

# Data Management

## Supervised Machine Learning: Regressions



Malka Guillot

HEC Liège | ECON2306

# Table of contents

1. Introduction
2. Overview of ML process
3. Sample splitting to measure model fit
4. Pre-processing
5. Regression algorithms

## References

-  Békés & Kézdi (2022) 7.1->7.8 & 10.1->10.4
-  introduction to Statistical Learning chap 1. & 2 & 5.1
- Kleinberg, Ludwig, Mullainathan, and Obermeyer (2015), "Prediction Policy Problems." American Economic Review, 105 (5), pp. 491-95.
- Mullainathan and Spiess (2017), "Machine Learning: An Applied Econometric Approach", Journal of Economic Perspectives, 31 (2), pp. 87-106,

# Introduction

---

## [Motivation] Setting

- **Output variable** (or dependent variable)  $y$ 
  - Dependent variable, target variable, outcomes, labels
- Set of **input variables**  $X = (x_1, x_2, \dots, x_n)$ 
  - Explanatory variables, independent variables, predictors
- Data generating process

$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

$$\mathcal{X} \in \mathbb{R}^{n \times p}, \mathcal{Y} \in \mathbb{R}^p$$

## Statistical model to approach the DGP

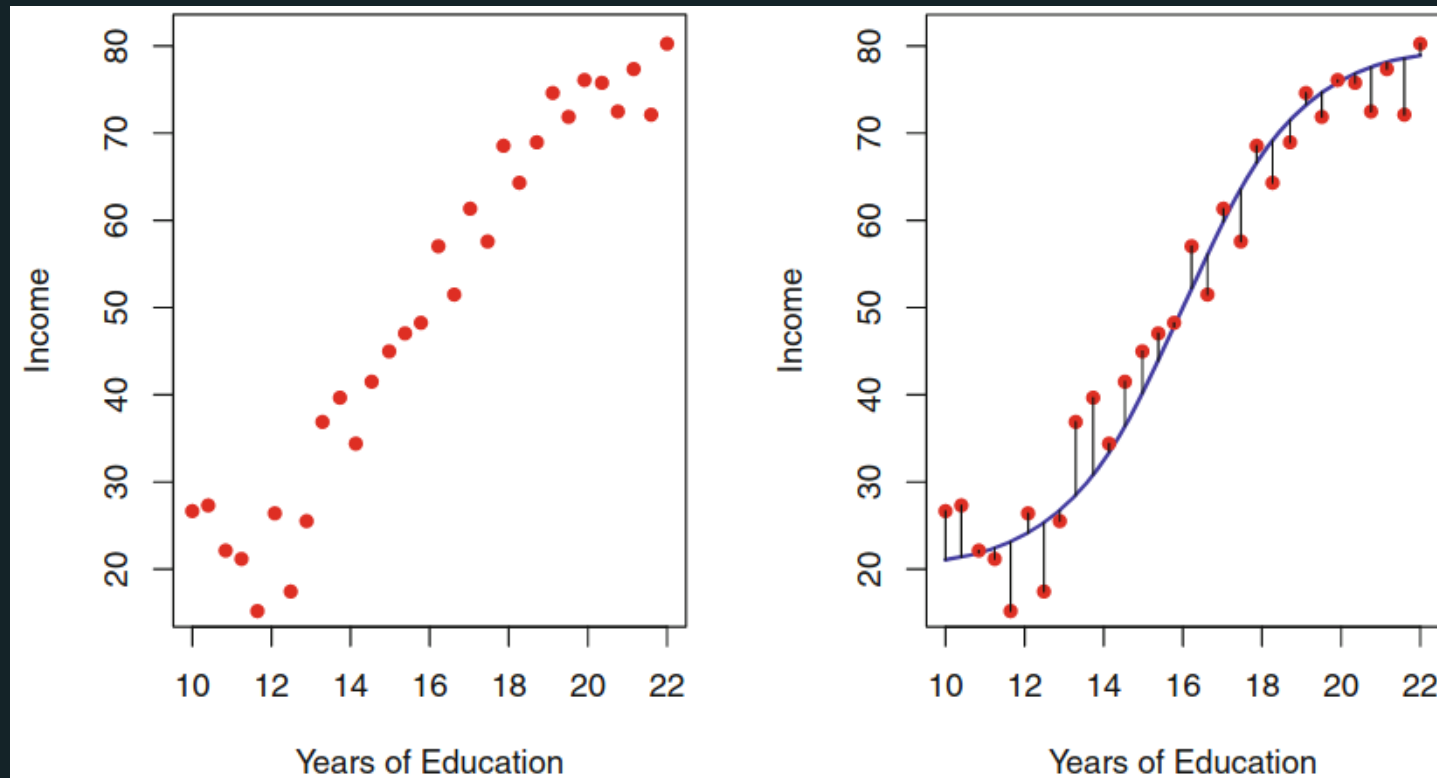
Concretely, finding  $f(\cdot)$  s.t.

$$Y = f(X) + \epsilon$$

- $f(X)$  is an unknown function of a matrix of predictors  
 $X = (x_1, \dots, x_p)$ ,
- $y$ : a scalar outcome variable
- an error term  $\epsilon$  with mean zero.
- While  $X$  and  $y$  are known,  $f(\cdot)$  is unknown.

**Goal of statistical learning:** to utilize a set of approaches to estimate the “best”  $f(\cdot)$  for the problem at hand.

## Example: income as a function of education



## [Motivation] Goals of data analysis

- **Description**: Document observed relationships between  $X$  and  $y$
- **Causation**: Learn about causal relationships: If we change  $X$ , how will  $y$  change?
- **Prediction**: Be able to guess the value of  $y$  from the  $X$  variables.

⇒ Uncovering the data generating process



## [Description] Unsupervised learning

Estimating functions without the aid of outcome variable  $y$ .

- We only observe  $X$  and want to learn something about its structure
- **Clustering**: Partition data into homogeneous groups based on  $X$
- **Dimensionality reduction** (e.g. PCA)

## [Description] Modelling the mean-dependence

- Going further than the correlation analysis
  - Cf. example on management quality and firm size
- Regression analysis uncovers mean-dependence between two variables, controlling for other variables:
  - comparing average values of a **dependent variable** ( $y$ )
  - for observations that are different in the set of **explanatory variables** ( $X$ ).
  - Doing so uncovers the pattern of association between  $y$  and  $X$ .

⇒ Econometrics


[Prediction] Guess the values of  $y$  from the  $X$  variables

- Estimating functions with **known observations** and **outcome** data.
  - We observe data on  $y$  and  $X$ 
    - $y$  = labels
    - $X$  = predictors
  - We want to learn the mapping  $\hat{y} = \hat{f}(X)$
- We predict  $y$  by  $\hat{y} : \hat{f}(X)$ 
  - Useful for observations for which  $y$  is unknown and  $X$  is known

⇒ Supervised machine learning

# Supervised Machine learning and/or Econometrics

- **Common objective:** to build a predictive model, for a variable of interest, using explanatory variables (or features)
- **Different cultures:**
  - *E*: probabilistic models designed to describe economic phenomena
  - *ML*: algorithms search through a set of possible prediction models that capture the best relationship

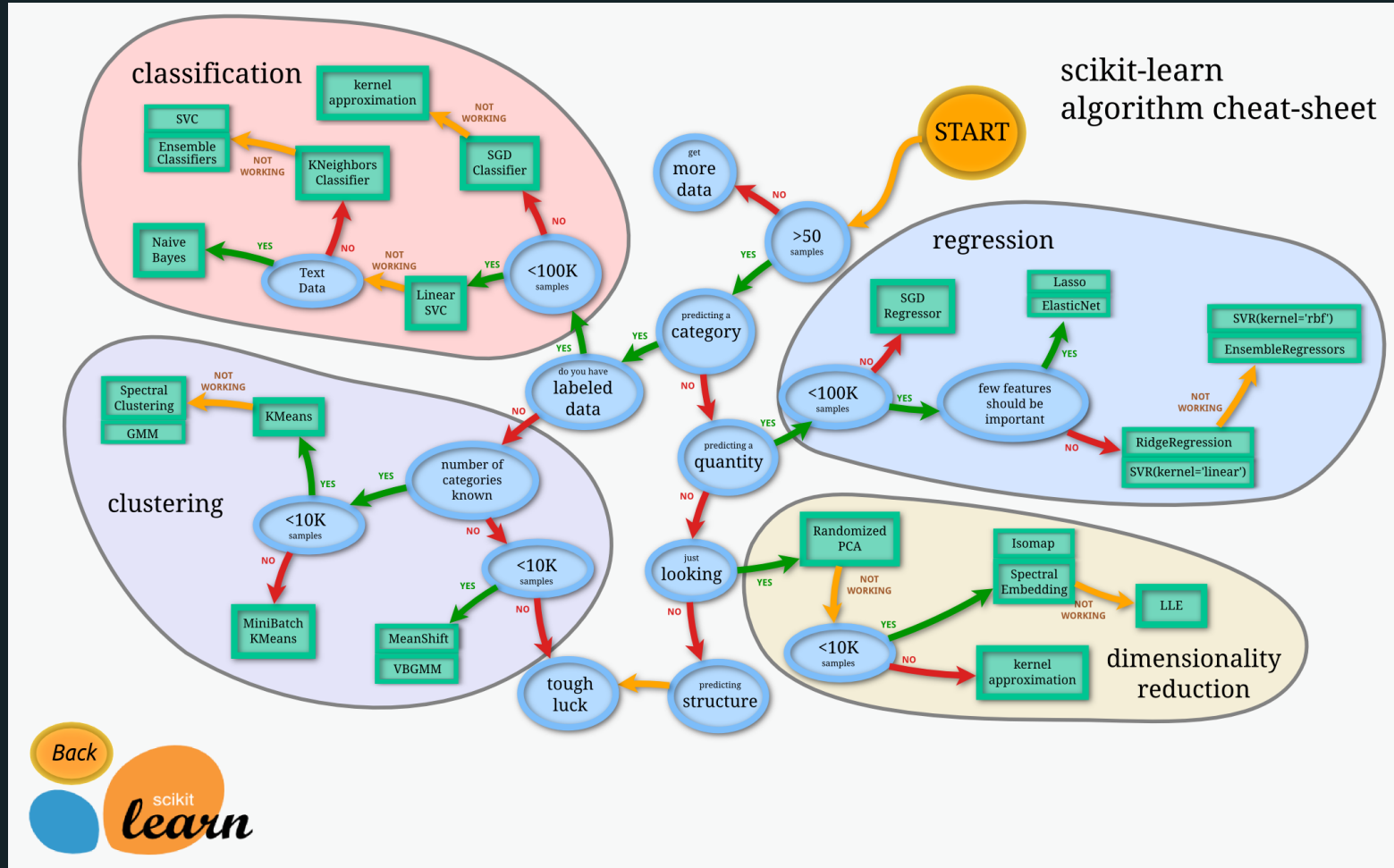
 Charpentier A., Flachaire, E. & Ly, A. (2018). *Econometrics and Machine Learning*. *Economics and Statistics*, 505-506, 147-169.

# Prediction setup

- $y$  continuous/quantitative (eg. price)
  - **Regression** problem
- $y$  discrete/binary (eg. price)
  - Probability prediction
  - **Classification** problem
- Time series prediction

Key idea: systematically combine estimation and model selection

# Panorama of the models



Source: **scikit-learn**

## Panorama of the model eco-system

- Econometrics: **statsmodels**
- Machine Learning: **scikit-learn**
- Deep Learning: **keras**, pytorch, TensorFlow...

# Overview of ML process

---



# How does ML work?

- Econometrics:
  - estimation procedure gives a single set of estimated coefficients
- Machine learning is different
  - Search through a set of possible prediction models
    - Which one does capture best the relationship?
- Estimation of the model on a **Training set**
- Test for the goodness of fit on the **test set** (and avoid overfitting)

# Typical supervised ML procedure

## 1. Labeled data

- Decompose the observations in test/training set
- Pre-processing of the input variables

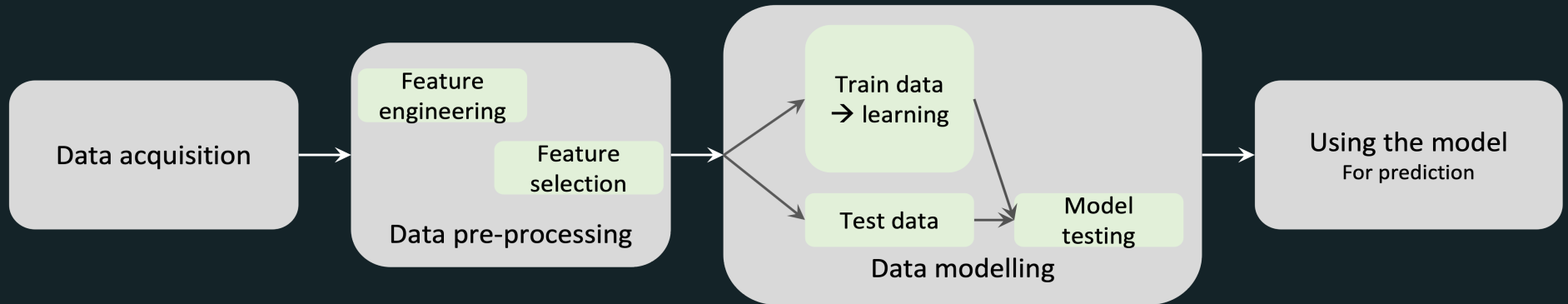
## 2. Learning:

- On a restricted sample: the **Training set**
- Pre-processing of the input
- An algorithm is used to "learn" the association pattern between the input variables and the label

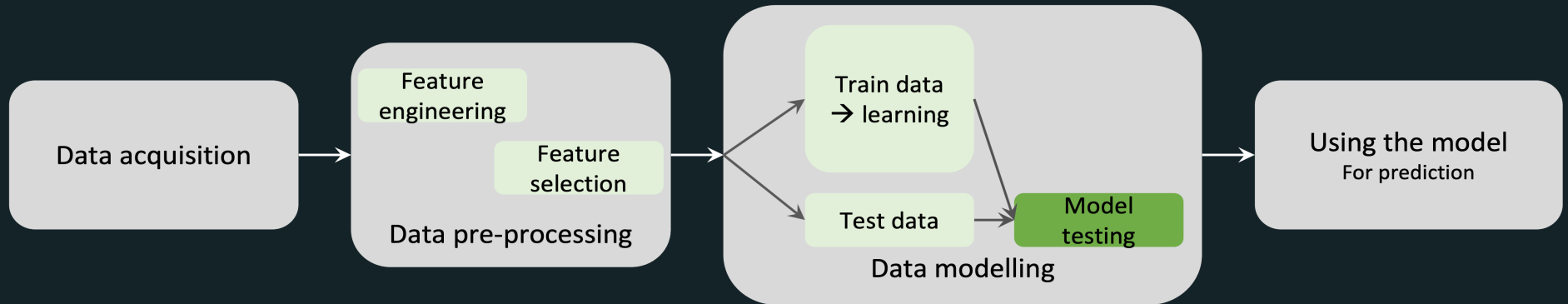
## 3. Assessing how the model fits the data

- We test the validity of the model on a **test set**
  - = A sample of the labeled data that had been held out
- Fit measured using the "mean squared error"

# ML procedure



# How to choose between models?



## Reducible and irreducible error

$\hat{f}(X) = \hat{Y}$  estimated function

$f(X) + \epsilon = \hat{Y}$  true function

- **Reducible error:**  $\hat{f}$  is used to estimate  $f$ , but not perfect
  - Accuracy can be improved by adding more features
- **Irreducible error:**  $\epsilon$  = all other features that can be used to predict  $f$ 
  - Unobserved  $\rightarrow$  irreducible

## Reducible and irreducible error

$$\begin{aligned} E(y - \hat{y})^2 &= E[f(X) + \epsilon - \hat{f}(X)]^2 \\ &= \underbrace{[f(X) - \hat{f}(X)]^2}_{\text{Reducible}} + \overbrace{\text{Var}(\epsilon)}^{\text{Irreducible}} \end{aligned}$$

⇒ **Objective:** estimating  $f$  with the aim of minimizing the reducible error

## The Prediction error

- Predicted value  $\hat{y}_j$  for target observation  $j$
- Actual value  $y_j$  for target observation  $j$
- Prediction error

$$e_j = \hat{y}_j - y_j$$

- Error = predicted alue - actual value

## Mean Squared Error (MSE)

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

- **Regression setting:** the **mean squared error** is a metric of how well a model fits the data.
- In sample: computed on the training set

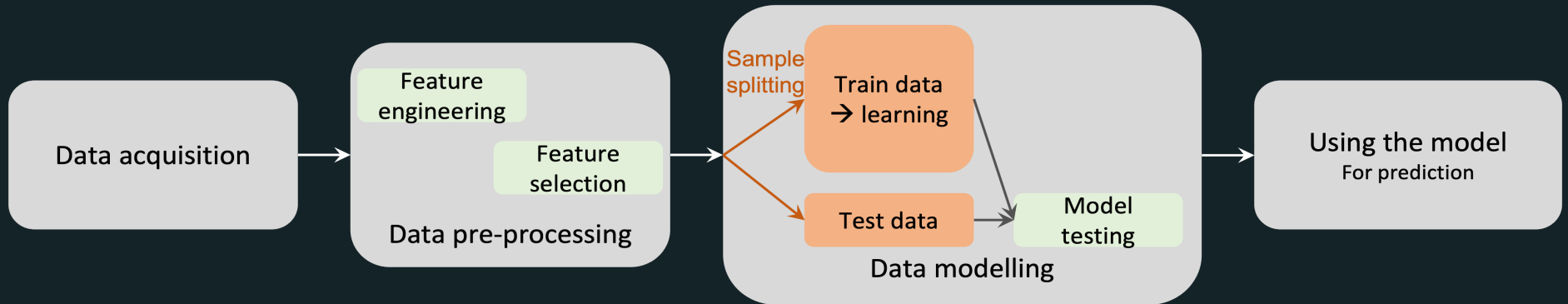
⇒ **Training MSE**



# Sample splitting approach to measuring model fit

---

# Sample splitting



## Resampling methods

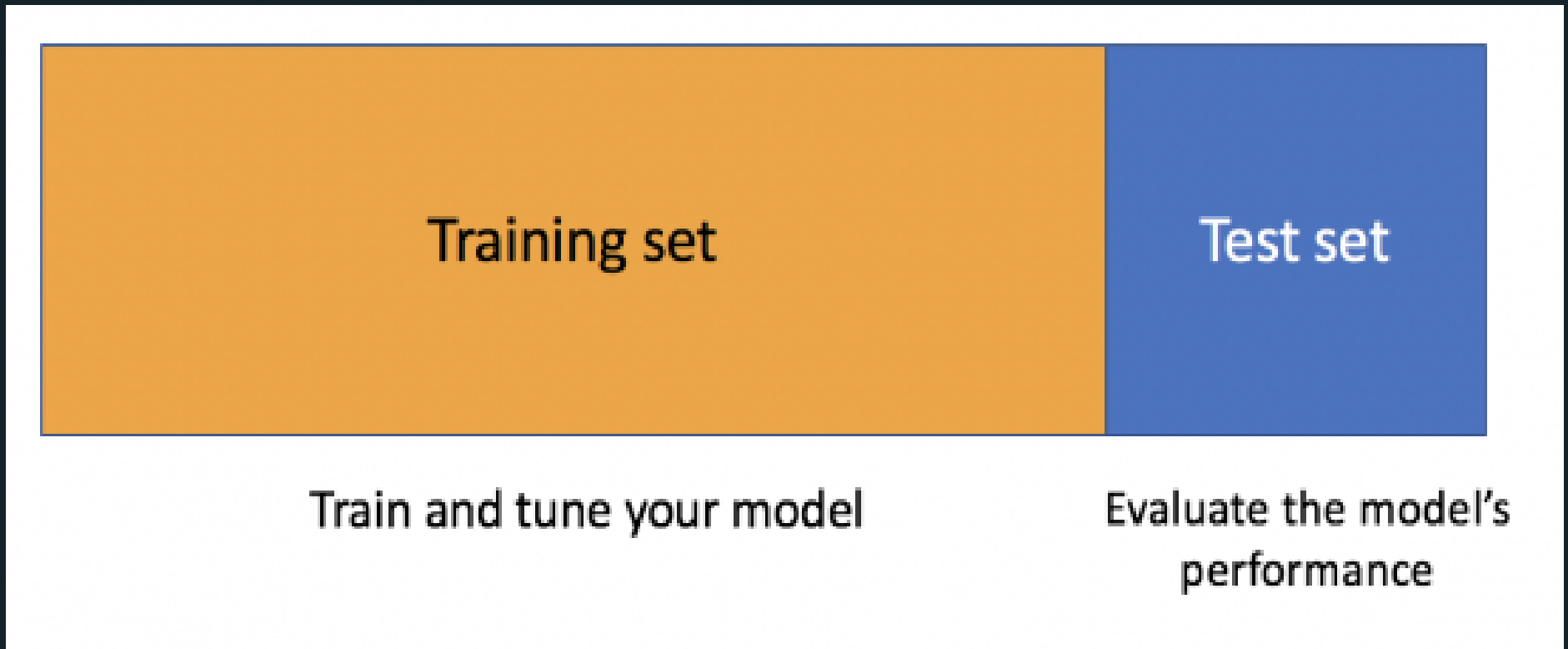
Estimate the test error rate by

**holding out** a subset of the training observations from the fitting process,

+ then **applying** the statistical learning method to those held out observations

## Validation set approach

- Labeled data **randomly** into two parts: training and test (validation) sets.



## Two concerns

- Arbitrariness of split
- Only use parts of the data for estimation
  - we tend to overestimate test MSE because our estimate of  $f(x)$  is less precise

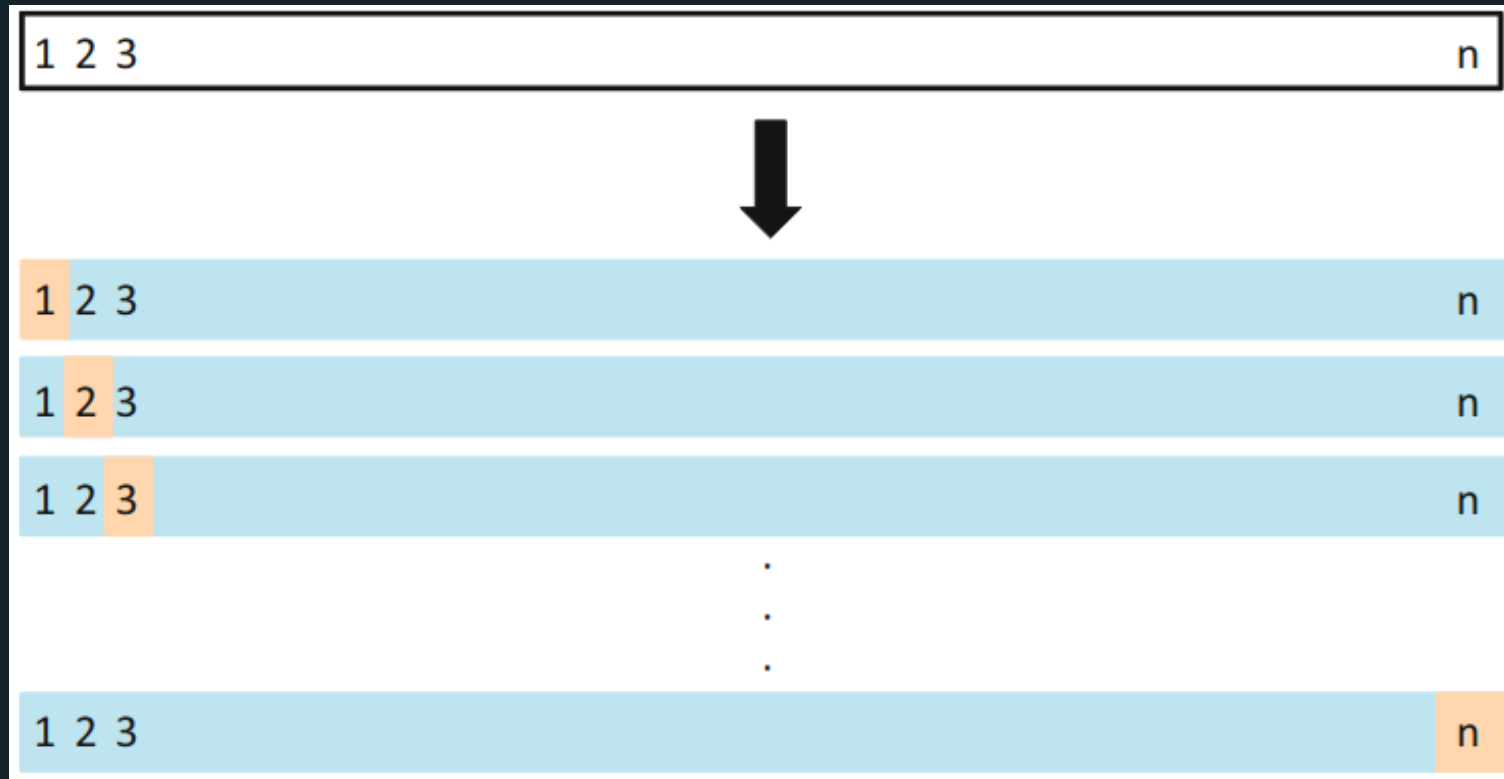
## Leave-One-Out Cross-Validation (LOOC)

- Fit on  $n - 1$  training observations, and a prediction the Last
- Iterate  $n$  times
- Assess the average model fit across each test set.

Estimate for the test MSE:

$$CV_n = \sum_{i=1}^n MSE_i$$

# Leave-One-Out Cross-Validation (LOOC)



- less bias than the validation set approach
- always yield the same results
- potentially too expensive to implement

## $k$ -fold Cross-validation

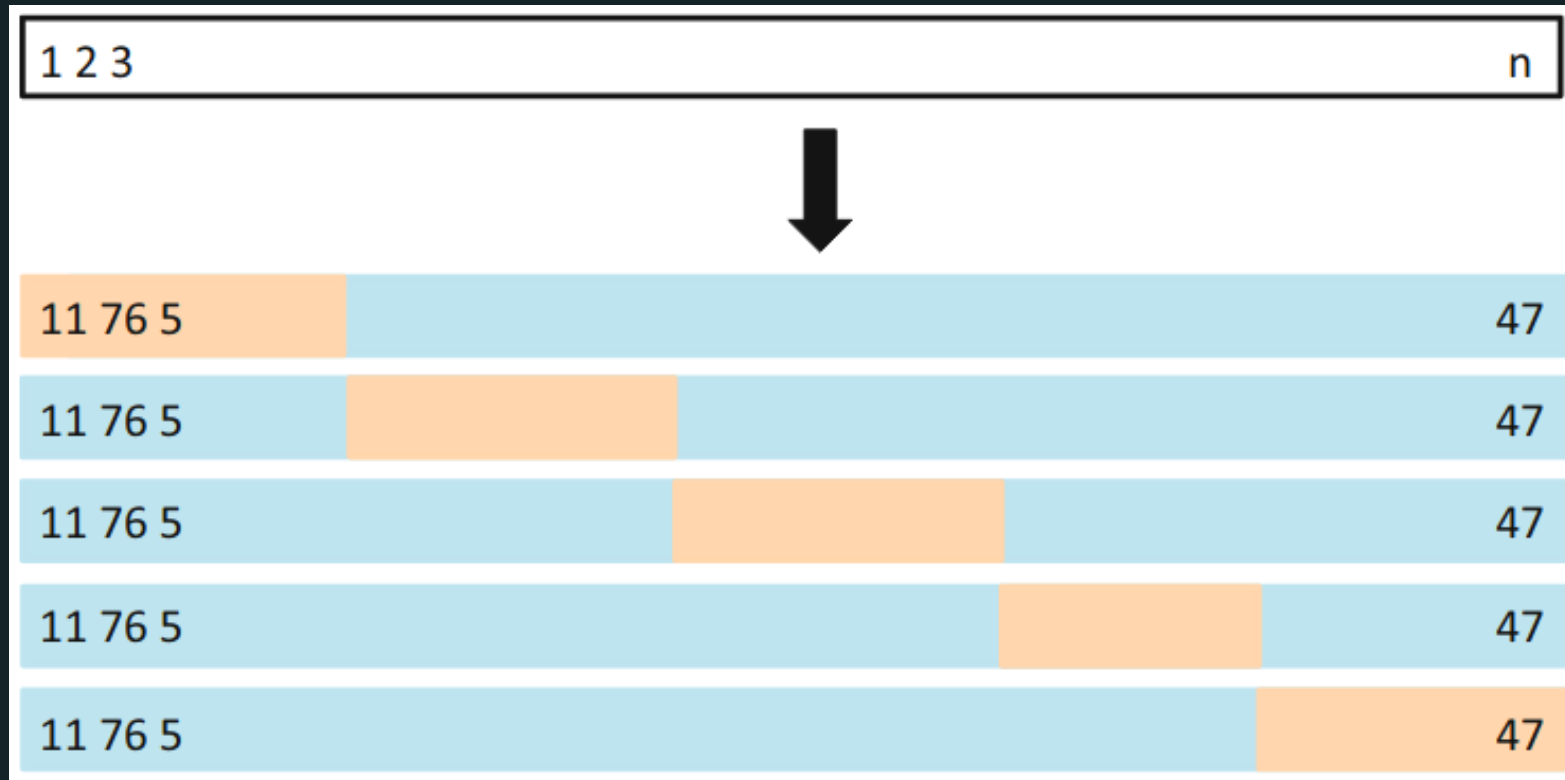
- Leave-One-Out Cross-Validation with  $k = 1$
- Randomly dividing the data into the set of observations into  $k$  groups
- 1st fold is treated as a validation set, and the method is fit on the remaining  $k - 1$  folds
- Iterate  $k$  times

Estimate for the test MSE:

$$CV_k = \sum_{i=1}^k MSE_i$$



# $k$ -fold Cross-validation



⇒ Arguably the contribution to econometrics: Cross-validation (to estimate test MSE)!

## Measuring the fit

To estimate model fit we need to partition the data:

1. **Training set**: data used to fit the model

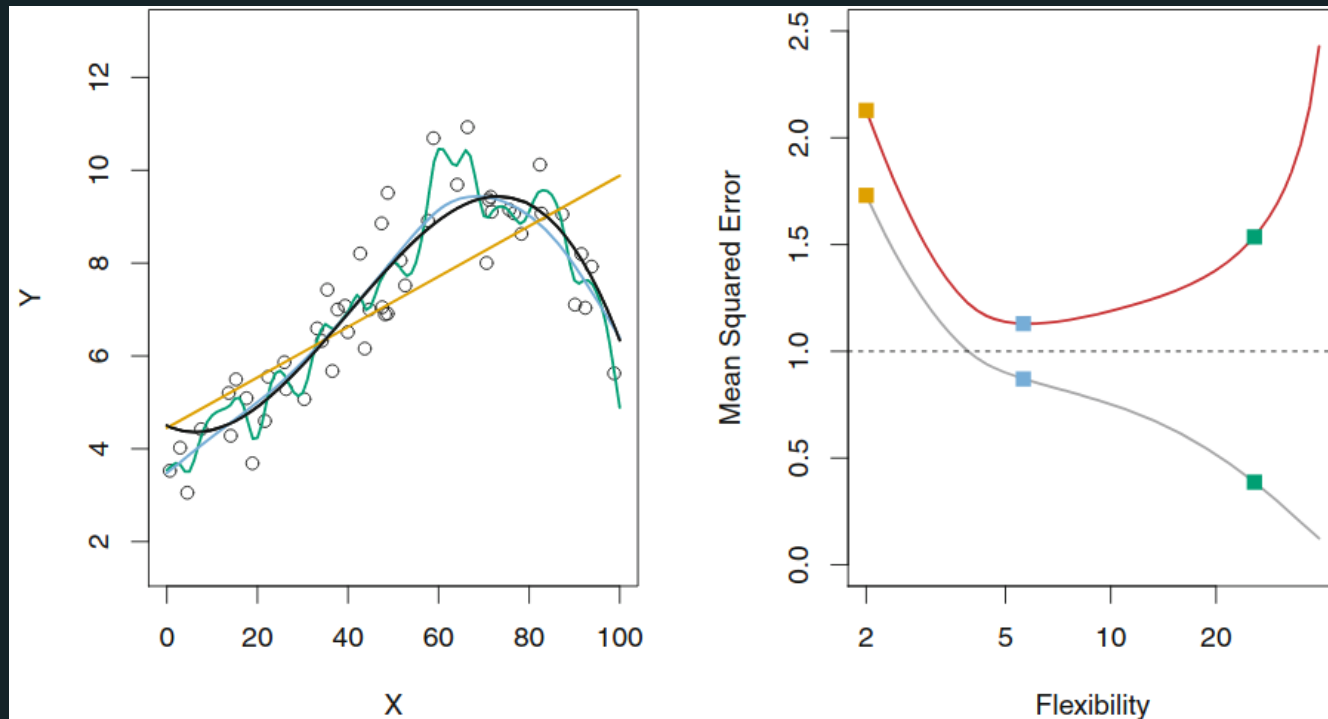
- **Training MSE**: how well our model fits the training data.

1. **Test set**: data used to test the fit

- **Test MSE**: how well our model fits new data

We are most concerned in **minimizing test MSE**

# Training MSE, test MSE and model flexibility



Red (grey) curve is test (train) MSE

Increasing model flexibility tends to **decrease** training MSE but will eventually **increase** test MSE

# Overfitting

- As model flexibility increases, training MSE will decrease, but the test MSE may not.
- When a given method yields a small training MSE but a large test MSE, we are said to be **overfitting** the data.
- (We almost always expect the training MSE to be smaller than the test MSE)
- Estimating test MSE is important, but requires training data...

## Conclusion on model selection

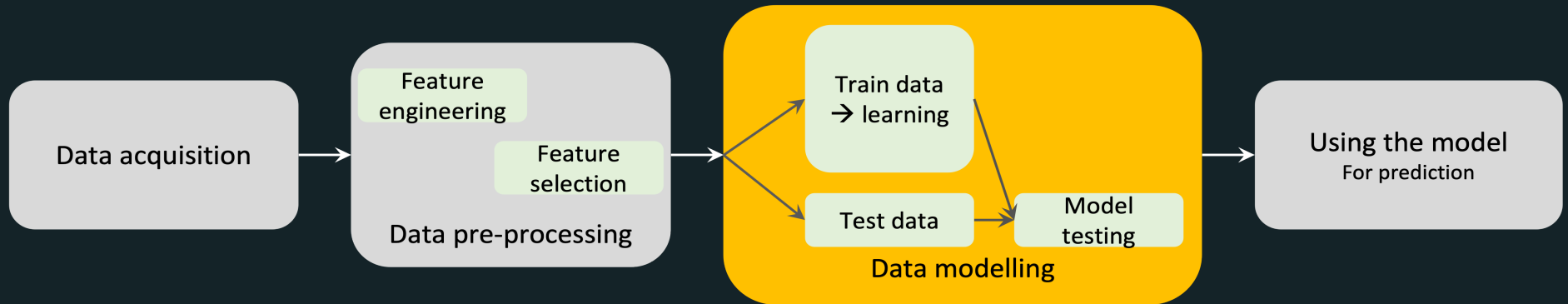
= finding the best fit while avoiding overfitting and aiming for high external validity

1. Train the model on the training set
2. Evaluate predictive performances on using the test (validation) set
  - Evaluation criterion (loss function) = RMSE (root mean squared error)

# Supervised ML: Regressions

---

# What are the main categories of models?



# Model building

- Deciding about the predictors to include in the model and their functional form.
- We have strong computers, cloud, etc - why could not we try out all possible models and pick the best one?
  - Too many models possible!
- 2 methods to build models:
  1. By hand -- mix domain knowledge and statistics
  2. By Smart algorithms = machine learning



# 1. Linear Regression as a predictive model

---

## Linear Regression (Theory)

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon$$

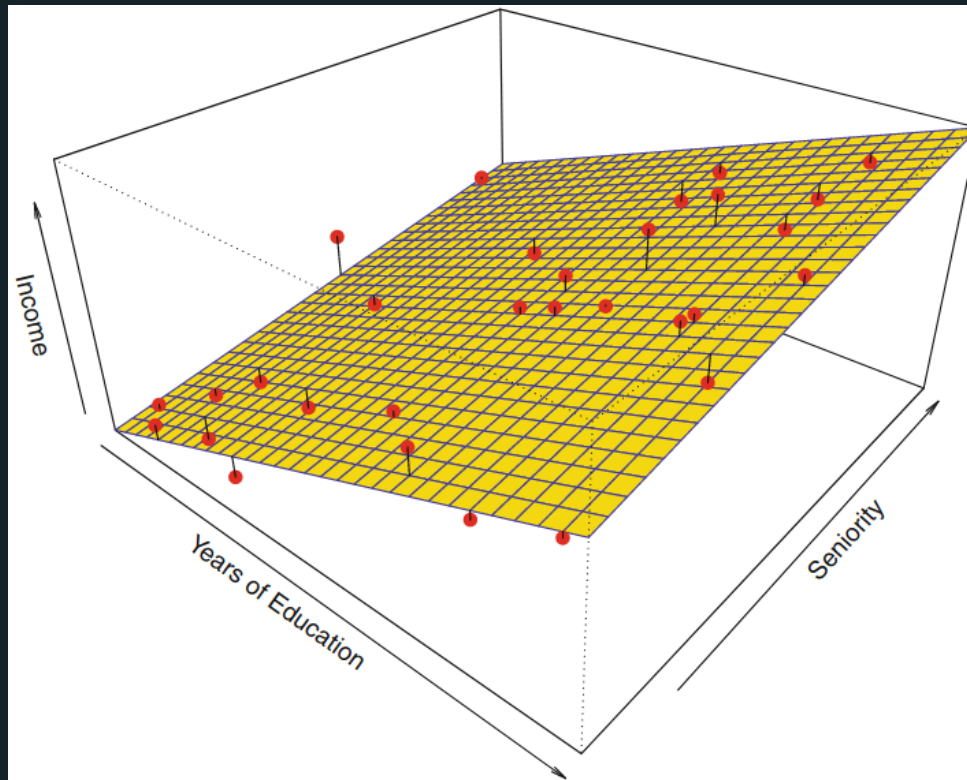
= one of the simplest algorithms for doing supervised learning

A good starting point before studying more complex learning methods

*Notes:*

Interpretation of  $\beta_j$  = the average effect on  $Y$  of a unit increase in  $X_j$  holding all other predictors fixed

# Linear estimate



# Implementation in python

Using statsmodel:

```
import statsmodels.formula.api as smf  
  
reg1 = smf.ols(formula="lnw~female", data=df).fit(cov_type="HC1")  
reg1.summary()
```

## Extensions of the Linear Model

Going further model's assumptions:

- the **additive**: the effect of changes in a predictor  $x_j$  on the response  $Y$  is independent of the values of the other predictors
- **linearity**: the change in the response  $y$  due to a one-unit change in  $x_j$  is constant

## Interactions

- Adding interacted variable can help
- Should respect the **hierarchy principle**:
  - if an interaction is included, the model should always include the main effects as well

## Non Linearity

- Include transformed versions of the predictors in the model
- ⇒ Including polynomials in  $X$  may provide a better fit

# Linear Models: pros and cons

- **Pros:**
  - Interpretability
  - Good predictive performance
  - Accuracy measures for
    - coefficient estimates (standard errors and confidence intervals)
    - the model
- **Cons:**
  - When  $p > n$
  - Tend to over-fit training data.
  - Cannot handle multicollinearity.



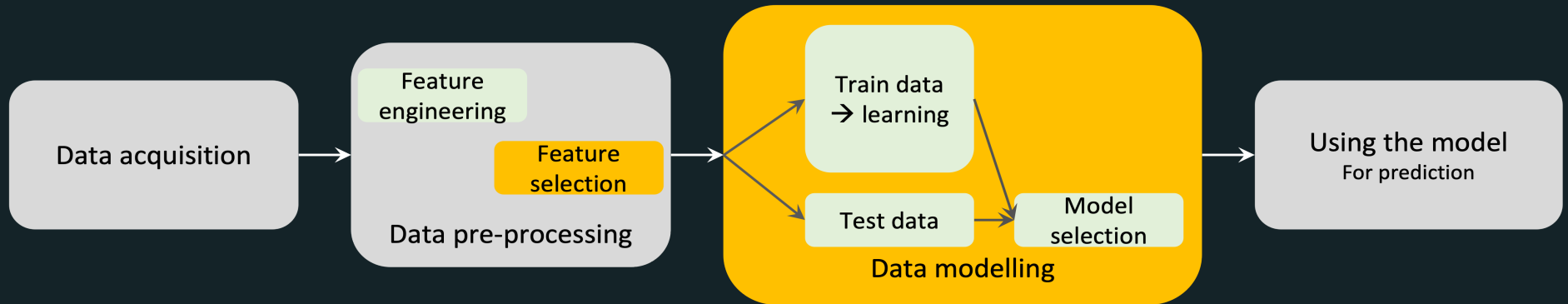
## Generalization of the Linear Models

- **Regularized fitting:** ridge regression and lasso
- **Non-linearity:** nearest neighbor methods
- **Interactions:** Tree-based methods, random forests and boosting

## 2. Regularized linear regressions

---

# Feature selection with regularization



## Why Regularization?

- Key question: which features to enter into model, how to select?
  - There is room for an automatic selection process.
- Solution against **overfitting**
- Allow High-Dimensional Predictors
  - $p \gg n$ : OLS no longer has a unique solution
  - $x_i$  "high-dimensional" i.e. very many regressors
    - pixels on a picture

# LASSO

Least Absolute Shrinkage and Selection Operator

LASSO proposes a method to build a model which just **includes the most important predictors**.

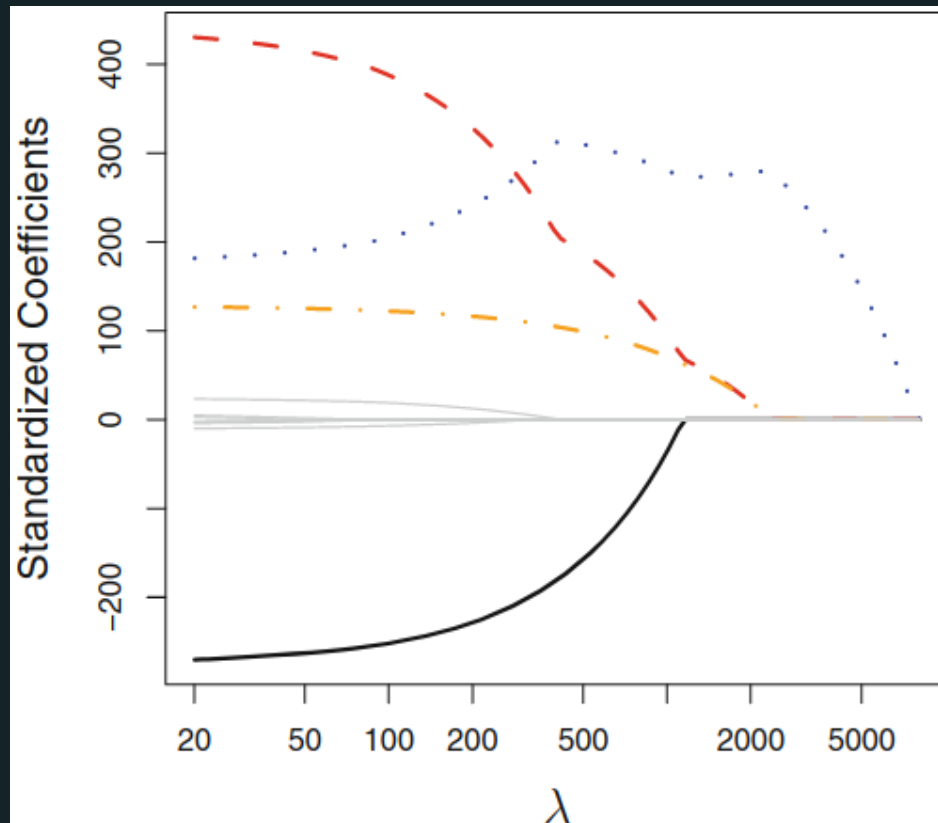
Modifies the way regression coefficients are estimated by adding a penalty term for too many coefficients

## Lasso

$$\hat{\beta} = \operatorname{argmin}_{\beta} \frac{1}{n} \sum_{i=1}^n \left( y_i - (\beta_0 + \sum_{j=1}^k \beta_j x_{ij}) \right)^2 + \lambda \sum_{j=1}^k |\beta_j|$$

- $R(\beta) \sum_{j=1}^k |\beta_j|$  = **regularization function** with a L1 norm penalty function
- $\lambda$  is a **hyperparameter** (or tuning) where higher values increase regularization

# Lasso Coefficients

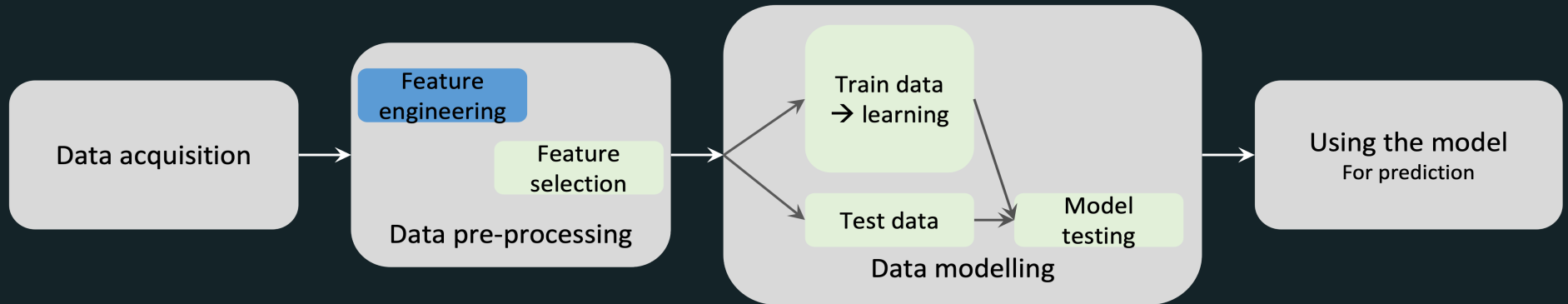


# Working on the features: pre-processing

---



# Pre-processing



# Feature engineering: what $x$ variables to have and in what functional form

Feature engineering typically includes

- deciding on what predictor variables to include
- exploring and addressing potential missing values and extreme values
- deciding about function forms of the predictor variables
- designing possible interactions of the predictor variables

# What to do with different type of variables

- If binary (e.g, yes/no; male/female; 1/2)
  - create a 0/1 binary variable
- If string / factor
  - check values, and create a set of binaries.
- Continuous
  - Make sure it is stored as number
  - standardization/ normalization
- Text – Natural Language Processing.
  - Mining the text to get useful info.
  - May simply go and find some words, and create binaries

## What's next?

- Classification
  - Logistic regression
  - Different measure of the performance
- Unsupervised learning
  - Dimensionality reduction
  - Clustering methods