**Functions for extending random forest:**

***cforest\_crossValidate***(data, full.model ,foldVector,pred.names, response, formula,threshold,binaryresponse,fact)

SUMMARY:

This function is designed to evaluate the predictive performance of "cforest" models using cross-validation. This function only works for binary or continuous response variables. Some of the output is only available for the binary response variable case.

PARAMETERS:

* data
  + Data frame containing all response and predictor variables
* full.model
  + "cforest" object fitted with the entire data set (object returned from the function 'cforest' in the 'party' package)
* foldVector
  + Vector (length equal to number of rows in 'data') specifying which fold each observation belongs to. For 10-fold C-V, for example, this could be generated by sampling the vector c(1:10) with replacement. In other cases – especially those in which the observations are not independent -- a "blocking" factor may be used to define the fold vector. For example, spatially autocorrelated regions could be treated as "folds" and thereby left out of the model fitting process (and used as an out-of-bag validation set) one at a time. This strategy can result in a more honest appraisal of predictive performance.
* pred.names
  + Character vector (length equal to the number of predictor variables) specifying the names of the predictor variables. Note that the names must match the appropriate column names in 'data'.
* response
  + String representing the name of the response variable. Note that the name must match the appropriate column names in 'data'.
* formula
  + R formula object specifying the response and predictor variable (e.g., as.formula(a~b+c))
* threshold
  + [OPTIONAL] For binary response variables, a threshold above which predictions should be classified as "TRUE"
* binaryresponse
  + Logical value indicating whether (TRUE) or not (FALSE) the response variable is binary.
* fact
  + Logical value indicating whether (TRUE) or not (FALSE) the response variable is coded as a factor variable in R. NOTE: if you have a binary response (binary), it should be coded as a factor prior to running this function.

OUTPUT

A list containing a set of performance metrics:

* CV\_RMSE
  + RMSE for C-V (out of bag) data
* real\_RMSE
  + RMSE for training (in bag) data
* [binary response only] CV\_auc
  + area under the ROC curve for C-V data
* [binary response only] real\_auc
  + area under the ROC curve for training data
* [binary response only] CV\_maxkappa
  + for binary response data, the most informative threshold, determined using the Cohen Kappa statistic (threshold that maximizes the Kappa statistic) .
* [binary response only] confusion.mat
  + confusion matrix given a set threshold (if none is specified, then the maxkappa threshold is used).
* [binary response only] sensitivity
  + true positive rate
* [binary response only] specificity
  + true negative rate
* [binary response only] toterror
  + total error rate
* deviance\_explained\_CV
  + McFadden's pseudo R-squared for C-V data
* deviance\_explained\_real
  + McFadden's pseudo R-squared for training data

***RF\_UnivariatePlots***(object, varimp, data, predictors, labels, allpredictors,plot.layout,plot)

SUMMARY

Based on the function 'gbm.plot' from the 'dismo' package. Function to plot random forest response variables. These are partial dependence plots, whereby all predictor variables are held constant at their mean value except for one focal variable. Note that fitted functions are centered by subtracting their mean.

PARAMETERS

* object
  + "cforest" object fitted with the entire data set (object returned from the function 'cforest' in the 'party' package)
* varimp
  + Object returned from the function 'varimp' in the 'party' package
* data
  + Data frame containing all response and predictor variables
* predictors
  + Character vector specifying the names of the predictor variables to plot. Note that the names must match the appropriate column names in 'data'.
* labels
  + Character vector (length equal to the number of predictor variables) specifying names of the predictor variables to plot. Note that the names must be in the same order as 'predictors'.
* allpredictors
  + Character vector (length equal to the total number of predictor variables in the model) specifying the names of all predictor variables. Note that the names must match the appropriate column names in 'data'.
* plot.layout
  + A vector of the form ‘c(nrow, ncol)’. Figures will be drawn in an ‘nrow’-by-‘ncol’ array on the device.
* plot
  + A logical value indicating whether (TRUE) or not (FALSE) to produce partial dependence plots of the despredictor variables. If FALSE, the function will produce a data frame containing raw data that may be plotted in a separate R function or exported for plotting in another software (such as SigmaPlot).

OUTPUT

Depending on the value of the 'plot' variable, this will either produce plots, or a a data frame containing partial dependencies. If plot=FALSE, the data frame produced will have a column for each predictor variable.

***RF\_FindInteractions***(object,data,predictors)

SUMMARY

Based on the function ' gbm.interactions' in the 'dismo' package. This function assesses the strength of interaction for all possible pairs of predictor variables. The function assesses the magnitude of 2nd order interaction effects in random forest models. This is achieved by: (1) forming predictions for each predictor pair across a wide gradient; (2) fitting an additive linear model that relates these predictions to the predictor pair, with the the predictors fitted as factors; (3) calculating the mean value of the residuals, the magnitude ofwhich increases with the strength of any interaction effect; (4) results are stored in an array; (5). finally, the most important interactions are identified.

PARAMETERS

* object
  + "cforest" object fitted with the entire data set (object returned from the function 'cforest' in the 'party' package)
* data
  + Data frame containing all response and predictor variables
* predictors
  + Character vector specifying the names of the predictor variables to plot. Note that the names must match the appropriate column names in 'data'.

OUTPUT

List with four elements:

* interactions1
  + Table of nPredictorsXnPredictors where each element indicates an index of interaction computed using root mean squared residual error.
* interactions2
  + Table of nPredictorsXnPredictors where each element indicates an index of interaction computed using relative entropy.
* rank.list1
  + List of the top interactions from 'interactions1'. Note that this list (and rank.list2) is just a rough attempt to identify the top interactions. The full set of interactions (e.g., interactions1) should be examined for interactions that may have been missed.
* rank.list2
  + List of the top interactions from 'interactions2'.

***RF\_InteractionPlots***(x,y,object,data,predictors,family)

SUMMARY

Based on the 'gbm.perspec' function from the 'dismo' package. Takes a random forest object produced by 'party' and plots a perspective plot showing predicted values for two predictors as specified by number using x and y. Values for all other variables are set at their mean by default.

PARAMETERS

* x
  + index of the first predictor variable of the interacting pair(refer to the "rank.list" produced by the RF\_FindInteractions function)
* y
  + index of the second predictor variable of the interacting pair(refer to the "rank.list" produced by the RF\_FindInteractions function)
* object
  + "cforest" object fitted with the entire data set (object returned from the function 'cforest' in the 'party' package)
* data
  + Data frame containing all response and predictor variables
* predictors
  + Character vector (length equal to the total number of predictor variables in the model) specifying the names of all predictor variables. Note that the names must match the appropriate column names in 'data'.
* family
  + distribution family of response variable, analogous to a glm model. Must be either 'gaussian', 'bernoulli', or 'poisson'

OUTPUT

Perspective plot for visualizing a bivariate interaction.

Based on the 'gbm.perspec' function in the 'dismo' package.