Big Data and Economics

Introduction to Machine Learning

Kyle Coombs, adapated from Tyler Ransom
Bates College | ECON/DCS 368

Table of contents

Prologue

Prologue

- **Computers closed:** Introduction to Machine Learning
- **Computers open:** Application of **tidymodels** in R to train ML models
- We'll discuss the differences between machine learning and econometrics
    - What can each camp learn from the other?
- Today we'll discuss the basics of machine learning
    - What is the intuition?
    - What are the goals? How do we measure accuracy?
    - What is the bias-variance tradeoff?
    - How do we tune models?

What is Machine Learning?

What is Machine Learning?

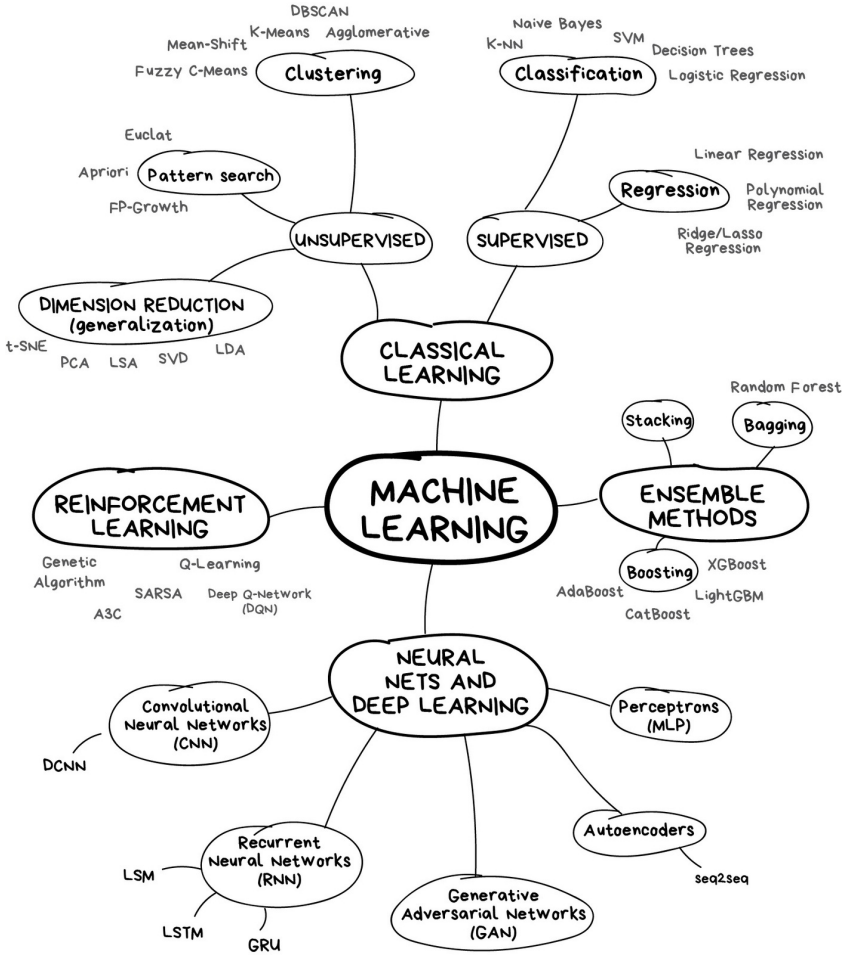
ML: Allowing computers to learn for themselves without being explicitly programmed

- **USPS:** Computers read handwritten addresses and sort mail accordingly
- **Google:** AlphaGo, AlphaZero (computers that are world-class chess, go players)
- **Apple/Amazon/Microsoft:** Siri, Alexa, Cortana voice assistants understand speech
- **Facebook:** automatically finds and tags faces in a photo

In each of the above examples, the machine is "learning" to do something only humans had previously been able to do

Put differently, the machine was not programmed to read numbers or recognize voices -- it was given a bunch of examples of numbers and human voices and came up with a way to predict what's a number or a voice and what isn't

Map of Machine Learning

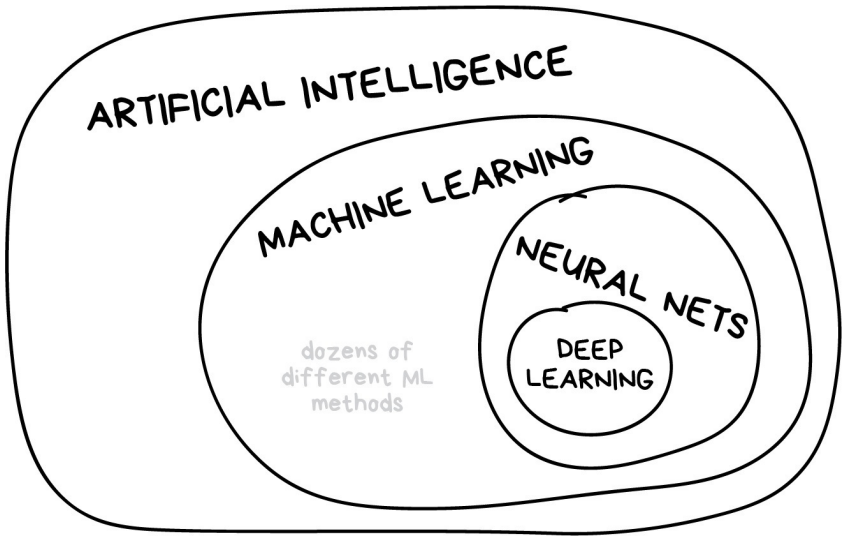


Map of Machine Learning from Motaz Saad. We'll do trees, forests, and penalization!

Artificial intelligence vs. Machine

AI: Constructing machines (robots, computers) to think and act like human beings

Thus, machine learning is a (large) subset of AI



Map of AI to Machine Learning from Motaz Saad.

Econometrics vs. Machine Learning

Econometrics vs. Machine Learning

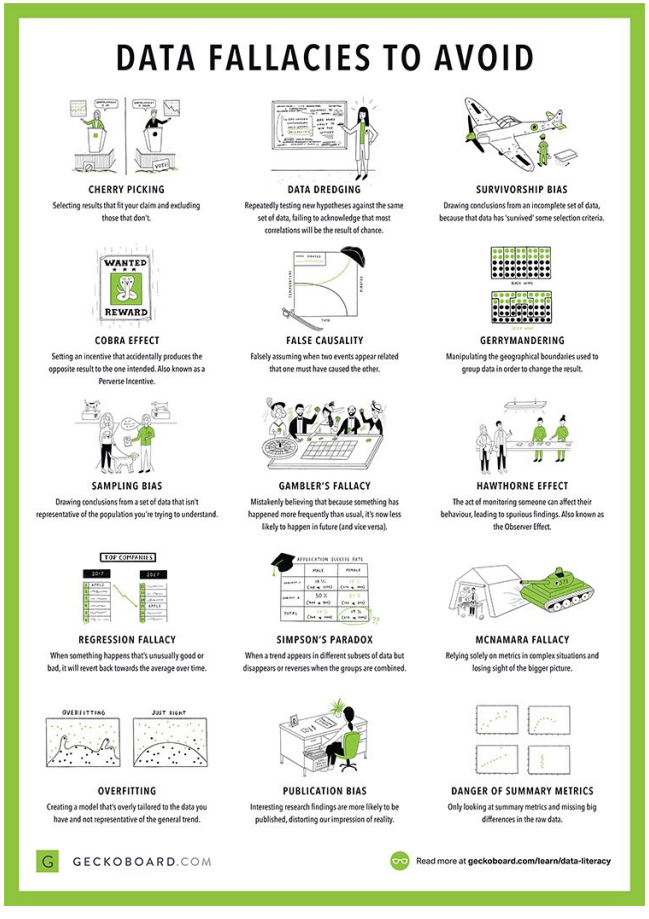
- **Econometrics** is all about understanding the causal relationship between a policy variable x and an outcome y
- **Machine Learning** is all about maximizing out-of-sample prediction
- **Econometrics** is all about finding $\hat{\beta}$
- **Machine Learning** is all about finding $\hat{y}$

Important questions

- How we can combine the tools of economic theory, econometrics, and ML to build better empirical economic models?
    - Answers come from various lectures given by Susan Athey, who is an economics professor at Stanford and who is the foremost expert in these matters
    - A nice podcast on the topic is available here (11/16/2012 episode)
- In what ways do econometrics and machine learning differ?
    - It helps to lay out exactly what the strengths of limitations of each approach is, so that we know what the comparative advantage is of each
- Why is it important to be aware of so-called "data fallacies"?

Data Fallacies

Let's briefly discuss the following Data Fallacies (full content available here, download a PDF of the image below)



Importance of data

- In social science (and business) settings, we are typically interested in understanding various phenomena
    - Will improving a neighborhood improve people's chances of escaping poverty?
    - Will increasing the minimum wage increase unemployment?
        - What about an increase to the Earned Income Tax Credit?
    - How does reducing the price of healthcare change health outcomes?
- We can try to answer these questions using economic theory
- But in many cases theory only predicts a sign, not a magnitude

Types of variables

There are typically three **types of variables** available to us in any kind of data set:

1. Outcome variable
    - It is the variable we are trying to explain (earnings, employment, job applications, physical health, etc.)
2. Treatment variable
    - It is a variable that can be changed by a policymaker to affect outcomes (minimum wage, class size, insurance, etc.)
3. Control variables
    - These are variables that explain outcomes, but that cannot be changed by a policymaker (e.g. demographic variables, longstanding cultural traditions, medical technology etc.)

Scarcity of high-quality data

How ubiquitous are the three types of data?

- **Observational data** is by far the most common type of data. Why? because it's much easier to collect: we simply need to measure the variables we're interested in. And observational data can be used for many different purposes
- **Quasi-experimental data** is the next most common type. This is observational data, but restricted to instances where there are some randomly or quasi-randomly assigned variable(s) available
- **Experimental data** is least common, simply because it costs money to run an experiment, and typically the experiment is used to analyze a specific question, and can't easily be used to answer other questions

Which fallacies is machine learning good at addressing? Which fallacies is econometrics good at addressing?

Goal of econometrics

The goal of econometrics is to make counterfactual predictions:

- What *would happen* to a child's test scores if she were assigned to a smaller class?
- What *would happen* to a child's lifetime earnings if she were moved to a higher mobility neighborhood?
- What *would happen* to labor supply if the earned income tax credit were increased?
- Other ideas?

Goal of econometrics (cont'd)

- We don't get to observe the world under these alternative policies, so we can't simply find the answers in the data
- Knowing the counterfactual requires being able to measure a **causal effect**
  - i.e. "the goal of econometrics is to find $\hat{\beta}$" where here we mean $\hat{\beta}$ to be the causal impact of X on y
- Being able to measure a causal effect requires making assumptions. That's what economics is all about!

Ways to measure causal effects (cont'd)

Ways to measure causal effects (cont'd)

1. Field experiments (i.e. collect experimental data)
    - Causal effects immediately visible due to experimental design
2. "Reduced form" methods using quasi-experimental or observational data
    - These methods include instrumental variables, regression discontinuity, and difference-in-differences (among others)
    - The goal is to separate "good" variation (i.e. randomness in the instrument or randomness in the cutoff) from "bad" variation (non-random variation in treatment)
3. Structural models using observational data
    - Make assumptions (e.g. profit maximization, rational decision-making)
    - Estimate primitives of an economic model (e.g. preferences, production fn.)
    - Use the estimated primitives to predict what would happen under a counterfactual scenario

Primary statistical concern of 'metrics

- The primary statistical concern of econometrics is sampling error
  - In other words, the goal is to quantify the uncertainty around $\hat{\beta}$ due to randomness in the sampling of the population
  - This is the infamous standard error that econometricians obsess over
- One wild thing about econometrics is that there is no format attention paid to model misspecification error!
  - The functional form and specification of the model are assumed to be 100% correct, such that the only error that remains is the sampling error
  - Sampling error is what generates the standard errors that we use in our hypothesis testing

Goal of machine learning

- In contrast, the goal of machine learning is to come up with the best possible out-of-sample prediction
    - Or the primary concern of machine learning being $\hat{y}$
- To get this prediction, a lot of effort is spent on validating many possible models
- However, if the world changes in a fundamental way, the trained predictive model is no longer useful!

Primary statistical concern of ML

- The primary statistical concern of machine learning is model misspecification error
- The goal is to make sure that the best prediction is had by tuning and validating many different kinds of models
- This is what machine learning practitioners obsess about
- Concepts:
  - **regularization** (i.e. penalizing overly complex models)
  - **prediction accuracy** (i.e. how well does the model predict out-of-sample)
  - the **bias-variance tradeoff** (i.e. the tradeoff between overly simple and overly complex models)
  - **cross-validation** (i.e. tuning parameters to maximize out-of-sample fit)

'metrics and ML can learn from each

Econometrics	Machine Learning
Less emphasis on standard errors, more emphasis on model misspecification and model selection	Find ways to obtain causal estimates from observational data that still predict well out-of-sample
Do more model validation (e.g. Delavande and Zafar, 2019)	Figure out how to implement methods like instrumental variables in machine learning models
Use more types of data	(e.g. better click prediction leveraging quasi-experimental or experimental data)
Test assumptions that come baked-in to models	

(Each cell is what the one field can learn from the other)

Machine learning methods in economics

- **Regression penalization**: which of all these variables do I put in my regression?
  - **LASSO:** penalizes the sum of the absolute values of coefficients in model
  - **Ridge:** penalizes the sum of the squared values of coefficients in model
  - Example: Derenoncourt (2022) predicts historical Black migration patterns to estimate the effect of the Great Migration on upward mobility in black communities
- **Decision trees**: which variables give me the best prediction?
  - **Random Forests**: (roughly) aggregate your trees to get better predictions
  - Example: Kleinberg et al. (2018) use trees + qausirandom assignment of judges to predict optimal bail decisions that minimize crime to assess "mistakes" by judges
- **Causal Forests**: split the sample to see how varied the causal effects of a treatment are?
  - Extension of random forests to causal inference
  - Sometimes leverages **bootstrapping**
  - Example: Jon Davis and Sara Heller (2017) estimate how treatment effects vary for at-risk youth in a summer job program

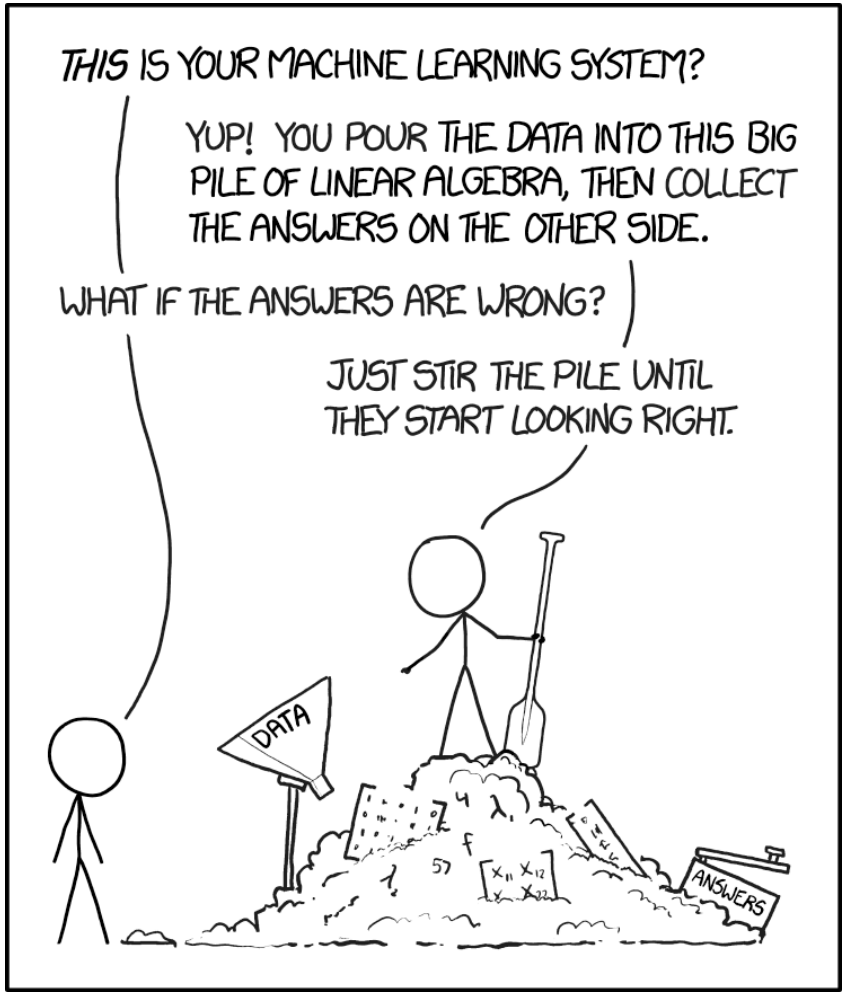
Fundamentals of Machine Learning

Objective of Machine Learning

The fundamental objective is to maximize out-of-sample "fit"

- But how is this possible given that -- by definition -- we don't see what's not in our sample?
- The solution is to choose functions that predict well in-sample, but penalize them from being too complex
- **Regularization** is the tool by which in-sample fit is penalized, i.e. regularization prevents overly complex functions from being chosen by the algorithm
- **Overfitting** is when we put too much emphasis on in-sample fit, leading us to make poor out-of-sample fit

Not the objective of ML



Taken from the always poignant XKCD. Sometimes it's hard to tell what's going on inside the black box!

Elements of Machine Learning

1. Loss function (this is how one measures how well a particular algorithm predicts in- or out-of-sample)
2. Algorithm (a way to generate prediction rules in-sample that can generate to out-of-sample)
3. Training data (the sample on which the algorithm estimates)
4. Validation data (the sample on which algorithm tuning occurs)
5. Test data (the "out-of-sample" data which is used to measure predictive power on unseen cases)

The algorithm typically comes with **tuning parameters** which are ways to regularize the in-sample fit

Cross-validation is how tuning parameters are chosen

Example

- Suppose you want to predict adulthood earnings
- You have a large number of relevant variables
- What would you do?
    - You would want to have a model that can detect non-linear relationships (like a USPS handwriting reader)
    - You would also want to have a model that you can tractably estimate
    - And a model that will predict well out-of-sample

Option 1: separate dummies for people

- In this scenario, you run `feols(log(earnings) ~ as.factor(person))`
- What you get is a separate adulthood earnings prediction for every single person
- But what to do when given a new person that's not in the sample?
    - Which person in the sample is the one you should use for prediction?
- The resulting prediction will have horrible out-of-sample fit, even though it has perfect in-sample fit
- This is a classic case of **overfitting**
- We say that this prediciton has **high variance** (i.e. the algorithm thinks random noise is something that is important to the model)

Option 2: as a function of social mobility

- Let's use the social mobility of those born in 25th percentile!
- In this scenario, you simply run `lm(log(earnings) ~ kfr_p25)`
- When given a new person with a given `kfr_p25`, you will only look at the mobility of children growing up at 25th percentile
- This algorithm will result in **underfitting** because the functional form and features it uses for prediction are too simplistic
- We say that this prediction has **high bias** (i.e. the algorithm does not think enough variation is important to the model)

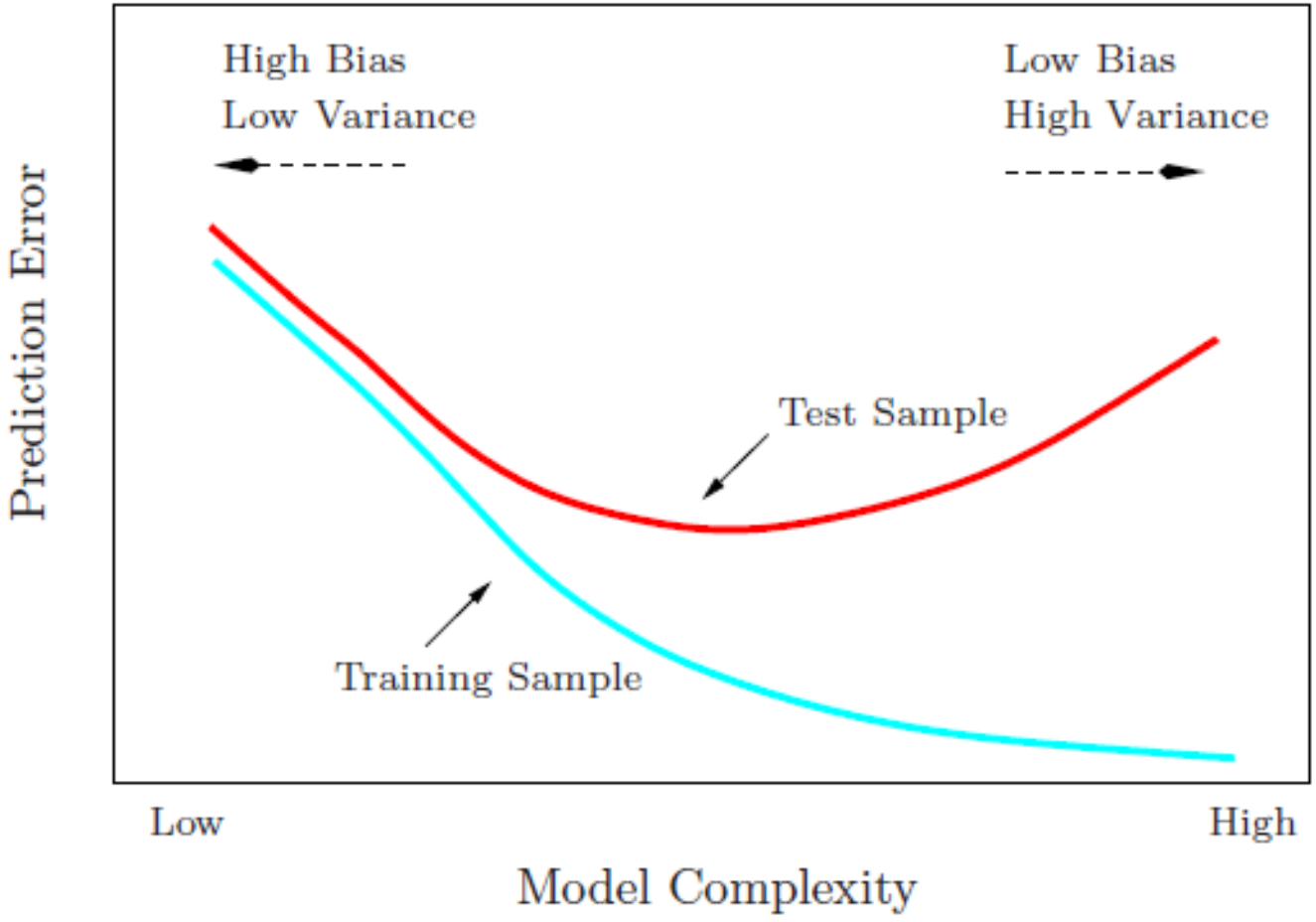
Bias-variance tradeoff

The **bias-variance tradeoff** refers to the fact that we need to find a model that is complex enough to generalize to new datasets, but is simple enough that it doesn't "hallucinate" random noise as being important

The way to optimally trade off bias and variance is via **regularization**

Visualizing the bias-variance tradeoff

The following graphic from p. 194 of Hastie, Tsibshirani, and Friedman's *Elements of Statistical Learning* illustrates this tradeoff:



Measuring prediction accuracy

Measuring prediction accuracy when y is continuous

$$\text{Mean Squared Error (MSE)} = \frac{1}{N}\sum_i (y_i - \hat{y}_i)^2$$

$$\text{Root Mean Squared Error (RMSE)} = \sqrt{\frac{1}{N}\sum_i (y_i - \hat{y}_i)^2}$$

$$\text{Mean Absolute Error (MAE)} = \frac{1}{N}\sum_i |y_i - \hat{y}_i|$$

where N is the sample size

Measuring prediction accuracy

Measuring prediction accuracy when y is continuous

$$\text{Mean Squared Error (MSE)} = \frac{1}{N}\sum_i (y_i - \hat{y}_i)^2$$

$$\text{Root Mean Squared Error (RMSE)} = \sqrt{\frac{1}{N}\sum_i (y_i - \hat{y}_i)^2}$$

$$\text{Mean Absolute Error (MAE)} = \frac{1}{N}\sum_i |y_i - \hat{y}_i|$$

where N is the sample size

Measuring prediction accuracy when y is binary

The **confusion matrix** which compares how often y and $\hat{y}$ align (i.e. for what fraction of cases $\hat{y} = 0$ when $y = 0$)

Example confusion matrix

	$\hat{y}$	
y	0	1
0	True negative	False positive
1	False negative	True positive

Using the confusion matrix

	$\hat{y}$	
y	0	1
0	True negative	False positive
1	False negative	True positive

Where are Type I and Type II errors in a confusion matrix?

Using the confusion matrix

	$\hat{y}$	
y	0	1
0	True negative	False positive
1	False negative	True positive

Where are Type I and Type II errors in a confusion matrix?

- **Type I error**: false positive (i.e. $\hat{y} = 1$ when $y = 0$)
- **Type II error**: false negative (i.e. $\hat{y} = 0$ when $y = 1$)

The three most commonly used quantities that are computed from the confusion matrix are:

1. **sensitivity or recall:** fraction of $y = 1$ have $\hat{y} = 1$? (What is the true positive rate?)
2. **specificity:** fraction of $y = 0$ have $\hat{y} = 0$? (What is the true negative rate?)
3. **precision:** fraction of $\hat{y} = 1$ have $y = 1$? (What is the rate at which positive predictions are true?)

The goal is to trade off Type I and Type II errors in classification

F1 score

The **F1 score** is the most common way to quantify the tradeoff between Type I and Type II errors

$$F1 = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}}$$

- $F1 \in [0, 1]$ with 1 being best
- It is the harmonic mean of recall and precision
- There are a bunch of other quantities that one could compute from the confusion matrix, but we won't cover any of those

Why use the confusion matrix?

- We do not want to "game" our accuracy measure by always predicting "negative" (or always predicting "positive")
- Consider the case of classifying emails as "spam" or "ham"
    - There are few "spam" messages relative to "ham" messages
    - If only 1% of messages are spam, we don't want to say an algorithm is superior if it always predicts "ham" correctly, but does not pin down the 1% of spam

The F1 measure attempts to quantify the tradeoff between Type I and Type II errors (false negatives and false positives) that would be rampant if we were to always predict "ham" in the email example.

Cross validation

How do we decide what level of complexity our algorithm should be, especially when we can't see out-of-sample?

The answer is we choose values of the **tuning parameters** that maximize out-of-sample prediction

For example:

- **Decision Trees**: the maximum depth of the tree or the min. number of observations within leaves
- **Regression penalization**: the λ that comes in front of LASSO, Ridge, and elastic net regularization[1]
  - LASSO:

$$\min_{\beta_1,\ldots,\beta_K} \frac{1}{N}\left(\sum_{i=1}^{N}\left(y_i - \sum_{i=1}^{K}\beta_k X_i^k\right)\right)^2 + \lambda\left|\sum_{i=1}^{K}\beta^k\right|_1$$

- There are many, many more!

[1] This will make more sense later.

Splitting the sample

To peform cross-validation, one needs to split the sample. There are differing opinions:

Camp A ("Holdout")

1. Training data (~70%)
2. Test ("holdout") data (~30%)

Camp B ("Cross-validation")

1. Training data (~60%)
2. Validation data (~20%)
3. Test data (~20%)

This is related to bootstrapping!

Sample is split randomly to how it was generated (e.g. if it's panel data, sample *units*, not observations)

It is ideal to follow the "Cross-validation" camp, but in cases where you don't have many observations (training examples), you may have to go the "Holdout" route.

Test/train/hold-out in Kleinberg et al.

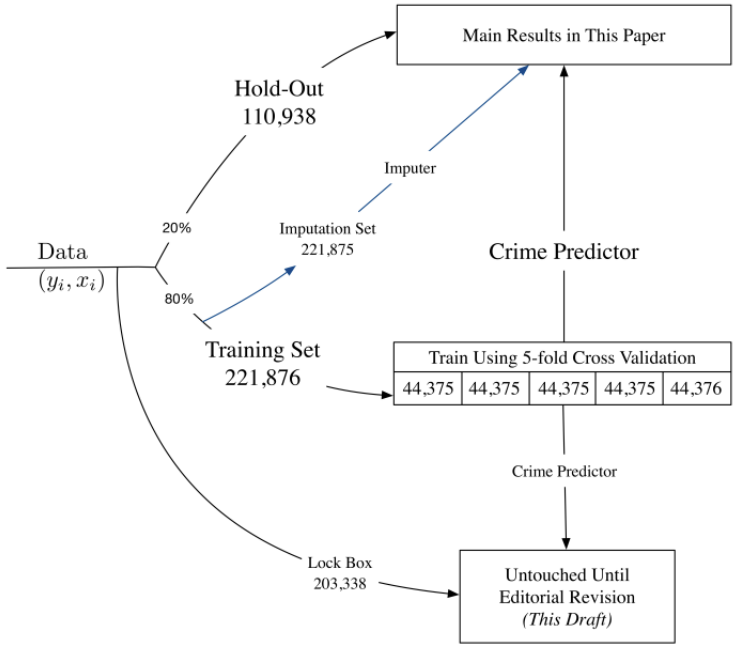
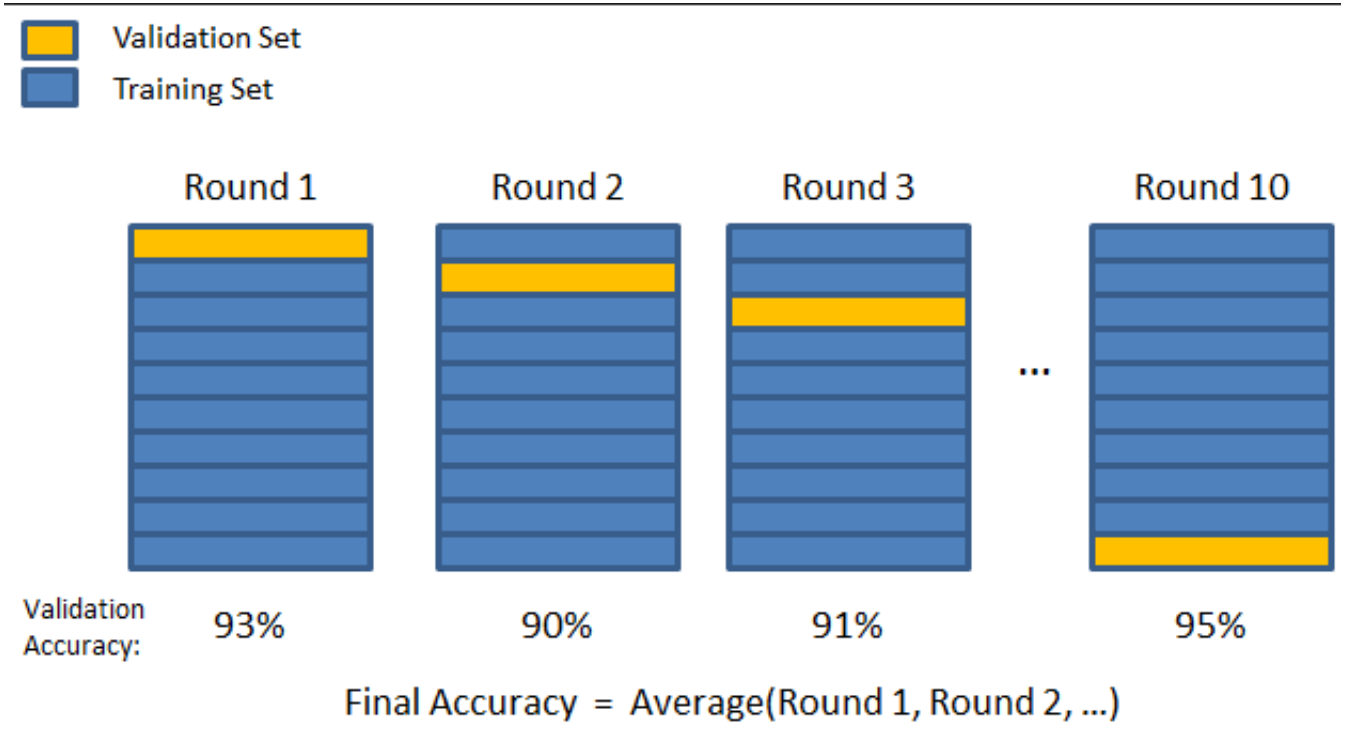


Figure I
Partition of New York City Data (2008-13)
into Data Sets Used for Prediction and Evaluation

This shows the way the New York City data were randomly partitioned to do ML predictions of optimal bail decisions using data of judicial decisions in New York City. Source: Kleinberg et al. (2019) "Human Decisions and Machine Predictions"

k-fold cross-validation

Due to randomness in our data, it may be better to do the cross validation multiple times. To do so, we take the 80% training-plus-validation sample and randomly divide it into the 60/20 components k number of times. Typically k is between 3 and 10. (See graphic below)



In the extreme, one can do *nested* k-fold cross-validation, wherein the test set is shuffled some number of times and the k-fold CV is repeated each time. So we would end up with "3k" fold CV, for example.

What next? Try stuff!

- Navigate to kaggle.com
  - Create an account (should take 30 seconds)
  - Validate your email
- Then go to this practice example of Decision Trees by Brendan Cullen
- You'll need to download the Oregon School test and train data from Kaggle
  - Stick them in a working directory for today's demo and read them in,
- Problem Set 6 will feature these data:
  - You'll essentially write me a problem set that tests whether someone knows how to use the `tidymodels` package in R
  - I want you to practice `tidymodels` and documenting instructions for something you're learning in real time
- There may be some bumps to using this tutorial! Power through and learn by doing!
  - You may need to install some packages (e.g. `rio`, `skimr`, etc.)

Next lecture: Decision Trees and Judicial Decisions