CWRU DSCI353-453: Week06b p2 Caret Package Review

Profs: R. H. French, L. S. Bruckman, P. Leu, K. Davis, S. Cirlos

TAs: W. Oltjen, K. Hernandez, M. Li, M. Li, D. Colvin

23 February, 2023

Contents

6.1.3.1 Caret Package Overview, The Frontend to ML in R

- NOTE: Tidymodels package is the tidyverse version of Caret Package

Caret can work with tidyverse also

```
library(tidymodels)
```

```
## Error in library(tidymodels): there is no package called 'tidymodels'
```

```
?tidymodels
```

```
## No documentation for 'tidymodels' in specified packages and libraries:
## you could try '??tidymodels'
```

When you are first getting started with machine learning,

- the situation is very similar:
  - you can learn to use some of the tools,
  - without knowing the deep mathematics that makes those tools work.

Having said that, the above analogy is somewhat imperfect.

At some point, as you progress to more advanced topics,

- it will be very beneficial to know the underlying mathematics.

Ok, so you don't need to know that much math to get stared,

- but you're not entirely off the hook.
- As I noted above, you still need to know how to use the tools properly.

In some sense, this is one of the challenges of using machine learning tools in R:

- many of them are difficult to use.

R has many packages for implementing various machine learning methods,

- but unfortunately many of these tools were designed separately,
  - and they are not always consistent in how they work.
- The syntax for some of the machine learning tools is very awkward,
  - and syntax from one tool to the next is not always the same.
- If you don't know where to start, machine learning in R can become very confusing.

This is why I recommend using the caret package to do machine learning in R.

6.1.3.1.1 A quick introduction to caret

- For starters, let's discuss what caret is.

The caret package is a set of tools for building machine learning models in R.

The name "caret" stands for **Classification And REgression Training**.

As the name implies, the caret package gives you a toolkit

- for building classification models and regression models.

Moreover, caret provides you with essential tools for:

- Data preparation, including:
  - imputation,
  - centering/scaling data,
  - removing correlated predictors,
  - reducing skewness
- Data splitting
- Model evaluation
- Variable selection

6.1.3.1.2 Caret simplifies machine learning in R

- While caret has broad functionality,
- the real reason to use caret
- is that it's simple and easy to use.

As noted above, one of the major problems with machine learning in R

- is that most of R's different machine learning tools have different interfaces.
- They almost all "work" a little differently from one another:
  - the syntax is slightly different from one modeling tool to the next;
  - tools for different parts of the machine learning workflow don't always "work well" together;
  - tools for fine tuning models or performing critical functions may be awkward or difficult to work with.
- Said succinctly, R has many machine learning tools,
  - but they can be extremely clumsy to work with.

Caret solves this problem.

To simplify the process,

- caret provides tools for almost every part of the model building process,
- and moreover, provides a common interface to these different machine learning methods.

For example, caret provides a simple, common interface

- to almost every machine learning algorithm in R.
- When using caret, different learning methods

    - like linear regression,
    - neural networks, and
    - support vector machines,
- all share a common syntax
    - (the syntax is basically identical, except for a few minor changes).

Moreover, additional parts of the machine learning workflow

- like cross validation and parameter tuning
- are built directly into this common interface.

To say that more simply,

- caret provides you with an easy-to-use toolkit
    - for building many different model types and
    - executing critical parts of the ML workflow.
- This simple interface enables rapid, iterative modeling.

In turn, this iterative workflow will allow you to develop good models

- faster,
- with less effort, and
- with less frustration.

6.1.3.1.3 Caret's syntax

- Now that you've been introduced to caret,
    - let's return to the example above (of *mpg* vs. *wt*)
    - and see how caret works.

Again, imagine you want to learn the relationship between *mpg* and *wt*.

As noted above, in mathematical terms, this means

- identifying a function, $f(x)$, that describes the relationship between wt and mpg.

Here in this example,

- we're going to make an additional assumption
    - that will simplify the process somewhat:
- we're going to assume that the relationship is linear;
- we'll assume that that it can be described by a straight line
    - of the form $f(x) = \beta_0 + \beta_1 x$.

In terms of our modeling effort,

- this means that we'll be using linear regression
- to build our machine learning model.

Without going into the details of linear regression

- let's look at how we implement linear regression with caret.

6.1.3.1.4 The `train()` function

- The core of caret's functionality is the `train()` function.

`train()` is the function that we use to "train" the model.

- That is, train is the function that will "learn"
- the relationship between mpg and wt.

Let's take a look at this syntactically.

Here is the syntax for a linear regression model, regressing mpg on wt.

```
#~~~~~~~~~~~~~~~~~~~~~~~~~
# Build model using train()
#~~~~~~~~~~~~~~~~~~~~~~~~~
require(caret)
```

```
## Loading required package: caret
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
model.mtcars_lm <- train(mpg ~ wt, data = mtcars, method = "lm" )
```

That's it.

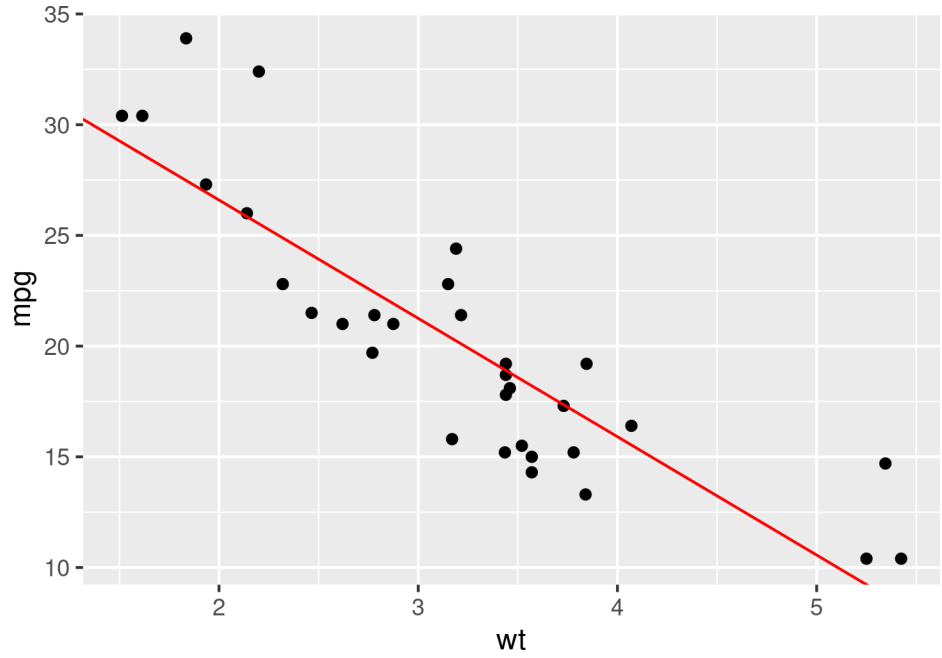
The syntax for building a linear regression is extremely simple with caret.

Now that we have a simple model,

- let's quickly extract the regression coefficients
- and plot the model
  - i.e., plot the linear function
  - that describes the relationship between mpg and wt.

```
#~~~~~~~~~~~~~~~~~~~~~~~~~
# Retrieve coefficients for
#  - slope
#  - intercept
#~~~~~~~~~~~~~~~~~~~~~~~~~
coef.icept <- coef(model.mtcars_lm$finalModel)[1]
coef.slope <- coef(model.mtcars_lm$finalModel)[2]

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Plot scatterplot and regression line
#  using ggplot()
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
ggplot(data = mtcars, aes(x = wt, y = mpg)) + geom_point() +
  geom_abline(slope = coef.slope, intercept = coef.icept, color = "red")
```



Now, let's look more closely at the syntax and how it works.

When training a model using *train*(), you only need to tell it a few things:

- The dataset you're working with
- The target variable you're trying to predict
  - e.g., the *mpg* variable
- The input variable
  - e.g., the *wt* variable
- The machine learning method you want to use
  - in this case "linear regression"

6.1.3.1.5 Formula notation

- In caret's syntax,
  - you identify the target variable and
  - input variables using the "formula notation."

The basic syntax for formula notation is $y \sim x$,

- where y is your target variable or response,
- and x is your predictor.

Effectively, $y \sim x$ tells caret

- "I want to predict y on the basis of a single input, x."

Now, with this knowledge about caret's formula syntax, let's reexamine the above code.

- Because we want to predict mpg on the basis of *wt*,
  - we use the formula *mpg wt*.
- Again, this line of code is the "formula"
  - that tells *train*() our target response variable
  - and our input predictor variable.
- If we translate this line of code into English,
  - we're effectively telling *train*(),
  - "build a model that predicts *mpg* (miles per gallon)

- on the basis of wt (car weight)."

6.1.3.1.6 The data = parameter

- The $train()$ function also has a data = parameter.

This basically tells the $train()$ function

- what dataset we're using to build the model.

Said differently,

- if we're using the formula $mpg \sim wt$
  - to indicate the target and predictor variables,
- then we're using the $data = parameter$ -to tell caret where to find those variables.

So basically, $data = mtcars$

- tells the caret function that the data and
  - the relevant variables
- can be found in the mtcars dataset.

6.1.3.1.7 The method = parameter

- Finally, we see the $method = parameter$.
  - This parameter indicates what machine learning method
    - we want to use to predict y.
  - In this case, we're building a linear regression model,
  - so we are using the argument "lm".

Keep in mind, however, we could select a different learning method.

- Although it's beyond the current scope
  - to discuss all of the possible learning methods that we could use here,
  - there are many different methods we could use.
- For example, if we wanted to use the k-nearest neighbor technique,
  - we could use the "knn" argument instead.
- If we did that, $train()$ would still predict mpg on the basis of wt,
  - but would use a different statistical technique to make that prediction.
- This would yield a different model;
  - a model that makes different predictions.

As you learn more about machine learning, and

- want to try out more advanced machine learning techniques,
  - this is how you can implement them.
- You simply change the learning method
  - by changing the argument of the method = parameter.

This is a good place to reiterate one of caret's primary advantages:

- Switching between model types is extremely easy
  - when we use caret's $train()$ function.
- Again, if you want to use linear regression to model your data,
  - you just type in "lm" for the argument to $method =$;
- if you want to change the learning method to k-nearest neighbor,
  - you just replace "lm" with "knn".

Caret's syntax allows you to very easily change the learning method.

- In turn, this allows you to "try out" and evaluate

  - many different learning methods rapidly and iteratively.
- You can just re-run your code
  - with different values for the method parameter,
- and compare the results for each method.

6.1.3.1.8 Next steps

- Now that you have a high-level understanding of caret,
  - you're ready to dive deeper into machine learning in R.

Keep in mind though, if you're new to machine learning,

- there's still lots more to learn.

Machine learning is intricate and fascinatingly complex.

Moreover, caret has a variety of additional tools for model building.