Big Data and Economics

Introduction to Machine Learning

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

Table of contents

Prologue

Prologue

- 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?

Hack-a-thon

Questions

Attribution

These slides are largely adapted from Tyler Ransom's graduate course at the University of Oklahoma

- He gets more into the weeds than I do here
- Give it a look if you think this stuff is cool!

What is Machine Learning?

What is Machine Learning?

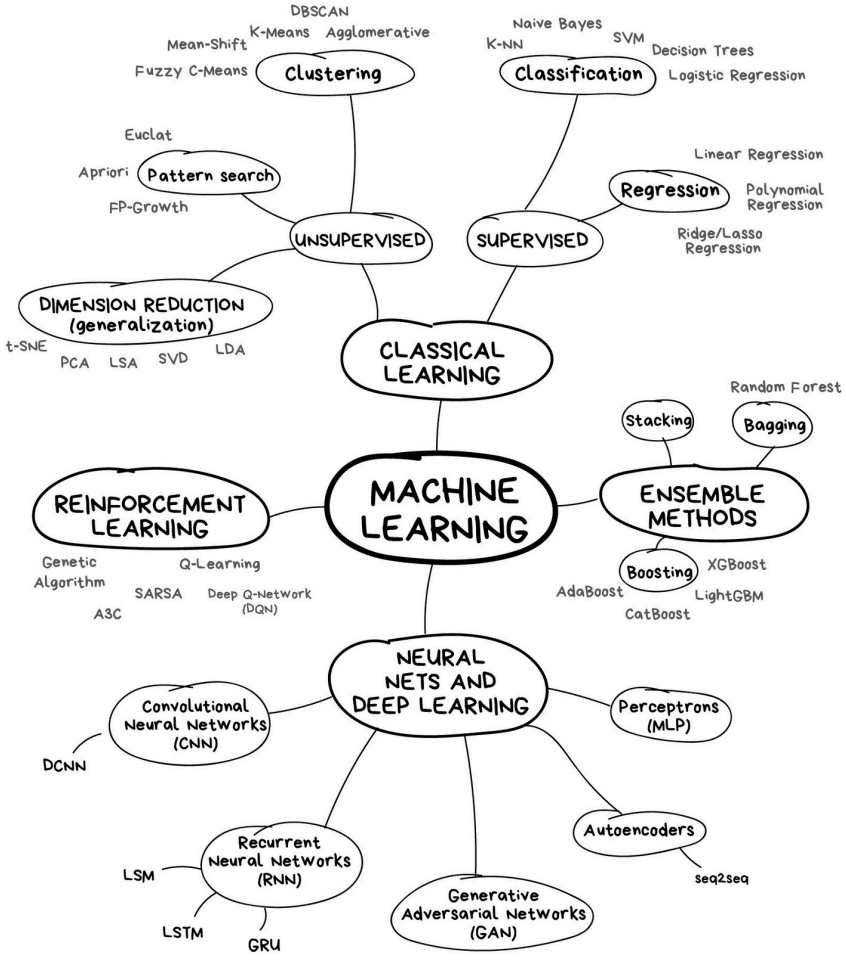
ML: Allowing computers to learn for themselves without being explicitly programmed

- **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

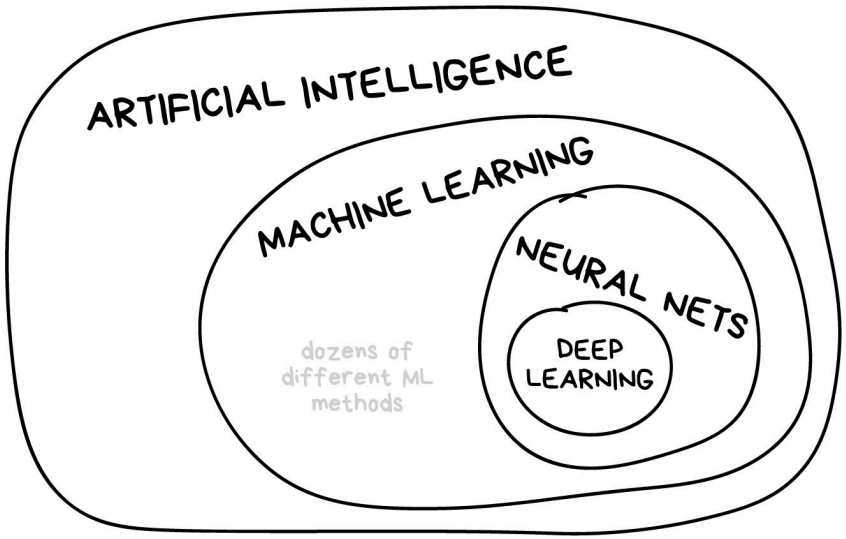


Map of Machine Learning from Motaz Saad. We'll do trees, forests, and penalization!

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

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?
- 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!

Primary statistical concern of 'metrics

- A 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 (almost) no formal 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

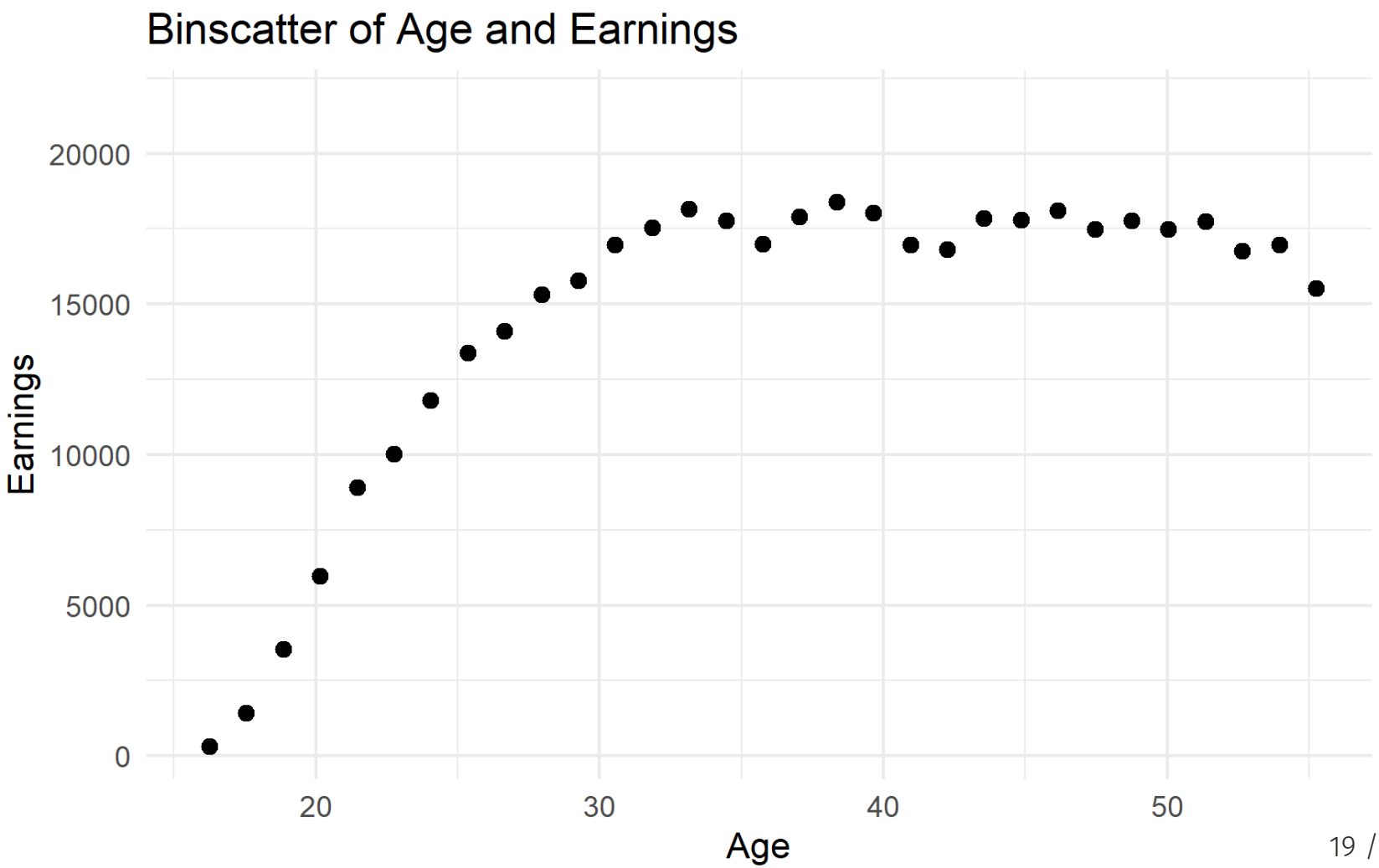
- In contrast, the goal of machine learning is to come up with the best possible out-of-sample prediction
    - Or the primary concern of machine learning being $\hat{y}$
- To get this prediction, a lot of effort is spent on validating many possible models
- However, if the world changes in a fundamental 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)

Example: predicting fit

Should I fit a straight line or curve through this bin scatter of age and earnings?



Example: Compare fits

Seems like we probably want some sort of curve

'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

- **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
- **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

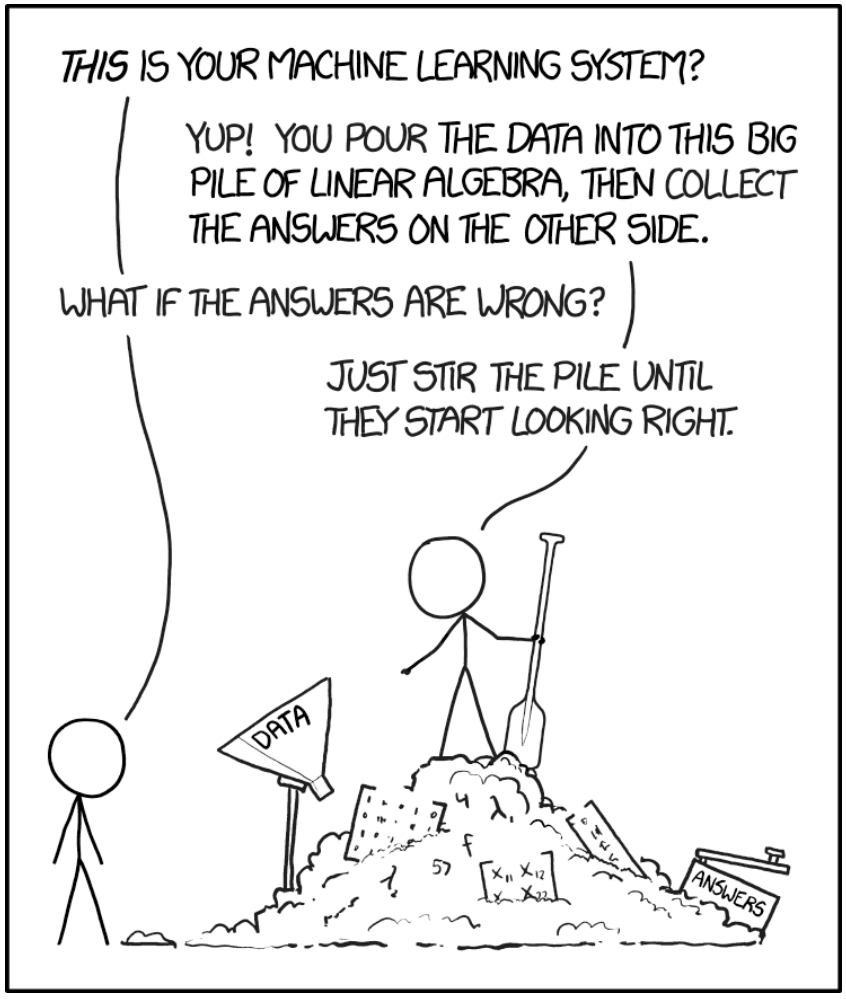
Fundamentals of Machine Learning

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
- **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

Typical ML workflow

1. Split a dataset into a train and test set or training, validation, and test set
    - Do not touch the "test" or "holdout" data until the very end of analysis
2. Train a model on the training set
3. Validate the model on the validation set based by measuring the prediction accuracy
    - Ideally you find ways to cross-validate
4. Tune model parameters to minimize some "loss" function on the validation set
    - Could be to minimize out-of-sample MSE, in-sample MSE, or a mix
5. Take the best model and make predictions using the test data

Lots of little deviations to the above!

Example

- Suppose you want to predict a person's earnings using Current Population Survey data
- You have a large number of relevant variables
- What would you do?
    - You would also want to have a model that you can estimate
    - You probably want to have a model that can detect non-linear relationships (like a USPS handwriting reader)
    - And a model that will predict well out-of-sample

```
## # A tibble: 5 × 12
##      id data_id treat   age  educ black  hisp  marr nodegree   re74   re75
##   <int> <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>  <dbl>  <dbl>
## 1     1 CPS1        0    45    11     0     0     1        1 21517. 25244.
## 2     2 CPS1        0    21    14     0     0     0        0  3176.  5853.
## 3     3 CPS1        0    38    12     0     0     1        0 23039. 25131.
## 4     4 CPS1        0    48     6     0     0     1        1 24994. 25244.
## 5     5 CPS1        0    18     8     0     0     1        1  1669. 10728.
## # ℹ 1 more variable: re78 <dbl>
```

Test vs. Train

```
set.seed(1)
cps_split ← rsample::initial_split(cps_mixtape, prop = 0.5)
cps_train ← rsample::training(cps_split)
cps_test ← rsample::testing(cps_split)
```

- Above I split the sample into a training and test set using syntax from **rsample**
- The training set is used to estimate the model
- The test set is used to see how well the model predicts out-of-sample
- All these functions do is create two random subsamples and reskins them

Option 1: separate dummies for people

```
dummies ← feols(re74 ~ 0 | id, data = cps_train)
cps_train←mutate(cps_train,
    pred_re74_fe = predict(dummies, newdata = cps_train), # predict earnings
    resid_re74_fe= residuals(dummies, newdata = cps_train)) %>%
  dplyr::select(id,educ,re74,pred_re74_fe,resid_re74_fe)
head(cps_train)
```

```
## # A tibble: 6 × 5
##      id  educ   re74 pred_re74_fe resid_re74_fe
##   <int> <dbl>  <dbl>        <dbl>         <dbl>
## 1  1017    14     0            0              0
## 2  8004    12 25862.       25862.             0
## 3  4775    12 25849.       25849.             0
## 4 10369    17 21552.       21552.             0
## 5 13218     6     0            0              0
## 6  9725    12 25862.       25862.             0
```

- What you get is a separate adulthood earnings prediction for every single person
  - Perfect in-sample prediction!

Option 1: new person from test?

```
cps_test ← mutate(cps_test,
    pred_re74_fe=predict(dummies,newdata=cps_test),
    resid_re74_fe=re74-pred_re74_fe) %>%
  dplyr::select(id,educ,re74,pred_re74_fe,resid_re74_fe)
head(cps_test,5)
```

```
## # A tibble: 5 × 5
##      id  educ   re74 pred_re74_fe resid_re74_fe
##   <int> <dbl>  <dbl>        <dbl>         <dbl>
## 1     1    11 21517.           NA            NA
## 2     2    14  3176.           NA            NA
## 3     4     6 24994.           NA            NA
## 4     5     8  1669.           NA            NA
## 5     6    11 16366.           NA            NA
```

- But what to do when given a new person that's not in the sample?
- This 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 predictionton has **high variance** (i.e. the algorithm thinks random noise is something that is important to the model)

Option 2: as a function of education

- Let's use education (highest grade completed) as a predictor

```
education ← feols(re74 ~ educ, data = cps_train)
cps_train ← mutate(cps_train,
    pred_re74_educ = predict(education, newdata = cps_train),
    resid_re75_educ=residuals(education,newdata = cps_train)) %>%
  dplyr::select(id,re74,educ,pred_re74_fe,pred_re74_educ)
head(cps_train)
```

```
## # A tibble: 6 × 5
##      id   re74  educ pred_re74_fe pred_re74_educ
##   <int>  <dbl> <dbl>        <dbl>          <dbl>
## 1  1017     0     14            0          14718.
## 2  8004 25862.    12        25862.         14031.
## 3  4775 25849.    12        25849.         14031.
## 4 10369 21552.    17        21552.         15749.
## 5 13218     0      6            0          11969.
## 6  9725 25862.    12        25862.         14031.
```

- The in sample fit is pretty bad -- it returns the average earnings for a person of a given education level

Option 2: add a new person

But if we add a new person, then it can make a prediction, just inaccurately

```
cps_test ← mutate(cps_test,
    pred_re74_educ = predict(education, newdata = cps_test),
      resid_re75_educ=residuals(education,newdata = cps_test)) %>%
  dplyr::select(id,educ,re74,educ,pred_re74_fe,pred_re74_educ)
head(cps_test)
```

```
## # A tibble: 6 × 5
##      id  educ   re74 pred_re74_fe pred_re74_educ
##   <int> <dbl>  <dbl>        <dbl>          <dbl>
## 1     1    11 21517.           NA         13687.
## 2     2    14  3176.           NA         14718.
## 3     4     6 24994.           NA         11969.
## 4     5     8  1669.           NA         12656.
## 5     6    11 16366.           NA         13687.
## 6    10    12 25862.           NA         14031.
```

- Out of sample it can return a value, but it is way off
- This algorithm will result in **underfitting** because the functional form and variables used for prediction are too simple
- We say that this prediction has **high bias** (i.e. the algorithm does not think enough variation is important to the model)

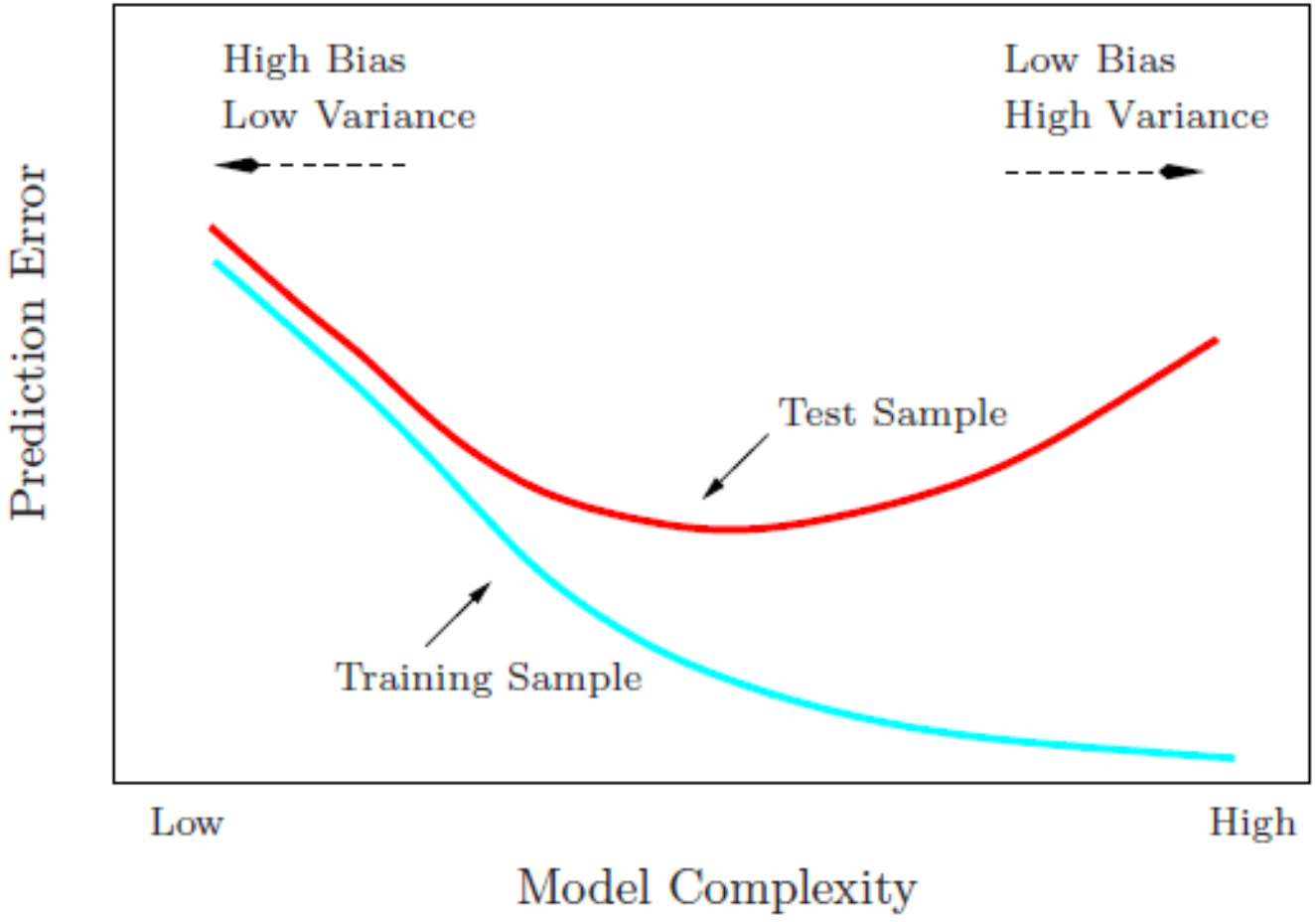
Bias-variance tradeoff

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:



Prediction accuracy of continuous y

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

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

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

where N is the sample size

```
bind_rows(list('In/Train'=cps_train, 'Out/Test'=cps_test),.id='sample') %>%
  group_by(sample) %>%
  summarise(MSE Fixed Effect = mean((re74 - pred_re74_fe)^2),
    MSE Education = mean((re74 - pred_re74_educ)^2),
    Root MSE Fixed Effect = sqrt(MSE Fixed Effect),
    Root MSE Education = sqrt(MSE Education)) %>%
  knitr::kable(digits=2,format.args=list(big.mark=","))
```

sample	MSE Fixed Effect	MSE Education	Root MSE Fixed Effect	Root MSE Education
In/Train	0	91,084,172	0	9,543.80
Out/Test	NA	90,494,387	NA	9,512.85

Prediction accuracy of binary y

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

	$\hat{y}$	
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?

Prediction accuracy of binary y

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

	$\hat{y}$	
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[1]

[1] The most common way to quantify this tradeoff is `(F1)` score. Check Tyler Ransom's slides for more on different metrics in the confusion matrix.

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[2]

[2] 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
- **Random Forests**: the same as decision trees plus the number of trees (bootstrap samples) in the forest
- **Regression penalization**: the λ that comes in front of LASSO, Ridge, and elastic net regularization
- There are many, many more!

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%)

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.

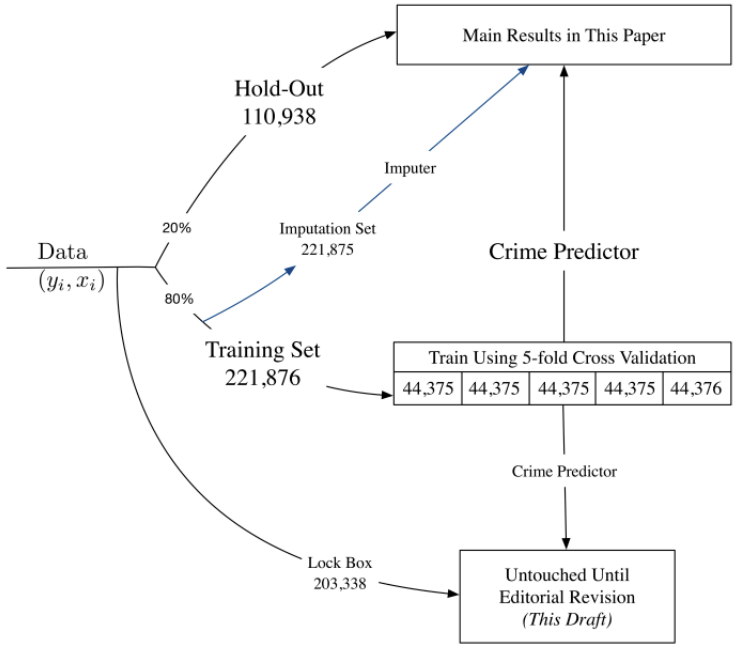
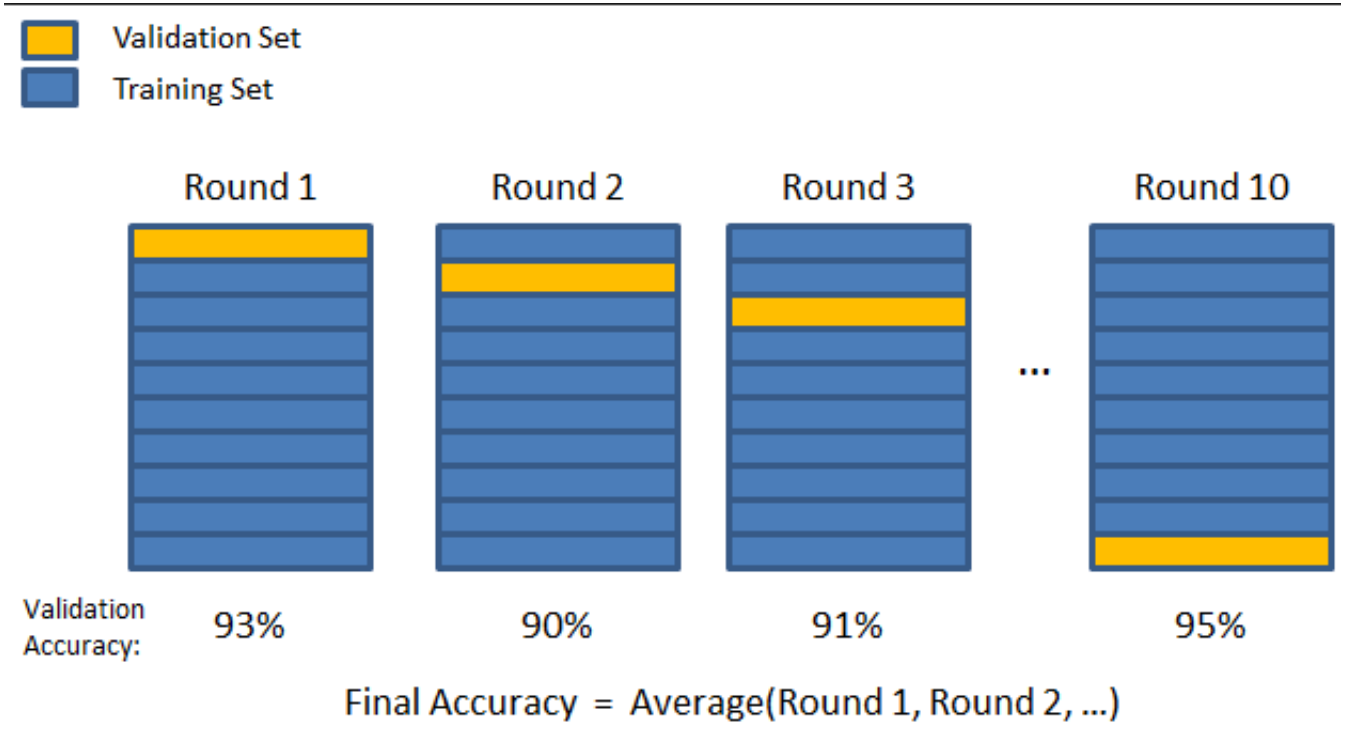


Figure I
Partition of New York City Data (2008-13)
into Data Sets Used for Prediction and Evaluation

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

- To avoid overfitting one particular sample, it is usually 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)



Intro to Statistical Learning Demo

- Navigate to https://tinyurl.com/islrecon368, a website with interactive demos of machine learning resampling from the textbook Introduction to Statistical Learning
    - Before starting use `install.packages('tidymodels')` and `install.packages('ISLR')` in R to install the necessary packages, then work through the codeblocks
    - This demo uses simple datasets to use resampling to decide how many polynomials to use in a regression model
    - It also teaches you how to use the **tidymodels** package, which is a tidyverse-friendly way to do machine learning
- Work through implementing different resampling methods in R like test/train, k-fold cross-validation, and bootstrapping

Comprehension questions

Demo: https://tinyurl.com/islrecon368

1. Plot the relationship between mpg and horsepower as shown below. Would a straight-line or curved-line fit better? Why?

```
ggplot(Auto,aes(x=mpg,y=horsepower)) + geom_point()
```

5.1 (skip 5.2)

1. What is `set.seed()`? Why is it important to use it when splitting the sample?
2. Why can't you test your model on the training data?

5.3

1. Why did you replace degree with the function `tune()`?
2. How does k-fold cross-validation allow you to tune your model?
3. Why does RSME (R^2) start to increase (decrease) after two polynomials?

5.4

1. Use `View(broom::tidy(Portfolio_boots$splits[[1]]))` to see the first bootstrap sample. What do you think "Analysis" and "Assessment" can be used for?
2. Try using bootstrapping to return standard deviation for the polynomial model of auto. *Hint: Use the* `poly_wf`.

Next lecture: Decision Trees and Judicial Decisions