



Recipe: `recipe(.formula, .data)`

#Helps to create a recipe.

`prep()` *# Estimate the required preprocessing transformations.*

`bake(new_data= NULL)` *#Applies a recipe to a data set.*

Standardization and Normalization:

`step_log()`

#Normalizes predictors by applying log transformation.

`step_normalize()`

#Normalizes predictors by forcing mean = 0, sd = 1.

`step_center()`

#Normalizes numeric columns by forcing mean = 0.

`step_scale()`

#Normalizes predictors by forcing sd = 1.

Handling Missing Data:

`step_impute_median()`

#Imputes missing numeric values with median

`step_impute_mean()`

#Imputes missing numeric values with mean.

`step_impute_mode()`

#Imputes missing numeric values with mode.

Encoding Categorical Variables:

`step_dummy()`

#Creates dummy variables for categorical predictors.

`step_other()`

#Collapses infrequent factors into an "other" category.

Feature Engineering:

`step_mutate()`

#Creates a new feature from an existing one.

`step_interact()`

#Creates interaction terms between features.

Tidymodels Cheatsheet

Data Filtering and Columns Selection:

`step_rm(!c())` *#Selects specific variables to retain in the dataset.*

`step_slice()` *#Filters rows using their position.*

`step_filter()` *#Filters rows using logical condition.*

`all_nominal_predictors()`

#Selects all categorical predictor variables in the dataset.

`all_numeric_predictors()`

Selects all numeric predictor variables in the dataset.

Linear Regression:

`linear_reg()`

#Specifies a linear reg model

`set_engine("lm")`

#Set computational engine to fit the data.

`set_mode()`

#To define the purpose of model (regression/classification)



Ridge & Lasso Regression:

`linear_reg(penalty=0.1, mixture=)`

`set_engine("glmnet")`

#To set the computational engine to fit the data.

Decision Tree:

When datasets have non-linear relationships.

`decision_tree()` *# Specifies a decision tree model .*

`set_engine("rpart")`

#To set the computational engine to fit the data.

Logistic Regression:

Predict the probability of an outcome that can be one of two possible states.

`logistic_reg()` *#Specifies a logistic reg model .*

Gradient Boosted Models:

Automatically handles missing values and does not require scaling of data.

`boost_tree()` *#Specifies gradient boosted model.*

Random Forest:

Delivers high accuracy & Robust against overfitting.

`rand_forest()` *#Specifies Random Forest Model.*

Correlation Matrix:

`cor(use="complete.obs")`

#helps to create a correlation matrix.

`corrplot()` *#helps to visualize the correlation matrix.*

Workflows:

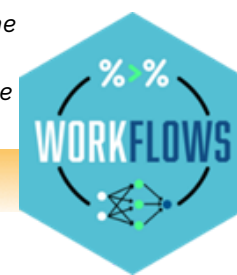
`workflow()` *#Helps to combine recipe and model together.*

`add_recipe()` *#add your recipe to the workflow.*

`add_model()` *#add your model to the workflow.*

`fit(.data)`

#This helps to fit/run the workflow to the data.



Data Splitting:

`initial_split(.data, prop=)` *#define your split and data proportion.*

`training()` *#creates a train dataset.*

`testing()` *#creates a test dataset.*

Making Predictions:

`predict (workflow, .data)` *#make prediction on test data using fitted workflow.*

`bind_rows()` *#put all predictions metrics in one dataset.*

`bind_cols()` *#add predictions to the original data.*



Model Evaluation & Metrics:

`metrics(truth =, estimate =)`

#used to extract rmse, mae and rsq.

truth *#indicates actual dependent variable values.*

estimate *#indicates predicted values of dependent variables.*