Many-Dimensional Data

Summer 2025

Irene Iodice

Data Science in Economics

Road–map

Applications in Economics

Limitations of OLS
- Diagnostics Fail to Rescue Us

Evaluating Model Performance

Beyond OLS
- Subset Selection
- Shrinkage Estimators

Applications

Why Should We Care About High-Dimensional Data?

- Modern data sets often measure $p \gg n$ features.
- **Opportunities:** richer signals, personalised predictions, automated text/image analysis.
- **Challenges:** overfitting, interpretability, computational burden.
- Central question of this lecture: *How can we learn reliably when p is large?*

What Do We Mean by n and p?

n – *observations*, p – *predictors*

Rows vs. columns in your data matrix.

Low-dimensional example

- $n = 2\,000$ patients
- $p = 3$ covariates (age, sex, BMI)

High-dimensional example

- $n = 200$ individuals
- $p = 500\,000$ SNPs (genetics)

Key takeaway

Same statistical questions—prediction, inference—become harder when p grows.

Section 1

Applications in Economics

Selected Use-Cases

- **Healthcare:** predicting sepsis risk from hundreds of sensor streams (Kleinberg *et al.*, 2015).
- **Finance:** Predict loan default in Latin America where credit history is absent with mobile phone metadata (Bjorkegren Grissen, 2017).
- **Urban policy:** mapping poverty from satellite imagery with millions of pixel features (Naik *et al.*, 2017).
- **Policing:** estimating weapon possession likelihood from rich incident reports (Goel *et al.*, 2016).

Common pattern

All tasks involve *hundreds to millions* of predictors

Section 2

Limitations of OLS

Measuring the Quality of Fit

Goal

Evaluate how well our model's predictions align with actual outcomes.

Key metric in regression: *Mean Squared Error (MSE)*

$$\text{MSE}_i = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

- y_i is the true value for observation i.
- $\hat{f}(x_i)$ is the predicted value from our model.
- MSE quantifies the average squared difference between predictions and actual values.

Interpretation

A lower MSE means better fit — smaller prediction errors on average.

Bias–Variance Decomposition

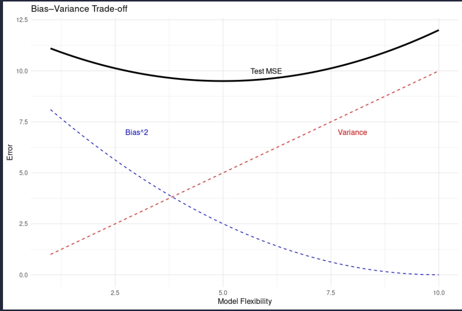
- For any fixed test point x_0, the expected test MSE splits into 3 non–negative parts

$$\mathbb{E}\big[(y_0 - \hat{f}(x_0))^2\big] = \underbrace{\mathrm{Var}\big[\hat{f}(x_0)\big]}_{\text{variance}} + \underbrace{\big(\mathrm{Bias}\big[\hat{f}(x_0)\big]\big)^2}_{\text{bias}^2} + \underbrace{\mathrm{Var}(\varepsilon)}_{\text{irreducible error}}$$

- **Variance**: how much $\hat{f}$ would change if we refit on a new training (unobserved) set.
- **Bias**: error introduced by approximating the unknown, possibly complex f with a simpler model.
- The irreducible error $\mathrm{Var}(\varepsilon)$ comes from intrinsic noise

The Bias–Variance Trade-Off

- Increasing model *flexibility* (# of p, polynomial degree, # of splits in a decision tree) $\Rightarrow \downarrow$ bias but $\uparrow$ variance.
- Test MSE is U-shaped: initially falls as bias drops, then rises when variance dominates.
- Optimal flexibility balances the two curves (vertical dotted line in the usual schematic).
- **Practical rule**: very simple methods risk high bias, highly flexible ones risk high variance; we need to balance!



*

.

```r
flexibility <- seq(1, 10, by = 0.1)
bias2 <- (10 - flexibility)^2 / 10
variance <- flexibility
test_mse <- bias2 + variance + 2  # include irreducible error

df <- data.frame(flexibility, bias2, variance, test_mse)

ggplot(df, aes(x = flexibility)) +
  geom_line(aes(y = bias2), color = "blue", linetype = "dashed") +
  geom_line(aes(y = variance), color = "red", linetype = "dashed") +
  geom_line(aes(y = test_mse), color = "black", size = 1.2) +
  labs(title = "Bias-Variance Trade-off", y = "Error", x = "Model Flexibility") +
  annotate("text", x = 3, y = 7, label = "Bias^2", color = "blue") +
  annotate("text", x = 8, y = 7, label = "Variance", color = "red") +
  annotate("text", x = 6, y = 11, label = "Test MSE", color = "black") +
  theme_minimal()
```

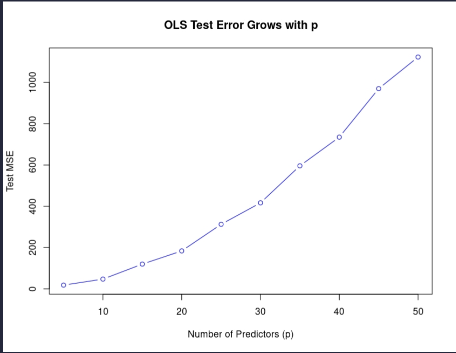
Why Ordinary Least Squares Breaks Down

The OLS estimator

$$\hat{\boldsymbol{\beta}}_{\text{OLS}} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \Big(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij}\Big)^2.$$

1. **Rank deficiency (**$p \geq n$**)**
   - Design matrix X loses full rank.
   - $X^\top X\boldsymbol{\beta} = X^\top y$ have *infinitely* many solutions.
2. **Variance explosion (multicollinearity)**
   - Highly correlated predictors $\rightarrow$ small eigenvalues of $X^\top X$.
   - Coefficient estimates fluctuate wildly across samples.
3. **Noise variables galore**
   - Including many irrelevant X_j adds variance but no signal.
   - Test MSE rises even when $p < n$.
4. **Zero coefficients are rare**
   - OLS seldom yields exact zeros, hurting interpretability and parsimony.

Visual: Variance Explosion



MC simulation: For each $p \in \{5, 10, \dots, 50\}$, we simulate $n = 50$ observations with correlated predictors ($\rho = 0.9$). Half of the coefficients are set to zero. We compute the test mean squared error of OLS across 200 repetitions. **Result:** MSE increases with dimensionality (overfitting)

Monte-Carlo Sim

```r
set.seed(1234)
# Store average test MSE for each value of p
test_mse <- sapply(seq(5, 50, by = 5), function(p) {
  # Set true coefficients: alternate 1s and 0s (half signal, half noise)
  beta <- rep(c(1, 0), length.out = p)
  # Create a p x p covariance matrix with strong correlation (rho = 0.9)
  Sigma <- matrix(0.9, p, p) + diag(p) * 0.1  # ensures diag = 1

  # Run 200 MC simulations for this p
  mse <- replicate(200, {
    # Simulate training data: n = 50 observations, p predictors
    X <- mvrnorm(50, rep(0, p), Sigma)
    y <- X %*% beta + rnorm(50)  # generate response with noise

    # Simulate independent test data: n = 100 observations
    test_X <- mvrnorm(100, rep(0, p), Sigma)
    test_y <- test_X %*% beta + rnorm(100)

    # Fit OLS on training data and predict on test data
    y_hat <- predict(lm(y ~ X), newdata = data.frame(X = test_X))

    # Compute mean squared error on test set
    mean((test_y - y_hat)^2)
  })
 # Return average test MSE over 200 simulations
  mean(mse)
})
plot(seq(5, 50, by = 5), test_mse, type = "b", col = "blue",
     xlab = "Number of Predictors (p)", ylab = "Test MSE",
     main = "OLS Test Error Grows with p")
```

Subsection 1

Diagnostics Fail to Rescue Us

Classical Model Diagnostics are ineffective

Training fit $\neq$ Generalisation

- Least– squares chooses $\hat{\beta}$ to minimise the insample RSS, so the training MSE $\text{MSE}_{\text{train}} = \text{RSS}/n$ is *optimistically biased*.
- Adding predictors always drives $\text{RSS}_{\text{train}}$ down and R^2_{train} up — even if the new variables contain only noise.
- The test error, by contrast, follows the Ushape from the bias–variance tradeoff and may rise once variance dominates.

Implication

In high dimensions, metrics computed on the *training* data (R^2, RSS, adjusted R^2) are **not reliable** for model selection.

Alternative model diagnostics

Four Classical Criteria

- **AIC:** $\frac{1}{n}\left(\text{RSS} + 2p\hat{\sigma}^2\right)$
- **BIC:** $\frac{1}{n}(\text{RSS} + \log(n)\, p\, \hat{\sigma}^2)$ — heavier penalty than AIC.
- **Adjusted** R^2**:** $1 - \frac{\text{RSS}/(n-p-1)}{\text{TSS}/(n-1)}$, penalises added predictors.

Interpretation

AIC, and BIC: **lower is better**. Adjusted R^2: **higher is better**. All help against overfitting.

Classical Model Diagnostics Lose Their Bite

- R^2 and adjusted R^2 always increase with additional predictors.
- Information criteria (AIC/BIC) rely on asymptotics where p is fixed.
- Hypothesis tests for individual β_j become unreliable due to multiple-testing and multicollinearity.

Bottom line

OLS provides *neither* stable predictions *nor* valid inference in high dimensions.

Section 3

Evaluating Model Performance

Why Simple Training Error Is Misleading

- Training error always non-increasing in model complexity.
- Need *test-set* performance — but data is scarce.

Solution

Resampling methods emulate the test-set scenario.

Cross-Validation in a Nutshell

Validation-set split

- One random split into train/test.
- Fast but high variance.

K-fold CV

- Partition data into K chunks.
- Cycle each chunk as test-fold.
- Typical: $K = 5$ or 10.

Leave-One-Out CV (LOOCV) – extreme case $K = n$: minimal bias, maximal computation.

Visual: The Validation Set Approach

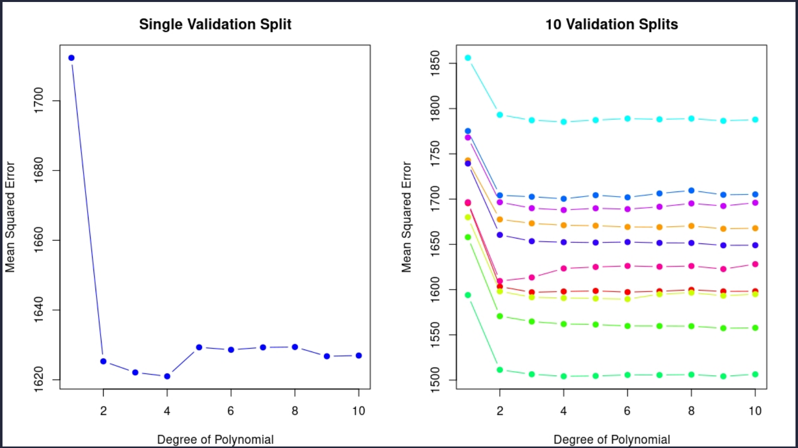
Full Dataset (n observations)

Training Set	Validation Set

Fit model $\hat{f}(x_{\text{train}})$

Evaluate MSE $\sum(y - \hat{f}(x_{\text{test}}))^2$

Validation of Mincerian Equation



Best approximation of the **wage–age** relationship is quadratic, but there is high variability in estimated test error across different validation splits.

Visual: K-Fold Cross-Validation

Full Dataset (n observations split into $K = 5$ folds)

Train on $K - 1$ folds, test on 1 fold. Repeat K times.

LOOCV as a Limiting Case of K-Fold

- When $K = n$, each observation is used once as the validation set.
- Repeat n times: each time, fit the model on $n-1$ points and test on the left-out one.
- This gives n test errors, one for each observation:

$$\text{MSE}_i = (y_i - \hat{f}^{(-i)}(x_i))^2$$

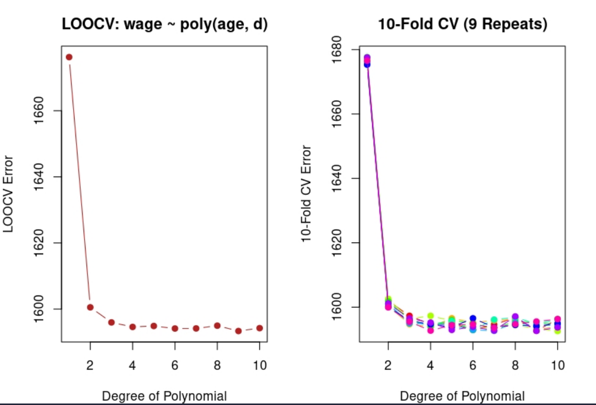
- The LOOCV estimate of test error is the average:

$$\mathrm{CV}(n) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{MSE}_i$$

Key Insight

LOOCV uses almost all the data for training each time $\Rightarrow$ low bias, but computationally expensive.

K-fold validation of Mincerian Equation



Estimated test error across different validation splits is similar!

Monte-Carlo Sim

```r
library(boot)
library(ISLR2) # to access the Wage dataset
set.seed(1234)
n_repeats <- 9
cv_errors_matrix <- matrix(NA, nrow = n_repeats, ncol = length(degrees))

for (i in 1:n_repeats) {
  fold_errors <- sapply(degrees, function(d) {
    fit <- glm(wage ~ poly(age, d), data = Wage)
    cv.glm(Wage, fit, K = 10)$delta[1]
  })
  cv_errors_matrix[i, ] <- fold_errors
}

matplot(degrees, t(cv_errors_matrix), type = "b", pch = 19, lty = 1,
        col = rainbow(n_repeats),
        xlab = "Degree of Polynomial", ylab = "10-Fold CV Error",
        main = "10-Fold CV (9 repeats)")
```

Section 4

Beyond OLS

Beyond Least Squares: Three Solutions for Many Predictors

When p is large, ordinary least squares (OLS) becomes unreliable:

- Overfitting
- High variance
- Poor generalization

Three major classes of solutions:

1. **Subset Selection**
2. **Shrinkage (Regularization)**
3. **Dimension Reduction**

Each tackles high-dimensionality differently. Let's explore how.

Subsection 1

Subset Selection

Algorithm: Best Subset Selection

Algorithm 1: Best Subset Selection

Input : A response vector y, predictor matrix X with p columns

Output: Selected model $\mathcal{M}_k$ with best performance

1. Let $\mathcal{M}_0$ denote the **null model**, containing no predictors.
Predicts the sample mean for all observations.

2. For $k = 1, 2, \ldots, p$:
(a) Fit all $\binom{p}{k}$ models with exactly k predictors.
(b) Select the model $\mathcal{M}_k$ with the lowest RSS (or highest R^2).

3. Choose the best model among $\{\mathcal{M}_0, \ldots, \mathcal{M}_p\}$ using:

- C_p, AIC, BIC, or Adjusted R^2
- K-fold Cross-Validation.

Best-Subset vs Stepwise: Pros and Cons

Method	Pros	Cons
Best Subset	Conceptually simple, can yield sparse interpretable models.	Computationally infeasible if $p \gtrsim 40$. Requires evaluating all 2^p models.
Forward/Backward Stepwise	Much cheaper: only $\sim \frac{p(p+1)}{2}$ models. Fast even for large p.	Greedy: may miss the globally best model. Sensitive to order of entry/removal.

Model selection criteria:

- Cp, BIC, Adjusted R^2 can help pick the best subset size.

Subsection 2

Shrinkage Estimators

Penalized linear models

$$\min_{\beta\in\mathbb{R}^p}\left\{\underbrace{l(\alpha,\beta)}_{\text{loss function}}+n\lambda\sum_{j=1}^{p}\underbrace{k_j(|\beta_j|)}_{\text{penality shrinkage}}\right\}$$

where:

- $l(\alpha,\beta)=\frac{1}{n}\sum_{i=1}^{n}\left(v_i-(\alpha+\mathbf{x}_i'\beta)\right)^2$ in Gaussian linear reg. (RSS)
- $k_j(.)$ increasing cost function that penalizes dev of β_j from zero
- $\lambda\geq 0$ adjusts the margin (or 'complexity') of the solution (typically chosen using a held-out sample or K-fold Cross Validation)
- The sample size n term scales down the penalty term to compensate for the increased amount of information present in larger dataset.

Common functions for $k_j(.)$

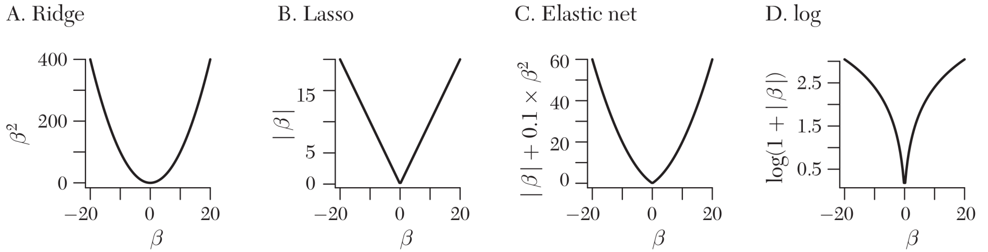


Figure 1

Note: From left to right, L_2 costs (ridge, Hoerl and Kennard 1970), L_1 (lasso, Tibshirani 1996), the "elastic net" mixture of L_1 and L_2 (Zou and Hastie 2005), and the log penalty (Candès, Wakin, and Boyd 2008).

Ridge Regression (ℓ_2 shrinkage)

- Adds a quadratic penalty to keep coefficients small:

$$\hat{\beta}^{\text{ridge}} = \arg\min_{\beta} \left\{ \text{RSS}(\beta) \ + \ \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

- **No variable selection**: all $\beta_j \neq 0$ (unless $\lambda \to \infty$).
- Great when predictors are *many* and *strongly correlated*; shrinks them toward each other to reduce variance.
- Choose tuning parameter λ via K-fold CV.

Key intuition

Shrinkage trades a little bias for a large drop in variance $\Rightarrow$ lower test error.

Lasso Regression (ℓ_1 shrinkage & selection)

- Penalises absolute values:

$$\hat{\beta}^{\text{lasso}} = \arg\min_{\beta} \left\{ \text{RSS}(\beta) \; + \; \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

- Automatic variable selection: ℓ_1 geometry creates corners $\Rightarrow$ many coefficients shrink to **exactly 0**.
- Produces sparse, interpretable models – convenient when $p \gg n$.
- Same tuning workflow: search λ on a log-grid with CV.

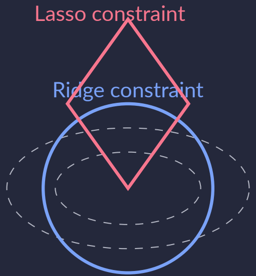
Sparsity property

Lasso can mimic best-subset selection without the 2^p cost.

Why Lasso selects but Ridge doesn't

$$\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq s \tag{6.8}$$

$$\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 \leq s \tag{6.9}$$



Elastic Net: $\ell_1 + \ell_2$

$$\hat{\beta}^{\text{EN}} = \arg\min_{\beta} \left\{ \text{RSS}(\beta) + \lambda \left[\alpha \sum_j |\beta_j| + (1-\alpha) \sum_j \beta_j^2 \right] \right\}$$

- $\alpha \in [0, 1]$: mixing parameter $\alpha = 1$ = Lasso, $\alpha = 0$ = Ridge.
- Keeps sparsity *and* grouping effect: correlated predictors tend to enter or drop together.
- Recommended when p is large and predictors are correlated.

Two hyper-parameters

Tune λ *and* α with nested CV or a 2-D grid search.

Shrinkage Cheat-Sheet

Method	Penalty	Sparsity?	Best for
Ridge	$\sum \beta_j^2$	No	Multicollinearity, $p < n$
Lasso	$\sum \lvert\beta_j\rvert$	Yes	Interpretation, $p \gg n$
Elastic Net	$\ell_1 + \ell_2$	Yes	Correlated groups, $p \gg n$

When Should You Use Ridge, Lasso, or Elastic Net?

Rule of Thumb

- **Lasso:** Sparse solutions — you want variable selection and interpretability.
- **Ridge:** Dense solutions — all predictors matter, but may be highly correlated.
- **Elastic Net:** Mix of both — many predictors, some collinear, some irrelevant.

Example Use Cases:

- *Lasso*: Selecting the most predictive demographics in a wage model.
- *Ridge*: Forecasting GDP growth with 120 macro indicators (FRED-MD).
- *Elastic Net*: Modelling wage exposure with state × industry dummies.

Pipeline for Penalized Regression

1. **Preprocess Data**
   - Standardize predictors (z-scores): essential for Lasso/Ridge penalties.
   - Dummy-encode categorical variables; impute or remove missing values.
2. **Define Tuning Grid**
   - **Lasso/Ridge:** $\lambda \in \{10^{-4}, \dots, 10^{2}\}$, log-spaced.
   - **Elastic Net:** Cross-product grid of λ and $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$.
3. **Cross-Validation**
   - Use $K = 5$ or 10-fold CV to select $(\lambda^{\star}, \alpha^{\star})$.
   - Select hyperparameters minimizing CV error (or deviance).
4. **Refit Final Model** on full training set using best parameters.
5. **Evaluate Performance** on a held-out test set or via nested CV.

Recommended packages: `glmnet` (R), `sklearn.linear_model` (Python), `tidymodels`.

Feature Engineering Before Regularization

- **Standardize:** All predictors should have zero mean and unit variance.
- **Dummy-encode:** Convert factors to 0/1 indicators.
- **Create interactions:** Especially for theoretically relevant terms (e.g., gender $\times$ occupation).
- **Handle nonlinearity:** Use polynomial terms or (preferably) splines.
- **Deal with missingness:** Impute (mean/median/model-based) or drop rows/columns as appropriate.

Why it matters

Lasso and Ridge penalize raw coefficient magnitudes — this only makes sense when predictors are on the same scale.

Why Log-Spaced Grid for λ?

Tuning the penalty parameter λ: crucial for balancing fit and complexity in Lasso/Ridge.

Why use a log grid?

- **Nonlinear shrinkage:** Small changes in λ near zero cause large shifts in coefficients.
- **Efficient resolution:** Denser sampling in the sensitive range (e.g., 10^{-4} to 1).
- **Avoid waste:** Linear spacing overrepresents large λ, where all $\beta_j = 0$.
- **Invariance to scale:** Log grid works well across data magnitudes — especially after standardization.

Typical search grid:

$$\lambda \in \{10^{-4}, 10^{-3.9}, \dots, 10^{2}\} \quad \text{(100 values, log-spaced)}$$

Example: Lasso with Cross-Validation in R

```
library(glmnet)
X <- model.matrix(wage ~ ., data = Wage)[, -1]
y <- Wage$wage
fit <- cv.glmnet(X, y, alpha = 1, standardize = TRUE) # alpha = 1: Lasso, alpha = 0: Ridge,
plot(fit)
coef(fit, s = "lambda.min")  #  select the λ (tun. prm) that min the CV error.
```

Section 5

Applications

Case Study: Machine-Learning the Minimum-Wage Effect

Paper: *Seeing Beyond the Trees* — Cengiz et al. (2024) JLE

- Goal: estimate wage & employment effects on *all* workers likely to earn the minimum wage, not just teens.
- Problem: true treatment status (being bound by the minimum wage) is **latent**.
- **ML step**: use gradient-boosted trees to predict, for every CPS individual, the probability p_i of earning $\leq$ current minimum wage, based on rich demographics $\{$age, gender, race, education, industry$\}$.
- Construct two data-driven groups $\rightarrow$ *high-probability* (top 10 $\rightarrow$ *high-recall* (captures 75
- Apply event-study (DiD) around 159 state-level minimum-wage hikes, using ML groups as treated cohorts.

Elastic-Net Step in Dube & Lindner (2021)

Goal — Predict the *latent* probability an individual earns $\leq$ the binding minimum wage.

1. **Outcome (training label)**

$$y_i = 1\{\text{hourly wage}_i \leq \text{MW}_{st}\},$$

   built from CPS 2013 micro data (chosen *before* policy window).

2. **Features (X)** Age (splines), gender, race, education, marital status, industry, occupation, hours, state fixed effects, and their interactions $\Rightarrow p \approx 150$ predictors after dummies / splines.

3. **Model** — Logistic Elastic-Net

$$\min_{\beta} \underbrace{-\frac{1}{n}\sum y_i \log \hat{p}_i + (1-y_i)\log(1-\hat{p}_i)}_{\text{log-loss}} + \lambda\Big[\alpha\|\beta\|_1 + (1-\alpha)\|\beta\|_2^2\Big],$$

   with predictors z-scored.

4. **Tuning**
   - $K = 10$–fold CV on a (λ, α) grid ($\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$).
   - Chosen by minimum CV deviance (AUC close behind).
   - Optimal: $\alpha \approx 0.5$ — mix of Lasso & Ridge.

Findings from the ML-Enhanced Design

- **Wage effect**: +2–3% average real wages for high-probability group over five years.
- **Employment, Unemployment, Participation:**
  - No systematic job losses in either ML group.
  - Unemployment and labour-force participation essentially unchanged.
- **Why ML mattered:**
  1. Increases coverage: $\approx 75\%$ of all affected workers vs. traditional teen-only designs.
  2. Improves precision: larger treated sample $\Rightarrow$ tighter confidence bands.
  3. Flexible, replicable treatment assignment—can be updated as labour-force composition changes.

Pipeline takeaway

Combine predictive ML (to learn who is treated) with causal DiD (to estimate policy impact) $\Rightarrow$ scalable framework for other labour-market programs.

Case: Machine Labor (Angrist & Frandsen 2022, JLE)

- **Paper:** Angrist, J. D. & Frandsen, B. (2022). *Machine Labor. Journal of Labor Economics*, 40(S1): S97–S140.

ML Setup: Machine Labor

- **Outcome:** $Y_i = \log(\text{weekly wage}_i)$ of male college graduates.
- **Treatment:** D: college attributes (e.g., private/elite attendance). **Controls:** X: high-dimensional college-application variables (approx. 384 features: number of schools applied to/accepted, SAT scores, interactions).
- **Model:** Linear regression:
$$Y = \alpha D + X'\beta + \varepsilon.$$
  Use post-double-selection Lasso (Belloni et al., 2014): run Lasso of Y on X and of D on X, take union of selected X_S, then OLS on (D, X_S).
- **Tuning:** Penalty λ chosen by plug-in rule and 10-fold CV (via `lassopack`).

Post-Double-Selection (PDS) Lasso

Goal: Estimate the treatment effect α in high-dimensional settings:

$$Y = \alpha D + X'\beta + \varepsilon$$

where X has many potential confounders (e.g., test scores, demographics).

Steps:

1. Run Lasso of Y on X → select controls X_Y
2. Run Lasso of D on X → select controls X_D
3. Define $X_S = X_Y \cup X_D$ (union of selected variables)
4. Run OLS of Y on D and X_S

Why this works:

- Captures variables predictive of Y *or* D (helps control for confounding)
- Avoids overfitting by reducing dimensionality via Lasso
- Allows valid inference on α even if $p \gg n$

Main Findings: Machine Labor

- **OLS + Lasso (PDS):** College effects from PDS match full-model OLS. E.g., private-college premium $\approx 0.02–0.04$ (PDS) vs. 0.017 (full OLS). Conclusion: no elite/quality premium.
- **Tuning robustness:** Different λ values change variable count (e.g., 18 vs. 100 vs. 112), but estimates of α remain stable.
- **Single vs. Double selection:** Lasso on Y alone yields inflated effect (≈ 0.08). Double-selection corrects bias.
- **IV first-stage:** Lasso IV helps reduce bias but is outperformed by LIML and split-sample IV. Risk of *pretest bias* with ML-selected instruments.
- **Conclusion:** ML controls replicate baseline. ML *helps check*, not generate, identification. Classical IV tools still preferred.

Application III: Social Spillovers in Movie Consumption (Gilchrist & Sands 2016, JPE)

- **Paper:** Gilchrist, D. S. & Sands, E. G. (2016). *Something to Talk About: Social Spillovers in Movie Consumption. Journal of Political Economy*, 124(5): 1268–1304.

Application: Social Spillovers in Movie Consumption

Paper: Gilchrist, D.S. & Sands, E.G. (2016). *Something to Talk About: Social Spillovers in Movie Consumption*, *Journal of Political Economy*, 124(5): 1268–1304.

Goal: Identify causal effect of early movie attendance on subsequent viewership — i.e., social momentum effects.

Challenge: Early attendance may reflect unobserved movie quality → OLS biased.

Identification Strategy with ML-IV

Outcome (Y_i): Total movie attendance over 5 weekends post-release.
Treatment (D_i): Opening-weekend attendance.
Controls (X): Movie fixed effects (genre, marketing, release timing).
Instruments (Z):

- High-dimensional local weather variables (rainfall, temperature, etc.)
- Include interactions (e.g., rain × weekend × region).
- Weather affects opening attendance but is plausibly exogenous to total quality.

Machine Learning step: Use **Lasso** to select relevant weather instruments from many candidates.

Model Setup: Lasso-IV Framework

Two-Stage Least Squares (2SLS) with Lasso-selected instruments:

$$\text{First stage:} \quad D = Z\pi + v \quad \text{(select } Z \text{ using Lasso)}$$

$$\text{Second stage:} \quad Y = \alpha D + X'\beta + \varepsilon$$

- Lasso shrinks irrelevant instruments to zero – reduces overfitting risk.
- Penalty λ chosen by cross-validation.
- Validates causal impact of D using exogenous variation in weather.

Main Findings: Social Spillovers

- **Social Effect:** A 1% increase in opening-week attendance leads to a **2% increase** in cumulative 5-week attendance.
- **Local Containment:** Effects are **local to the city** — no cross-city contagion.
- **No Quality Learning:** Effect is **independent of movie reviews or quality**, suggesting a **social experience motive**.

ML vs. Naive OLS

- Naive OLS fails to isolate exogenous variation — confounded by appeal.
- ML-IV (Lasso-2SLS) delivers more credible identification.

References I

- James, G., Witten, D., Hastie, T., Tibshirani, R. (2021). *An Introduction to Statistical Learning with Applications in R* (2nd ed.). Springer. www.statlearning.com
- Belloni, A., Chernozhukov, V., & Hansen, C. (2011). *Inference for high-dimensional sparse econometric models*. Advances in Economics and Econometrics.
- Cengiz, D., Dube, A., Lindner, A., Zentler-Munro, R. (2022). *The minimum-wage employment effect reconsidered*. Journal of Labor Economics.
- Chernozhukov, V., et al. (2019). *Inference in factorial experiments with high-dimensional covariates*. arXiv:1903.10075.
- Hastie, T., Tibshirani, R., Friedmann, J. (2023). *Statistical Learning with Sparsity: The LASSO and Generalizations*.