Methods

In building our models, we use a multitude of statistical methodologies. We outline here the central methods used, both to familiarize the reader and to establish the notation we use throughout this paper.

Statistics is often split up into prediction and inference. Both are important to us here: we use inference to answer our research questions and prediction—through cross validation—to evaluate the validity of our models.

Our models combine logistic regression and multilevel regression. To evaluate our models, we use graphical tools, such as the separation plot, to examine the performance.

Logistic Regression

Statistical models are often split into regression models—models with a quantitative response—and classification models—models with a categorical response. Thus, it may seem odd that we are using a regression model when our response variable, ride rating, is a binary outcome.

But, when modeling a binary variable Y, we consider it a Bernoulli random variable,

$$Y = \text{Bernoulli}(p),$$

where p is the probability the outcome is 1 and $1 - p$ is the probability the outcome is 0. So our outcome variable Y may be binary, but the primary quantity of interest behind that outcome is p, a quantitative variable. This is why we consider logistic regression a regression method.

Logistic regression is one form of generalized linear regression. Recall that in linear regression we use data with response variable y_i and j predictors $x_{i1}, \ldots, x_{ij}$ to fit the best-fitting linear function

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_j x_{ij} + \epsilon,$$

where $\epsilon \sim N(0, \sigma^2)$, by estimating $\beta_0, \ldots, \beta_j$. We can equivalently write,

$$Y \sim N(\beta_0 + \beta_1 x_1 + \ldots + \beta_j x_j, \sigma^2).$$

We could, in fact, try to predict p with a linear regression, though such a model will always have the problem of predicting probabilities outside of the range of $[0, 1]$. That's not a recipe for simple interpretation or reliable predictions. Generalized linear regression uses a "link function," g, to modify the regression so the range of the response more accurately reflects the practical range of the variable:

$$g(y_i) = \beta_0 + \beta_1 x_1 + \ldots + \beta_j x_j + \epsilon.$$

In logistic regression, the "link" function is the logit function, $\text{logit} : [0, 1] \to \mathbb{R}$,

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right).$$

This function is also known as the log-odds, because odds are defined as $p/1 - p$ for any probability p.

So in logistic regression, we model the probability of $y_i = 1$ as,

$$\mathbb{P}(y_i = 1) = \text{logit}^{-1}(\beta_0 + \beta_1 x_1 + \ldots + \beta_j x_j).$$

Notice that the inverse logit function[1] maps values from $\mathbb{R}$ to $[0, 1]$. Thus, the function provides a convenient way to map linear combinations of other variables onto values that are valid probabilities. Other such functions exist and are also used for regressions with binary responses, such as the probit function. Logistic regression, however, is easier to interpret and more efficient to compute.

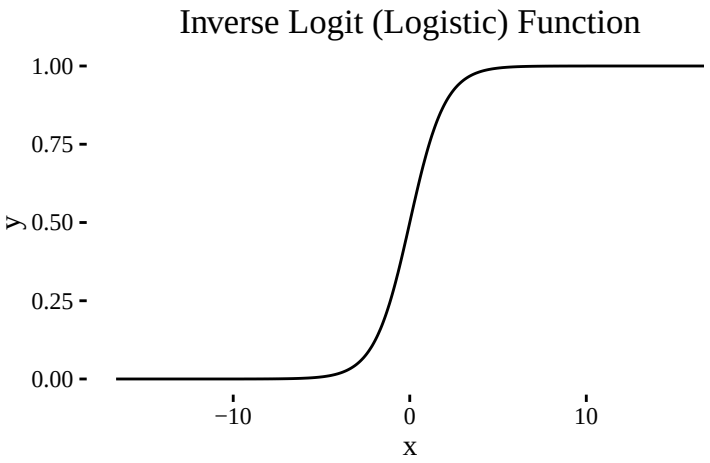

Figure 1: The inverse logit function gives a convenient way to map linear combinations of real numbers to valid probability values.

Though coefficients from a logistic regression can't be interpreted the same way as a linear model, they do have a convenient interpretation. Because the linear part of the model represents log odds, the coefficients are log odds ratios; That is, the exponentiated coefficients $e^{\beta_1}, \ldots, e^{\beta_j}$ represent the multiplicative effect a one-unit increase in the corresponding predictor has on the odds. For example, if we fit a simple logistic regression with $\mathbb{P}(Y = 1) = \text{logit}^{-1}(\alpha + \beta + X)$, we could interpret $e^{\beta} = 1.1$ as meaning that a one unit increase in X gives a 10% increase in the odds that $Y = 1$.

We started out talking about the ways logistic regression is both a classification and regression model. We will mostly care about logistic regression as a regression model. One of the main ways we will assess our models is through the accuracy of predictions. But, in the classification framework, in order to predict the outcomes $\hat{Y}$, we need to choose a probability threshold to decide which outcome we should predict for a given observation. When we care mainly about prediction, this approach can make sense. Our interest, however, lies in prediction. So rather than choose an arbitrary threshold, we will evaluate the accuracy of our predicted probabilities themselves. (How exactly do we do this? See section .)

Hierarchical Models and Mixed Effects Models

Data often contain hierarchies. For example, a set of student test scores may contain the hierarchy of districts and schools those students attend. Or a set of soil samples may have been taken at several distinct sites, thus having a hierarchy of sites and samples. In the bike ride data we examine, there is the hierarchy of riders and rides.

We will talk about different "levels" of variables corresponding to places in this hierarchy. When we refer to "ride-level variables," we refer to variables that are specific to a ride, whereas we refer to "rider-level variables" as those specific to the rider, and thus also all the rides that rider takes. For example, we consider length a ride-level variable and total number of rides taken a rider-level variable.

We will also discuss road segment-level variables, which are variables that are specific to the road segments in the route of a ride (e.g. length, presence of bike lanes, etc). But there isn't a clear road segment-ride hierarchy:

[1] sometimes called the logistic function

each ride contains multiple road segments and each road segment is contained by multiple rides. Thus, this isn't a case where multilevel modeling is applicable. (The ideas behind it, though, may be fruitfully adapted.)

Gelman describes two traditional ways of dealing with hierarchical data that multilevel models contrasts with: "complete pooling" and "no pooling."[2] In "complete pooling," we ignore the group-level variables, and give identical estimates for parameters for every group. In "no pooling," we do entirely separate regressions for each group. Multilevel models are a comprimise between these extremes ("partial pooling", as Gelman calls it) where all the data is considered in a single regression with some parameters shared between groups and some different between groups. (This will become clearer when we introduce examples of models.)

These multilevel models work for other forms of regression, but we will focus on logistic regression, as it is the method we use in this paper. We will be using notation adapted from Gelman and Hill's description of multilevel models.[3] Consider a data set composed of

- i observations of a binary response variable y_i,
- m observation level predictors $X_i = x_i^1, \ldots, x_i^m$,
- j groups in which the observations are split into,
- l group-level predictors $U_{j[i]} = u_{j[i]}^1, \ldots, u_{j[i]}^l$, where $j[i]$ is the group of the ith observation.

We could fit a model where the intercept varies by group:

$$\mathbb{P}(y_i = 1) = \text{logit}^{-1}(\alpha_{j[i]} + X_i\beta),$$
$$\alpha_{j[i]} \sim N(\gamma_0 + U_{j[i]}\gamma, \sigma_\alpha^2),$$

where $\alpha_{j[i]}$ is the intercept for the jth group, β is a vector coefficients for the observation-level predictors, γ_0 are the group-level intercepts, and γ is a vector of coefficients for the group-level predictors. We could also specify a similar model where there are no group-level predictors, such that we simply have different intercepts for each group,

$$\alpha_{j[i]} \sim N(\gamma_0, \sigma_\alpha^2),$$

We can also consider a model that has slopes varying by group. For simplicity, let's consider just one observation-level predictor, x_i, that will have varying slopes $\beta_{j[i]}$ as well as one group-level predictor. We could specify the model as,

$$\mathbb{P}(y_i = 1) = \text{logit}^{-1}(\alpha_{j[i]} + \beta_{j[i]}x_i),$$

$$\begin{pmatrix} \alpha_j \\ \beta_j \end{pmatrix} = N\left(\begin{pmatrix} \gamma_0^\alpha + \gamma_1^\alpha u_j \\ \gamma_0^\beta + \gamma_1^\beta u_j \end{pmatrix}, \begin{pmatrix} \sigma_\alpha^2 & \rho\sigma_\alpha\sigma_\beta \\ \rho\sigma_\alpha\sigma_\beta & \sigma_\beta^2 \end{pmatrix}\right).$$

Tools for evaluating models

After fitting our models, we will want to know, how do each of our models compare? Did adding a particular term enhance or diminish the accuracy of our model? While we are fitting our models for inference, we will be evaluating them by the accuracy of their predictions. That is, how accurately and confidently can the model dicern which rides were rated as good and which as bad?

The predictions we will evaluate will be done in the context of cross validation. Usually, we will just use a simpy 2-fold cross validation, where we split the data into two random samples, fitting the model on one set (the "training set"), and using the model to predict on the other set (the "testing set") to check accuracy. We

[2] @gelman, p. 7
[3] @gelman, p. 251–252

can also do k-fold cross validation, where we split the data into k samples, and for each of the samples we train the model with the other $k-1$ samples and then test with the left out sample.

When splitting the data into testing and training sets, we can not do a simple random sample. If we do, we will end up with riders in the testing set that aren't in the training set, which our models that use rider information cannot predict on. Thus, we will usually do stratified sampling of rides with riders as strata.

When evaluating the predictions from a model, statistics such as classifications rate, false-positive rate, and true-negative rate can be calculated for each validation. For a more comprehensive look at predictive accuracy, we use the separation plot as well as homemade plot to assess our models.

The Separation Plot

The separation plot, created by Greenhill et. al.[4], is designed to show how well a logistic regression model can distinguish between high and low probability events.

Creating a separation plot first requires a model fit to training data and testing data to evaluate predictive accuracy on. From the testing data, we need a vector Y of observed binary response data and a vector $\hat{Y}$ of predicted probabilities of a 1 for each observation, predicted using our model fitted to training data.

We plot the data $(Y, \hat{Y})$ as a sequence of vertical strips, colored according to observed outcome, Y, and ordered from low to high probability based on $\hat{Y}$. A curve is superimposed upon the stripes showing the $\hat{Y}$ as a line graph. And finally, a small triangle is placed indicated the point at which the two colors of lines would meet if all observations $Y = 0$ were placed to the left of all the $Y = 1$ observations; in other words, showing where the boundary would be if the two classes were perfectly separated by the model.

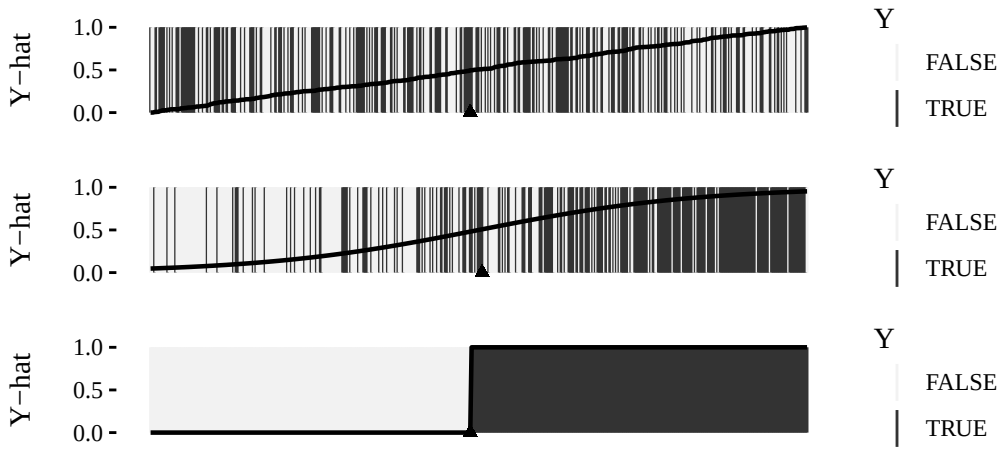

Figure 2: Above we have examples of three separation plots. The first plot shows what it looks like when Y and $\hat{Y}$ are uncorrelated. The second plot shows a fairly good model, where the Y are generated as Bernoulli($\hat{Y}$). The third plot shows a model where the responses are fully separated.

Separation plots don't do well with larger sample sizes: if there are two many observations, it becomes difficult to read. This is because the resolution of the medium the plot is displayed on may not be fine-grained enough to show a single observation's line, thus obscuring the pattern. In these cases, we can plot a random sample or modify the graph to be more suitable.

[4] @greenhill2011

Our Probability Plots

One graphical tool we adapted for this project maps probabilities of the outcome against continuous variables. We use this for exploration to understand the relationship between predictors and our binary response, but it's also useful for evaluating our models.

The goal of the plot is to show a measure of an emperical $\hat{p}$, the probability that the response is 1, on the y-axis, and a continuous variable X on the x-axis. A simple approach would be to bin the observations by the values of X, and within each bin, model the data as a bernoulli random variable and compute a local value for p.

The problem with the binning approach is that we get some error due to how we happen to bin things. An alternative is to do the same computation for a sliding window rather than a bin. This reduces the error due to our binning, but we still have to choose a proper window width. We will choose the window width that minimizes the leave-one-out error (the sum of the squares of the error in predicting a single observations based on a regression fitted to the rest of the observations.)

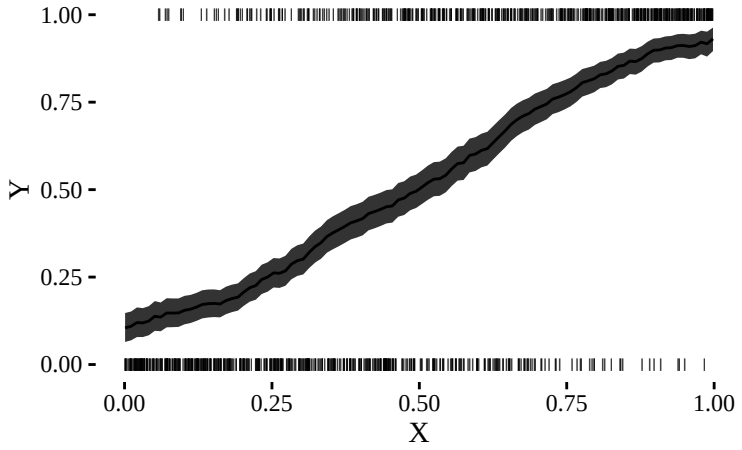

Figure 3: Example of probability plot. 1000 Xs were sampled from the distribution Uniform$(0, 1)$ and each Y_i was generated from the distribution $Y_i \sim$ Bernoulli(X_i). The smoothed curve shows an approximation of p for each value of X_i, with 90% confidence intervals in grey.

We also make use of this plot to compare the predicted probabilities for a testing set to the empirical probablilities in the data. J. Esarey and A. Pierce advocate using this in their heatmap plot to assess model fit, showing that such a plot will show model mispecification problems better than other measures commonly used, like AIC.[5]

[5] @esarey2012