Create models with *parsnip* : : **CHEAT SHEET**

Basics

parsnip provides a tidy, unified interface to a range of models from other packages.It helps avoid having to remember how to properly call the modeling functions of those external packages.

A parsnip specification is made up of 3 main components:

1. The type of **model** to be used, such as Random Forest (`rand_forest()`) or linear regression (`linear_reg()`)
2. How will the model be used, or **mode**. The two most common are “regression” and “classification”.
3. The computational **engine**, or program that will actually execute the training. It could be an external R package, such as `ranger`, or even an engine outside of R, such as Stan or Apache Spark.

*Define type of **model***

```
rand_forest(mtry = 10, trees = 2000) |>
  set_engine("ranger", importance = "impurity") |>
  set_mode("regression")
```

*Select an **engine***

*Set the **mode***

set_engine(object, engine, ...**)** - Specifies which package or system will be used to fit the model, along with any arguments specific to that software.

set_args(object, ...**)** - Modifies the arguments of a model specification

set_mode(object, mode, …**)** - Changes the model's mode.

show_engines(x**)** - The possible engines for a model can depend on what packages are loaded. Some parsnip extension add engines to existing models.

show_engines(“linear_reg”)

Legends

ENGINE TAGS

Engine name — h2o 12 — *Mode support numbers*

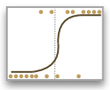
‘1 2’ means engine h2o supports classification and regression

MODE SUPPORT NUMBERS

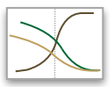
1 - Classification
2 - Regression
3 - Censored Regression
4 - Quantile Regression

Classification Only

*(mode = "classification")

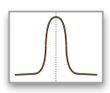


logistic_reg(engine = "glm", penalty, mixture**)** - Generalized linear model for binary outcomes. A linear combination of the predictors is used to model the log odds of an event. brulee 1 gee 1 glm 1 glmer 1 glmnet 1 h2o 1 keras 1 LiblineaR 1 spark 1 stan 1 stan_glmer 1



multinom_reg(engine = "nnet", penalty , mixture**)** - Uses linear predictors to predict multiclass data using the multinomial distribution. brulee 1 glmnet 1 h2o 1 keras 1 nnet 1 spark 1

naive_Bayes(smoothness , Laplace, engine = "klaR"**)** - Uses Bayes' theorem to compute the probability of each class, given the predictor values. h2o 1 klaR 1 naivebayes 1

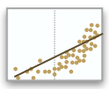


null_model(engine = "parsnip"**)** - Fit a single mean or largest class model. This is the user-facing function for the null_model() specification. parsnip 1

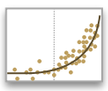
ordinal_reg(ordinal_link, odds_link, penalty, mixture, engine = "polr" **)** - Defines a generalized linear model that predicts an ordinal outcome. rpartScore 1 polr 1 vgam 1 vglm 1

Regression Only

*_reg(mode = "regression")



linear_reg(engine = "lm", penalty, mixture**)** - Defines a model that can predict numeric values from predictors using a linear function. brulee 2 gee 2 glm 2 glmer 2 glmnet 2 gls 2 h2o 2 keras 2 lm 2 lme 2 quantreg 2 spark 2 stan 2 stan_glmer 2



poisson_reg(penalty , mixture, engine = "glm"**)** - Defines a generalized linear model for count data that follow a Poisson distribution. gee 2 glm 2 glmer 2 glmnet 2 h2o 2 hurdle 2 stan 2 stan_glmer 2 zeroinfl 2

General Use

decision_tree(mode, engine = "rpart", cost_complexity, tree_depth, min_n**)** - A set of if/then statements that creates a tree-based structure. C5.0 1 partykit 123 rpart 123 spark 12

mars(mode, engine = "earth", num_terms, prod_degree, prune_method**)** - Uses artificial features for some predictors. These features resemble hinge functions and the result is a model that is a segmented regression in small dimensions. earth 12

mlp(mode, engine = "nnet", hidden_units, penalty, dropout, epochs, activation, learn_rate**)** - Defines a multilayer perceptron model (a.k.a. a single layer, feed-forward neural network). nnet 12 brulee 12 brulee_two_layer 12 keras 12 grnn 12

gen_additive_mod(mode, select_features, adjust_deg_free, engine = "mgcv"**)** - Uses smoothed functions of numeric predictors in a generalized linear model. mgcv 12

General Use (cont’)

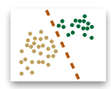
nearest_neighbor(mode, engine = "kknn", neighbors , weight_func, dist_power**)** - Uses the K most similar data points from the training set to predict new samples. knn 12

pls(mode, predictor_prop , num_comp, engine = "mixOmics"**)** - Uses latent variables to model the data. Similar to a supervised version of PCA. mixOmics 12

Discriminant

discrim_*(mode = "classification")

discrim_flexible(num_terms, prod_degree, prune_method, engine = "earth"**)** - Fits a discriminant analysis model that uses nonlinear features created using MARS. earth 1

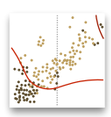


discrim_regularized(frac_common_cov, frac_identity, engine = "klaR"**)** - Estimates a multivariate distribution for the predictors separately for the data in each class. The model’s structure can be LDA, QDA, or a combination. Each probability class is computed using Bayes’s theorem, given the predictor values. klaR 1

Estimates a multivariate distribution for the predictors separately for the data in each class using a method described below. Each class’ probability is computed using Bayes’ theorem, given the predictor values.

discrim_*(mode = "classification", regularization_method, engine = "MASS")

discrim_linear(penalty**)** - Uses Gaussian with a common covariance matrix to perform the estimate. MASS 1 mda 1 sda 1 sparsediscrim 1



discrim_quad() - Uses Gaussian with separate covariance matrices to perform the estimate. MASS 1 sparsediscrim 1

Create models with *parsnip* : : **CHEAT SHEET**

Support Vector Machine

Classification: Maximizes the width of the margin between classes using a method described below. Regression: Optimizes a robust loss function only affected by very large model residuals and uses an additional method described below.

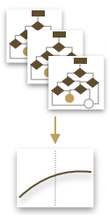
svm_*(mode, cost)

svm_linear(engine = "LiblineaR", margin**)** - Classification: A linear class boundary. Regression: Uses a linear fit.
kernlab 12 LiblineaR 12

svm_poly(engine = "kernlab", degree, scale_factor**)** - Classification: A polynomial class boundary. Regression: Uses polynomial functions of the predictors. kernlab 12

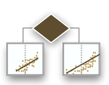
svm_rbf(engine = "kernlab", rbf_sigma**)** - Classification: A nonlinear class boundary. Regression: Uses nonlinear functions of the predictors. kernlab 12

Feature Rules



rule_fit(mode, mtry , trees, min_n, tree_depth, learn_rate, loss_reduction, sample_size, stop_iter, penalty, engine = "xrf"**)** - Derives simple feature rules from a tree ensemble and uses them as features in a regularized model.
xrf 12 h2o 1

C5_rules(mode = "classification", treesmin_n, engine = "C5.0"**)** - Derives feature rules from a tree for prediction. A single tree or boosted ensemble can be used. C5.0 1

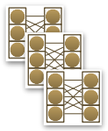


cubist_rules(mode = "regression", committees, neighbors, max_rules, engine = "Cubist"**)** - Derives simple feature rules from a tree ensemble and creates regression models within each rule. Cubist 2

Ensemble

"E Pluribus Unum"

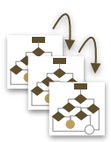
bag_mars(mode, num_terms, prod_degree, prune_method, engine = "earth"**)** - Ensemble of generalized linear models that use artificial features for some predictors. These features resemble hinge functions and the result is a model that is a segmented regression in small dimensions.
earth 12



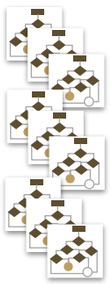
bag_mlp(mode, hidden_units, penalty, epochs, engine = "nnet"**)** - An ensemble of single layer, feed-forward neural networks. nnet 12

bag_tree(mode, cost_complexity = 0, tree_depth , min_n = 2, class_cost, engine = "rpart") - Ensemble of decision trees. C5.0 1 rpart 123

bart(mode, engine = "dbarts", trees, prior_terminal_node_coef, prior_terminal_node_expo, prior_outcome_range**)** - Tree ensemble model that uses Bayesian analysis to assemble the ensemble. bart 12



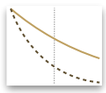
boost_tree(mode, engine = "xgboost", mtrytrees, min_n, tree_depth, learn_rate, loss_reduction, sample_size, stop_iter**)** - Creates a series of decision trees forming an ensemble. Each tree depends on the results of previous trees. All trees in the ensemble are combined to produce a final prediction.
C5.0 1 catboost 12 h2o 12 lightgbm 12 mboost 3 spark 12 xgboost 124



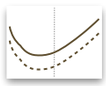
rand_forest(mode, engine = "ranger", mtry , trees, min_n**)** - Creates a large number of decision trees, each independent of the others. The final prediction uses all predictions from the individual trees and combines them.
aorsf 123 grf 124 h2o 12 partykit 123 randomForest 12 ranger 12 spark 12

Survival

*(mode = "censored regression")



proportional_hazards(engine = "survival", penalty , mixture**)** - Defines a model for the hazard function as a multiplicative function of covariates times a baseline hazard. glmnet 3 survival 3



survival_reg(engine = "survival", dist**)** - Defines a parametric survival model.
flexsurv 3 flexsurvspline 3 survival 3

Operations

library(tidymodels)
lm_spec <- linear_reg() |> set_engine("lm")

METHODS

fit(object, ...**)** - Estimates parameters for a given model from a set of data.
lm_fit <- fit(lm_spec, mpg ~ ., data = mtcars)

predict(object, …**)**
predict(lm_fit, mtcars)

autoplot(object, ...**)** - Uses `ggplot2` to draw a particular plot for an object of a particular class

update(object, ...**)** - Updates and (by default) re-fit a model. It does this by extracting the call stored in the object, updating the call and evaluating that call.

TIDIERS

augment(x, ...**)** - Augment data with model results
augment(lm_fit, mtcars)

glance(x, ...**)** - Construct a single row summary "glance" of a model fit
glance(lm_fit)

tidy(x, ...**)** - Turn an object into a tidy tibble
tidy(lm_fit)

GENERAL

repair_call(x, data**)** - When the user passes a formula to fit() and the underlying model function uses a formula, the call object produced by fit() may not be usable by other functions.

Operations (cont')

GENERAL (cont')

control_parsnip(verbosity = 1L, catch = FALSE**)** - Pass options to the fit.model_spec() function to control its output and computations.
control_parsnip(verbosity = 2)

show_engines(x**)** - The possible engines for a model can depend on what packages are loaded. Some parsnip extension add engines to existing models.
show_engines("linear_reg")

translate(x, ...**)** - Translates a model specification into a code object that is specific to a particular engine (e.g. R package). It translates generic parameters to their counterparts.
translate(lm_spec)

multi_predict(object, ...**)** - For some models, predictions can be made on sub-models in the model object.

EXTRACT

extract_spec_parsnip(x, ...**)** - Returns a parsnip model specification.
extract_spec_parsnip(lm_fit)

extract_fit_engine(x, ...**)** - Returns the engine specific fit embedded within a parsnip model fit. For example, when using linear_reg() with the "lm" engine, this returns the underlying lm object. extract_parameter_set_dials(x, ...) - Returns a set of dials parameter objects.
extract_fit_engine(lm_fit)

extract_parameter_dials(x, parameter, ...**)** - Returns a single dials parameter object.

extract_fit_time(x, summarize = TRUE, ...**)** - returns a tibble with fit times. The fit times correspond to the time for the parsnip engine to fit and do not include other portions of the elapsed time in fit.model_spec().