CWRU DSCI351-351M-451: Exploratory Data Science

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

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

01 December, 2022

Contents

15.2.2.1 Class Readings, Assignments, Syllabus Topics

15.2.2.1.1 Course Evaluations Are Open Now

- lets get to 90% response rate
- We want statistically significant results!
  - I look for suggestions on how to improve the course
- https://webapps.case.edu/courseevals/

15.2.2.1.2 Reading, Lab Exercises, SemProjects

- Readings:
  - For today: French & Bruckman 2020
  - For next class: Khalilnejad 2020
- Laboratory Exercises:
  - LE7 : Due Thursday Dec. 8nd
  - LE7 :
- Office Hours: (Class Canvas Calendar for Zoom Link)
  - Wednesday @ 4:00 PM to 5:00 PM, Will Oltjen
  - Saturday @ 3:00 PM to 4:00 PM, Kristen Hernandez
  - **Office Hours are on Zoom, and recorded**

- Semester Projects
  - DSCI 451 Students Biweekly Update 6 Due Friday November 18th
  - DSCI 451 Students
    - Next
  - All DSCI 351/351M/451 Students:
    - **Peer Grading of Report Out #3 Due today (or by Saturday)**
  - Exams
    - Final: Monday December 19, 2022, 12:00PM - 3:00PM, Nord 356 or remote

15.2.2.2 A quick introduction to machine learning in R with caret

- If you've been using R for a while,
  - and you've been working with
    - basic data visualization and data exploration techniques,
  - the next logical step is to start learning some machine learning.

To help you begin learning about machine learning in R,

- lets introduce you to an R package: the `caret` package.

We'll build a very simple machine learning model

- as a way to learn some of `caret`'s basic syntax and functionality.

But before diving into `caret`,

- let's quickly discuss what machine learning is
  - and why we use it.

15.2.2.2.1 What is machine learning?

- Machine learning is
  - the study of data-driven, computational methods
  - for making inferences and predictions.

Without going into extreme depth here,

- let's unpack that by looking at an example.

15.2.2.2.2 A simple example

- Imagine that you want to understand
  - the relationship between car weight and car fuel efficiency
    - (i.e., miles per gallon);
  - how is fuel efficiency effected by a car's weight?

To answer this question,

- you could obtain a dataset with several different car models, and
- attempt to identify a relationship between
  - weight (which we'll call `wt`) and
  - miles per gallon (which we'll call `mpg`).

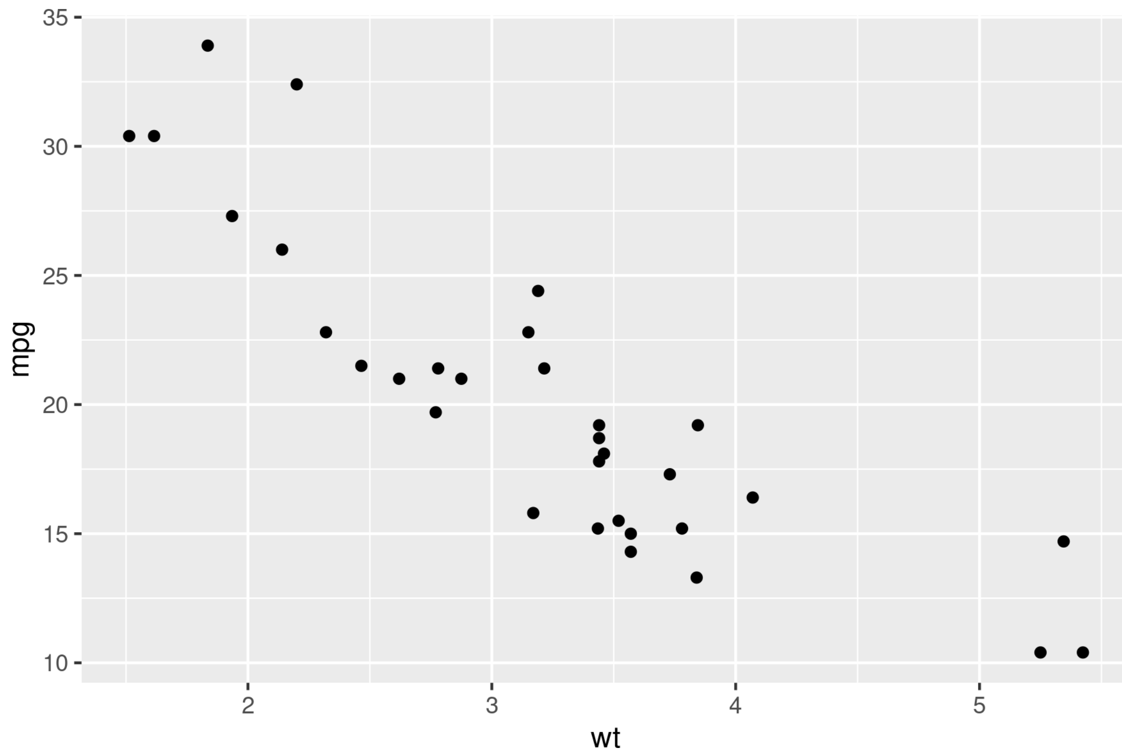
A good starting point would be some EDA

- simply plot the data,
- so first, we'll create a scatterplot using R's ggplot:

```
require(ggplot2)
```

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

```r
ggplot(data = mtcars, aes(x = wt, y = mpg)) +
  geom_point()
```



Just examining the data visually,

- it's pretty clear that there's some relationship.

But if we want to make more precise claims

- about the relationship between wt and mpg,
- we need to specify this relationship mathematically.

So, as we press forward in our analysis,

- we'll make the assumption that
  - this relationship can be described mathematically;
- more precisely, we'll assume that
  - this relationship can be described by some mathematical function, $f(x)$.
- In the case of the above example, we'll be making the assumption
  - that "miles per gallon" can be described
  - as a function of "car weight".

Assuming this type of mathematical relationship,

- machine learning provides a set of methods
  - for identifying that relationship.

Said differently, machine learning provides a set of computational methods

- that accept data observations as inputs,
- and subsequently estimate that mathematical function, $f(x)$;
- machine learning methods learn the relationship
  - by being trained with an input dataset.

Ultimately, once we have this mathematical function (a model),

- we can use that model to make predictions and inferences.

15.2.2.2.3 How much math you really need to know

- What we just discussed
    - about "estimating functions" and "mathematical relationships"
        - might cause you to ask a question:
    - "how much math do I need to know to do machine learning?"

Ok, here is some good news:

- to implement basic machine learning techniques,
- you don't need to know much math.

To be clear, there is quite a bit of math involved in machine learning,

- but most of that math is taken care of for you.

For the most part, R libraries and functions

- perform the mathematical calculations for you.

You just need to know

- which functions to use, and
- when to use them.

Here's an analogy: if you were a carpenter,

- you wouldn't need to build your own power tools.
    - your own drill and power saw.
- Therefore, you wouldn't need to understand
    - the mathematics, physics, and electrical engineering principles
    - that would be required to construct those tools from scratch.
- You could just go and buy them "off the shelf."
- To be clear, you'd still need to learn how to use those tools,
    - but you wouldn't need a deep understanding
    - of math and electrical engineering to operate them.

When you're 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.
- 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 the caret package is useful for machine learning in R.

15.2.2.2.4 A quick introduction to the `caret` package

- 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, for training and testing
- Model evaluation
- Variable selection

15.2.2.2.5 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.

15.2.2.2.6 `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`.

15.2.2.2.7 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: lattice
```

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

That's it. The syntax for building a linear regression

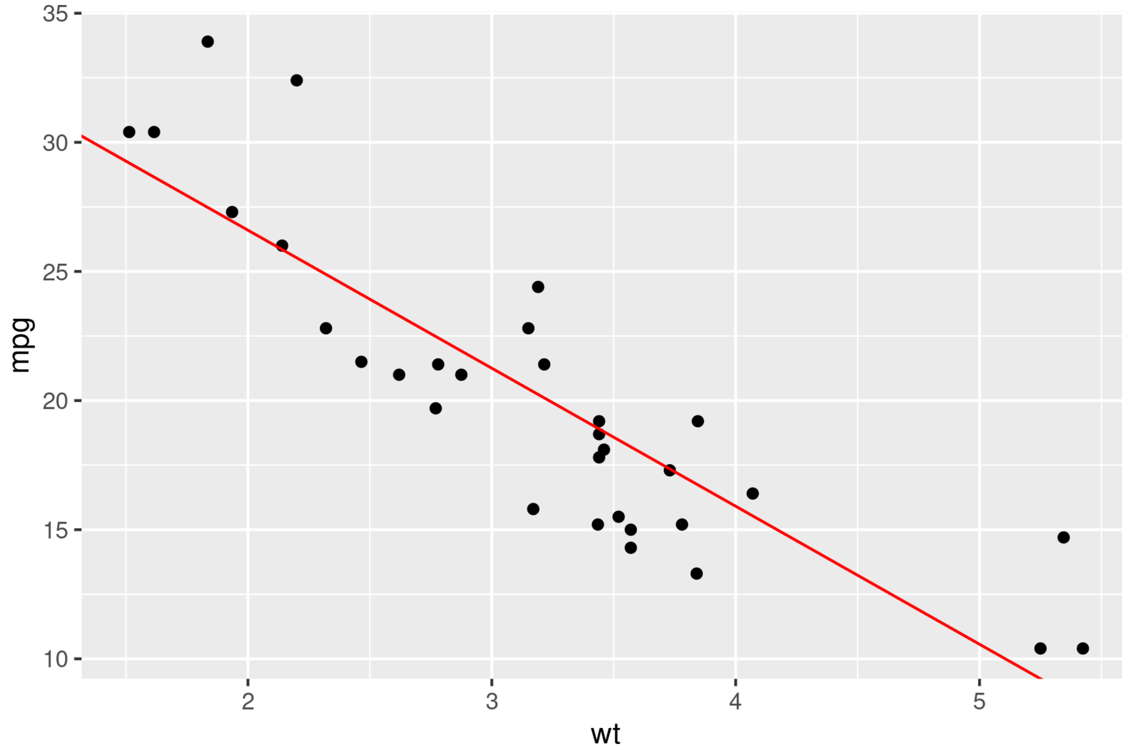
- is extremely simple with `caret`.

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")

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

Effectively, `y ~ 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)."

15.2.2.2.9 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 ~ 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.

15.2.2.2.10 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.

15.2.2.2.11 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.

We've just scratched the surface here.

15.2.2.3 The Perceptron (Neural Networks)

- Lets start looking into the broad topic of Neural Networks for Machine Learning

Artificial Neural Networks (ANN)

- are the supervised learning techniques
- whose logic is similar to biological neural systems.

A simple ANN technique is the single-layer perceptron and

- it is a classification technique
- estimating a binary attribute
- whose value can be 0 or 1.

The single layer perceptron models

- are as shown in the following figure:

The perceptron works like a neuron

- in the sense that it sums the impact
- of all the inputs and outputs to 1
- if the sum is above a defined threshold.

The model is based on the following parameters:

- A weight for each feature, defining its impact
- A threshold above which the estimated output is 1

Starting from the features, the model estimates the attribute through these steps

- Compute the output through a linear regression:
  - multiply each feature by its weight and sum all of them
- Estimate the attribute
  - to 1 if the output is above the threshold
  - and to 0 otherwise

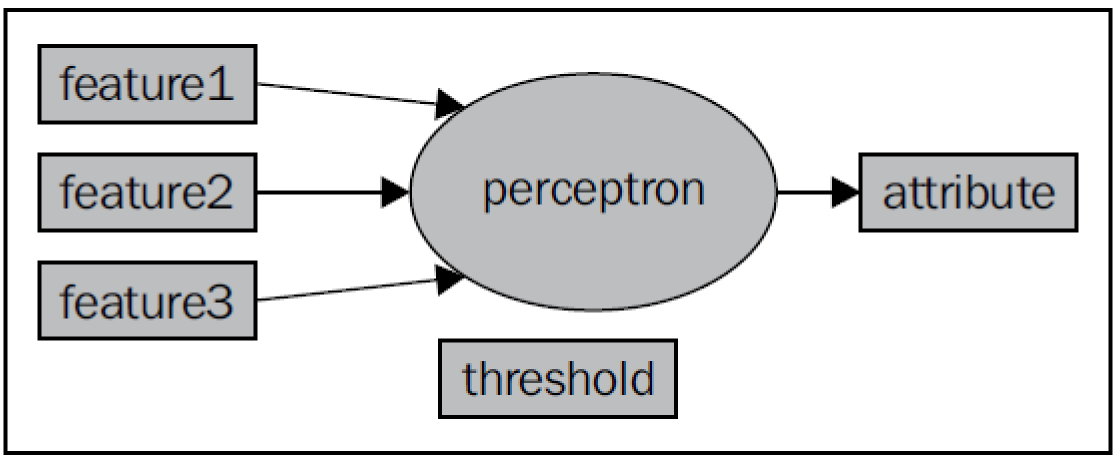


Figure 1: RMLE-7p10.png