Supervised Learning: Setting the Scene

36-600

Fall 2021

The Setting

The setting for *supervised learning* is that you have a collection of $p + 1$ measurements (recorded in columns of a data frame) for each of n objects (recorded in rows of a data frame). Of those measurements, p comprise the *predictor* (or *independent*) variables, with the last one being the *response* (or *dependent*) variable.

The goal: to model the data-generating process (i.e., to "learn a statistical model") and to discover associations between the predictor variables and the response variable. (While keeping in mind that "association is not causation.")

A statistical model is

$$y|\mathbf{x} = f(\mathbf{x}) + \epsilon\,,$$

where $f(\cdot)$ is a deterministic function that represents the expected value $E[y|\mathbf{x}]$ (the average observed value of y, given $\mathbf{x} = \{x_1, \ldots, x_p\}$...aka "the regression line"), and ϵ is an "error" that we assume is randomly sampled from some distribution, like the normal distribution. Predicted response values are given by

$$\hat{y}|\mathbf{x} = \hat{f}\,(\mathbf{x})$$

where the hats indicate estimated quantities.

An Overarching Question: Inference...or Prediction?

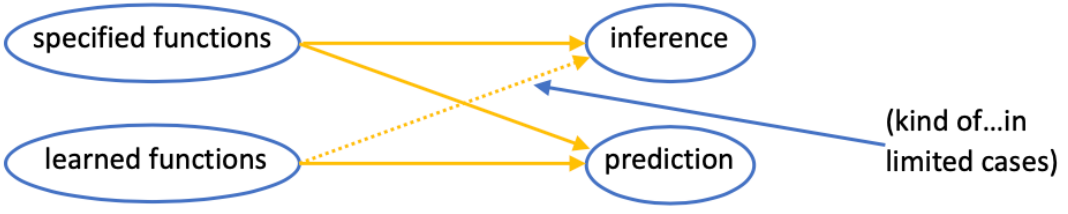
Inference: learning a statistical model and then examining and interpreting it.

- "Adding one to the number of comorbidities leads to this amount of increased cost, on average."

Prediction: learning a statistical model and then treating it as a black box.

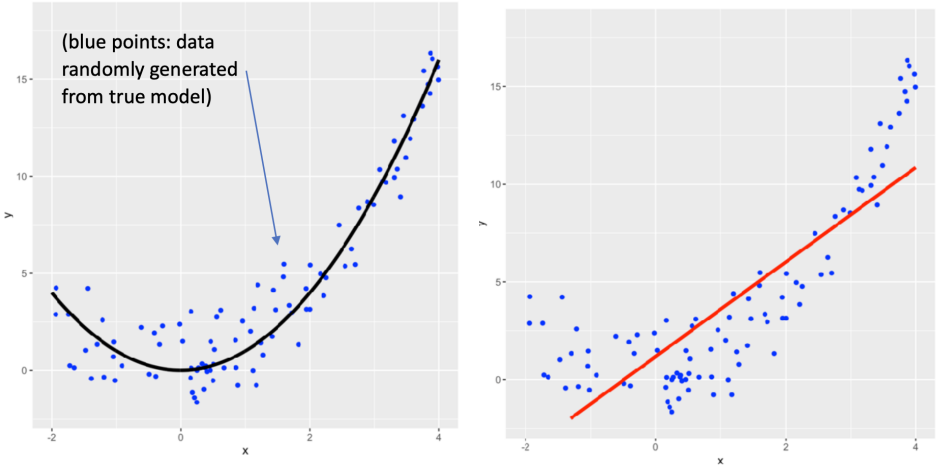
- "What is the average total insurance claim for a 60-year-old female with one intervention, no prescribed drugs, three ER visits, no complications, no comorbidities, and with a treatment duration of 100 days?"

Why does this question matter? It matters because it impacts the possible suite of models that you can apply to your dataset.



- "specified functions": functions you can write down yourself, on paper (e.g., linear regression)
- "learned functions": functions a machine learns via algorithm, given data (e.g., random forest)

Inference vs. Prediction: the Tradeoff



As shown in the left panel, data (blue points) are generated from a smoothly varying non-linear model (black line), with random noise added. In the right panel, the red line shows a simple linear regression fit to the observed data.

The statistical model of linear regression is fully parameterized and completely inflexible. As you can see, it does not provide a good estimate of $f(\mathbf{x})$...but it has the virtue of being easy to interpret.

The basic tradeoff: the more flexible a model is, the better predictions it will generate, but the harder it will be to explain.

It is very important to determine, at the start of any analysis, whether the goal is inference or prediction, and if it is inference, how important inference is, and how much inferential ability you are willing to give up if more predictive models provide substantially better fits to your data.

Inference vs. Prediction: Two More Important Points

- If inference is your goal, you should still always include "learned function" models like random forest in your suite of models that you will apply in an analysis.

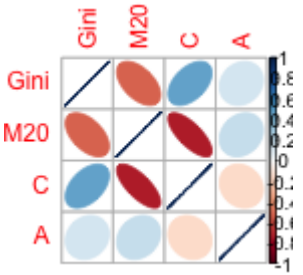
Why? If you do not try such nonlinear models, you will never know how much better such models might perform…this information is critical. If inference is your goal, and the fit for a random forest model much better that observed with linear regression, then you know that any inferences you draw from the latter model may not reflect reality (i.e., your inferences may be afflicted by biases, i.e., systematic deviations from the underlying truth).

- If prediction is your goal, you should still always include "specified function" models like linear regression in your suite of models that you will apply in an analysis.

Why? For the simple reason that the true association between the predictors and the response could be (approximately) linear. If this is the case, machine learning will not help you…and you will get the best of both worlds (inferential ability *and* predictive ability).

In short: try all reasonable models, regardless of the analysis goal. Then sort out the results.

Inferential Models: Beware Multicollinearity!



Assume that the variables above are (at least some of) your predictor variables in a given analysis.

Multicollinearity denotes the situation where one or more predictor variables are linearly related to other predictor variables.

Let's say we are trying to learn a multiple linear regression model with two predictor variables:

$$\hat{y}|x_1, x_2 = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \, .$$

Furthermore, let's suppose with are afflicted with perfect collinearity (i.e., that one of the predictors is deterministically and linearly related to the other):

$$x_2 = \beta_0' + \beta_1' x_1 \, .$$

Then:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2(\beta_0' + \beta_1' x_1) = \beta_0'' + \beta_1'' x_1 \, .$$

We cannot constrain the slope β_2 at all: we've lost the ability to make inferences about it!

Inferential Models: Beware Multicollinearity!

To continue this example:

Multicollinearity is rarely perfect, and so it typical situations we will get estimates for β_1 and β_2. The issue that will arise is that these slope estimates will have heightened uncertainty: the estimated standard errors for these slopes will be systematically larger. Thus we won't be able to make inferences that are as firm as when the two variables are uncorrelated.

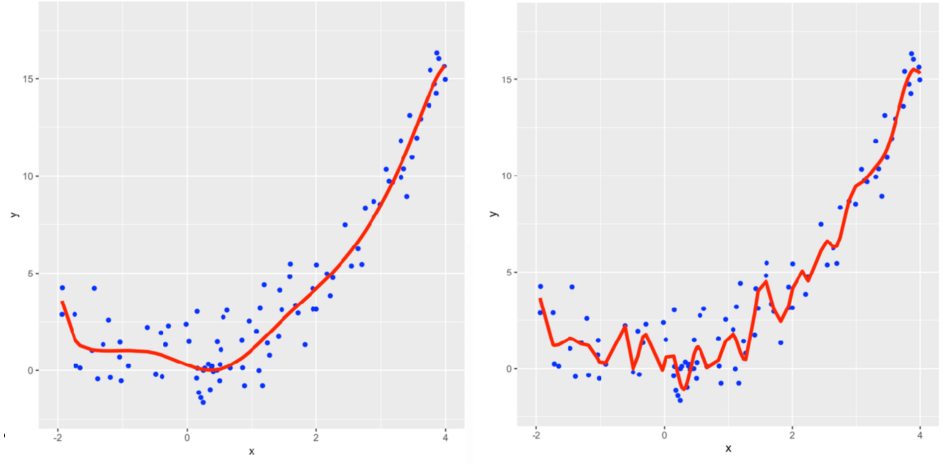
We will return to this issue when we discuss linear regression.

In the meantime, it suffices to say that:

- the effect of multicollinearity can be estimated after the fact;
- variables that have high "multicollinearity scores" can be removed from the data frame, iteratively, until we achieve of set of variables conducive to inference;
- if we remove variables, prediction *may* suffer; and, perhaps most importantly,
- if your analysis goal is prediction, *you can ignore the effect of multicollinearity completely*...it is a purely inferential issue.

As a side note, avoiding multicollinearity is one of the by-products of good experimental design!

Training Your Statistical Model



The right panel above shows a model that is overly flexible: it can provide a good estimate of $f(\mathbf{x})$ but it goes too far and overfits by modeling the noisy deviations from $f(\mathbf{x})$. Such a model is not "generalizable": it will tend to do a bad job of predicting the response given a new predictor $\mathbf{x}_o$ that was not used in learning the model.

The left panel shows a model that is close to being the optimal model: neither too flexible nor inflexible.

The question: how can we select the model at left, when we know that as we add flexibility, we will get, e.g., smaller and smaller sums of squared errors? (In other words, $\hat{y}$ will get closer and closer to y, so that eventually the sum of squared errors, $\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$, will be zero!)

Training Your Statistical Model

The answer to the question posed on the last slide: don't use all your data to train your model!

There are two common approaches for "holding out" data:

- You can split the data into two groups: one used to train models, and another used to test them. By assessing models using "held-out" test set data, we act to ensure that we get a generalizable(!) estimate of $f(\mathbf{x})$. There is no universally accepted rule regarding the proportion of data to use to train models; conventional choices include using 70% or 80%. The rest are used to test models.
- We can repeat data splitting k times, with each datum being placed in the "held-out" group exactly once. This is *k-fold cross validation*. The general heuristic is that k should be 5 or 10. k-fold cross validation is the preferred approach, for reasons of statistical performance, but the tradeoff is that CV analyses take $\sim k$ times longer than analyses that utilize data splitting, and involve more intricate coding.

Assessing Your Statistical Model

When the response variable is quantitative (like it is in the example in these notes), the most commonly used assessment metric is the mean-squared error, or MSE, computed with the *test-set data* only:

$$MSE = \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} (y_i - \hat{y}_i)^2 \, .$$

y_i is the value of the response for the i^{th} test-set datum and $\hat{y}_i$ is the predicted response value for the i^{th} test-set datum. The square-root of the MSE (the root-mean-squared error, or RMSE) indicates the average difference between $\hat{y}_i$ and y_i, and thus its value is affected by units. (In other words, don't fixate on the value of the MSE, but rather how much it changes by when we try different statistical models.)

When the response variable is categorical, the choice of metric is not quite so clear-cut. Common ones include the *misclassification rate* or *MCR* (what percentage of predictions are wrong) and the *area under curve*. Note that interpretation can be affected by *class imbalance*: if two classes are equally represented in a dataset, an MCR of 2% is quite good; but if one class comprises 99% of the data, that 2% is actually terrible. (Always know how many data are in each class!) We will return to classification model assessment metrics in a future lecture.

Reproducibility

An important aspect of a statistical analysis is that it be reproducible. You should...

1. Record your analysis in a notebook, via, e.g., `R Markdown` or `Jupyter`. A notebook should be complete such that if you give it and datasets to someone, that someone should be able to recreate the entire analysis and achieve the exact same results. To ensure the achivement of the exact same results, you should...

2. Manually set the random-number generator seed *before each instance of random sampling in your analysis* (such as when you assign data to training or test sets, or to folds):

```
set.seed(101)    # can be any number...
sample(10,3)     # sample three numbers between 1 and 10 inclusive
```

```
## [1]  9 10  6
```

```
set.seed(101)
sample(10,3)     # voila: the same three numbers!
```

```
## [1]  9 10  6
```

Model Selection

As mentioned on the first slide, model selection is picking the best model from a suite of possible models. This can be as simple as picking the regression model with the best MSE or the classification model with the best MCR. However, two things must be kept in mind:

- To ensure an apples-to-apples comparison of metrics, every model should be learned using *the same training and test set data*! Do not resample the data between the time when you, e.g., perform linear regression and when you perform random forest. (Why? See the next point.) The first thing you do in the statistical learning part of your analyses: establish the data split. Then don't touch the data again!

- An assessment metric is a *random variable*, i.e., if you choose different data to be in your training set, the metric will be different. (That is why resampling your train-test split between applications of models will ultimately lead to apples-to-oranges comparisons.)