Introduction to Classical ML and Its Application in Econometric Research

Hsiu Hsuan Yeh

Dept of Econ

National Taiwan University

2023-06-14

Research Question

- who is a minimum wage worker
  - identify the potential workers who have been working had the minimum wage been different
- what is the effect of increasing minimum wage
  - increasing employment
  - decreasing unemployment
  - increasing labor force participation(LFP)

minimum wage classification

- traditional way
  - specify demographic groups such as the teens, low educational
  - distribution based (Cengiz et al. (2019))
- machine learning
  - larger and more various groups
  - data driven which is no functional form

machine learning models

1. elastic net
2. decision tree
3. random forest
4. gradient boosting machine(GBM)
5. support vector machine(SVM)
6. neural network

overfitting

in machine learning model our goal is to do good generalization:

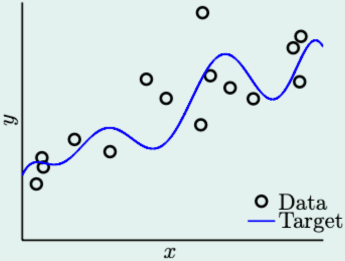
- minimize the in sample error
- let the in sample error as close as the out sample error

when dose overfitting(bad generalization, low bias high variance) happen?

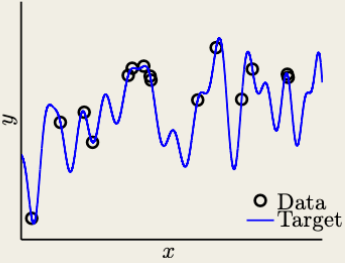
- small data size
- noise: stochastic, deterministic

overfitting

10-th order target function + noise



50-th order target function noiselessly



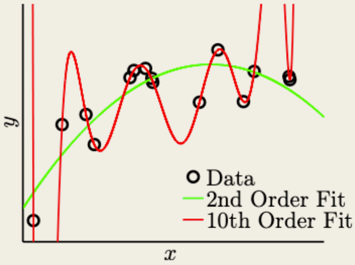
overfitting from best $g_2 \in \mathcal{H}_2$ to best $g_{10} \in \mathcal{H}_{10}$?

overfitting

10-th order target function + noise

	$g_2 \in \mathcal{H}_2$	$g_{10} \in \mathcal{H}_{10}$
E_{in}	0.050	0.034
E_{out}	0.127	**9.00**

50-th order target function noiselessly



	$g_2 \in \mathcal{H}_2$	$g_{10} \in \mathcal{H}_{10}$
E_{in}	0.029	0.00001
E_{out}	0.120	**7680**

overfitting from g_2 to g_{10}? **both yes!**

overfitting

how to combat overfitting?

- validation: leave one out, cross validation
- early stop
- blending (mix multiple model)

elastic net

elastic net = weighted lasso and Ridge

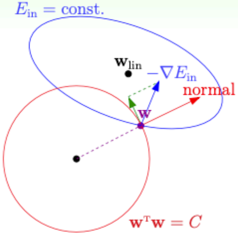
$$\min_{\beta} Q(\beta) + \lambda(\alpha||\beta||_2 + (1-\alpha)||\beta||_1)$$

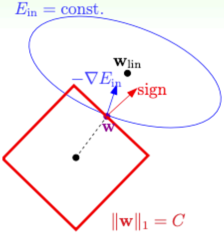
where

- Q: objective function (loss)
- λ: penalty term
- α: ratio of mixture

in this paper, thu author build a complex model by including all the features, their four-way interactions, and all of the interactions with the quadratic, cubic, and quartic terms of the age variable.

elastic net





L2 Regularizer

$$\Omega(\mathbf{w}) = \sum_{q=0}^{Q} w_q^2 = \|\mathbf{w}\|_2^2$$

- convex, differentiable everywhere
- easy to optimize

L1 Regularizer

$$\Omega(\mathbf{w}) = \sum_{q=0}^{Q} |w_q| = \|\mathbf{w}\|_1$$

- convex, **not** differentiable everywhere
- **sparsity** in solution

L1 useful if needing **sparse solution**

Adaptive Boosting

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_3, y_3), (\mathbf{x}_4, y_4)\}$$

$$\overset{\text{bootstrap}}{\Longrightarrow} \quad \tilde{\mathcal{D}}_t = \{(\mathbf{x}_1, y_1), (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_4, y_4)\}$$

weighted E_{in} on $\mathcal{D}$

$$E_{\text{in}}^{\mathbf{u}}(h) = \frac{1}{4}\sum_{n=1}^{4} u_n^{(t)} \cdot [\![y_n \neq h(\mathbf{x}_n)]\!]$$

$(\mathbf{x}_1, y_1)$, $u_1 = 2$
$(\mathbf{x}_2, y_2)$, $u_2 = 1$
$(\mathbf{x}_3, y_3)$, $u_3 = 0$
$(\mathbf{x}_4, y_4)$, $u_4 = 1$

E_{in} on $\tilde{\mathcal{D}}_t$

$$E_{\text{in}}^{0/1}(h) = \frac{1}{4}\sum_{(\mathbf{x},y)\in\tilde{\mathcal{D}}_t} [\![y \neq h(\mathbf{x})]\!]$$

$(\mathbf{x}_1, y_1)$, $(\mathbf{x}_1, y_1)$
$(\mathbf{x}_2, y_2)$

$(\mathbf{x}_4, y_4)$

Adaptive Boosting

'improving' bagging for binary classification:
how to re-weight for **more diverse hypotheses**?

$$g_t \leftarrow \operatorname*{argmin}_{h \in \mathcal{H}} \left(\sum_{n=1}^{N} u_n^{(t)} [\![y_n \neq h(\mathbf{x}_n)]\!] \right)$$

$$g_{t+1} \leftarrow \operatorname*{argmin}_{h \in \mathcal{H}} \left(\sum_{n=1}^{N} u_n^{(t+1)} [\![y_n \neq h(\mathbf{x}_n)]\!] \right)$$

if g_t '**not good**' for $\mathbf{u}^{(t+1)}$ $\Longrightarrow$ g_t-like hypotheses not returned as g_{t+1}
$\Longrightarrow$ g_{t+1} diverse from g_t

idea: **construct** $\mathbf{u}^{(t+1)}$ to make g_t **random-like**

$$\frac{\sum_{n=1}^{N} u_n^{(t+1)} [\![y_n \neq g_t(\mathbf{x}_n)]\!]}{\sum_{n=1}^{N} u_n^{(t+1)}} = \frac{1}{2}$$

Adaptive Boosting

$$\Sigma_{n=1}^{N} u_n^{t+1} \mathbb{1}(y_n = g_t(x_n)) = \Sigma_{n=1}^{N} u_n^{t+1} \mathbb{1}(y_n \neq g_t(x_n))$$

$$\Sigma_{n=1}^{N} \Sigma_{n=1}^{N} u_n^{t+1} \mathbb{1}(y_n \neq g_t(x_n)) \mathbb{1}(y_n = g_t(x_n)) = \Sigma_{n=1}^{N} \Sigma_{n=1}^{N} u_n^{t+1} \mathbb{1}(y_n = g_t(x_n)) \mathbb{1}(y_n \neq g_t(x_n))$$

$$\Sigma_{n=1}^{N} \frac{\Sigma_{n=1}^{N} u_n^{t+1} \mathbb{1}(y_n \neq g_t(x_n))}{\Sigma_{n=1}^{N} u_n^{t+1}} \mathbb{1}(y_n = g_t(x_n)) = \Sigma_{n=1}^{N} \frac{\Sigma_{n=1}^{N} u_n^{t+1} \mathbb{1}(y_n = g_t(x_n))}{\Sigma_{n=1}^{N} u_n^{t+1}} \mathbb{1}(y_n \neq g_t(x_n))$$

Adaptive Boosting

$$\Sigma_{n=1}^{N} \epsilon_{t+1} \mathbb{1}(y_n = g_t(x_n)) = \Sigma_{n=1}^{N} (1 - \epsilon_{t+1}) \mathbb{1}(y_n \neq g_t(x_n))$$

$$\Sigma_{n=1}^{N} \sqrt{\epsilon_{t+1}^2} \mathbb{1}(y_n = g_t(x_n)) = \Sigma_{n=1}^{N} \sqrt{(1 - \epsilon_{t+1})^2} \mathbb{1}(y_n \neq g_t(x_n))$$

$$\Sigma_{n=1}^{N} \sqrt{\frac{\epsilon_{t+1}^2}{\epsilon_{t+1}(1 - \epsilon_{t+1})}} \mathbb{1}(y_n = g_t(x_n)) = \Sigma_{n=1}^{N} \sqrt{\frac{(1 - \epsilon_{t+1})^2}{\epsilon_{t+1}(1 - \epsilon_{t+1})}} \mathbb{1}(y_n \neq g_t(x$$

$$\Sigma_{n=1}^{N} \sqrt{\frac{\epsilon_{t+1}}{(1 - \epsilon_{t+1})}} \mathbb{1}(y_n = g_t(x_n)) = \Sigma_{n=1}^{N} \sqrt{\frac{(1 - \epsilon_{t+1})}{\epsilon_{t+1}}} \mathbb{1}(y_n \neq g_t(x_n))$$

$$\Sigma_{n=1}^{N} \frac{1}{\blacklozenge_{t+1}} \mathbb{1}(y_n = g_t(x_n)) = \Sigma_{n=1}^{N} \blacklozenge_{t+1} \mathbb{1}(y_n \neq g_t(x_n))$$

$$\begin{cases} \blacklozenge_{t+1} >= 1 & \iff \epsilon_{t+1} <= \frac{1}{2} \\ \alpha_{t+1} = \log(\blacklozenge_{t+1}) >= 0 & \iff \blacklozenge_{t+1} >= 1 \end{cases}$$

Adaptive Boosting

$\mathbf{u}^{(1)} = [\frac{1}{N}, \frac{1}{N}, \cdots, \frac{1}{N}]$

for $t = 1, 2, \ldots, T$

1. obtain g_t by $\mathcal{A}(\mathcal{D}, \mathbf{u}^{(t)})$, where $\mathcal{A}$ tries to minimize $\mathbf{u}^{(t)}$-weighted 0/1 error
2. update $\mathbf{u}^{(t)}$ to $\mathbf{u}^{(t+1)}$ by

$$[\![y_n \neq g_t(\mathbf{x}_n)]\!] \text{ (incorrect examples): } u_n^{(t+1)} \leftarrow u_n^{(t)} \cdot \blacklozenge_t$$

$$[\![y_n = g_t(\mathbf{x}_n)]\!] \text{ (correct examples): } u_n^{(t+1)} \leftarrow u_n^{(t)} / \blacklozenge_t$$

where $\blacklozenge_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$ and $\epsilon_t = \frac{\sum_{n=1}^N u_n^{(t)} [\![y_n \neq g_t(\mathbf{x}_n)]\!]}{\sum_{n=1}^N u_n^{(t)}}$

3. compute $\alpha_t = \ln(\blacklozenge_t)$

return $G(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^T \alpha_t g_t(\mathbf{x})\right)$

Adaptive Boosting

Julia implementation

```julia
@inbounds for t = eachindex(s, k, theta, alpha)
    lossₜ = dropdims(mean(incorrect_ptr, dims=1); dims=1)
    weighted_lossₜ = dropdims(mean(uₜ  .* incorrect_ptr, dims=1); dims=1)

    idx = argmin(weighted_lossₜ)
    loss[t] = lossₜ[idx]
    best_pred[:, t] = pred[:, idx]

    k[t] = ceil(Int64, idx[1]/n)
    theta[t] = θ[idx[1]]
    s[t] = [-1, 1][idx[2]]

    opt_correct_ptr = best_pred[:, t] .== labels
    opt_incorrect_ptr = best_pred[:, t] .!= labels

    ϵₜ = (opt_incorrect_ptr' * uₜ) / sum(uₜ)
    diamondₜ = sqrt((1-ϵₜ)/ϵₜ)
    alpha[t] = log(diamondₜ)

    uₜ = (opt_incorrect_ptr .* uₜ) * diamondₜ +
          (opt_correct_ptr .* uₜ) / diamondₜ
    next!(p)
end
```

Gradient Boosting

AdaBoost Revisited: Example Weights

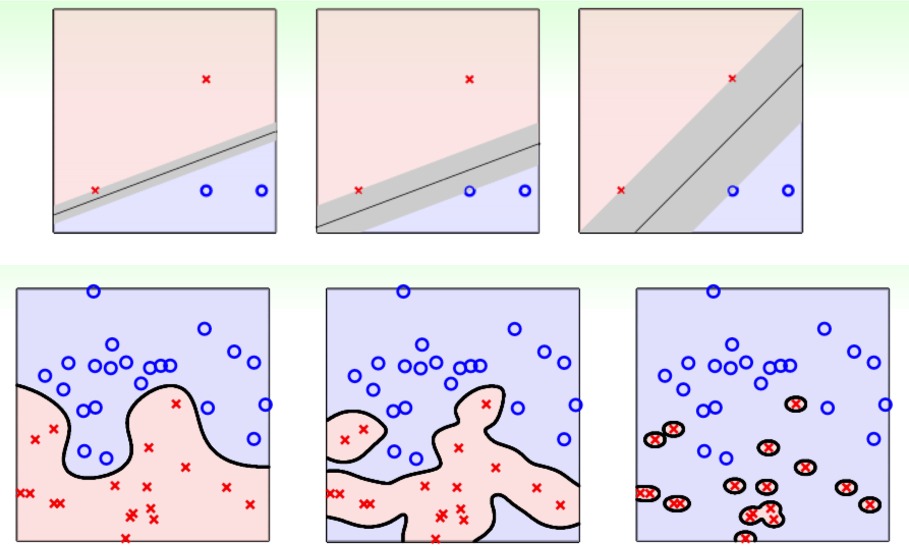
$$u_n^{(t+1)} = \begin{cases} u_n^{(t)} \cdot \lozenge_t & \textbf{if incorrect} \\ u_n^{(t)} / \lozenge_t & \textbf{if correct} \end{cases}$$

$$= u_n^{(t)} \cdot \lozenge_t^{-y_n g_t(\mathbf{x}_n)} = u_n^{(t)} \cdot \exp\left(-y_n \alpha_t g_t(\mathbf{x}_n)\right)$$

$$u_n^{(T+1)} = u_n^{(1)} \cdot \prod_{t=1}^{T} \exp(-y_n \alpha_t g_t(\mathbf{x}_n)) = \frac{1}{N} \cdot \exp\left(-y_n \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_n)\right)$$

- recall: $G(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t g_t(\mathbf{x})\right)$
- $\sum_{t=1}^{T} \alpha_t g_t(\mathbf{x})$: **voting score** of $\{g_t\}$ on $\mathbf{x}$

SVM



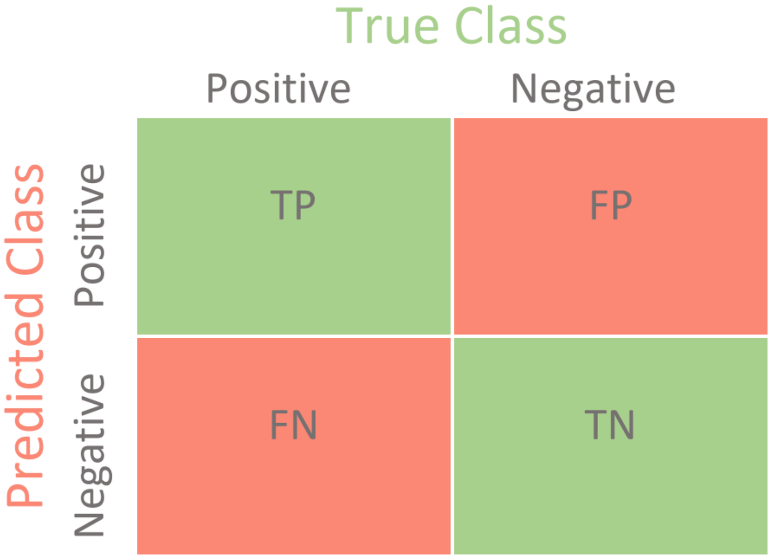
methodology

- use the machine learning model to predict who might be a potential minimum wage worker
  - hourly wage of less than 125% of the statutory minimum wage
  - high probability group: comprises the 10% of the population with the hightest likelihood of being affected by the policy.
  - high recall group: 75% of all minimum wage workers are captured
- worker-level selection criterion
  - there had not had been any prominent minimum wage events in the past 20 quarters
  - there is a prominent minimum wage change in the next 12 quarters
  - 469,174 observations, randomly draw 150, 000 for training

methodology

- prominent minimum wage change
  - (real)minimum wages increased by more than \$0.25
  - at least 2% of the workforce earned between the new minimum wage and the old minimum wage
- use DID to esitmate the change of wage, employment, unemployment, LFP in two groups
  - 8-year window around 172 prominent state-level minimum wage events
  - $y^g_{st} = \Sigma^4_{\tau=-3}\beta_\tau treat^\tau_{st} + \Omega_{st} + \mu_s + \rho_t + u_{st}$
  - Ω_{st}: for small or federal in-creases (Cengiz et al. (2019))
  - $treat^\tau_{st}$: whether the minimum wage was increased τ years from date t in state s
  - cluster standard error by states

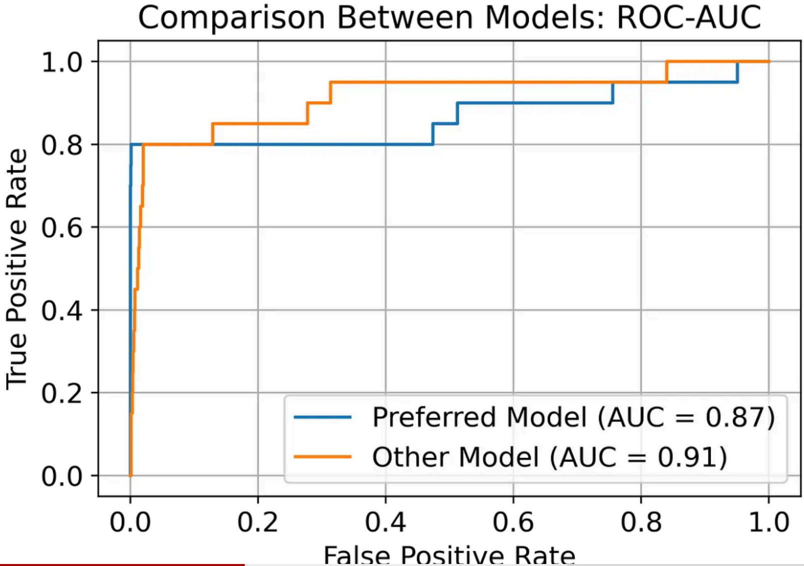
confusion matrix



confusion matrix

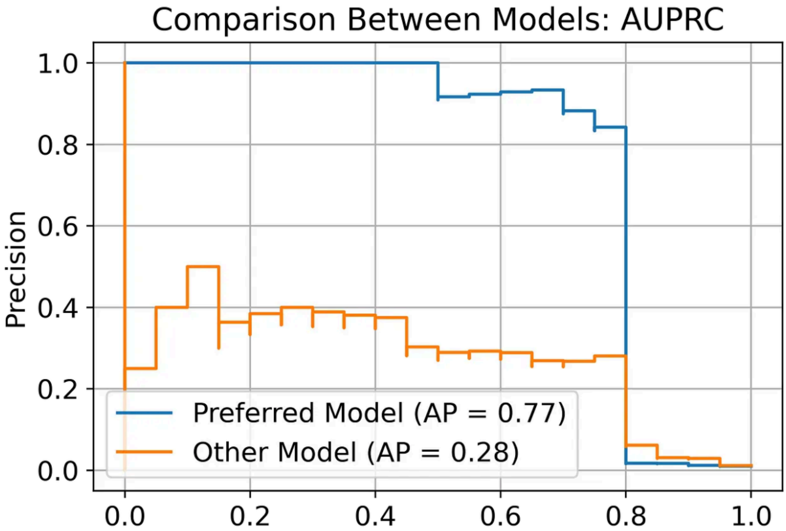
- Accuracy: $\frac{TP+TN}{TP+FN+FP+TN}$
- precision: $\frac{TP}{TP+FP}$
- recall(sensitivity, true positive rate): $\frac{TP}{TP+FN}$
- false positive rate: $\frac{FP}{FP+TN}$

ROC AUC



Comparison Between Models: ROC-AUC
True Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
Preferred Model (AUC = 0.87)
Other Model (AUC = 0.91)

AUPRC

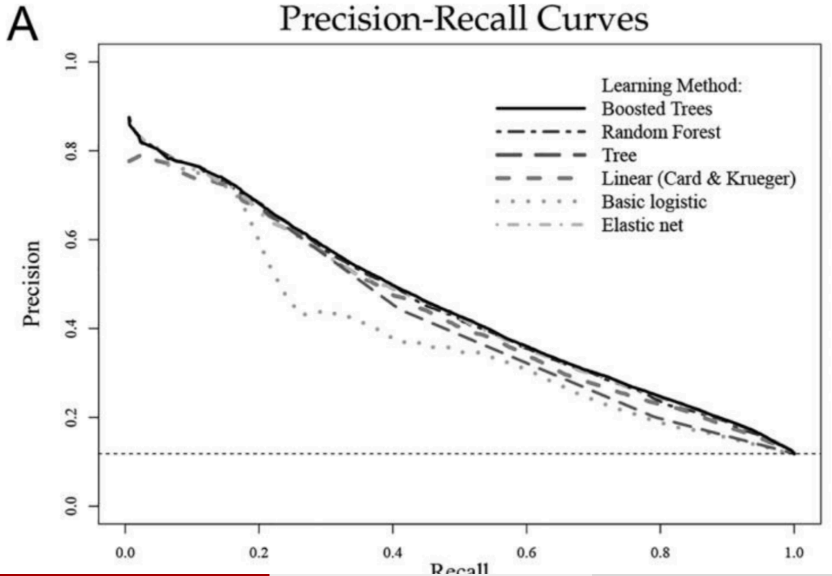


Comparison Between Models: AUPRC
Precision
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Preferred Model (AP = 0.77)
Other Model (AP = 0.28)

data and features

- minimum wage worker prediciton: 1979-2019 CPS-ORG
  - Education group
  - Age group
  - Gender
  - Rural residency
  - Martial(married and spouse is present)
  - Race
  - Hispanic
  - Veteran
- labor market outcomes estimation: 1979-2019 CPS-Basic

evaluation


A
Precision-Recall Curves
Learning Method:
Boosted Trees
Random Forest
Tree
Linear (Card & Krueger)
Basic logistic
Elastic net
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0

evaluation

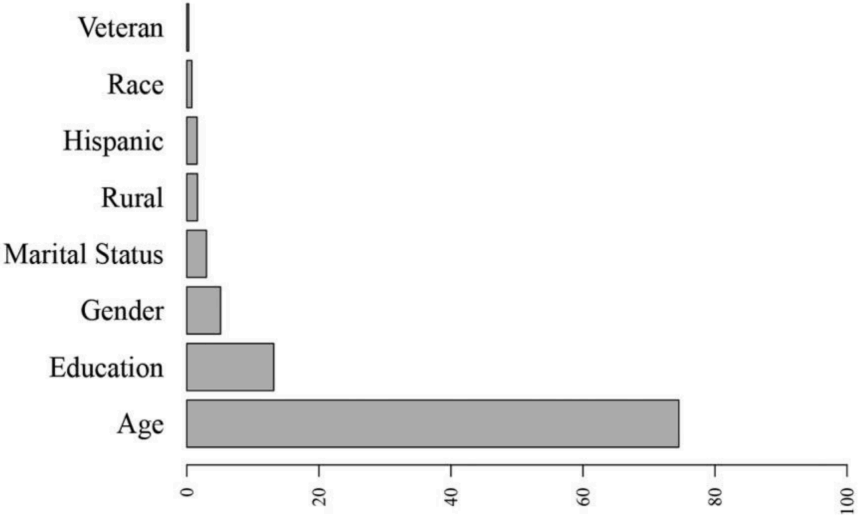
- high probability group
  - threshold probability 0.35
  - precision: 0.6
  - recall: 0.36
- high recall group
  - threshold probability 0.12
  - precision: 0.35
  - recall: 0.75
- low probability group: predicted probability ≤ 0.12

who is a minimum wage worker

Demographic Characteristics for Each Predicted Probability Decile

	Teen (1)	$20 \leq \text{Age} < 30$ (2)	LTHS (3)	HSG (4)	Female (5)	White (6)	Black or Hispanic (7)
Most likely decile	.719	.038	.752	.145	.592	.837	.244
Probability decile 9	.047	.405	.534	.238	.674	.847	.359
Probability decile 8	.004	.341	.344	.437	.594	.834	.243
Probability decile 7	.004	.298	.187	.575	.571	.833	.351
Probability decile 6	.000	.191	.085	.660	.673	.873	.150
Probability decile 5	.000	.187	.100	.475	.492	.784	.253
Probability decile 4	.000	.178	.067	.236	.512	.794	.237
Probability decile 3	.000	.162	.004	.297	.404	.865	.175
Probability decile 2	.000	.088	.000	.143	.385	.848	.122
Least likely decile	.000	.015	.000	.039	.314	.741	.134

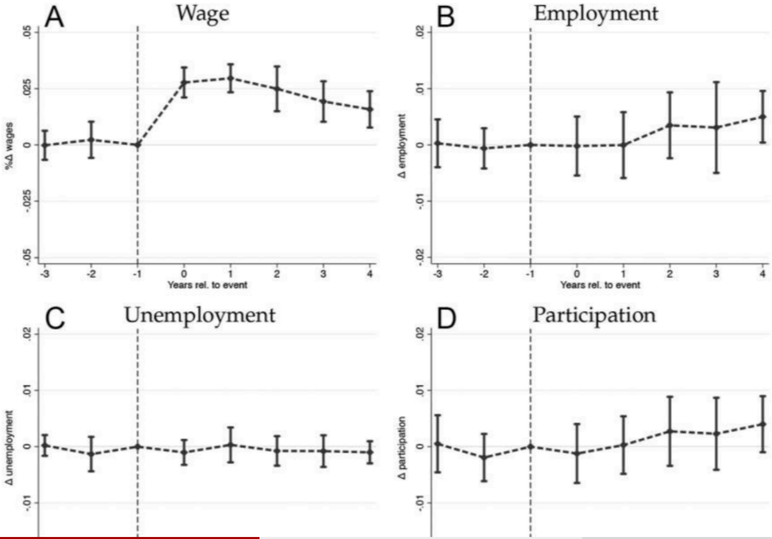
who is a minimum wage worker



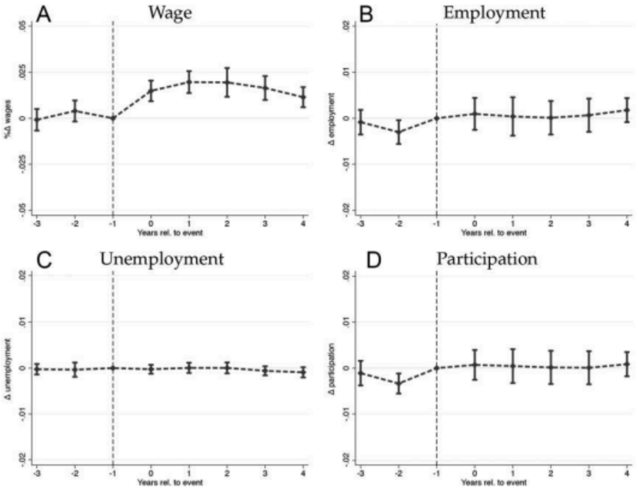
labor market outcome

- 5-year averaged posttreatment estimates: $\frac{1}{5}\Sigma_{\tau=0}^{4}(\beta_{\tau} - \beta_{-1})$
- wage
  - high probability: 2.3%(SE, 0.3%)
  - high recall: 1.6% (SE, 0.3%)
  - low probability: -0.1% (SE, 0.3%)

high probability estimation



high recall estimation



hands-on machine learning

Let's apply the framework using Taiwan's Data

data

- source: , 2000-2006
- features: countycat, sex, martial, educat, agecat
  - martial: married or not
  - educat
    - 1: below junior high
    - 2: high school
    - 3: at least college
  - agecat: group every 5 years from 15 to 70 as one category

tidymodels

pipeline for training a machine learning models

1. before training: preprocess and specify the model
   - recipes, parsnip
2. hyperparameters tuning and evaluation
   - rsample, tune, yardstick
3. prediction and evaluation
   - yardstick

data

```
> data
# A tibble: 153,134 × 6
   countycat    sex        martial educat agecat group
   <dbl+lbl>    <dbl+lbl>    <dbl>  <dbl>  <dbl> <dbl>
 1 22 [高雄市] 1 [男]            1      2     40     3
 2 22 [高雄市] 0 [女]            0      2     20     2
 3 22 [高雄市] 1 [男]            1      1     55     2
 4 22 [高雄市] 0 [女]            1      1     45     2
 5 22 [高雄市] 1 [男]            0      1     30     2
 6 22 [高雄市] 1 [男]            0      2     40     2
 7 22 [高雄市] 1 [男]            1      1     25     3
 8 22 [高雄市] 0 [女]            0      2     20     2
 9 22 [高雄市] 1 [男]            0      1     30     2
10 22 [高雄市] 0 [女]            0      2     20     3
# ℹ 153,124 more rows
# ℹ Use `print(n = ...)` to see more rows
```

helper function

```
# split
set.seed(20230225)
split <- initial_split(data, prop=0.8, strata=group)
data_train <- training(split)

formula <- group ~ countycat + sex + martial + educat + agecat

# helper function for tuning
create_workflow <- function(spec, preprocessor=formula){
    wf <- workflow() %>%
        add_model(spec) %>%
        add_formula(preprocessor)

    return(wf)
}
```

helper function

```r
tune_result <- function(wf, grid){
    set.seed(20230225)
    cv <- vfold_cv(data_train, v=10, strata=group)

    res <- wf %>%
        tune_grid(
            resamples=cv,
            grid=grid,
            metrics=metric_set(pr_auc),
            # control=control_grid(save_workflow=T)
        )

    return(res)
}
```

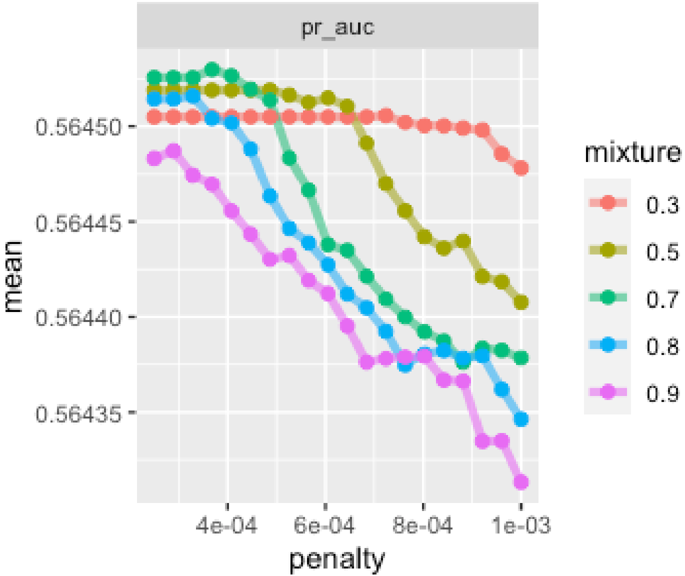
helper function

```r
# fit on the training split and calculate the metric on testing split
test <- function(model, name, preprocessor=formula){
    res <- last_fit(model, split=split, preprocessor=preprocessor) %>%
        collect_predictions() %>%
        mutate(model=name)

    return(res)
}
```

elastic net

```
en_spec <- multinom_reg(
    penalty=tune(),  # regularization
    mixture=tune()  # alpha: ratio of L1 and L2 regularization
) %>%
    set_mode("classification") %>%
    set_engine("glmnet")

wf <- create_workflow(en_spec)
grid <- expand.grid(
    penalty=seq(1e-5, 1e-2, length.out=20),
    mixture=c(0.1, 0.3, 0.5, 0.7, 0.9)
)
en_tune <- tune_result(wf, grid)
autoplot(en_tune)
elastic_net <- wf %>%
    finalize_workflow(select_best(en_tune, metric="pr_auc"))

# elastic_net <- update(en_spec, penalty=1e-5, mixture=0.1)

elastic_net_test <- test(elastic_net, "Elastic Net")
```

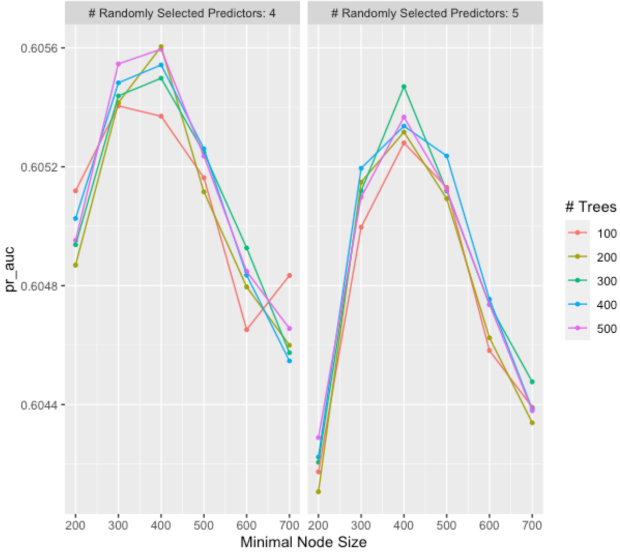
elastic net


pr_auc
0.56450
0.56445
0.56440
0.56435
mean
4e-04
6e-04
8e-04
1e-03
penalty
mixture
0.3
0.5
0.7
0.8
0.9

random forest

```r
rf_spec <- rand_forest(
    mtry=tune(),  # number of predictors
    trees=tune(),  # number of trees
    min_n=tune()  # stop split when the sample < min_n
) %>%
    set_mode("classification") %>%
    set_engine("ranger")

wf <- create_workflow(rf_spec)
grid <- expand.grid(
    mtry=c(4, 5),
    trees=c(100, 200, 300, 400, 500),
    min_n=c(200, 300, 400, 500, 600, 700)
)
rf_tune <- tune_result(wf, grid)
autoplot(rf_tune)
random_forest <- wf %>%
    finalize_workflow(select_best(rf_tune, metric="pr_auc"))

# random_forest <- update(rf_spec, mtry=4, trees=200, min_n=400)
```

random forest



XGBoost

```r
xgb_spec <- boost_tree(
    mtry=tune(),  # number of predictors
    trees=tune(),  # number of trees
    min_n=tune(),  # stop split when the sample < min_n(minimal node size)
    tree_depth=tune(),  # usually: 3 ~ 10
    learn_rate=tune(),
    loss_reduction=tune(),  # stop when loss reduction < loss_reduction(minimal loss reduction)
    sample_size=tune(),
    stop_iter=tune()  # stop when iterations > stop_iter(iterations before stopping)
) %>%
    set_mode("classification") %>%
    set_engine("xgboost")
```

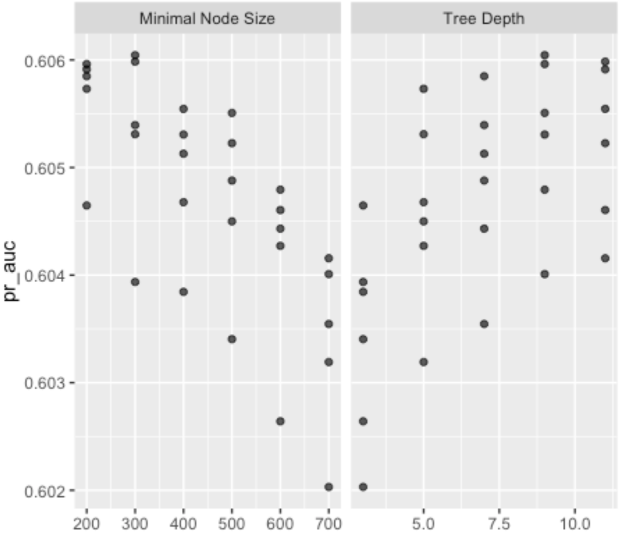
XGBoost

- tuning strategy
  1. tune min_n, tree_depth
  2. default suggestion
     - mtry: 80%of the predictors
     - trees: 200
     - loss_reduction = 0
     - sample_size: 80% of the sample size
     - stop_iter = 2000(should be large enough)
  3. tune mtry, sample_size
  4. tune trees, loss_reduction, stop_iter
  5. tune the learn_rate

XGBoost

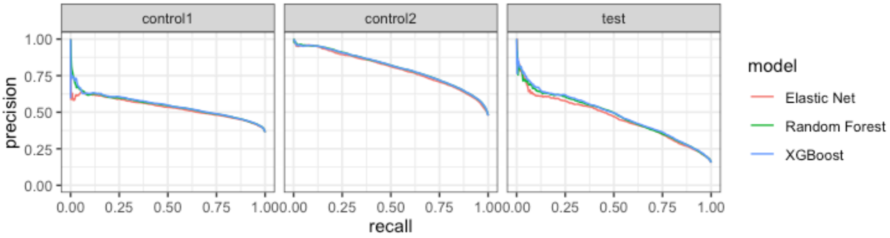
```r
wf <- create_workflow(xgb_spec)
grid <- expand.grid(
    mtry=5,
    trees=400,
    min_n=300,
    tree_depth=9,
    # learn_rate=0.3,
    loss_reduction=1e-2,
    sample_size=0.9,
    stop_iter=800,

    # mtry=c(3, 4, 5),
    # trees=c(50, 200, 400, 600),
    # min_n=c(200, 300, 400, 500, 600, 700),
    # tree_depth=c(3, 5, 7, 9, 11)
    learn_rate=seq(1e-1, 1, length.out=10)
    # loss_reduction=c(1e-1, 1e-2, 1e-3, 1e-4, 1e-5),
    # sample_size=c(0.3, 0.5, 0.6, 0.7, 0.8, 0.9)
    # stop_iter=c(50, 200, 400, 800)
)
xgb_tune <- tune_result(wf, grid)
autoplot(xgb_tune)

xgboost <- wf %>%
    finalize_workflow(select_best(xgb_tune, metric="pr_auc"))

# xgboost <- update(
#     xgb_spec, mtry=5, trees=400, min_n=300, tree_depth=9, learn_rate=0.3,
#     loss_reduction=1e-2, sample_size=0.9, stop_iter=800
```

XGBoost



Minimal Node Size
Tree Depth
pr_auc
0.606
0.605
0.604
0.603
0.602
200
300
400
500
600
700
5.0
7.5
10.0

AUPRC


control1
control2
test
1.00
0.75
0.50
0.25
0.00
precision
0.00
0.25
0.50
0.75
1.00
recall
model
Elastic Net
Random Forest
XGBoost