Training and Testing Models

Brian Kim

General Supervised Learning Framework Revisited

In supervised learning, we are focused on finding the relationship between a **label** y and **features** x.

$$y = f(x)$$

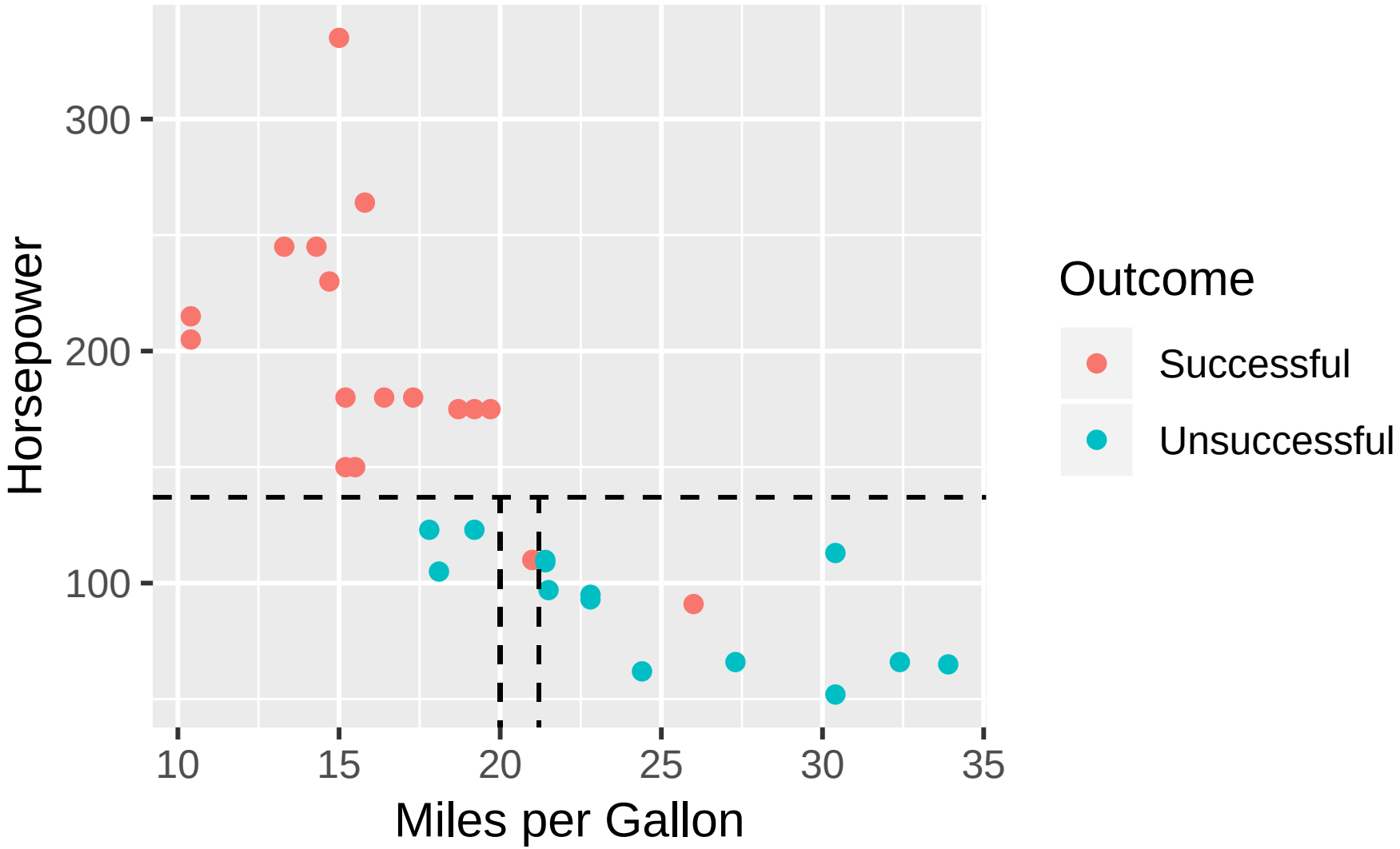
"Learning" is finding a function f that minimizes **future error** in recovering y.

Motivation

Suppose we have a machine learning model (e.g. decision trees). How do we know it can **generalize** to the overall population?

That is, we know that the model fits well on the data we used ... precisely because we fit it on that data. How do we know it will predict **future** data points well?

Example


Horsepower
300
200
100
10
15
20
25
30
35
Miles per Gallon
Outcome
Successful
Unsuccessful

Overfitting

The main issue is that our model isn't **generalizable** to the overall population because it is fit to be **too specific to our data**. This is called **overfitting**.

Validation

In order to make sure our results are generalizable, we typical **hold out** a portion of our data to **test** the model on. That is, we split our data into two subsets:

- The **training set** is the subset that we use to build the model on.
- The **test set** is the subset that we use to evaluate how well our model is doing.

Selecting the Train/Test Split

A basic way of doing this might be just a simple random split. That is, we randomly assign a portion (for example, half) of the data as the **training set** and the rest as the **test set**.

But what if the test set that we choose just happens to be unusual in some way due to random chance?

Cross-Validation

A popular method is **k-fold cross-validation**. To do k-fold cross validation, you

1. **Split the data** into k equal partitions.
2. Designate one partition as the **test set** and the rest as the **training test**.
3. **Fit the model** using the training set and evaluate on the test set.
4. **Choose another** of the remaining partitions that you haven't set aside as a test set yet.
5. **Repeat** until every single partition has been a test set once.

Example: Three-fold Cross-Validation

To do a **three-fold cross-validation**, you first split the data into three equal parts. Let's call them **A, B, and C**.

First, you fit the model on **B** and **C**, then test on **A**.

Then, you fit the model on **A** and **C**, then test on **B**.

Finally, you fit the model on **A** and **B**, then test on **C**.

Other Methods

There are many possible ways to validate your results.

- Holdout sample
- Leave one out cross validation
- Random subsampling
- Temporal holdouts