Caret & caretEnsemble

“Classification and Regression Training”

The rest of this presentation assumes you have some familiarity with machine learning in general, but are either new to R or hoping to get faster at building models. Please feel free to ask any questions, especially if my assumption is wrong.

Very simplified steps for most supervised machine learning workflows:

1. Get data. Review quality and general shape of each variable, decide which to keep and transform.
2. Split data into a train/test set. Typical: 80% train and 20% test or 70/30
3. Fit a model to the training set. Possibly more than one. (this means learn the relationships of the inputs to the output(s))
4. Evaluate the model on the test set. If it’s pretty good, keep it. If not, go back to step 3

There are rules of thumb for which models work better for which data sets (large, categorical, sparse, etc.) but many ML practitioners try the same 5-7 basic ones all the time to start with because each data set could be an exception to these fuzzy rules. What if you want to streamline this model selection step?

Why use caret: easy, standardized input so you can quickly try many common classification or regression models while you work on something else.

* Some models can take days to run. Just write a few lines of code and walk away!
* Each package looks for input in a slightly different way. With caret you don’t have to care, just make a loop with the same inputs!
* Handles splitting into test/train, imputation, cross-validation, feature selection & importance, and parameter tuning in the same standardized way.
* Caution: once you choose a model, you do want to make sure you understand it and the parameters and assumptions. Also be careful which variables you use, e.g. leakage.
* Also includes a few options for resampling imbalanced classes.
* Will usually load packages for you if you don’t have them (some exceptions)
* Time series, unsupervised learning, and survival analysis out of scope, but exploration steps can help you understand the data better to inform those steps.

Background: Created by Max Kuhn at RStudio 10 years ago.

The documentation is excellent and searchable: <http://topepo.github.io/caret/index.html>

## The basic structure

train.model <- train(Survived ~ ., data = titanic.train, method = "rf", metric = “Accuracy”, tuneGrid = tune.grid, trControl = train.control)

* **Data** -This one is pretty self-explanatory—it’s the object from which you’re getting your training data.
* **Method** - This is the specific machine learning algorithm that you want to deploy. The one you’re using for the moment, xgbTree, is a form of extreme gradient boosted decision trees.
* **Metric** – how do we base our choice of tuning parameters to give the best final model? Do you want the highest accuracy, ROC, Kappa, RMSE, etc on the held-out samples?
* **tuneGrid** - This is a data frame of parameters that you can pass to your model training and have the model train and evaluate for those parameters and then move on to the next set of parameters. This is model dependent, but you’ll see how you can better understand how to use it.
* **trContro**l -The train control options let you specify how you want to do cross-validation techniques for training.

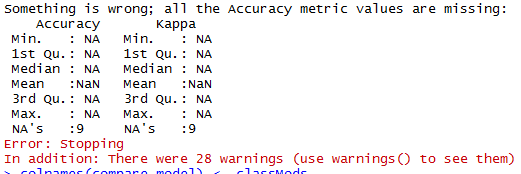
## Cautions:

### Zero-Variance Predictors cause many models to fail!

Identify those that:

1. <20% unique values in the training set
2. AND Ratio of the most frequent value of a predictor to the second most frequent value is > 20

Error when you end up with these zeros:



Some models (linear, neural networks) don’t do well with multicollinearity. Decision trees are resistant, but can be harder to interpret.

Example: Chicago Traffic Crashes data at https://github.com/npolishchuk/ChicagoTrafficCrashes

* Excellent source for practice data, usually fairly clean and variables explained clearly.
* Bonus, automated dashboards for a quick look at each variable.

<https://data.cityofchicago.org/Transportation/Traffic-Crashes-Crashes-Dashboard/8tdq-a5dp>

* Follow along with example here:

<https://github.com/npolishchuk/ChicagoTrafficCrashes/>

Feel free to clone and use as you wish!

Can you find predictors that are highly correlated? *(HINT: use PCA to reduce or findCorrelation to identify and remove)*

## Extend this example – Download the R file

* Add more data from Chicago Data Portal or other sites. What other features could help make this more accurate?
* Consider seasonality/weekly/hourly cycles
* Tune parameters more
* Try more types of models
* Multiple classes: type of injury (e.g. fatal, non-apparent)
* Consider 2 vehicle, multivehicle, or pedestrian incidents

# Additional info:

Easy workshop with sample data and github repo: https://www.gokhan.io/post/caret-workshop/

From the ground up crash tutorial: <https://www.r-bloggers.com/a-quick-introduction-to-machine-learning-in-r-with-caret/>

*Introduction to Machine Learning in R*

http://rpubs.com/Mentors\_Ubiqum/caret-for-loops