7 `train` Models By Tag

The following is a basic list of model types or relevant characteristics. There entires in these lists are arguable. For example: random forests theoretically use feature selection but effectively may not, support vector machines use L2 regularization etc.

Contents

- Kernel Method
- L1 Regularization
- L2 Regularization
- Linear Classifier
- Linear Regression
- Logic Regression
- Logistic Regression
- Mixture Model
- Model Tree
- Multivariate Adaptive Regression Splines
- Neural Network
- Oblique Tree
- Ordinal Outcomes
- Partial Least Squares
- Patient Rule Induction Method
- Polynomial Model
- Prototype Models
- Quantile Regression
- Radial Basis Function
- Random Forest
- Regularization
- Relevance Vector Machines
- Ridge Regression
- Robust Methods
- Robust Model
- ROC Curves
- Rule-Based Model
- Self-Organising Maps

- String Kernel
- Support Vector Machines
- Supports Class Probabilities
- Text Mining
- Tree-Based Model
- Two Class Only

7.0.1 Accepts Case Weights

(back to contents)

Adjacent Categories Probability Model for Ordinal Data

```
method = 'vglmAdjCat'
```

Type: Classification

Tuning parameters:

- `parallel` (Parallel Curves)
- `link` (Link Function)

Required packages: `VGAM`

Bagged CART

```
method = 'treebag'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `ipred` , `plyr` , `e1071`

A model-specific variable importance metric is available.

Bagged Flexible Discriminant Analysis

```
method = 'bagFDA'
```

Type: Classification

Tuning parameters:

- `degree` (Product Degree)
- `nprune` (#Terms)

Required packages: `earth` , `mda`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train` , the `earth` package is fully loaded when this model is used.

Bagged MARS

```
method = 'bagEarth'
```

Type: Regression, Classification

Tuning parameters:

- `nprune` (#Terms)

- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Bagged MARS using gCV Pruning

```
method = 'bagEarthGCV'
```

Type: Regression, Classification

Tuning parameters:

- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Bayesian Generalized Linear Model

```
method = 'bayesglm'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `arm`

Boosted Generalized Additive Model

```
method = 'gamboost'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `prune` (AIC Prune?)

Required packages: `mboost` , `plyr` , `import`

Notes: The `prune` option for this model enables the number of iterations to be determined by the optimal AIC value across all iterations. See the examples in `?mboost::mstop` . If pruning is not used, the ensemble makes predictions using the exact value of the `mstop` tuning parameter value.

Boosted Generalized Linear Model

```
method = 'glmboost'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `prune` (AIC Prune?)

Required packages: `plyr` , `mboost`

A model-specific variable importance metric is available. Notes: The `prune` option for this model enables the number of iterations to be determined by the optimal AIC value across all iterations. See the examples in `?mboost::mstop` . If pruning is not used, the ensemble makes predictions using the exact value of the `mstop` tuning parameter value.

Boosted Tree

```
method = 'blackboost'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (#Trees)
- `maxdepth` (Max Tree Depth)

Required packages: `party` , `mboost` , `plyr`

C5.0

```
method = 'C5.0'
```

Type: Classification

Tuning parameters:

- `trials` (# Boosting Iterations)

- `model` (Model Type)
- `winnow` (Winnow)

Required packages: `C50` , `plyr`

A model-specific variable importance metric is available.

CART

```
method = 'rpart'
```

Type: Regression, Classification

Tuning parameters:

- `cp` (Complexity Parameter)

Required packages: `rpart`

A model-specific variable importance metric is available.

CART

```
method = 'rpart1SE'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `rpart`

A model-specific variable importance metric is available. Notes: This CART model replicates the same process used by the `rpart` function where the model complexity is determined using the one-standard error method. This procedure is replicated inside of the resampling done by `train` so that an external resampling estimate can be obtained.

CART

```
method = 'rpart2'
```

Type: Regression, Classification

Tuning parameters:

- `maxdepth` (Max Tree Depth)

Required packages: `rpart`

A model-specific variable importance metric is available.

CART or Ordinal Responses

```
method = 'rpartScore'
```

Type: Classification

Tuning parameters:

- `cp` (Complexity Parameter)
- `split` (Split Function)

- `prune` (Pruning Measure)

Required packages: `rpartScore` , `plyr`

A model-specific variable importance metric is available.

CHi-squared Automated Interaction Detection

```
method = 'chaid'
```

Type: Classification

Tuning parameters:

- `alpha2` (Merging Threshold)
- `alpha3` (Splitting former Merged Threshold)
- `alpha4` (Splitting former Merged Threshold)

Required packages: `CHAID`

Conditional Inference Random Forest

```
method = 'cforest'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `party`

A model-specific variable importance metric is available.

Conditional Inference Tree

```
method = 'ctree'
```

Type: Classification, Regression

Tuning parameters:

- `mincriterion` (1 - P-Value Threshold)

Required packages: `party`

Conditional Inference Tree

```
method = 'ctree2'
```

Type: Regression, Classification

Tuning parameters:

- `maxdepth` (Max Tree Depth)
- `mincriterion` (1 - P-Value Threshold)

Required packages: `party`

Continuation Ratio Model for Ordinal Data

```
method = 'vglmContRatio'
```

Type: Classification

Tuning parameters:

- `parallel` (Parallel Curves)
- `link` (Link Function)

Required packages: `VGAM`

Cost-Sensitive C5.0

```
method = 'C5.0Cost'
```

Type: Classification

Tuning parameters:

- `trials` (# Boosting Iterations)
- `model` (Model Type)
- `winnow` (Winnow)
- `cost` (Cost)

Required packages: `C50` , `plyr`

A model-specific variable importance metric is available.

Cost-Sensitive CART

```
method = 'rpartCost'
```

Type: Classification

Tuning parameters:

- `cp` (Complexity Parameter)
- `Cost` (Cost)

Required packages: `rpart` , `plyr`

Cumulative Probability Model for Ordinal Data

```
method = 'vglmCumulative'
```

Type: Classification

Tuning parameters:

- `parallel` (Parallel Curves)
- `link` (Link Function)

Required packages: `VGAM`

DeepBoost

```
method = 'deepboost'
```

Type: Classification

Tuning parameters:

- `num_iter` (# Boosting Iterations)
- `tree_depth` (Tree Depth)
- `beta` (L1 Regularization)
- `lambda` (Tree Depth Regularization)
- `loss_type` (Loss)

Required packages: `deepboost`

eXtreme Gradient Boosting

```
method = 'xgbDART'
```

Type: Regression, Classification

Tuning parameters:

- `nrounds` (# Boosting Iterations)
- `max_depth` (Max Tree Depth)
- `eta` (Shrinkage)
- `gamma` (Minimum Loss Reduction)
- `subsample` (Subsample Percentage)
- `colsample_bytree` (Subsample Ratio of Columns)
- `rate_drop` (Fraction of Trees Dropped)
- `skip_drop` (Prob. of Skipping Drop-out)
- `min_child_weight` (Minimum Sum of Instance Weight)

Required packages: `xgboost` , `plyr`

A model-specific variable importance metric is available.

eXtreme Gradient Boosting

```
method = 'xgbTree'
```

Type: Regression, Classification

Tuning parameters:

- `nrounds` (# Boosting Iterations)
- `max_depth` (Max Tree Depth)
- `eta` (Shrinkage)
- `gamma` (Minimum Loss Reduction)
- `colsample_bytree` (Subsample Ratio of Columns)
- `min_child_weight` (Minimum Sum of Instance Weight)
- `subsample` (Subsample Percentage)

Required packages: `xgboost` , `plyr`

A model-specific variable importance metric is available.

Flexible Discriminant Analysis

```
method = 'fda'
```

Type: Classification

Tuning parameters:

- `degree` (Product Degree)
- `nprune` (#Terms)

Required packages: `earth` , `mda`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train` , the `earth` package is fully loaded when this model is used.

Generalized Linear Model

```
method = 'glm'
```

Type: Regression, Classification

No tuning parameters for this model

A model-specific variable importance metric is available.

Generalized Linear Model with Stepwise Feature Selection

```
method = 'glmStepAIC'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `MASS`

Linear Regression

```
method = 'lm'
```

Type: Regression

Tuning parameters:

- `intercept` (intercept)

A model-specific variable importance metric is available.

Linear Regression with Stepwise Selection

```
method = 'lmStepAIC'
```

Type: Regression

No tuning parameters for this model

Required packages: `MASS`

Model Averaged Neural Network

```
method = 'avNNet'
```

Type: Classification, Regression

Tuning parameters:

- `size` (#Hidden Units)
- `decay` (Weight Decay)
- `bag` (Bagging)

Required packages: `nnet`

Multivariate Adaptive Regression Spline

```
method = 'earth'
```

Type: Regression, Classification

Tuning parameters:

- `nprune` (#Terms)

- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Multivariate Adaptive Regression Splines

```
method = 'gcvEarth'
```

Type: Regression, Classification

Tuning parameters:

- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Negative Binomial Generalized Linear Model

```
method = 'glm.nb'
```

Type: Regression

Tuning parameters:

- `link` (Link Function)

Required packages: `MASS`

A model-specific variable importance metric is available.

Neural Network

```
method = 'nnet'
```

Type: Classification, Regression

Tuning parameters:

- `size` (#Hidden Units)
- `decay` (Weight Decay)

Required packages: `nnet`

A model-specific variable importance metric is available.

Neural Networks with Feature Extraction

```
method = 'pcaNNet'
```

Type: Classification, Regression

Tuning parameters:

- `size` (#Hidden Units)
- `decay` (Weight Decay)

Required packages: `nnet`

Ordered Logistic or Probit Regression

```
method = 'polr'
```

Type: Classification

Tuning parameters:

- `method` (parameter)

Required packages: `MASS`

A model-specific variable importance metric is available.

Penalized Discriminant Analysis

```
method = 'pda'
```

Type: Classification

Tuning parameters:

- `lambda` (Shrinkage Penalty Coefficient)

Required packages: `mda`

Penalized Discriminant Analysis

```
method = 'pda2'
```

Type: Classification

Tuning parameters:

- `df` (Degrees of Freedom)

Required packages: `mda`

Penalized Multinomial Regression

```
method = 'multinom'
```

Type: Classification

Tuning parameters:

- `decay` (Weight Decay)

Required packages: `nnet`

A model-specific variable importance metric is available.

Projection Pursuit Regression

```
method = 'ppr'
```

Type: Regression

Tuning parameters:

- `nterms` (# Terms)

Random Forest

```
method = 'ranger'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `splitrule` (Splitting Rule)
- `min.node.size` (Minimal Node Size)

Required packages: `e1071` , `ranger` , `dplyr`

A model-specific variable importance metric is available.

Robust Linear Model

```
method = 'rlm'
```

Type: Regression

Tuning parameters:

- `intercept` (intercept)
- `psi` (psi)

Required packages: `MASS`

A model-specific variable importance metric is available.

Single C5.0 Ruleset

```
method = 'C5.0Rules'
```

Type: Classification

No tuning parameters for this model

Required packages: `C50`

A model-specific variable importance metric is available.

Single C5.0 Tree

```
method = 'C5.0Tree'
```

Type: Classification

No tuning parameters for this model

Required packages: `C50`

A model-specific variable importance metric is available.

Stochastic Gradient Boosting

```
method = 'gbm'
```

Type: Regression, Classification

Tuning parameters:

- `n.trees` (# Boosting Iterations)

- `interaction.depth` (Max Tree Depth)
- `shrinkage` (Shrinkage)
- `n.minobsinnode` (Min. Terminal Node Size)

Required packages: `gbm` , `plyr`

A model-specific variable importance metric is available.

Tree Models from Genetic Algorithms

```
method = 'evtree'
```

Type: Regression, Classification

Tuning parameters:

- `alpha` (Complexity Parameter)

Required packages: `evtree`

7.0.2 Bagging

(back to contents)

Bagged AdaBoost

```
method = 'AdaBag'
```

Type: Classification

Tuning parameters:

- `mfinal` (#Trees)
- `maxdepth` (Max Tree Depth)

Required packages: `adabag`, `plyr`

A model-specific variable importance metric is available.

Bagged CART

```
method = 'treebag'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `ipred`, `plyr`, `e1071`

A model-specific variable importance metric is available.

Bagged Flexible Discriminant Analysis

```
method = 'bagFDA'
```

Type: Classification

Tuning parameters:

- `degree` (Product Degree)
- `nprune` (#Terms)

Required packages: `earth`, `mda`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Bagged Logic Regression

```
method = 'logicBag'
```

Type: Regression, Classification

Tuning parameters:

- `nleaves` (Maximum Number of Leaves)
- `ntrees` (Number of Trees)

Required packages: `logicFS`

Notes: Unlike other packages used by `train`, the `logicFS` package is fully loaded when this model is used.

Bagged MARS

```
method = 'bagEarth'
```

Type: Regression, Classification

Tuning parameters:

- `nprune` (#Terms)
- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Bagged MARS using gCV Pruning

```
method = 'bagEarthGCV'
```

Type: Regression, Classification

Tuning parameters:

- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Bagged Model

```
method = 'bag'
```

Type: Regression, Classification

Tuning parameters:

- `vars` (#Randomly Selected Predictors)

Required packages: `caret`

Conditional Inference Random Forest

```
method = 'cforest'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `party`

A model-specific variable importance metric is available.

Ensembles of Generalized Linear Models

```
method = 'randomGLM'
```

Type: Regression, Classification

Tuning parameters:

- `maxInteractionOrder` (Interaction Order)

Required packages: `randomGLM`

Notes: Unlike other packages used by `train` , the `randomGLM` package is fully loaded when this model is used.

Model Averaged Neural Network

```
method = 'avNNet'
```

Type: Classification, Regression

Tuning parameters:

- `size` (#Hidden Units)
- `decay` (Weight Decay)
- `bag` (Bagging)

Required packages: `nnet`

Parallel Random Forest

```
method = 'parRF'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `e1071` , `randomForest` , `foreach` , `import`

A model-specific variable importance metric is available.

Quantile Random Forest

```
method = 'qrf'
```

Type: Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `quantregForest`

Quantile Regression Neural Network

```
method = 'qrnn'
```

Type: Regression

Tuning parameters:

- `n.hidden` (#Hidden Units)
- `penalty` (Weight Decay)
- `bag` (Bagged Models?)

Required packages: `qrnn`

Random Ferns

```
method = 'rFerns'
```

Type: Classification

Tuning parameters:

- `depth` (Fern Depth)

Required packages: `rFerns`

Random Forest

```
method = 'ranger'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `splitrule` (Splitting Rule)
- `min.node.size` (Minimal Node Size)

Required packages: `e1071` , `ranger` , `dplyr`

A model-specific variable importance metric is available.

Random Forest

```
method = 'Rborist'
```

Type: Classification, Regression

Tuning parameters:

- `predFixed` (#Randomly Selected Predictors)
- `minNode` (Minimal Node Size)

Required packages: `Rborist`

A model-specific variable importance metric is available.

Random Forest

```
method = 'rf'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `randomForest`

A model-specific variable importance metric is available.

Random Forest by Randomization

```
method = 'extraTrees'
```

Type: Regression, Classification

Tuning parameters:

- `mtry` (# Randomly Selected Predictors)
- `numRandomCuts` (# Random Cuts)

Required packages: `extraTrees`

Random Forest Rule-Based Model

```
method = 'rfRules'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `maxdepth` (Maximum Rule Depth)

Required packages: `randomForest` , `inTrees` , `plyr`

A model-specific variable importance metric is available.

Regularized Random Forest

```
method = 'RRF'
```

Type: Regression, Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `coefReg` (Regularization Value)
- `coefImp` (Importance Coefficient)

Required packages: `randomForest` , `RRF`

A model-specific variable importance metric is available.

Regularized Random Forest

```
method = 'RRFglobal'
```

Type: Regression, Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `coefReg` (Regularization Value)

Required packages: `RRF`

A model-specific variable importance metric is available.

Weighted Subspace Random Forest

```
method = 'wsrf'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `wsrf`

7.0.3 Bayesian Model

(back to contents)

Bayesian Additive Regression Trees

```
method = 'bartMachine'
```

Type: Classification, Regression

Tuning parameters:

- `num_trees` (#Trees)
- `k` (Prior Boundary)
- `alpha` (Base Terminal Node Hyperparameter)
- `beta` (Power Terminal Node Hyperparameter)
- `nu` (Degrees of Freedom)

Required packages: `bartMachine`

A model-specific variable importance metric is available.

Bayesian Generalized Linear Model

```
method = 'bayesglm'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `arm`

Bayesian Regularized Neural Networks

```
method = 'brnn'
```

Type: Regression

Tuning parameters:

- `neurons` (# Neurons)

Required packages: `brnn`

Bayesian Ridge Regression

```
method = 'bridge'
```

Type: Regression

No tuning parameters for this model

Required packages: `monomvn`

Bayesian Ridge Regression (Model Averaged)

```
method = 'blassoAveraged'
```

Type: Regression

No tuning parameters for this model

Required packages: `monomvn`

Notes: This model makes predictions by averaging the predictions based on the posterior estimates of the regression coefficients. While it is possible that some of these posterior estimates are zero for non-informative predictors, the final predicted value may be a function of many (or even all) predictors.

Model Averaged Naive Bayes Classifier

```
method = 'manb'
```

Type: Classification

Tuning parameters:

- `smooth` (Smoothing Parameter)
- `prior` (Prior Probability)

Required packages: `bnclassify`

Naive Bayes

```
method = 'naive_bayes'
```

Type: Classification

Tuning parameters:

- `laplace` (Laplace Correction)
- `usekernel` (Distribution Type)
- `adjust` (Bandwidth Adjustment)

Required packages: `naivebayes`

Naive Bayes

```
method = 'nb'
```

Type: Classification

Tuning parameters:

- `fL` (Laplace Correction)
- `usekernel` (Distribution Type)
- `adjust` (Bandwidth Adjustment)

Required packages: `klaR`

Naive Bayes Classifier

```
method = 'nbDiscrete'
```

Type: Classification

Tuning parameters:

- `smooth` (Smoothing Parameter)

Required packages: `bnclassify`

Naive Bayes Classifier with Attribute Weighting

```
method = 'awnb'
```

Type: Classification

Tuning parameters:

- `smooth` (Smoothing Parameter)

Required packages: `bnclassify`

Semi-Naive Structure Learner Wrapper

```
method = 'nbSearch'
```

Type: Classification

Tuning parameters:

- `k` (#Folds)
- `epsilon` (Minimum Absolute Improvement)
- `smooth` (Smoothing Parameter)
- `final_smooth` (Final Smoothing Parameter)

- `direction` (Search Direction)

Required packages: `bnclassify`

Spike and Slab Regression

```
method = 'spikeslab'
```

Type: Regression

Tuning parameters:

- `vars` (Variables Retained)

Required packages: `spikeslab` , `plyr`

Notes: Unlike other packages used by `train` , the `spikeslab` package is fully loaded when this model is used.

The Bayesian lasso

```
method = 'blasso'
```

Type: Regression

Tuning parameters:

- `sparsity` (Sparsity Threshold)

Required packages: `monomvn`

Notes: This model creates predictions using the mean of the posterior distributions but sets some parameters specifically to zero based on the tuning parameter `sparsity`. For example, when `sparsity = .5`, only coefficients where at least half the posterior estimates are nonzero are used.

Tree Augmented Naive Bayes Classifier

```
method = 'tan'
```

Type: Classification

Tuning parameters:

- `score` (Score Function)
- `smooth` (Smoothing Parameter)

Required packages: `bnclassify`

Tree Augmented Naive Bayes Classifier Structure Learner Wrapper

```
method = 'tanSearch'
```

Type: Classification

Tuning parameters:

- `k` (#Folds)
- `epsilon` (Minimum Absolute Improvement)
- `smooth` (Smoothing Parameter)

- `final_smooth` (Final Smoothing Parameter)
- `sp` (Super-Parent)

Required packages: `bnclassify`

Tree Augmented Naive Bayes Classifier with Attribute Weighting

```
method = 'awtan'
```

Type: Classification

Tuning parameters:

- `score` (Score Function)
- `smooth` (Smoothing Parameter)

Required packages: `bnclassify`

Variational Bayesian Multinomial Probit Regression

```
method = 'vbmpRadial'
```

Type: Classification

Tuning parameters:

- `estimateTheta` (Theta Estimated)

Required packages: `vbmp`

7.0.4 Binary Predictors Only

(back to contents)

Bagged Logic Regression

```
method = 'logicBag'
```

Type: Regression, Classification

Tuning parameters:

- `nleaves` (Maximum Number of Leaves)
- `ntrees` (Number of Trees)

Required packages: `logicFS`

Notes: Unlike other packages used by `train`, the `logicFS` package is fully loaded when this model is used.

Binary Discriminant Analysis

```
method = 'binda'
```

Type: Classification

Tuning parameters:

- `lambda.freqs` (Shrinkage Intensity)

Required packages: `binda`

Logic Regression

```
method = 'logreg'
```

Type: Regression, Classification

Tuning parameters:

- `treesize` (Maximum Number of Leaves)
- `ntrees` (Number of Trees)

Required packages: `LogicReg`

7.0.5 Boosting

(back to contents)

AdaBoost Classification Trees

```
method = 'adaboost'
```

Type: Classification

Tuning parameters:

- `nIter` (#Trees)
- `method` (Method)

Required packages: `fastAdaboost`

AdaBoost.M1

```
method = 'AdaBoost.M1'
```

Type: Classification

Tuning parameters:

- `mfinal` (#Trees)
- `maxdepth` (Max Tree Depth)
- `coeflearn` (Coefficient Type)

Required packages: `adabag`, `plyr`

A model-specific variable importance metric is available.

Bagged AdaBoost

```
method = 'AdaBag'
```

Type: Classification

Tuning parameters:

- `mfinal` (#Trees)
- `maxdepth` (Max Tree Depth)

Required packages: `adabag`, `plyr`

A model-specific variable importance metric is available.

Boosted Classification Trees

```
method = 'ada'
```

Type: Classification

Tuning parameters:

- `iter` (#Trees)
- `maxdepth` (Max Tree Depth)
- `nu` (Learning Rate)

Required packages: `ada` , `plyr`

Boosted Generalized Additive Model

```
method = 'gamboost'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `prune` (AIC Prune?)

Required packages: `mboost` , `plyr` , `import`

Notes: The `prune` option for this model enables the number of iterations to be determined by the optimal AIC value across all iterations. See the examples in `?mboost::mstop` . If pruning is not used, the ensemble makes predictions using the exact value of the `mstop` tuning parameter value.

Boosted Generalized Linear Model

```
method = 'glmboost'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `prune` (AIC Prune?)

Required packages: `plyr` , `mboost`

A model-specific variable importance metric is available. Notes: The `prune` option for this model enables the number of iterations to be determined by the optimal AIC value across all iterations. See the examples in `?mboost::mstop` . If pruning is not used, the ensemble makes predictions using the exact value of the `mstop` tuning parameter value.

Boosted Linear Model

```
method = 'BstLm'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `nu` (Shrinkage)

Required packages: `bst` , `plyr`

Boosted Logistic Regression

```
method = 'LogitBoost'
```

Type: Classification

Tuning parameters:

- `nIter` (# Boosting Iterations)

Required packages: `caTools`

Boosted Smoothing Spline

```
method = 'bstSm'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `nu` (Shrinkage)

Required packages: `bst` , `plyr`

Boosted Tree

```
method = 'blackboost'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (#Trees)
- `maxdepth` (Max Tree Depth)

Required packages: `party` , `mboost` , `plyr`

Boosted Tree

```
method = 'bstTree'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `maxdepth` (Max Tree Depth)
- `nu` (Shrinkage)

Required packages: `bst` , `plyr`

C5.0

```
method = 'C5.0'
```

Type: Classification

Tuning parameters:

- `trials` (# Boosting Iterations)

- `model` (Model Type)
- `winnow` (Winnow)

Required packages: `C50` , `plyr`

A model-specific variable importance metric is available.

Cost-Sensitive C5.0

```
method = 'C5.0Cost'
```

Type: Classification

Tuning parameters:

- `trials` (# Boosting Iterations)
- `model` (Model Type)
- `winnow` (Winnow)
- `cost` (Cost)

Required packages: `C50` , `plyr`

A model-specific variable importance metric is available.

Cubist

```
method = 'cubist'
```

Type: Regression

Tuning parameters:

- committees (#Committees)
- neighbors (#Instances)

Required packages: Cubist

A model-specific variable importance metric is available.

DeepBoost

```
method = 'deepboost'
```

Type: Classification

Tuning parameters:

- num_iter (# Boosting Iterations)
- tree_depth (Tree Depth)
- beta (L1 Regularization)
- lambda (Tree Depth Regularization)
- loss_type (Loss)

Required packages: deepboost

eXtreme Gradient Boosting

```
method = 'xgbDART'
```

Type: Regression, Classification

Tuning parameters:

- nrounds (# Boosting Iterations)

- `max_depth` (Max Tree Depth)
- `eta` (Shrinkage)
- `gamma` (Minimum Loss Reduction)
- `subsample` (Subsample Percentage)
- `colsample_bytree` (Subsample Ratio of Columns)
- `rate_drop` (Fraction of Trees Dropped)
- `skip_drop` (Prob. of Skipping Drop-out)
- `min_child_weight` (Minimum Sum of Instance Weight)

Required packages: `xgboost` , `plyr`

A model-specific variable importance metric is available.

eXtreme Gradient Boosting

```
method = 'xgbLinear'
```

Type: Regression, Classification

Tuning parameters:

- `nrounds` (# Boosting Iterations)
- `lambda` (L2 Regularization)
- `alpha` (L1 Regularization)
- `eta` (Learning Rate)

Required packages: `xgboost`

A model-specific variable importance metric is available.

eXtreme Gradient Boosting

```
method = 'xgbTree'
```

Type: Regression, Classification

Tuning parameters:

- `nrounds` (# Boosting Iterations)
- `max_depth` (Max Tree Depth)
- `eta` (Shrinkage)
- `gamma` (Minimum Loss Reduction)
- `colsample_bytree` (Subsample Ratio of Columns)
- `min_child_weight` (Minimum Sum of Instance Weight)
- `subsample` (Subsample Percentage)

Required packages: `xgboost` , `plyr`

A model-specific variable importance metric is available.

Gradient Boosting Machines

```
method = 'gbm_h2o'
```

Type: Regression, Classification

Tuning parameters:

- `ntrees` (# Boosting Iterations)
- `max_depth` (Max Tree Depth)
- `min_rows` (Min. Terminal Node Size)
- `learn_rate` (Shrinkage)

- `col_sample_rate` (#Randomly Selected Predictors)

Required packages: `h2o`

A model-specific variable importance metric is available.

Stochastic Gradient Boosting

```
method = 'gbm'
```

Type: Regression, Classification

Tuning parameters:

- `n.trees` (# Boosting Iterations)
- `interaction.depth` (Max Tree Depth)
- `shrinkage` (Shrinkage)
- `n.minobsinnode` (Min. Terminal Node Size)

Required packages: `gbm` , `plyr`

A model-specific variable importance metric is available.

7.0.6 Categorical Predictors Only

(back to contents)

Model Averaged Naive Bayes Classifier

```
method = 'manb'
```

Type: Classification

Tuning parameters:

- `smooth` (Smoothing Parameter)
- `prior` (Prior Probability)

Required packages: `bnclassify`

Naive Bayes Classifier

```
method = 'nbDiscrete'
```

Type: Classification

Tuning parameters:

- `smooth` (Smoothing Parameter)

Required packages: `bnclassify`

Naive Bayes Classifier with Attribute Weighting

```
method = 'awnb'
```

Type: Classification

Tuning parameters:

- `smooth` (Smoothing Parameter)

Required packages: `bnclassify`

Semi-Naive Structure Learner Wrapper

```
method = 'nbSearch'
```

Type: Classification

Tuning parameters:

- `k` (#Folds)
- `epsilon` (Minimum Absolute Improvement)
- `smooth` (Smoothing Parameter)
- `final_smooth` (Final Smoothing Parameter)
- `direction` (Search Direction)

Required packages: `bnclassify`

Tree Augmented Naive Bayes Classifier

```
method = 'tan'
```

Type: Classification

Tuning parameters:

- `score` (Score Function)
- `smooth` (Smoothing Parameter)

Required packages: `bnclassify`

Tree Augmented Naive Bayes Classifier Structure Learner Wrapper

```
method = 'tanSearch'
```

Type: Classification

Tuning parameters:

- `k` (#Folds)
- `epsilon` (Minimum Absolute Improvement)
- `smooth` (Smoothing Parameter)
- `final_smooth` (Final Smoothing Parameter)
- `sp` (Super-Parent)

Required packages: `bnclassify`

Tree Augmented Naive Bayes Classifier with Attribute Weighting

```
method = 'awtan'
```

Type: Classification

Tuning parameters:

- `score` (Score Function)
- `smooth` (Smoothing Parameter)

Required packages: `bnclassify`

7.0.7 Cost Sensitive Learning

(back to contents)

Cost-Sensitive C5.0

```
method = 'C5.0Cost'
```

Type: Classification

Tuning parameters:

- `trials` (# Boosting Iterations)
- `model` (Model Type)
- `winnow` (Winnow)
- `cost` (Cost)

Required packages: `C50` , `plyr`

A model-specific variable importance metric is available.

Cost-Sensitive CART

```
method = 'rpartCost'
```

Type: Classification

Tuning parameters:

- `cp` (Complexity Parameter)
- `Cost` (Cost)

Required packages: `rpart` , `plyr`

L2 Regularized Linear Support Vector Machines with Class Weights

```
method = 'svmLinearWeights2'
```

Type: Classification

Tuning parameters:

- `cost` (Cost)
- `Loss` (Loss Function)
- `weight` (Class Weight)

Required packages: `LiblineaR`

Linear Support Vector Machines with Class Weights

```
method = 'svmLinearWeights'
```

Type: Classification

Tuning parameters:

- `cost` (Cost)
- `weight` (Class Weight)

Required packages: `e1071`

Multilayer Perceptron Network with Dropout

```
method = 'mlpKerasDropoutCost'
```

Type: Classification

Tuning parameters:

- `size` (#Hidden Units)
- `dropout` (Dropout Rate)
- `batch_size` (Batch Size)
- `lr` (Learning Rate)
- `rho` (Rho)
- `decay` (Learning Rate Decay)
- `cost` (Cost)
- `activation` (Activation Function)

Required packages: `keras`

Notes: After `train` completes, the keras model object is serialized so that it can be used between R session. When predicting, the code will temporarily unsearalize the object. To make the predictions more efficient, the user might want to use `keras::unsearlize_model(object$finalModel$object)` in the current R session so that that operation is only done once. Also, this model cannot be run in parallel due to the nature of how tensorflow does the computations. Finally, the cost parameter weights the first class in the outcome vector. Unlike other packages used by `train`, the `dplyr` package is fully loaded when this model is used.

Multilayer Perceptron Network with Weight Decay

```
method = 'mlpKerasDecayCost'
```

Type: Classification

Tuning parameters:

- `size` (#Hidden Units)
- `lambda` (L2 Regularization)
- `batch_size` (Batch Size)
- `lr` (Learning Rate)
- `rho` (Rho)
- `decay` (Learning Rate Decay)
- `cost` (Cost)
- `activation` (Activation Function)

Required packages: `keras`

Notes: After `train` completes, the keras model object is serialized so that it can be used between R session. When predicting, the code will temporarily unsearalize the object. To make the predictions more efficient, the user might want to use `keras::unsearlize_model(object$finalModel$object)` in the current R session so that that operation is only done once. Also, this model cannot be run in parallel due to the nature of how tensorflow does the computations. Finally, the cost parameter weights the first class in the outcome vector. Unlike other packages used by `train`, the `dplyr` package is fully loaded when this model is used.

Support Vector Machines with Class Weights

```
method = 'svmRadialWeights'
```

Type: Classification

Tuning parameters:

- `sigma` (Sigma)
- `C` (Cost)
- `Weight` (Weight)

Required packages: `kernlab`

7.0.8 Discriminant Analysis

(back to contents)

Adaptive Mixture Discriminant Analysis

```
method = 'amdai'
```

Type: Classification

Tuning parameters:

- `model` (Model Type)

Required packages: `adaptDA`

Binary Discriminant Analysis

```
method = 'binda'
```

Type: Classification

Tuning parameters:

- `lambda.freqs` (Shrinkage Intensity)

Required packages: `binda`

Diagonal Discriminant Analysis

```
method = 'dda'
```

Type: Classification

Tuning parameters:

- `model` (Model)
- `shrinkage` (Shrinkage Type)

Required packages: `sparsediscrim`

Distance Weighted Discrimination with Polynomial Kernel

```
method = 'dwdPoly'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)
- `degree` (Polynomial Degree)
- `scale` (Scale)

Required packages: `kerndwd`

Distance Weighted Discrimination with Radial Basis Function Kernel

```
method = 'dwdRadial'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)
- `sigma` (Sigma)

Required packages: `kernlab` , `kerndwd`

Factor-Based Linear Discriminant Analysis

```
method = 'RFlda'
```

Type: Classification

Tuning parameters:

- `q` (# Factors)

Required packages: `HiDimDA`

Heteroscedastic Discriminant Analysis

```
method = 'hda'
```

Type: Classification

Tuning parameters:

- `gamma` (Gamma)
- `lambda` (Lambda)
- `newdim` (Dimension of the Discriminative Subspace)

Required packages: `hda`

High Dimensional Discriminant Analysis

```
method = 'hdda'
```

Type: Classification

Tuning parameters:

- `threshold` (Threshold)
- `model` (Model Type)

Required packages: `HDclassif`

High-Dimensional Regularized Discriminant Analysis

```
method = 'hdrda'
```

Type: Classification

Tuning parameters:

- `gamma` (Gamma)

- `lambda` (Lambda)
- `shrinkage_type` (Shrinkage Type)

Required packages: `sparsediscrim`

Linear Discriminant Analysis

```
method = 'lda'
```

Type: Classification

No tuning parameters for this model

Required packages: `MASS`

Linear Discriminant Analysis

```
method = 'lda2'
```

Type: Classification

Tuning parameters:

- `dimen` (#Discriminant Functions)

Required packages: `MASS`

Linear Discriminant Analysis with Stepwise Feature Selection

```
method = 'stepLDA'
```

Type: Classification

Tuning parameters:

- `maxvar` (Maximum #Variables)
- `direction` (Search Direction)

Required packages: `klaR` , `MASS`

Linear Distance Weighted Discrimination

```
method = 'dwdLinear'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)

Required packages: `kerndwd`

Localized Linear Discriminant Analysis

```
method = 'loclda'
```

Type: Classification

Tuning parameters:

- `k` (#Nearest Neighbors)

Required packages: `klaR`

Maximum Uncertainty Linear Discriminant Analysis

```
method = 'Mlda'
```

Type: Classification

No tuning parameters for this model

Required packages: `HiDimDA`

Mixture Discriminant Analysis

```
method = 'mda'
```

Type: Classification

Tuning parameters:

- `subclasses` (#Subclasses Per Class)

Required packages: `mda`

Penalized Discriminant Analysis

```
method = 'pda'
```

Type: Classification

Tuning parameters:

- `lambda` (Shrinkage Penalty Coefficient)

Required packages: `mda`

Penalized Discriminant Analysis

```
method = 'pda2'
```

Type: Classification

Tuning parameters:

- `df` (Degrees of Freedom)

Required packages: `mda`

Penalized Linear Discriminant Analysis

```
method = 'PenalizedLDA'
```

Type: Classification

Tuning parameters:

- `lambda` (L1 Penalty)
- `K` (#Discriminant Functions)

Required packages: `penalizedLDA` , `plyr`

Quadratic Discriminant Analysis

```
method = 'qda'
```

Type: Classification

No tuning parameters for this model

Required packages: `MASS`

Quadratic Discriminant Analysis with Stepwise Feature Selection

```
method = 'stepQDA'
```

Type: Classification

Tuning parameters:

- `maxvar` (Maximum #Variables)
- `direction` (Search Direction)

Required packages: `klaR` , `MASS`

Regularized Discriminant Analysis

```
method = 'rda'
```

Type: Classification

Tuning parameters:

- `gamma` (Gamma)
- `lambda` (Lambda)

Required packages: `klaR`

Regularized Linear Discriminant Analysis

```
method = 'rlda'
```

Type: Classification

Tuning parameters:

- `estimator` (Regularization Method)

Required packages: `sparsediscrim`

Robust Linear Discriminant Analysis

```
method = 'Linda'
```

Type: Classification

No tuning parameters for this model

Required packages: `rrcov`

Robust Mixture Discriminant Analysis

```
method = 'rmda'
```

Type: Classification

Tuning parameters:

- `K` (#Subclasses Per Class)
- `model` (Model)

Required packages: `robustDA`

Robust Quadratic Discriminant Analysis

```
method = 'QdaCov'
```

Type: Classification

No tuning parameters for this model

Required packages: `rrcov`

Robust Regularized Linear Discriminant Analysis

```
method = 'rrlda'
```

Type: Classification

Tuning parameters:

- `lambda` (Penalty Parameter)
- `hp` (Robustness Parameter)
- `penalty` (Penalty Type)

Required packages: `rrlda`

Notes: Unlike other packages used by `train`, the `rrlda` package is fully loaded when this model is used.

Shrinkage Discriminant Analysis

```
method = 'sda'
```

Type: Classification

Tuning parameters:

- `diagonal` (Diagonalize)
- `lambda` (shrinkage)

Required packages: `sda`

Sparse Linear Discriminant Analysis

```
method = 'sparseLDA'
```

Type: Classification

Tuning parameters:

- `NumVars` (# Predictors)
- `lambda` (Lambda)

Required packages: `sparseLDA`

Sparse Mixture Discriminant Analysis

```
method = 'smda'
```

Type: Classification

Tuning parameters:

- `NumVars` (# Predictors)
- `lambda` (Lambda)
- `R` (# Subclasses)

Required packages: `sparseLDA`

Stabilized Linear Discriminant Analysis

```
method = 'slda'
```

Type: Classification

No tuning parameters for this model

Required packages: `ipred`

7.0.9 Distance Weighted Discrimination

(back to contents)

Distance Weighted Discrimination with Polynomial Kernel

```
method = 'dwdPoly'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)

- `degree` (Polynomial Degree)
- `scale` (Scale)

Required packages: `kerndwd`

Distance Weighted Discrimination with Radial Basis Function Kernel

```
method = 'dwdRadial'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)
- `sigma` (Sigma)

Required packages: `kernlab` , `kerndwd`

Linear Distance Weighted Discrimination

```
method = 'dwdLinear'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)

Required packages: `kerndwd`

Sparse Distance Weighted Discrimination

```
method = 'sdwd'
```

Type: Classification

Tuning parameters:

- `lambda` (L1 Penalty)
- `lambda2` (L2 Penalty)

Required packages: `sdwd`

A model-specific variable importance metric is available.

7.0.10 Ensemble Model

(back to contents)

AdaBoost Classification Trees

```
method = 'adaboost'
```

Type: Classification

Tuning parameters:

- `nIter` (#Trees)
- `method` (Method)

Required packages: `fastAdaboost`

AdaBoost.M1

```
method = 'AdaBoost.M1'
```

Type: Classification

Tuning parameters:

- `mfinal` (#Trees)
- `maxdepth` (Max Tree Depth)
- `coeflearn` (Coefficient Type)

Required packages: `adabag`, `plyr`

A model-specific variable importance metric is available.

Bagged AdaBoost

```
method = 'AdaBag'
```

Type: Classification

Tuning parameters:

- `mfinal` (#Trees)
- `maxdepth` (Max Tree Depth)

Required packages: `adabag`, `plyr`

A model-specific variable importance metric is available.

Bagged CART

```
method = 'treebag'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `ipred` , `plyr` , `e1071`

A model-specific variable importance metric is available.

Bagged Flexible Discriminant Analysis

```
method = 'bagFDA'
```

Type: Classification

Tuning parameters:

- `degree` (Product Degree)
- `nprune` (#Terms)

Required packages: `earth` , `mda`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train` , the `earth` package is fully loaded when this model is used.

Bagged Logic Regression

```
method = 'logicBag'
```

Type: Regression, Classification

Tuning parameters:

- `nleaves` (Maximum Number of Leaves)
- `ntrees` (Number of Trees)

Required packages: `logicFS`

Notes: Unlike other packages used by `train`, the `logicFS` package is fully loaded when this model is used.

Bagged MARS

```
method = 'bagEarth'
```

Type: Regression, Classification

Tuning parameters:

- `nprune` (#Terms)
- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Bagged MARS using gCV Pruning

```
method = 'bagEarthGCV'
```

Type: Regression, Classification

Tuning parameters:

- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Bagged Model

```
method = 'bag'
```

Type: Regression, Classification

Tuning parameters:

- `vars` (#Randomly Selected Predictors)

Required packages: `caret`

Boosted Classification Trees

```
method = 'ada'
```

Type: Classification

Tuning parameters:

- `iter` (#Trees)

- `maxdepth` (Max Tree Depth)
- `nu` (Learning Rate)

Required packages: `ada` , `plyr`

Boosted Generalized Additive Model

```
method = 'gamboost'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `prune` (AIC Prune?)

Required packages: `mboost` , `plyr` , `import`

Notes: The `prune` option for this model enables the number of iterations to be determined by the optimal AIC value across all iterations. See the examples in `?mboost::mstop` . If pruning is not used, the ensemble makes predictions using the exact value of the `mstop` tuning parameter value.

Boosted Generalized Linear Model

```
method = 'glmboost'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `prune` (AIC Prune?)

Required packages: `plyr` , `mboost`

A model-specific variable importance metric is available. Notes: The `prune` option for this model enables the number of iterations to be determined by the optimal AIC value across all iterations. See the examples in `?mboost::mstop` . If pruning is not used, the ensemble makes predictions using the exact value of the `mstop` tuning parameter value.

Boosted Linear Model

```
method = 'BstLm'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `nu` (Shrinkage)

Required packages: `bst` , `plyr`

Boosted Logistic Regression

```
method = 'LogitBoost'
```

Type: Classification

Tuning parameters:

- `nIter` (# Boosting Iterations)

Required packages: `caTools`

Boosted Smoothing Spline

```
method = 'bstSm'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `nu` (Shrinkage)

Required packages: `bst` , `plyr`

Boosted Tree

```
method = 'blackboost'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (#Trees)
- `maxdepth` (Max Tree Depth)

Required packages: `party` , `mboost` , `plyr`

Boosted Tree

```
method = 'bstTree'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `maxdepth` (Max Tree Depth)
- `nu` (Shrinkage)

Required packages: `bst` , `plyr`

C5.0

```
method = 'C5.0'
```

Type: Classification

Tuning parameters:

- `trials` (# Boosting Iterations)
- `model` (Model Type)
- `winnow` (Winnow)

Required packages: `C50` , `plyr`

A model-specific variable importance metric is available.

Conditional Inference Random Forest

```
method = 'cforest'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `party`

A model-specific variable importance metric is available.

Cost-Sensitive C5.0

```
method = 'C5.0Cost'
```

Type: Classification

Tuning parameters:

- `trials` (# Boosting Iterations)
- `model` (Model Type)
- `winnow` (Winnow)
- `cost` (Cost)

Required packages: `C50` , `plyr`

A model-specific variable importance metric is available.

Cubist

```
method = 'cubist'
```

Type: Regression

Tuning parameters:

- committees (#Committees)
- neighbors (#Instances)

Required packages: Cubist

A model-specific variable importance metric is available.

DeepBoost

```
method = 'deepboost'
```

Type: Classification

Tuning parameters:

- num_iter (# Boosting Iterations)
- tree_depth (Tree Depth)
- beta (L1 Regularization)
- lambda (Tree Depth Regularization)
- loss_type (Loss)

Required packages: deepboost

Ensembles of Generalized Linear Models

```
method = 'randomGLM'
```

Type: Regression, Classification

Tuning parameters:

- `maxInteractionOrder` (Interaction Order)

Required packages: `randomGLM`

Notes: Unlike other packages used by `train`, the `randomGLM` package is fully loaded when this model is used.

eXtreme Gradient Boosting

```
method = 'xgbDART'
```

Type: Regression, Classification

Tuning parameters:

- `nrounds` (# Boosting Iterations)
- `max_depth` (Max Tree Depth)
- `eta` (Shrinkage)
- `gamma` (Minimum Loss Reduction)
- `subsample` (Subsample Percentage)
- `colsample_bytree` (Subsample Ratio of Columns)
- `rate_drop` (Fraction of Trees Dropped)
- `skip_drop` (Prob. of Skipping Drop-out)
- `min_child_weight` (Minimum Sum of Instance Weight)

Required packages: `xgboost`, `plyr`

A model-specific variable importance metric is available.

eXtreme Gradient Boosting

```
method = 'xgbLinear'
```

Type: Regression, Classification

Tuning parameters:

- `nrounds` (# Boosting Iterations)
- `lambda` (L2 Regularization)
- `alpha` (L1 Regularization)
- `eta` (Learning Rate)

Required packages: `xgboost`

A model-specific variable importance metric is available.

eXtreme Gradient Boosting

```
method = 'xgbTree'
```

Type: Regression, Classification

Tuning parameters:

- `nrounds` (# Boosting Iterations)
- `max_depth` (Max Tree Depth)
- `eta` (Shrinkage)
- `gamma` (Minimum Loss Reduction)
- `colsample_bytree` (Subsample Ratio of Columns)
- `min_child_weight` (Minimum Sum of Instance Weight)
- `subsample` (Subsample Percentage)

Required packages: `xgboost` , `plyr`

A model-specific variable importance metric is available.

Gradient Boosting Machines

```
method = 'gbm_h2o'
```

Type: Regression, Classification

Tuning parameters:

- `ntrees` (# Boosting Iterations)
- `max_depth` (Max Tree Depth)
- `min_rows` (Min. Terminal Node Size)
- `learn_rate` (Shrinkage)
- `col_sample_rate` (#Randomly Selected Predictors)

Required packages: `h2o`

A model-specific variable importance metric is available.

Model Averaged Neural Network

```
method = 'avNNet'
```

Type: Classification, Regression

Tuning parameters:

- `size` (#Hidden Units)
- `decay` (Weight Decay)

- `bag` (Bagging)

Required packages: `nnet`

Oblique Random Forest

```
method = 'ORFlog'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Oblique Random Forest

```
method = 'ORFpls'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Oblique Random Forest

```
method = 'ORFridge'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Oblique Random Forest

```
method = 'ORFsvm'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Parallel Random Forest

```
method = 'parRF'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `e1071` , `randomForest` , `foreach` , `import`

A model-specific variable importance metric is available.

Quantile Random Forest

```
method = 'qrf'
```

Type: Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `quantregForest`

Quantile Regression Neural Network

```
method = 'qrnn'
```

Type: Regression

Tuning parameters:

- `n.hidden` (#Hidden Units)
- `penalty` (Weight Decay)
- `bag` (Bagged Models?)

Required packages: `qrnn`

Random Ferns

```
method = 'rFerns'
```

Type: Classification

Tuning parameters:

- `depth` (Fern Depth)

Required packages: `rFerns`

Random Forest

```
method = 'ranger'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `splitrule` (Splitting Rule)
- `min.node.size` (Minimal Node Size)

Required packages: `e1071` , `ranger` , `dplyr`

A model-specific variable importance metric is available.

Random Forest

```
method = 'Rborist'
```

Type: Classification, Regression

Tuning parameters:

- `predFixed` (#Randomly Selected Predictors)
- `minNode` (Minimal Node Size)

Required packages: `Rborist`

A model-specific variable importance metric is available.

Random Forest

```
method = 'rf'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `randomForest`

A model-specific variable importance metric is available.

Random Forest by Randomization

```
method = 'extraTrees'
```

Type: Regression, Classification

Tuning parameters:

- `mtry` (# Randomly Selected Predictors)
- `numRandomCuts` (# Random Cuts)

Required packages: `extraTrees`

Random Forest Rule-Based Model

```
method = 'rfRules'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `maxdepth` (Maximum Rule Depth)

Required packages: `randomForest` , `inTrees` , `plyr`

A model-specific variable importance metric is available.

Regularized Random Forest

```
method = 'RRF'
```

Type: Regression, Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `coefReg` (Regularization Value)
- `coefImp` (Importance Coefficient)

Required packages: `randomForest` , `RRF`

A model-specific variable importance metric is available.

Regularized Random Forest

```
method = 'RRFglobal'
```

Type: Regression, Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `coefReg` (Regularization Value)

Required packages: `RRF`

A model-specific variable importance metric is available.

Rotation Forest

```
method = 'rotationForest'
```

Type: Classification

Tuning parameters:

- K (#Variable Subsets)
- L (Ensemble Size)

Required packages: rotationForest

A model-specific variable importance metric is available.

Rotation Forest

```
method = 'rotationForestCp'
```

Type: Classification

Tuning parameters:

- K (#Variable Subsets)
- L (Ensemble Size)
- cp (Complexity Parameter)

Required packages: rpart , plyr , rotationForest

A model-specific variable importance metric is available.

Stochastic Gradient Boosting

```
method = 'gbm'
```

Type: Regression, Classification

Tuning parameters:

- n.trees (# Boosting Iterations)

- `interaction.depth` (Max Tree Depth)
- `shrinkage` (Shrinkage)
- `n.minobsinnode` (Min. Terminal Node Size)

Required packages: `gbm` , `plyr`

A model-specific variable importance metric is available.

Tree-Based Ensembles

```
method = 'nodeHarvest'
```

Type: Regression, Classification

Tuning parameters:

- `maxinter` (Maximum Interaction Depth)
- `mode` (Prediction Mode)

Required packages: `nodeHarvest`

Weighted Subspace Random Forest

```
method = 'wsrf'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `wsrf`

7.0.11 Feature Extraction

(back to contents)

Independent Component Regression

```
method = 'icr'
```

Type: Regression

Tuning parameters:

- `n.comp` (#Components)

Required packages: `fastICA`

Neural Networks with Feature Extraction

```
method = 'pcaNNet'
```

Type: Classification, Regression

Tuning parameters:

- `size` (#Hidden Units)
- `decay` (Weight Decay)

Required packages: `nnet`

Partial Least Squares

```
method = 'kernelpls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Partial Least Squares

```
method = 'pls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Partial Least Squares

```
method = 'simpls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Partial Least Squares

```
method = 'widekernelpls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Principal Component Analysis

```
method = 'pcr'
```

Type: Regression

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

Projection Pursuit Regression

```
method = 'ppr'
```

Type: Regression

Tuning parameters:

- `nterms` (# Terms)

Sparse Partial Least Squares

```
method = 'spls'
```

Type: Regression, Classification

Tuning parameters:

- `K` (#Components)
- `eta` (Threshold)
- `kappa` (Kappa)

Required packages: `spls`

Supervised Principal Component Analysis

```
method = 'superpc'
```

Type: Regression

Tuning parameters:

- `threshold` (Threshold)
- `n.components` (#Components)

Required packages: `superpc`

7.0.12 Feature Selection Wrapper

(back to contents)

Generalized Linear Model with Stepwise Feature Selection

```
method = 'glmStepAIC'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `MASS`

Linear Discriminant Analysis with Stepwise Feature Selection

```
method = 'stepLDA'
```

Type: Classification

Tuning parameters:

- `maxvar` (Maximum #Variables)
- `direction` (Search Direction)

Required packages: `klaR` , `MASS`

Linear Regression with Backwards Selection

```
method = 'leapBackward'
```

Type: Regression

Tuning parameters:

- `nvmax` (Maximum Number of Predictors)

Required packages: `leaps`

Linear Regression with Forward Selection

```
method = 'leapForward'
```

Type: Regression

Tuning parameters:

- `nvmax` (Maximum Number of Predictors)

Required packages: `leaps`

Linear Regression with Stepwise Selection

```
method = 'leapSeq'
```

Type: Regression

Tuning parameters:

- `nvmax` (Maximum Number of Predictors)

Required packages: `leaps`

Linear Regression with Stepwise Selection

```
method = 'lmStepAIC'
```

Type: Regression

No tuning parameters for this model

Required packages: `MASS`

Quadratic Discriminant Analysis with Stepwise Feature Selection

```
method = 'stepQDA'
```

Type: Classification

Tuning parameters:

- `maxvar` (Maximum #Variables)
- `direction` (Search Direction)

Required packages: `klaR` , `MASS`

Ridge Regression with Variable Selection

```
method = 'foba'
```

Type: Regression

Tuning parameters:

- `k` (#Variables Retained)
- `lambda` (L2 Penalty)

Required packages: `foba`

7.0.13 Gaussian Process

(back to contents)

Gaussian Process

```
method = 'gausssprLinear'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `kernlab`

Gaussian Process with Polynomial Kernel

```
method = 'gausssprPoly'
```

Type: Regression, Classification

Tuning parameters:

- `degree` (Polynomial Degree)
- `scale` (Scale)

Required packages: `kernlab`

Gaussian Process with Radial Basis Function Kernel

```
method = 'gausssprRadial'
```

Type: Regression, Classification

Tuning parameters:

- `sigma` (Sigma)

Required packages: `kernlab`

Variational Bayesian Multinomial Probit Regression

```
method = 'vbmpRadial'
```

Type: Classification

Tuning parameters:

- `estimateTheta` (Theta Estimated)

Required packages: `vbmp`

7.0.14 Generalized Additive Model

(back to contents)

Boosted Generalized Additive Model

```
method = 'gamboost'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `prune` (AIC Prune?)

Required packages: `mboost` , `plyr` , `import`

Notes: The `prune` option for this model enables the number of iterations to be determined by the optimal AIC value across all iterations. See the examples in `?mboost::mstop` . If pruning is not used, the ensemble makes predictions using the exact value of the `mstop` tuning parameter value.

Generalized Additive Model using LOESS

```
method = 'gamLoess'
```

Type: Regression, Classification

Tuning parameters:

- `span` (Span)
- `degree` (Degree)

Required packages: `gam`

A model-specific variable importance metric is available. Notes: Which terms enter the model in a nonlinear manner is determined by the number of unique values for the predictor. For example, if a predictor only has four unique values, most basis expansion method will fail because there are not enough granularity in the data. By default, a predictor must have at least 10 unique values to be used in a nonlinear basis expansion. Unlike other packages used by `train` , the `gam` package is fully loaded when this model is used.

Generalized Additive Model using Splines

```
method = 'bam'
```

Type: Regression, Classification

Tuning parameters:

- `select` (Feature Selection)
- `method` (Method)

Required packages: `mgcv`

A model-specific variable importance metric is available. Notes: Which terms enter the model in a nonlinear manner is determined by the number of unique values for the predictor. For example, if a predictor only has four unique values, most basis expansion method will fail because there are not enough granularity in the data. By default, a

predictor must have at least 10 unique values to be used in a nonlinear basis expansion. Unlike other packages used by `train`, the `mgcv` package is fully loaded when this model is used.

Generalized Additive Model using Splines

```
method = 'gam'
```

Type: Regression, Classification

Tuning parameters:

- `select` (Feature Selection)
- `method` (Method)

Required packages: `mgcv`

A model-specific variable importance metric is available. Notes: Which terms enter the model in a nonlinear manner is determined by the number of unique values for the predictor. For example, if a predictor only has four unique values, most basis expansion method will fail because there are not enough granularity in the data. By default, a predictor must have at least 10 unique values to be used in a nonlinear basis expansion. Unlike other packages used by `train`, the `mgcv` package is fully loaded when this model is used.

Generalized Additive Model using Splines

```
method = 'gamSpline'
```

Type: Regression, Classification

Tuning parameters:

- `df` (Degrees of Freedom)

Required packages: `gam`

A model-specific variable importance metric is available. Notes: Which terms enter the model in a nonlinear manner is determined by the number of unique values for the predictor. For example, if a predictor only has four unique values, most basis expansion method will fail because there are not enough granularity in the data. By default, a predictor must have at least 10 unique values to be used in a nonlinear basis expansion. Unlike other packages used by `train` , the `gam` package is fully loaded when this model is used.

7.0.15 Generalized Linear Model

(back to contents)

Bayesian Generalized Linear Model

```
method = 'bayesglm'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `arm`

Boosted Generalized Linear Model

```
method = 'glmboost'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `prune` (AIC Prune?)

Required packages: `plyr` , `mboost`

A model-specific variable importance metric is available. Notes: The `prune` option for this model enables the number of iterations to be determined by the optimal AIC value across all iterations. See the examples in `?mboost::mstop` . If pruning is not used, the ensemble makes predictions using the exact value of the `mstop` tuning parameter value.

Ensembles of Generalized Linear Models

```
method = 'randomGLM'
```

Type: Regression, Classification

Tuning parameters:

- `maxInteractionOrder` (Interaction Order)

Required packages: `randomGLM`

Notes: Unlike other packages used by `train`, the `randomGLM` package is fully loaded when this model is used.

Generalized Additive Model using LOESS

```
method = 'gamLoess'
```

Type: Regression, Classification

Tuning parameters:

- `span` (Span)
- `degree` (Degree)

Required packages: `gam`

A model-specific variable importance metric is available. Notes: Which terms enter the model in a nonlinear manner is determined by the number of unique values for the predictor. For example, if a predictor only has four unique values, most basis expansion method will fail because there are not enough granularity in the data. By default, a predictor must have at least 10 unique values to be used in a nonlinear basis expansion. Unlike other packages used by `train`, the `gam` package is fully loaded when this model is used.

Generalized Additive Model using Splines

```
method = 'bam'
```

Type: Regression, Classification

Tuning parameters:

- `select` (Feature Selection)
- `method` (Method)

Required packages: `mgcv`

A model-specific variable importance metric is available. Notes: Which terms enter the model in a nonlinear manner is determined by the number of unique values for the predictor. For example, if a predictor only has four unique values, most basis expansion method will fail because there are not enough granularity in the data. By default, a predictor must have at least 10 unique values to be used in a nonlinear basis expansion. Unlike other packages used by `train` , the `mgcv` package is fully loaded when this model is used.

Generalized Additive Model using Splines

```
method = 'gam'
```

Type: Regression, Classification

Tuning parameters:

- `select` (Feature Selection)
- `method` (Method)

Required packages: `mgcv`

A model-specific variable importance metric is available. Notes: Which terms enter the model in a nonlinear manner is determined by the number of unique values for the predictor. For example, if a predictor only has four unique values, most basis expansion method will fail because there are not enough granularity in the data. By default, a predictor must have at least 10 unique values to be used in a nonlinear basis expansion. Unlike other packages used by `train`, the `mgcv` package is fully loaded when this model is used.

Generalized Additive Model using Splines

```
method = 'gamSpline'
```

Type: Regression, Classification

Tuning parameters:

- `df` (Degrees of Freedom)

Required packages: `gam`

A model-specific variable importance metric is available. Notes: Which terms enter the model in a nonlinear manner is determined by the number of unique values for the predictor. For example, if a predictor only has four unique values, most basis expansion method will fail because there are not enough granularity in the data. By default, a predictor must have at least 10 unique values to be used in a nonlinear basis expansion. Unlike other packages used by `train`, the `gam` package is fully loaded when this model is used.

Generalized Linear Model

```
method = 'glm'
```

Type: Regression, Classification

No tuning parameters for this model

A model-specific variable importance metric is available.

Generalized Linear Model with Stepwise Feature Selection

```
method = 'glmStepAIC'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `MASS`

glmnet

```
method = 'glmnet_h2o'
```

Type: Regression, Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `lambda` (Regularization Parameter)

Required packages: `h2o`

A model-specific variable importance metric is available.

glmnet

```
method = 'glmnet'
```

Type: Regression, Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `lambda` (Regularization Parameter)

Required packages: `glmnet` , `Matrix`

A model-specific variable importance metric is available.

Multi-Step Adaptive MCP-Net

```
method = 'msaenet'
```

Type: Regression, Classification

Tuning parameters:

- `alphas` (Alpha)
- `nsteps` (#Adaptive Estimation Steps)
- `scale` (Adaptive Weight Scaling Factor)

Required packages: `msaenet`

A model-specific variable importance metric is available.

Negative Binomial Generalized Linear Model

```
method = 'glm.nb'
```

Type: Regression

Tuning parameters:

- `link` (Link Function)

Required packages: `MASS`

A model-specific variable importance metric is available.

Penalized Ordinal Regression

```
method = 'ordinalNet'
```

Type: Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `criteria` (Selection Criterion)
- `link` (Link Function)

Required packages: `ordinalNet` , `plyr`

A model-specific variable importance metric is available. Notes:

Requires ordinalNet package version >= 2.0

7.0.16 Handle Missing Predictor Data

(back to contents)

AdaBoost.M1

```
method = 'AdaBoost.M1'
```

Type: Classification

Tuning parameters:

- `mfinal` (#Trees)
- `maxdepth` (Max Tree Depth)
- `coeflearn` (Coefficient Type)

Required packages: `adabag`, `plyr`

A model-specific variable importance metric is available.

Bagged AdaBoost

```
method = 'AdaBag'
```

Type: Classification

Tuning parameters:

- `mfinal` (#Trees)
- `maxdepth` (Max Tree Depth)

Required packages: `adabag` , `plyr`

A model-specific variable importance metric is available.

Boosted Classification Trees

```
method = 'ada'
```

Type: Classification

Tuning parameters:

- `iter` (#Trees)
- `maxdepth` (Max Tree Depth)
- `nu` (Learning Rate)

Required packages: `ada` , `plyr`

C5.0

```
method = 'C5.0'
```

Type: Classification

Tuning parameters:

- `trials` (# Boosting Iterations)
- `model` (Model Type)
- `winnow` (Winnow)

Required packages: `C50` , `plyr`

A model-specific variable importance metric is available.

CART

```
method = 'rpart'
```

Type: Regression, Classification

Tuning parameters:

- `cp` (Complexity Parameter)

Required packages: `rpart`

A model-specific variable importance metric is available.

CART

```
method = 'rpart1SE'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `rpart`

A model-specific variable importance metric is available. Notes: This CART model replicates the same process used by the `rpart` function where the model complexity is determined using the one-standard error method. This procedure is replicated inside of the resampling done by `train` so that an external resampling estimate can be obtained.

CART

```
method = 'rpart2'
```

Type: Regression, Classification

Tuning parameters:

- `maxdepth` (Max Tree Depth)

Required packages: `rpart`

A model-specific variable importance metric is available.

CART or Ordinal Responses

```
method = 'rpartScore'
```

Type: Classification

Tuning parameters:

- `cp` (Complexity Parameter)
- `split` (Split Function)
- `prune` (Pruning Measure)

Required packages: `rpartScore` , `plyr`

A model-specific variable importance metric is available.

Cost-Sensitive C5.0

```
method = 'C5.0Cost'
```

Type: Classification

Tuning parameters:

- `trials` (# Boosting Iterations)
- `model` (Model Type)
- `winnow` (Winnow)
- `cost` (Cost)

Required packages: `C50` , `plyr`

A model-specific variable importance metric is available.

Cost-Sensitive CART

```
method = 'rpartCost'
```

Type: Classification

Tuning parameters:

- `cp` (Complexity Parameter)
- `Cost` (Cost)

Required packages: `rpart` , `plyr`

Single C5.0 Ruleset

```
method = 'C5.0Rules'
```

Type: Classification

No tuning parameters for this model

Required packages: `C50`

A model-specific variable importance metric is available.

Single C5.0 Tree

```
method = 'C5.0Tree'
```

Type: Classification

No tuning parameters for this model

Required packages: `C50`

A model-specific variable importance metric is available.

7.0.17 Implicit Feature Selection

(back to contents)

AdaBoost Classification Trees

```
method = 'adaboost'
```

Type: Classification

Tuning parameters:

- `nIter` (#Trees)
- `method` (Method)

Required packages: `fastAdaboost`

AdaBoost.M1

```
method = 'AdaBoost.M1'
```

Type: Classification

Tuning parameters:

- `mfinal` (#Trees)
- `maxdepth` (Max Tree Depth)
- `coeflearn` (Coefficient Type)

Required packages: `adabag` , `plyr`

A model-specific variable importance metric is available.

Bagged AdaBoost

```
method = 'AdaBag'
```

Type: Classification

Tuning parameters:

- `mfinal` (#Trees)
- `maxdepth` (Max Tree Depth)

Required packages: `adabag`, `plyr`

A model-specific variable importance metric is available.

Bagged Flexible Discriminant Analysis

```
method = 'bagFDA'
```

Type: Classification

Tuning parameters:

- `degree` (Product Degree)
- `nprune` (#Terms)

Required packages: `earth`, `mda`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Bagged MARS

```
method = 'bagEarth'
```

Type: Regression, Classification

Tuning parameters:

- `nprune` (#Terms)
- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Bagged MARS using gCV Pruning

```
method = 'bagEarthGCV'
```

Type: Regression, Classification

Tuning parameters:

- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Bayesian Additive Regression Trees

```
method = 'bartMachine'
```

Type: Classification, Regression

Tuning parameters:

- `num_trees` (#Trees)
- `k` (Prior Boundary)

- `alpha` (Base Terminal Node Hyperparameter)
- `beta` (Power Terminal Node Hyperparameter)
- `nu` (Degrees of Freedom)

Required packages: `bartMachine`

A model-specific variable importance metric is available.

Boosted Classification Trees

```
method = 'ada'
```

Type: Classification

Tuning parameters:

- `iter` (#Trees)
- `maxdepth` (Max Tree Depth)
- `nu` (Learning Rate)

Required packages: `ada` , `plyr`

Boosted Generalized Additive Model

```
method = 'gamboost'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `prune` (AIC Prune?)

Required packages: `mboost` , `plyr` , `import`

Notes: The `prune` option for this model enables the number of iterations to be determined by the optimal AIC value across all iterations. See the examples in `?mboost::mstop` . If pruning is not used, the ensemble makes predictions using the exact value of the `mstop` tuning parameter value.

Boosted Linear Model

```
method = 'BstLm'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `nu` (Shrinkage)

Required packages: `bst` , `plyr`

Boosted Logistic Regression

```
method = 'LogitBoost'
```

Type: Classification

Tuning parameters:

- `nIter` (# Boosting Iterations)

Required packages: `caTools`

Boosted Smoothing Spline

```
method = 'bstSm'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `nu` (Shrinkage)

Required packages: `bst` , `plyr`

C4.5-like Trees

```
method = 'J48'
```

Type: Classification

Tuning parameters:

- `C` (Confidence Threshold)
- `M` (Minimum Instances Per Leaf)

Required packages: `RWeka`

C5.0

```
method = 'C5.0'
```

Type: Classification

Tuning parameters:

- `trials` (# Boosting Iterations)
- `model` (Model Type)
- `winnow` (Winnow)

Required packages: `C50` , `plyr`

A model-specific variable importance metric is available.

CART

```
method = 'rpart'
```

Type: Regression, Classification

Tuning parameters:

- `cp` (Complexity Parameter)

Required packages: `rpart`

A model-specific variable importance metric is available.

CART

```
method = 'rpart1SE'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `rpart`

A model-specific variable importance metric is available. Notes: This CART model replicates the same process used by the `rpart` function where the model complexity is determined using the one-standard error method. This procedure is replicated inside of the resampling done by `train` so that an external resampling estimate can be obtained.

CART

```
method = 'rpart2'
```

Type: Regression, Classification

Tuning parameters:

- `maxdepth` (Max Tree Depth)

Required packages: `rpart`

A model-specific variable importance metric is available.

CART or Ordinal Responses

```
method = 'rpartScore'
```

Type: Classification

Tuning parameters:

- `cp` (Complexity Parameter)
- `split` (Split Function)

- `prune` (Pruning Measure)

Required packages: `rpartScore` , `plyr`

A model-specific variable importance metric is available.

CHi-squared Automated Interaction Detection

```
method = 'chaid'
```

Type: Classification

Tuning parameters:

- `alpha2` (Merging Threshold)
- `alpha3` (Splitting former Merged Threshold)
- `alpha4` (Splitting former Merged Threshold)

Required packages: `CHAID`

Conditional Inference Random Forest

```
method = 'cforest'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `party`

A model-specific variable importance metric is available.

Conditional Inference Tree

```
method = 'ctree'
```

Type: Classification, Regression

Tuning parameters:

- `mincriterion` (1 - P-Value Threshold)

Required packages: `party`

Conditional Inference Tree

```
method = 'ctree2'
```

Type: Regression, Classification

Tuning parameters:

- `maxdepth` (Max Tree Depth)
- `mincriterion` (1 - P-Value Threshold)

Required packages: `party`

Cost-Sensitive C5.0

```
method = 'C5.0Cost'
```

Type: Classification

Tuning parameters:

- `trials` (# Boosting Iterations)
- `model` (Model Type)
- `winnow` (Winnow)
- `cost` (Cost)

Required packages: `C50` , `plyr`

A model-specific variable importance metric is available.

Cost-Sensitive CART

```
method = 'rpartCost'
```

Type: Classification

Tuning parameters:

- `cp` (Complexity Parameter)
- `Cost` (Cost)

Required packages: `rpart` , `plyr`

Cubist

```
method = 'cubist'
```

Type: Regression

Tuning parameters:

- `committees` (#Committees)
- `neighbors` (#Instances)

Required packages: `Cubist`

A model-specific variable importance metric is available.

DeepBoost

```
method = 'deepboost'
```

Type: Classification

Tuning parameters:

- `num_iter` (# Boosting Iterations)
- `tree_depth` (Tree Depth)
- `beta` (L1 Regularization)
- `lambda` (Tree Depth Regularization)
- `loss_type` (Loss)

Required packages: `deepboost`

Elasticnet

```
method = 'enet'
```

Type: Regression

Tuning parameters:

- `fraction` (Fraction of Full Solution)

- `lambda` (Weight Decay)

Required packages: `elasticnet`

eXtreme Gradient Boosting

```
method = 'xgbDART'
```

Type: Regression, Classification

Tuning parameters:

- `nrounds` (# Boosting Iterations)
- `max_depth` (Max Tree Depth)
- `eta` (Shrinkage)
- `gamma` (Minimum Loss Reduction)
- `subsample` (Subsample Percentage)
- `colsample_bytree` (Subsample Ratio of Columns)
- `rate_drop` (Fraction of Trees Dropped)
- `skip_drop` (Prob. of Skipping Drop-out)
- `min_child_weight` (Minimum Sum of Instance Weight)

Required packages: `xgboost` , `plyr`

A model-specific variable importance metric is available.

eXtreme Gradient Boosting

```
method = 'xgbLinear'
```

Type: Regression, Classification

Tuning parameters:

- `nrounds` (# Boosting Iterations)
- `lambda` (L2 Regularization)
- `alpha` (L1 Regularization)
- `eta` (Learning Rate)

Required packages: `xgboost`

A model-specific variable importance metric is available.

eXtreme Gradient Boosting

```
method = 'xgbTree'
```

Type: Regression, Classification

Tuning parameters:

- `nrounds` (# Boosting Iterations)
- `max_depth` (Max Tree Depth)
- `eta` (Shrinkage)
- `gamma` (Minimum Loss Reduction)
- `colsample_bytree` (Subsample Ratio of Columns)
- `min_child_weight` (Minimum Sum of Instance Weight)
- `subsample` (Subsample Percentage)

Required packages: `xgboost` , `plyr`

A model-specific variable importance metric is available.

Flexible Discriminant Analysis

```
method = 'fda'
```

Type: Classification

Tuning parameters:

- `degree` (Product Degree)
- `nprune` (#Terms)

Required packages: `earth` , `mda`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train` , the `earth` package is fully loaded when this model is used.

Generalized Linear Model with Stepwise Feature Selection

```
method = 'glmStepAIC'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `MASS`

glmnet

```
method = 'glmnet_h2o'
```

Type: Regression, Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `lambda` (Regularization Parameter)

Required packages: `h2o`

A model-specific variable importance metric is available.

glmnet

```
method = 'glmnet'
```

Type: Regression, Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `lambda` (Regularization Parameter)

Required packages: `glmnet` , `Matrix`

A model-specific variable importance metric is available.

Gradient Boosting Machines

```
method = 'gbm_h2o'
```

Type: Regression, Classification

Tuning parameters:

- `ntrees` (# Boosting Iterations)

- `max_depth` (Max Tree Depth)
- `min_rows` (Min. Terminal Node Size)
- `learn_rate` (Shrinkage)
- `col_sample_rate` (#Randomly Selected Predictors)

Required packages: `h2o`

A model-specific variable importance metric is available.

Least Angle Regression

```
method = 'lars'
```

Type: Regression

Tuning parameters:

- `fraction` (Fraction)

Required packages: `lars`

Least Angle Regression

```
method = 'lars2'
```

Type: Regression

Tuning parameters:

- `step` (#Steps)

Required packages: `lars`

Logistic Model Trees

```
method = 'LMT'
```

Type: Classification

Tuning parameters:

- `iter` (# Iteratons)

Required packages: `RWeka`

Model Rules

```
method = 'M5Rules'
```

Type: Regression

Tuning parameters:

- `pruned` (Pruned)
- `smoothed` (Smoothed)

Required packages: `RWeka`

Model Tree

```
method = 'M5'
```

Type: Regression

Tuning parameters:

- `pruned` (Pruned)
- `smoothed` (Smoothed)
- `rules` (Rules)

Required packages: `RWeka`

Multi-Step Adaptive MCP-Net

```
method = 'msaenet'
```

Type: Regression, Classification

Tuning parameters:

- `alphas` (Alpha)
- `nsteps` (#Adaptive Estimation Steps)
- `scale` (Adaptive Weight Scaling Factor)

Required packages: `msaenet`

A model-specific variable importance metric is available.

Multivariate Adaptive Regression Spline

```
method = 'earth'
```

Type: Regression, Classification

Tuning parameters:

- `nprune` (#Terms)
- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Multivariate Adaptive Regression Splines

```
method = 'gcvEarth'
```

Type: Regression, Classification

Tuning parameters:

- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Nearest Shrunken Centroids

```
method = 'pam'
```

Type: Classification

Tuning parameters:

- `threshold` (Shrinkage Threshold)

Required packages: `pamr`

A model-specific variable importance metric is available.

Non-Convex Penalized Quantile Regression

```
method = 'rqnc'
```

Type: Regression

Tuning parameters:

- `lambda` (L1 Penalty)
- `penalty` (Penalty Type)

Required packages: `rqPen`

Oblique Random Forest

```
method = 'ORFlog'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Oblique Random Forest

```
method = 'ORFpls'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Oblique Random Forest

```
method = 'ORFridge'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Oblique Random Forest

```
method = 'ORFsvm'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Parallel Random Forest

```
method = 'parRF'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `e1071`, `randomForest`, `foreach`, `import`

A model-specific variable importance metric is available.

Penalized Linear Discriminant Analysis

```
method = 'PenalizedLDA'
```

Type: Classification

Tuning parameters:

- `lambda` (L1 Penalty)
- `K` (#Discriminant Functions)

Required packages: `penalizedLDA` , `plyr`

Penalized Linear Regression

```
method = 'penalized'
```

Type: Regression

Tuning parameters:

- `lambda1` (L1 Penalty)
- `lambda2` (L2 Penalty)

Required packages: `penalized`

Penalized Ordinal Regression

```
method = 'ordinalNet'
```

Type: Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `criteria` (Selection Criterion)

- `link` (Link Function)

Required packages: `ordinalNet` , `plyr`

A model-specific variable importance metric is available. Notes: Requires ordinalNet package version >= 2.0

Quantile Random Forest

```
method = 'qrf'
```

Type: Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `quantregForest`

Quantile Regression with LASSO penalty

```
method = 'rqlasso'
```

Type: Regression

Tuning parameters:

- `lambda` (L1 Penalty)

Required packages: `rqPen`

Random Ferns

```
method = 'rFerns'
```

Type: Classification

Tuning parameters:

- `depth` (Fern Depth)

Required packages: `rFerns`

Random Forest

```
method = 'ranger'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `splitrule` (Splitting Rule)
- `min.node.size` (Minimal Node Size)

Required packages: `e1071` , `ranger` , `dplyr`

A model-specific variable importance metric is available.

Random Forest

```
method = 'Rborist'
```

Type: Classification, Regression

Tuning parameters:

- `predFixed` (#Randomly Selected Predictors)
- `minNode` (Minimal Node Size)

Required packages: `Rborist`

A model-specific variable importance metric is available.

Random Forest

```
method = 'rf'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `randomForest`

A model-specific variable importance metric is available.

Random Forest by Randomization

```
method = 'extraTrees'
```

Type: Regression, Classification

Tuning parameters:

- `mtry` (# Randomly Selected Predictors)
- `numRandomCuts` (# Random Cuts)

Required packages: `extraTrees`

Random Forest Rule-Based Model

```
method = 'rfRules'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `maxdepth` (Maximum Rule Depth)

Required packages: `randomForest` , `inTrees` , `plyr`

A model-specific variable importance metric is available.

Regularized Random Forest

```
method = 'RRF'
```

Type: Regression, Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `coefReg` (Regularization Value)
- `coefImp` (Importance Coefficient)

Required packages: `randomForest` , `RRF`

A model-specific variable importance metric is available.

Regularized Random Forest

```
method = 'RRFglobal'
```

Type: Regression, Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `coefReg` (Regularization Value)

Required packages: `RRF`

A model-specific variable importance metric is available.

Relaxed Lasso

```
method = 'relaxo'
```

Type: Regression

Tuning parameters:

- `lambda` (Penalty Parameter)
- `phi` (Relaxation Parameter)

Required packages: `relaxo` , `plyr`

Rotation Forest

```
method = 'rotationForest'
```

Type: Classification

Tuning parameters:

- `K` (#Variable Subsets)
- `L` (Ensemble Size)

Required packages: `rotationForest`

A model-specific variable importance metric is available.

Rotation Forest

```
method = 'rotationForestCp'
```

Type: Classification

Tuning parameters:

- `K` (#Variable Subsets)
- `L` (Ensemble Size)
- `cp` (Complexity Parameter)

Required packages: `rpart` , `plyr` , `rotationForest`

A model-specific variable importance metric is available.

Rule-Based Classifier

```
method = 'JRip'
```

Type: Classification

Tuning parameters:

- `NumOpt` (# Optimizations)
- `NumFolds` (# Folds)
- `MinWeights` (Min Weights)

Required packages: `RWeka`

A model-specific variable importance metric is available.

Rule-Based Classifier

```
method = 'PART'
```

Type: Classification

Tuning parameters:

- `threshold` (Confidence Threshold)
- `pruned` (Pruning)

Required packages: `RWeka`

A model-specific variable importance metric is available.

Single C5.0 Ruleset

```
method = 'C5.0Rules'
```

Type: Classification

No tuning parameters for this model

Required packages: `C50`

A model-specific variable importance metric is available.

Single C5.0 Tree

```
method = 'C5.0Tree'
```

Type: Classification

No tuning parameters for this model

Required packages: `C50`

A model-specific variable importance metric is available.

Single Rule Classification

```
method = 'OneR'
```

Type: Classification

No tuning parameters for this model

Required packages: `RWeka`

Sparse Distance Weighted Discrimination

```
method = 'sdwd'
```

Type: Classification

Tuning parameters:

- `lambda` (L1 Penalty)
- `lambda2` (L2 Penalty)

Required packages: `sdwd`

A model-specific variable importance metric is available.

Sparse Linear Discriminant Analysis

```
method = 'sparseLDA'
```

Type: Classification

Tuning parameters:

- `NumVars` (# Predictors)
- `lambda` (Lambda)

Required packages: `sparseLDA`

Sparse Mixture Discriminant Analysis

```
method = 'smda'
```

Type: Classification

Tuning parameters:

- `NumVars` (# Predictors)
- `lambda` (Lambda)

- `R` (# Subclasses)

Required packages: `sparseLDA`

Spike and Slab Regression

```
method = 'spikeslab'
```

Type: Regression

Tuning parameters:

- `vars` (Variables Retained)

Required packages: `spikeslab` , `plyr`

Notes: Unlike other packages used by `train` , the `spikeslab` package is fully loaded when this model is used.

Stochastic Gradient Boosting

```
method = 'gbm'
```

Type: Regression, Classification

Tuning parameters:

- `n.trees` (# Boosting Iterations)
- `interaction.depth` (Max Tree Depth)
- `shrinkage` (Shrinkage)
- `n.minobsinnode` (Min. Terminal Node Size)

Required packages: `gbm` , `plyr`

A model-specific variable importance metric is available.

The Bayesian lasso

```
method = 'blasso'
```

Type: Regression

Tuning parameters:

- `sparsity` (Sparsity Threshold)

Required packages: `monomvn`

Notes: This model creates predictions using the mean of the posterior distributions but sets some parameters specifically to zero based on the tuning parameter `sparsity` . For example, when `sparsity = .5` , only coefficients where at least half the posterior estimates are nonzero are used.

The lasso

```
method = 'lasso'
```

Type: Regression

Tuning parameters:

- `fraction` (Fraction of Full Solution)

Required packages: `elasticnet`

Tree Models from Genetic Algorithms

```
method = 'evtree'
```

Type: Regression, Classification

Tuning parameters:

- `alpha` (Complexity Parameter)

Required packages: `evtree`

Tree-Based Ensembles

```
method = 'nodeHarvest'
```

Type: Regression, Classification

Tuning parameters:

- `maxinter` (Maximum Interaction Depth)
- `mode` (Prediction Mode)

Required packages: `nodeHarvest`

Weighted Subspace Random Forest

```
method = 'wsrf'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `wsrf`

7.0.18 Kernel Method

(back to contents)

Distance Weighted Discrimination with Polynomial Kernel

```
method = 'dwdPoly'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)
- `degree` (Polynomial Degree)
- `scale` (Scale)

Required packages: `kerndwd`

Distance Weighted Discrimination with Radial Basis Function Kernel

```
method = 'dwdRadial'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)
- `sigma` (Sigma)

Required packages: `kernlab` , `kerndwd`

Gaussian Process

```
method = 'gausssprLinear'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `kernlab`

Gaussian Process with Polynomial Kernel

```
method = 'gausssprPoly'
```

Type: Regression, Classification

Tuning parameters:

- `degree` (Polynomial Degree)
- `scale` (Scale)

Required packages: `kernlab`

Gaussian Process with Radial Basis Function Kernel

```
method = 'gausssprRadial'
```

Type: Regression, Classification

Tuning parameters:

- `sigma` (Sigma)

Required packages: `kernlab`

L2 Regularized Linear Support Vector Machines with Class Weights

```
method = 'svmLinearWeights2'
```

Type: Classification

Tuning parameters:

- `cost` (Cost)
- `Loss` (Loss Function)
- `weight` (Class Weight)

Required packages: `LiblineaR`

L2 Regularized Support Vector Machine (dual) with Linear Kernel

```
method = 'svmLinear3'
```

Type: Regression, Classification

Tuning parameters:

- `cost` (Cost)
- `Loss` (Loss Function)

Required packages: `LiblineaR`

Least Squares Support Vector Machine

```
method = 'lssvmLinear'
```

Type: Classification

Tuning parameters:

- `tau` (Regularization Parameter)

Required packages: `kernlab`

Least Squares Support Vector Machine with Polynomial Kernel

```
method = 'lssvmPoly'
```

Type: Classification

Tuning parameters:

- `degree` (Polynomial Degree)
- `scale` (Scale)
- `tau` (Regularization Parameter)

Required packages: `kernlab`

Least Squares Support Vector Machine with Radial Basis Function Kernel

```
method = 'lssvmRadial'
```

Type: Classification

Tuning parameters:

- `sigma` (Sigma)
- `tau` (Regularization Parameter)

Required packages: `kernlab`

Linear Distance Weighted Discrimination

```
method = 'dwdLinear'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)

Required packages: `kerndwd`

Linear Support Vector Machines with Class Weights

```
method = 'svmLinearWeights'
```

Type: Classification

Tuning parameters:

- `cost` (Cost)
- `weight` (Class Weight)

Required packages: `e1071`

Oblique Random Forest

```
method = 'ORFsvm'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train` , the `obliqueRF` package is fully loaded when this model is used.

Partial Least Squares

```
method = 'kernelpls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Polynomial Kernel Regularized Least Squares

```
method = 'krlsPoly'
```

Type: Regression

Tuning parameters:

- `lambda` (Regularization Parameter)
- `degree` (Polynomial Degree)

Required packages: `KRLS`

Radial Basis Function Kernel Regularized Least Squares

```
method = 'krlsRadial'
```

Type: Regression

Tuning parameters:

- `lambda` (Regularization Parameter)
- `sigma` (Sigma)

Required packages: `KRLS` , `kernlab`

Relevance Vector Machines with Linear Kernel

```
method = 'rvmLinear'
```

Type: Regression

No tuning parameters for this model

Required packages: `kernlab`

Relevance Vector Machines with Polynomial Kernel

```
method = 'rvmPoly'
```

Type: Regression

Tuning parameters:

- `scale` (Scale)
- `degree` (Polynomial Degree)

Required packages: `kernlab`

Relevance Vector Machines with Radial Basis Function Kernel

```
method = 'rvmRadial'
```

Type: Regression

Tuning parameters:

- `sigma` (Sigma)

Required packages: `kernlab`

Support Vector Machines with Boundrange String Kernel

```
method = 'svmBoundrangeString'
```

Type: Regression, Classification

Tuning parameters:

- `length` (length)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Class Weights

```
method = 'svmRadialWeights'
```

Type: Classification

Tuning parameters:

- `sigma` (Sigma)
- `C` (Cost)
- `Weight` (Weight)

Required packages: `kernlab`

Support Vector Machines with Exponential String Kernel

```
method = 'svmExpoString'
```

Type: Regression, Classification

Tuning parameters:

- `lambda` (lambda)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Linear Kernel

```
method = 'svmLinear'
```

Type: Regression, Classification

Tuning parameters:

- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Linear Kernel

```
method = 'svmLinear2'
```

Type: Regression, Classification

Tuning parameters:

- `cost` (Cost)

Required packages: `e1071`

Support Vector Machines with Polynomial Kernel

```
method = 'svmPoly'
```

Type: Regression, Classification

Tuning parameters:

- `degree` (Polynomial Degree)
- `scale` (Scale)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Radial Basis Function Kernel

```
method = 'svmRadial'
```

Type: Regression, Classification

Tuning parameters:

- `sigma` (Sigma)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Radial Basis Function Kernel

```
method = 'svmRadialCost'
```

Type: Regression, Classification

Tuning parameters:

- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Radial Basis Function Kernel

```
method = 'svmRadialSigma'
```

Type: Regression, Classification

Tuning parameters:

- `sigma` (Sigma)
- `C` (Cost)

Required packages: `kernlab`

Notes: This SVM model tunes over the cost parameter and the RBF kernel parameter sigma. In the latter case, using `tuneLength` will, at most, evaluate six values of the kernel parameter. This enables a broad search over the cost parameter and a relatively narrow search over `sigma`

Support Vector Machines with Spectrum String Kernel

```
method = 'svmSpectrumString'
```

Type: Regression, Classification

Tuning parameters:

- `length` (length)
- `C` (Cost)

Required packages: `kernlab`

7.0.19 L1 Regularization

(back to contents)

Bayesian Ridge Regression (Model Averaged)

```
method = 'blassoAveraged'
```

Type: Regression

No tuning parameters for this model

Required packages: `monomvn`

Notes: This model makes predictions by averaging the predictions based on the posterior estimates of the regression coefficients. While it is possible that some of these posterior estimates are zero for non-informative predictors, the final predicted value may be a function of many (or even all) predictors.

DeepBoost

```
method = 'deepboost'
```

Type: Classification

Tuning parameters:

- `num_iter` (# Boosting Iterations)
- `tree_depth` (Tree Depth)
- `beta` (L1 Regularization)
- `lambda` (Tree Depth Regularization)
- `loss_type` (Loss)

Required packages: `deepboost`

Elasticnet

```
method = 'enet'
```

Type: Regression

Tuning parameters:

- `fraction` (Fraction of Full Solution)
- `lambda` (Weight Decay)

Required packages: `elasticnet`

glmnet

```
method = 'glmnet_h2o'
```

Type: Regression, Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `lambda` (Regularization Parameter)

Required packages: `h2o`

A model-specific variable importance metric is available.

glmnet

```
method = 'glmnet'
```

Type: Regression, Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `lambda` (Regularization Parameter)

Required packages: `glmnet` , `Matrix`

A model-specific variable importance metric is available.

Least Angle Regression

```
method = 'lars'
```

Type: Regression

Tuning parameters:

- `fraction` (Fraction)

Required packages: `lars`

Least Angle Regression

```
method = 'lars2'
```

Type: Regression

Tuning parameters:

- `step` (#Steps)

Required packages: `lars`

Multi-Step Adaptive MCP-Net

```
method = 'msaenet'
```

Type: Regression, Classification

Tuning parameters:

- `alphas` (Alpha)
- `nsteps` (#Adaptive Estimation Steps)
- `scale` (Adaptive Weight Scaling Factor)

Required packages: `msaenet`

A model-specific variable importance metric is available.

Non-Convex Penalized Quantile Regression

```
method = 'rqnc'
```

Type: Regression

Tuning parameters:

- `lambda` (L1 Penalty)
- `penalty` (Penalty Type)

Required packages: `rqPen`

Penalized Linear Discriminant Analysis

```
method = 'PenalizedLDA'
```

Type: Classification

Tuning parameters:

- `lambda` (L1 Penalty)
- `K` (#Discriminant Functions)

Required packages: `penalizedLDA` , `plyr`

Penalized Linear Regression

```
method = 'penalized'
```

Type: Regression

Tuning parameters:

- `lambda1` (L1 Penalty)
- `lambda2` (L2 Penalty)

Required packages: `penalized`

Penalized Ordinal Regression

```
method = 'ordinalNet'
```

Type: Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `criteria` (Selection Criterion)
- `link` (Link Function)

Required packages: `ordinalNet` , `plyr`

A model-specific variable importance metric is available. Notes: Requires ordinalNet package version >= 2.0

Quantile Regression with LASSO penalty

```
method = 'rqlasso'
```

Type: Regression

Tuning parameters:

- `lambda` (L1 Penalty)

Required packages: `rqPen`

Regularized Logistic Regression

```
method = 'regLogistic'
```

Type: Classification

Tuning parameters:

- `cost` (Cost)
- `loss` (Loss Function)
- `epsilon` (Tolerance)

Required packages: `LiblineaR`

Relaxed Lasso

```
method = 'relaxo'
```

Type: Regression

Tuning parameters:

- `lambda` (Penalty Parameter)
- `phi` (Relaxation Parameter)

Required packages: `relaxo` , `plyr`

Sparse Distance Weighted Discrimination

```
method = 'sdwd'
```

Type: Classification

Tuning parameters:

- `lambda` (L1 Penalty)
- `lambda2` (L2 Penalty)

Required packages: `sdwd`

A model-specific variable importance metric is available.

Sparse Linear Discriminant Analysis

```
method = 'sparseLDA'
```

Type: Classification

Tuning parameters:

- `NumVars` (# Predictors)
- `lambda` (Lambda)

Required packages: `sparseLDA`

Sparse Mixture Discriminant Analysis

```
method = 'smda'
```

Type: Classification

Tuning parameters:

- `NumVars` (# Predictors)
- `lambda` (Lambda)
- `R` (# Subclasses)

Required packages: `sparseLDA`

Sparse Partial Least Squares

```
method = 'spls'
```

Type: Regression, Classification

Tuning parameters:

- `K` (#Components)
- `eta` (Threshold)
- `kappa` (Kappa)

Required packages: `spls`

The Bayesian lasso

```
method = 'blasso'
```

Type: Regression

Tuning parameters:

- `sparsity` (Sparsity Threshold)

Required packages: `monomvn`

Notes: This model creates predictions using the mean of the posterior distributions but sets some parameters specifically to zero based on the tuning parameter `sparsity`. For example, when `sparsity = .5`, only coefficients where at least half the posterior estimates are nonzero are used.

The lasso

```
method = 'lasso'
```

Type: Regression

Tuning parameters:

- `fraction` (Fraction of Full Solution)

Required packages: `elasticnet`

7.0.20 L2 Regularization

(back to contents)

Bayesian Ridge Regression

```
method = 'bridge'
```

Type: Regression

No tuning parameters for this model

Required packages: `monomvn`

Distance Weighted Discrimination with Polynomial Kernel

```
method = 'dwdPoly'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)
- `degree` (Polynomial Degree)
- `scale` (Scale)

Required packages: `kerndwd`

Distance Weighted Discrimination with Radial Basis Function Kernel

```
method = 'dwdRadial'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)

- `sigma` (Sigma)

Required packages: `kernlab` , `kerndwd`

glmnet

```
method = 'glmnet_h2o'
```

Type: Regression, Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `lambda` (Regularization Parameter)

Required packages: `h2o`

A model-specific variable importance metric is available.

glmnet

```
method = 'glmnet'
```

Type: Regression, Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `lambda` (Regularization Parameter)

Required packages: `glmnet` , `Matrix`

A model-specific variable importance metric is available.

Linear Distance Weighted Discrimination

```
method = 'dwdLinear'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)

Required packages: `kerndwd`

Model Averaged Neural Network

```
method = 'avNNet'
```

Type: Classification, Regression

Tuning parameters:

- `size` (#Hidden Units)
- `decay` (Weight Decay)
- `bag` (Bagging)

Required packages: `nnet`

Multi-Layer Perceptron

```
method = 'mlpWeightDecay'
```

Type: Regression, Classification

Tuning parameters:

- `size` (#Hidden Units)
- `decay` (Weight Decay)

Required packages: `RSNNS`

Multi-Layer Perceptron, multiple layers

```
method = 'mlpWeightDecayML'
```

Type: Regression, Classification

Tuning parameters:

- `layer1` (#Hidden Units layer1)
- `layer2` (#Hidden Units layer2)
- `layer3` (#Hidden Units layer3)
- `decay` (Weight Decay)

Required packages: `RSNNS`

Multi-Step Adaptive MCP-Net

```
method = 'msaenet'
```

Type: Regression, Classification

Tuning parameters:

- `alphas` (Alpha)
- `nsteps` (#Adaptive Estimation Steps)
- `scale` (Adaptive Weight Scaling Factor)

Required packages: `msaenet`

A model-specific variable importance metric is available.

Multilayer Perceptron Network by Stochastic Gradient Descent

```
method = 'mlpSGD'
```

Type: Regression, Classification

Tuning parameters:

- `size` (#Hidden Units)
- `l2reg` (L2 Regularization)
- `lambda` (RMSE Gradient Scaling)
- `learn_rate` (Learning Rate)
- `momentum` (Momentum)
- `gamma` (Learning Rate Decay)
- `minibatchsz` (Batch Size)
- `repeats` (#Models)

Required packages: `FCNN4R` , `plyr`

A model-specific variable importance metric is available.

Multilayer Perceptron Network with Weight Decay

```
method = 'mlpKerasDecay'
```

Type: Regression, Classification

Tuning parameters:

- `size` (#Hidden Units)
- `lambda` (L2 Regularization)
- `batch_size` (Batch Size)
- `lr` (Learning Rate)
- `rho` (Rho)
- `decay` (Learning Rate Decay)
- `activation` (Activation Function)

Required packages: `keras`

Notes: After `train` completes, the keras model object is serialized so that it can be used between R session. When predicting, the code will temporarily unsearalize the object. To make the predictions more efficient, the user might want to use `keras::unsearlize_model(object$finalModel$object)` in the current R session so that that operation is only done once. Also, this model cannot be run in parallel due to the nature of how tensorflow does the computations. Unlike other packages used by `train`, the `dplyr` package is fully loaded when this model is used.

Multilayer Perceptron Network with Weight Decay

```
method = 'mlpKerasDecayCost'
```

Type: Classification

Tuning parameters:

- `size` (#Hidden Units)
- `lambda` (L2 Regularization)
- `batch_size` (Batch Size)
- `lr` (Learning Rate)
- `rho` (Rho)
- `decay` (Learning Rate Decay)
- `cost` (Cost)
- `activation` (Activation Function)

Required packages: `keras`

Notes: After `train` completes, the keras model object is serialized so that it can be used between R session. When predicting, the code will temporarily unsearalize the object. To make the predictions more efficient, the user might want to use `keras::unsearlize_model(object$finalModel$object)` in the current R session so that that operation is only done once. Also, this model cannot be run in parallel due to the nature of how tensorflow does the computations. Finally, the cost parameter weights the first class in the outcome vector. Unlike other packages used by `train`, the `dplyr` package is fully loaded when this model is used.

Neural Network

```
method = 'nnet'
```

Type: Classification, Regression

Tuning parameters:

- `size` (#Hidden Units)
- `decay` (Weight Decay)

Required packages: `nnet`

A model-specific variable importance metric is available.

Neural Networks with Feature Extraction

```
method = 'pcaNNet'
```

Type: Classification, Regression

Tuning parameters:

- `size` (#Hidden Units)
- `decay` (Weight Decay)

Required packages: `nnet`

Oblique Random Forest

```
method = 'ORFridge'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train` , the `obliqueRF` package is fully loaded when this model is used.

Penalized Linear Regression

```
method = 'penalized'
```

Type: Regression

Tuning parameters:

- `lambda1` (L1 Penalty)
- `lambda2` (L2 Penalty)

Required packages: `penalized`

Penalized Logistic Regression

```
method = 'plr'
```

Type: Classification

Tuning parameters:

- `lambda` (L2 Penalty)
- `cp` (Complexity Parameter)

Required packages: `stepPlr`

Penalized Multinomial Regression

```
method = 'multinom'
```

Type: Classification

Tuning parameters:

- `decay` (Weight Decay)

Required packages: `nnet`

A model-specific variable importance metric is available.

Penalized Ordinal Regression

```
method = 'ordinalNet'
```

Type: Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `criteria` (Selection Criterion)
- `link` (Link Function)

Required packages: `ordinalNet` , `plyr`

A model-specific variable importance metric is available. Notes:
Requires ordinalNet package version >= 2.0

Polynomial Kernel Regularized Least Squares

```
method = 'krlsPoly'
```

Type: Regression

Tuning parameters:

- `lambda` (Regularization Parameter)
- `degree` (Polynomial Degree)

Required packages: `KRLS`

Quantile Regression Neural Network

```
method = 'qrnn'
```

Type: Regression

Tuning parameters:

- `n.hidden` (#Hidden Units)
- `penalty` (Weight Decay)
- `bag` (Bagged Models?)

Required packages: `qrnn`

Radial Basis Function Kernel Regularized Least Squares

```
method = 'krlsRadial'
```

Type: Regression

Tuning parameters:

- `lambda` (Regularization Parameter)
- `sigma` (Sigma)

Required packages: `KRLS` , `kernlab`

Radial Basis Function Network

```
method = 'rbf'
```

Type: Classification, Regression

Tuning parameters:

- `size` (#Hidden Units)

Required packages: `RSNNS`

Radial Basis Function Network

```
method = 'rbfDDA'
```

Type: Regression, Classification

Tuning parameters:

- `negativeThreshold` (Activation Limit for Conflicting Classes)

Required packages: `RSNNS`

Regularized Logistic Regression

```
method = 'regLogistic'
```

Type: Classification

Tuning parameters:

- `cost` (Cost)
- `loss` (Loss Function)
- `epsilon` (Tolerance)

Required packages: `LiblineaR`

Relaxed Lasso

```
method = 'relaxo'
```

Type: Regression

Tuning parameters:

- `lambda` (Penalty Parameter)
- `phi` (Relaxation Parameter)

Required packages: `relaxo` , `plyr`

Ridge Regression

```
method = 'ridge'
```

Type: Regression

Tuning parameters:

- `lambda` (Weight Decay)

Required packages: `elasticnet`

Ridge Regression with Variable Selection

```
method = 'foba'
```

Type: Regression

Tuning parameters:

- `k` (#Variables Retained)
- `lambda` (L2 Penalty)

Required packages: `foba`

Sparse Distance Weighted Discrimination

```
method = 'sdwd'
```

Type: Classification

Tuning parameters:

- `lambda` (L1 Penalty)
- `lambda2` (L2 Penalty)

Required packages: `sdwd`

A model-specific variable importance metric is available.

7.0.21 Linear Classifier

(back to contents)

Adjacent Categories Probability Model for Ordinal Data

```
method = 'vglmAdjCat'
```

Type: Classification

Tuning parameters:

- `parallel` (Parallel Curves)
- `link` (Link Function)

Required packages: `VGAM`

Bagged Logic Regression

```
method = 'logicBag'
```

Type: Regression, Classification

Tuning parameters:

- `nleaves` (Maximum Number of Leaves)
- `ntrees` (Number of Trees)

Required packages: `logicFS`

Notes: Unlike other packages used by `train`, the `logicFS` package is fully loaded when this model is used.

Bayesian Generalized Linear Model

```
method = 'bayesglm'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `arm`

Boosted Generalized Linear Model

```
method = 'glmboost'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `prune` (AIC Prune?)

Required packages: `plyr`, `mboost`

A model-specific variable importance metric is available. Notes: The `prune` option for this model enables the number of iterations to be determined by the optimal AIC value across all iterations. See the

examples in `?mboost::mstop`. If pruning is not used, the ensemble makes predictions using the exact value of the `mstop` tuning parameter value.

Continuation Ratio Model for Ordinal Data

```
method = 'vglmContRatio'
```

Type: Classification

Tuning parameters:

- `parallel` (Parallel Curves)
- `link` (Link Function)

Required packages: `VGAM`

Cumulative Probability Model for Ordinal Data

```
method = 'vglmCumulative'
```

Type: Classification

Tuning parameters:

- `parallel` (Parallel Curves)
- `link` (Link Function)

Required packages: `VGAM`

Diagonal Discriminant Analysis

```
method = 'dda'
```

Type: Classification

Tuning parameters:

- `model` (Model)
- `shrinkage` (Shrinkage Type)

Required packages: `sparsediscrim`

Ensembles of Generalized Linear Models

```
method = 'randomGLM'
```

Type: Regression, Classification

Tuning parameters:

- `maxInteractionOrder` (Interaction Order)

Required packages: `randomGLM`

Notes: Unlike other packages used by `train` , the `randomGLM` package is fully loaded when this model is used.

Factor-Based Linear Discriminant Analysis

```
method = 'RFlda'
```

Type: Classification

Tuning parameters:

- `q` (# Factors)

Required packages: `HiDimDA`

Gaussian Process

```
method = 'gaussprLinear'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `kernlab`

Generalized Linear Model

```
method = 'glm'
```

Type: Regression, Classification

No tuning parameters for this model

A model-specific variable importance metric is available.

Generalized Linear Model with Stepwise Feature Selection

```
method = 'glmStepAIC'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `MASS`

Generalized Partial Least Squares

```
method = 'gpls'
```

Type: Classification

Tuning parameters:

- `K.prov` (#Components)

Required packages: `gpls`

glmnet

```
method = 'glmnet_h2o'
```

Type: Regression, Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `lambda` (Regularization Parameter)

Required packages: `h2o`

A model-specific variable importance metric is available.

glmnet

```
method = 'glmnet'
```

Type: Regression, Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `lambda` (Regularization Parameter)

Required packages: `glmnet` , `Matrix`

A model-specific variable importance metric is available.

Heteroscedastic Discriminant Analysis

```
method = 'hda'
```

Type: Classification

Tuning parameters:

- `gamma` (Gamma)
- `lambda` (Lambda)
- `newdim` (Dimension of the Discriminative Subspace)

Required packages: `hda`

High Dimensional Discriminant Analysis

```
method = 'hdda'
```

Type: Classification

Tuning parameters:

- `threshold` (Threshold)
- `model` (Model Type)

Required packages: `HDclassif`

High-Dimensional Regularized Discriminant Analysis

```
method = 'hdrda'
```

Type: Classification

Tuning parameters:

- `gamma` (Gamma)
- `lambda` (Lambda)
- `shrinkage_type` (Shrinkage Type)

Required packages: `sparsediscrim`

L2 Regularized Linear Support Vector Machines with Class Weights

```
method = 'svmLinearWeights2'
```

Type: Classification

Tuning parameters:

- `cost` (Cost)
- `Loss` (Loss Function)
- `weight` (Class Weight)

Required packages: `LiblineaR`

L2 Regularized Support Vector Machine (dual) with Linear Kernel

```
method = 'svmLinear3'
```

Type: Regression, Classification

Tuning parameters:

- `cost` (Cost)
- `Loss` (Loss Function)

Required packages: `LiblineaR`

Least Squares Support Vector Machine

```
method = 'lssvmLinear'
```

Type: Classification

Tuning parameters:

- `tau` (Regularization Parameter)

Required packages: `kernlab`

Linear Discriminant Analysis

```
method = 'lda'
```

Type: Classification

No tuning parameters for this model

Required packages: `MASS`

Linear Discriminant Analysis

```
method = 'lda2'
```

Type: Classification

Tuning parameters:

- `dimen` (#Discriminant Functions)

Required packages: `MASS`

Linear Discriminant Analysis with Stepwise Feature Selection

```
method = 'stepLDA'
```

Type: Classification

Tuning parameters:

- `maxvar` (Maximum #Variables)
- `direction` (Search Direction)

Required packages: `klaR` , `MASS`

Linear Distance Weighted Discrimination

```
method = 'dwdLinear'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)

Required packages: `kerndwd`

Linear Support Vector Machines with Class Weights

```
method = 'svmLinearWeights'
```

Type: Classification

Tuning parameters:

- `cost` (Cost)
- `weight` (Class Weight)

Required packages: `e1071`

Localized Linear Discriminant Analysis

```
method = 'loclda'
```

Type: Classification

Tuning parameters:

- `k` (#Nearest Neighbors)

Required packages: `klaR`

Logic Regression

```
method = 'logreg'
```

Type: Regression, Classification

Tuning parameters:

- `treesize` (Maximum Number of Leaves)
- `ntrees` (Number of Trees)

Required packages: `LogicReg`

Logistic Model Trees

```
method = 'LMT'
```

Type: Classification

Tuning parameters:

- `iter` (# Iteratons)

Required packages: `RWeka`

Maximum Uncertainty Linear Discriminant Analysis

```
method = 'Mlda'
```

Type: Classification

No tuning parameters for this model

Required packages: `HiDimDA`

Multi-Step Adaptive MCP-Net

```
method = 'msaenet'
```

Type: Regression, Classification

Tuning parameters:

- `alphas` (Alpha)
- `nsteps` (#Adaptive Estimation Steps)
- `scale` (Adaptive Weight Scaling Factor)

Required packages: `msaenet`

A model-specific variable importance metric is available.

Nearest Shrunken Centroids

```
method = 'pam'
```

Type: Classification

Tuning parameters:

- `threshold` (Shrinkage Threshold)

Required packages: `pamr`

A model-specific variable importance metric is available.

Ordered Logistic or Probit Regression

```
method = 'polr'
```

Type: Classification

Tuning parameters:

- `method` (parameter)

Required packages: `MASS`

A model-specific variable importance metric is available.

Partial Least Squares

```
method = 'kernelpls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Partial Least Squares

```
method = 'pls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Partial Least Squares

```
method = 'simpls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Partial Least Squares

```
method = 'widekernelpls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Penalized Linear Discriminant Analysis

```
method = 'PenalizedLDA'
```

Type: Classification

Tuning parameters:

- `lambda` (L1 Penalty)
- `K` (#Discriminant Functions)

Required packages: `penalizedLDA` , `plyr`

Penalized Logistic Regression

```
method = 'plr'
```

Type: Classification

Tuning parameters:

- `lambda` (L2 Penalty)
- `cp` (Complexity Parameter)

Required packages: `stepPlr`

Penalized Multinomial Regression

```
method = 'multinom'
```

Type: Classification

Tuning parameters:

- `decay` (Weight Decay)

Required packages: `nnet`

A model-specific variable importance metric is available.

Penalized Ordinal Regression

```
method = 'ordinalNet'
```

Type: Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `criteria` (Selection Criterion)
- `link` (Link Function)

Required packages: `ordinalNet` , `plyr`

A model-specific variable importance metric is available. Notes: Requires ordinalNet package version >= 2.0

Regularized Discriminant Analysis

```
method = 'rda'
```

Type: Classification

Tuning parameters:

- `gamma` (Gamma)
- `lambda` (Lambda)

Required packages: `klaR`

Regularized Linear Discriminant Analysis

```
method = 'rlda'
```

Type: Classification

Tuning parameters:

- `estimator` (Regularization Method)

Required packages: `sparsediscrim`

Regularized Logistic Regression

```
method = 'regLogistic'
```

Type: Classification

Tuning parameters:

- `cost` (Cost)
- `loss` (Loss Function)
- `epsilon` (Tolerance)

Required packages: `LiblineaR`

Robust Linear Discriminant Analysis

```
method = 'Linda'
```

Type: Classification

No tuning parameters for this model

Required packages: `rrcov`

Robust Regularized Linear Discriminant Analysis

```
method = 'rrlda'
```

Type: Classification

Tuning parameters:

- `lambda` (Penalty Parameter)

- `hp` (Robustness Parameter)
- `penalty` (Penalty Type)

Required packages: `rrlda`

Notes: Unlike other packages used by `train`, the `rrlda` package is fully loaded when this model is used.

Robust SIMCA

```
method = 'RSimca'
```

Type: Classification

No tuning parameters for this model

Required packages: `rrcovHD`

Notes: Unlike other packages used by `train`, the `rrcovHD` package is fully loaded when this model is used.

Shrinkage Discriminant Analysis

```
method = 'sda'
```

Type: Classification

Tuning parameters:

- `diagonal` (Diagonalize)
- `lambda` (shrinkage)

Required packages: `sda`

Sparse Distance Weighted Discrimination

```
method = 'sdwd'
```

Type: Classification

Tuning parameters:

- `lambda` (L1 Penalty)
- `lambda2` (L2 Penalty)

Required packages: `sdwd`

A model-specific variable importance metric is available.

Sparse Linear Discriminant Analysis

```
method = 'sparseLDA'
```

Type: Classification

Tuning parameters:

- `NumVars` (# Predictors)
- `lambda` (Lambda)

Required packages: `sparseLDA`

Sparse Partial Least Squares

```
method = 'spls'
```

Type: Regression, Classification

Tuning parameters:

- `K` (#Components)
- `eta` (Threshold)
- `kappa` (Kappa)

Required packages: `spls`

Stabilized Linear Discriminant Analysis

```
method = 'slda'
```

Type: Classification

No tuning parameters for this model

Required packages: `ipred`

Support Vector Machines with Linear Kernel

```
method = 'svmLinear'
```

Type: Regression, Classification

Tuning parameters:

- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Linear Kernel

```
method = 'svmLinear2'
```

Type: Regression, Classification

Tuning parameters:

- `cost` (Cost)

Required packages: `e1071`

7.0.22 Linear Regression

(back to contents)

Bagged Logic Regression

```
method = 'logicBag'
```

Type: Regression, Classification

Tuning parameters:

- `nleaves` (Maximum Number of Leaves)
- `ntrees` (Number of Trees)

Required packages: `logicFS`

Notes: Unlike other packages used by `train`, the `logicFS` package is fully loaded when this model is used.

Bayesian Ridge Regression

```
method = 'bridge'
```

Type: Regression

No tuning parameters for this model

Required packages: `monomvn`

Bayesian Ridge Regression (Model Averaged)

```
method = 'blassoAveraged'
```

Type: Regression

No tuning parameters for this model

Required packages: `monomvn`

Notes: This model makes predictions by averaging the predictions based on the posterior estimates of the regression coefficients. While it is possible that some of these posterior estimates are zero for non-informative predictors, the final predicted value may be a function of many (or even all) predictors.

Boosted Linear Model

```
method = 'BstLm'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `nu` (Shrinkage)

Required packages: `bst` , `plyr`

Cubist

```
method = 'cubist'
```

Type: Regression

Tuning parameters:

- `committees` (#Committees)
- `neighbors` (#Instances)

Required packages: `Cubist`

A model-specific variable importance metric is available.

Elasticnet

```
method = 'enet'
```

Type: Regression

Tuning parameters:

- `fraction` (Fraction of Full Solution)
- `lambda` (Weight Decay)

Required packages: `elasticnet`

glmnet

```
method = 'glmnet_h2o'
```

Type: Regression, Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `lambda` (Regularization Parameter)

Required packages: `h2o`

A model-specific variable importance metric is available.

glmnet

```
method = 'glmnet'
```

Type: Regression, Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `lambda` (Regularization Parameter)

Required packages: `glmnet` , `Matrix`

A model-specific variable importance metric is available.

Independent Component Regression

```
method = 'icr'
```

Type: Regression

Tuning parameters:

- `n.comp` (#Components)

Required packages: `fastICA`

L2 Regularized Support Vector Machine (dual) with Linear Kernel

```
method = 'svmLinear3'
```

Type: Regression, Classification

Tuning parameters:

- `cost` (Cost)
- `Loss` (Loss Function)

Required packages: `LiblineaR`

Least Angle Regression

```
method = 'lars'
```

Type: Regression

Tuning parameters:

- `fraction` (Fraction)

Required packages: `lars`

Least Angle Regression

```
method = 'lars2'
```

Type: Regression

Tuning parameters:

- `step` (#Steps)

Required packages: `lars`

Linear Regression

```
method = 'lm'
```

Type: Regression

Tuning parameters:

- `intercept` (intercept)

A model-specific variable importance metric is available.

Linear Regression with Backwards Selection

```
method = 'leapBackward'
```

Type: Regression

Tuning parameters:

- `nvmax` (Maximum Number of Predictors)

Required packages: `leaps`

Linear Regression with Forward Selection

```
method = 'leapForward'
```

Type: Regression

Tuning parameters:

- `nvmax` (Maximum Number of Predictors)

Required packages: `leaps`

Linear Regression with Stepwise Selection

```
method = 'leapSeq'
```

Type: Regression

Tuning parameters:

- `nvmax` (Maximum Number of Predictors)

Required packages: `leaps`

Linear Regression with Stepwise Selection

```
method = 'lmStepAIC'
```

Type: Regression

No tuning parameters for this model

Required packages: `MASS`

Logic Regression

```
method = 'logreg'
```

Type: Regression, Classification

Tuning parameters:

- `treesize` (Maximum Number of Leaves)
- `ntrees` (Number of Trees)

Required packages: `LogicReg`

Model Rules

```
method = 'M5Rules'
```

Type: Regression

Tuning parameters:

- `pruned` (Pruned)
- `smoothed` (Smoothed)

Required packages: `RWeka`

Model Tree

```
method = 'M5'
```

Type: Regression

Tuning parameters:

- `pruned` (Pruned)
- `smoothed` (Smoothed)
- `rules` (Rules)

Required packages: `RWeka`

Multi-Step Adaptive MCP-Net

```
method = 'msaenet'
```

Type: Regression, Classification

Tuning parameters:

- `alphas` (Alpha)
- `nsteps` (#Adaptive Estimation Steps)
- `scale` (Adaptive Weight Scaling Factor)

Required packages: `msaenet`

A model-specific variable importance metric is available.

Non-Convex Penalized Quantile Regression

```
method = 'rqnc'
```

Type: Regression

Tuning parameters:

- `lambda` (L1 Penalty)
- `penalty` (Penalty Type)

Required packages: `rqPen`

Non-Negative Least Squares

```
method = 'nnls'
```

Type: Regression

No tuning parameters for this model

Required packages: `nnls`

A model-specific variable importance metric is available.

Partial Least Squares

```
method = 'kernelpls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Partial Least Squares

```
method = 'pls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Partial Least Squares

```
method = 'simpls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Partial Least Squares

```
method = 'widekernelpls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Penalized Linear Regression

```
method = 'penalized'
```

Type: Regression

Tuning parameters:

- `lambda1` (L1 Penalty)
- `lambda2` (L2 Penalty)

Required packages: `penalized`

Penalized Ordinal Regression

```
method = 'ordinalNet'
```

Type: Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `criteria` (Selection Criterion)
- `link` (Link Function)

Required packages: `ordinalNet` , `plyr`

A model-specific variable importance metric is available. Notes: Requires ordinalNet package version >= 2.0

Principal Component Analysis

```
method = 'pcr'
```

Type: Regression

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

Quantile Regression with LASSO penalty

```
method = 'rqlasso'
```

Type: Regression

Tuning parameters:

- `lambda` (L1 Penalty)

Required packages: `rqPen`

Relaxed Lasso

```
method = 'relaxo'
```

Type: Regression

Tuning parameters:

- `lambda` (Penalty Parameter)
- `phi` (Relaxation Parameter)

Required packages: `relaxo` , `plyr`

Relevance Vector Machines with Linear Kernel

```
method = 'rvmLinear'
```

Type: Regression

No tuning parameters for this model

Required packages: `kernlab`

Ridge Regression

```
method = 'ridge'
```

Type: Regression

Tuning parameters:

- `lambda` (Weight Decay)

Required packages: `elasticnet`

Ridge Regression with Variable Selection

```
method = 'foba'
```

Type: Regression

Tuning parameters:

- `k` (#Variables Retained)
- `lambda` (L2 Penalty)

Required packages: `foba`

Robust Linear Model

```
method = 'rlm'
```

Type: Regression

Tuning parameters:

- `intercept` (intercept)
- `psi` (psi)

Required packages: `MASS`

A model-specific variable importance metric is available.

Sparse Partial Least Squares

```
method = 'spls'
```

Type: Regression, Classification

Tuning parameters:

- `K` (#Components)
- `eta` (Threshold)
- `kappa` (Kappa)

Required packages: `spls`

Spike and Slab Regression

```
method = 'spikeslab'
```

Type: Regression

Tuning parameters:

- `vars` (Variables Retained)

Required packages: `spikeslab` , `plyr`

Notes: Unlike other packages used by `train`, the `spikeslab` package is fully loaded when this model is used.

Supervised Principal Component Analysis

```
method = 'superpc'
```

Type: Regression

Tuning parameters:

- `threshold` (Threshold)
- `n.components` (#Components)

Required packages: `superpc`

Support Vector Machines with Linear Kernel

```
method = 'svmLinear'
```

Type: Regression, Classification

Tuning parameters:

- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Linear Kernel

```
method = 'svmLinear2'
```

Type: Regression, Classification

Tuning parameters:

- `cost` (Cost)

Required packages: `e1071`

The Bayesian lasso

```
method = 'blasso'
```

Type: Regression

Tuning parameters:

- `sparsity` (Sparsity Threshold)

Required packages: `monomvn`

Notes: This model creates predictions using the mean of the posterior distributions but sets some parameters specifically to zero based on the tuning parameter `sparsity`. For example, when `sparsity = .5`, only coefficients where at least half the posterior estimates are nonzero are used.

The lasso

```
method = 'lasso'
```

Type: Regression

Tuning parameters:

- `fraction` (Fraction of Full Solution)

Required packages: `elasticnet`

7.0.23 Logic Regression

(back to contents)

Bagged Logic Regression

```
method = 'logicBag'
```

Type: Regression, Classification

Tuning parameters:

- `nleaves` (Maximum Number of Leaves)
- `ntrees` (Number of Trees)

Required packages: `logicFS`

Notes: Unlike other packages used by `train`, the `logicFS` package is fully loaded when this model is used.

Logic Regression

```
method = 'logreg'
```

Type: Regression, Classification

Tuning parameters:

- `treesize` (Maximum Number of Leaves)
- `ntrees` (Number of Trees)

Required packages: `LogicReg`

7.0.24 Logistic Regression

(back to contents)

Adjacent Categories Probability Model for Ordinal Data

```
method = 'vglmAdjCat'
```

Type: Classification

Tuning parameters:

- `parallel` (Parallel Curves)
- `link` (Link Function)

Required packages: `VGAM`

Bagged Logic Regression

```
method = 'logicBag'
```

Type: Regression, Classification

Tuning parameters:

- `nleaves` (Maximum Number of Leaves)
- `ntrees` (Number of Trees)

Required packages: `logicFS`

Notes: Unlike other packages used by `train`, the `logicFS` package is fully loaded when this model is used.

Bayesian Generalized Linear Model

```
method = 'bayesglm'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `arm`

Boosted Logistic Regression

```
method = 'LogitBoost'
```

Type: Classification

Tuning parameters:

- `nIter` (# Boosting Iterations)

Required packages: `caTools`

Continuation Ratio Model for Ordinal Data

```
method = 'vglmContRatio'
```

Type: Classification

Tuning parameters:

- `parallel` (Parallel Curves)
- `link` (Link Function)

Required packages: `VGAM`

Cumulative Probability Model for Ordinal Data

```
method = 'vglmCumulative'
```

Type: Classification

Tuning parameters:

- `parallel` (Parallel Curves)
- `link` (Link Function)

Required packages: `VGAM`

Generalized Partial Least Squares

```
method = 'gpls'
```

Type: Classification

Tuning parameters:

- `K.prov` (#Components)

Required packages: `gpls`

Logic Regression

```
method = 'logreg'
```

Type: Regression, Classification

Tuning parameters:

- `treesize` (Maximum Number of Leaves)
- `ntrees` (Number of Trees)

Required packages: `LogicReg`

Logistic Model Trees

```
method = 'LMT'
```

Type: Classification

Tuning parameters:

- `iter` (# Iteratons)

Required packages: `RWeka`

Oblique Random Forest

```
method = 'ORFlog'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Ordered Logistic or Probit Regression

```
method = 'polr'
```

Type: Classification

Tuning parameters:

- `method` (parameter)

Required packages: `MASS`

A model-specific variable importance metric is available.

Penalized Logistic Regression

```
method = 'plr'
```

Type: Classification

Tuning parameters:

- `lambda` (L2 Penalty)
- `cp` (Complexity Parameter)

Required packages: `stepPlr`

Penalized Multinomial Regression

```
method = 'multinom'
```

Type: Classification

Tuning parameters:

- `decay` (Weight Decay)

Required packages: `nnet`

A model-specific variable importance metric is available.

7.0.25 Mixture Model

(back to contents)

Adaptive Mixture Discriminant Analysis

```
method = 'amdai'
```

Type: Classification

Tuning parameters:

- `model` (Model Type)

Required packages: `adaptDA`

Mixture Discriminant Analysis

```
method = 'mda'
```

Type: Classification

Tuning parameters:

- `subclasses` (#Subclasses Per Class)

Required packages: `mda`

Robust Mixture Discriminant Analysis

```
method = 'rmda'
```

Type: Classification

Tuning parameters:

- `K` (#Subclasses Per Class)
- `model` (Model)

Required packages: `robustDA`

Sparse Mixture Discriminant Analysis

```
method = 'smda'
```

Type: Classification

Tuning parameters:

- `NumVars` (# Predictors)
- `lambda` (Lambda)
- `R` (# Subclasses)

Required packages: `sparseLDA`

7.0.26 Model Tree

(back to contents)

Cubist

```
method = 'cubist'
```

Type: Regression

Tuning parameters:

- `committees` (#Committees)
- `neighbors` (#Instances)

Required packages: `Cubist`

A model-specific variable importance metric is available.

Logistic Model Trees

```
method = 'LMT'
```

Type: Classification

Tuning parameters:

- `iter` (# Iteratons)

Required packages: `RWeka`

Model Rules

```
method = 'M5Rules'
```

Type: Regression

Tuning parameters:

- `pruned` (Pruned)
- `smoothed` (Smoothed)

Required packages: `RWeka`

Model Tree

```
method = 'M5'
```

Type: Regression

Tuning parameters:

- `pruned` (Pruned)
- `smoothed` (Smoothed)
- `rules` (Rules)

Required packages: `RWeka`

7.0.27 Multivariate Adaptive Regression Splines

(back to contents)

Bagged Flexible Discriminant Analysis

```
method = 'bagFDA'
```

Type: Classification

Tuning parameters:

- `degree` (Product Degree)
- `nprune` (#Terms)

Required packages: `earth` , `mda`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train` , the `earth` package is fully loaded when this model is used.

Bagged MARS

```
method = 'bagEarth'
```

Type: Regression, Classification

Tuning parameters:

- `nprune` (#Terms)
- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Bagged MARS using gCV Pruning

```
method = 'bagEarthGCV'
```

Type: Regression, Classification

Tuning parameters:

- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Flexible Discriminant Analysis

```
method = 'fda'
```

Type: Classification

Tuning parameters:

- `degree` (Product Degree)
- `nprune` (#Terms)

Required packages: `earth` , `mda`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train` , the `earth` package is fully loaded when this model is used.

Multivariate Adaptive Regression Spline

```
method = 'earth'
```

Type: Regression, Classification

Tuning parameters:

- `nprune` (#Terms)
- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train` , the `earth` package is fully loaded when this model is used.

Multivariate Adaptive Regression Splines

```
method = 'gcvEarth'
```

Type: Regression, Classification

Tuning parameters:

- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

7.0.28 Neural Network

(back to contents)

Bayesian Regularized Neural Networks

```
method = 'brnn'
```

Type: Regression

Tuning parameters:

- `neurons` (# Neurons)

Required packages: `brnn`

Extreme Learning Machine

```
method = 'elm'
```

Type: Classification, Regression

Tuning parameters:

- `nhid` (#Hidden Units)
- `actfun` (Activation Function)

Required packages: `elmNN`

Model Averaged Neural Network

```
method = 'avNNet'
```

Type: Classification, Regression

Tuning parameters:

- `size` (#Hidden Units)
- `decay` (Weight Decay)
- `bag` (Bagging)

Required packages: `nnet`

Monotone Multi-Layer Perceptron Neural Network

```
method = 'monmlp'
```

Type: Classification, Regression

Tuning parameters:

- `hidden1` (#Hidden Units)
- `n.ensemble` (#Models)

Required packages: `monmlp`

Multi-Layer Perceptron

```
method = 'mlp'
```

Type: Regression, Classification

Tuning parameters:

- `size` (#Hidden Units)

Required packages: `RSNNS`

Multi-Layer Perceptron

```
method = 'mlpWeightDecay'
```

Type: Regression, Classification

Tuning parameters:

- `size` (#Hidden Units)
- `decay` (Weight Decay)

Required packages: `RSNNS`

Multi-Layer Perceptron, multiple layers

```
method = 'mlpWeightDecayML'
```

Type: Regression, Classification

Tuning parameters:

- `layer1` (#Hidden Units layer1)
- `layer2` (#Hidden Units layer2)
- `layer3` (#Hidden Units layer3)
- `decay` (Weight Decay)

Required packages: `RSNNS`

Multi-Layer Perceptron, with multiple layers

```
method = 'mlpML'
```

Type: Regression, Classification

Tuning parameters:

- `layer1` (#Hidden Units layer1)
- `layer2` (#Hidden Units layer2)
- `layer3` (#Hidden Units layer3)

Required packages: `RSNNS`

Multilayer Perceptron Network by Stochastic Gradient Descent

```
method = 'mlpSGD'
```

Type: Regression, Classification

Tuning parameters:

- `size` (#Hidden Units)
- `l2reg` (L2 Regularization)
- `lambda` (RMSE Gradient Scaling)
- `learn_rate` (Learning Rate)
- `momentum` (Momentum)
- `gamma` (Learning Rate Decay)
- `minibatchsz` (Batch Size)
- `repeats` (#Models)

Required packages: `FCNN4R` , `plyr`

A model-specific variable importance metric is available.

Multilayer Perceptron Network with Dropout

```
method = 'mlpKerasDropout'
```

Type: Regression, Classification

Tuning parameters:

- `size` (#Hidden Units)
- `dropout` (Dropout Rate)
- `batch_size` (Batch Size)

- `lr` (Learning Rate)
- `rho` (Rho)
- `decay` (Learning Rate Decay)
- `activation` (Activation Function)

Required packages: `keras`

Notes: After `train` completes, the keras model object is serialized so that it can be used between R session. When predicting, the code will temporarily unsearalize the object. To make the predictions more efficient, the user might want to use `keras::unsearlize_model(object$finalModel$object)` in the current R session so that that operation is only done once. Also, this model cannot be run in parallel due to the nature of how tensorflow does the computations. Unlike other packages used by `train`, the `dplyr` package is fully loaded when this model is used.

Multilayer Perceptron Network with Dropout

```
method = 'mlpKerasDropoutCost'
```

Type: Classification

Tuning parameters:

- `size` (#Hidden Units)
- `dropout` (Dropout Rate)
- `batch_size` (Batch Size)
- `lr` (Learning Rate)
- `rho` (Rho)

- `decay` (Learning Rate Decay)
- `cost` (Cost)
- `activation` (Activation Function)

Required packages: `keras`

Notes: After `train` completes, the keras model object is serialized so that it can be used between R session. When predicting, the code will temporarily unsearalize the object. To make the predictions more efficient, the user might want to use `keras::unsearlize_model(object$finalModel$object)` in the current R session so that that operation is only done once. Also, this model cannot be run in parallel due to the nature of how tensorflow does the computations. Finally, the cost parameter weights the first class in the outcome vector. Unlike other packages used by `train`, the `dplyr` package is fully loaded when this model is used.

Multilayer Perceptron Network with Weight Decay

```
method = 'mlpKerasDecay'
```

Type: Regression, Classification

Tuning parameters:

- `size` (#Hidden Units)
- `lambda` (L2 Regularization)
- `batch_size` (Batch Size)
- `lr` (Learning Rate)
- `rho` (Rho)

- `decay` (Learning Rate Decay)
- `activation` (Activation Function)

Required packages: `keras`

Notes: After `train` completes, the keras model object is serialized so that it can be used between R session. When predicting, the code will temporarily unsearalize the object. To make the predictions more efficient, the user might want to use `keras::unsearlize_model(object$finalModel$object)` in the current R session so that that operation is only done once. Also, this model cannot be run in parallel due to the nature of how tensorflow does the computations. Unlike other packages used by `train`, the `dplyr` package is fully loaded when this model is used.

Multilayer Perceptron Network with Weight Decay

```
method = 'mlpKerasDecayCost'
```

Type: Classification

Tuning parameters:

- `size` (#Hidden Units)
- `lambda` (L2 Regularization)
- `batch_size` (Batch Size)
- `lr` (Learning Rate)
- `rho` (Rho)
- `decay` (Learning Rate Decay)
- `cost` (Cost)

- `activation` (Activation Function)

Required packages: `keras`

Notes: After `train` completes, the keras model object is serialized so that it can be used between R session. When predicting, the code will temporarily unsearalize the object. To make the predictions more efficient, the user might want to use `keras::unsearlize_model(object$finalModel$object)` in the current R session so that that operation is only done once. Also, this model cannot be run in parallel due to the nature of how tensorflow does the computations. Finally, the cost parameter weights the first class in the outcome vector. Unlike other packages used by `train`, the `dplyr` package is fully loaded when this model is used.

Neural Network

```
method = 'mxnet'
```

Type: Classification, Regression

Tuning parameters:

- `layer1` (#Hidden Units in Layer 1)
- `layer2` (#Hidden Units in Layer 2)
- `layer3` (#Hidden Units in Layer 3)
- `learning.rate` (Learning Rate)
- `momentum` (Momentum)
- `dropout` (Dropout Rate)
- `activation` (Activation Function)

Required packages: `mxnet`

Notes: The `mxnet` package is not yet on CRAN. See http://mxnet.io for installation instructions.

Neural Network

```
method = 'mxnetAdam'
```

Type: Classification, Regression

Tuning parameters:

- `layer1` (#Hidden Units in Layer 1)
- `layer2` (#Hidden Units in Layer 2)
- `layer3` (#Hidden Units in Layer 3)
- `dropout` (Dropout Rate)
- `beta1` (beta1)
- `beta2` (beta2)
- `learningrate` (Learning Rate)
- `activation` (Activation Function)

Required packages: `mxnet`

Notes: The `mxnet` package is not yet on CRAN. See http://mxnet.io for installation instructions. Users are strongly advised to define `num.round` themselves.

Neural Network

```
method = 'neuralnet'
```

Type: Regression

Tuning parameters:

- `layer1` (#Hidden Units in Layer 1)
- `layer2` (#Hidden Units in Layer 2)
- `layer3` (#Hidden Units in Layer 3)

Required packages: `neuralnet`

Neural Network

```
method = 'nnet'
```

Type: Classification, Regression

Tuning parameters:

- `size` (#Hidden Units)
- `decay` (Weight Decay)

Required packages: `nnet`

A model-specific variable importance metric is available.

Neural Networks with Feature Extraction

```
method = 'pcaNNet'
```

Type: Classification, Regression

Tuning parameters:

- `size` (#Hidden Units)
- `decay` (Weight Decay)

Required packages: `nnet`

Penalized Multinomial Regression

```
method = 'multinom'
```

Type: Classification

Tuning parameters:

- `decay` (Weight Decay)

Required packages: `nnet`

A model-specific variable importance metric is available.

Quantile Regression Neural Network

```
method = 'qrnn'
```

Type: Regression

Tuning parameters:

- `n.hidden` (#Hidden Units)
- `penalty` (Weight Decay)

- `bag` (Bagged Models?)

Required packages: `qrnn`

Radial Basis Function Network

```
method = 'rbf'
```

Type: Classification, Regression

Tuning parameters:

- `size` (#Hidden Units)

Required packages: `RSNNS`

Radial Basis Function Network

```
method = 'rbfDDA'
```

Type: Regression, Classification

Tuning parameters:

- `negativeThreshold` (Activation Limit for Conflicting Classes)

Required packages: `RSNNS`

Stacked AutoEncoder Deep Neural Network

```
method = 'dnn'
```

Type: Classification, Regression

Tuning parameters:

- `layer1` (Hidden Layer 1)
- `layer2` (Hidden Layer 2)
- `layer3` (Hidden Layer 3)
- `hidden_dropout` (Hidden Dropouts)
- `visible_dropout` (Visible Dropout)

Required packages: `deepnet`

7.0.29 Oblique Tree

(back to contents)

Oblique Random Forest

```
method = 'ORFlog'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Oblique Random Forest

```
method = 'ORFpls'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Oblique Random Forest

```
method = 'ORFridge'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Oblique Random Forest

```
method = 'ORFsvm'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

7.0.30 Ordinal Outcomes

(back to contents)

Adjacent Categories Probability Model for Ordinal Data

```
method = 'vglmAdjCat'
```

Type: Classification

Tuning parameters:

- `parallel` (Parallel Curves)
- `link` (Link Function)

Required packages: `VGAM`

CART or Ordinal Responses

```
method = 'rpartScore'
```

Type: Classification

Tuning parameters:

- `cp` (Complexity Parameter)
- `split` (Split Function)
- `prune` (Pruning Measure)

Required packages: `rpartScore` , `plyr`

A model-specific variable importance metric is available.

Continuation Ratio Model for Ordinal Data

```
method = 'vglmContRatio'
```

Type: Classification

Tuning parameters:

- `parallel` (Parallel Curves)
- `link` (Link Function)

Required packages: `VGAM`

Cumulative Probability Model for Ordinal Data

```
method = 'vglmCumulative'
```

Type: Classification

Tuning parameters:

- `parallel` (Parallel Curves)
- `link` (Link Function)

Required packages: `VGAM`

Ordered Logistic or Probit Regression

```
method = 'polr'
```

Type: Classification

Tuning parameters:

- `method` (parameter)

Required packages: `MASS`

A model-specific variable importance metric is available.

Penalized Ordinal Regression

```
method = 'ordinalNet'
```

Type: Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `criteria` (Selection Criterion)
- `link` (Link Function)

Required packages: `ordinalNet` , `plyr`

A model-specific variable importance metric is available. Notes:

Requires ordinalNet package version >= 2.0

7.0.31 Partial Least Squares

(back to contents)

Generalized Partial Least Squares

```
method = 'gpls'
```

Type: Classification

Tuning parameters:

- `K.prov` (#Components)

Required packages: `gpls`

Oblique Random Forest

```
method = 'ORFpls'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Partial Least Squares

```
method = 'kernelpls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Partial Least Squares

```
method = 'pls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Partial Least Squares

```
method = 'simpls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Partial Least Squares

```
method = 'widekernelpls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Partial Least Squares Generalized Linear Models

```
method = 'plsRglm'
```

Type: Classification, Regression

Tuning parameters:

- `nt` (#PLS Components)
- `alpha.pvals.expli` (p-Value threshold)

Required packages: `plsRglm`

Notes: Unlike other packages used by `train`, the `plsRglm` package is fully loaded when this model is used.

Sparse Partial Least Squares

```
method = 'spls'
```

Type: Regression, Classification

Tuning parameters:

- `K` (#Components)
- `eta` (Threshold)
- `kappa` (Kappa)

Required packages: `spls`

7.0.32 Patient Rule Induction Method

(back to contents)

Patient Rule Induction Method

```
method = 'PRIM'
```

Type: Classification

Tuning parameters:

- `peel.alpha` (peeling quantile)
- `paste.alpha` (pasting quantile)
- `mass.min` (minimum mass)

Required packages: `supervisedPRIM`

7.0.33 Polynomial Model

(back to contents)

Diagonal Discriminant Analysis

```
method = 'dda'
```

Type: Classification

Tuning parameters:

- `model` (Model)
- `shrinkage` (Shrinkage Type)

Required packages: `sparsediscrim`

Distance Weighted Discrimination with Polynomial Kernel

```
method = 'dwdPoly'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)
- `degree` (Polynomial Degree)
- `scale` (Scale)

Required packages: `kerndwd`

Gaussian Process with Polynomial Kernel

```
method = 'gausssprPoly'
```

Type: Regression, Classification

Tuning parameters:

- `degree` (Polynomial Degree)
- `scale` (Scale)

Required packages: `kernlab`

High-Dimensional Regularized Discriminant Analysis

```
method = 'hdrda'
```

Type: Classification

Tuning parameters:

- `gamma` (Gamma)
- `lambda` (Lambda)
- `shrinkage_type` (Shrinkage Type)

Required packages: `sparsediscrim`

Least Squares Support Vector Machine with Polynomial Kernel

```
method = 'lssvmPoly'
```

Type: Classification

Tuning parameters:

- `degree` (Polynomial Degree)
- `scale` (Scale)
- `tau` (Regularization Parameter)

Required packages: `kernlab`

Penalized Discriminant Analysis

```
method = 'pda'
```

Type: Classification

Tuning parameters:

- `lambda` (Shrinkage Penalty Coefficient)

Required packages: `mda`

Penalized Discriminant Analysis

```
method = 'pda2'
```

Type: Classification

Tuning parameters:

- `df` (Degrees of Freedom)

Required packages: `mda`

Polynomial Kernel Regularized Least Squares

```
method = 'krlsPoly'
```

Type: Regression

Tuning parameters:

- `lambda` (Regularization Parameter)
- `degree` (Polynomial Degree)

Required packages: `KRLS`

Quadratic Discriminant Analysis

```
method = 'qda'
```

Type: Classification

No tuning parameters for this model

Required packages: `MASS`

Quadratic Discriminant Analysis with Stepwise Feature Selection

```
method = 'stepQDA'
```

Type: Classification

Tuning parameters:

- `maxvar` (Maximum #Variables)
- `direction` (Search Direction)

Required packages: `klaR` , `MASS`

Regularized Discriminant Analysis

```
method = 'rda'
```

Type: Classification

Tuning parameters:

- `gamma` (Gamma)
- `lambda` (Lambda)

Required packages: `klaR`

Regularized Linear Discriminant Analysis

```
method = 'rlda'
```

Type: Classification

Tuning parameters:

- `estimator` (Regularization Method)

Required packages: `sparsediscrim`

Relevance Vector Machines with Polynomial Kernel

```
method = 'rvmPoly'
```

Type: Regression

Tuning parameters:

- `scale` (Scale)
- `degree` (Polynomial Degree)

Required packages: `kernlab`

Robust Quadratic Discriminant Analysis

```
method = 'QdaCov'
```

Type: Classification

No tuning parameters for this model

Required packages: `rrcov`

Support Vector Machines with Polynomial Kernel

```
method = 'svmPoly'
```

Type: Regression, Classification

Tuning parameters:

- `degree` (Polynomial Degree)
- `scale` (Scale)
- `C` (Cost)

Required packages: `kernlab`

7.0.34 Prototype Models

(back to contents)

Cubist

```
method = 'cubist'
```

Type: Regression

Tuning parameters:

- `committees` (#Committees)
- `neighbors` (#Instances)

Required packages: `Cubist`

A model-specific variable importance metric is available.

Greedy Prototype Selection

```
method = 'protoclass'
```

Type: Classification

Tuning parameters:

- `eps` (Ball Size)
- `Minkowski` (Distance Order)

Required packages: `proxy` , `protoclass`

k-Nearest Neighbors

```
method = 'kknn'
```

Type: Regression, Classification

Tuning parameters:

- `kmax` (Max. #Neighbors)
- `distance` (Distance)
- `kernel` (Kernel)

Required packages: `kknn`

k-Nearest Neighbors

```
method = 'knn'
```

Type: Classification, Regression

Tuning parameters:

- `k` (#Neighbors)

Learning Vector Quantization

```
method = 'lvq'
```

Type: Classification

Tuning parameters:

- `size` (Codebook Size)
- `k` (#Prototypes)

Required packages: `class`

Nearest Shrunken Centroids

```
method = 'pam'
```

Type: Classification

Tuning parameters:

- `threshold` (Shrinkage Threshold)

Required packages: `pamr`

A model-specific variable importance metric is available.

Optimal Weighted Nearest Neighbor Classifier

```
method = 'ownn'
```

Type: Classification

Tuning parameters:

- `K` (#Neighbors)

Required packages: `snn`

Stabilized Nearest Neighbor Classifier

```
method = 'snn'
```

Type: Classification

Tuning parameters:

- `lambda` (Stabilization Parameter)

Required packages: `snn`

7.0.35 Quantile Regression

(back to contents)

Non-Convex Penalized Quantile Regression

```
method = 'rqnc'
```

Type: Regression

Tuning parameters:

- `lambda` (L1 Penalty)
- `penalty` (Penalty Type)

Required packages: `rqPen`

Quantile Random Forest

```
method = 'qrf'
```

Type: Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `quantregForest`

Quantile Regression Neural Network

```
method = 'qrnn'
```

Type: Regression

Tuning parameters:

- `n.hidden` (#Hidden Units)
- `penalty` (Weight Decay)
- `bag` (Bagged Models?)

Required packages: `qrnn`

Quantile Regression with LASSO penalty

```
method = 'rqlasso'
```

Type: Regression

Tuning parameters:

- `lambda` (L1 Penalty)

Required packages: `rqPen`

7.0.36 Radial Basis Function

(back to contents)

Distance Weighted Discrimination with Radial Basis Function Kernel

```
method = 'dwdRadial'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)
- `sigma` (Sigma)

Required packages: `kernlab` , `kerndwd`

Gaussian Process with Radial Basis Function Kernel

```
method = 'gausssprRadial'
```

Type: Regression, Classification

Tuning parameters:

- `sigma` (Sigma)

Required packages: `kernlab`

Least Squares Support Vector Machine with Radial Basis Function Kernel

```
method = 'lssvmRadial'
```

Type: Classification

Tuning parameters:

- `sigma` (Sigma)
- `tau` (Regularization Parameter)

Required packages: `kernlab`

Radial Basis Function Kernel Regularized Least Squares

```
method = 'krlsRadial'
```

Type: Regression

Tuning parameters:

- `lambda` (Regularization Parameter)
- `sigma` (Sigma)

Required packages: `KRLS` , `kernlab`

Radial Basis Function Network

```
method = 'rbf'
```

Type: Classification, Regression

Tuning parameters:

- `size` (#Hidden Units)

Required packages: `RSNNS`

Radial Basis Function Network

```
method = 'rbfDDA'
```

Type: Regression, Classification

Tuning parameters:

- `negativeThreshold` (Activation Limit for Conflicting Classes)

Required packages: `RSNNS`

Relevance Vector Machines with Radial Basis Function Kernel

```
method = 'rvmRadial'
```

Type: Regression

Tuning parameters:

- `sigma` (Sigma)

Required packages: `kernlab`

Support Vector Machines with Class Weights

```
method = 'svmRadialWeights'
```

Type: Classification

Tuning parameters:

- `sigma` (Sigma)
- `C` (Cost)
- `Weight` (Weight)

Required packages: `kernlab`

Support Vector Machines with Radial Basis Function Kernel

```
method = 'svmRadial'
```

Type: Regression, Classification

Tuning parameters:

- `sigma` (Sigma)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Radial Basis Function Kernel

```
method = 'svmRadialCost'
```

Type: Regression, Classification

Tuning parameters:

- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Radial Basis Function Kernel

```
method = 'svmRadialSigma'
```

Type: Regression, Classification

Tuning parameters:

- `sigma` (Sigma)
- `C` (Cost)

Required packages: `kernlab`

Notes: This SVM model tunes over the cost parameter and the RBF kernel parameter sigma. In the latter case, using `tuneLength` will, at most, evaluate six values of the kernel parameter. This enables a broad search over the cost parameter and a relatively narrow search over `sigma`

Variational Bayesian Multinomial Probit Regression

```
method = 'vbmpRadial'
```

Type: Classification

Tuning parameters:

- `estimateTheta` (Theta Estimated)

Required packages: `vbmp`

7.0.37 Random Forest

(back to contents)

Conditional Inference Random Forest

```
method = 'cforest'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `party`

A model-specific variable importance metric is available.

Oblique Random Forest

```
method = 'ORFlog'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train` , the `obliqueRF` package is fully loaded when this model is used.

Oblique Random Forest

```
method = 'ORFpls'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Oblique Random Forest

```
method = 'ORFridge'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Oblique Random Forest

```
method = 'ORFsvm'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Parallel Random Forest

```
method = 'parRF'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `e1071`, `randomForest`, `foreach`, `import`

A model-specific variable importance metric is available.

Quantile Random Forest

```
method = 'qrf'
```

Type: Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `quantregForest`

Random Ferns

```
method = 'rFerns'
```

Type: Classification

Tuning parameters:

- `depth` (Fern Depth)

Required packages: `rFerns`

Random Forest

```
method = 'ranger'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `splitrule` (Splitting Rule)
- `min.node.size` (Minimal Node Size)

Required packages: `e1071` , `ranger` , `dplyr`

A model-specific variable importance metric is available.

Random Forest

```
method = 'Rborist'
```

Type: Classification, Regression

Tuning parameters:

- `predFixed` (#Randomly Selected Predictors)
- `minNode` (Minimal Node Size)

Required packages: `Rborist`

A model-specific variable importance metric is available.

Random Forest

```
method = 'rf'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `randomForest`

A model-specific variable importance metric is available.

Random Forest by Randomization

```
method = 'extraTrees'
```

Type: Regression, Classification

Tuning parameters:

- `mtry` (# Randomly Selected Predictors)
- `numRandomCuts` (# Random Cuts)

Required packages: `extraTrees`

Random Forest Rule-Based Model

```
method = 'rfRules'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `maxdepth` (Maximum Rule Depth)

Required packages: `randomForest` , `inTrees` , `plyr`

A model-specific variable importance metric is available.

Regularized Random Forest

```
method = 'RRF'
```

Type: Regression, Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `coefReg` (Regularization Value)
- `coefImp` (Importance Coefficient)

Required packages: `randomForest` , `RRF`

A model-specific variable importance metric is available.

Regularized Random Forest

```
method = 'RRFglobal'
```

Type: Regression, Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `coefReg` (Regularization Value)

Required packages: `RRF`

A model-specific variable importance metric is available.

Weighted Subspace Random Forest

```
method = 'wsrf'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `wsrf`

7.0.38 Regularization

(back to contents)

Bayesian Regularized Neural Networks

```
method = 'brnn'
```

Type: Regression

Tuning parameters:

- `neurons` (# Neurons)

Required packages: `brnn`

Diagonal Discriminant Analysis

```
method = 'dda'
```

Type: Classification

Tuning parameters:

- `model` (Model)
- `shrinkage` (Shrinkage Type)

Required packages: `sparsediscrim`

Heteroscedastic Discriminant Analysis

```
method = 'hda'
```

Type: Classification

Tuning parameters:

- `gamma` (Gamma)
- `lambda` (Lambda)
- `newdim` (Dimension of the Discriminative Subspace)

Required packages: `hda`

High-Dimensional Regularized Discriminant Analysis

```
method = 'hdrda'
```

Type: Classification

Tuning parameters:

- `gamma` (Gamma)
- `lambda` (Lambda)
- `shrinkage_type` (Shrinkage Type)

Required packages: `sparsediscrim`

Regularized Discriminant Analysis

```
method = 'rda'
```

Type: Classification

Tuning parameters:

- `gamma` (Gamma)
- `lambda` (Lambda)

Required packages: `klaR`

Regularized Linear Discriminant Analysis

```
method = 'rlda'
```

Type: Classification

Tuning parameters:

- `estimator` (Regularization Method)

Required packages: `sparsediscrim`

Regularized Random Forest

```
method = 'RRF'
```

Type: Regression, Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `coefReg` (Regularization Value)
- `coefImp` (Importance Coefficient)

Required packages: `randomForest` , `RRF`

A model-specific variable importance metric is available.

Regularized Random Forest

```
method = 'RRFglobal'
```

Type: Regression, Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `coefReg` (Regularization Value)

Required packages: `RRF`

A model-specific variable importance metric is available.

Robust Regularized Linear Discriminant Analysis

```
method = 'rrlda'
```

Type: Classification

Tuning parameters:

- `lambda` (Penalty Parameter)
- `hp` (Robustness Parameter)
- `penalty` (Penalty Type)

Required packages: `rrlda`

Notes: Unlike other packages used by `train`, the `rrlda` package is fully loaded when this model is used.

Shrinkage Discriminant Analysis

```
method = 'sda'
```

Type: Classification

Tuning parameters:

- `diagonal` (Diagonalize)
- `lambda` (shrinkage)

Required packages: `sda`

7.0.39 Relevance Vector Machines

(back to contents)

Relevance Vector Machines with Linear Kernel

```
method = 'rvmLinear'
```

Type: Regression

No tuning parameters for this model

Required packages: `kernlab`

Relevance Vector Machines with Polynomial Kernel

```
method = 'rvmPoly'
```

Type: Regression

Tuning parameters:

- `scale` (Scale)
- `degree` (Polynomial Degree)

Required packages: `kernlab`

Relevance Vector Machines with Radial Basis Function Kernel

```
method = 'rvmRadial'
```

Type: Regression

Tuning parameters:

- `sigma` (Sigma)

Required packages: `kernlab`

7.0.40 Ridge Regression

(back to contents)

Oblique Random Forest

```
method = 'ORFridge'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Ridge Regression with Variable Selection

```
method = 'foba'
```

Type: Regression

Tuning parameters:

- `k` (#Variables Retained)
- `lambda` (L2 Penalty)

Required packages: `foba`

7.0.41 Robust Methods

(back to contents)

L2 Regularized Linear Support Vector Machines with Class Weights

```
method = 'svmLinearWeights2'
```

Type: Classification

Tuning parameters:

- `cost` (Cost)
- `Loss` (Loss Function)
- `weight` (Class Weight)

Required packages: `LiblineaR`

L2 Regularized Support Vector Machine (dual) with Linear Kernel

```
method = 'svmLinear3'
```

Type: Regression, Classification

Tuning parameters:

- `cost` (Cost)
- `Loss` (Loss Function)

Required packages: `LiblineaR`

Linear Support Vector Machines with Class Weights

```
method = 'svmLinearWeights'
```

Type: Classification

Tuning parameters:

- `cost` (Cost)

- `weight` (Class Weight)

Required packages: `e1071`

Regularized Logistic Regression

```
method = 'regLogistic'
```

Type: Classification

Tuning parameters:

- `cost` (Cost)
- `loss` (Loss Function)
- `epsilon` (Tolerance)

Required packages: `LiblineaR`

Relevance Vector Machines with Linear Kernel

```
method = 'rvmLinear'
```

Type: Regression

No tuning parameters for this model

Required packages: `kernlab`

Relevance Vector Machines with Polynomial Kernel

```
method = 'rvmPoly'
```

Type: Regression

Tuning parameters:

- `scale` (Scale)
- `degree` (Polynomial Degree)

Required packages: `kernlab`

Relevance Vector Machines with Radial Basis Function Kernel

```
method = 'rvmRadial'
```

Type: Regression

Tuning parameters:

- `sigma` (Sigma)

Required packages: `kernlab`

Robust Mixture Discriminant Analysis

```
method = 'rmda'
```

Type: Classification

Tuning parameters:

- `K` (#Subclasses Per Class)
- `model` (Model)

Required packages: `robustDA`

Support Vector Machines with Boundrange String Kernel

```
method = 'svmBoundrangeString'
```

Type: Regression, Classification

Tuning parameters:

- `length` (length)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Exponential String Kernel

```
method = 'svmExpoString'
```

Type: Regression, Classification

Tuning parameters:

- `lambda` (lambda)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Linear Kernel

```
method = 'svmLinear'
```

Type: Regression, Classification

Tuning parameters:

- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Linear Kernel

```
method = 'svmLinear2'
```

Type: Regression, Classification

Tuning parameters:

- `cost` (Cost)

Required packages: `e1071`

Support Vector Machines with Polynomial Kernel

```
method = 'svmPoly'
```

Type: Regression, Classification

Tuning parameters:

- `degree` (Polynomial Degree)
- `scale` (Scale)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Radial Basis Function Kernel

```
method = 'svmRadial'
```

Type: Regression, Classification

Tuning parameters:

- `sigma` (Sigma)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Radial Basis Function Kernel

```
method = 'svmRadialSigma'
```

Type: Regression, Classification

Tuning parameters:

- `sigma` (Sigma)
- `C` (Cost)

Required packages: `kernlab`

Notes: This SVM model tunes over the cost parameter and the RBF kernel parameter sigma. In the latter case, using `tuneLength` will, at most, evaluate six values of the kernel parameter. This enables a broad search over the cost parameter and a relatively narrow search over `sigma`

Support Vector Machines with Spectrum String Kernel

```
method = 'svmSpectrumString'
```

Type: Regression, Classification

Tuning parameters:

- `length` (length)
- `C` (Cost)

Required packages: `kernlab`

7.0.42 Robust Model

(back to contents)

Quantile Random Forest

```
method = 'qrf'
```

Type: Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `quantregForest`

Quantile Regression Neural Network

```
method = 'qrnn'
```

Type: Regression

Tuning parameters:

- `n.hidden` (#Hidden Units)
- `penalty` (Weight Decay)
- `bag` (Bagged Models?)

Required packages: `qrnn`

Robust Linear Discriminant Analysis

```
method = 'Linda'
```

Type: Classification

No tuning parameters for this model

Required packages: `rrcov`

Robust Linear Model

```
method = 'rlm'
```

Type: Regression

Tuning parameters:

- `intercept` (intercept)
- `psi` (psi)

Required packages: `MASS`

A model-specific variable importance metric is available.

Robust Regularized Linear Discriminant Analysis

```
method = 'rrlda'
```

Type: Classification

Tuning parameters:

- `lambda` (Penalty Parameter)
- `hp` (Robustness Parameter)
- `penalty` (Penalty Type)

Required packages: `rrlda`

Notes: Unlike other packages used by `train`, the `rrlda` package is fully loaded when this model is used.

Robust SIMCA

```
method = 'RSimca'
```

Type: Classification

No tuning parameters for this model

Required packages: `rrcovHD`

Notes: Unlike other packages used by `train`, the `rrcovHD` package is fully loaded when this model is used.

SIMCA

```
method = 'CSimca'
```

Type: Classification

No tuning parameters for this model

Required packages: `rrcov` , `rrcovHD`

7.0.43 ROC Curves

(back to contents)

ROC-Based Classifier

```
method = 'rocc'
```

Type: Classification

Tuning parameters:

- `xgenes` (#Variables Retained)

Required packages: `rocc`

7.0.44 Rule-Based Model

(back to contents)

Adaptive-Network-Based Fuzzy Inference System

```
method = 'ANFIS'
```

Type: Regression

Tuning parameters:

- `num.labels` (#Fuzzy Terms)
- `max.iter` (Max. Iterations)

Required packages: `frbs`

C5.0

```
method = 'C5.0'
```

Type: Classification

Tuning parameters:

- `trials` (# Boosting Iterations)
- `model` (Model Type)
- `winnow` (Winnow)

Required packages: `C50` , `plyr`

A model-specific variable importance metric is available.

Cost-Sensitive C5.0

```
method = 'C5.0Cost'
```

Type: Classification

Tuning parameters:

- `trials` (# Boosting Iterations)
- `model` (Model Type)
- `winnow` (Winnow)
- `cost` (Cost)

Required packages: `C50` , `plyr`

A model-specific variable importance metric is available.

Cubist

```
method = 'cubist'
```

Type: Regression

Tuning parameters:

- `committees` (#Committees)
- `neighbors` (#Instances)

Required packages: `Cubist`

A model-specific variable importance metric is available.

Dynamic Evolving Neural-Fuzzy Inference System

```
method = 'DENFIS'
```

Type: Regression

Tuning parameters:

- `Dthr` (Threshold)
- `max.iter` (Max. Iterations)

Required packages: `frbs`

Fuzzy Inference Rules by Descent Method

```
method = 'FIR.DM'
```

Type: Regression

Tuning parameters:

- `num.labels` (#Fuzzy Terms)
- `max.iter` (Max. Iterations)

Required packages: `frbs`

Fuzzy Rules Using Chi's Method

```
method = 'FRBCS.CHI'
```

Type: Classification

Tuning parameters:

- num.labels (#Fuzzy Terms)
- type.mf (Membership Function)

Required packages: frbs

Fuzzy Rules Using Genetic Cooperative-Competitive Learning and Pittsburgh

```
method = 'FH.GBML'
```

Type: Classification

Tuning parameters:

- max.num.rule (Max. #Rules)
- popu.size (Population Size)
- max.gen (Max. Generations)

Required packages: frbs

Fuzzy Rules Using the Structural Learning Algorithm on Vague Environment

```
method = 'SLAVE'
```

Type: Classification

Tuning parameters:

- num.labels (#Fuzzy Terms)
- max.iter (Max. Iterations)

- `max.gen` (Max. Generations)

Required packages: `frbs`

Fuzzy Rules via MOGUL

```
method = 'GFS.FR.MOGUL'
```

Type: Regression

Tuning parameters:

- `max.gen` (Max. Generations)
- `max.iter` (Max. Iterations)
- `max.tune` (Max. Tuning Iterations)

Required packages: `frbs`

Fuzzy Rules via Thrift

```
method = 'GFS.THRIFT'
```

Type: Regression

Tuning parameters:

- `popu.size` (Population Size)
- `num.labels` (# Fuzzy Labels)
- `max.gen` (Max. Generations)

Required packages: `frbs`

Fuzzy Rules with Weight Factor

```
method = 'FRBCS.W'
```

Type: Classification

Tuning parameters:

- `num.labels` (#Fuzzy Terms)
- `type.mf` (Membership Function)

Required packages: `frbs`

Genetic Lateral Tuning and Rule Selection of Linguistic Fuzzy Systems

```
method = 'GFS.LT.RS'
```

Type: Regression

Tuning parameters:

- `popu.size` (Population Size)
- `num.labels` (# Fuzzy Labels)
- `max.gen` (Max. Generations)

Required packages: `frbs`

Hybrid Neural Fuzzy Inference System

```
method = 'HYFIS'
```

Type: Regression

Tuning parameters:

- `num.labels` (#Fuzzy Terms)
- `max.iter` (Max. Iterations)

Required packages: `frbs`

Model Rules

```
method = 'M5Rules'
```

Type: Regression

Tuning parameters:

- `pruned` (Pruned)
- `smoothed` (Smoothed)

Required packages: `RWeka`

Model Tree

```
method = 'M5'
```

Type: Regression

Tuning parameters:

- `pruned` (Pruned)
- `smoothed` (Smoothed)

- `rules` (Rules)

Required packages: `RWeka`

Patient Rule Induction Method

```
method = 'PRIM'
```

Type: Classification

Tuning parameters:

- `peel.alpha` (peeling quantile)
- `paste.alpha` (pasting quantile)
- `mass.min` (minimum mass)

Required packages: `supervisedPRIM`

Random Forest Rule-Based Model

```
method = 'rfRules'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `maxdepth` (Maximum Rule Depth)

Required packages: `randomForest` , `inTrees` , `plyr`

A model-specific variable importance metric is available.

Rule-Based Classifier

```
method = 'JRip'
```

Type: Classification

Tuning parameters:

- `NumOpt` (# Optimizations)
- `NumFolds` (# Folds)
- `MinWeights` (Min Weights)

Required packages: `RWeka`

A model-specific variable importance metric is available.

Rule-Based Classifier

```
method = 'PART'
```

Type: Classification

Tuning parameters:

- `threshold` (Confidence Threshold)
- `pruned` (Pruning)

Required packages: `RWeka`

A model-specific variable importance metric is available.

Simplified TSK Fuzzy Rules

```
method = 'FS.HGD'
```

Type: Regression

Tuning parameters:

- `num.labels` (#Fuzzy Terms)
- `max.iter` (Max. Iterations)

Required packages: `frbs`

Single C5.0 Ruleset

```
method = 'C5.0Rules'
```

Type: Classification

No tuning parameters for this model

Required packages: `C50`

A model-specific variable importance metric is available.

Single Rule Classification

```
method = 'OneR'
```

Type: Classification

No tuning parameters for this model

Required packages: `RWeka`

Subtractive Clustering and Fuzzy c-Means Rules

```
method = 'SBC'
```

Type: Regression

Tuning parameters:

- `r.a` (Radius)
- `eps.high` (Upper Threshold)
- `eps.low` (Lower Threshold)

Required packages: `frbs`

Wang and Mendel Fuzzy Rules

```
method = 'WM'
```

Type: Regression

Tuning parameters:

- `num.labels` (#Fuzzy Terms)
- `type.mf` (Membership Function)

Required packages: `frbs`

7.0.45 Self-Organising Maps

(back to contents)

Self-Organizing Maps

```
method = 'xyf'
```

Type: Classification, Regression

Tuning parameters:

- `xdim` (Rows)
- `ydim` (Columns)
- `user.weights` (Layer Weight)
- `topo` (Topology)

Required packages: `kohonen`

Notes: As of version 3.0.0 of the kohonen package, the argument `user.weights` replaces the old `alpha` parameter. `user.weights` is usually a vector of relative weights such as `c(1, 3)` but is parameterized here as a proportion such as `c(1-.75, .75)` where the .75 is the value of the tuning parameter passed to `train` and indicates that the outcome layer has 3 times the weight as the predictor layer.

7.0.46 String Kernel

(back to contents)

Support Vector Machines with Boundrange String Kernel

```
method = 'svmBoundrangeString'
```

Type: Regression, Classification

Tuning parameters:

- `length` (length)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Exponential String Kernel

```
method = 'svmExpoString'
```

Type: Regression, Classification

Tuning parameters:

- `lambda` (lambda)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Spectrum String Kernel

```
method = 'svmSpectrumString'
```

Type: Regression, Classification

Tuning parameters:

- `length` (length)
- `C` (Cost)

Required packages: `kernlab`

7.0.47 Support Vector Machines

(back to contents)

L2 Regularized Linear Support Vector Machines with Class Weights

```
method = 'svmLinearWeights2'
```

Type: Classification

Tuning parameters:

- `cost` (Cost)
- `Loss` (Loss Function)
- `weight` (Class Weight)

Required packages: `LiblineaR`

L2 Regularized Support Vector Machine (dual) with Linear Kernel

```
method = 'svmLinear3'
```

Type: Regression, Classification

Tuning parameters:

- `cost` (Cost)
- `Loss` (Loss Function)

Required packages: `LiblineaR`

Least Squares Support Vector Machine

```
method = 'lssvmLinear'
```

Type: Classification

Tuning parameters:

- `tau` (Regularization Parameter)

Required packages: `kernlab`

Least Squares Support Vector Machine with Polynomial Kernel

```
method = 'lssvmPoly'
```

Type: Classification

Tuning parameters:

- `degree` (Polynomial Degree)
- `scale` (Scale)
- `tau` (Regularization Parameter)

Required packages: `kernlab`

Least Squares Support Vector Machine with Radial Basis Function Kernel

```
method = 'lssvmRadial'
```

Type: Classification

Tuning parameters:

- `sigma` (Sigma)
- `tau` (Regularization Parameter)

Required packages: `kernlab`

Linear Support Vector Machines with Class Weights

```
method = 'svmLinearWeights'
```

Type: Classification

Tuning parameters:

- `cost` (Cost)
- `weight` (Class Weight)

Required packages: `e1071`

Support Vector Machines with Boundrange String Kernel

```
method = 'svmBoundrangeString'
```

Type: Regression, Classification

Tuning parameters:

- `length` (length)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Class Weights

```
method = 'svmRadialWeights'
```

Type: Classification

Tuning parameters:

- `sigma` (Sigma)
- `C` (Cost)
- `Weight` (Weight)

Required packages: `kernlab`

Support Vector Machines with Exponential String Kernel

```
method = 'svmExpoString'
```

Type: Regression, Classification

Tuning parameters:

- `lambda` (lambda)

- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Linear Kernel

```
method = 'svmLinear'
```

Type: Regression, Classification

Tuning parameters:

- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Linear Kernel

```
method = 'svmLinear2'
```

Type: Regression, Classification

Tuning parameters:

- `cost` (Cost)

Required packages: `e1071`

Support Vector Machines with Polynomial Kernel

```
method = 'svmPoly'
```

Type: Regression, Classification

Tuning parameters:

- `degree` (Polynomial Degree)
- `scale` (Scale)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Radial Basis Function Kernel

```
method = 'svmRadial'
```

Type: Regression, Classification

Tuning parameters:

- `sigma` (Sigma)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Radial Basis Function Kernel

```
method = 'svmRadialCost'
```

Type: Regression, Classification

Tuning parameters:

- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Radial Basis Function Kernel

```
method = 'svmRadialSigma'
```

Type: Regression, Classification

Tuning parameters:

- `sigma` (Sigma)
- `C` (Cost)

Required packages: `kernlab`

Notes: This SVM model tunes over the cost parameter and the RBF kernel parameter sigma. In the latter case, using `tuneLength` will, at most, evaluate six values of the kernel parameter. This enables a broad search over the cost parameter and a relatively narrow search over `sigma`

Support Vector Machines with Spectrum String Kernel

```
method = 'svmSpectrumString'
```

Type: Regression, Classification

Tuning parameters:

- `length` (length)
- `C` (Cost)

Required packages: `kernlab`

7.0.48 Supports Class Probabilities

(back to contents)

AdaBoost Classification Trees

```
method = 'adaboost'
```

Type: Classification

Tuning parameters:

- `nIter` (#Trees)
- `method` (Method)

Required packages: `fastAdaboost`

AdaBoost.M1

```
method = 'AdaBoost.M1'
```

Type: Classification

Tuning parameters:

- `mfinal` (#Trees)
- `maxdepth` (Max Tree Depth)
- `coeflearn` (Coefficient Type)

Required packages: `adabag` , `plyr`

A model-specific variable importance metric is available.

Adaptive Mixture Discriminant Analysis

```
method = 'amdai'
```

Type: Classification

Tuning parameters:

- `model` (Model Type)

Required packages: `adaptDA`

Adjacent Categories Probability Model for Ordinal Data

```
method = 'vglmAdjCat'
```

Type: Classification

Tuning parameters:

- `parallel` (Parallel Curves)
- `link` (Link Function)

Required packages: `VGAM`

Bagged AdaBoost

```
method = 'AdaBag'
```

Type: Classification

Tuning parameters:

- `mfinal` (#Trees)
- `maxdepth` (Max Tree Depth)

Required packages: `adabag`, `plyr`

A model-specific variable importance metric is available.

Bagged CART

```
method = 'treebag'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `ipred`, `plyr`, `e1071`

A model-specific variable importance metric is available.

Bagged Flexible Discriminant Analysis

```
method = 'bagFDA'
```

Type: Classification

Tuning parameters:

- `degree` (Product Degree)
- `nprune` (#Terms)

Required packages: `earth` , `mda`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train` , the `earth` package is fully loaded when this model is used.

Bagged Logic Regression

```
method = 'logicBag'
```

Type: Regression, Classification

Tuning parameters:

- `nleaves` (Maximum Number of Leaves)
- `ntrees` (Number of Trees)

Required packages: `logicFS`

Notes: Unlike other packages used by `train` , the `logicFS` package is fully loaded when this model is used.

Bagged MARS

```
method = 'bagEarth'
```

Type: Regression, Classification

Tuning parameters:

- `nprune` (#Terms)
- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Bagged MARS using gCV Pruning

```
method = 'bagEarthGCV'
```

Type: Regression, Classification

Tuning parameters:

- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Bagged Model

```
method = 'bag'
```

Type: Regression, Classification

Tuning parameters:

- `vars` (#Randomly Selected Predictors)

Required packages: `caret`

Bayesian Additive Regression Trees

```
method = 'bartMachine'
```

Type: Classification, Regression

Tuning parameters:

- `num_trees` (#Trees)
- `k` (Prior Boundary)
- `alpha` (Base Terminal Node Hyperparameter)
- `beta` (Power Terminal Node Hyperparameter)
- `nu` (Degrees of Freedom)

Required packages: `bartMachine`

A model-specific variable importance metric is available.

Bayesian Generalized Linear Model

```
method = 'bayesglm'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `arm`

Binary Discriminant Analysis

```
method = 'binda'
```

Type: Classification

Tuning parameters:

- `lambda.freqs` (Shrinkage Intensity)

Required packages: `binda`

Boosted Classification Trees

```
method = 'ada'
```

Type: Classification

Tuning parameters:

- `iter` (#Trees)
- `maxdepth` (Max Tree Depth)
- `nu` (Learning Rate)

Required packages: `ada` , `plyr`

Boosted Generalized Additive Model

```
method = 'gamboost'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `prune` (AIC Prune?)

Required packages: `mboost` , `plyr` , `import`

Notes: The `prune` option for this model enables the number of iterations to be determined by the optimal AIC value across all iterations. See the examples in `?mboost::mstop` . If pruning is not used, the ensemble makes predictions using the exact value of the `mstop` tuning parameter value.

Boosted Generalized Linear Model

```
method = 'glmboost'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `prune` (AIC Prune?)

Required packages: `plyr` , `mboost`

A model-specific variable importance metric is available. Notes: The `prune` option for this model enables the number of iterations to be determined by the optimal AIC value across all iterations. See the examples in `?mboost::mstop` . If pruning is not used, the ensemble makes predictions using the exact value of the `mstop` tuning parameter value.

Boosted Logistic Regression

```
method = 'LogitBoost'
```

Type: Classification

Tuning parameters:

- `nIter` (# Boosting Iterations)

Required packages: `caTools`

Boosted Tree

```
method = 'blackboost'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (#Trees)
- `maxdepth` (Max Tree Depth)

Required packages: `party` , `mboost` , `plyr`

C4.5-like Trees

```
method = 'J48'
```

Type: Classification

Tuning parameters:

- `C` (Confidence Threshold)
- `M` (Minimum Instances Per Leaf)

Required packages: `RWeka`

C5.0

```
method = 'C5.0'
```

Type: Classification

Tuning parameters:

- `trials` (# Boosting Iterations)
- `model` (Model Type)
- `winnow` (Winnow)

Required packages: `C50` , `plyr`

A model-specific variable importance metric is available.

CART

```
method = 'rpart'
```

Type: Regression, Classification

Tuning parameters:

- `cp` (Complexity Parameter)

Required packages: `rpart`

A model-specific variable importance metric is available.

CART

```
method = 'rpart1SE'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `rpart`

A model-specific variable importance metric is available. Notes: This CART model replicates the same process used by the `rpart` function where the model complexity is determined using the one-standard error method. This procedure is replicated inside of the resampling done by `train` so that an external resampling estimate can be obtained.

CART

```
method = 'rpart2'
```

Type: Regression, Classification

Tuning parameters:

- `maxdepth` (Max Tree Depth)

Required packages: `rpart`

A model-specific variable importance metric is available.

CHi-squared Automated Interaction Detection

```
method = 'chaid'
```

Type: Classification

Tuning parameters:

- `alpha2` (Merging Threshold)
- `alpha3` (Splitting former Merged Threshold)
- `alpha4` (Splitting former Merged Threshold)

Required packages: `CHAID`

Conditional Inference Random Forest

```
method = 'cforest'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `party`

A model-specific variable importance metric is available.

Conditional Inference Tree

```
method = 'ctree'
```

Type: Classification, Regression

Tuning parameters:

- `mincriterion` (1 - P-Value Threshold)

Required packages: `party`

Conditional Inference Tree

```
method = 'ctree2'
```

Type: Regression, Classification

Tuning parameters:

- `maxdepth` (Max Tree Depth)
- `mincriterion` (1 - P-Value Threshold)

Required packages: `party`

Continuation Ratio Model for Ordinal Data

```
method = 'vglmContRatio'
```

Type: Classification

Tuning parameters:

- `parallel` (Parallel Curves)
- `link` (Link Function)

Required packages: `VGAM`

Cumulative Probability Model for Ordinal Data

```
method = 'vglmCumulative'
```

Type: Classification

Tuning parameters:

- `parallel` (Parallel Curves)
- `link` (Link Function)

Required packages: `VGAM`

Diagonal Discriminant Analysis

```
method = 'dda'
```

Type: Classification

Tuning parameters:

- `model` (Model)
- `shrinkage` (Shrinkage Type)

Required packages: `sparsediscrim`

Distance Weighted Discrimination with Polynomial Kernel

```
method = 'dwdPoly'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)
- `degree` (Polynomial Degree)
- `scale` (Scale)

Required packages: `kerndwd`

Distance Weighted Discrimination with Radial Basis Function Kernel

```
method = 'dwdRadial'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)
- `sigma` (Sigma)

Required packages: `kernlab` , `kerndwd`

Ensembles of Generalized Linear Models

```
method = 'randomGLM'
```

Type: Regression, Classification

Tuning parameters:

- `maxInteractionOrder` (Interaction Order)

Required packages: `randomGLM`

Notes: Unlike other packages used by `train`, the `randomGLM` package is fully loaded when this model is used.

eXtreme Gradient Boosting

```
method = 'xgbDART'
```

Type: Regression, Classification

Tuning parameters:

- `nrounds` (# Boosting Iterations)
- `max_depth` (Max Tree Depth)
- `eta` (Shrinkage)
- `gamma` (Minimum Loss Reduction)
- `subsample` (Subsample Percentage)
- `colsample_bytree` (Subsample Ratio of Columns)
- `rate_drop` (Fraction of Trees Dropped)
- `skip_drop` (Prob. of Skipping Drop-out)
- `min_child_weight` (Minimum Sum of Instance Weight)

Required packages: `xgboost` , `plyr`

A model-specific variable importance metric is available.

eXtreme Gradient Boosting

```
method = 'xgbLinear'
```

Type: Regression, Classification

Tuning parameters:

- `nrounds` (# Boosting Iterations)
- `lambda` (L2 Regularization)
- `alpha` (L1 Regularization)
- `eta` (Learning Rate)

Required packages: `xgboost`

A model-specific variable importance metric is available.

eXtreme Gradient Boosting

```
method = 'xgbTree'
```

Type: Regression, Classification

Tuning parameters:

- `nrounds` (# Boosting Iterations)
- `max_depth` (Max Tree Depth)
- `eta` (Shrinkage)

- `gamma` (Minimum Loss Reduction)
- `colsample_bytree` (Subsample Ratio of Columns)
- `min_child_weight` (Minimum Sum of Instance Weight)
- `subsample` (Subsample Percentage)

Required packages: `xgboost` , `plyr`

A model-specific variable importance metric is available.

Flexible Discriminant Analysis

```
method = 'fda'
```

Type: Classification

Tuning parameters:

- `degree` (Product Degree)
- `nprune` (#Terms)

Required packages: `earth` , `mda`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train` , the `earth` package is fully loaded when this model is used.

Gaussian Process

```
method = 'gausssprLinear'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `kernlab`

Gaussian Process with Polynomial Kernel

```
method = 'gausssprPoly'
```

Type: Regression, Classification

Tuning parameters:

- `degree` (Polynomial Degree)
- `scale` (Scale)

Required packages: `kernlab`

Gaussian Process with Radial Basis Function Kernel

```
method = 'gausssprRadial'
```

Type: Regression, Classification

Tuning parameters:

- `sigma` (Sigma)

Required packages: `kernlab`

Generalized Additive Model using LOESS

```
method = 'gamLoess'
```

Type: Regression, Classification

Tuning parameters:

- `span` (Span)
- `degree` (Degree)

Required packages: `gam`

A model-specific variable importance metric is available. Notes: Which terms enter the model in a nonlinear manner is determined by the number of unique values for the predictor. For example, if a predictor only has four unique values, most basis expansion method will fail because there are not enough granularity in the data. By default, a predictor must have at least 10 unique values to be used in a nonlinear basis expansion. Unlike other packages used by `train`, the `gam` package is fully loaded when this model is used.

Generalized Additive Model using Splines

```
method = 'bam'
```

Type: Regression, Classification

Tuning parameters:

- `select` (Feature Selection)
- `method` (Method)

Required packages: `mgcv`

A model-specific variable importance metric is available. Notes: Which terms enter the model in a nonlinear manner is determined by the number of unique values for the predictor. For example, if a predictor only has four unique values, most basis expansion method will fail because there are not enough granularity in the data. By default, a predictor must have at least 10 unique values to be used in a nonlinear basis expansion. Unlike other packages used by `train`, the `mgcv` package is fully loaded when this model is used.

Generalized Additive Model using Splines

```
method = 'gam'
```

Type: Regression, Classification

Tuning parameters:

- `select` (Feature Selection)
- `method` (Method)

Required packages: `mgcv`

A model-specific variable importance metric is available. Notes: Which terms enter the model in a nonlinear manner is determined by the number of unique values for the predictor. For example, if a predictor only has four unique values, most basis expansion method will fail because there are not enough granularity in the data. By default, a predictor must have at least 10 unique values to be used in a nonlinear basis expansion. Unlike other packages used by `train`, the `mgcv` package is fully loaded when this model is used.

Generalized Additive Model using Splines

```
method = 'gamSpline'
```

Type: Regression, Classification

Tuning parameters:

- `df` (Degrees of Freedom)

Required packages: `gam`

A model-specific variable importance metric is available. Notes: Which terms enter the model in a nonlinear manner is determined by the number of unique values for the predictor. For example, if a predictor only has four unique values, most basis expansion method will fail because there are not enough granularity in the data. By default, a predictor must have at least 10 unique values to be used in a nonlinear basis expansion. Unlike other packages used by `train` , the `gam` package is fully loaded when this model is used.

Generalized Linear Model

```
method = 'glm'
```

Type: Regression, Classification

No tuning parameters for this model

A model-specific variable importance metric is available.

Generalized Linear Model with Stepwise Feature Selection

```
method = 'glmStepAIC'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `MASS`

Generalized Partial Least Squares

```
method = 'gpls'
```

Type: Classification

Tuning parameters:

- `K.prov` (#Components)

Required packages: `gpls`

glmnet

```
method = 'glmnet_h2o'
```

Type: Regression, Classification

Tuning parameters:

- `alpha` (Mixing Percentage)

- `lambda` (Regularization Parameter)

Required packages: `h2o`

A model-specific variable importance metric is available.

glmnet

```
method = 'glmnet'
```

Type: Regression, Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `lambda` (Regularization Parameter)

Required packages: `glmnet` , `Matrix`

A model-specific variable importance metric is available.

Gradient Boosting Machines

```
method = 'gbm_h2o'
```

Type: Regression, Classification

Tuning parameters:

- `ntrees` (# Boosting Iterations)
- `max_depth` (Max Tree Depth)
- `min_rows` (Min. Terminal Node Size)

- `learn_rate` (Shrinkage)
- `col_sample_rate` (#Randomly Selected Predictors)

Required packages: `h2o`

A model-specific variable importance metric is available.

Heteroscedastic Discriminant Analysis

```
method = 'hda'
```

Type: Classification

Tuning parameters:

- `gamma` (Gamma)
- `lambda` (Lambda)
- `newdim` (Dimension of the Discriminative Subspace)

Required packages: `hda`

High Dimensional Discriminant Analysis

```
method = 'hdda'
```

Type: Classification

Tuning parameters:

- `threshold` (Threshold)
- `model` (Model Type)

Required packages: `HDclassif`

High-Dimensional Regularized Discriminant Analysis

```
method = 'hdrda'
```

Type: Classification

Tuning parameters:

- `gamma` (Gamma)
- `lambda` (Lambda)
- `shrinkage_type` (Shrinkage Type)

Required packages: `sparsediscrim`

k-Nearest Neighbors

```
method = 'kknn'
```

Type: Regression, Classification

Tuning parameters:

- `kmax` (Max. #Neighbors)
- `distance` (Distance)
- `kernel` (Kernel)

Required packages: `kknn`

k-Nearest Neighbors

```
method = 'knn'
```

Type: Classification, Regression

Tuning parameters:

- `k` (#Neighbors)

Linear Discriminant Analysis

```
method = 'lda'
```

Type: Classification

No tuning parameters for this model

Required packages: `MASS`

Linear Discriminant Analysis

```
method = 'lda2'
```

Type: Classification

Tuning parameters:

- `dimen` (#Discriminant Functions)

Required packages: `MASS`

Linear Discriminant Analysis with Stepwise Feature Selection

```
method = 'stepLDA'
```

Type: Classification

Tuning parameters:

- `maxvar` (Maximum #Variables)
- `direction` (Search Direction)

Required packages: `klaR` , `MASS`

Linear Distance Weighted Discrimination

```
method = 'dwdLinear'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)

Required packages: `kerndwd`

Linear Support Vector Machines with Class Weights

```
method = 'svmLinearWeights'
```

Type: Classification

Tuning parameters:

- `cost` (Cost)
- `weight` (Class Weight)

Required packages: `e1071`

Localized Linear Discriminant Analysis

```
method = 'loclda'
```

Type: Classification

Tuning parameters:

- `k` (#Nearest Neighbors)

Required packages: `klaR`

Logic Regression

```
method = 'logreg'
```

Type: Regression, Classification

Tuning parameters:

- `treesize` (Maximum Number of Leaves)
- `ntrees` (Number of Trees)

Required packages: `LogicReg`

Logistic Model Trees

```
method = 'LMT'
```

Type: Classification

Tuning parameters:

- `iter` (# Iteratons)

Required packages: `RWeka`

Mixture Discriminant Analysis

```
method = 'mda'
```

Type: Classification

Tuning parameters:

- `subclasses` (#Subclasses Per Class)

Required packages: `mda`

Model Averaged Naive Bayes Classifier

```
method = 'manb'
```

Type: Classification

Tuning parameters:

- `smooth` (Smoothing Parameter)

- `prior` (Prior Probability)

Required packages: `bnclassify`

Model Averaged Neural Network

```
method = 'avNNet'
```

Type: Classification, Regression

Tuning parameters:

- `size` (#Hidden Units)
- `decay` (Weight Decay)
- `bag` (Bagging)

Required packages: `nnet`

Monotone Multi-Layer Perceptron Neural Network

```
method = 'monmlp'
```

Type: Classification, Regression

Tuning parameters:

- `hidden1` (#Hidden Units)
- `n.ensemble` (#Models)

Required packages: `monmlp`

Multi-Layer Perceptron

```
method = 'mlp'
```

Type: Regression, Classification

Tuning parameters:

- `size` (#Hidden Units)

Required packages: `RSNNS`

Multi-Layer Perceptron

```
method = 'mlpWeightDecay'
```

Type: Regression, Classification

Tuning parameters:

- `size` (#Hidden Units)
- `decay` (Weight Decay)

Required packages: `RSNNS`

Multi-Layer Perceptron, multiple layers

```
method = 'mlpWeightDecayML'
```

Type: Regression, Classification

Tuning parameters:

- `layer1` (#Hidden Units layer1)
- `layer2` (#Hidden Units layer2)
- `layer3` (#Hidden Units layer3)
- `decay` (Weight Decay)

Required packages: `RSNNS`

Multi-Layer Perceptron, with multiple layers

```
method = 'mlpML'
```

Type: Regression, Classification

Tuning parameters:

- `layer1` (#Hidden Units layer1)
- `layer2` (#Hidden Units layer2)
- `layer3` (#Hidden Units layer3)

Required packages: `RSNNS`

Multi-Step Adaptive MCP-Net

```
method = 'msaenet'
```

Type: Regression, Classification

Tuning parameters:

- `alphas` (Alpha)
- `nsteps` (#Adaptive Estimation Steps)

- `scale` (Adaptive Weight Scaling Factor)

Required packages: `msaenet`

A model-specific variable importance metric is available.

Multilayer Perceptron Network by Stochastic Gradient Descent

```
method = 'mlpSGD'
```

Type: Regression, Classification

Tuning parameters:

- `size` (#Hidden Units)
- `l2reg` (L2 Regularization)
- `lambda` (RMSE Gradient Scaling)
- `learn_rate` (Learning Rate)
- `momentum` (Momentum)
- `gamma` (Learning Rate Decay)
- `minibatchsz` (Batch Size)
- `repeats` (#Models)

Required packages: `FCNN4R` , `plyr`

A model-specific variable importance metric is available.

Multilayer Perceptron Network with Dropout

```
method = 'mlpKerasDropout'
```

Type: Regression, Classification

Tuning parameters:

- `size` (#Hidden Units)
- `dropout` (Dropout Rate)
- `batch_size` (Batch Size)
- `lr` (Learning Rate)
- `rho` (Rho)
- `decay` (Learning Rate Decay)
- `activation` (Activation Function)

Required packages: `keras`

Notes: After `train` completes, the keras model object is serialized so that it can be used between R session. When predicting, the code will temporarily unsearalize the object. To make the predictions more efficient, the user might want to use `keras::unsearlize_model(object$finalModel$object)` in the current R session so that that operation is only done once. Also, this model cannot be run in parallel due to the nature of how tensorflow does the computations. Unlike other packages used by `train`, the `dplyr` package is fully loaded when this model is used.

Multilayer Perceptron Network with Dropout

```
method = 'mlpKerasDropoutCost'
```

Type: Classification

Tuning parameters:

- `size` (#Hidden Units)
- `dropout` (Dropout Rate)
- `batch_size` (Batch Size)
- `lr` (Learning Rate)
- `rho` (Rho)
- `decay` (Learning Rate Decay)
- `cost` (Cost)
- `activation` (Activation Function)

Required packages: `keras`

Notes: After `train` completes, the keras model object is serialized so that it can be used between R session. When predicting, the code will temporarily unsearalize the object. To make the predictions more efficient, the user might want to use `keras::unsearlize_model(object$finalModel$object)` in the current R session so that that operation is only done once. Also, this model cannot be run in parallel due to the nature of how tensorflow does the computations. Finally, the cost parameter weights the first class in the outcome vector. Unlike other packages used by `train`, the `dplyr` package is fully loaded when this model is used.

Multilayer Perceptron Network with Weight Decay

```
method = 'mlpKerasDecay'
```

Type: Regression, Classification

Tuning parameters:

- `size` (#Hidden Units)
- `lambda` (L2 Regularization)
- `batch_size` (Batch Size)
- `lr` (Learning Rate)
- `rho` (Rho)
- `decay` (Learning Rate Decay)
- `activation` (Activation Function)

Required packages: `keras`

Notes: After `train` completes, the keras model object is serialized so that it can be used between R session. When predicting, the code will temporarily unsearalize the object. To make the predictions more efficient, the user might want to use `keras::unsearlize_model(object$finalModel$object)` in the current R session so that that operation is only done once. Also, this model cannot be run in parallel due to the nature of how tensorflow does the computations. Unlike other packages used by `train`, the `dplyr` package is fully loaded when this model is used.

Multilayer Perceptron Network with Weight Decay

```
method = 'mlpKerasDecayCost'
```

Type: Classification

Tuning parameters:

- `size` (#Hidden Units)
- `lambda` (L2 Regularization)
- `batch_size` (Batch Size)
- `lr` (Learning Rate)
- `rho` (Rho)
- `decay` (Learning Rate Decay)
- `cost` (Cost)
- `activation` (Activation Function)

Required packages: `keras`

Notes: After `train` completes, the keras model object is serialized so that it can be used between R session. When predicting, the code will temporarily unsearalize the object. To make the predictions more efficient, the user might want to use `keras::unsearlize_model(object$finalModel$object)` in the current R session so that that operation is only done once. Also, this model cannot be run in parallel due to the nature of how tensorflow does the computations. Finally, the cost parameter weights the first class in the outcome vector. Unlike other packages used by `train`, the `dplyr` package is fully loaded when this model is used.

Multivariate Adaptive Regression Spline

```
method = 'earth'
```

Type: Regression, Classification

Tuning parameters:

- `nprune` (#Terms)
- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Multivariate Adaptive Regression Splines

```
method = 'gcvEarth'
```

Type: Regression, Classification

Tuning parameters:

- `degree` (Product Degree)

Required packages: `earth`

A model-specific variable importance metric is available. Notes: Unlike other packages used by `train`, the `earth` package is fully loaded when this model is used.

Naive Bayes

```
method = 'naive_bayes'
```

Type: Classification

Tuning parameters:

- `laplace` (Laplace Correction)
- `usekernel` (Distribution Type)
- `adjust` (Bandwidth Adjustment)

Required packages: `naivebayes`

Naive Bayes

```
method = 'nb'
```

Type: Classification

Tuning parameters:

- `fL` (Laplace Correction)
- `usekernel` (Distribution Type)
- `adjust` (Bandwidth Adjustment)

Required packages: `klaR`

Naive Bayes Classifier

```
method = 'nbDiscrete'
```

Type: Classification

Tuning parameters:

- `smooth` (Smoothing Parameter)

Required packages: `bnclassify`

Naive Bayes Classifier with Attribute Weighting

```
method = 'awnb'
```

Type: Classification

Tuning parameters:

- `smooth` (Smoothing Parameter)

Required packages: `bnclassify`

Nearest Shrunken Centroids

```
method = 'pam'
```

Type: Classification

Tuning parameters:

- `threshold` (Shrinkage Threshold)

Required packages: `pamr`

A model-specific variable importance metric is available.

Neural Network

```
method = 'mxnet'
```

Type: Classification, Regression

Tuning parameters:

- `layer1` (#Hidden Units in Layer 1)
- `layer2` (#Hidden Units in Layer 2)
- `layer3` (#Hidden Units in Layer 3)
- `learning.rate` (Learning Rate)
- `momentum` (Momentum)
- `dropout` (Dropout Rate)
- `activation` (Activation Function)

Required packages: `mxnet`

Notes: The `mxnet` package is not yet on CRAN. See http://mxnet.io for installation instructions.

Neural Network

```
method = 'mxnetAdam'
```

Type: Classification, Regression

Tuning parameters:

- `layer1` (#Hidden Units in Layer 1)
- `layer2` (#Hidden Units in Layer 2)
- `layer3` (#Hidden Units in Layer 3)
- `dropout` (Dropout Rate)
- `beta1` (beta1)
- `beta2` (beta2)
- `learningrate` (Learning Rate)

- `activation` (Activation Function)

Required packages: `mxnet`

Notes: The `mxnet` package is not yet on CRAN. See http://mxnet.io for installation instructions. Users are strongly advised to define `num.round` themselves.

Neural Network

```
method = 'nnet'
```

Type: Classification, Regression

Tuning parameters:

- `size` (#Hidden Units)
- `decay` (Weight Decay)

Required packages: `nnet`

A model-specific variable importance metric is available.

Neural Networks with Feature Extraction

```
method = 'pcaNNet'
```

Type: Classification, Regression

Tuning parameters:

- `size` (#Hidden Units)

- `decay` (Weight Decay)

Required packages: `nnet`

Non-Informative Model

```
method = 'null'
```

Type: Classification, Regression

No tuning parameters for this model

Notes: Since this model always predicts the same value, R-squared values will always be estimated to be NA.

Oblique Random Forest

```
method = 'ORFlog'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Oblique Random Forest

```
method = 'ORFpls'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Oblique Random Forest

```
method = 'ORFridge'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Oblique Random Forest

```
method = 'ORFsvm'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Ordered Logistic or Probit Regression

```
method = 'polr'
```

Type: Classification

Tuning parameters:

- `method` (parameter)

Required packages: `MASS`

A model-specific variable importance metric is available.

Parallel Random Forest

```
method = 'parRF'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `e1071` , `randomForest` , `foreach` , `import`

A model-specific variable importance metric is available.

Partial Least Squares

```
method = 'kernelpls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Partial Least Squares

```
method = 'pls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Partial Least Squares

```
method = 'simpls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Partial Least Squares

```
method = 'widekernelpls'
```

Type: Regression, Classification

Tuning parameters:

- `ncomp` (#Components)

Required packages: `pls`

A model-specific variable importance metric is available.

Partial Least Squares Generalized Linear Models

```
method = 'plsRglm'
```

Type: Classification, Regression

Tuning parameters:

- `nt` (#PLS Components)
- `alpha.pvals.expli` (p-Value threshold)

Required packages: `plsRglm`

Notes: Unlike other packages used by `train`, the `plsRglm` package is fully loaded when this model is used.

Patient Rule Induction Method

```
method = 'PRIM'
```

Type: Classification

Tuning parameters:

- `peel.alpha` (peeling quantile)
- `paste.alpha` (pasting quantile)
- `mass.min` (minimum mass)

Required packages: `supervisedPRIM`

Penalized Discriminant Analysis

```
method = 'pda'
```

Type: Classification

Tuning parameters:

- `lambda` (Shrinkage Penalty Coefficient)

Required packages: `mda`

Penalized Discriminant Analysis

```
method = 'pda2'
```

Type: Classification

Tuning parameters:

- `df` (Degrees of Freedom)

Required packages: `mda`

Penalized Logistic Regression

```
method = 'plr'
```

Type: Classification

Tuning parameters:

- `lambda` (L2 Penalty)
- `cp` (Complexity Parameter)

Required packages: `stepPlr`

Penalized Multinomial Regression

```
method = 'multinom'
```

Type: Classification

Tuning parameters:

- `decay` (Weight Decay)

Required packages: `nnet`

A model-specific variable importance metric is available.

Penalized Ordinal Regression

```
method = 'ordinalNet'
```

Type: Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `criteria` (Selection Criterion)
- `link` (Link Function)

Required packages: `ordinalNet` , `plyr`

A model-specific variable importance metric is available. Notes:
Requires ordinalNet package version >= 2.0

Quadratic Discriminant Analysis

```
method = 'qda'
```

Type: Classification

No tuning parameters for this model

Required packages: `MASS`

Quadratic Discriminant Analysis with Stepwise Feature Selection

```
method = 'stepQDA'
```

Type: Classification

Tuning parameters:

- `maxvar` (Maximum #Variables)
- `direction` (Search Direction)

Required packages: `klaR` , `MASS`

Radial Basis Function Network

```
method = 'rbf'
```

Type: Classification, Regression

Tuning parameters:

- `size` (#Hidden Units)

Required packages: `RSNNS`

Radial Basis Function Network

```
method = 'rbfDDA'
```

Type: Regression, Classification

Tuning parameters:

- `negativeThreshold` (Activation Limit for Conflicting Classes)

Required packages: `RSNNS`

Random Forest

```
method = 'ranger'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `splitrule` (Splitting Rule)
- `min.node.size` (Minimal Node Size)

Required packages: `e1071` , `ranger` , `dplyr`

A model-specific variable importance metric is available.

Random Forest

```
method = 'Rborist'
```

Type: Classification, Regression

Tuning parameters:

- `predFixed` (#Randomly Selected Predictors)
- `minNode` (Minimal Node Size)

Required packages: `Rborist`

A model-specific variable importance metric is available.

Random Forest

```
method = 'rf'
```

Type: Classification, Regression

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `randomForest`

A model-specific variable importance metric is available.

Random Forest by Randomization

```
method = 'extraTrees'
```

Type: Regression, Classification

Tuning parameters:

- `mtry` (# Randomly Selected Predictors)
- `numRandomCuts` (# Random Cuts)

Required packages: `extraTrees`

Regularized Discriminant Analysis

```
method = 'rda'
```

Type: Classification

Tuning parameters:

- `gamma` (Gamma)
- `lambda` (Lambda)

Required packages: `klaR`

Regularized Linear Discriminant Analysis

```
method = 'rlda'
```

Type: Classification

Tuning parameters:

- `estimator` (Regularization Method)

Required packages: `sparsediscrim`

Regularized Logistic Regression

```
method = 'regLogistic'
```

Type: Classification

Tuning parameters:

- `cost` (Cost)
- `loss` (Loss Function)
- `epsilon` (Tolerance)

Required packages: `LiblineaR`

Regularized Random Forest

```
method = 'RRF'
```

Type: Regression, Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `coefReg` (Regularization Value)
- `coefImp` (Importance Coefficient)

Required packages: `randomForest` , `RRF`

A model-specific variable importance metric is available.

Regularized Random Forest

```
method = 'RRFglobal'
```

Type: Regression, Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)
- `coefReg` (Regularization Value)

Required packages: `RRF`

A model-specific variable importance metric is available.

Robust Linear Discriminant Analysis

```
method = 'Linda'
```

Type: Classification

No tuning parameters for this model

Required packages: `rrcov`

Robust Mixture Discriminant Analysis

```
method = 'rmda'
```

Type: Classification

Tuning parameters:

- `K` (#Subclasses Per Class)
- `model` (Model)

Required packages: `robustDA`

Robust Quadratic Discriminant Analysis

```
method = 'QdaCov'
```

Type: Classification

No tuning parameters for this model

Required packages: `rrcov`

Robust Regularized Linear Discriminant Analysis

```
method = 'rrlda'
```

Type: Classification

Tuning parameters:

- `lambda` (Penalty Parameter)
- `hp` (Robustness Parameter)
- `penalty` (Penalty Type)

Required packages: `rrlda`

Notes: Unlike other packages used by `train`, the `rrlda` package is fully loaded when this model is used.

Rotation Forest

```
method = 'rotationForest'
```

Type: Classification

Tuning parameters:

- `K` (#Variable Subsets)
- `L` (Ensemble Size)

Required packages: `rotationForest`

A model-specific variable importance metric is available.

Rotation Forest

```
method = 'rotationForestCp'
```

Type: Classification

Tuning parameters:

- `K` (#Variable Subsets)
- `L` (Ensemble Size)
- `cp` (Complexity Parameter)

Required packages: `rpart` , `plyr` , `rotationForest`

A model-specific variable importance metric is available.

Rule-Based Classifier

```
method = 'JRip'
```

Type: Classification

Tuning parameters:

- `NumOpt` (# Optimizations)
- `NumFolds` (# Folds)
- `MinWeights` (Min Weights)

Required packages: `RWeka`

A model-specific variable importance metric is available.

Rule-Based Classifier

```
method = 'PART'
```

Type: Classification

Tuning parameters:

- `threshold` (Confidence Threshold)
- `pruned` (Pruning)

Required packages: `RWeka`

A model-specific variable importance metric is available.

Self-Organizing Maps

```
method = 'xyf'
```

Type: Classification, Regression

Tuning parameters:

- `xdim` (Rows)
- `ydim` (Columns)
- `user.weights` (Layer Weight)
- `topo` (Topology)

Required packages: `kohonen`

Notes: As of version 3.0.0 of the kohonen package, the argument `user.weights` replaces the old `alpha` parameter. `user.weights` is usually a vector of relative weights such as `c(1, 3)` but is parameterized here as a proportion such as `c(1-.75, .75)` where the .75 is the value of the tuning parameter passed to `train` and indicates that the outcome layer has 3 times the weight as the predictor layer.

Semi-Naive Structure Learner Wrapper

```
method = 'nbSearch'
```

Type: Classification

Tuning parameters:

- `k` (#Folds)

- `epsilon` (Minimum Absolute Improvement)
- `smooth` (Smoothing Parameter)
- `final_smooth` (Final Smoothing Parameter)
- `direction` (Search Direction)

Required packages: `bnclassify`

Shrinkage Discriminant Analysis

```
method = 'sda'
```

Type: Classification

Tuning parameters:

- `diagonal` (Diagonalize)
- `lambda` (shrinkage)

Required packages: `sda`

Single C5.0 Ruleset

```
method = 'C5.0Rules'
```

Type: Classification

No tuning parameters for this model

Required packages: `C50`

A model-specific variable importance metric is available.

Single C5.0 Tree

```
method = 'C5.0Tree'
```

Type: Classification

No tuning parameters for this model

Required packages: `C50`

A model-specific variable importance metric is available.

Single Rule Classification

```
method = 'OneR'
```

Type: Classification

No tuning parameters for this model

Required packages: `RWeka`

Sparse Distance Weighted Discrimination

```
method = 'sdwd'
```

Type: Classification

Tuning parameters:

- `lambda` (L1 Penalty)

- `lambda2` (L2 Penalty)

Required packages: `sdwd`

A model-specific variable importance metric is available.

Sparse Linear Discriminant Analysis

```
method = 'sparseLDA'
```

Type: Classification

Tuning parameters:

- `NumVars` (# Predictors)
- `lambda` (Lambda)

Required packages: `sparseLDA`

Sparse Partial Least Squares

```
method = 'spls'
```

Type: Regression, Classification

Tuning parameters:

- `K` (#Components)
- `eta` (Threshold)
- `kappa` (Kappa)

Required packages: `spls`

Stabilized Linear Discriminant Analysis

```
method = 'slda'
```

Type: Classification

No tuning parameters for this model

Required packages: `ipred`

Stacked AutoEncoder Deep Neural Network

```
method = 'dnn'
```

Type: Classification, Regression

Tuning parameters:

- `layer1` (Hidden Layer 1)
- `layer2` (Hidden Layer 2)
- `layer3` (Hidden Layer 3)
- `hidden_dropout` (Hidden Dropouts)
- `visible_dropout` (Visible Dropout)

Required packages: `deepnet`

Stochastic Gradient Boosting

```
method = 'gbm'
```

Type: Regression, Classification

Tuning parameters:

- `n.trees` (# Boosting Iterations)
- `interaction.depth` (Max Tree Depth)
- `shrinkage` (Shrinkage)
- `n.minobsinnode` (Min. Terminal Node Size)

Required packages: `gbm` , `plyr`

A model-specific variable importance metric is available.

Support Vector Machines with Boundrange String Kernel

```
method = 'svmBoundrangeString'
```

Type: Regression, Classification

Tuning parameters:

- `length` (length)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Class Weights

```
method = 'svmRadialWeights'
```

Type: Classification

Tuning parameters:

- `sigma` (Sigma)
- `C` (Cost)
- `Weight` (Weight)

Required packages: `kernlab`

Support Vector Machines with Exponential String Kernel

```
method = 'svmExpoString'
```

Type: Regression, Classification

Tuning parameters:

- `lambda` (lambda)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Linear Kernel

```
method = 'svmLinear'
```

Type: Regression, Classification

Tuning parameters:

- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Linear Kernel

```
method = 'svmLinear2'
```

Type: Regression, Classification

Tuning parameters:

- `cost` (Cost)

Required packages: `e1071`

Support Vector Machines with Polynomial Kernel

```
method = 'svmPoly'
```

Type: Regression, Classification

Tuning parameters:

- `degree` (Polynomial Degree)
- `scale` (Scale)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Radial Basis Function Kernel

```
method = 'svmRadial'
```

Type: Regression, Classification

Tuning parameters:

- `sigma` (Sigma)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Radial Basis Function Kernel

```
method = 'svmRadialCost'
```

Type: Regression, Classification

Tuning parameters:

- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Radial Basis Function Kernel

```
method = 'svmRadialSigma'
```

Type: Regression, Classification

Tuning parameters:

- `sigma` (Sigma)
- `C` (Cost)

Required packages: `kernlab`

Notes: This SVM model tunes over the cost parameter and the RBF kernel parameter sigma. In the latter case, using `tuneLength` will, at most, evaluate six values of the kernel parameter. This enables a broad search over the cost parameter and a relatively narrow search over `sigma`

Support Vector Machines with Spectrum String Kernel

```
method = 'svmSpectrumString'
```

Type: Regression, Classification

Tuning parameters:

- `length` (length)
- `C` (Cost)

Required packages: `kernlab`

Tree Augmented Naive Bayes Classifier

```
method = 'tan'
```

Type: Classification

Tuning parameters:

- `score` (Score Function)
- `smooth` (Smoothing Parameter)

Required packages: `bnclassify`

Tree Augmented Naive Bayes Classifier Structure Learner Wrapper

```
method = 'tanSearch'
```

Type: Classification

Tuning parameters:

- `k` (#Folds)
- `epsilon` (Minimum Absolute Improvement)
- `smooth` (Smoothing Parameter)
- `final_smooth` (Final Smoothing Parameter)
- `sp` (Super-Parent)

Required packages: `bnclassify`

Tree Augmented Naive Bayes Classifier with Attribute Weighting

```
method = 'awtan'
```

Type: Classification

Tuning parameters:

- `score` (Score Function)
- `smooth` (Smoothing Parameter)

Required packages: `bnclassify`

Tree Models from Genetic Algorithms

```
method = 'evtree'
```

Type: Regression, Classification

Tuning parameters:

- `alpha` (Complexity Parameter)

Required packages: `evtree`

Tree-Based Ensembles

```
method = 'nodeHarvest'
```

Type: Regression, Classification

Tuning parameters:

- `maxinter` (Maximum Interaction Depth)
- `mode` (Prediction Mode)

Required packages: `nodeHarvest`

Variational Bayesian Multinomial Probit Regression

```
method = 'vbmpRadial'
```

Type: Classification

Tuning parameters:

- `estimateTheta` (Theta Estimated)

Required packages: `vbmp`

Weighted Subspace Random Forest

```
method = 'wsrf'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `wsrf`

7.0.49 Text Mining

(back to contents)

Support Vector Machines with Boundrange String Kernel

```
method = 'svmBoundrangeString'
```

Type: Regression, Classification

Tuning parameters:

- `length` (length)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Exponential String Kernel

```
method = 'svmExpoString'
```

Type: Regression, Classification

Tuning parameters:

- `lambda` (lambda)
- `C` (Cost)

Required packages: `kernlab`

Support Vector Machines with Spectrum String Kernel

```
method = 'svmSpectrumString'
```

Type: Regression, Classification

Tuning parameters:

- `length` (length)
- `C` (Cost)

Required packages: `kernlab`

7.0.50 Tree-Based Model

(back to contents)

AdaBoost Classification Trees

```
method = 'adaboost'
```

Type: Classification

Tuning parameters:

- `nIter` (#Trees)
- `method` (Method)

Required packages: `fastAdaboost`

AdaBoost.M1

```
method = 'AdaBoost.M1'
```

Type: Classification

Tuning parameters:

- `mfinal` (#Trees)
- `maxdepth` (Max Tree Depth)
- `coeflearn` (Coefficient Type)

Required packages: `adabag`, `plyr`

A model-specific variable importance metric is available.

Bagged AdaBoost

```
method = 'AdaBag'
```

Type: Classification

Tuning parameters:

- `mfinal` (#Trees)
- `maxdepth` (Max Tree Depth)

Required packages: `adabag`, `plyr`

A model-specific variable importance metric is available.

Bagged CART

```
method = 'treebag'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `ipred`, `plyr`, `e1071`

A model-specific variable importance metric is available.

Bayesian Additive Regression Trees

```
method = 'bartMachine'
```

Type: Classification, Regression

Tuning parameters:

- `num_trees` (#Trees)
- `k` (Prior Boundary)
- `alpha` (Base Terminal Node Hyperparameter)
- `beta` (Power Terminal Node Hyperparameter)
- `nu` (Degrees of Freedom)

Required packages: `bartMachine`

A model-specific variable importance metric is available.

Boosted Classification Trees

```
method = 'ada'
```

Type: Classification

Tuning parameters:

- `iter` (#Trees)
- `maxdepth` (Max Tree Depth)
- `nu` (Learning Rate)

Required packages: `ada` , `plyr`

Boosted Logistic Regression

```
method = 'LogitBoost'
```

Type: Classification

Tuning parameters:

- `nIter` (# Boosting Iterations)

Required packages: `caTools`

Boosted Tree

```
method = 'blackboost'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (#Trees)
- `maxdepth` (Max Tree Depth)

Required packages: `party` , `mboost` , `plyr`

Boosted Tree

```
method = 'bstTree'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `maxdepth` (Max Tree Depth)
- `nu` (Shrinkage)

Required packages: `bst` , `plyr`

C4.5-like Trees

```
method = 'J48'
```

Type: Classification

Tuning parameters:

- `C` (Confidence Threshold)
- `M` (Minimum Instances Per Leaf)

Required packages: `RWeka`

C5.0

```
method = 'C5.0'
```

Type: Classification

Tuning parameters:

- `trials` (# Boosting Iterations)
- `model` (Model Type)
- `winnow` (Winnow)

Required packages: `C50` , `plyr`

A model-specific variable importance metric is available.

CART

```
method = 'rpart'
```

Type: Regression, Classification

Tuning parameters:

- `cp` (Complexity Parameter)

Required packages: `rpart`

A model-specific variable importance metric is available.

CART

```
method = 'rpart1SE'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `rpart`

A model-specific variable importance metric is available. Notes: This CART model replicates the same process used by the `rpart` function where the model complexity is determined using the one-standard error method. This procedure is replicated inside of the resampling done by `train` so that an external resampling estimate can be obtained.

CART

```
method = 'rpart2'
```

Type: Regression, Classification

Tuning parameters:

- `maxdepth` (Max Tree Depth)

Required packages: `rpart`

A model-specific variable importance metric is available.

CART or Ordinal Responses

```
method = 'rpartScore'
```

Type: Classification

Tuning parameters:

- `cp` (Complexity Parameter)
- `split` (Split Function)
- `prune` (Pruning Measure)

Required packages: `rpartScore` , `plyr`

A model-specific variable importance metric is available.

CHi-squared Automated Interaction Detection

```
method = 'chaid'
```

Type: Classification

Tuning parameters:

- `alpha2` (Merging Threshold)
- `alpha3` (Splitting former Merged Threshold)
- `alpha4` (Splitting former Merged Threshold)

Required packages: `CHAID`

Conditional Inference Tree

```
method = 'ctree'
```

Type: Classification, Regression

Tuning parameters:

- `mincriterion` (1 - P-Value Threshold)

Required packages: `party`

Conditional Inference Tree

```
method = 'ctree2'
```

Type: Regression, Classification

Tuning parameters:

- `maxdepth` (Max Tree Depth)
- `mincriterion` (1 - P-Value Threshold)

Required packages: `party`

Cost-Sensitive C5.0

```
method = 'C5.0Cost'
```

Type: Classification

Tuning parameters:

- `trials` (# Boosting Iterations)
- `model` (Model Type)
- `winnow` (Winnow)
- `cost` (Cost)

Required packages: `C50` , `plyr`

A model-specific variable importance metric is available.

Cost-Sensitive CART

```
method = 'rpartCost'
```

Type: Classification

Tuning parameters:

- `cp` (Complexity Parameter)
- `Cost` (Cost)

Required packages: `rpart` , `plyr`

DeepBoost

```
method = 'deepboost'
```

Type: Classification

Tuning parameters:

- `num_iter` (# Boosting Iterations)
- `tree_depth` (Tree Depth)
- `beta` (L1 Regularization)
- `lambda` (Tree Depth Regularization)
- `loss_type` (Loss)

Required packages: `deepboost`

eXtreme Gradient Boosting

```
method = 'xgbDART'
```

Type: Regression, Classification

Tuning parameters:

- `nrounds` (# Boosting Iterations)
- `max_depth` (Max Tree Depth)
- `eta` (Shrinkage)
- `gamma` (Minimum Loss Reduction)
- `subsample` (Subsample Percentage)
- `colsample_bytree` (Subsample Ratio of Columns)

- `rate_drop` (Fraction of Trees Dropped)
- `skip_drop` (Prob. of Skipping Drop-out)
- `min_child_weight` (Minimum Sum of Instance Weight)

Required packages: `xgboost` , `plyr`

A model-specific variable importance metric is available.

eXtreme Gradient Boosting

```
method = 'xgbTree'
```

Type: Regression, Classification

Tuning parameters:

- `nrounds` (# Boosting Iterations)
- `max_depth` (Max Tree Depth)
- `eta` (Shrinkage)
- `gamma` (Minimum Loss Reduction)
- `colsample_bytree` (Subsample Ratio of Columns)
- `min_child_weight` (Minimum Sum of Instance Weight)
- `subsample` (Subsample Percentage)

Required packages: `xgboost` , `plyr`

A model-specific variable importance metric is available.

Gradient Boosting Machines

```
method = 'gbm_h2o'
```

Type: Regression, Classification

Tuning parameters:

- `ntrees` (# Boosting Iterations)
- `max_depth` (Max Tree Depth)
- `min_rows` (Min. Terminal Node Size)
- `learn_rate` (Shrinkage)
- `col_sample_rate` (#Randomly Selected Predictors)

Required packages: `h2o`

A model-specific variable importance metric is available.

Model Tree

```
method = 'M5'
```

Type: Regression

Tuning parameters:

- `pruned` (Pruned)
- `smoothed` (Smoothed)
- `rules` (Rules)

Required packages: `RWeka`

Rotation Forest

```
method = 'rotationForest'
```

Type: Classification

Tuning parameters:

- `K` (#Variable Subsets)
- `L` (Ensemble Size)

Required packages: `rotationForest`

A model-specific variable importance metric is available.

Rotation Forest

```
method = 'rotationForestCp'
```

Type: Classification

Tuning parameters:

- `K` (#Variable Subsets)
- `L` (Ensemble Size)
- `cp` (Complexity Parameter)

Required packages: `rpart` , `plyr` , `rotationForest`

A model-specific variable importance metric is available.

Single C5.0 Tree

```
method = 'C5.0Tree'
```

Type: Classification

No tuning parameters for this model

Required packages: `C50`

A model-specific variable importance metric is available.

Stochastic Gradient Boosting

```
method = 'gbm'
```

Type: Regression, Classification

Tuning parameters:

- `n.trees` (# Boosting Iterations)
- `interaction.depth` (Max Tree Depth)
- `shrinkage` (Shrinkage)
- `n.minobsinnode` (Min. Terminal Node Size)

Required packages: `gbm` , `plyr`

A model-specific variable importance metric is available.

Tree Models from Genetic Algorithms

```
method = 'evtree'
```

Type: Regression, Classification

Tuning parameters:

- `alpha` (Complexity Parameter)

Required packages: `evtree`

Tree-Based Ensembles

```
method = 'nodeHarvest'
```

Type: Regression, Classification

Tuning parameters:

- `maxinter` (Maximum Interaction Depth)
- `mode` (Prediction Mode)

Required packages: `nodeHarvest`

7.0.51 Two Class Only

(back to contents)

AdaBoost Classification Trees

```
method = 'adaboost'
```

Type: Classification

Tuning parameters:

- `nIter` (#Trees)
- `method` (Method)

Required packages: `fastAdaboost`

Bagged Logic Regression

```
method = 'logicBag'
```

Type: Regression, Classification

Tuning parameters:

- `nleaves` (Maximum Number of Leaves)
- `ntrees` (Number of Trees)

Required packages: `logicFS`

Notes: Unlike other packages used by `train`, the `logicFS` package is fully loaded when this model is used.

Bayesian Additive Regression Trees

```
method = 'bartMachine'
```

Type: Classification, Regression

Tuning parameters:

- `num_trees` (#Trees)
- `k` (Prior Boundary)
- `alpha` (Base Terminal Node Hyperparameter)
- `beta` (Power Terminal Node Hyperparameter)
- `nu` (Degrees of Freedom)

Required packages: `bartMachine`

A model-specific variable importance metric is available.

Binary Discriminant Analysis

```
method = 'binda'
```

Type: Classification

Tuning parameters:

- `lambda.freqs` (Shrinkage Intensity)

Required packages: `binda`

Boosted Classification Trees

```
method = 'ada'
```

Type: Classification

Tuning parameters:

- `iter` (#Trees)
- `maxdepth` (Max Tree Depth)
- `nu` (Learning Rate)

Required packages: `ada` , `plyr`

Boosted Generalized Additive Model

```
method = 'gamboost'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `prune` (AIC Prune?)

Required packages: `mboost` , `plyr` , `import`

Notes: The `prune` option for this model enables the number of iterations to be determined by the optimal AIC value across all iterations. See the examples in `?mboost::mstop` . If pruning is not used, the ensemble makes predictions using the exact value of the `mstop` tuning parameter value.

Boosted Generalized Linear Model

```
method = 'glmboost'
```

Type: Regression, Classification

Tuning parameters:

- `mstop` (# Boosting Iterations)
- `prune` (AIC Prune?)

Required packages: `plyr` , `mboost`

A model-specific variable importance metric is available. Notes: The `prune` option for this model enables the number of iterations to be determined by the optimal AIC value across all iterations. See the

examples in `?mboost::mstop`. If pruning is not used, the ensemble makes predictions using the exact value of the `mstop` tuning parameter value.

CHi-squared Automated Interaction Detection

```
method = 'chaid'
```

Type: Classification

Tuning parameters:

- `alpha2` (Merging Threshold)
- `alpha3` (Splitting former Merged Threshold)
- `alpha4` (Splitting former Merged Threshold)

Required packages: `CHAID`

Cost-Sensitive C5.0

```
method = 'C5.0Cost'
```

Type: Classification

Tuning parameters:

- `trials` (# Boosting Iterations)
- `model` (Model Type)
- `winnow` (Winnow)
- `cost` (Cost)

Required packages: `C50` , `plyr`

A model-specific variable importance metric is available.

Cost-Sensitive CART

```
method = 'rpartCost'
```

Type: Classification

Tuning parameters:

- `cp` (Complexity Parameter)
- `Cost` (Cost)

Required packages: `rpart` , `plyr`

DeepBoost

```
method = 'deepboost'
```

Type: Classification

Tuning parameters:

- `num_iter` (# Boosting Iterations)
- `tree_depth` (Tree Depth)
- `beta` (L1 Regularization)
- `lambda` (Tree Depth Regularization)
- `loss_type` (Loss)

Required packages: `deepboost`

Distance Weighted Discrimination with Polynomial Kernel

```
method = 'dwdPoly'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)
- `degree` (Polynomial Degree)
- `scale` (Scale)

Required packages: `kerndwd`

Distance Weighted Discrimination with Radial Basis Function Kernel

```
method = 'dwdRadial'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)
- `sigma` (Sigma)

Required packages: `kernlab` , `kerndwd`

Generalized Linear Model

```
method = 'glm'
```

Type: Regression, Classification

No tuning parameters for this model

A model-specific variable importance metric is available.

Generalized Linear Model with Stepwise Feature Selection

```
method = 'glmStepAIC'
```

Type: Regression, Classification

No tuning parameters for this model

Required packages: `MASS`

glmnet

```
method = 'glmnet_h2o'
```

Type: Regression, Classification

Tuning parameters:

- `alpha` (Mixing Percentage)
- `lambda` (Regularization Parameter)

Required packages: `h2o`

A model-specific variable importance metric is available.

L2 Regularized Linear Support Vector Machines with Class Weights

```
method = 'svmLinearWeights2'
```

Type: Classification

Tuning parameters:

- `cost` (Cost)
- `Loss` (Loss Function)
- `weight` (Class Weight)

Required packages: `LiblineaR`

Linear Distance Weighted Discrimination

```
method = 'dwdLinear'
```

Type: Classification

Tuning parameters:

- `lambda` (Regularization Parameter)
- `qval` (q)

Required packages: `kerndwd`

Linear Support Vector Machines with Class Weights

```
method = 'svmLinearWeights'
```

Type: Classification

Tuning parameters:

- `cost` (Cost)
- `weight` (Class Weight)

Required packages: `e1071`

Logic Regression

```
method = 'logreg'
```

Type: Regression, Classification

Tuning parameters:

- `treesize` (Maximum Number of Leaves)
- `ntrees` (Number of Trees)

Required packages: `LogicReg`

Multilayer Perceptron Network with Dropout

```
method = 'mlpKerasDropoutCost'
```

Type: Classification

Tuning parameters:

- `size` (#Hidden Units)
- `dropout` (Dropout Rate)
- `batch_size` (Batch Size)
- `lr` (Learning Rate)
- `rho` (Rho)
- `decay` (Learning Rate Decay)
- `cost` (Cost)
- `activation` (Activation Function)

Required packages: `keras`

Notes: After `train` completes, the keras model object is serialized so that it can be used between R session. When predicting, the code will temporarily unsearalize the object. To make the predictions more efficient, the user might want to use `keras::unsearlize_model(object$finalModel$object)` in the current R session so that that operation is only done once. Also, this model cannot be run in parallel due to the nature of how tensorflow does the computations. Finally, the cost parameter weights the first class in the outcome vector. Unlike other packages used by `train`, the `dplyr` package is fully loaded when this model is used.

Multilayer Perceptron Network with Weight Decay

```
method = 'mlpKerasDecayCost'
```

Type: Classification

Tuning parameters:

- `size` (#Hidden Units)
- `lambda` (L2 Regularization)
- `batch_size` (Batch Size)
- `lr` (Learning Rate)
- `rho` (Rho)
- `decay` (Learning Rate Decay)
- `cost` (Cost)
- `activation` (Activation Function)

Required packages: `keras`

Notes: After `train` completes, the keras model object is serialized so that it can be used between R session. When predicting, the code will temporarily unsearalize the object. To make the predictions more efficient, the user might want to use `keras::unsearlize_model(object$finalModel$object)` in the current R session so that that operation is only done once. Also, this model cannot be run in parallel due to the nature of how tensorflow does the computations. Finally, the cost parameter weights the first class in the outcome vector. Unlike other packages used by `train`, the `dplyr` package is fully loaded when this model is used.

Oblique Random Forest

```
method = 'ORFlog'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Oblique Random Forest

```
method = 'ORFpls'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Oblique Random Forest

```
method = 'ORFridge'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Oblique Random Forest

```
method = 'ORFsvm'
```

Type: Classification

Tuning parameters:

- `mtry` (#Randomly Selected Predictors)

Required packages: `obliqueRF`

Notes: Unlike other packages used by `train`, the `obliqueRF` package is fully loaded when this model is used.

Partial Least Squares Generalized Linear Models

```
method = 'plsRglm'
```

Type: Classification, Regression

Tuning parameters:

- `nt` (#PLS Components)
- `alpha.pvals.expli` (p-Value threshold)

Required packages: `plsRglm`

Notes: Unlike other packages used by `train`, the `plsRglm` package is fully loaded when this model is used.

Rotation Forest

```
method = 'rotationForest'
```

Type: Classification

Tuning parameters:

- `K` (#Variable Subsets)
- `L` (Ensemble Size)

Required packages: `rotationForest`

A model-specific variable importance metric is available.

Rotation Forest

```
method = 'rotationForestCp'
```

Type: Classification

Tuning parameters:

- `K` (#Variable Subsets)
- `L` (Ensemble Size)
- `cp` (Complexity Parameter)

Required packages: `rpart`, `plyr`, `rotationForest`

A model-specific variable importance metric is available.

Support Vector Machines with Class Weights

```
method = 'svmRadialWeights'
```

Type: Classification

Tuning parameters:

- `sigma` (Sigma)
- `C` (Cost)
- `Weight` (Weight)

Required packages: `kernlab`

Tree-Based Ensembles

```
method = 'nodeHarvest'
```

Type: Regression, Classification

Tuning parameters:

- `maxinter` (Maximum Interaction Depth)
- `mode` (Prediction Mode)

Required packages: `nodeHarvest`