Studying Social Inequality with Data Science

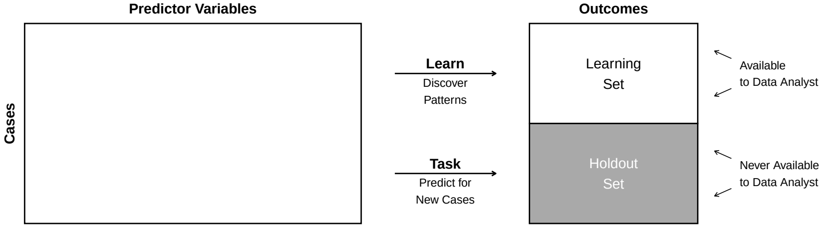
INFO 3370 / 5371
Spring 2023

Sample splitting

The model selection problem

In supervised machine learning, the goal is to

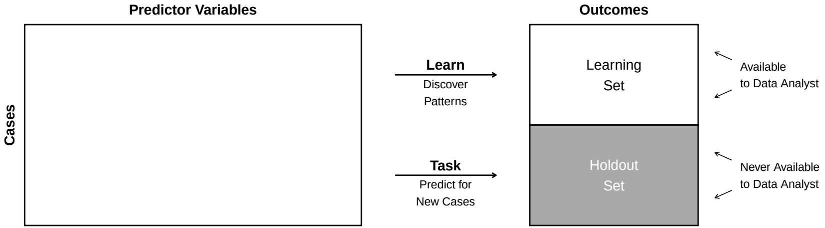
- learn patterns in the available data
- predict outcomes for previously unseen cases



The model selection problem

In supervised machine learning, the goal is to

- learn patterns in the available data
- predict outcomes for previously unseen cases



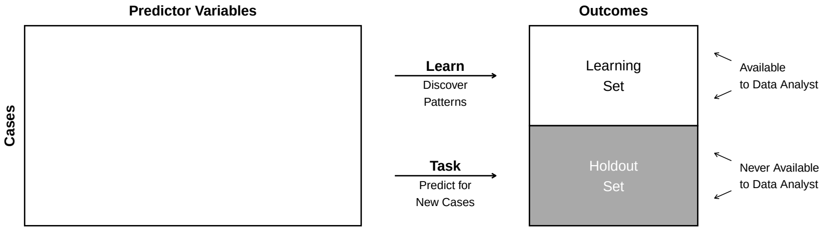
How do we know which method will do this well?

Key principle

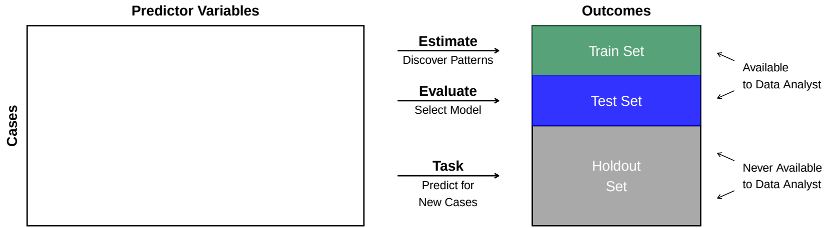
When a task involves unseen data,

- try to mimic that task with data you already have
- pick the method that performs best on your mimic task

Goal: Predict the unseen outcomes in a holdout set



Mimic the task: Sample split



Sample split in R

1. Load the data
2. Create a train-test split
3. Learn candidate prediction functions in the train set
4. Evaluate predictive performance in the test set
5. Estimate the chosen model in the full learning set and predict in the holdout set

Prepare environment

You'll want

- the `tidyverse` package
- the `rsample` package, which we will use to make the split
- use `set.seed()` with a number of your choosing to ensure reproducibility despite random sampling

```
library(tidyverse)
library(rsample)
set.seed(14850)
```

1. Load the data

```
learning <- read_csv("learning.csv")
holdout_public <- read_csv("holdout_public.csv")
```

2. Create a train-test split

2. Create a train-test split

In the `rsample` package,

- the `initial_split()` function will create a split

```
learning_split <- learning %>%
  initial_split(prop = 0.5)
```

2. Create a train-test split

In the `rsample` package,

- the `initial_split()` function will create a split

```
learning_split <- learning %>%
  initial_split(prop = 0.5)
```

- the `training()` and `testing()` functions will create data frames

```
train <- training(learning_split)
test <- testing(learning_split)
```

3. Learn candidate prediction functions on the train set

We will illustrate with OLS.

3. Learn candidate prediction functions on the train set

We will illustrate with OLS. We will consider

1. parent income
2. parent income + race + sex
3. parent income $\times$ race $\times$ sex

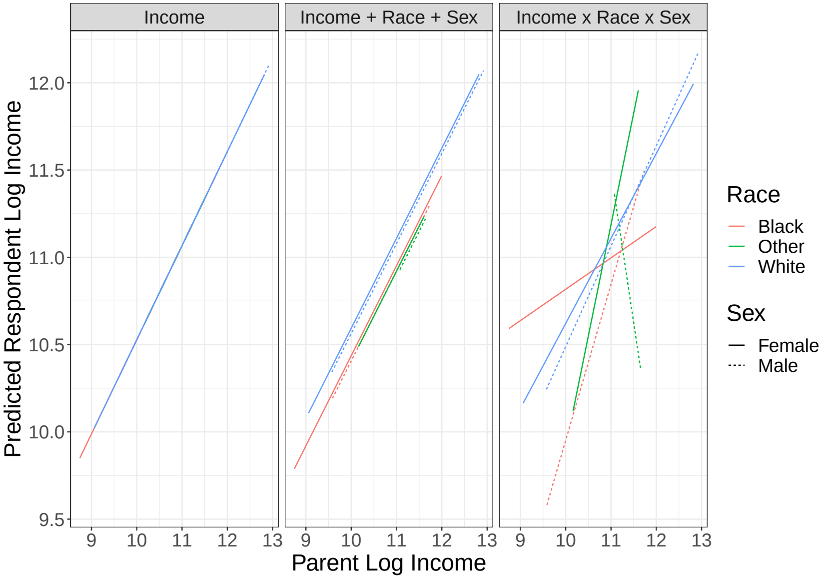
3. Learn candidate prediction functions on the train set

We will illustrate with OLS. We will consider

1. parent income
2. parent income + race + sex
3. parent income $\times$ race $\times$ sex

```
candidate_1 <- lm(g3_log_income ~ g2_log_income,
                  data = train)
candidate_2 <- lm(g3_log_income ~ g2_log_income +
                    race + sex,
                  data = train)
candidate_3 <- lm(g3_log_income ~ g2_log_income *
                    race * sex,
                  data = train)
```

3. Learn candidate prediction functions on the train set



4. Evaluate predictive performance on the test set

```
fitted <- test %>%
  mutate(candidate_1 = predict(candidate_1,
                               newdata = test),
         candidate_2 = predict(candidate_2,
                               newdata = test),
         candidate_3 = predict(candidate_3,
                               newdata = test)) %>%
  pivot_longer(cols = starts_with("candidate"),
               names_to = "model",
               values_to = "yhat")
```

4. Evaluate predictive performance on the test set

```
fitted %>%
  group_by(model) %>%
  mutate(error = g3_log_income - yhat) %>%
  mutate(squared_error = error ^ 2) %>%
  summarize(mse = mean(squared_error))
```

```
## # A tibble: 3 x 2
##   model         mse
##   <chr>       <dbl>
## 1 candidate_1 0.439
## 2 candidate_2 0.437
## 3 candidate_3 0.477
```

4. Evaluate predictive performance on the test set

```
fitted %>%
  group_by(model) %>%
  mutate(error = g3_log_income - yhat) %>%
  mutate(squared_error = error ^ 2) %>%
  summarize(mse = mean(squared_error))
```

```
## # A tibble: 3 x 2
##   model         mse
##   <chr>       <dbl>
## 1 candidate_1 0.439
## 2 candidate_2 0.437
## 3 candidate_3 0.477
```

Candidate 2 wins!

Side note: Train versus test set error

Side note: Train versus test set error

```
## # A tibble: 3 x 3
##   model                  train_set_mse test_set_mse
##   <chr>                          <dbl>        <dbl>
## 1 Income                         0.474        0.439
## 2 Income + Race + Sex            0.472        0.437
## 3 Income x Race x Sex            0.462        0.477
```

Side note: Train versus test set error

```
## # A tibble: 3 x 3
##   model                  train_set_mse test_set_mse
##   <chr>                          <dbl>        <dbl>
## 1 Income                         0.474        0.439
## 2 Income + Race + Sex            0.472        0.437
## 3 Income x Race x Sex            0.462        0.477
```

What happened?

Side note: Train versus test set error

```
## # A tibble: 3 x 3
##   model                   train_set_mse test_set_mse
##   <chr>                           <dbl>        <dbl>
## 1 Income                          0.474        0.439
## 2 Income + Race + Sex             0.472        0.437
## 3 Income x Race x Sex             0.462        0.477
```

What happened? Overfitting.

Side note: Train versus test set error

```
## # A tibble: 3 x 3
##   model                     train_set_mse test_set_mse
##   <chr>                             <dbl>        <dbl>
## 1 Income                            0.474        0.439
## 2 Income + Race + Sex               0.472        0.437
## 3 Income x Race x Sex               0.462        0.477
```

What happened? Overfitting.

- candidate 3 is very flexible

Side note: Train versus test set error

```
## # A tibble: 3 x 3
##   model                   train_set_mse test_set_mse
##   <chr>                           <dbl>        <dbl>
## 1 Income                          0.474        0.439
## 2 Income + Race + Sex             0.472        0.437
## 3 Income x Race x Sex             0.462        0.477
```

What happened? Overfitting.

- candidate 3 is very flexible
- discovers patterns that do not generalize

Side note: Train versus test set error

```
## # A tibble: 3 x 3
##   model                  train_set_mse test_set_mse
##   <chr>                          <dbl>        <dbl>
## 1 Income                         0.474        0.439
## 2 Income + Race + Sex            0.472        0.437
## 3 Income x Race x Sex            0.462        0.477
```

What happened? Overfitting.

- candidate 3 is very flexible
- discovers patterns that do not generalize
- performs poorly in test (and holdout)

5. Estimate in all of learning. Predict in the holdout set

With our chosen model, now estimate with all the data we have

```
chosen <- lm(g3_log_income ~ g2_log_income +
               race + sex,
             data = learning)
```

5. Estimate in all of learning. Predict in the holdout set

With our chosen model, now estimate with all the data we have

```
chosen <- lm(g3_log_income ~ g2_log_income +
               race + sex,
             data = learning)
```

Predict for the holdout set

```
predicted <- holdout_public %>%
  mutate(predicted = predict(chosen,
                             newdata = holdout_public))
```

Summary: Mimic the task with data you have