Introduction to ML with R

R workshops, 2021

Baruch College

Víctor Peña, January 2021

Logistics

- Course website: https://vicpena.github.io/workshops/2022/introML
- My email: victor.pena@baruch.cuny.edu
- I'll cover some section of Hands-On Machine Learning with R, by Boehmke and Greenwell
  - Website: https://bradleyboehmke.github.io/HOML/
- I'll post the code I write on the course website

Schedule

- Today
  - Introduction to ML, regularized regression, logistic regression
- Wednesday
  - Classification methods
- Friday
  - Clustering methods

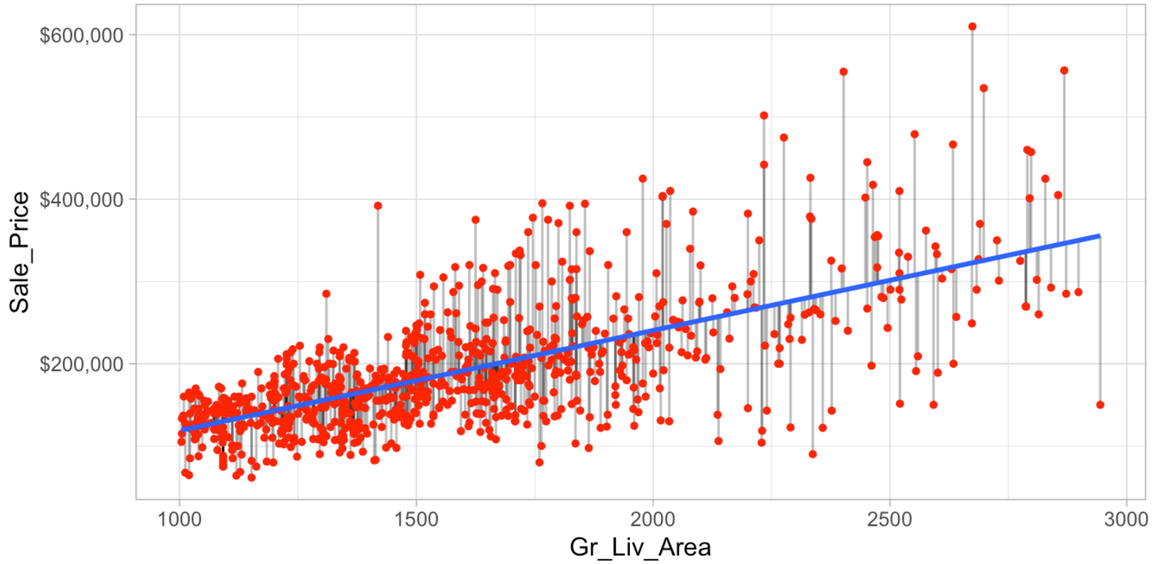
Introduction to ML

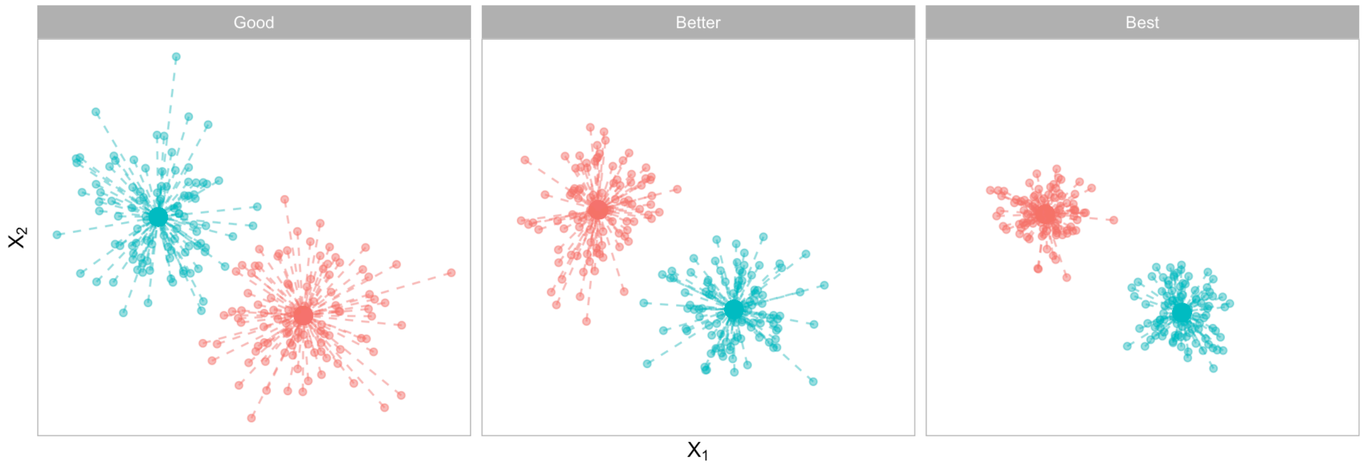
What is Machine Learning?

- Hard to say! There isn't an agreed-upon definition
- Collection of algorithms that "*learn*" from data
- When people say "machine learning", they're usually thinking about algorithms / methods that have been developed in the last 20 years from statisticians, computer scientists, and engineers

Machine Learning terminology

- **Supervised learning:** predictive models (predict an outcome given predictors)
- **Unsupervised learning:** descriptive models (clustering / segmentation)





Supervised learning

- Supervised learning: predictive models
  - We want to predict a target (i.e., an outcome) given a bunch of predictors
  - Two types:
    - **Regression:** predicting a continuous outcome (think usual linear regression)
    - **Classification:** predicting a categorical outcome (think logistic / multinomial regression)

Unsupervised learning

- **Descriptive analysis** of the dataset without making predictions
- Most often, unsupervised learning algorithms are either
  - **Clustering algorithms:** Creating groups of interesting observations or variables
  - **Dimensionality reduction algorithms:** Reduce the dimensionality of a large dataset to something that we can visualize, trying to lose as little information as possible [The goal is usually to find the "best" 2- or 3-dimensional representation of a dataset with many variables.]

Supervised learning

Modeling process in supervised learning

- The modeling pipeline usually looks something like this

1. **Pre-process the data:** read in the data, data cleaning, transform / create new variables

2. **Fit model(s):** run model(s), estimate / tune its parameters

3. **Assess model performance:** in ML, we're mostly interested in assessing predictive performance

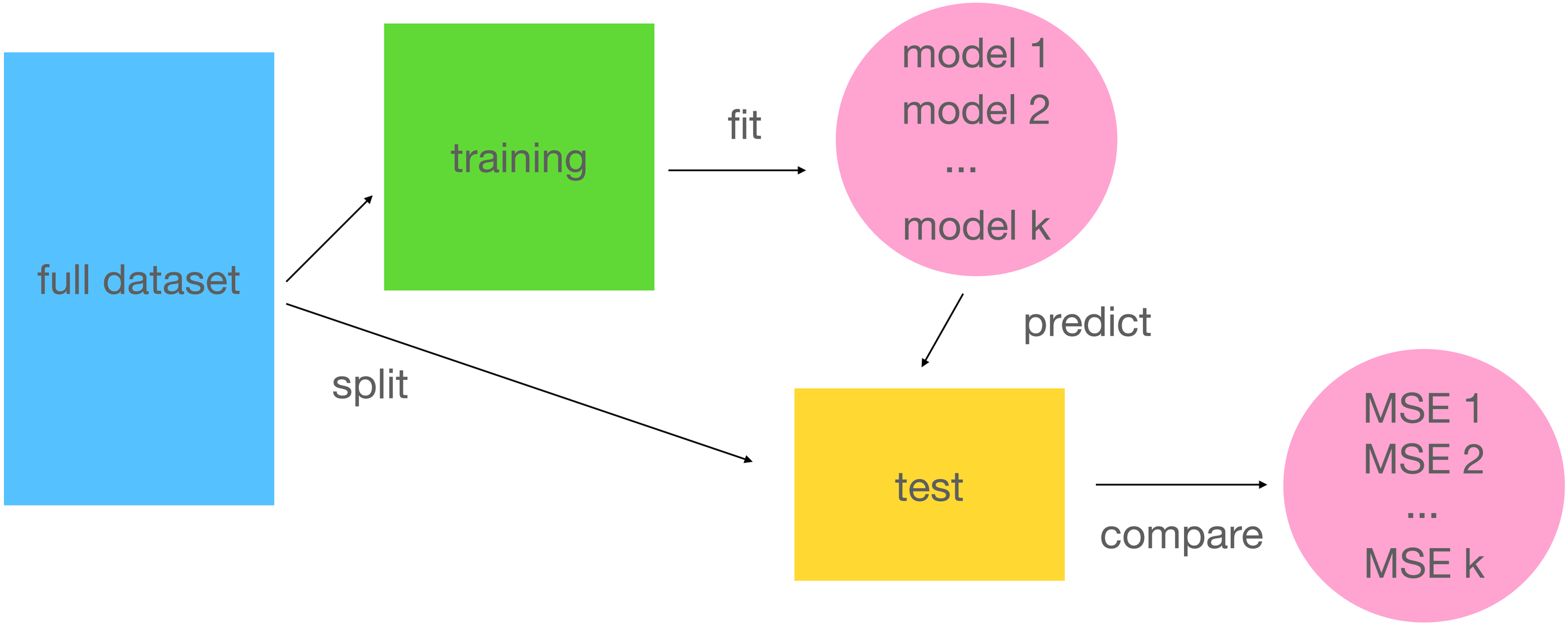
Assessing predictive performance

- There are different approaches to doing this, but the following is a popular approach
- When you get the dataset, split it into 2 separate sets:

1. **Training set:** We use this dataset to look at the data and try out different models. Here, we identify model(s) that fit the data well
2. **Test set:** We assess the predictive ability of the model(s) we identified in the training set by evaluating how good they are at predicting the observations in the test set (which were not used to train the models in step 1)

- If your sample size is small, you might not be able to do this, but if you can, I think it's a good idea

Conceptual framework



Assessing model performance in test set

- **Regression** (i.e. continuous outcome)
  - Mean squared error [average of (actual - predicted)^2], mean absolute error, etc.
- **Classification** (i.e. categorical outcomes)
  - Accuracy (% of observations that are well classified), precision, sensitivity, specificity, area under the curve (AUC)

Regularized regression

Regression suffers with correlated predictors

- The predictive performance of linear regression models suffers when the predictors are highly correlated
  - Standard errors are inflated, giving rise to intervals that are wide
  - Small changes in the predictors can lead to big changes in predictions
    - That is, our predictions become non-robust (i.e. unstable) to small changes in the predictors

Regularized regression helps!

- Regularized regression
  - Yields more "robust" estimates and usually outperforms linear regression in prediction tasks when there are highly correlated predictors
  - It can also handle datasets where there are more predictors than there are observations (with theoretical guarantees, under some conditions)

Example: elastic net

y_i outcome for i-th individual

$x_{i1}, x_{i2}, \ldots, x_{ip}$ predictors for i-th individual

$\beta_1, \beta_2, \ldots, \beta_p$ regression coefficients

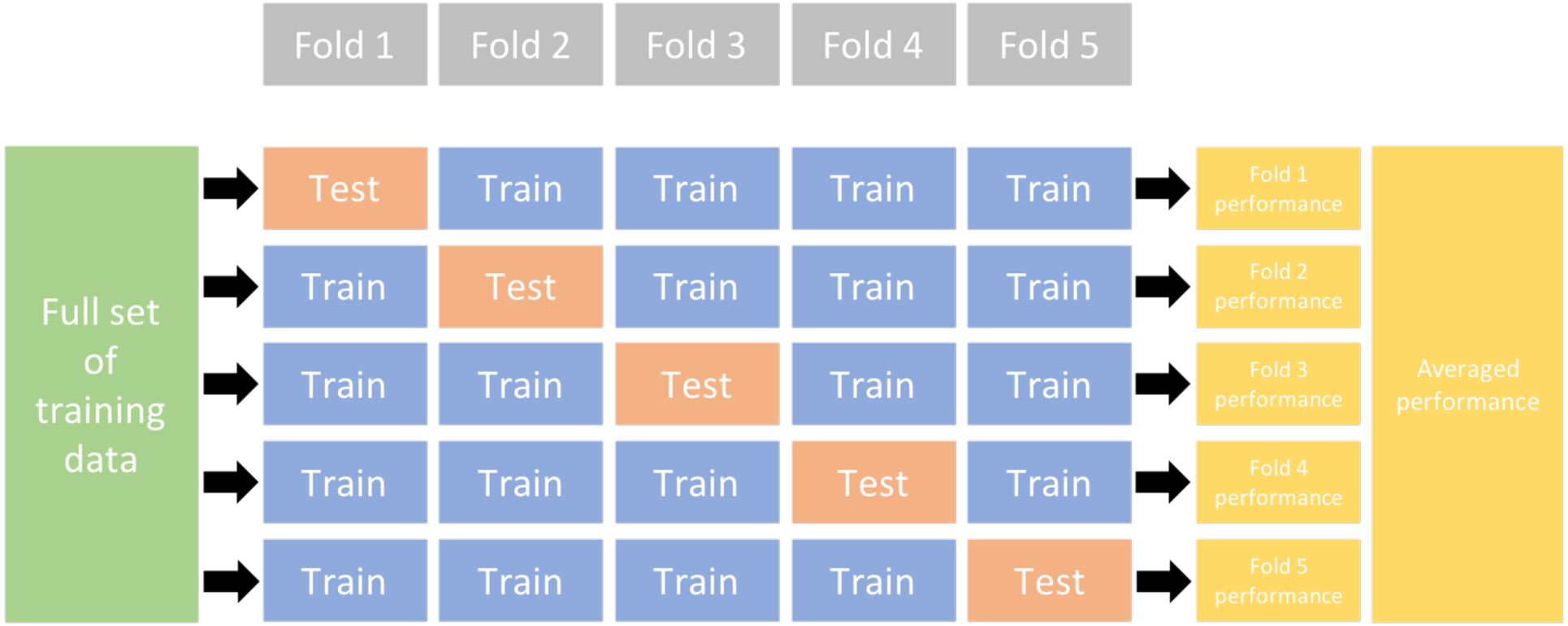
Ordinary Least Squares $$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2$$

Elastic net $$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2 \right\}$$

How does it work?

- Without getting into technical details...
  - Regularized regression adds additional parameters that reduce the variability in our estimation of the regression coefficients
  - How are those new parameters estimated?
    - Usually some form of cross-validation

Estimating using k-fold cross validation



How to implement regularized regression

- Thankfully, R has nice packages that allow us to fit regularized regressions easily
- For example, you won't have to implement cross-validation yourself, which is nice
- In this workshop, we're going to implement elastic nets, which are implemented in R in library(glmnet). We will tune the hyperparameters using library(caret)
- If you are interested in learning more technical details about regularized regression, you can read Chapter 6 of Hands on Machine Learning with R