Cheatography

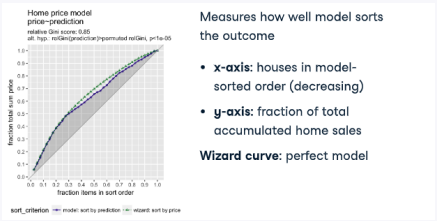
Supervised Learning in R: Regression Cheat Sheet

by Ivan Patel (patelivan) via cheatography.com/135316/cs/28145/

What is a Regression?

From a machine learning perspective, the term regression generally encompasses the prediction of continuous values. Statistically, these predictions are the expected value, or the average value one would observe for the given input values.
In R, use the lm() function, shown above, to code an OLS model. You can simply print the model to observe the coefficients' sign and magnitude.
After estimating the coefficients, you can always make predictions on your training data to assess the actual values v. predicted values. Use ggplot2() and a reference line of slope 1 to see if the points are close to the line or not.
You can also plot a gain curve to measure how well your model sorts the outcome. Useful when sorting instances is more important than predicting the exact outcome.

An example of the Gain Curve.



The diagonal line represents the gain curve if the outcome was sorted randomly. A perfect model would trace out the green curve. And our model chases out the blue curve. Per the above graph, the model correctly identified the top 30% highest home values and sorted them by price correctly.

We can also see that the top 30% of highest priced homes are 50% of total home sales ($).

Basic Regression Functions to get started

lm(formula = dependent variable ~ independent variable1 + independent variable2 + ..., data).

summary() - To get more details about the model results.

broom::glance() - See the model details in a tidier form.

predict(model, newdata)

ggplot(dframe, aes(x = pred, y = outcome)) + geom_point() + geom_abline()

GainCurvePlot(data, "prediction", "price", "model")

Evaluating a regression model

RMSE = sqrt(mean(res^2)). It is the typical prediction error of your model on the data. Want to minimize this.

R^2 = 1-(RSS/TSS). A measure of how well the model fits or explains the data.

One heuristic is to compare the RMSE to the standard deviation of the outcome. With a good model, the RMSE should be smaller.

RSS = sum(res^2).
TSS = sum(outcome value - mean of outcome)^2

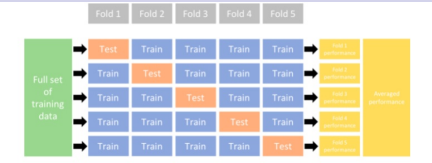
Important things to remember about Regression

Pros of Regression
Easy to fit and apply
Concise
Less prone to overfit
Interpretable - One can easily observe the signs and magnitude of each coefficient.
Things to look out for
Cannot express complex, non-linear, non-additive relationships. Non-linear relationships can be made linear by applying transformations.
Collinearity is when input variables are partially correlated. When independent variables are correlated, signs of the variables may not be what you expect. Moreover, it can be difficult to separate out the individual effects of collinear variables on the response.. Try manually removing offending variables, dimension reduction, or regularization.
Variance among residuals should be constant. Plot the residuals on the y-axis and predicted values on the x-axis. The errors should be evenly distributed between positive and negative, and have about the same magnitude above and below.
Autocorrelation - Errors are independent and uncorrelated. Plot the row_number() on the x-axis and residuals on the y-axis.

By **Ivan Patel** (patelivan)
cheatography.com/patelivan/

Published 28th May, 2021.
Last updated 28th May, 2021.

Training a regression model



It is crucial to split your data into training and test sets to evaluate how your model does on unseen data.

k-fold cv

k-fold cross-validation (aka k-fold CV) is a resampling method that randomly divides the training data into k groups (aka folds) of approximately equal size. The model is fit on k −1 folds and then the remaining fold is used to compute model performance. This procedure is repeated k times; each time, a different fold is treated as the validation set. Thus, the k-fold CV estimate is computed by averaging the k test errors, providing us with an approximation of the error we might expect on unseen data..

Training and testing using k-fold cv.

```
# Returns a list of nSplits.
# Each sub-list will contain two vectors; train and app.
splitPlan <- vtreat ::k Way Cro ssV ali dat ion (nRows, nSplits, NULL, NULL)
# Initialize a column of the approp riate length
dframe $pr ed.cv <- 0
# k is the number of folds
# splitPlan is the cross validation plan
for(i in 1:k) {
   # Get the ith split
    split <- splitP lan [[i]]
 # Build a model on the training data from this split (lm, in this case)
    model <- lm(fmla, data = dframe [sp lit $tr ain,])
   # make predic tions on the applic ation data from this split
    dfr ame $pr ed.c v[ spl it$app] <- predic t(m odel, newdata = dframe [sp lit $app,])
}
```

nRows - number of rows in training data
nSplits - number of folds.