Basic Machine Learning Concepts

Itamar Caspi

March 10, 2019 (updated: 2019-03-11)

Replicating this presentation

R packages used to produce this presentation

```
library(tidyverse) # for data wrangling and plotting
library(svglite) # for better looking plots
library(kableExtra) # for better looking tables
library(RefManageR) # for referencing
library(truncnorm) # for drawing from a truncated normal distribution
library(tidymodels) # for modelling
library(knitr) # for presenting tables
library(mlbench) # for the Boston Housing data
library(ggdag) # for plotting DAGs
```

If you are missing a package, run the following command

```
install.packages("package_name")
```

Alernatively, you can just use the pacman package that loads and installs packages:

```
if (!require("pacman")) install.packages("pacman")

pacman::p_load(tidyvers, svglite, kabelExtra, RefManageR,
               truncnorm, tidymodels, knitr, mlbench, ggdag)
```

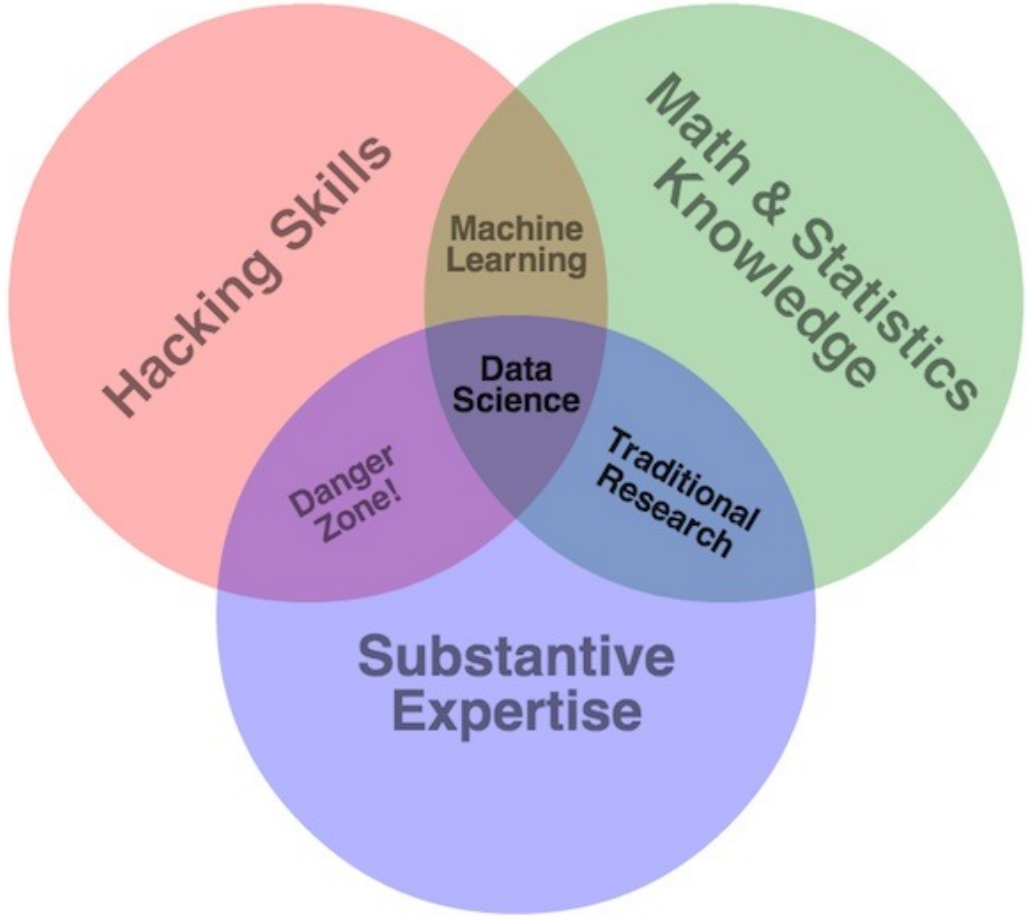
First things first: "Big Data"

> *"A billion years ago modern homo sapiens emerged. A billion minutes ago, Christianity began. A billion seconds ago, the IBM PC was released. A billion Google searches ago ... was this morning."*
>
> Hal Varian (2013)

The 4 Vs of big data:

- Volume - Scale of data.
- Velocity - Analysis of streaming data.
- Variety - Different forms of data.
- Veracity - Uncertainty of data.

"Data Science"



[*] Hacking ≈ coding

Outline

1. What is ML?
2. The problem of overfitting
3. Too complext? Regularize!
4. Putting it All Together
5. What's in it for economists?

What is ML?

So, what is ML?

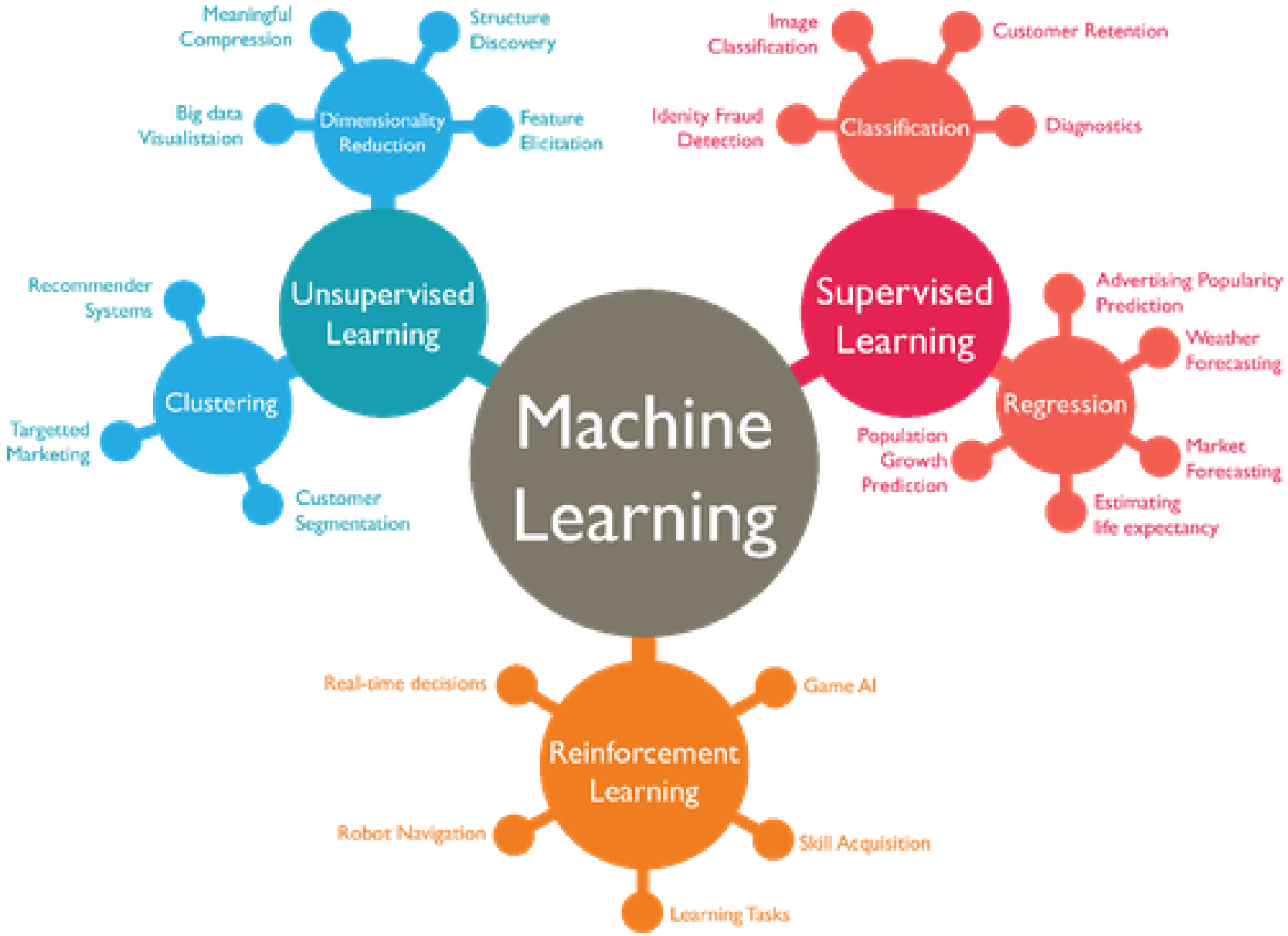
A concise definition by Athey (2018):

> "...[M]achine learning is a field that develops algorithms designed to be applied to datasets, with the main areas of focus being prediction (regression), classification, and clustering or grouping tasks."

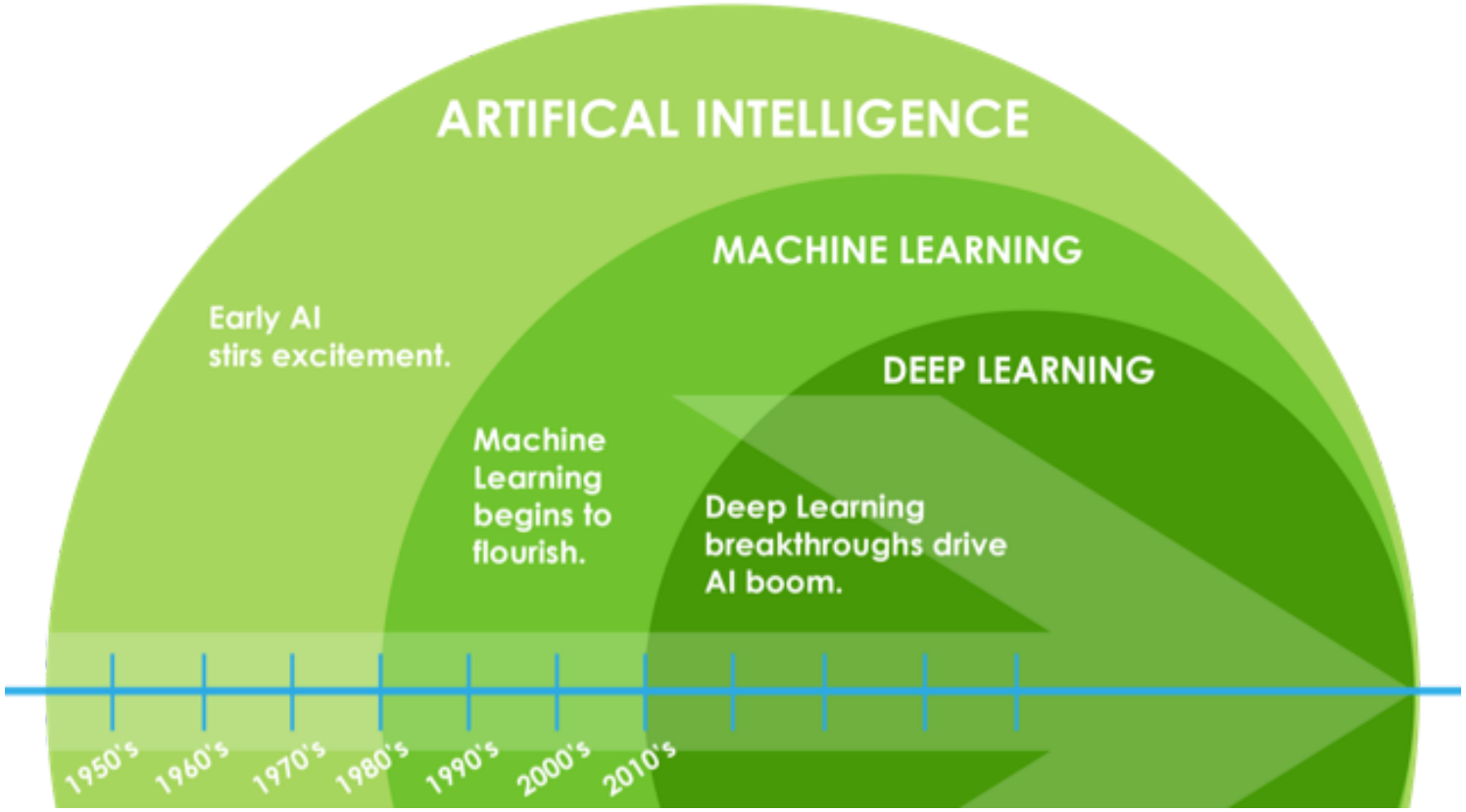
Specifically, there are three broad classifications of ML problems:

- supervised learning.
- unsupervised learning.
- reinforcement learning.

> Most of the hype you hear about in recent years relates to supervised learning, and in particular, deep learning.


Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualisation
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Identity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Population Growth Prediction
Market Forecasting
Estimating life expectancy
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

An aside: ML and artificial intelligence (AI)



Unsupervised learning

In *unsupervised* learning, the goal is to divide high-dimensional data into clusters that are **similar** in their set of features (X).

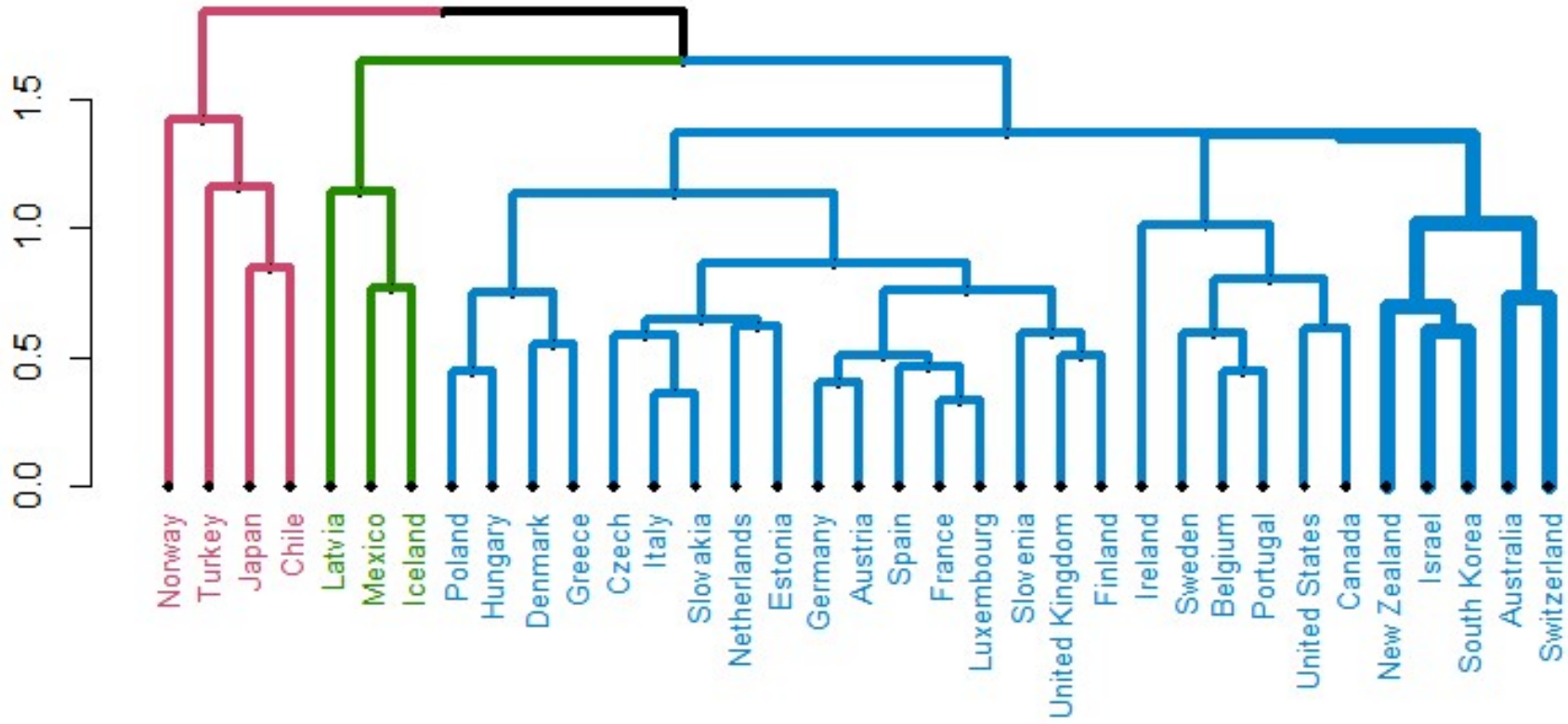
Examples of algorithms:

- principal component analysis
- k-means clustering
- Latent Dirichlet Allocation model

Applications:

- image recognition
- cluster analysis
- topic modelling

Example: Clustering OECD Inflation Rates



Source: Baudot-Trajtenberg and Caspi (2018).

Reinforcement learning (RL)

A definition by Sutton and Barto (2018):

> *"Reinforcement learning is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them."*

Prominent examples:

- Games (e.g., Chess, AlphaGo).
- Self driving cars.

Supervised learning

Consider the following data generating process (DGP):

$$Y = f(\boldsymbol{X}) + \epsilon$$

where Y is the outcome variable, $\boldsymbol{X}$ is a $1 \times p$ vector of "features", and ϵ is the irreducible error.

- **Training set** ("in-sample"): $\{(x_i, y_i)\}_{i=1}^{n}$
- **Test set** ("out-of-sample"): $\{(x_i, y_i)\}_{i=n+1}^{m}$

> Typical assumptions: (1) independent observations; (2) stable DGP across training *and* test sets.

The goal of supervised learning

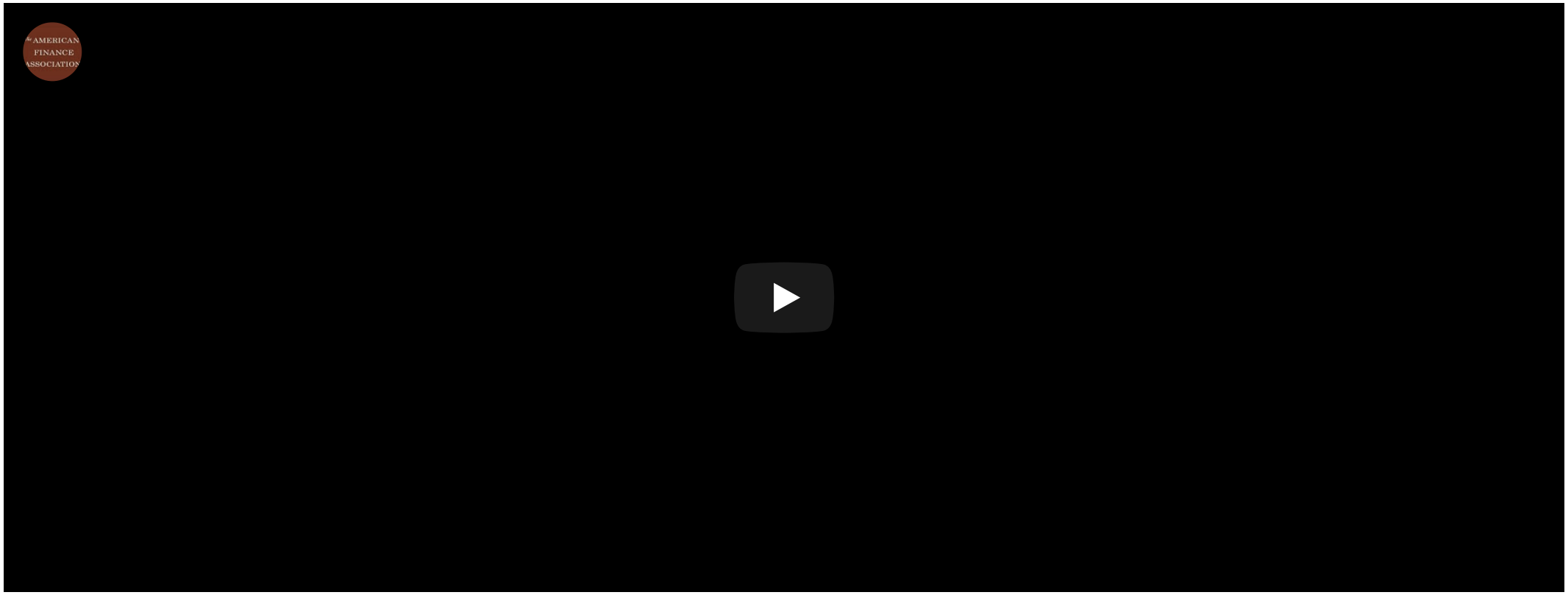
Use a labelled test set (X and Y are known) to construct $\hat{f}(X)$ such that it *generalizes* to unseen test set (only X is known).

EXAMPLE: Consider the task of spam detection:

ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
ham	Ok lar... Joking wif u oni...
spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
ham	U dun say so early hor... U c already then say...
ham	Nah I don't think he goes to usf, he lives around here though
spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, å£1.50 to rcv
ham	Even my brother is not like to speak with me. They treat me like aids patent.
ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune
spam	WINNER!! As a valued network customer you have been selected to receivea å£900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.
spam	Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on 08002986030

In this case, $Y = \{spam, ham\}$, and X is the email text.

Traditional vs. modern approach to supervised learning



Traditional vs. modern approach to supervised learning

- Traditional: rules based, e.g., define dictionaries of "good" and "bad" words, and use it to classify text.
- Modern: learn from data, e.g., label text as "good" or "bad" and let the model estimate rules from (training) data.

More real world applications of ML

task	**outcome** (Y)	**features** (X)
credit score	probability of default	loan history, payment record...
fraud detection	fraud / no fraud	transaction history, timing, amount...
voice recognition	word	recordings
sentiment analysis	good / bad review	text
image classification	cat / not cat	image
overdraft prediction	yes / no	bank-account history
customer churn prediction	churn / no churn	customer features

> A rather mind blowing example: Amazon's "Anticipatory Package Shipping" patent (December 2013): Imagine Amazon's algorithms reaching such levels of accuracy, casing it to change its business model from shopping-then-shipping to shipping-then-shopping!

Supervised learning algorithms

ML comes with a rich set of parametric and non-parametric prediction algorithms (approximate year of discovery in parenthesis):

- Linear and logistic regression (1805, 1958).
- Decision and regression trees (1984).
- K-Nearest neighbors (1967).
- Support vector machines (1990s).
- Neural networks (1940s, 1970s, 1980s, 1990s).
- Simulation methods (Random forests (2001), bagging (2001), boosting (1990)).
- etc.

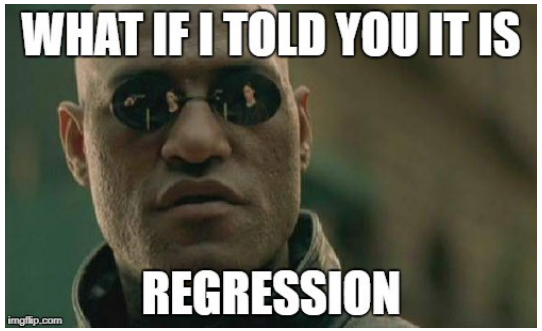
So, why now?

- ML methods are data-hungry and computationally expensive. Hence,

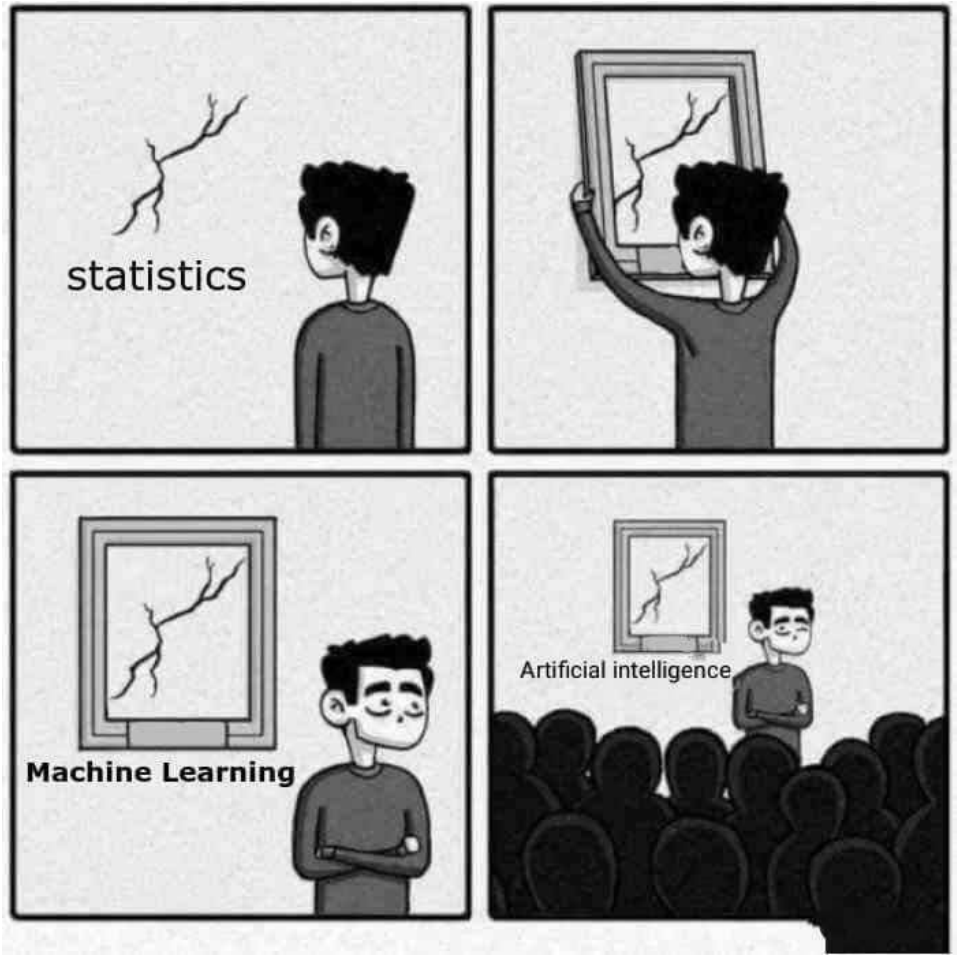
$$\textbf{big data} + \textbf{computational advancements} = \textbf{the rise of ML}$$

- Nevertheless,

> *"[S]upervised learning [...] may involve high dimensions, non-linearities, binary variables, etc., but at the end of the day it's still just regression."*
>
> — **Francis X. Diebold**



Wait, is ML just glorified statistics?



The "two cultures" (Breiman, 2001):

- Statisticians assume a data generating process and try to learn about it using data (parameters, confidence intervals, assumptions.)
- Computer scientists treat the data mechanism as unknown and try to predict or classify with the most accuracy.

Overfitting

Prediction accuracy

Before we define overfitting, we need to be more explicit about what we mean by "good prediction."

- Let (x^0, y^0) denote a single realization from the (unseen) test set.
- Define a **loss function** L in terms of predictions $\hat{y}^0 = \hat{f}(x^0)$ and the "ground truth" y^0, where $\hat{f}$ is estimated using the *training* set.
- Examples
  - squared error (SE): $L(\hat{y}^0, y^0) = (y^0 - \hat{f}(x^0))^2$
  - absolute error (AE): $L(\hat{y}^0, y^0) = |y^0 - \hat{f}(x^0)|$
- There are other possible forms of loss function (e.g., in classification, as the probability of misclassifying a case, or in terms of economic costs.)

The bias-variance decomposition

Under a **squared error loss function**, an optimal predictive model is one that minimizes the *expected* squared prediction error.

It can be shown that if the true model is $Y = f(X) + \epsilon$, then

$$\begin{aligned}
\mathbb{E}\left[\mathrm{SE}^0\right] &= \mathbb{E}\left[(y^0 - \hat{f}\,(x^0))^2\right] \\
&= \underbrace{\left(\mathbb{E}(\hat{f}\,(x^0)) - f(x^0)\right)^2}_{\text{bias}^2} + \underbrace{\mathbb{E}\left[\hat{f}\,(x^0) - \mathbb{E}(\hat{f}\,(x^0))\right]^2}_{\text{variance}} + \underbrace{\mathbb{E}\left[y^0 - f(x^0)\right]^2}_{\text{irreducible error}} \\
&= \underbrace{\mathrm{Bias}^2 + \mathbb{V}[\hat{f}\,(x^0)]}_{\text{reducible error}} + \sigma_\epsilon^2
\end{aligned}$$

where the expectation is over the training set *and* (x^0, y^0).

Intuition behind the bias variance trade-off

Imagine you are a teaching assistant grading exams. You grade the first exam. What is your best prediction of the next exam's grade?

- the first test score is an unbiased estimator of the mean grade.
- but it is extremely variable.
- any solution?

Lets simulate it!

Exam grade prediction simulation

Let's Draw 1000 grade duplets from the following truncated normal distribution

$$g_i \sim truncN(\mu = 75, \sigma = 15, a = 0, b = 100), \quad i = 1, 2$$

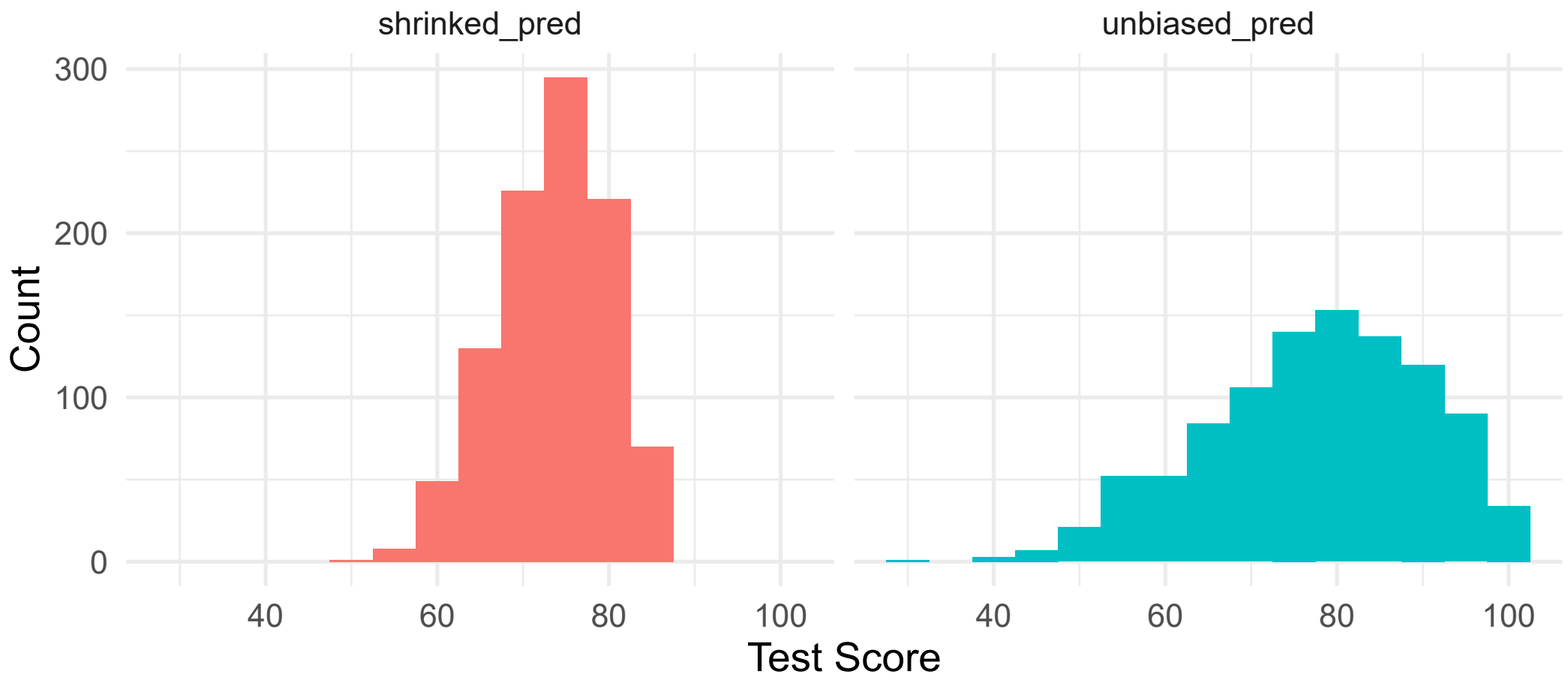
Next, calculate two types of predictions

- `unbiased_pred` is the first exam's grade.
- `shrinked_pred` is an average of the previous grade and a *prior* mean grade of 70.

Here a small sample from our simulated table:

attempt	grade1	grade2	unbiased_pred	shrinked_pred
124	87	80	87	78.5
446	59	84	59	64.5
685	47	73	47	58.5
969	44	76	44	57.0
413	83	68	83	76.5

The distribution of predictions



The MSE of grade predictions

unbiased_MSE	shrinked_MSE
324.192	217.9963

Hence, the shrinked prediction turns out to be better (in the sense of MSE) then the unbiased one!

QUESTION: Is this a general result? Why?

Regressions and the bias-variance trade-off

Consider the following hypothetical DGP:

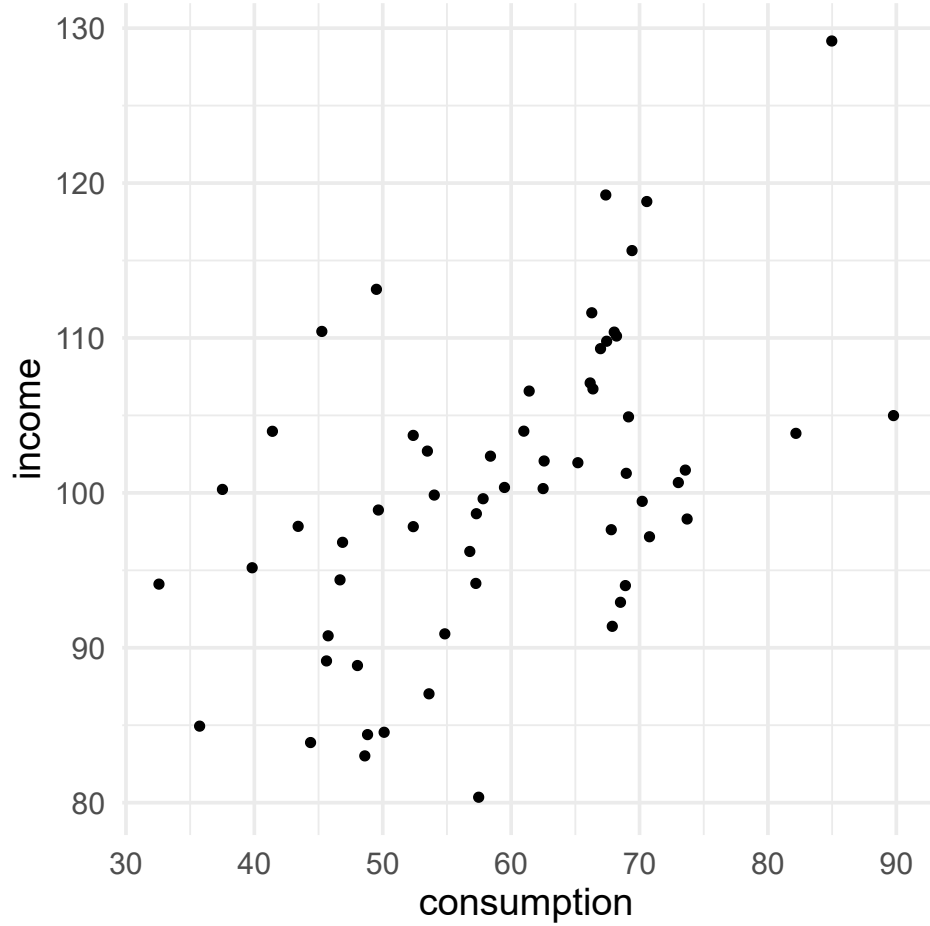
$$consumption_i = \beta_0 + \beta_1 \times income_i + \varepsilon_i$$

```
set.seed(1505) # for replicating the simulation

df <- crossing(economist = c("A", "B", "C"),
               obs = 1:20) %>%
  mutate(economist = as.factor(economist)) %>%
  mutate(income = rnorm(n(), mean = 100, sd = 10)) %>%
  mutate(consumption = 10 + 0.5 * income + rnorm(n(), sd = 10))
```

Scatterplot of the data

```
df %>%
  ggplot(aes(y = income,
             x = consumption)) +
  geom_point()
```

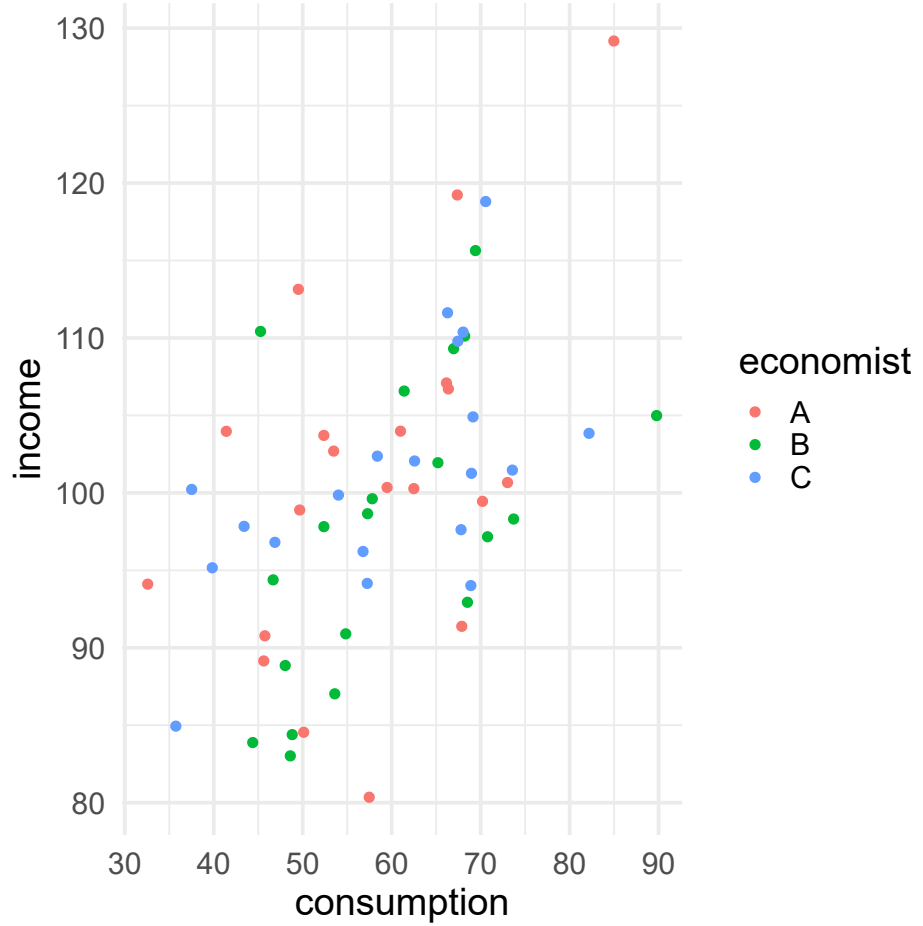


Split the sample between 3 economists

```
df %>%
  ggplot(aes(x = consumption,
             y = income,
             color = economist)) +
  geom_point()
```

```
knitr::kable(sample_n(df,6), format = "html")
```

economist	obs	income	consumption
A	4	98.89330	49.65828
C	14	84.94055	35.73597
C	3	97.83834	43.40877
A	10	102.69686	53.47575
C	8	109.78834	67.43844
A	12	103.71018	52.37188

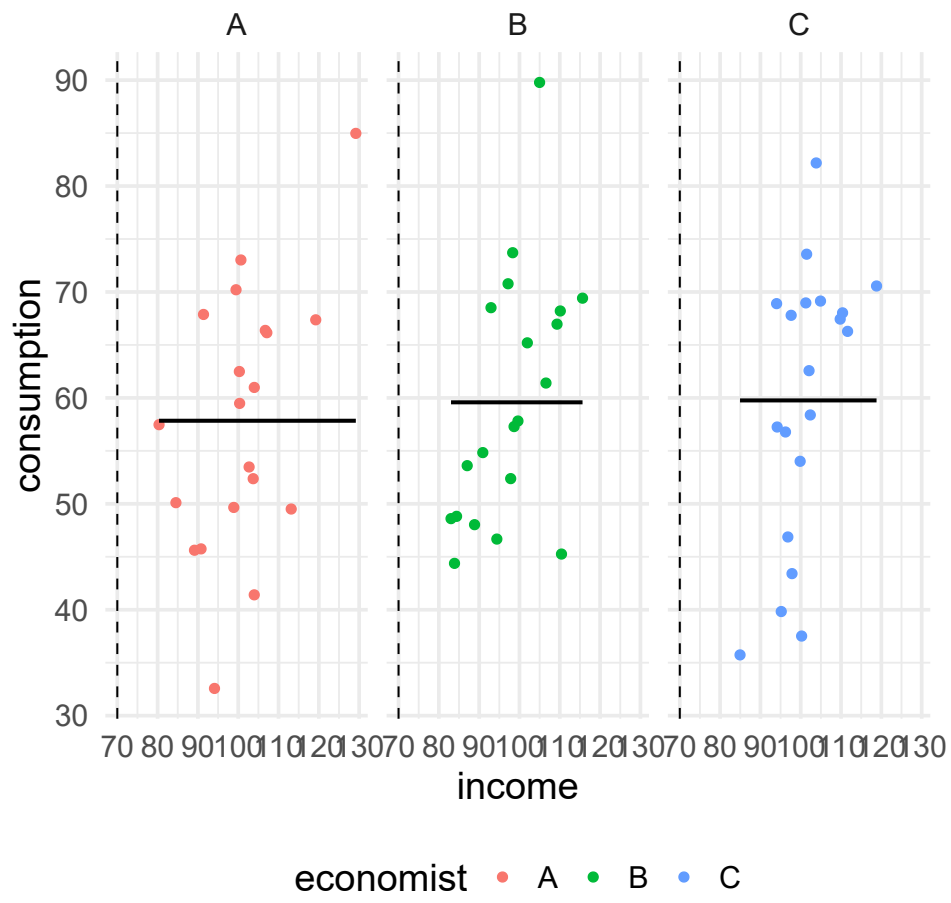


Underffiting: high bias, low variance

The model: unconditional mean

$$Y_i = \beta_0 + \varepsilon_i$$

```
df %>%
  ggplot(aes(y = consumption,
             x = income,
             color = economist)) +
  geom_point() +
  geom_smooth(method = lm,
              formula = y ~ 1,
              se = FALSE,
              color = "black") +
  facet_wrap(~ economist) +
  geom_vline(xintercept = 70, linetype = "dashed") +
  theme(legend.position = "bottom")
```

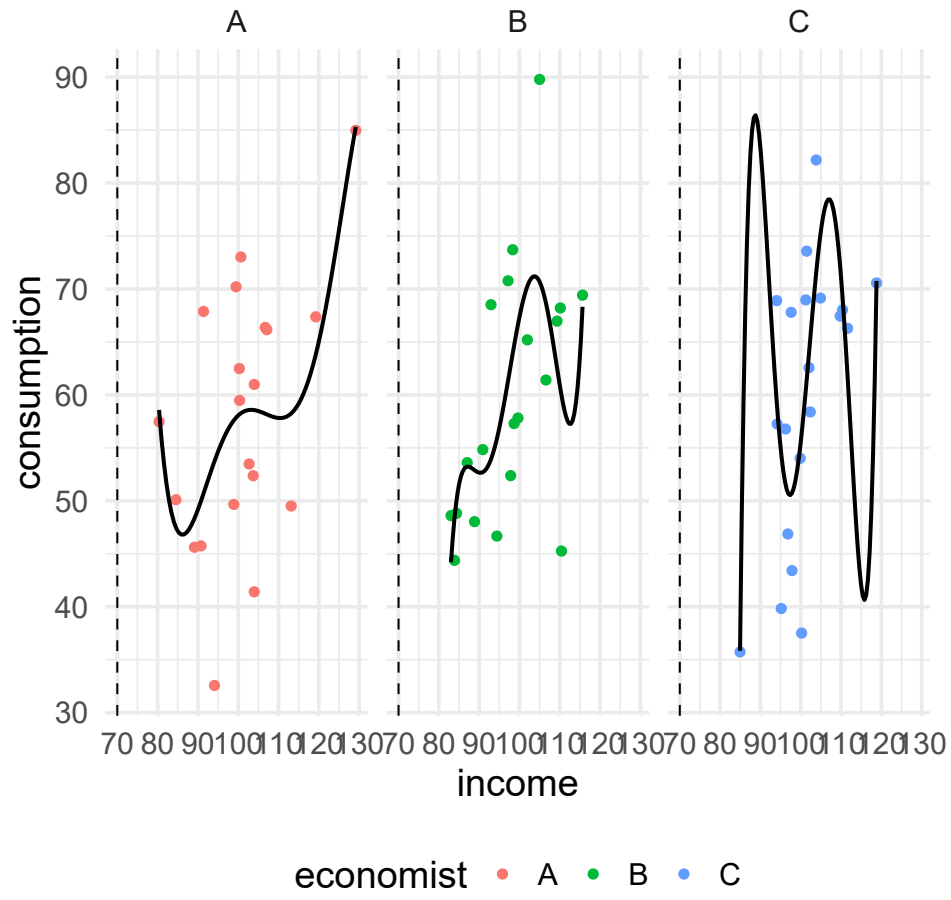


Overfitting: low Bias, high variance

The model: high-degree polynomial

$$Y_i = \beta_0 + \sum_{j=1}^{\lambda} \beta_j X_i^{\lambda} + \varepsilon_i$$

```
df %>%
  ggplot(aes(y = consumption,
             x = income,
             color = economist)) +
  geom_point() +
  geom_smooth(method = lm,
              formula = y ~ poly(x,5),
              se = FALSE,
              color = "black") +
  facet_wrap(~ economist) +
  geom_vline(xintercept = 70, linetype = "dashed") +
  theme(legend.position = "bottom")
```

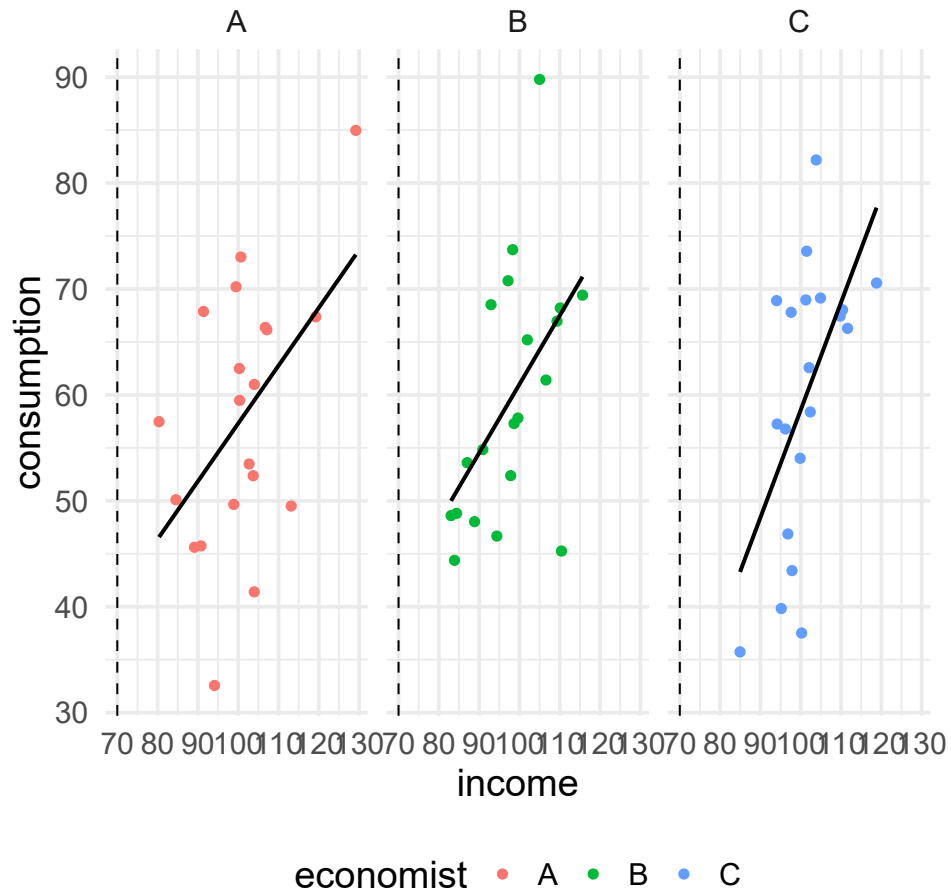


"Justfitting": bias and Variance are just right

The model: linear regression

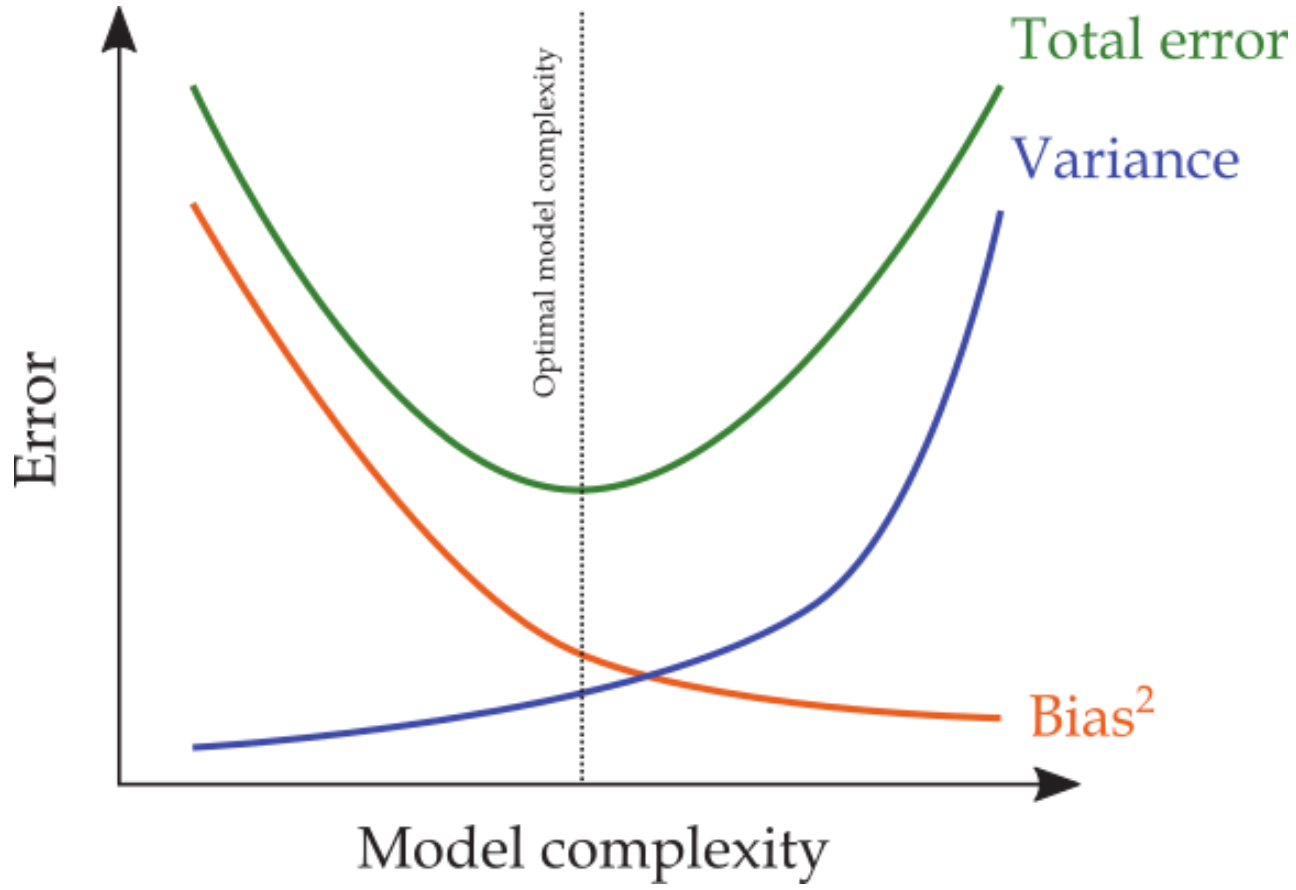
$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

```
df %>%
  ggplot(aes(y = consumption,
             x = income,
             color = economist)) +
  geom_point() +
  geom_smooth(method = lm,
              formula = y ~ x,
              se = FALSE,
              color = "black") +
  facet_wrap(~ economist) +
  geom_vline(xintercept = 70, linetype = "dashed") +
  theme(legend.position = "bottom")
```



The typical bias-variance trade-off in ML

Typically, ML models strive to find levels of bias and variance that are "just right":



When is the Bias-Variance Trade-off Important?

In low-dimensional settings ($n \gg p$)

- overfitting is highly **unlikely**
- training MSE closely approximates test MSE
- conventional tools (e.g., OLS) will perform well on a test set

INTUITION: As $n \to \infty$, insignificant terms will converge to their true value (zero).

In high-dimensional settings ($n \ll p$)

- overfitting is highly **likely**
- training MSE poorly approximates test MSE
- conventional tools tend to overfit

> $n \ll p$ is prevalent in big-data

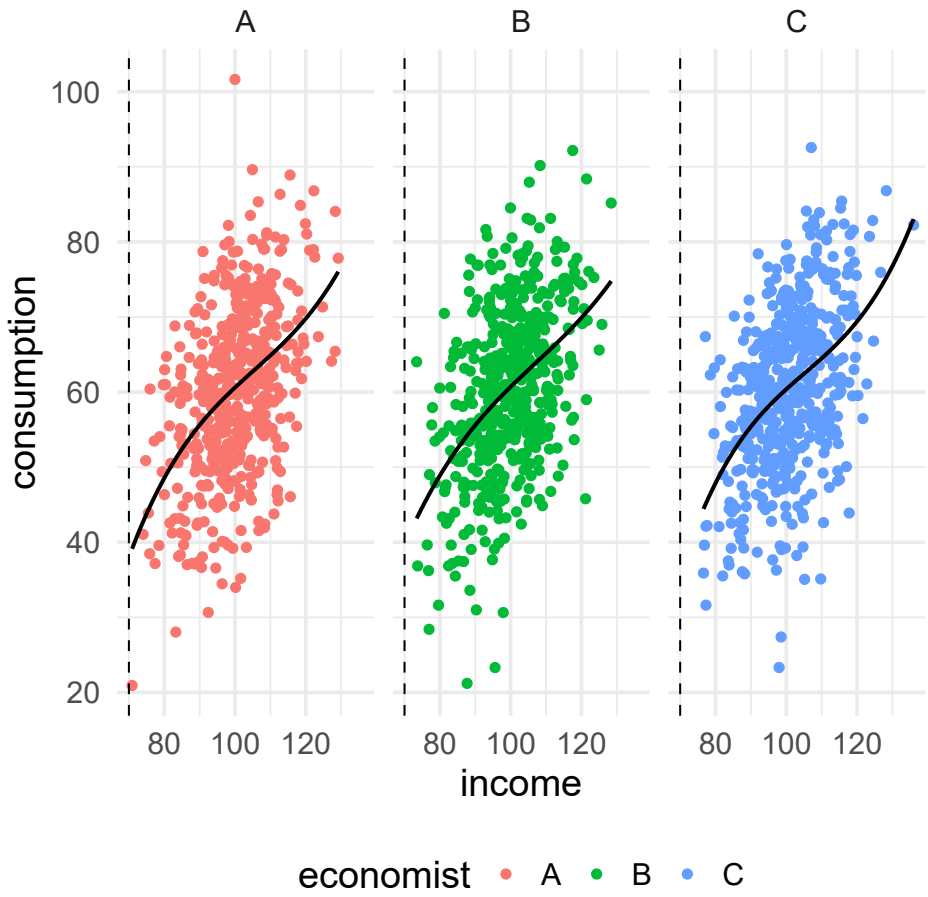
Bias-variance trade-off in low-dimensional settings

The model is a 3rd degree polynomial

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \beta_3 X_i^3 + \varepsilon_i$$

only now, the sample size for each economist increases to $N = 500$.

> **INTUITION:** as $n \to \infty$, $\hat{\beta}_2$ and $\hat{\beta}_3$ converge to their true value, zero.



Regularization

Regularization

- As the *complexity* of our model goes up, it will tend to overfit.
- **Regularization** is the act of penalizing models for their complexity level.

> Regularization typically results in simpler and more accurate (though "wrong") models.

How to penalize overfit?

The test set MSE is *unobservable*. How can we tell if a model overfits?

- Main idea in machine learning: use *less* data!
- Key point: fit the model to a subset of the training set, and **validate** the model using the subset that was *not* used to fit the model.
- How can this "magic" work? Recall the stable DGP assumption.

Validation

Split the sample to three folds: a training set, a validation set and a test set:

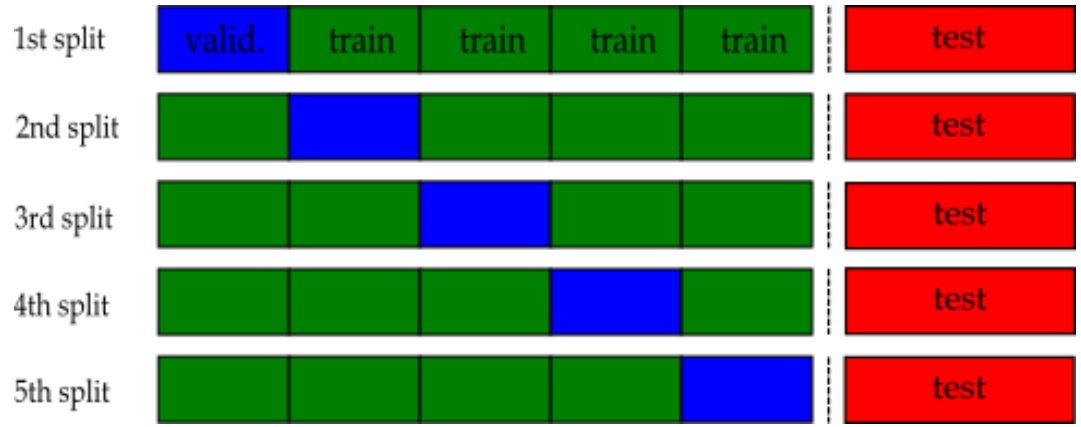
The algorithm:

1. Fit a model to the training set.
2. Use the model to predict outcomes from the validation set.
3. Use the mean of the squared prediction errors to approximate the test-MSE.

CONCERNS: (1) the algorithm might be sensitive to the choice of training and validation set; (2) the algorithm does not use all of the available information.

k-fold cross-validation

Split the training set into k roughly equal-sized parts ($k = 5$ in this example):



Approximate the test-MSE using the mean of k split-MSEs

$$\text{CV-MSE} = \frac{1}{k}\sum_{j=1}^{k}\text{MSE}_j$$

Which model to choose?

- Recall that the test-MSE is unobservable.
- CV-MSE is our best guess.
- CV-MSE is also a function of model complexity.
- Hence, model selection amounts to choosing the complexity level that minimizes CV-MSE.

Sounds familiar?

- In a way, you probably already know this:
  - Adjusted R2.
  - AIC, BIC (time series models).

  The above two measures indirectly take into account the overfitting that may occur due to the complexity of the model (i.e., adding too many covariates or lags).
- In ML we use the data to tune the level of complexity such that it maximizes prediction accuracy.

Putting it All Together

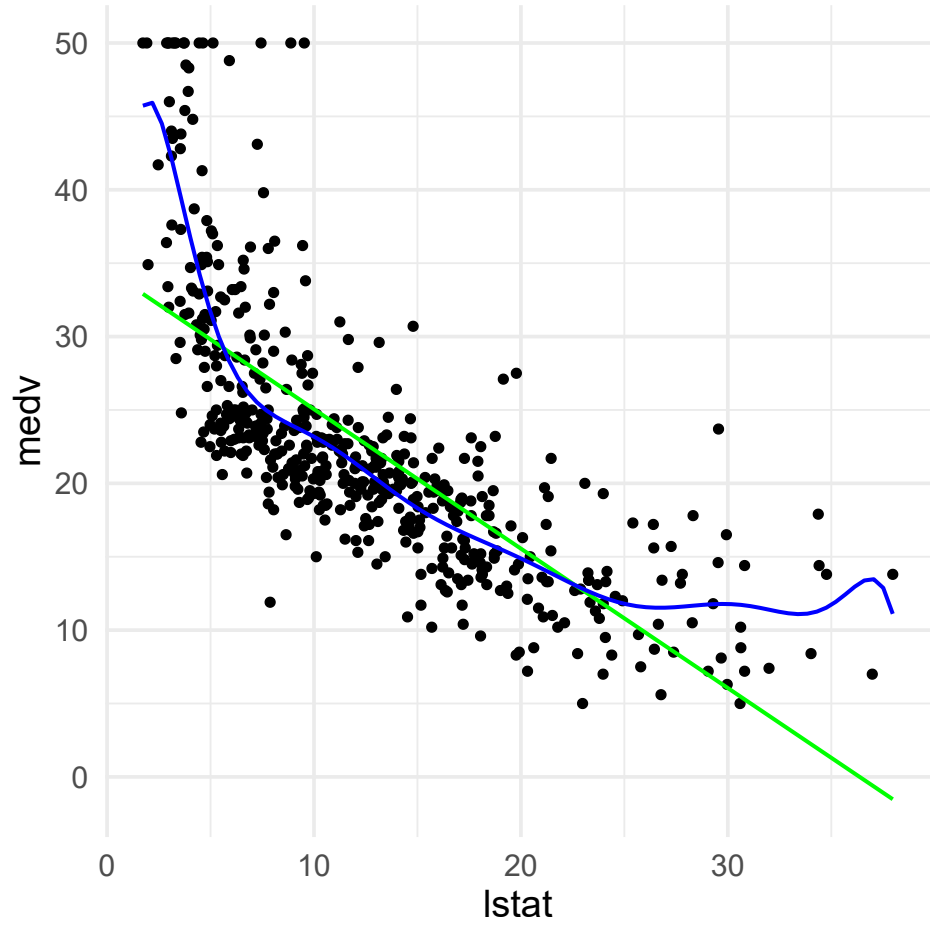
Toy problem: Predicting Boston housing prices

We will use the `BostonHousing`: housing data for 506 census tracts of Boston from the 1970 census (Harrison Jr and Rubinfeld, 1978)

- `medv` (outcome): median value of owner-occupied homes in USD 1000's.
- `lstat`(predictor): percentage of lower status of the population.

OBJECTIVE: Find the best prediction model within the class of polynomial regression.

Examples: in green: linear relation ($\lambda = 1$); in blue, $\lambda = 10$.



Step 1: The train-test split

We will use the `initial_split()`, `training()` and `testing()` functions from the rsample package to perform an initial train-test split

```
set.seed(1203) # for reproducability

df_split <- initial_split(df, prop = 0.75)
df_split
```

```
## <380/126/506>
```

```
training_df <- training(df_split)
testing_df  <- testing(df_split)

head(training_df, 5)
```

```
## # A tibble: 5 x 2
##    medv lstat
##   <dbl> <dbl>
## 1  21.6  9.14
## 2  34.7  4.03
## 3  36.2  5.33
## 4  28.7  5.21
## 5  22.9 12.4
```

Step 2: Prepare 10 folds for cross-validation

We will use the `vfold-cv()` function from the rsample package to split the training set to 10-folds:

```
cv_data <- training_df %>%
  vfold_cv(v = 10) %>%
  mutate(train     = map(splits, ~training(.x)),
         validate  = map(splits, ~testing(.x)))
cv_data
```

```
## #  10-fold cross-validation
## # A tibble: 10 x 4
##    splits           id     train              validate
##  * <list>           <chr>  <list>             <list>
##  1 <split [342/38]> Fold01 <tibble [342 x 2]> <tibble [38 x 2]>
##  2 <split [342/38]> Fold02 <tibble [342 x 2]> <tibble [38 x 2]>
##  3 <split [342/38]> Fold03 <tibble [342 x 2]> <tibble [38 x 2]>
##  4 <split [342/38]> Fold04 <tibble [342 x 2]> <tibble [38 x 2]>
##  5 <split [342/38]> Fold05 <tibble [342 x 2]> <tibble [38 x 2]>
##  6 <split [342/38]> Fold06 <tibble [342 x 2]> <tibble [38 x 2]>
##  7 <split [342/38]> Fold07 <tibble [342 x 2]> <tibble [38 x 2]>
##  8 <split [342/38]> Fold08 <tibble [342 x 2]> <tibble [38 x 2]>
##  9 <split [342/38]> Fold09 <tibble [342 x 2]> <tibble [38 x 2]>
## 10 <split [342/38]> Fold10 <tibble [342 x 2]> <tibble [38 x 2]>
```

Step 3: Set search range for lambda

We need to vary the polynomial degree parameter (λ) when building our models on the train data. In this example, we will set the range between 1 and 10:

```
cv_tune <- cv_data %>%
  crossing(lambda = 1:10)

cv_tune
```

```
## # A tibble: 100 x 5
##    splits           id     train              validate          lambda
##    <list>           <chr>  <list>             <list>             <int>
##  1 <split [342/38]> Fold01 <tibble [342 x 2]> <tibble [38 x 2]>      1
##  2 <split [342/38]> Fold01 <tibble [342 x 2]> <tibble [38 x 2]>      2
##  3 <split [342/38]> Fold01 <tibble [342 x 2]> <tibble [38 x 2]>      3
##  4 <split [342/38]> Fold01 <tibble [342 x 2]> <tibble [38 x 2]>      4
##  5 <split [342/38]> Fold01 <tibble [342 x 2]> <tibble [38 x 2]>      5
##  6 <split [342/38]> Fold01 <tibble [342 x 2]> <tibble [38 x 2]>      6
##  7 <split [342/38]> Fold01 <tibble [342 x 2]> <tibble [38 x 2]>      7
##  8 <split [342/38]> Fold01 <tibble [342 x 2]> <tibble [38 x 2]>      8
##  9 <split [342/38]> Fold01 <tibble [342 x 2]> <tibble [38 x 2]>      9
## 10 <split [342/38]> Fold01 <tibble [342 x 2]> <tibble [38 x 2]>     10
## # ... with 90 more rows
```

Step 4: Estimate CV-MSE

We now estimate the CV-MSE for each value of λ.

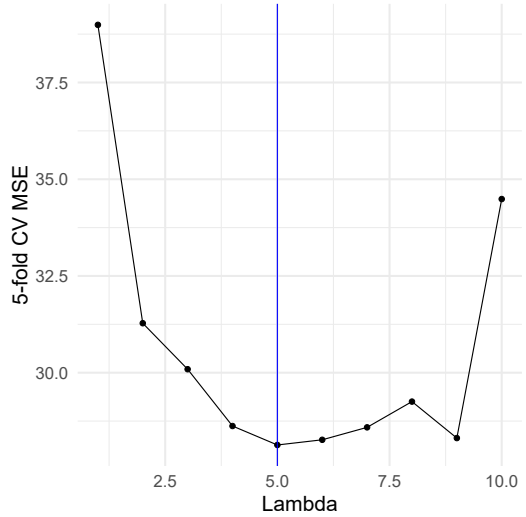
```
cv_mse <- cv_tune %>%
  mutate(model = map2(lambda, train, ~ lm(medv ~ poly(lstat, .x), data = .y))) %>%
  mutate(predicted = map2(model, validate, ~ augment(.x, newdata = .y))) %>%
  unnest(predicted) %>%
  group_by(lambda) %>%
  summarise(mse = mean((.fitted - medv)^2))

cv_mse
```

```
## # A tibble: 10 x 2
##    lambda   mse
##     <int> <dbl>
##  1      1  39.0
##  2      2  31.3
##  3      3  30.1
##  4      4  28.6
##  5      5  28.1
##  6      6  28.3
##  7      7  28.6
##  8      8  29.3
##  9      9  28.3
## 10     10  34.5
```

Step 5: Find the best model

Recall that the best performing model minimizes the CV-MSE.



> *"[I]n reality there is rarely if ever a true underlying model, and even if there was a true underlying model, selecting that model will not necessarily give the best forecasts..."*
>
> — **Rob J. Hyndman**

Step 6: Use the test set to evaluate the best model

Fit the best model ($\lambda = 5$) to the training set, make predictions on the test set, and calculate the test root mean square error (test-RMSE):


```
training_df %>%
  lm(medv ~ poly(lstat, 5), data = .) %>%  # fit model
  augment(newdata = testing_df) %>%        # predict unseen data
  rmse(medv, .fitted)                    # evaluate accuracy
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        5.11
```

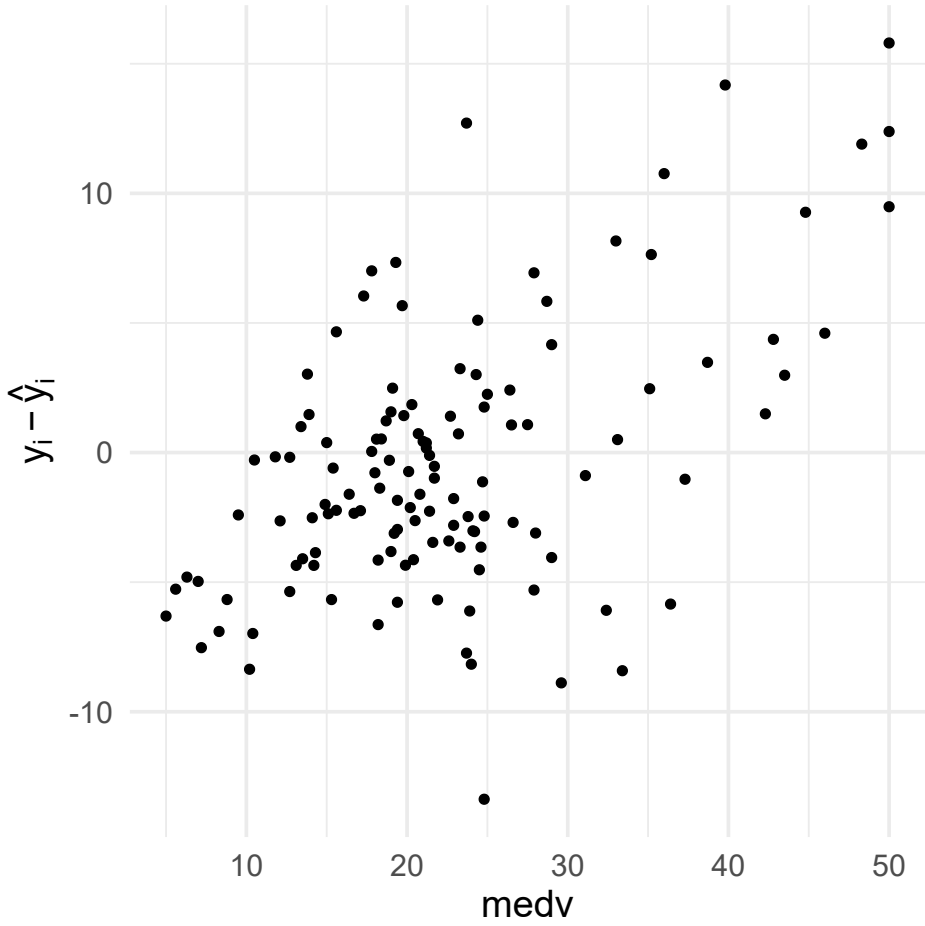
> **NOTE**: the test set RMSE is an estimator of the expected squared prediction error on unseen data *given* the best model.

An aside: plot your residuals

The distribution of the prediction errors $(y_i - \hat{y}_i)$ are another important sources of information about prediction quality.

```
training_df %>%
  lm(medv ~ poly(lstat, 5), data = .) %>%
  augment(newdata = testing_df) %>%
  mutate(error = medv - .fitted) %>%
  ggplot(aes(medv, error)) +
  geom_point() +
  labs(y = expression(y[i] - hat(y)[i]))
```

For example, see how biased the prediction for high levels of `medv` are.



ML and econometrics

ML vs. econometrics

Apart from jargon (Training set vs. in-sample, test-set vs. out of sample, learn vs. estimate, etc.) here is a summary of some of the key differences between ML and econometrics:

Machine Learning	**Econometrics**
prediction	causal inference
$\hat{Y}$	$\hat{\beta}$
minimize prediction error	unbiasedness, consictency, efficiency
---	statistical inference
stable environment	counterfactual analysis
black-box	structural

ML and causal inference

> *"Data are profoundly dumb about causal relationships."*
>
> — **Pearl and Mackenzie, *The Book of Why***

Intervention violates the stable DGP assumption:

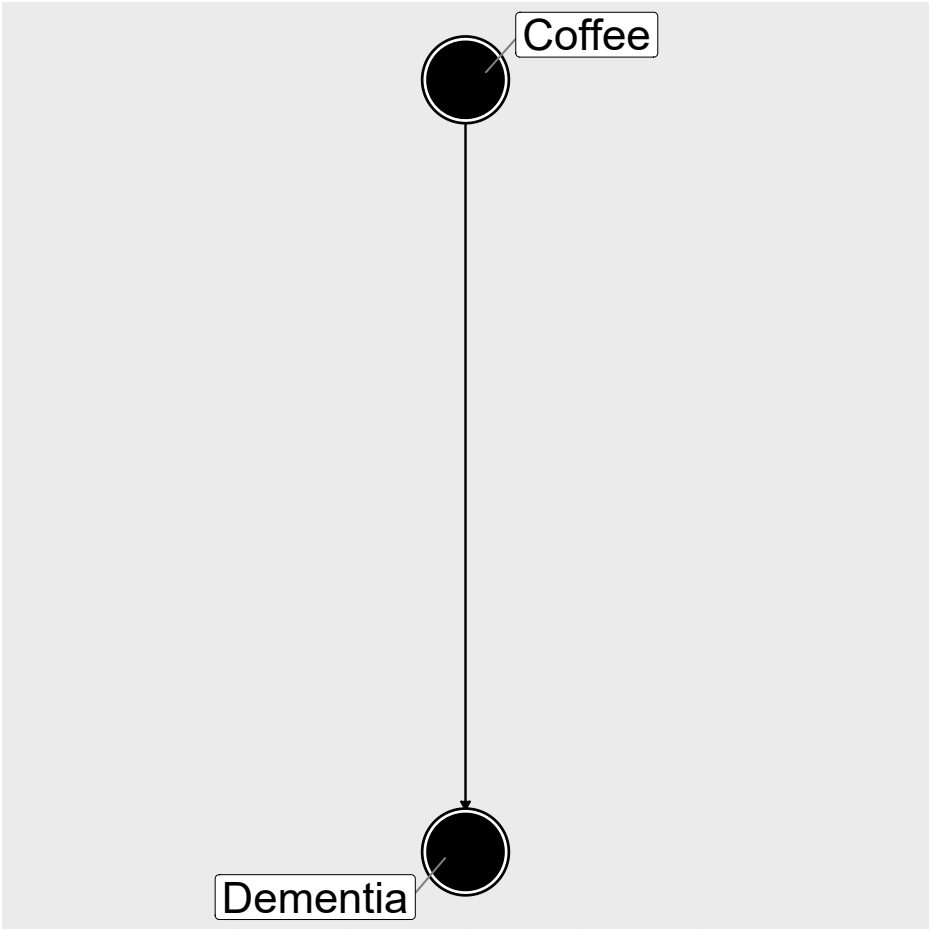
$$P(Y|X = x) \neq P(Y|do(X = x)),$$

where

- $P(Y|X = x)$ is the the probability of Y given that we *observe* $X = x$.
- $P(Y|do(X = x))$ is the the probability of Y given that we *intervene* to set $X = x$

"A new study shows that..."

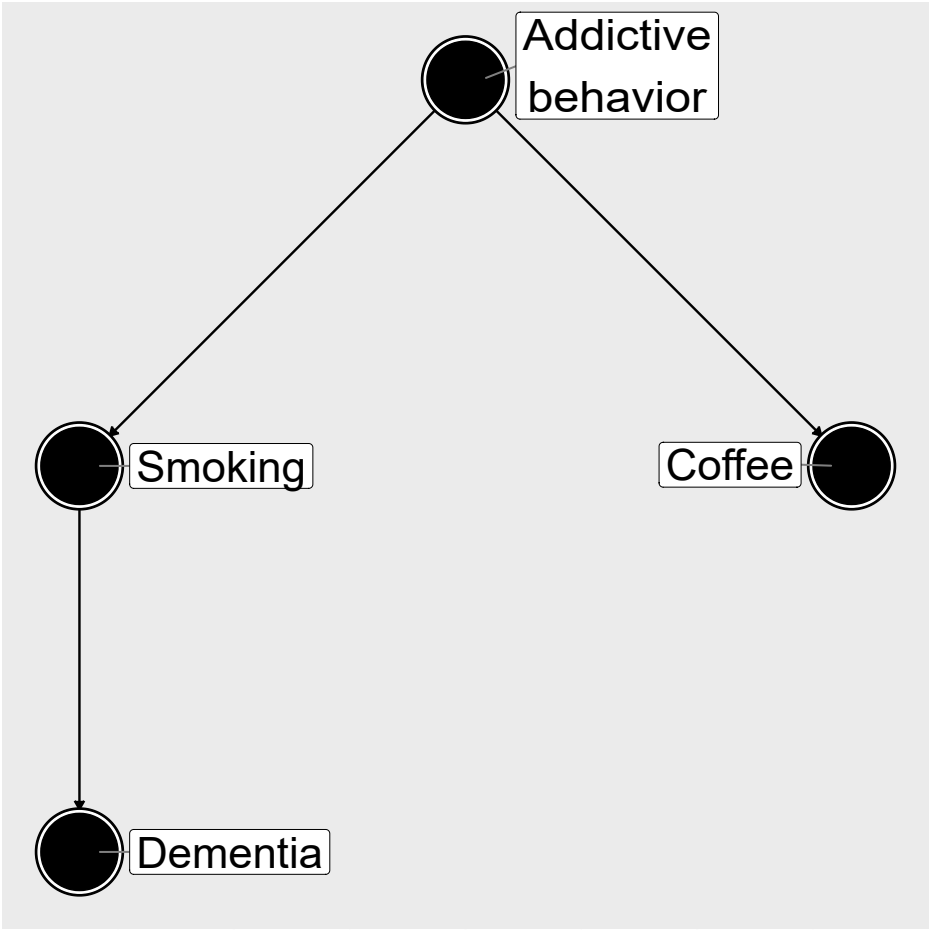
TOY PROBLEM: Say that we find in the data that `coffee` is a good predictor of `dementia`. Is avoiding `coffee` a good idea?



To explain or to predict?

- In this example `coffee` is a good *predictor* of `dementia`, despite not having any causal link to it.
- Controlling for `smoking` will give us the causal effect of `coffee` on `dementia`, which is zero.

> In general, causal inference always and everywhere dependes on *assumptions* about the DGP (i.e., the data never "speaks for itself").



ML in aid of econometrics

Consider the standard "treatment effect regression":

$$Y_i = \alpha + \underbrace{\tau D_i}_{\text{low dimensional}} + \underbrace{\sum_{j=1}^{p} \beta_j X_{ij}}_{\text{high dimensional}} + \varepsilon_i, \quad \text{for } i = 1, \ldots, n$$

where

- An outcome Y_i
- A treatment assignment $D_i \in \{0, 1\}$
- A vector of p control variables X_i

Our object of interest is often $\hat{\tau}$, the average treatment effect (ATE).

`slides::end()`

 Source code

References

[1] A. Agrawal, J. Gans and A. Goldfarb. *Prediction Machines: The Simple Economics of Artificial Intelligence*. Harvard Business Review Press, 2018.

[2] S. Athey. "The impact of machine learning on economics". In: *The Economics of Artificial Intelligence: An Agenda*. University of Chicago Press, 2018.

[3] L. Breiman. "Statistical modeling: The two cultures (with comments and a rejoinder by the author)". In: *Statistical science* 16.3 (2001), pp. 199-231.

[4] D. Harrison Jr and D. L. Rubinfeld. "Hedonic housing prices and the demand for clean air". In: *Journal of environmental economics and management* 5.1 (1978), pp. 81-102.

[5] G. Shmueli. "To explain or to predict?" In: *Statistical science* 25.3 (2010), pp. 289-310.

References

[1] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.