary function only calculates accuracy and kappa, so you need to specify another summary function that calculates area under ROC. There is the twoClassSummary function, that computes sensitivity, specificity and the area under the ROC curve. So adding summaryFunction=twoClassSummary, classProbs=TRUE to the trainControl function should allow you to specify metric="ROC" in the train function. (But only for 2-class problems!)

There are ways to calculate area under ROC for multiple classes. You could write your own summary function that does that and pass it to trainControl.

#2 | Posted 11 months ago

stmax

Permalink | Quote | Flag

0

I found this sample floating on the internet. I am using this & it looks like this solves the problem

```
multiClassSummary <- cmpfun(function (data, lev = NULL, model = NULL){

#Load Libraries
require(Metrics)
require(caret)

#Check data
if (!all(levels(data[, "pred"]) == levels(data[, "obs"])))
stop("levels of observed and predicted data do not match")

#Calculate custom one-vs-all stats for each class
prob_stats <- lapply(levels(data[, "pred"]), function(class){

#Grab one-vs-all data for the class
pred <- ifelse(data[, "pred"] == class, 1, 0)
obs <- ifelse(data[, "obs"] == class, 1, 0)
prob <- data[,class]

#Calculate one-vs-all AUC and logLoss and return
cap_prob <- pmin(pmax(prob, .000001), .999999)
prob_stats <- c(auc(obs, prob), logLoss(obs, cap_prob))
names(prob_stats) <- c('ROC', 'logLoss')
return(prob_stats)
})
prob_stats <- do.call(rbind, prob_stats)
rownames(prob_stats) <- paste('Class:', levels(data[, "pred"]))

#Calculate confusion matrix-based statistics
CM <- confusionMatrix(data[, "pred"], data[, "obs"])

#Aggregate and average class-wise stats
#Todo: add weights
class_stats <- cbind(CM$byClass, prob_stats)
class_stats <- colMeans(class_stats)

#Aggregate overall stats
overall_stats <- c(CM$overall)

#Combine overall with class-wise stats and remove some stats we don't want
stats <- c(overall_stats, class_stats)
stats <- stats[! names(stats) %in% c('AccuracyNull',
'Prevalence', 'Detection Prevalence')]

#Clean names and return
names(stats) <- gsub('[[:blank:]]+', '_', names(stats))
return(stats)

})
```

#3 | Posted 11 months ago

abhimanipal

Reply

+ Add attachment(s) ...

Post Reply

[✓] Email me when someone replies

Start Watching « Back to forum

About Our Team Careers Terms Privacy Contact/Support