Machine Learning in Economics

Wooyong Park

Agenda

- Penalized Regressions
  - → Ridge Regression and LASSO
- Tree-based Models
  - → Decision Trees and Forests
  - → Causal Trees

Disclaimer

I brought quite an extensive combination of useful materials. You can find tons of materials to learn, understand, and practice ML. That also means that there are tons of algorithms and processes that we can't cover in these slides.

Indeed, important areas of ML such as deep learning, cross validation, theoretical aspects of bias-variance tradeoffs are dropped out here.

Also, in these slides, we only focus on supervised ML, where there are correct answers and we want the model to find them.

What is our primary interest?

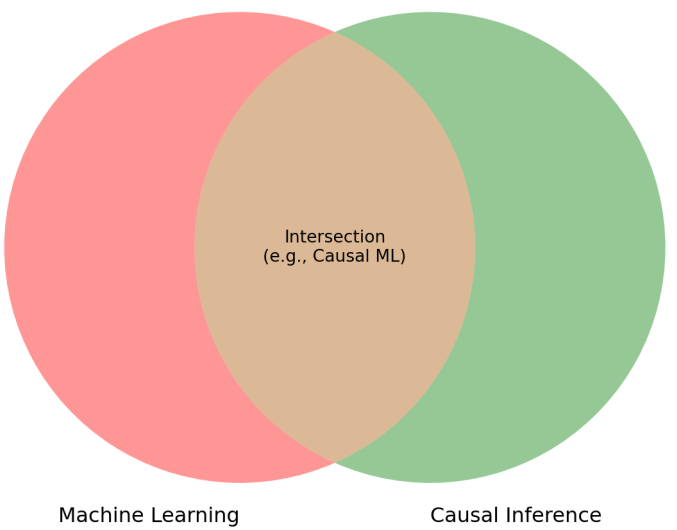
Machine Learning and Econometrics

Prediction

In ML and recent applied statistics, people want a better prediction of the future or a specific value from a new data, which often does not require statistical properties or CIs.

Estimation

In econometrics, people want to understand the DGP, obtain causal parameters and their statistical properties or CIs.



Machine Learning and Econometrics

Prediction

We find f such that

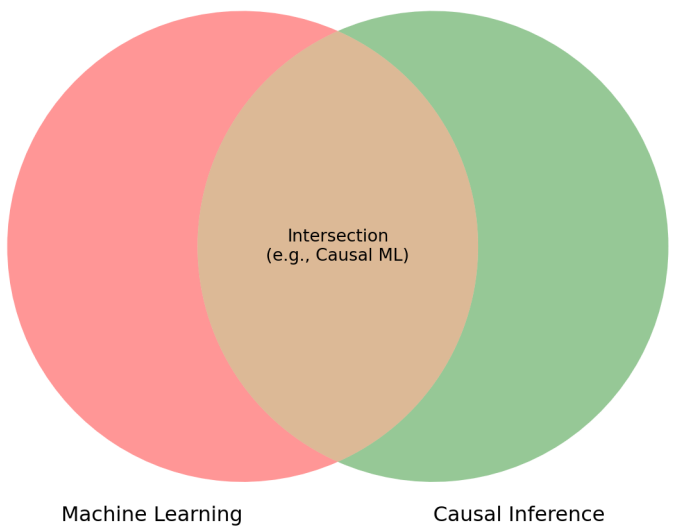
$$f(X_i) \approx y_i$$

Estimation

We find g such that

$g(X_i) \approx y_i$ and $\nabla g(X_i)$ captures the causal effect.

e.g. Hotel fees and reservation rates



Machine Learning and Econometrics

In most cases, we need both prediction and estimation.

Rather than relying on one strategy, why don't we allocate the tasks to whoever is good at them?

Example 1) Imputation with ML, and causal inference with IVs

Example 2) Select out covariates with LASSO to reduce the dimension, and do the rest with DiD

Example 3) Minimize the MSE on the suggested causal parameters rather than on outcomes.

Penalized Regressions

Background

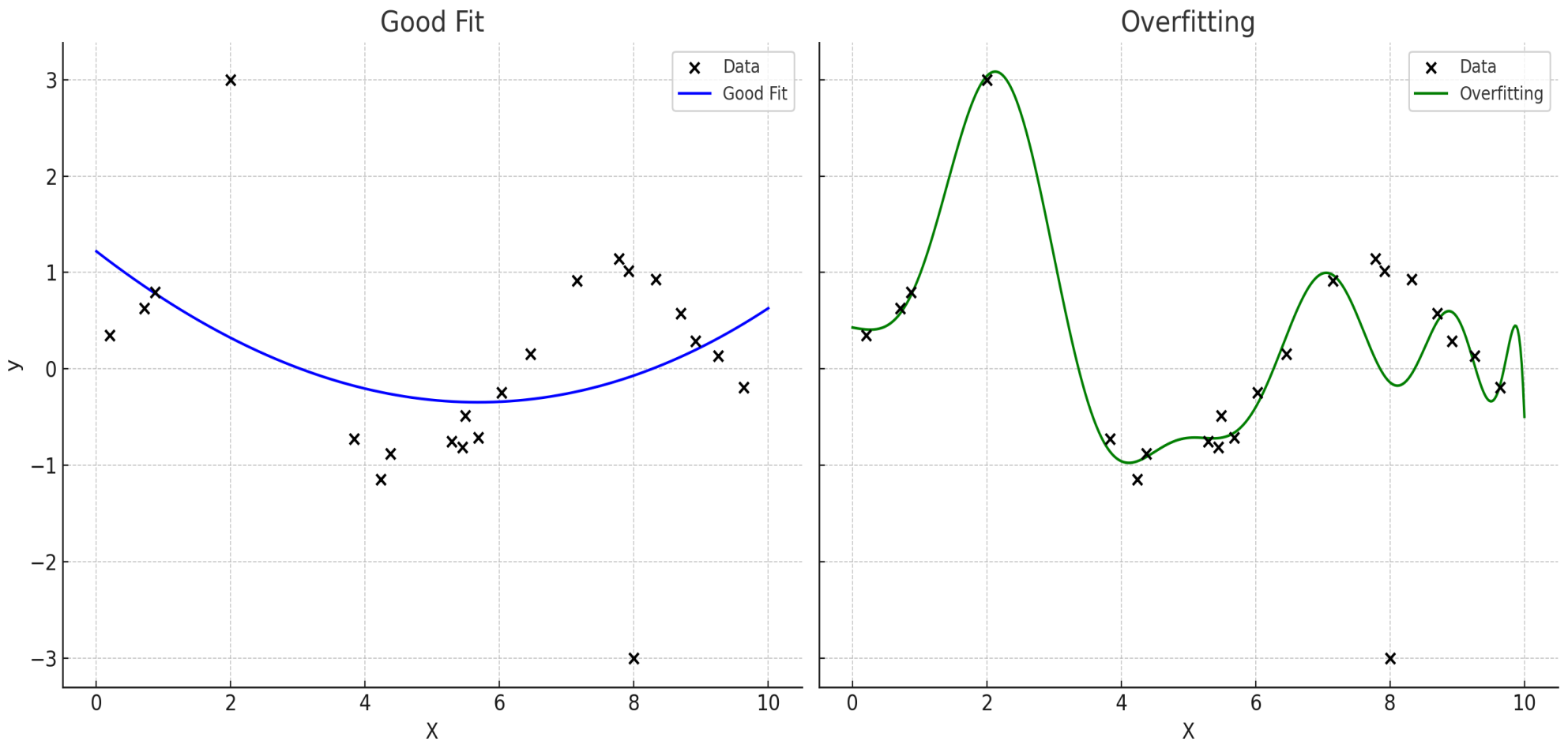
Penalized regressions, or regularized regressions, are regressions with loss functions modified to reduce the possibility of overfitting. Usually, large coefficients lead to overfitting. Thus, penalized regressions such as ridge regressions and LASSO(Least Absolute Shrinkage and Selection Operator) penalizes large sized coefficients, by including the size of coefficients into the loss function.

Loss Function of OLS

$$MSE = \sum_{i=1}^{n} (y_i - X_i' \beta)^2$$

Loss Function of Ridge/LASSO

The Problem with overfitting

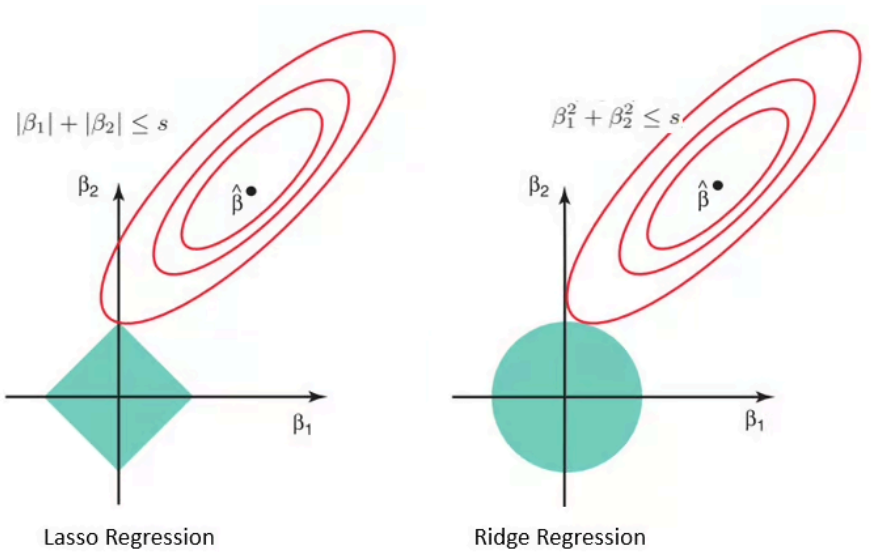


Ridge Regressions and LASSO

Details are in my **web document**.

Ridge

$$Loss(\beta;\alpha) = MSE + \alpha \cdot \sum_{k=1}^{K} \beta_k^2$$



Lasso Regression Ridge Regression

LASSO

$$Loss(\beta;\alpha) = MSE + \alpha \cdot \sum_{k=1}^{K} |\beta_k|$$

(Figure from Datacamp)

- Here, α is called a *hyperparameter*, a parameter that we choose to optimize the prediction.
- In the case of $\alpha = 0$, LASSO would be equivalent to OLS.
- LASSO shrinks the coefficients of less important features to zero

Practice with Python

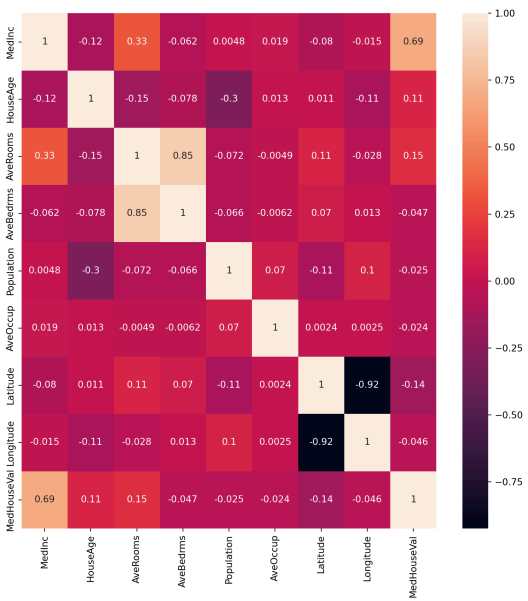
Loading Data

- Code

```
features:  Index(['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup',
       'Latitude', 'Longitude'],
      dtype='object')
```

- Code

```
target:  MedHouseVal
```



Data Splitting and Normalization

```
features = data.columns[0:8]
target = data.columns[-1]

#X and y values
X = data[features].values
y = data[target].values

#split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=17)

# Normalization
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

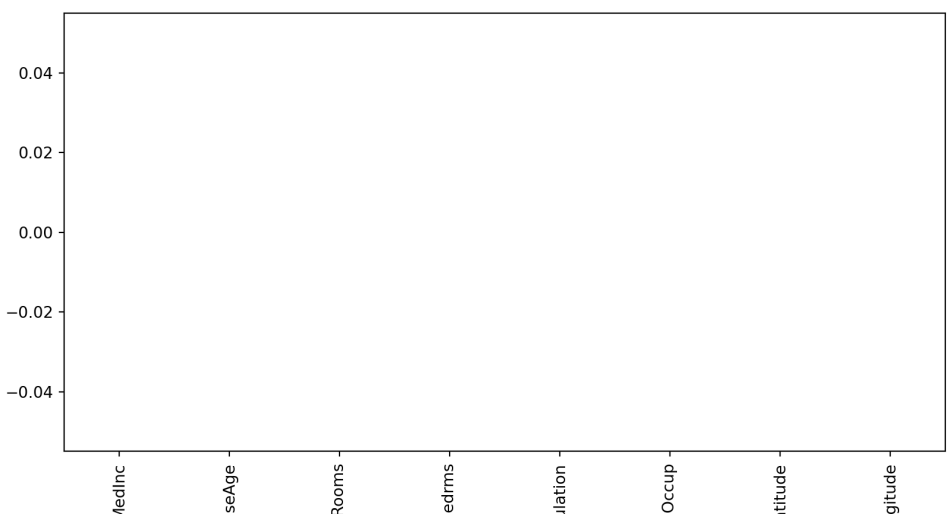
LASSO($\alpha = 3$)

```
#Lasso regression model

lasso = Lasso(alpha = 3)
lasso.fit(X_train,y_train)
```

Lasso

```
Lasso(alpha=3)
```

```

pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
```



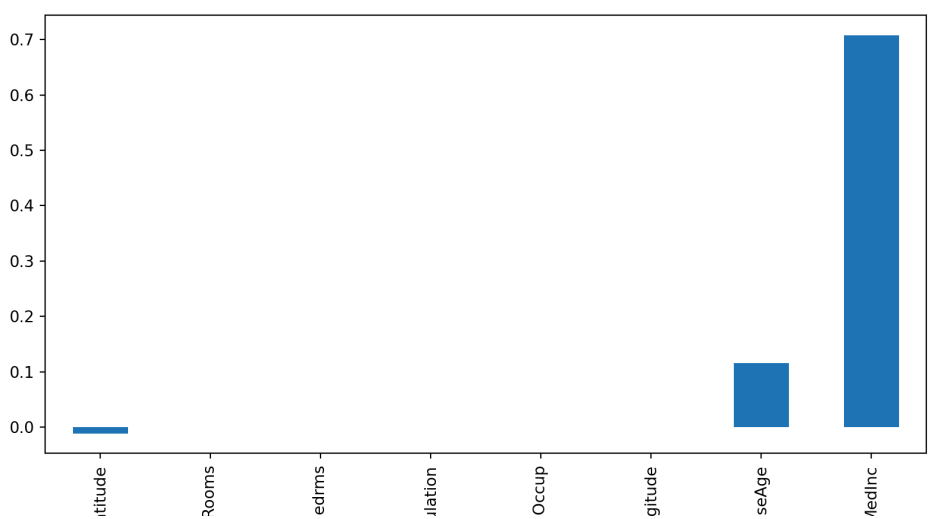
LASSO($\alpha = 0.1$)

```
#Lasso regression model

lasso = Lasso(alpha = 0.1)
lasso.fit(X_train,y_train)
```

Lasso

```
Lasso(alpha=0.1)
```

```

pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
```



Braghieri et al. (2022)

B. *Heterogeneity*

Heterogeneity by Predicted Susceptibility to Mental Illness.—In order to study whether the introduction of Facebook at a college led students on the margin of a depression diagnosis to take up depression-related services, we proceed in two steps: first, we estimate a least absolute shrinkage and selection operator (LASSO) to identify individuals who, based on baseline immutable characteristics, are more

susceptible to mental illness. Second, we show heterogeneous treatment effects based on our LASSO-predicted measure of susceptibility to mental illness.

The LASSO prediction is generated as follows: first, we construct an indicator that equals one if a student has ever been diagnosed with a mental health condition. Second, we consider a set of immutable individual-level characteristics (age, year in school, gender, race, an indicator for US citizenship, and height), generate all two-way interactions between these characteristics, and generate second- and third-order monomials of each characteristic. Third, we implement a LASSO procedure in the preperiod to predict our indicator for ever having been diagnosed with a mental health condition based on the immutable individual-level characteristics and functions thereof described above.

Braghieri et al. (2022)

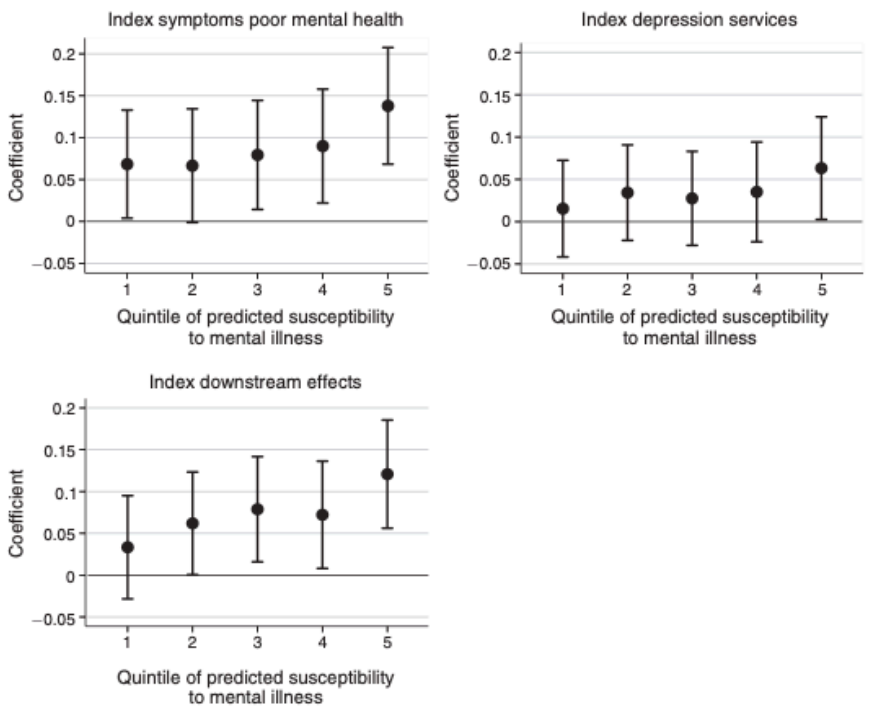


FIGURE 3. HETEROGENEOUS EFFECTS BY PREDICTED SUSCEPTIBILITY TO MENTAL ILLNESS

Notes: This figure explores the extent to which the effects of the introduction of Facebook at a college are heterogeneous depending on students' predicted susceptibility to mental illness. Specifically, it presents the estimates from equation (3) in which our indicator for post-Facebook introduction is interacted with a set of indicators for belonging to each quintile of a LASSO-predicted measure of susceptibility to mental illness. The outcome variable in the top-left panel is our index of symptoms of poor mental health; the outcome variable in the top-right panel is our index of depression services; the outcome variable in the bottom-left panel is our index measuring whether students reported that conditions related to poor mental health negatively affected their academic performance. All indices are standardized so that, in the preperiod, they have a mean of zero and a standard deviation of one. The estimates (also displayed in online Appendix Table A.5) are obtained using our preferred specification, namely the one including survey-wave fixed effects, college fixed effects, and controls. Our controls consist of age, age squared, gender, indicators for year in school (freshman, sophomore, junior, senior), indicators for race (White, Black, Hispanic, Asian, Indian, and other), and an indicator for international student. For a detailed description of the outcome, treatment, interaction, and control variables, see online Appendix Table A.31. The bars represent 95 percent confidence intervals. Standard errors are clustered at the college level.

Decision Trees

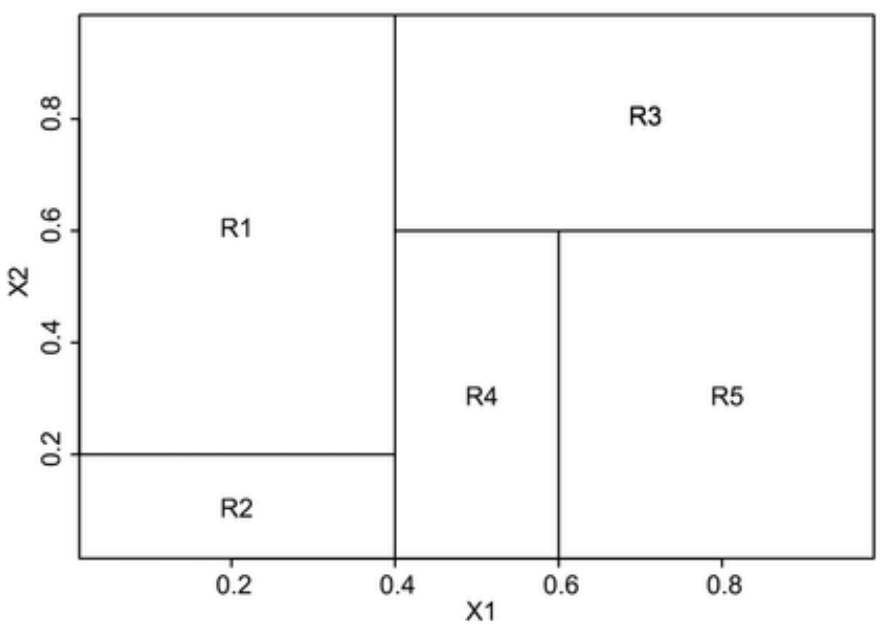
Decision Trees

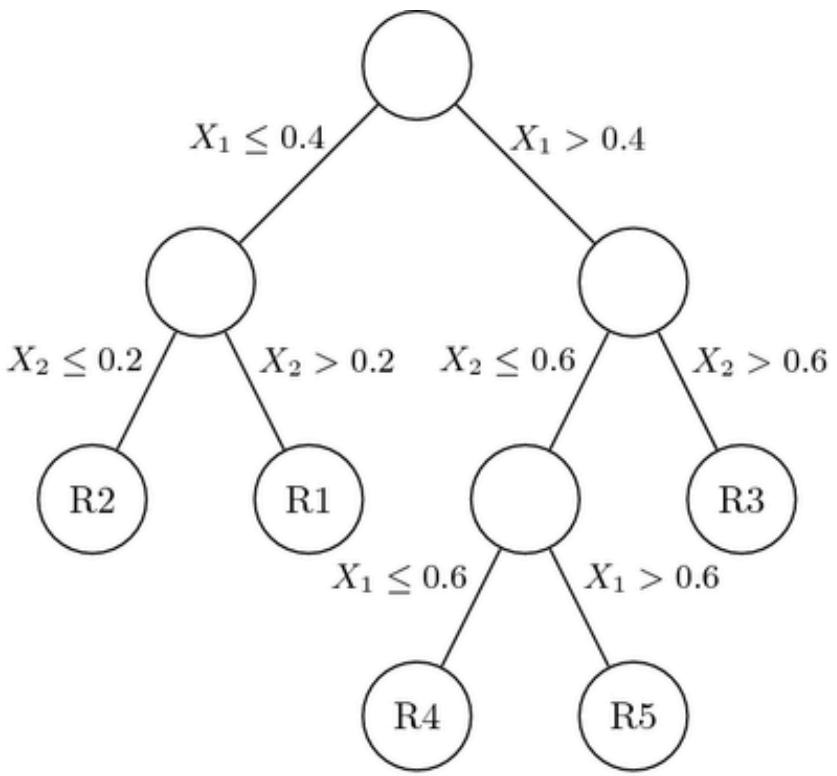
Trees are a supervised ML method to predict either a continuous outcome(**regression trees**) or a discrete outcome(**classification trees**). One favorable aspect is that they're "white-box" mechanism, which means we can understand how they chose the prediction.

There are also detrimental disadvantages, including the high variance of a model and ensembled models used to reduce that variance making the mechanism difficult to interpret.

Decision Trees

Trees recursively partition the covariate space so that going down to each node, the variance of the $\boldsymbol{y}$ value in the subset is low and $\boldsymbol{y}$'s are better predicted.





Practice with R/Python

Loading Data

R | Python

```r
# install.packages('tidyverse')
library(tidyverse)
library(tidymodels)
breast_cancer <- read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/breast-ca
colnames(breast_cancer) <- c('id', 'clump', 'cell_size', 'cell_shape', 'adhesion', 'epithlial'
head(breast_cancer)
```

```
# A tibble: 6 × 11
       id clump cell_size cell_shape adhesion epithlial bare_nuclei chromatin
    <dbl> <dbl>     <dbl>      <dbl>    <dbl>     <dbl> <chr>            <dbl>
1 1000025     5         1          1        1         2 1                    3
2 1002945     5         4          4        5         7 10                   3
3 1015425     3         1          1        1         2 2                    3
4 1016277     6         8          8        1         3 4                    3
5 1017023     4         1          1        3         2 1                    3
6 1017122     8        10         10        8         7 10                   9
# ℹ 3 more variables: normal_nucleoli <dbl>, mitoses <dbl>, class <dbl>
```

Preprocessing

R | Python

```r
breast_cancer <- breast_cancer %>%
  mutate(bare_nuclei = if_else(bare_nuclei =='?', NA, bare_nuclei)) %>%
  filter(!is.na(bare_nuclei)) %>%
  mutate(bare_nuclei = as.numeric(bare_nuclei),
         class = factor(class)) #2: benign, 4: malignant
```

Data Splitting and Normalization

R | Python

```r
# standardize X variables
breast_cancer <- breast_cancer %>%   mutate(across(-class, ~ scale(.) %>% as.vector(), .names =

# split the data
breast_cancer_split <- initial_split(breast_cancer, prop = 0.7, strata = class)

breast_cancer_train <- training(breast_cancer_split)
breast_cancer_test <- testing(breast_cancer_split)
```

Model Fit and Results

R | Python

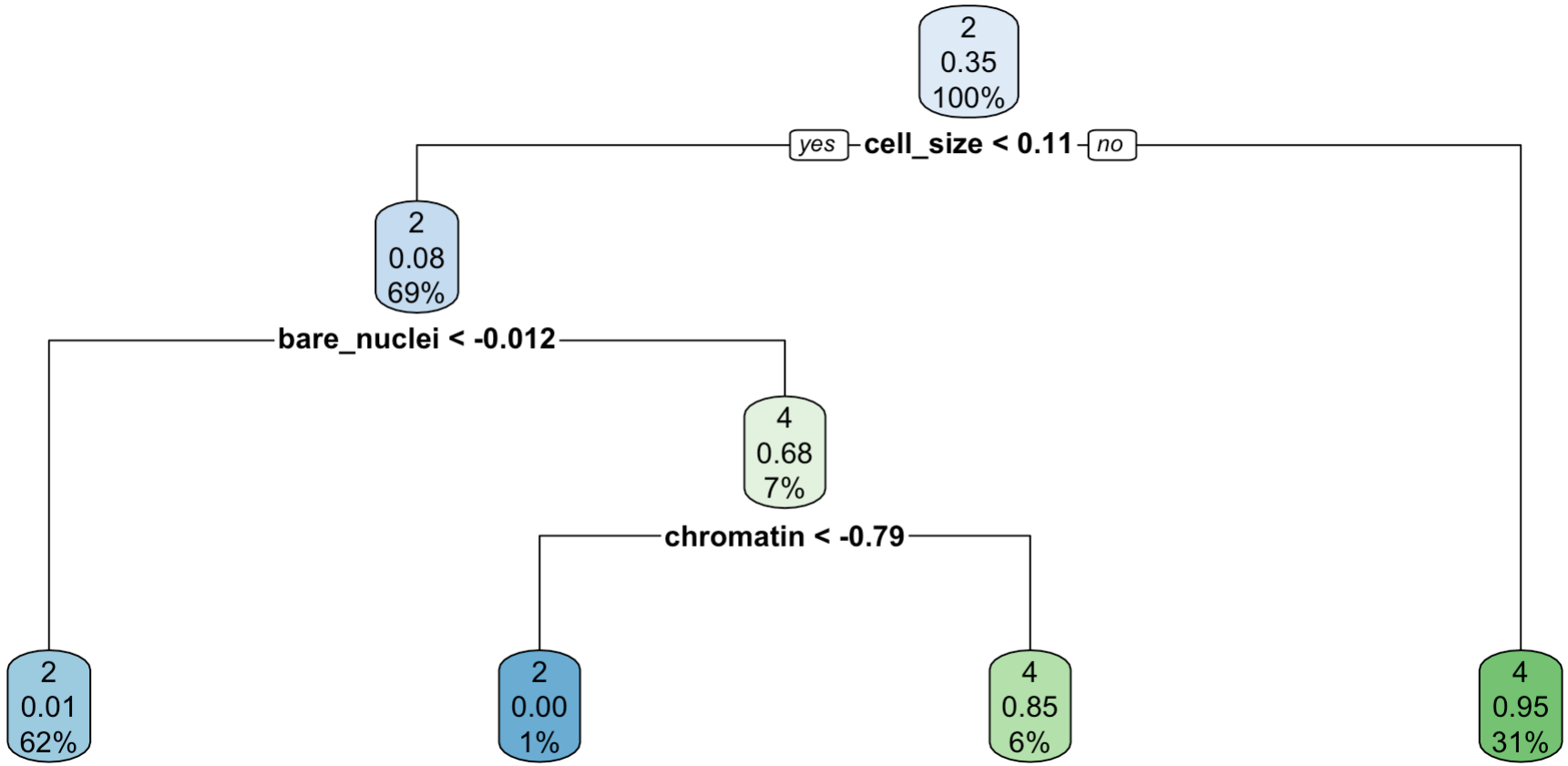
```r
# instantiate the engine
tree_spec <- decision_tree() %>%
  set_engine("rpart") %>%
  set_mode("classification")

# fit the model to the training data
breast_cancer_fit <- tree_spec %>%
```

```
# A tibble: 206 × 2
   .pred_class truth
   <fct>       <fct>
 1 2           2
 2 4           2
 3 4           2
 4 2           2
 5 2           4
 6 4           4
 7 4           4
 8 4           4
 9 2           2
10 4           4
# ℹ 196 more rows
```

Resulting Tree

R | Python

```r
# install.packages('rpart.plot')
library(rpart.plot)

rpart_tree <- extract_fit_engine(breast_cancer_fit)
rpart.plot(rpart_tree)
```


2
0.35
100%
yes
cell_size < 0.11
no
2
0.08
69%
bare_nuclei < -0.012
4
0.68
7%
chromatin < -0.79
2
0.01
62%
2
0.00
1%
4
0.85
6%
4
0.95
31%

Model Performance

R | Python

```r
# Confusion Matrix
conf_mat(pred, estimate = .pred_class, truth = truth)
```

```
          Truth
Prediction   2   4
         2 129   6
         4   5  66
```

```r
# Accuracy
accuracy(pred, estimate = .pred_class, truth = truth)
```

```
# A tibble: 1 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.947
```

Brief Address on Bagged Trees and Random Forests

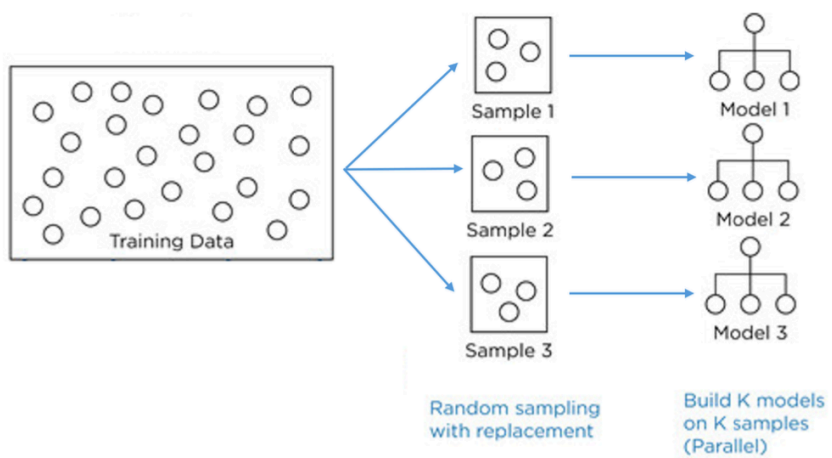
Bagged Trees

Bagging = Bootstrap + Aggregation

1. Bootstrapping: Sample *with* replacement → Multiple Training sets → Multiple Trees

Aggregation

2. average(in regression), majority vote(in classification)



Brief Address on Bagged Trees and Random Forests

Random Forests

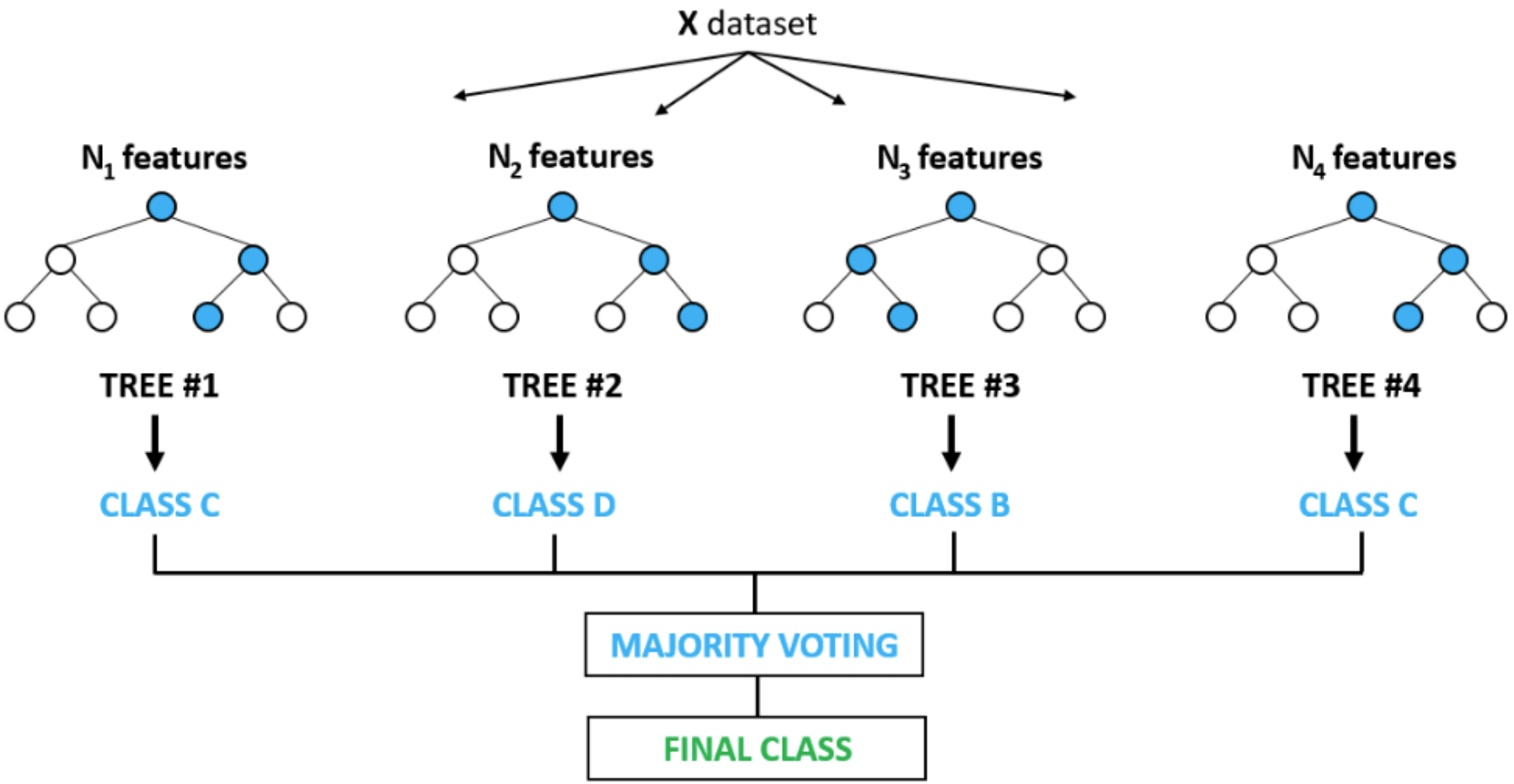
Multiple trees from bagged trees don't always lead to a greater performance, as the trees share the same predictors (it leads to correlation between trees).

Rather, we'd like to randomly select the set of predictors across trees, and then perform BAGG.

This is what random forest does.

Brief Address on Bagged Trees and Random Forests

Random Forests



Heterogeneous Treatment Effects

HTE has a wide array of meanings, but in essence, they all boil down to having multiple values of treatment effects, conditional on the characteristics of an observation.

Examples

1. Medicaid for 80+ vs Medicaid for 40-
2. ATE at a certain period of a certain group(Staggered Treatments)

CATE

The standard approach for HTEs are CATE(conditional average treatment effect).

$$\tau(x) := E[Y_i(1) - Y_i(0)|X_i = x]$$

X_i could be the age, sex, or any characteristics that could affect the treatment effect. In real practice, we often only allow “group” variables(i.e. classifying each sample into different subgroups and calculate their ATEs).

Let’s consider a simple case of a dummy, such as sex.

*Unconfoundedness

Just before we move on, we're going to assume that the treatment is randomized (conditional on the covariates).

$$W_i \perp (Y_i(0), Y_i(1)) | X_i$$

Pre-specified Hypotheses

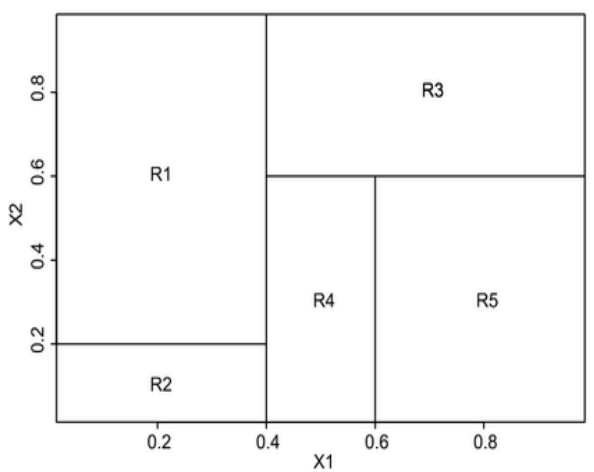
The question is, *"Is the treatment effect heterogeneous?"*

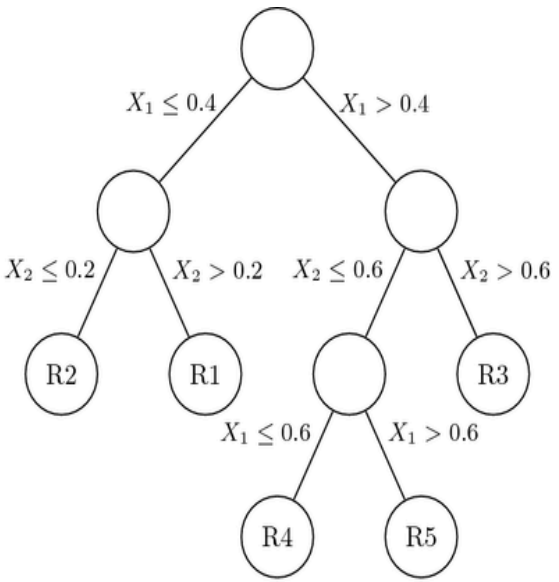
$$H_0 : E[Y(1) - Y(0)|G_i = 1] - E[Y(1) - Y(0)|G_i = 0] = 0$$

What happens if there are multiple values of G_i's to be considered? Also, what dimensions of covariates should we suspect HTEs?

They require a much more complex approach, which will not be discussed in these slides. And instead of the researcher-specified hypotheses, we want the data to specify the hypotheses by itself(**Data-driven hypotheses**).

Data-driven Hypotheses with Causal Trees





We are now dividing the sample into different subgroups defined by the covariates and the decision tree. Thus, each subgroup has a corresponding leaf, and HTEs can now be called **leaf effects**.

Causal Trees

These practices rely heavily on GCSI lab's **machine learning tutorials**, which also relies on the **General Social Surveys**.

```r
# The causalTree package is not in CRAN, the most common R repository.
# To install it, uncomment the next lines as appropriate.
# install.packages("devtools")  # if you don't have this installed yet.
# devtools::install_github('susanathey/causalTree')
library(causalTree)

# use e.g., install.packages("tidyverse") to install any of the following packages.
```

```r
# Read in data
data <- read.csv("https://docs.google.com/uc?id=1AQva5-vDlgBcM_Tv9yrO8yMYRfQJgqo_&export=downl
n <- nrow(data)

# Treatment: does the the gov't spend too much on "welfare" (1) or "assistance to the poor" (0
treatment <- "w"

# Outcome: 1 for 'yes', 0 for 'no'
outcome <- "y"

# Additional covariates
covariates <- c("age", "polviews", "income", "educ", "marital", "sex")
```

Causal Trees

```
1 head(data)
```

```
  X y w age polviews income educ marital sex
1 1 0 0  28        4     11   14       5   1
2 2 1 0  54        6     12   16       2   2
3 3 1 0  44        2     12   16       5   2
4 6 0 0  47        1      5   10       4   1
5 7 0 1  19        4      9   10       5   2
6 8 0 0  36        2      8   18       1   1
```

Causal tree divides the sample into three subsets: S^{tr}(`split`), S^{est}(`est`), and S^{te}(`test`).

- S^{tr}: the training set which constructs the model(or equivalently, the tree) to maximize heterogeneity in treatment effect estimates across leaves
- S^{est}: the set used to estimate the HTE(more rigorously, the TE of a leaf)
- S^{te}: the test set to assess the model performance

The algorithm is quite recursive. There are some aspects I don't understand well. I'd love to hear both questions and clarifications.

1. Tree Splitting

From the root, the data will be split into two groups to best minimize the risk function. There are four possible splitting rules, with **CT-H** being used most commonly and **Transformed Outcome** being the most intuitive. We will only cover **TO**.

Transformed Outcomes

Let

$$Y_i^* = \begin{cases} Y_i/p & (W_i = 1) \\ -Y_i/(1-p) & (W_i = 0) \end{cases}$$

where $p = N_{treat}/N$

is the treatment probability independent of the covariates.

Straightforward that

$$E[Y_i^* | X_i = x] = \tau_i(x)$$

1. Tree Splitting

From the root, the data will be split into two groups to best minimize the risk function. There are four possible splitting rules, with **CT-H** being used most commonly and **Transformed Outcome** being the most intuitive. We will only cover **TO**.

Risk Function

$$\hat{MSE}\left(S^{tr}, S^{tr}, \Pi\right) = \frac{1}{N^{tr}} \sum_{i \in S^{tr}} \{(Y_i^* - \hat{\tau}(X_i; S^{tr}, \Pi))^2 - Y_i^{*2}\}$$

where $\hat{\tau}(X_i; S^{tr}, \Pi)$

is similarily defined as in the next slide, but only made with the S^{tr} samples.

RMK) In each leaf, there must be at least one treated observation and one controlled observation in the S^{est} set to estimate $\tau(x)$

2. Leaf Effects

The estimates of leaf effects are obtained from S^{est}.

Given a tree Π (derived from S^{tr}),

$$\hat{\tau}(X_i; S^{est}, \Pi) =$$

$$\frac{1}{\sum_{i \in S^{est} \cap l(X_i;\Pi)} W_i} \sum_{i \in S^{est} \cap l(X_i;\Pi)} W_i Y_i - \frac{1}{\sum_{i \in S^{est} \cap l(X_i;\Pi)} (1 - W_i)} \sum_{i \in S^{est} \cap l(X_i;\Pi)} (1 -$$

3. Model Evaluation and Pruning

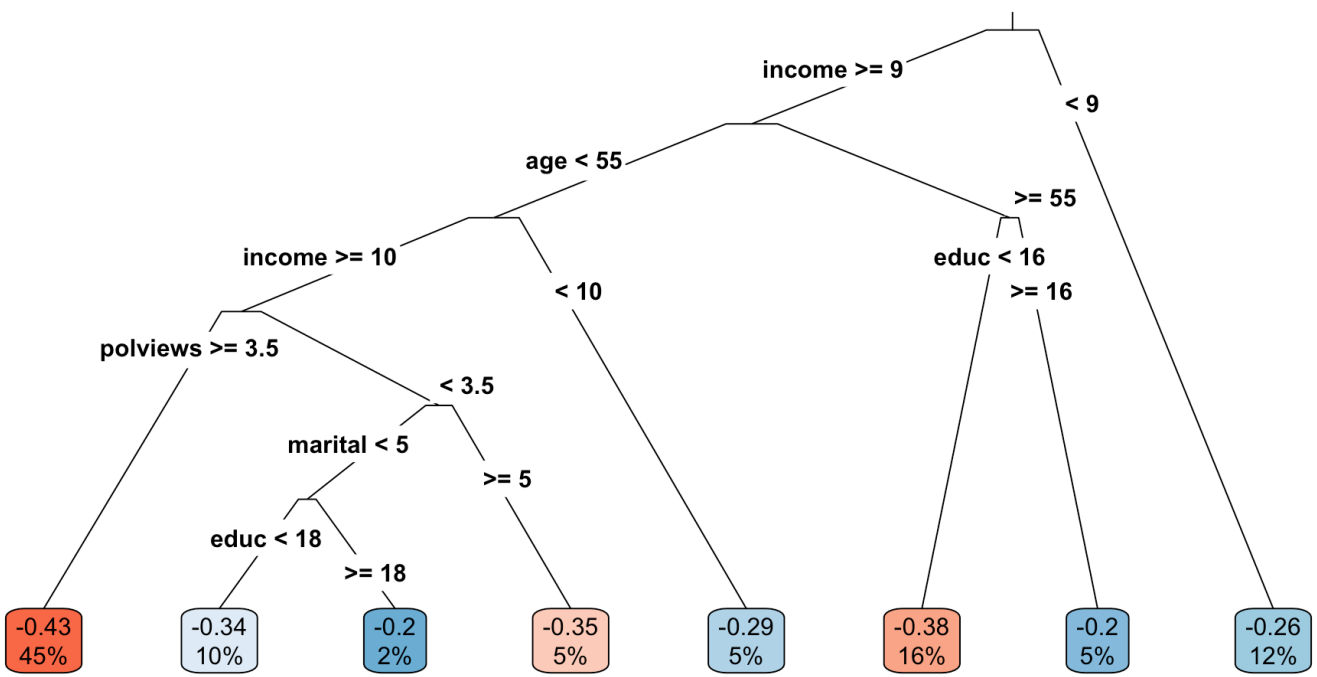
- We calculate the MSE_τ in the test set.
- Use cross-validation to select the right depth d^* of the partition that minimizes those MSE_τ's.
- Select partition Π^* by pruning Π to the depth d^*, pruning leaves that provide the smallest improvement in the goodness of fit.

Application

```
# Only valid for randomized data!

fmla <- paste(outcome, " ~", paste(covariates, collapse = " + "))

# Dividing data into three subsets
indices <- split(seq(nrow(data)), sort(seq(nrow(data)) %% 3))
names(indices) <- c('split', 'est', 'test')

# Fitting the forest
ct.unpruned <- honest.causalTree(
  formula=fmla,            # Define the model
  data=data[indices$split,],
  treatment=data[indices$split, treatment],
  est_data=data[indices$est,],
  est_treatment=data[indices$est, treatment],
  minsize=1,                # Min. number of treatment and control cases in each leaf
  HonestSampleSize=length(indices$est), #  Num obs used in estimation after splitting
  # We recommend not changing the parameters below
  split.Rule="CT",           # Define the splitting option
```

Application

```
rpart.plot(
  x=ct.pruned,          # Pruned tree
  type=3,               # Draw separate split labels for the left and right directions
  fallen=TRUE,          # Position the leaf nodes at the bottom of the graph
  leaf.round=1,         # Rounding of the corners of the leaf node boxes
  extra=100,            # Display the percentage of observations in the node
  branch=.1,            # Shape of the branch lines
  box.palette="RdBu") # Palette for coloring the node
```



References and Resources

Documents

- Causal Tree Documentation
- GCSI Lab Tutorial
- My Tutorial!

Courses

- ML & CI: A Short Course