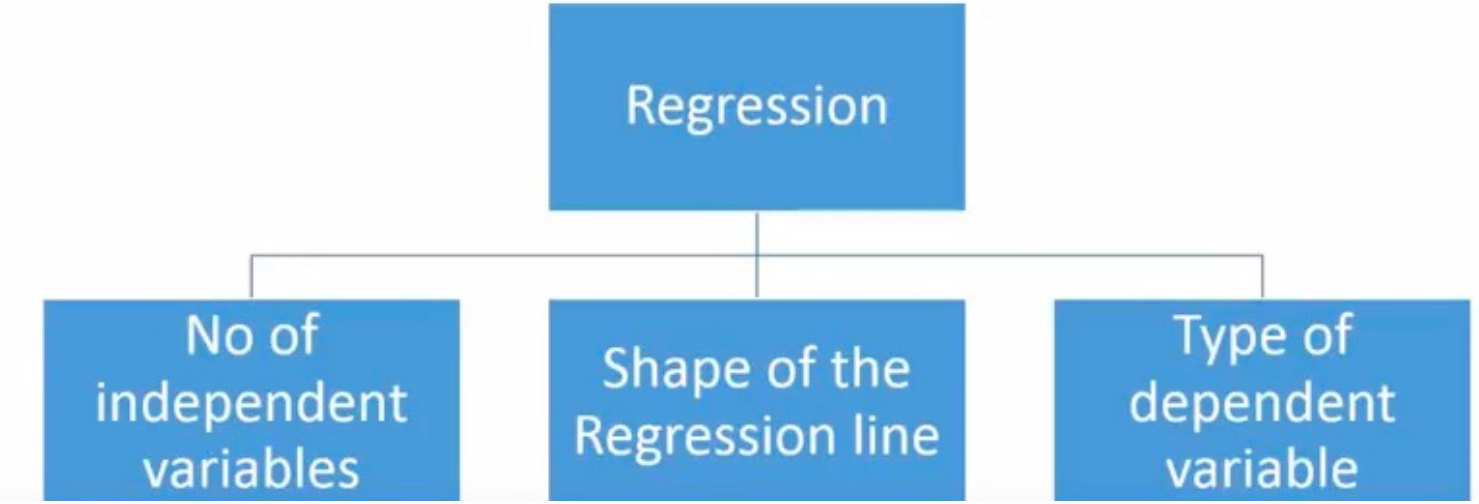
Tufts

CS135
Introduction to Machine Learning

Lecture 6: More on Regression and Feature Selection

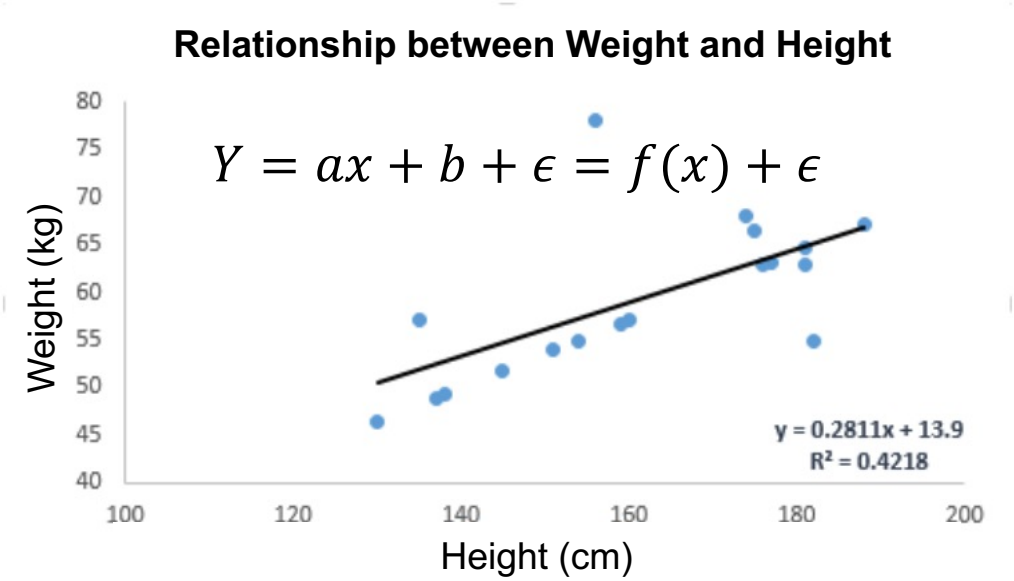
Types of Regression

- Various regression techniques available to make predictions
- Variants in regression type are due to 3 factors:



Linear Regression (Recap: Overview)

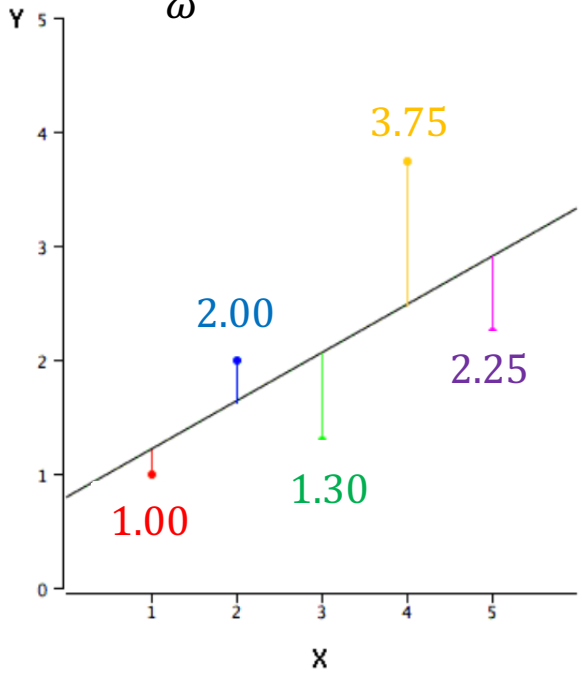
Relationship between **dependent variable (Y)** and one or more **independent variables (X)** using a **best fit straight line** (also known as regression line).



Linear Regression (Recap: Find best fit, i.e., *a* and *b*)

- Least Square Method (LSM)
  - Finds best-fit line by minimizing the sum of the squares of the vertical deviations from each data point to the line.

$$\min_{\omega} \|X\omega - y\|_2^2$$

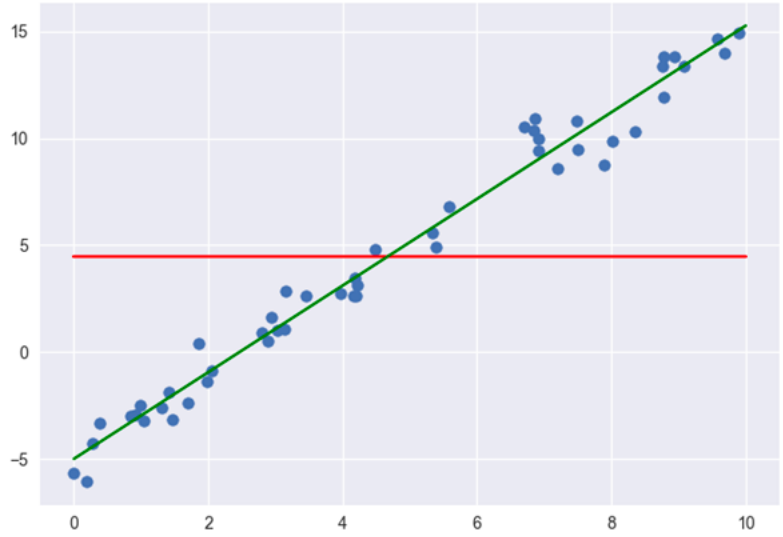


- We can evaluate the model performance using the metric **R-square**.

Linear Regression (Recap: Fitness of line)

- R-Squared
  - Simply explains how good is model when compared to the baseline model

baseline



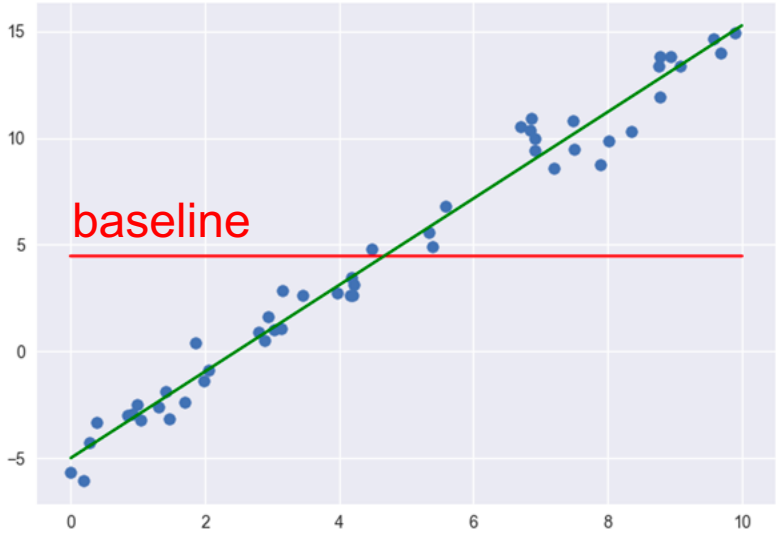
- Baseline model is mean of the observed response (i.e., Y)
- Relative to this we measure fitness via **Coefficient of Determination** (aka r^2)

Linear Regression (Recap: Measurement of Fitness)

- Simply explains how good is model when compared to the baseline model

$$R^2 = 1 - \frac{SSE}{SST}$$

where $SSE = \sum_{i=1}^{n}(y - \hat{y})^2$ & $SST = \sum_{i=1}^{n}(y - \bar{y})^2$



- Baseline model is mean of the observed response (i.e., Y)
- Relative to this we measure fitness via **Coefficient of Determination** (aka r^2)
- Determine how much variation is explained by the regression line

- What is R^2 of baseline?

SSR = SST, thus, R^2=0

Linear Regression (Recap: Important Points)

- Must be a **linear relationship** between *independent* and *dependent* variables
- Multiple regression suffers from **multicollinearity, autocorrelation, and heteroscedasticity**.
- Linear Regression is sensitive to **Outliers.**
  - It affects the regression line and, eventually, the forecasted values.
- **Multicollinearity** can increase the variance of coefficient estimates, making estimates sensitive to minor changes in the model.
  - Yields coefficients (i.e., weights that are unstable)
- The error terms must be **normally distributed**

COMING SOON

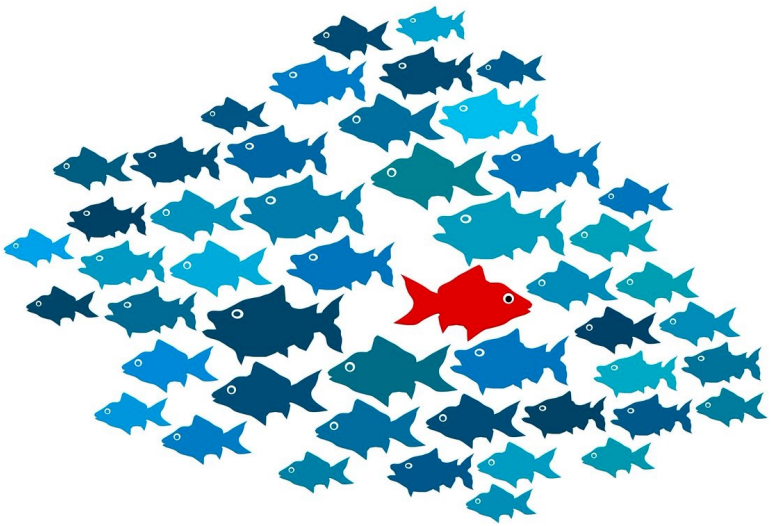
- Given multiple independent variables, we can go with **forward selection**, **backward elimination** & **stepwise approach for selecting** the most significant independent variables.

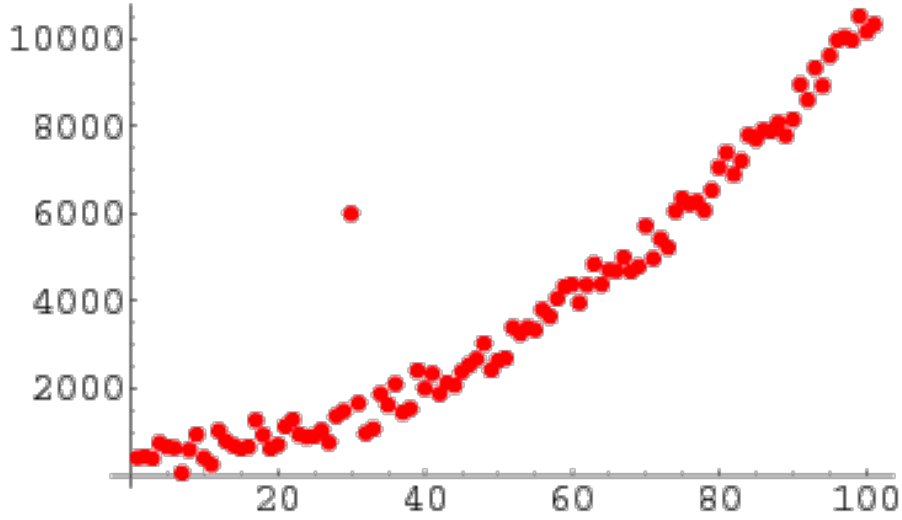
Linear Regression: Assumptions

- Unusual and Influential Data
  - Outliers
  - Leverage
  - Influence
- Heterosckedasticity
  - Non-constant variance
- Multicollinearity
  - Non-independence of x variables

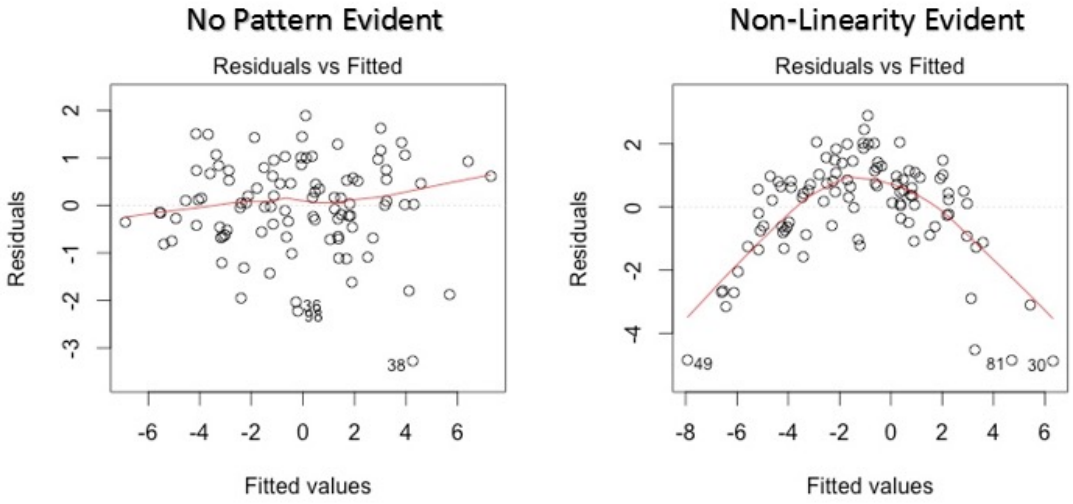
Linear Regression: Unusual and Influential Data

- Outliers
  - An observation with large residual.
    - An observation whose dependent-variable value is unusual given its values on the predictor variables.
    - An outlier may indicate a sample peculiarity or may indicate a data entry error or other problem.





Linear Regression: Residual vs Fitted Values

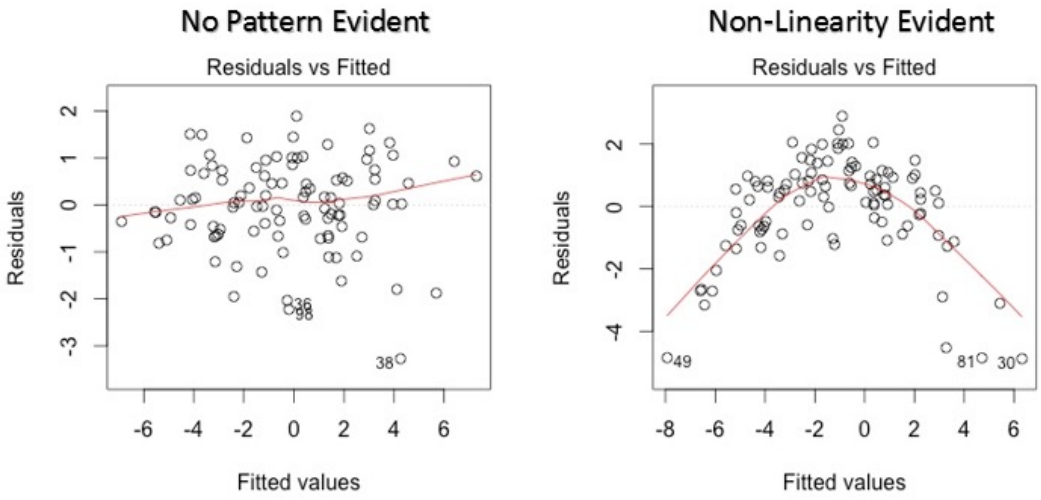


2 Issues to overcome

1. If any pattern (perhaps parabolic) exists in the plot, it suggests non-linearity in the data, and the model doesn't capture non-linear effects.
2. The funnel shape is evidence of non-constant variance i.e., heteroskedasticity.

Linear Regression: Residual vs Fitted Values

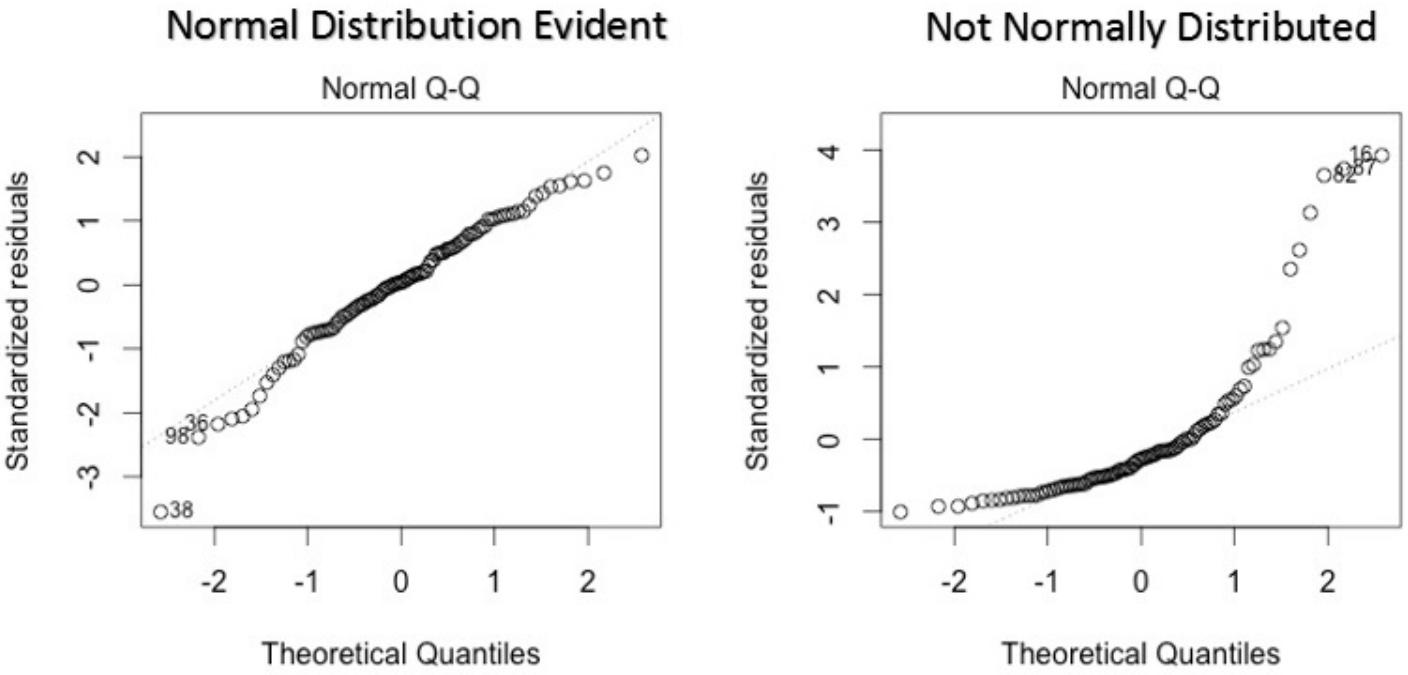
```
import statsmodels.formula.api as sm
import pandas as pd
import matplotlib.pyplot as plt
results = sm.ols(formula='Y ~ X1', data=df).fit()
Y_pred = results.predict(df[["X1"]])
residual = df["Y"].values - Y_pred
plt.scatter(df[["X1"]],residual)
plt.xlabel("X1 - a predictor")
plt.ylabel("residual")
plt.show()
```



Solution:

1. The issue of non-linearity can be overcome with a non-linear transformation of predictors, e.g., $\log X$, $\sqrt{X}$, X^2 transform the dependent variable.
2. Issues of heteroscedasticity can be overcome by transforming the response variable such as $\log Y$ or $\sqrt{Y}$; Also, a weighted least square method might tackle heteroskedasticity.

Linear Regression: Normal Q-Q Plot



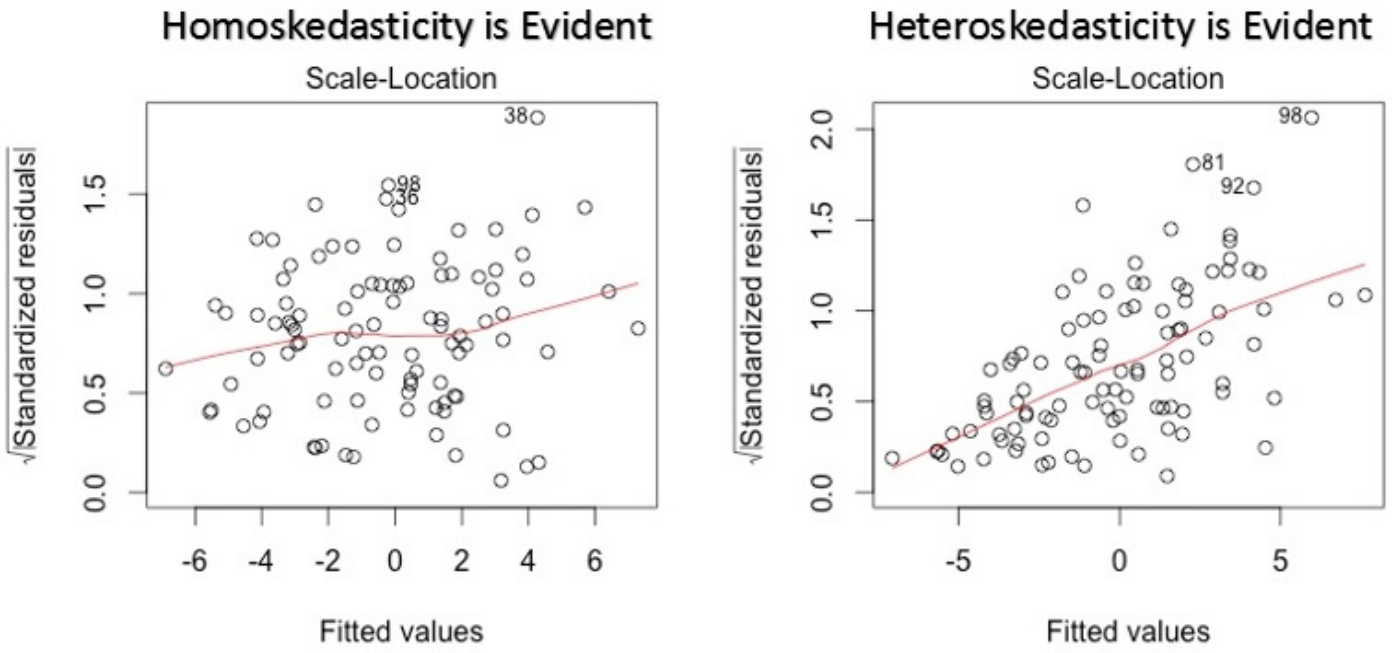
Solution:

1. If errors are not normally distributed, non–linear transformation of the variables (response or predictors) can improve model

```
import numpy as np
import pylab
import scipy.stats as stats

measurements = np.random.normal(loc = 20, scale = 5, size=100)
stats.probplot(measurements, dist="norm", plot=pylab)
pylab.show()
```

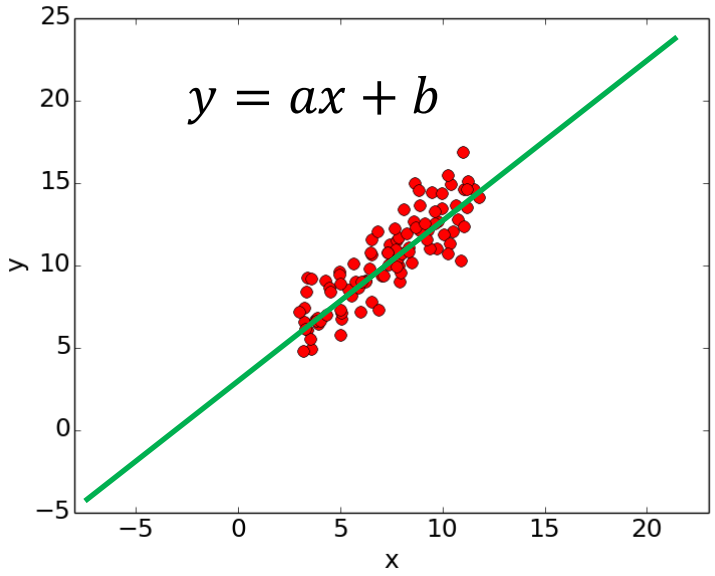
Linear Regression: Scale Location Plot



Solution:

1. Follow the solution for heteroskedasticity given in plot 1.

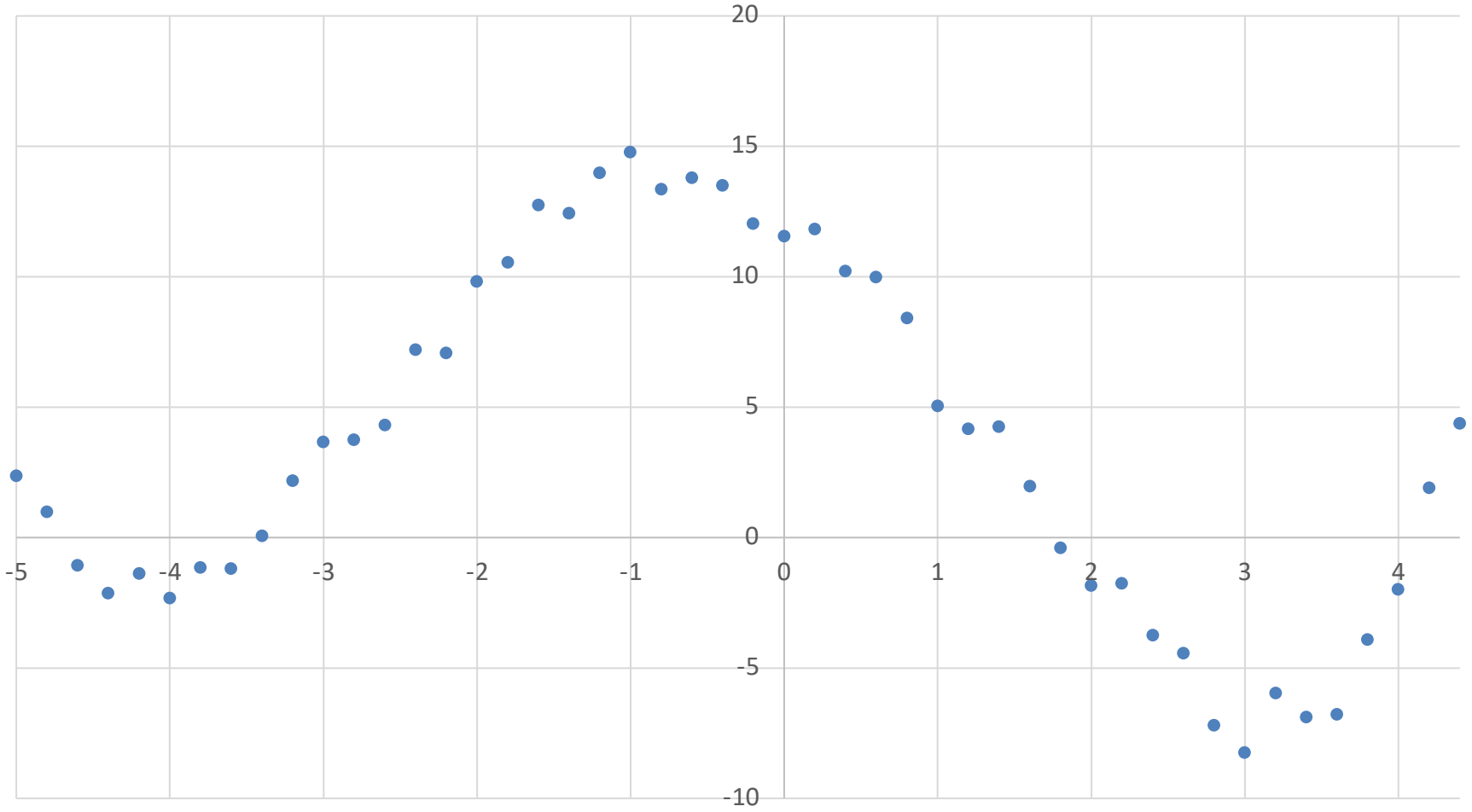
Minimizing Objective (i.e., minimizing MSE)



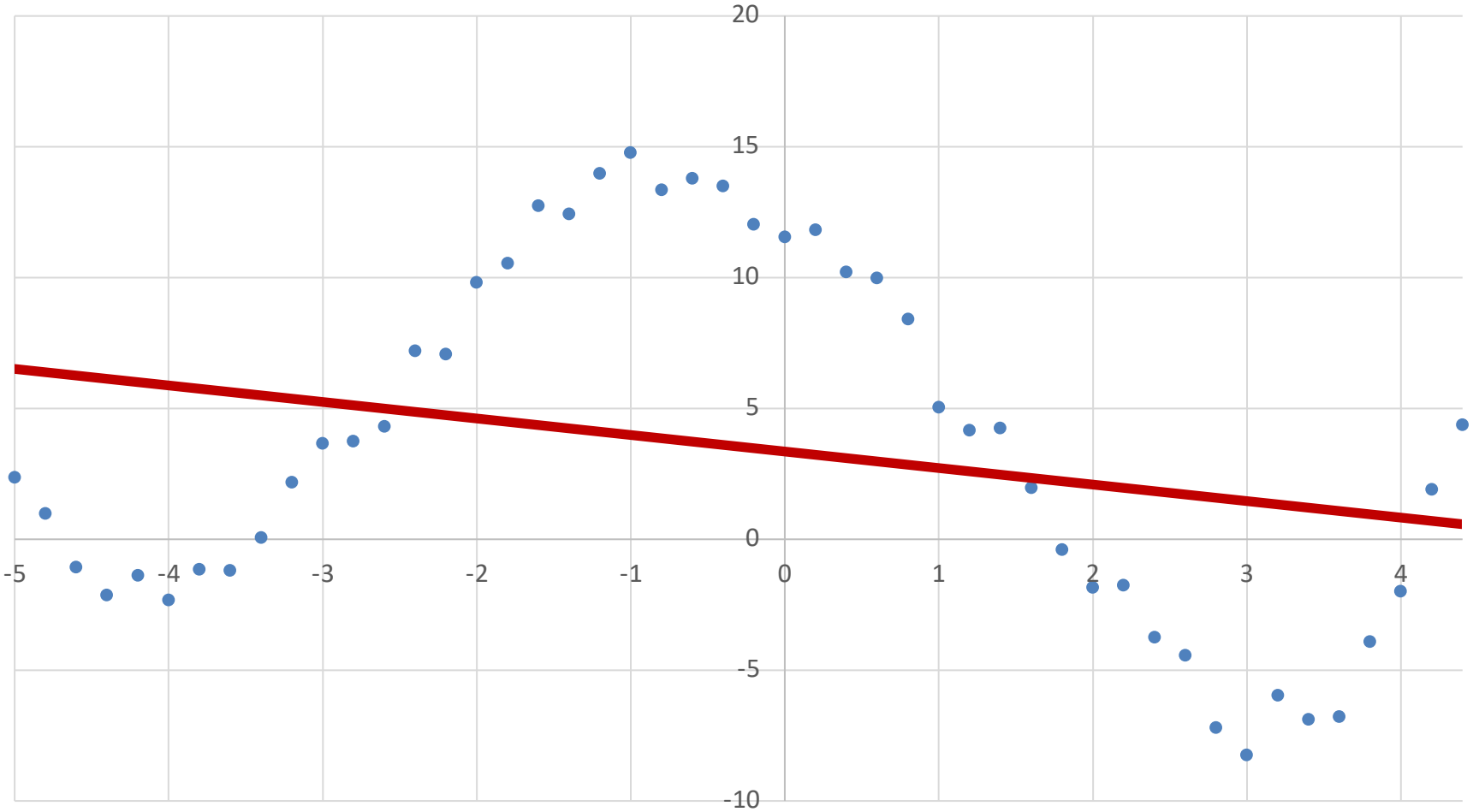
$$Error_{(a,b)} = \frac{1}{n}\sum_{i=1}^{n}(y_i - (ax_i + b))^2$$

Two parameters a and b

What if Data is Not Linear?



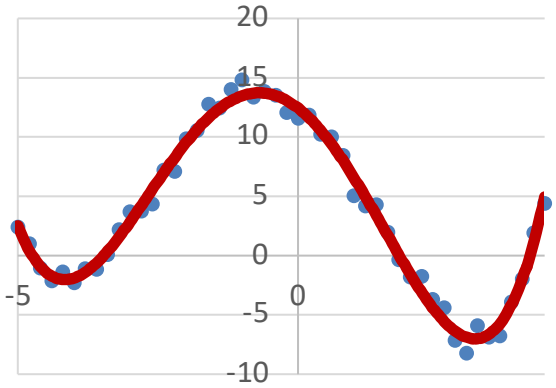
What if Data is Not Linear?



Fitting Polynomials

To learn a polynomial f of degree

$$k = 2, 3, \ldots :$$



- Produce new features containing all monomials:

$$\prod_{i=1}^{d} x_i^{k_i} = x_1^{k_1} x_2^{k_2} \ldots x_d^{k_d}$$

Where. $k_1 + k_2 + \ldots + k_d \leq k$

- Perform linear regression on the resulting new set of features.

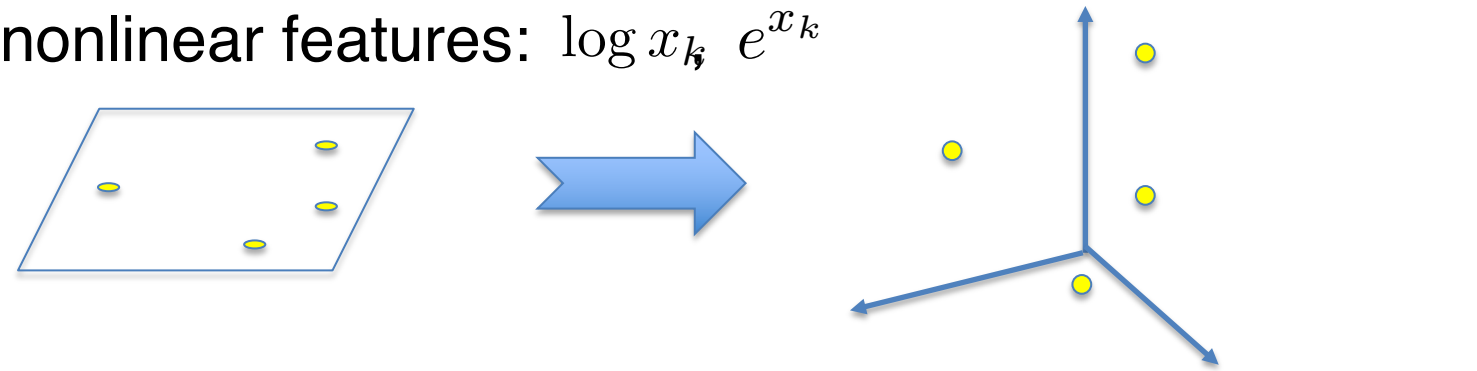
Different Basis Functions

Coefficients to be learned

Known **basis** functions

$$f(x) = \sum_{\ell=1}^{m} \beta_\ell f_\ell(x)$$

- Polynomials: basis functions are monomials
- Periodic functions: $\sin(\ell \frac{x}{T}), \cos(\ell \frac{x}{T}), \ell \in \mathbb{N}, x \in [0, T]$
- Other nonlinear features: $\log x_k$, e^{x_k}



$(x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$ $(f_1(x), f_2(x), \ldots, f_m(x)) \in \mathbb{R}^m$

Stone-Weierstrass Theorem

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a continuous function defined over a closed and bounded set $A \subset \mathbb{R}^d$. Then, for any $\delta > 0$, there exists a polynomial $p : \mathbb{R}^d \to \mathbb{R}$ such that:

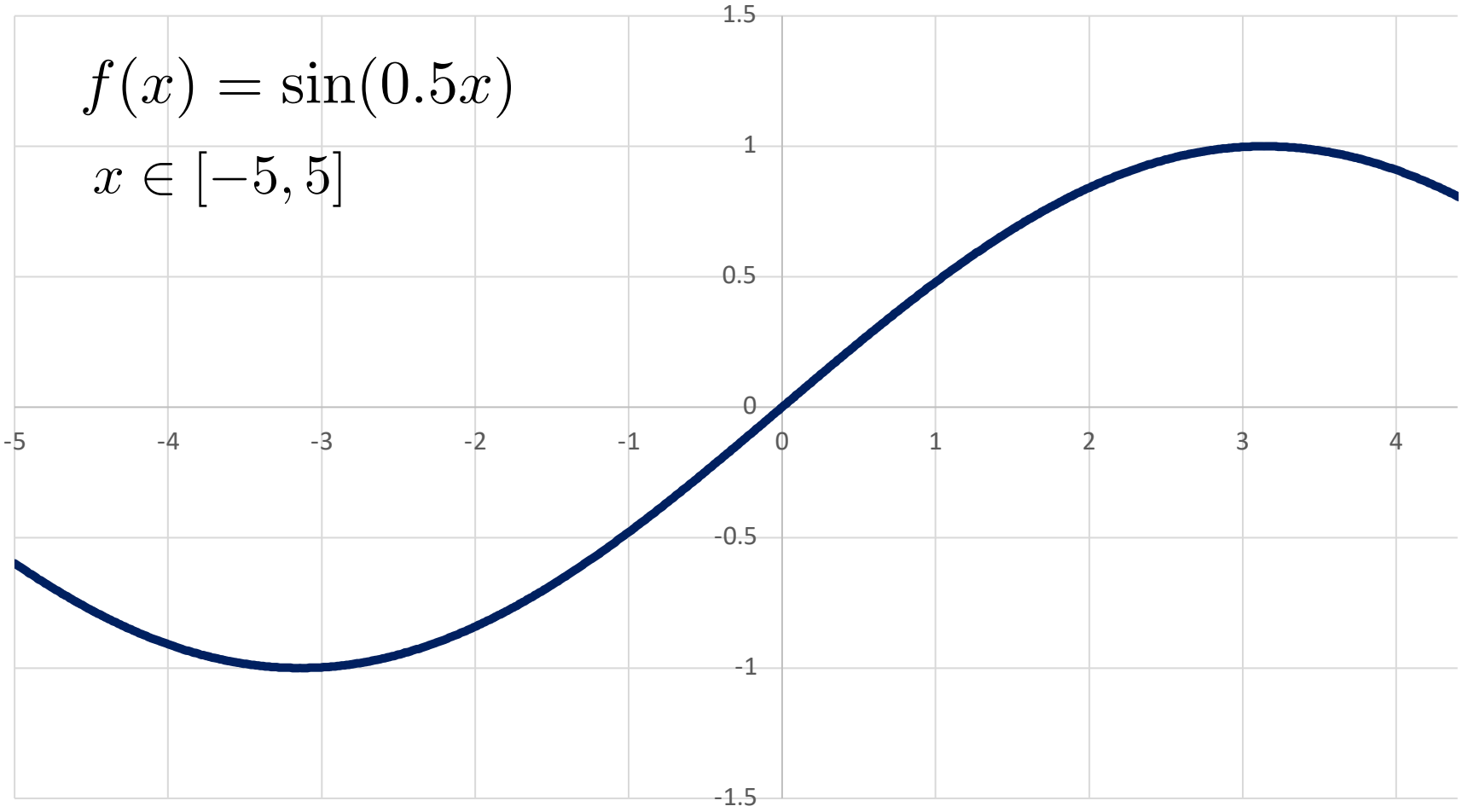
$$|f(x) - p(x)| \leq \delta$$

for all $x \in A$.

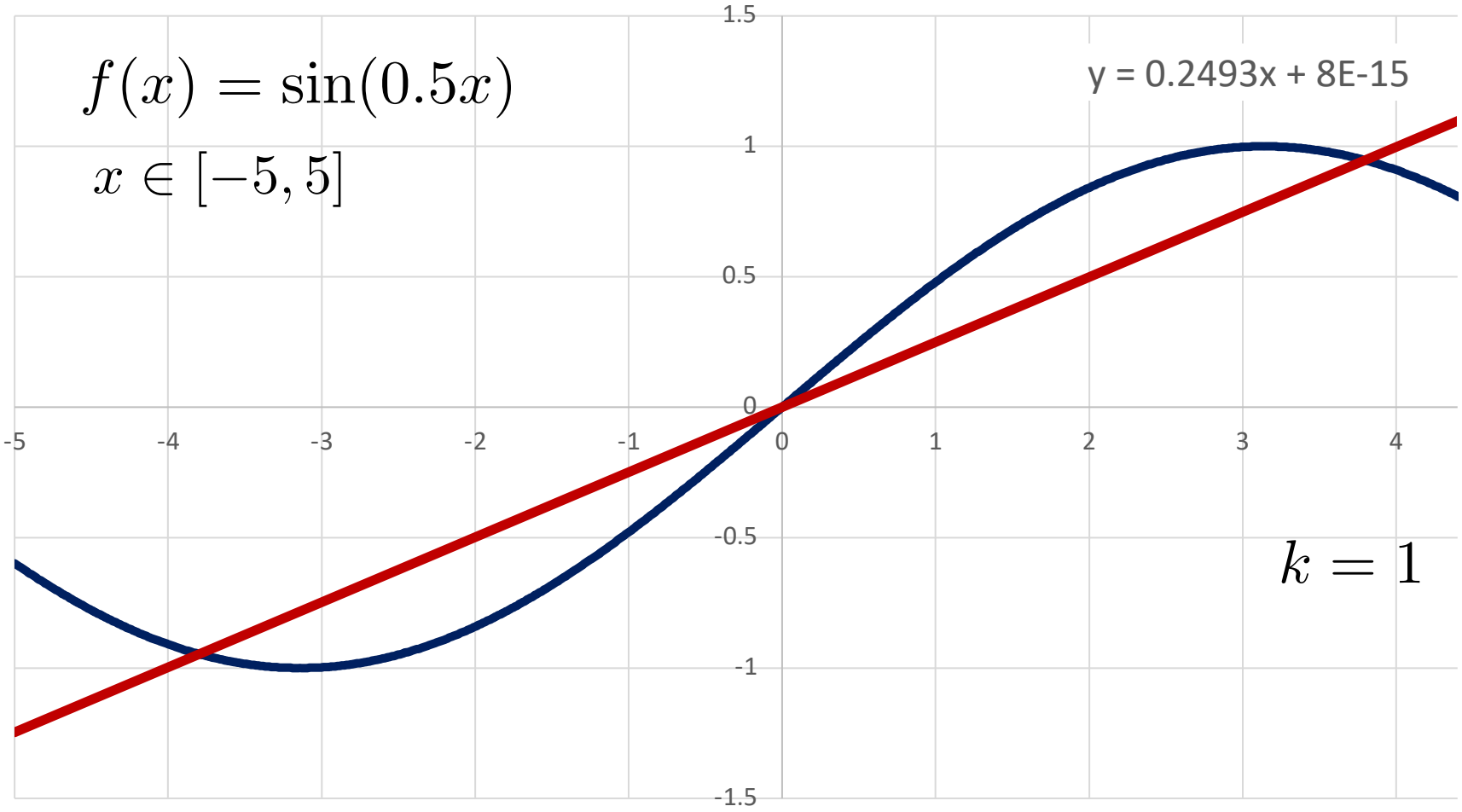
Stone-Weierstrass in Action

$$f(x) = \sin(0.5x)$$

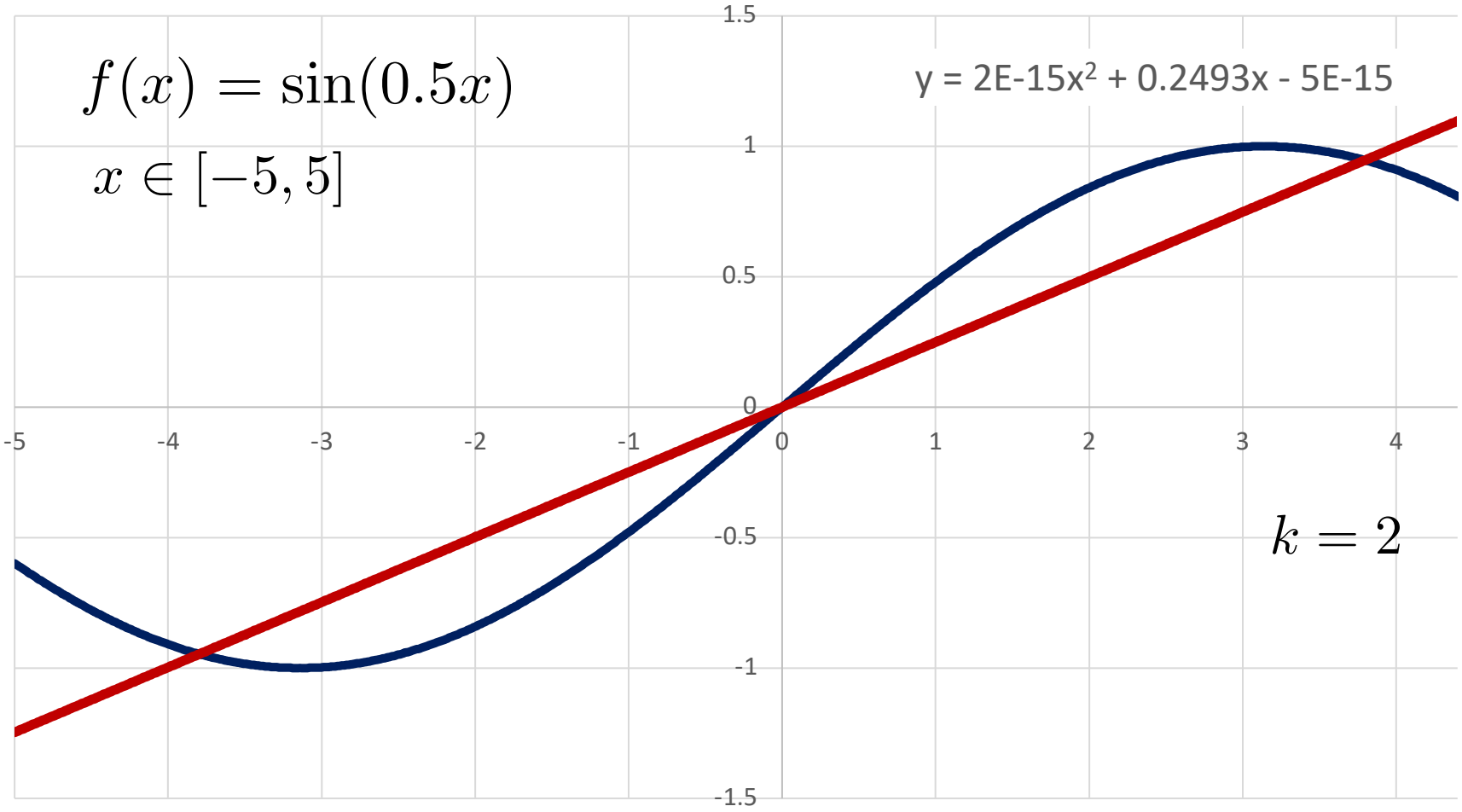
$$x \in [-5, 5]$$



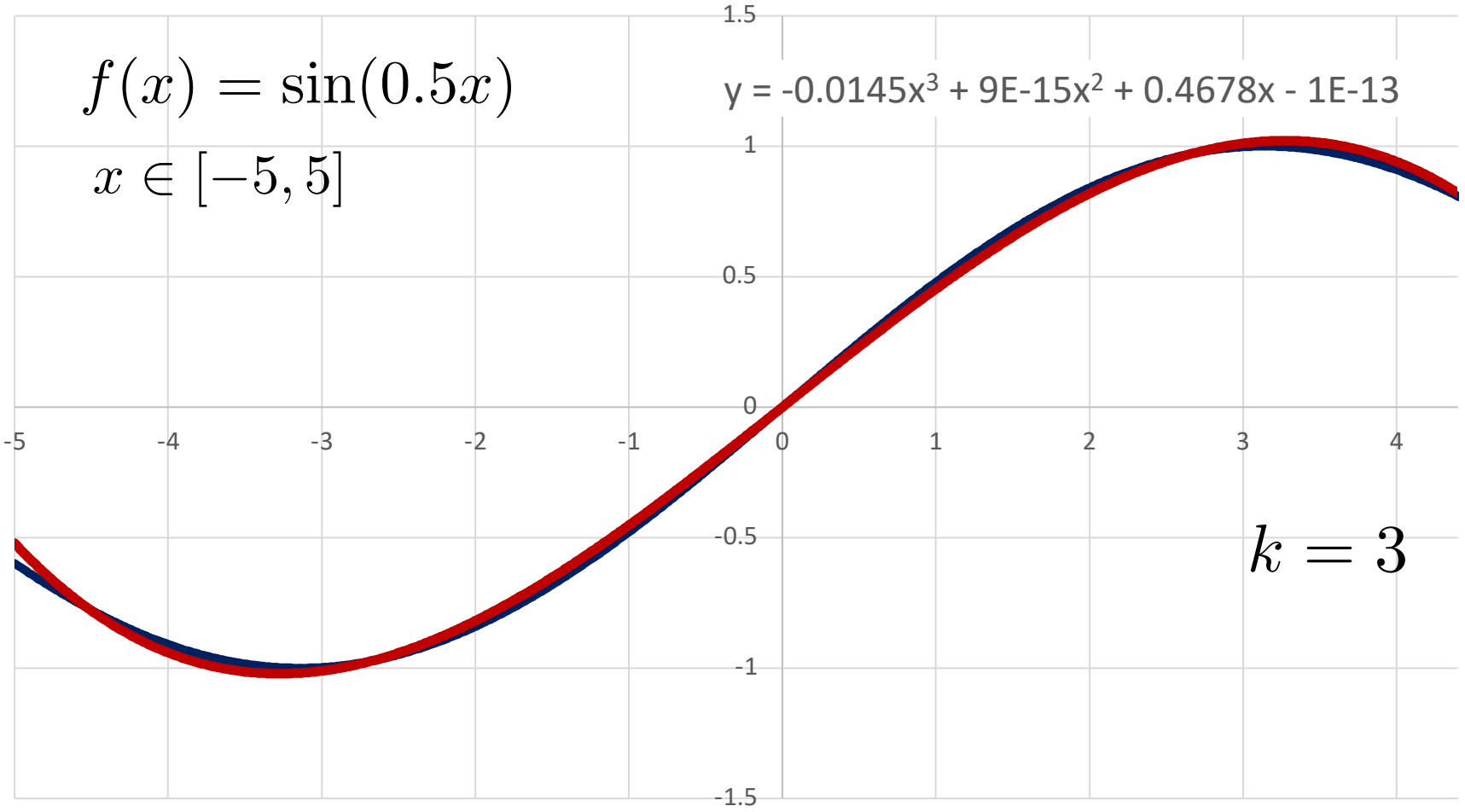
Stone-Weierstrass in Action



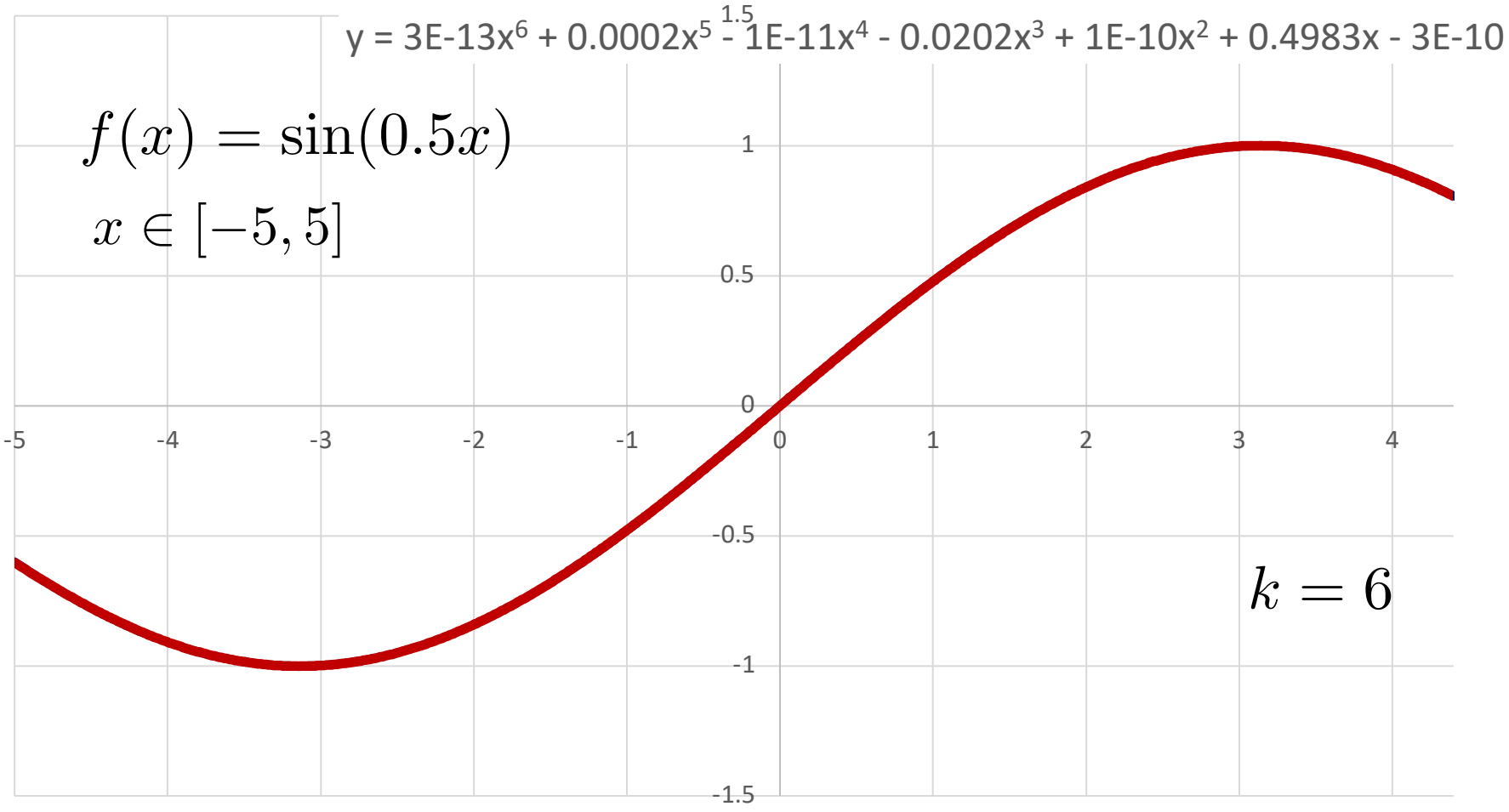
Stone-Weierstrass in Action



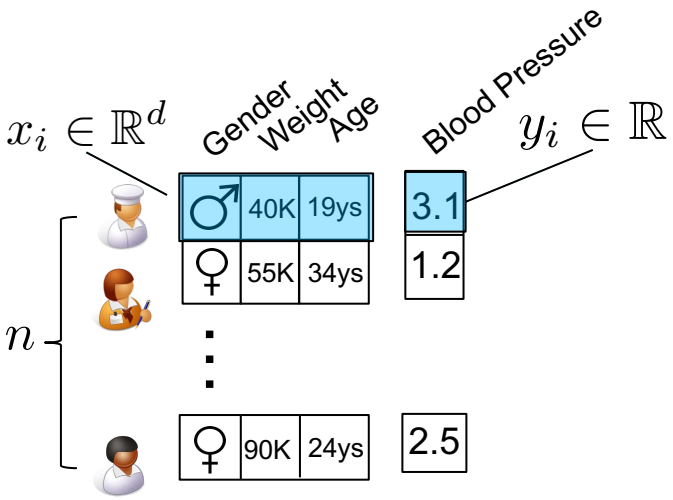
Stone-Weierstrass in Action

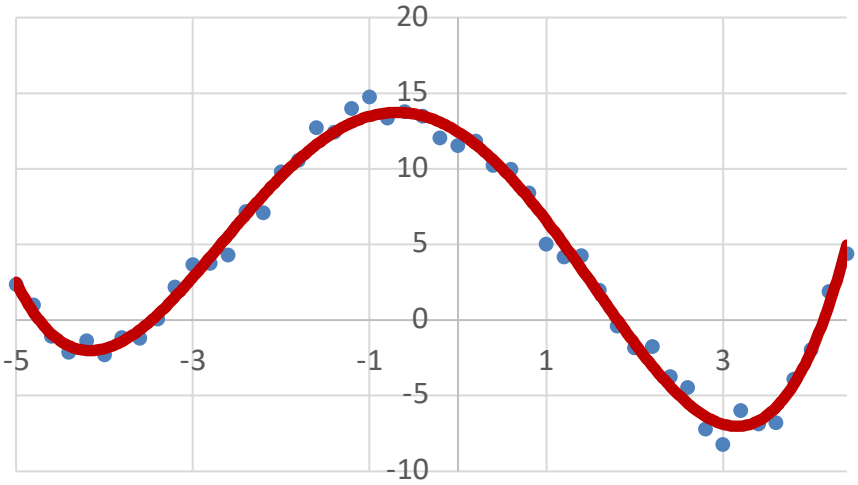


Stone-Weierstrass in Action



What Does This Imply?



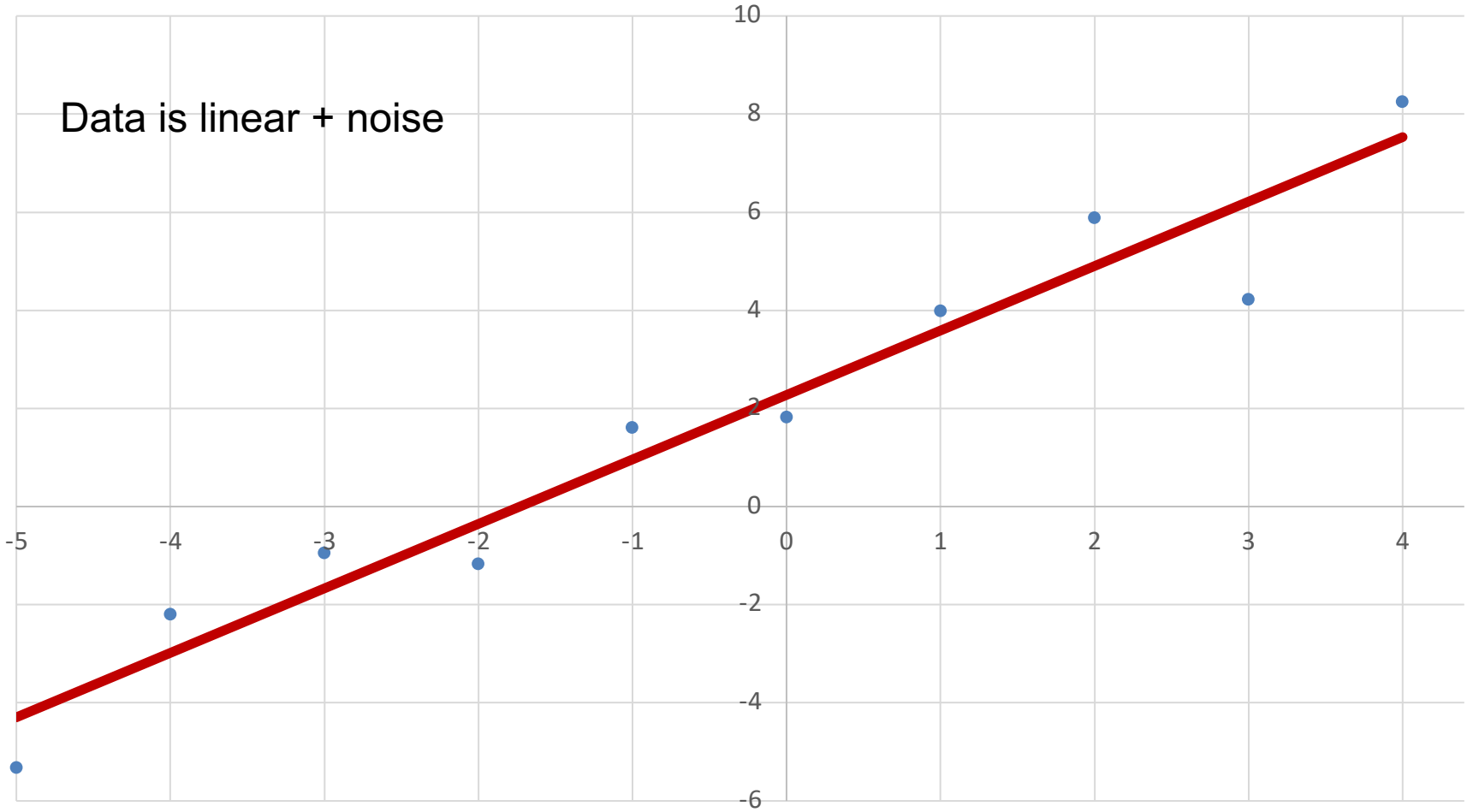


$$y_i = f(x_i) + \varepsilon_i, \quad i = 1, \ldots, n$$

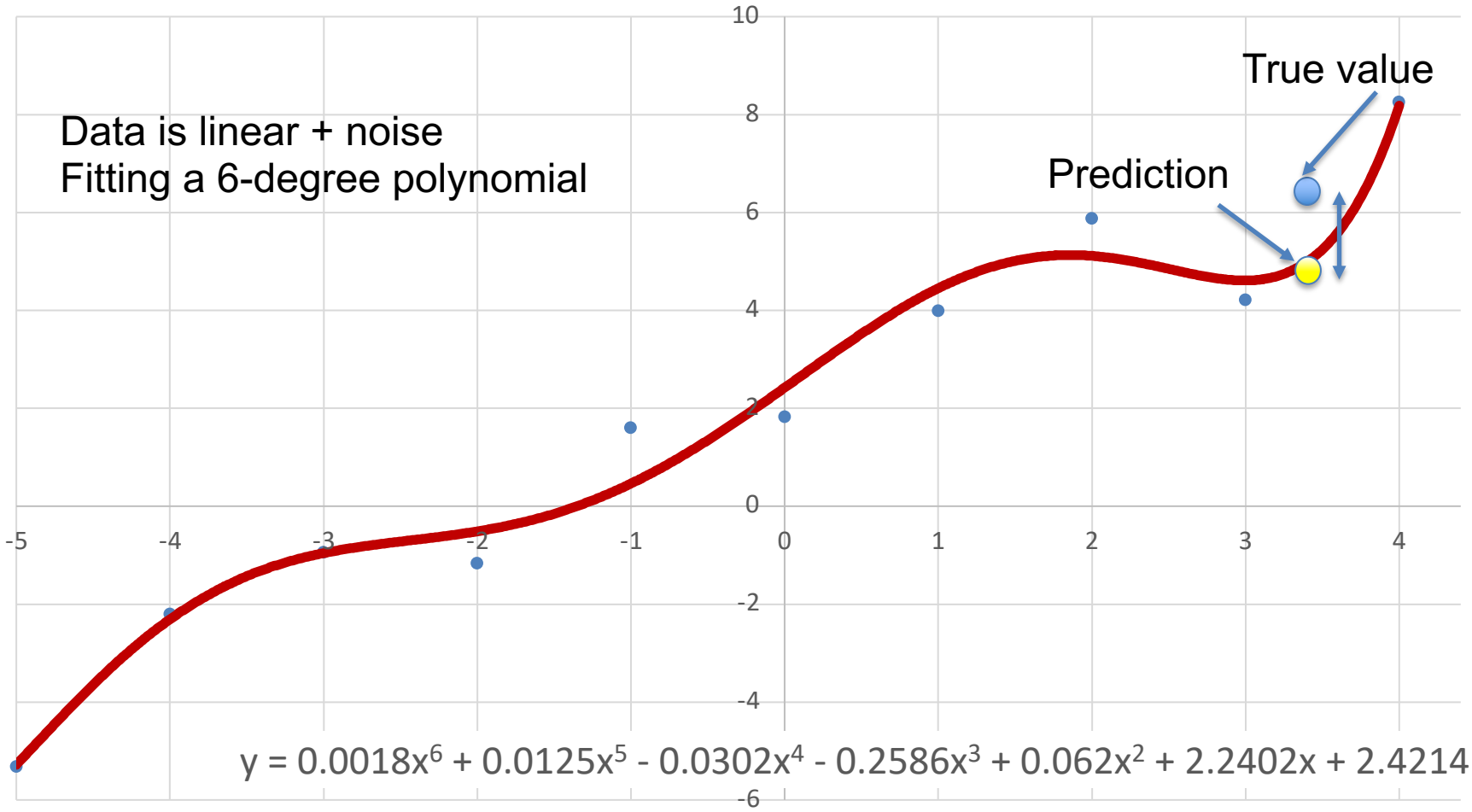
Suppose features x_i are in $[0, 100]^d$, and $f : \mathbb{R}^d \to \mathbb{R}$ is a continuous function.

Then, we can learn a polynomial that is **arbitrarily close to** f using linear regression!

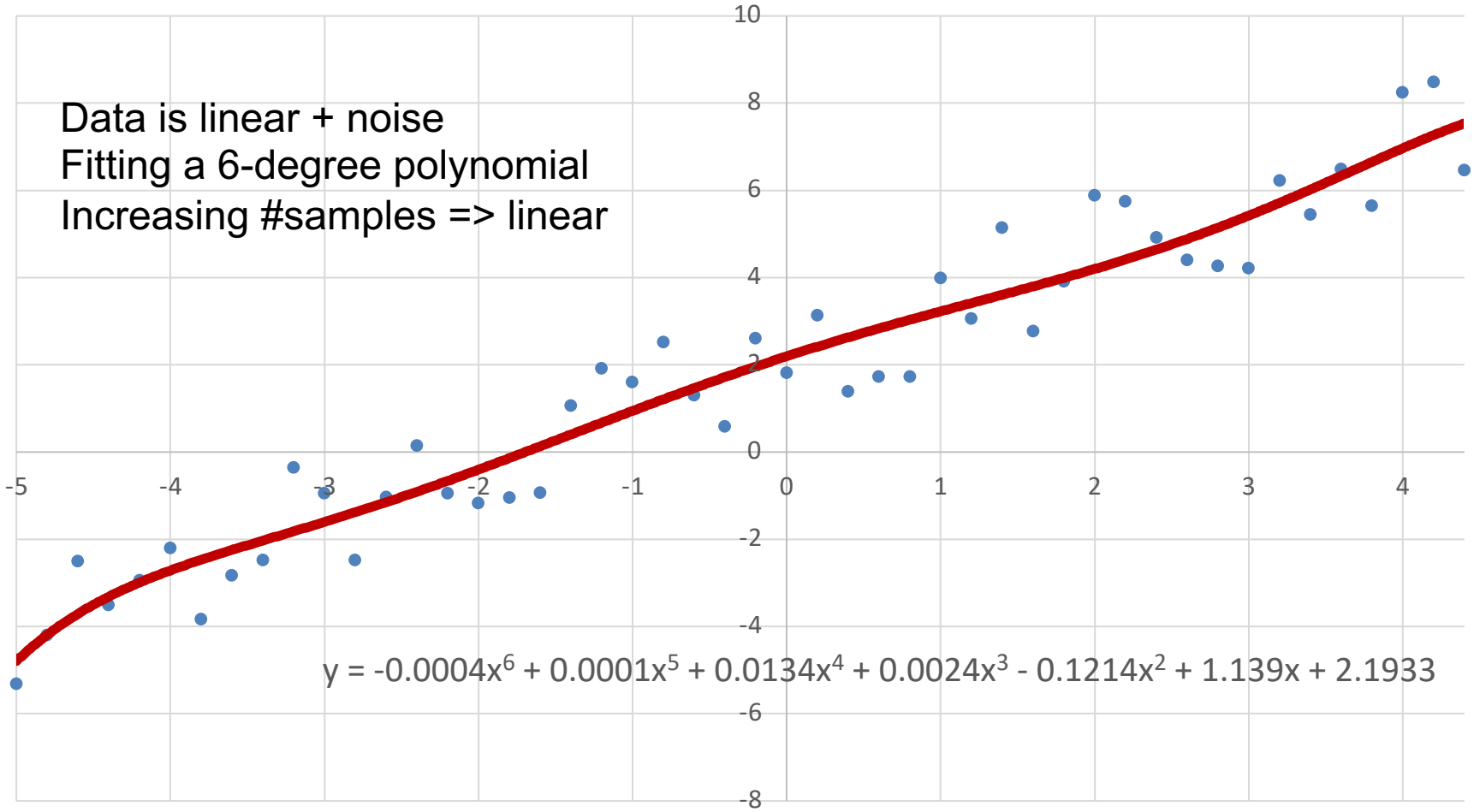
Over-fitting



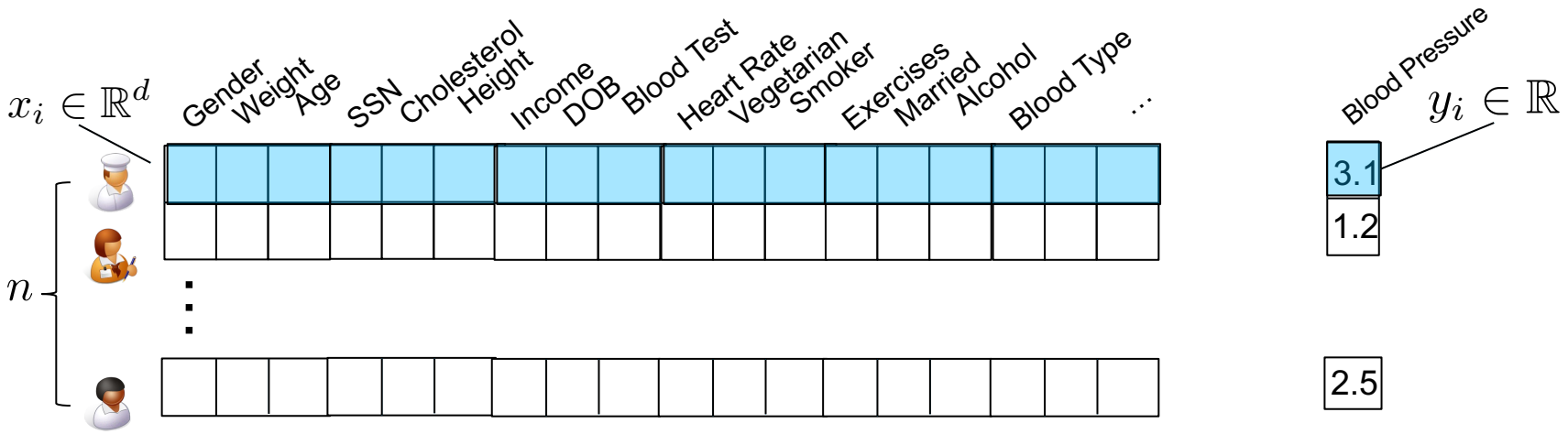
Over-fitting

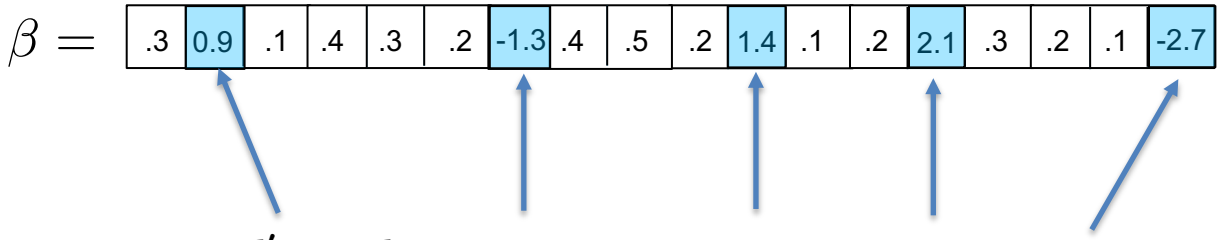


Over-fitting



Redundant Features May Exist in Data!





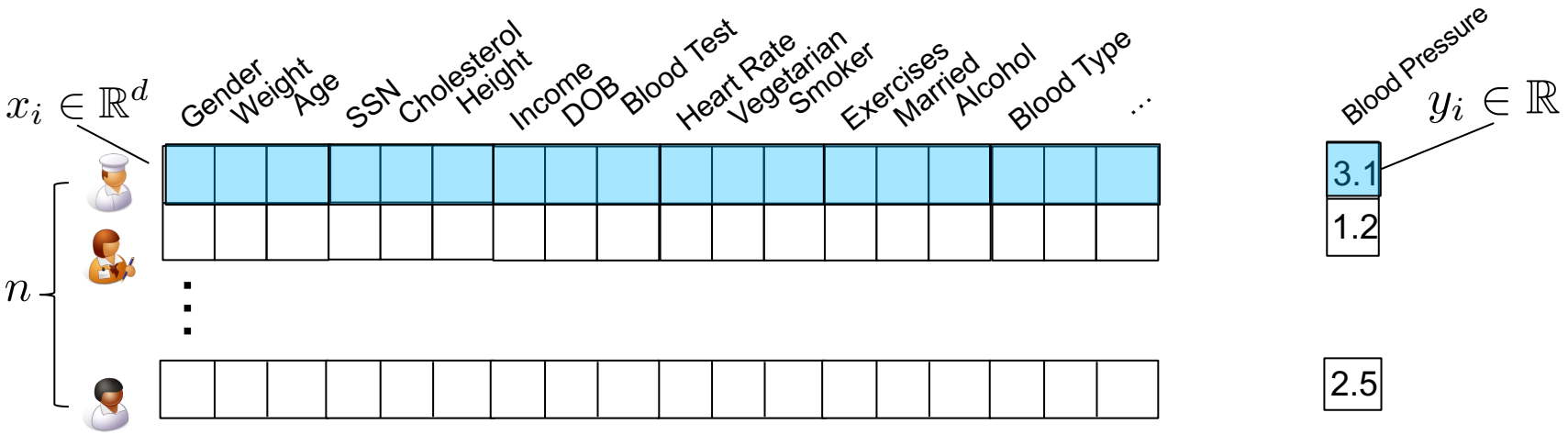
Only $d' \ll d$ features actually affect blood pressure!

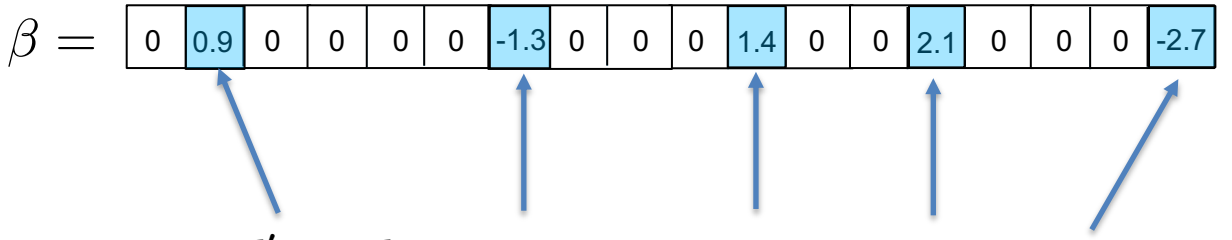
Linear Regression needs:

$$n = O(d) \gg O(d')$$

to learn β

Redundant Features May Exist in Data!





Only $d' \ll d$ features actually affect blood pressure!

Linear Regression needs:

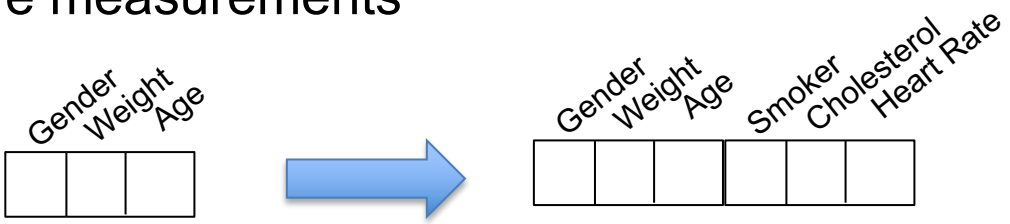
$$n = O(d) \gg O(d')$$

to learn β

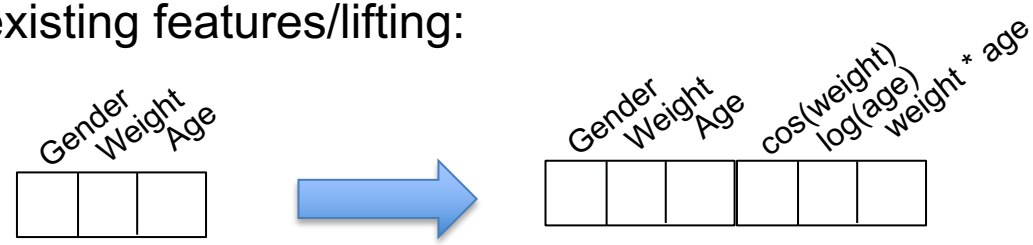
Summary

Increase number of features by:

- Collecting more measurements



- Transforming existing features/lifting:



- If features are redundant, regression will set corresponding weight to zero, but…

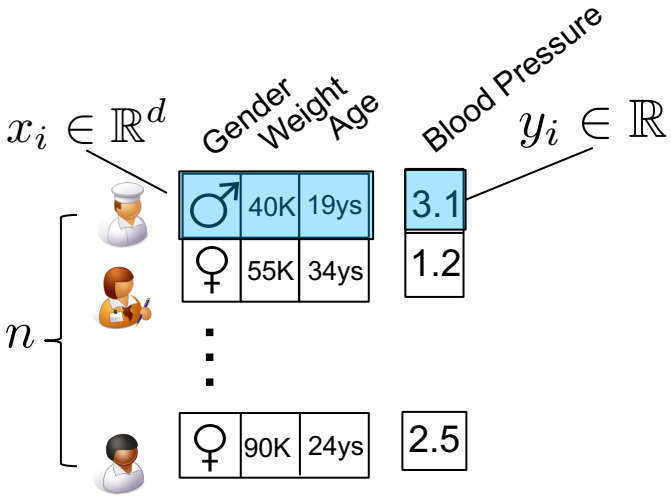
- ..it will require more samples to do this!!!

A New Challenge

- In practice, we are often just given a dataset

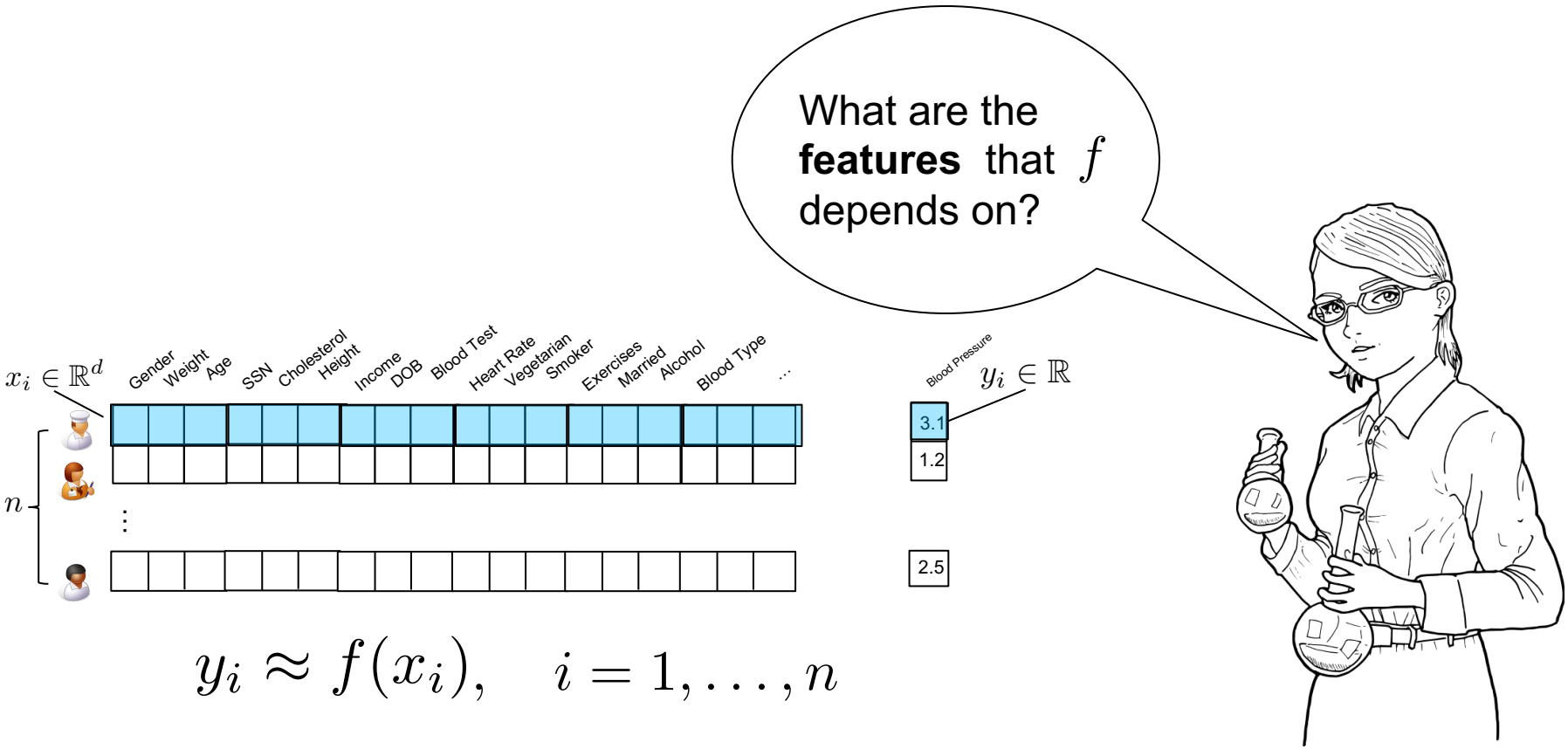
- We cannot increase n: we need to work with what we have

A New Challenge



$$y_i \approx f(x_i), \quad i = 1, \ldots, n$$

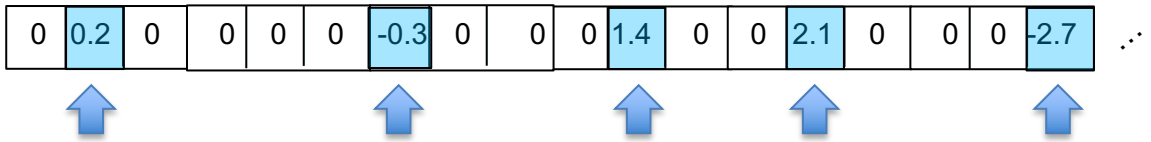
A New Challenge



Feature Selection!!!

Feature Selection

Q: How can we find out which features matter?



A Simple Solution: Just ask!!!

Experts

Statistician/Data Analyst

Ok, but can we do this from data alone?

- No experts
- Experts do not know either
- Discover features experts do not know
- …

Feature Selection

We actually need two things:

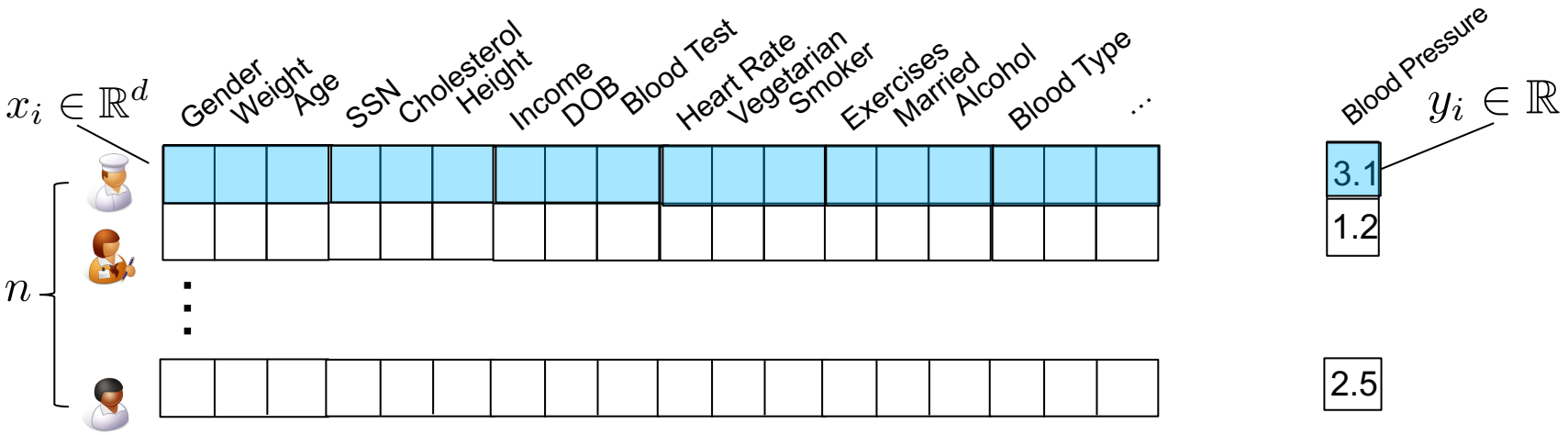
- A procedure for selecting features
- A way of measuring whether this selection is good

Feature Selection

We actually need two things:

- A procedure for selecting features
- **A way of measuring whether this selection is good**

How can I tell if I have a good set of features?

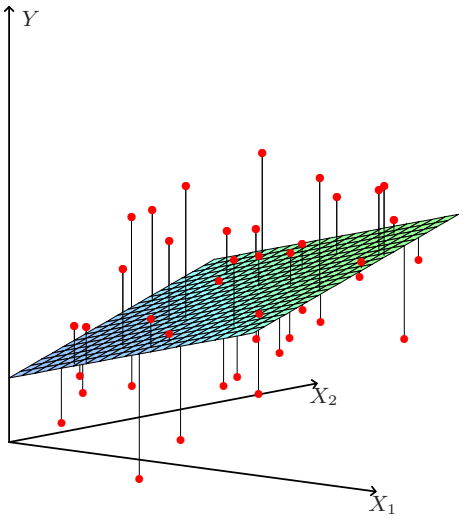


Least-Squares Estimator (LSE)

Residual Sum of Squares

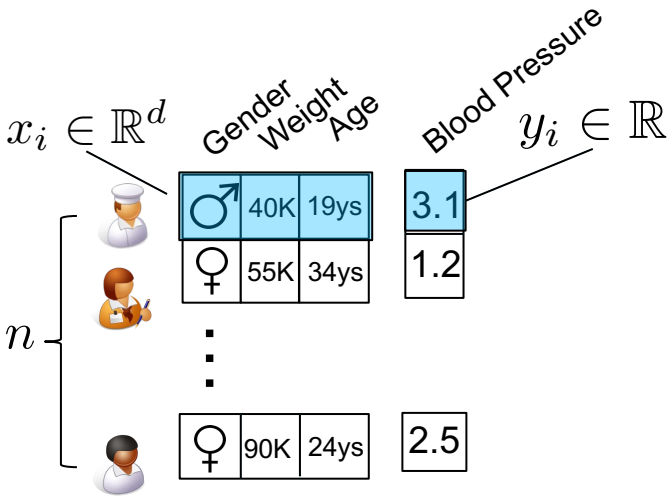
$\mathrm{RSS}(\beta)$

$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} (y_i - \langle \beta, x_i \rangle)^2$$
$$= \arg\min_{\beta \in \mathbb{R}^d} \|X\beta - y\|_2^2$$



Q: Can I use RSS to see if I have a good set of features?

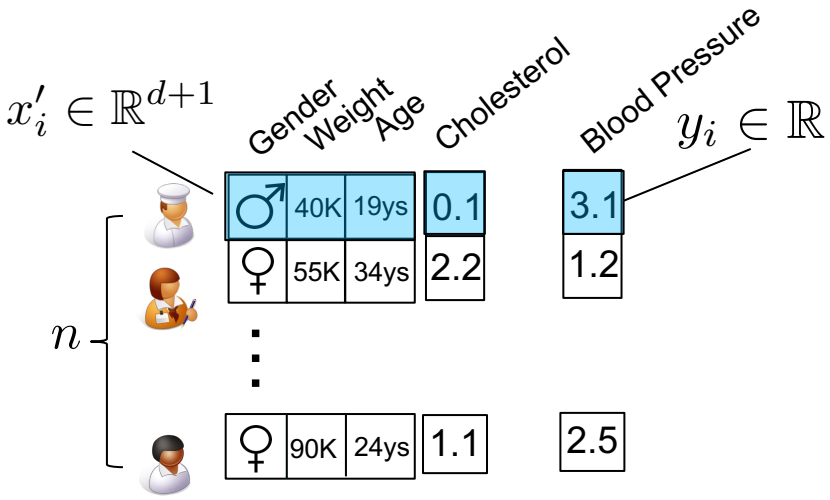
Residual Sum of Squares



$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} (y_i - \langle \beta, x_i \rangle)^2$$

$$\mathtt{RSS}(\hat{\beta}) = \sum_{i=1}^{n} (y_i - \hat{\beta}^\top x_i)$$

Residual Sum of Squares: Adding a New Feature



$$\hat{\beta}' = \arg\min_{\beta \in \mathbb{R}^{d+1}} \sum_{i=1}^{n} (y_i - \beta^\top x'_i)^2$$

$$\texttt{RSS}(\hat{\beta}') = \sum_{i=1}^{n} (y_i - \hat{\beta}'^\top x'_i)$$

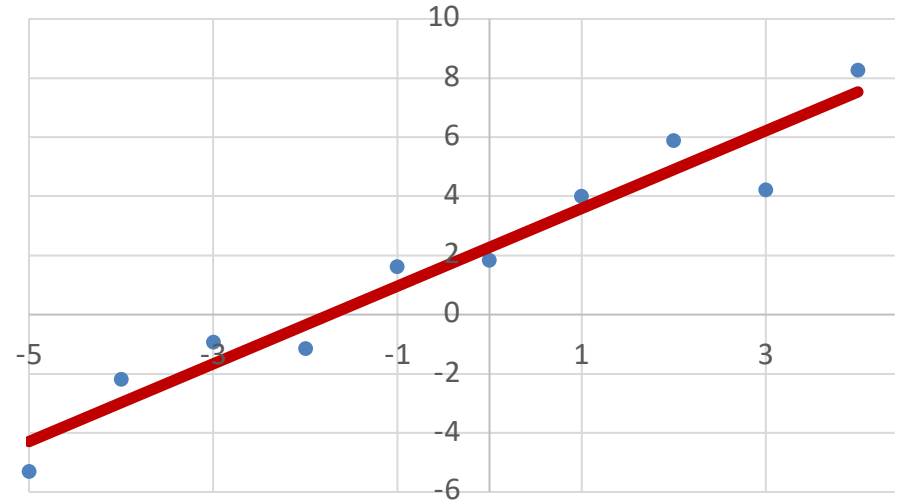
$$\overset{?}{\leqslant} \texttt{RSS}(\hat{\beta})$$

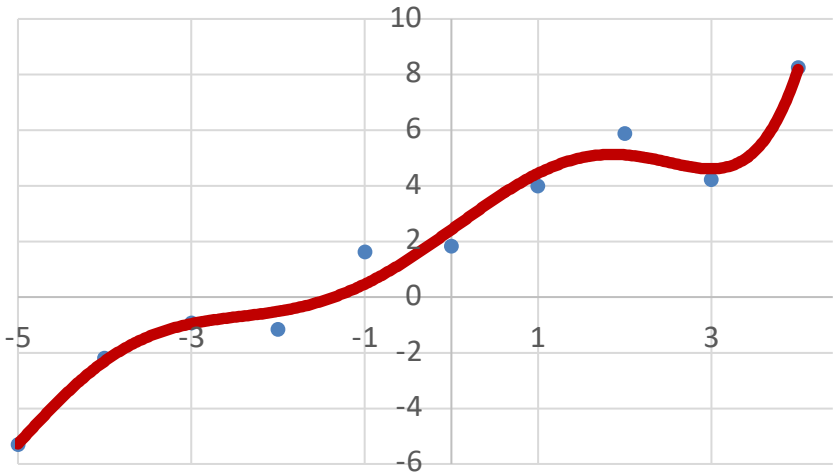
Adding Features Decreases RSS

Proof:

$$\begin{aligned}
\texttt{RSS}(\hat{\beta}') &= \min_{\beta \in \mathbb{R}^{d+1}} \texttt{RSS}(\beta) \\
&\leq \texttt{RSS}\big((\hat{\beta}, 0.0)\big) \\
&= \sum_{i=1}^{n} \left(y_i - (\hat{\beta}, 0.0)^\top x_i' \right)^2 \\
&= \sum_{i=1}^{n} \left(y_i - \hat{\beta}^\top x_i \right)^2 \\
&= \texttt{RSS}(\hat{\beta})
\end{aligned}$$

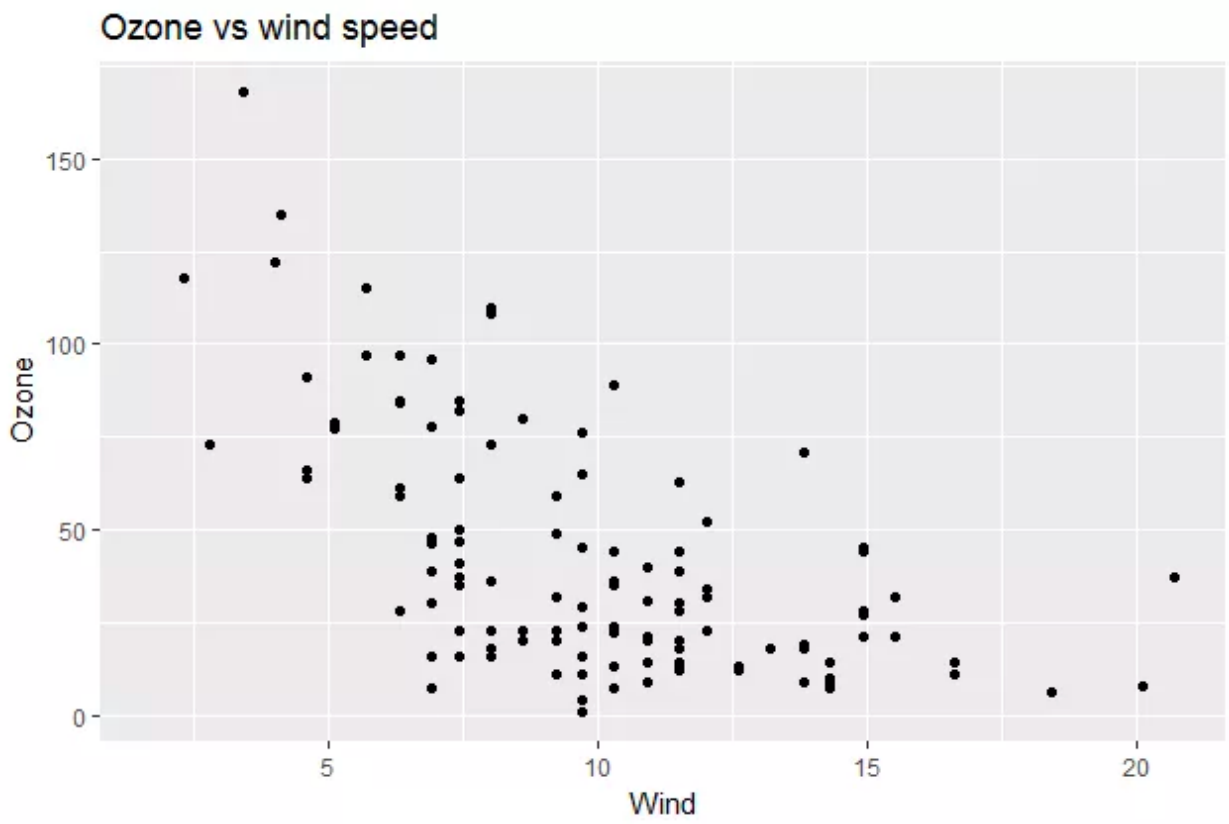
Same Principle As Overfitting!





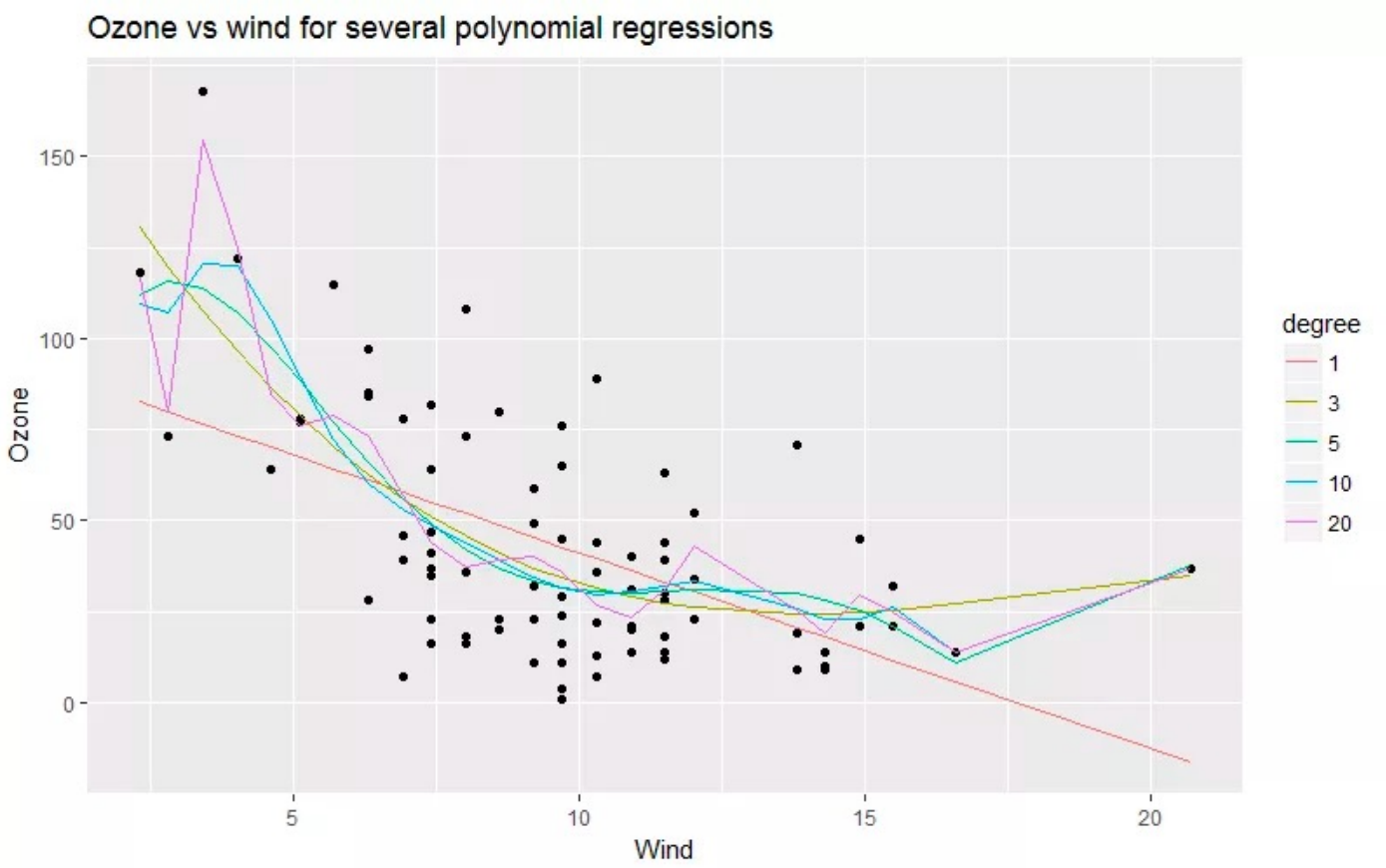
RSS(linear) > RSS(poly(6))

Overfitting (In Principle)



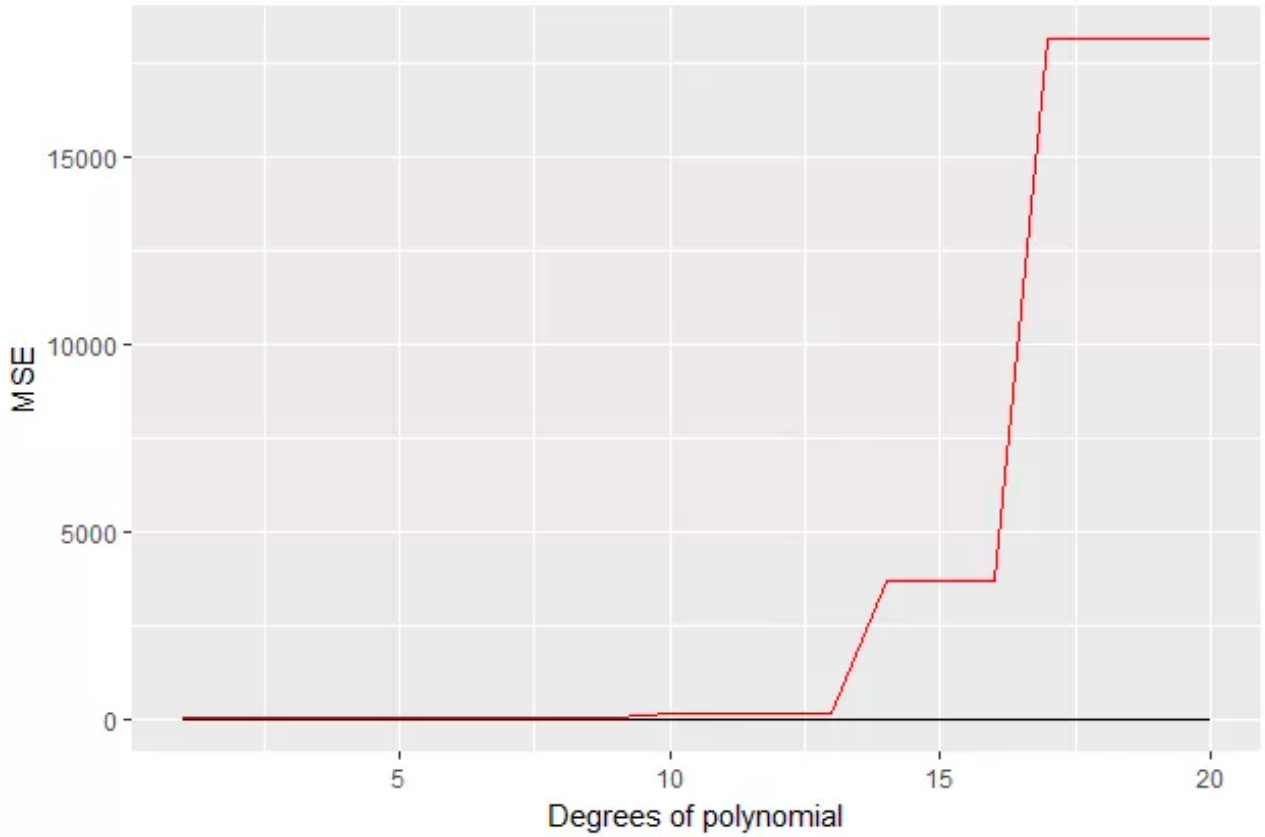
- The relationship does not seem linear hence, using polynomial regression may give some good results.

Overfitting (In Principle)



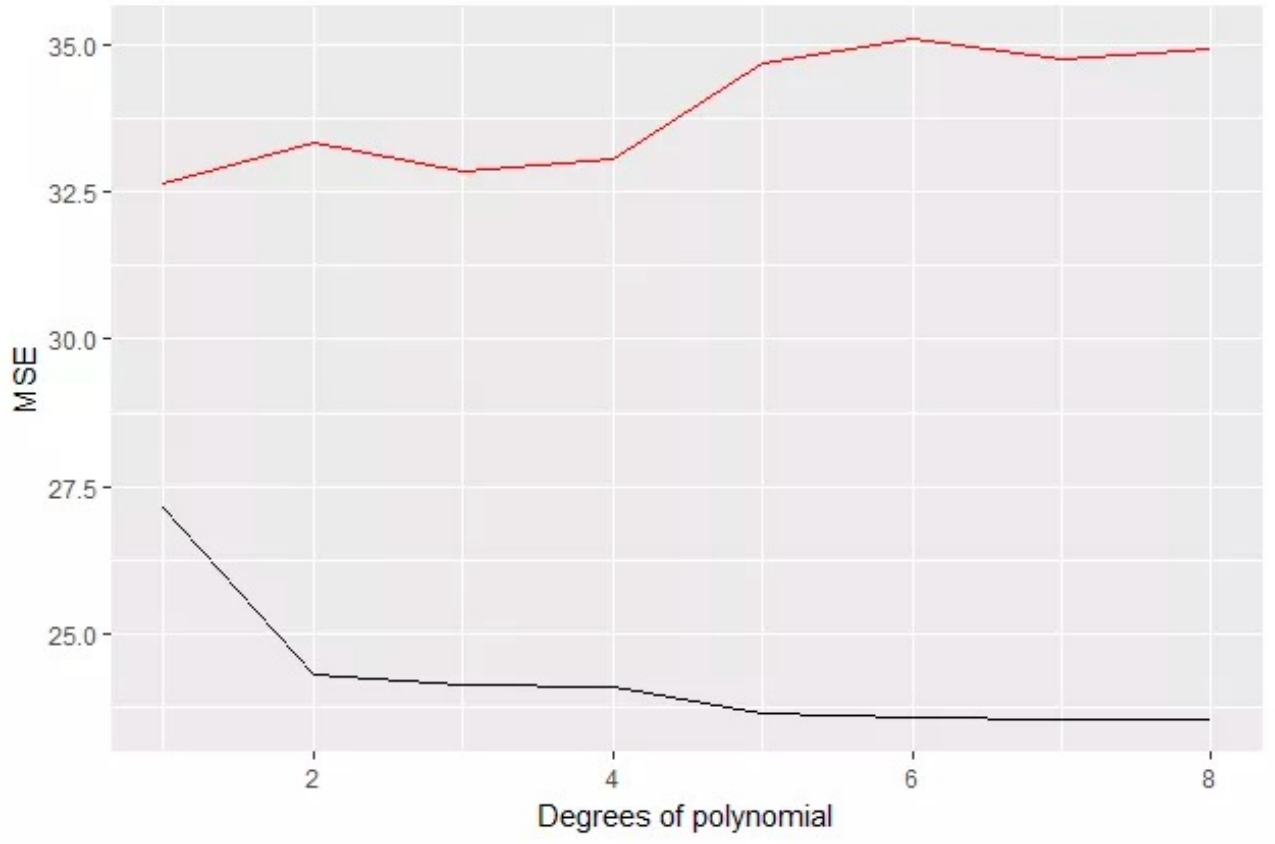
- Polynomial Regression: 5 fitted regression models
- Protocol Splits: 70% Training and 30% Test

Overfitting (In Principle)



- How is the mean square error (MSE) on training (black) vs testing (red)?

Overfitting (In Principle)

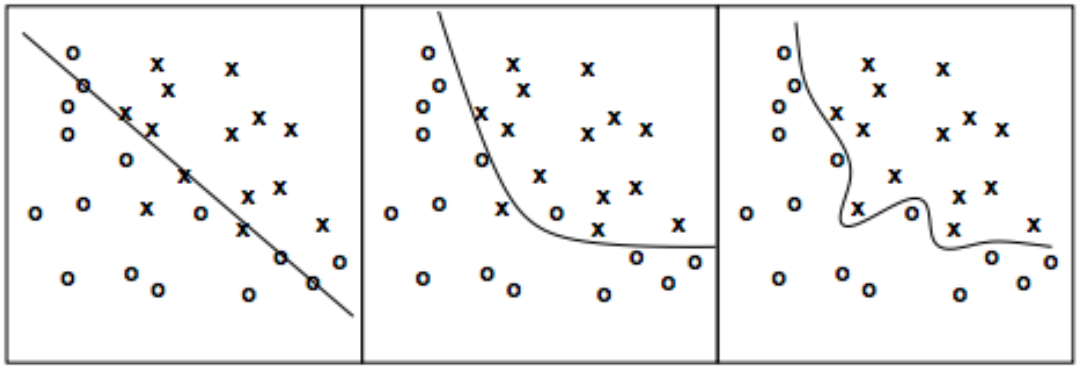


- Let's analyze MSE on test data for degrees 1 (black) and 8 (red)

Overfitting (In Principle)

- Occurs when model has too much freedom to fit data
  - Typically indicator that model used is too complex
- For instance: linear regression with a reasonable number of variables will never overfit the data, i.e., simple model of linear relationships between variables.
- On the other hand, random forest or neural net can easily overfit. They have a lot of parameters which they can minimize the loss function on (return later).

- **More complexity == More care must be put in**



Overfitting (Occham's Razor)

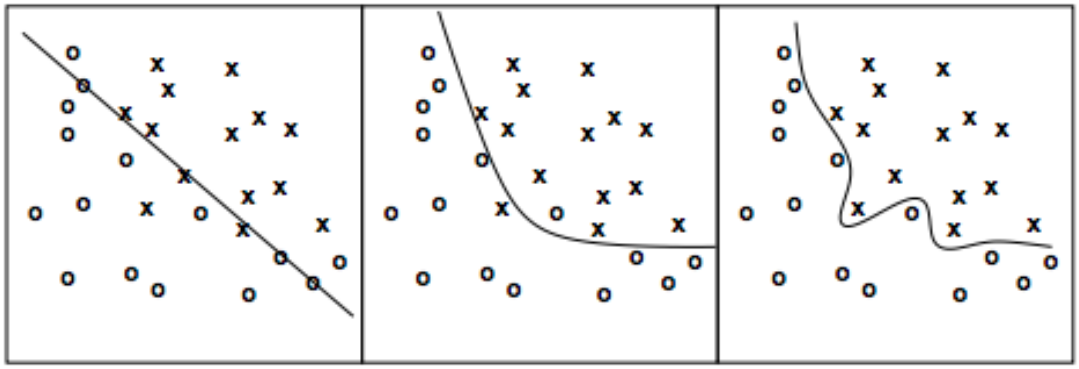
"All things being equal, the simplest solution tends to be the best one."

William of Ockham

Overfitting (Occham's Razor)

Overfitting (In Principle)

- Caused by a model having too much freedom. Hence most of the solutions to avoid overfitting add more constraints to the model:
  - Lasso and ridge regularization add a penalty for the parameters being too big or too numerous
  - Cross-validation assess the model performance on an independent data set
  - Early stopping stops the model when test error starts growing
  - And also: dropout, adding noise to input, …



Overfitting (In Principle)

- Caused by a model having too much freedom. Hence most of the solutions to avoid overfitting add more constraints to the model:
  - Lasso and ridge regularization add a penalty for the parameters being too big or too numerous
  - Cross-validation assess the model performance on an independent data set
  - Early stopping stops the model when test error starts growing
  - And also: dropout, adding noise to input, …

Solution:

Overfitting (In Principle)

We say a hypothesis 'overfits' the data if we can find a different hypothesis with more training error but less actual data error.

So, let *h* be a hypothesis. Let *h'* also be a hypothesis.

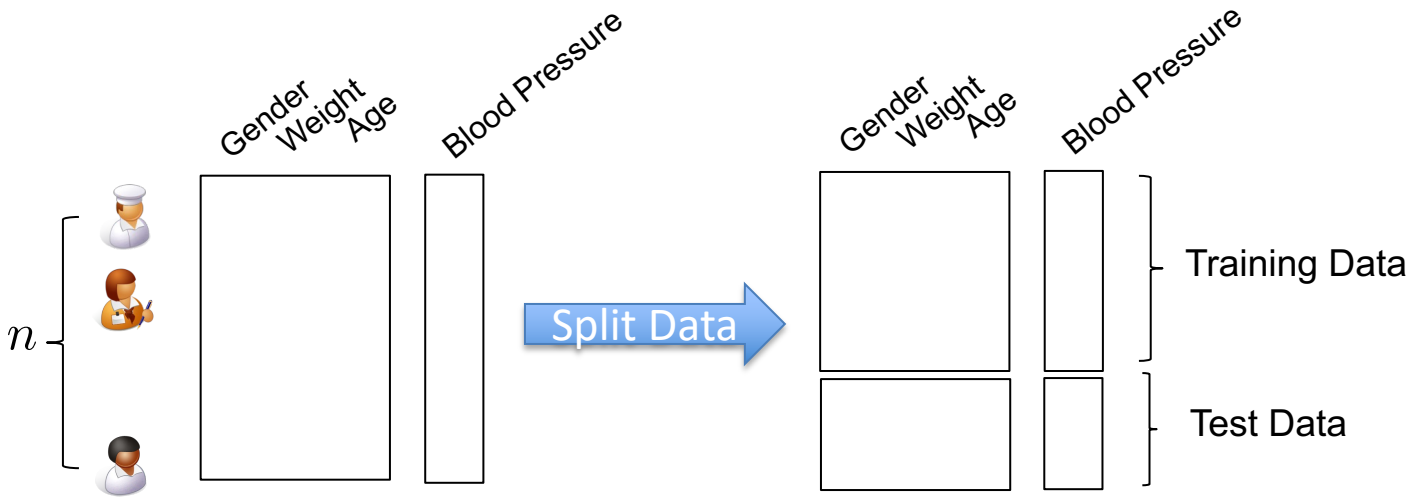
$$if\ error_{train}(h) < error_{train}(h')\ \&error_{train}(h) < error_{all\ data}(h')$$

Indication of overfitting: evaluation on training is robust, while testing yields poor performance

Solution:

- Avoid being too precise
  - Limit nodes or features
  - Limit training iterations
  - Use validation set

Estimating EPE



- **Train** $\hat{\beta}$ by minimizing:

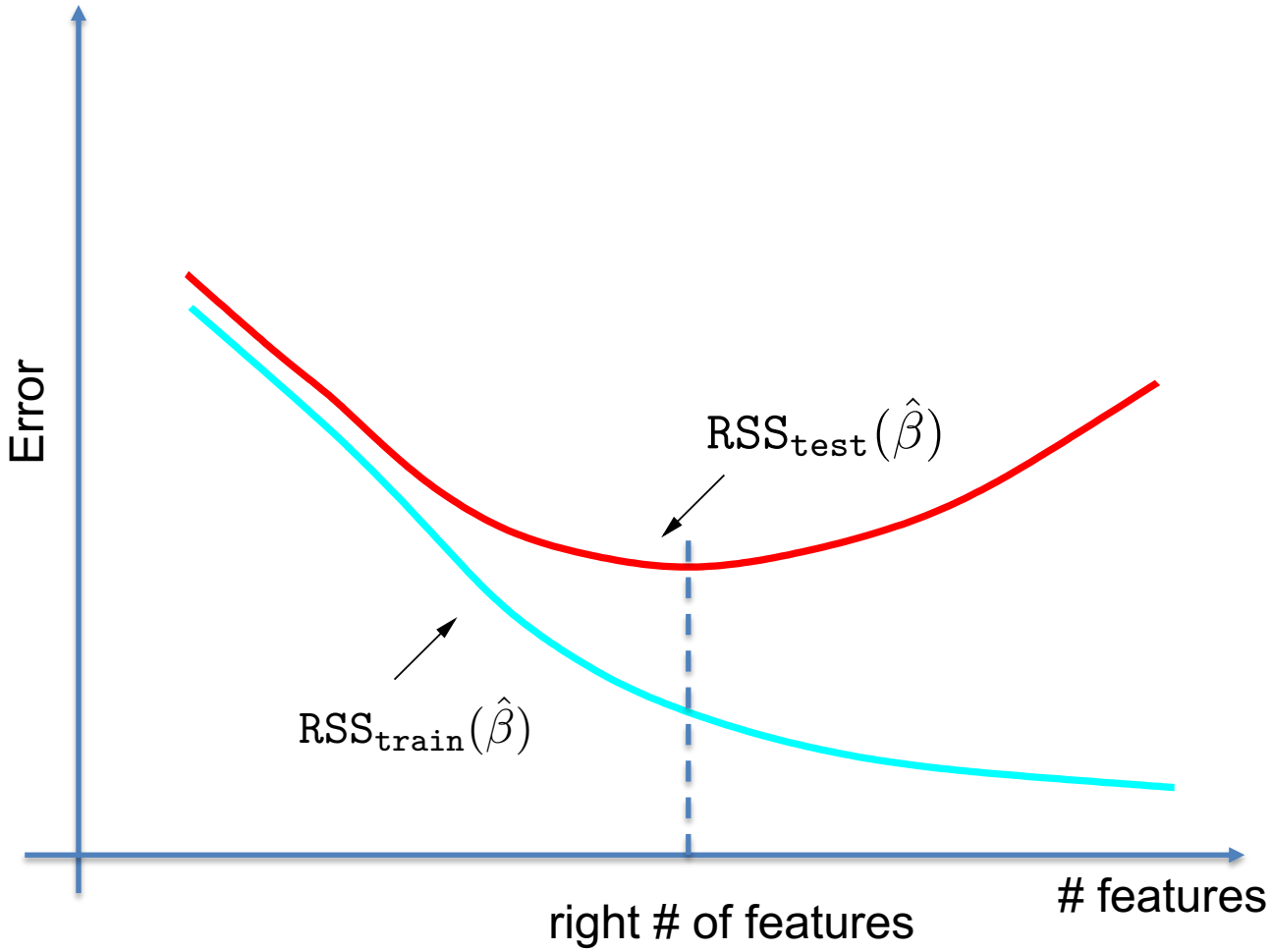
$$\text{RSS}_{\texttt{train}}(\beta) = \sum_{i \in \texttt{train}} (y_i - \beta^\top x_i)^2$$

- **Test** $\hat{\beta}$ by evaluating:

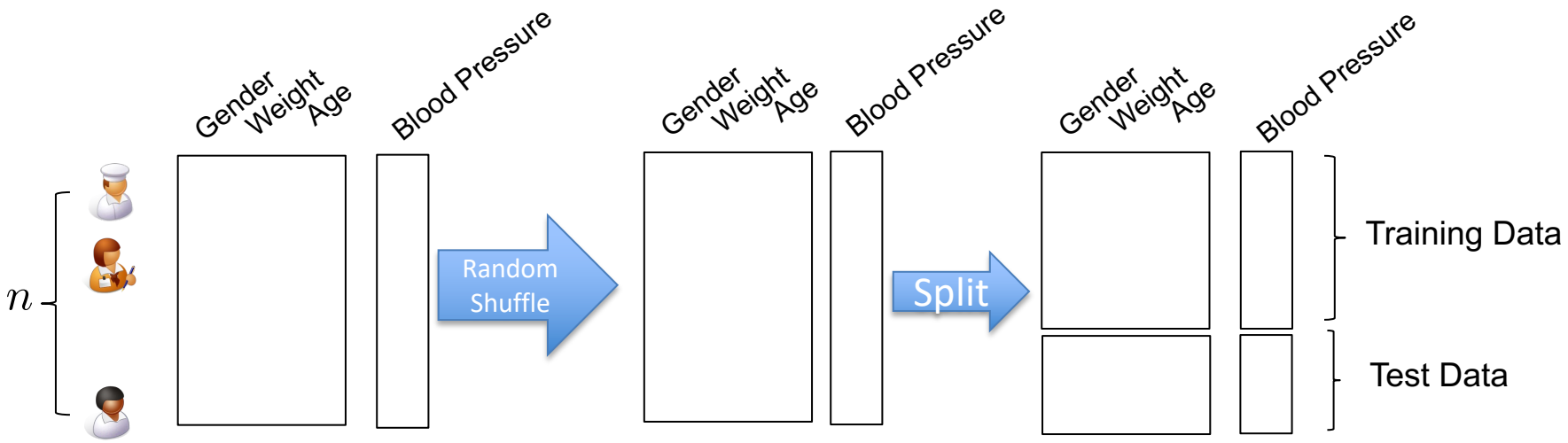
$$\text{RSS}_{\texttt{test}}(\hat{\beta}) = \sum_{i \in \texttt{test}} (y_i - \hat{\beta}^\top x_i)^2$$

"Proxy" for EPE!!

Feature Selection Revisited



Improvement #1



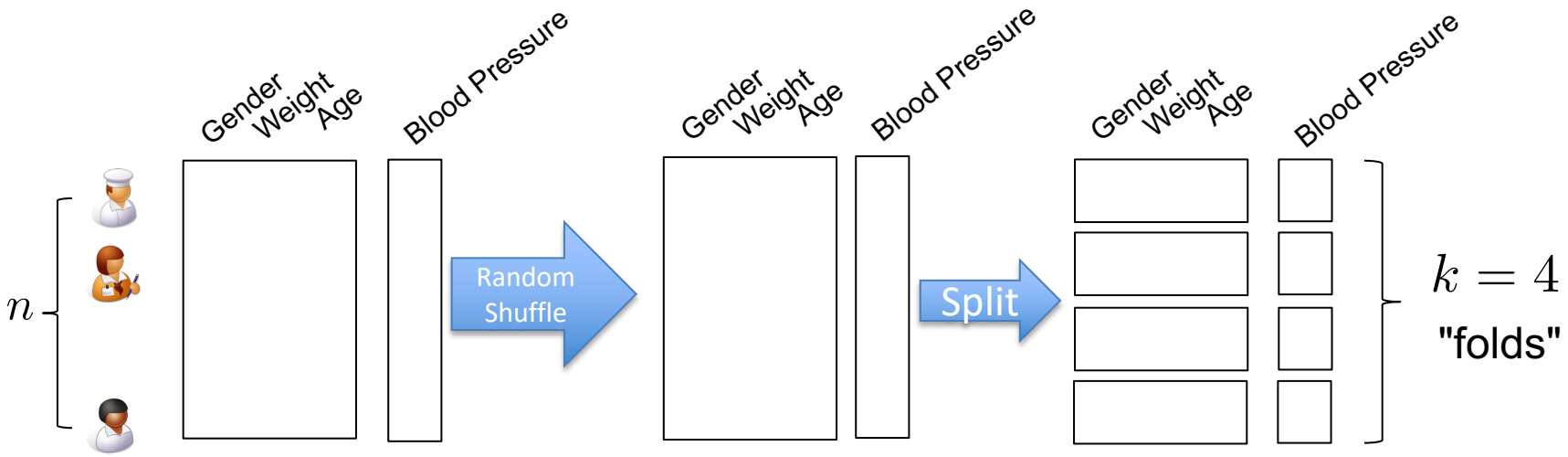
- **Train** $\hat{\beta}$ by minimizing:

$$\texttt{RSS}_{\texttt{train}}(\beta) = \sum_{i \in \texttt{train}} (y_i - \beta^\top x_i)^2$$

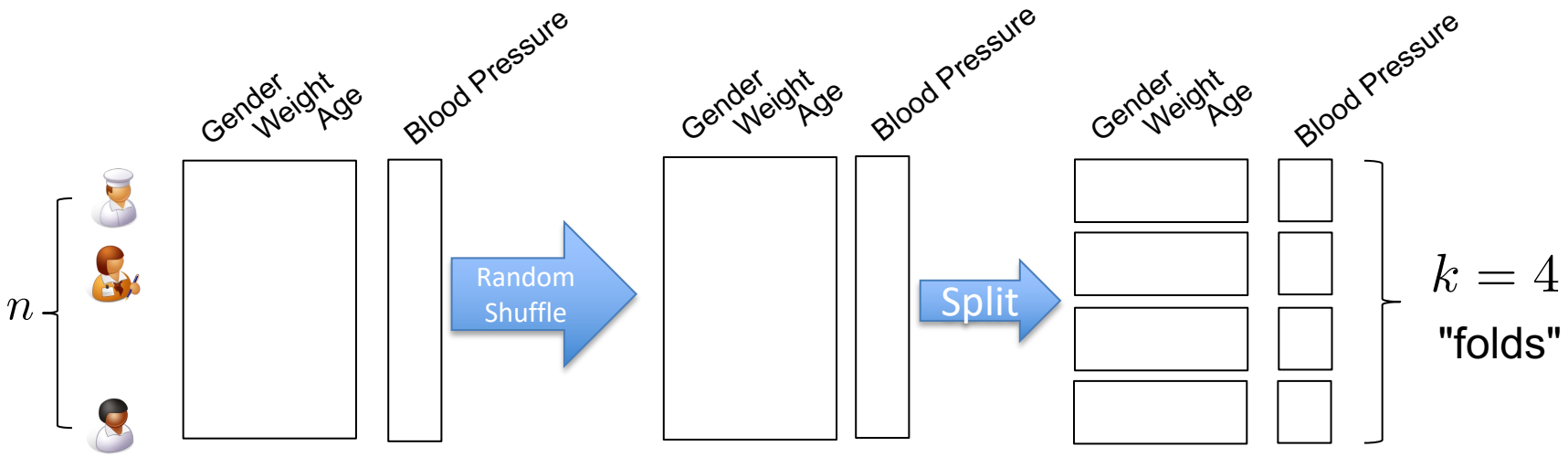
- **Test** $\hat{\beta}$ by evaluating:

$$\texttt{RSS}_{\texttt{test}}(\hat{\beta}) = \sum_{i \in \texttt{test}} (y_i - \hat{\beta}^\top x_i)^2$$

Improvement #2: k-fold Cross Validation

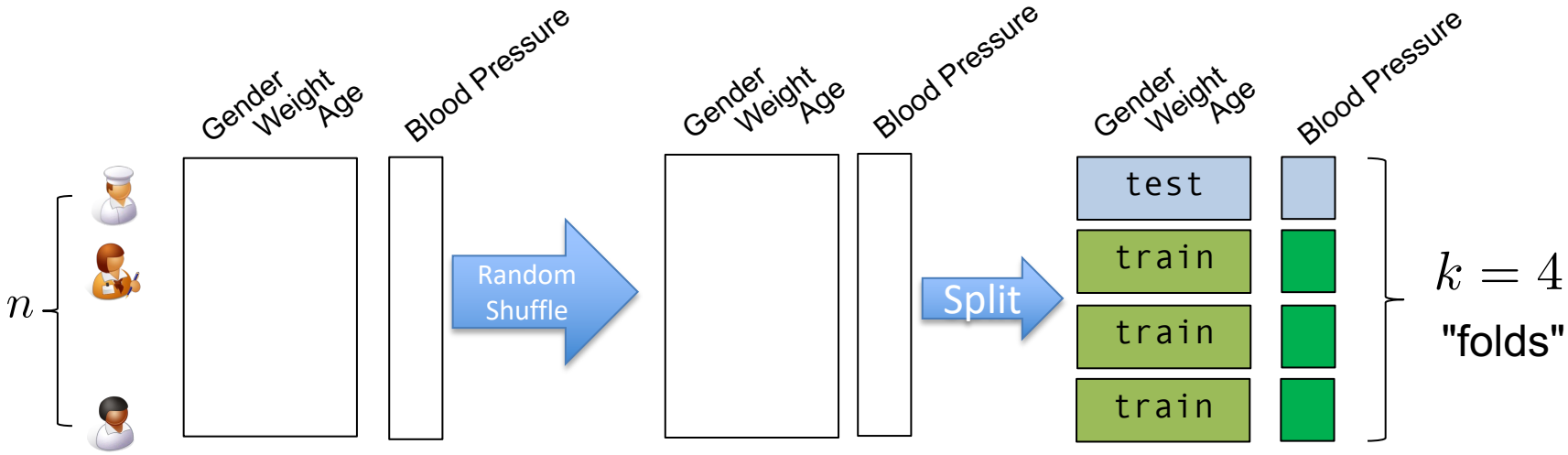


Improvement #2: k-fold Cross Validation



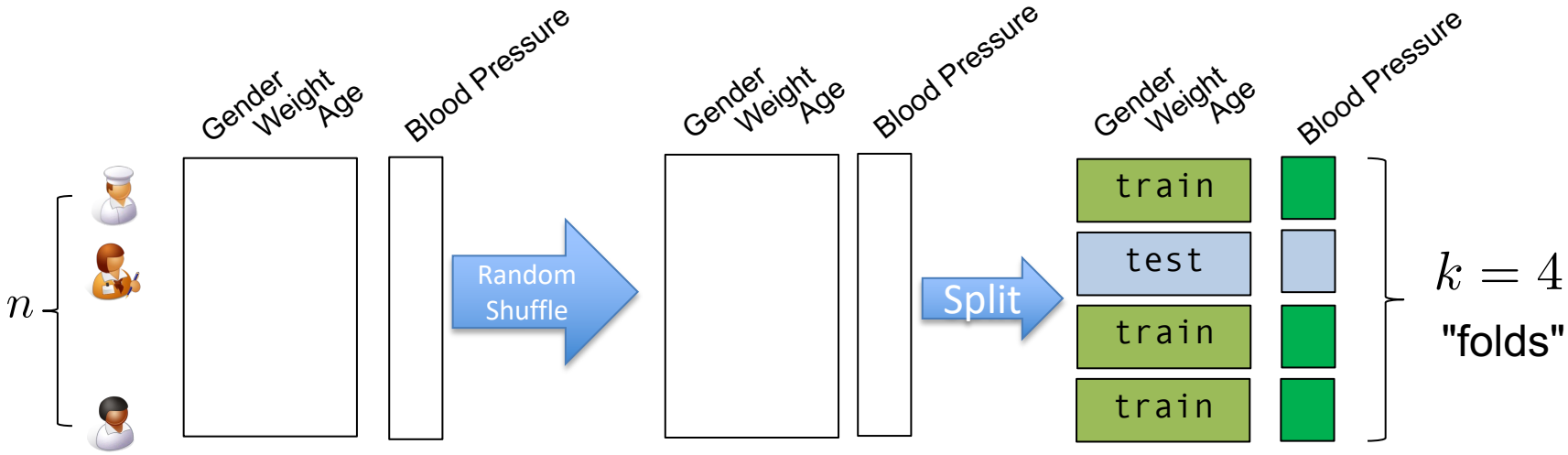
- For each fold $\ell = 1, \ldots, k$:
  - Set $\texttt{test}_\ell$ to include all data in fold ℓ.
  - Put remaining folds in $\texttt{train}_\ell$

Improvement #2: k-fold Cross Validation



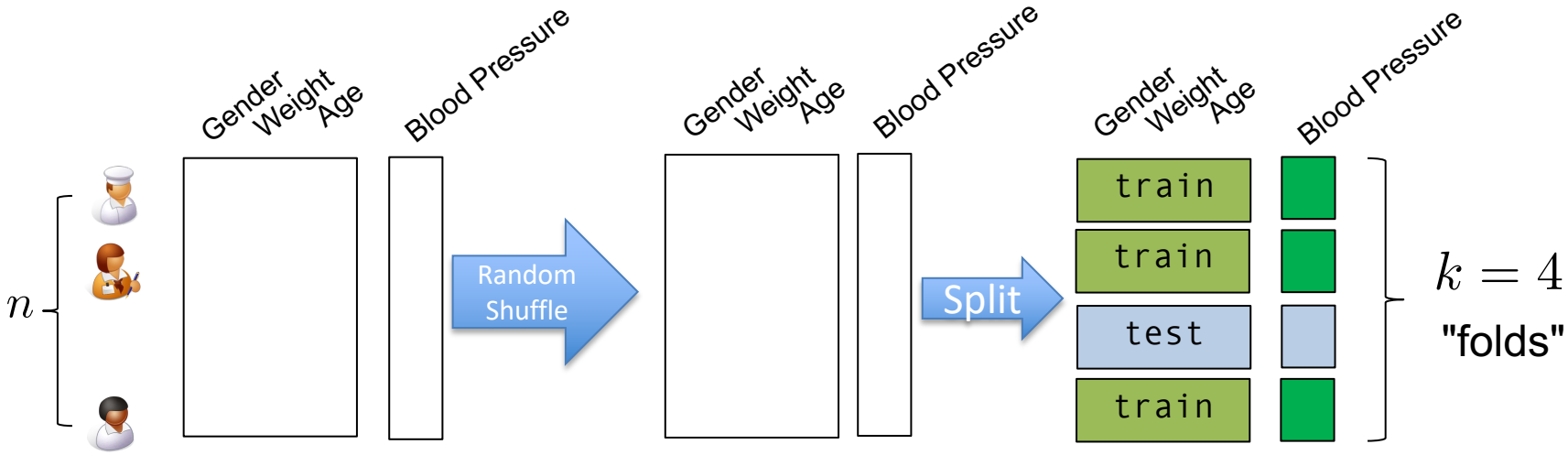
- For each fold $\ell = 1, \ldots, k$:
  - Set $\texttt{test}_\ell$ to include all data in fold ℓ.
  - Put remaining folds in $\texttt{train}_\ell$
  - **Train** $\hat{\beta}$ by minimizing: $\texttt{RSS}_{\texttt{train}_\ell}(\beta) = \sum_{i \in \texttt{train}_\ell} (y_i - \beta^\top x_i)^2$
  - **Test** $\hat{\beta}$ by evaluating: $\texttt{RSS}_{\texttt{test}_\ell}(\hat{\beta}) = \sum_{i \in \texttt{test}_\ell} (y_i - \hat{\beta}^\top x_i)^2$

Improvement #2: k-fold Cross Validation



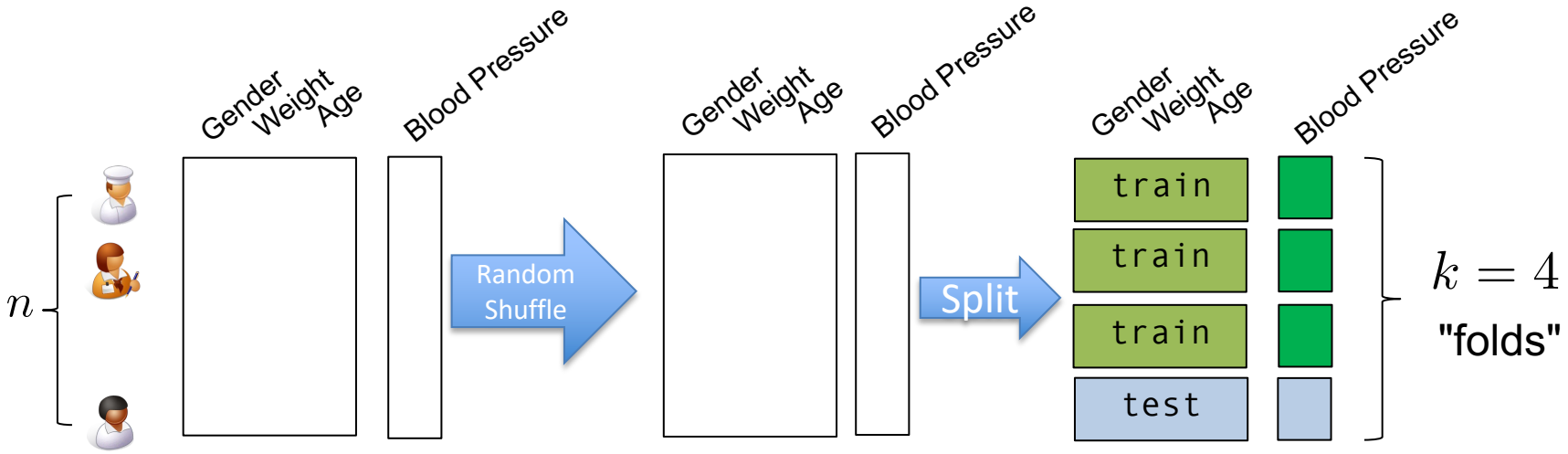
- For each fold $\ell = 1, \ldots, k$:
  - Set $\texttt{test}_\ell$ to include all data in fold ℓ.
  - Put remaining folds in $\texttt{train}_\ell$
  - **Train** $\hat{\beta}$ by minimizing: $\texttt{RSS}_{\texttt{train}_\ell}(\beta) = \sum_{i \in \texttt{train}_\ell} (y_i - \beta^\top x_i)^2$
  - **Test** $\hat{\beta}$ by evaluating: $\texttt{RSS}_{\texttt{test}_\ell}(\hat{\beta}) = \sum_{i \in \texttt{test}_\ell} (y_i - \hat{\beta}^\top x_i)^2$

Improvement #2: k-fold Cross Validation



- For each fold $\ell = 1, \ldots, k$:
  - Set $\texttt{test}_\ell$ to include all data in fold ℓ.
  - Put remaining folds in $\texttt{train}_\ell$
  - **Train** $\hat{\beta}$ by minimizing: $\texttt{RSS}_{\texttt{train}_\ell}(\beta) = \sum_{i \in \texttt{train}_\ell} (y_i - \beta^\top x_i)^2$
  - **Test** $\hat{\beta}$ by evaluating: $\texttt{RSS}_{\texttt{test}_\ell}(\hat{\beta}) = \sum_{i \in \texttt{test}_\ell} (y_i - \hat{\beta}^\top x_i)^2$

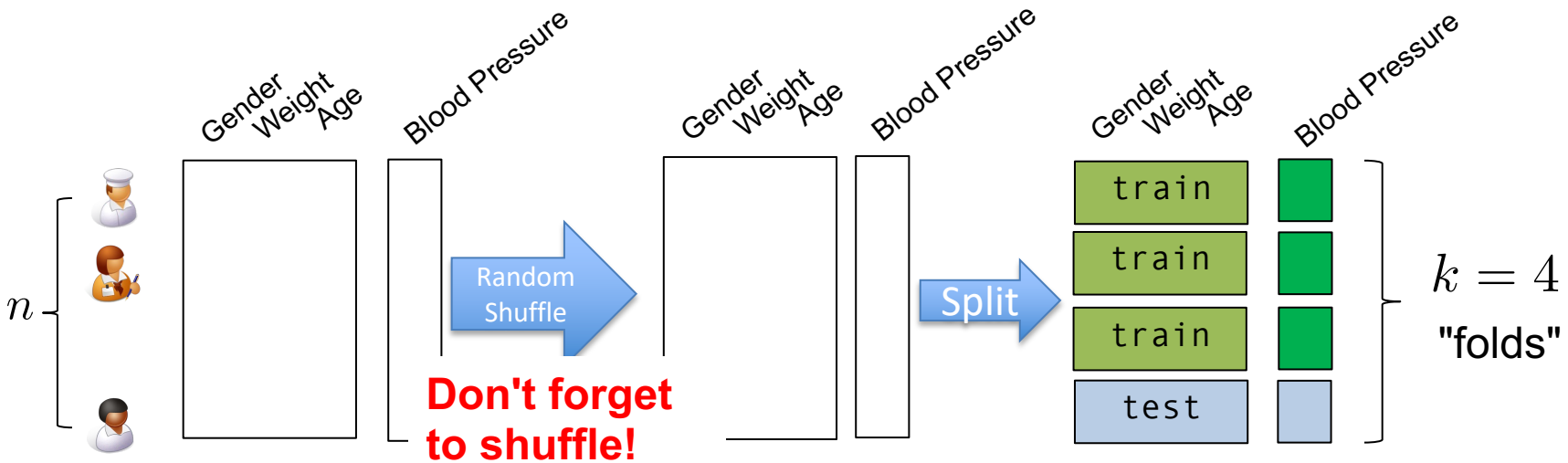
Improvement #2: k-fold Cross Validation



- For each fold $\ell = 1, \ldots, k$:
  - Set $\texttt{test}_\ell$ to include all data in fold ℓ.
  - Put remaining folds in $\texttt{train}_\ell$
  - **Train** $\hat{\beta}$ by minimizing: $\texttt{RSS}_{\texttt{train}_\ell}(\beta) = \sum_{i \in \texttt{train}_\ell} (y_i - \beta^\top x_i)^2$
  - **Test** $\hat{\beta}$ by evaluating: $\texttt{RSS}_{\texttt{test}_\ell}(\hat{\beta}) = \sum_{i \in \texttt{test}_\ell} (y_i - \hat{\beta}^\top x_i)^2$
- Quality of solution: $\overline{\texttt{RSS}} = \frac{1}{k} \sum_{\ell=1}^{k} \texttt{RSS}_{\texttt{test}_\ell}$

"Proxy" for EPE!!

k-fold Cross Validation

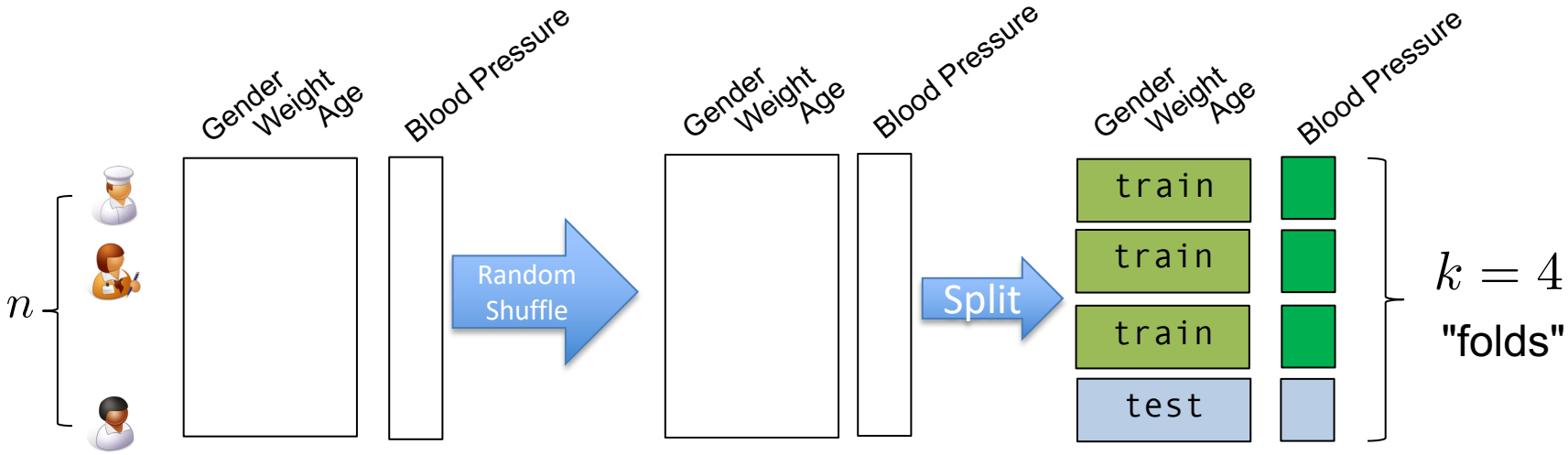


Cross-validation error:

$$\overline{\text{RSS}} = \frac{1}{k}\sum_{\ell=1}^{k} \text{RSS}_{\texttt{test}_\ell}$$

- **Less sensitive** to how split happens than train/test
- Can be applied to **other metrics** (accuracy, precision, recall, AUC)…
- Can be applied to **pick other parameters** of estimation procedure:
  - Feature selection
  - Number of iterations
  - …
- Can be used to compute **standard deviation, confidence intervals, etc.**

k-fold Cross Validation



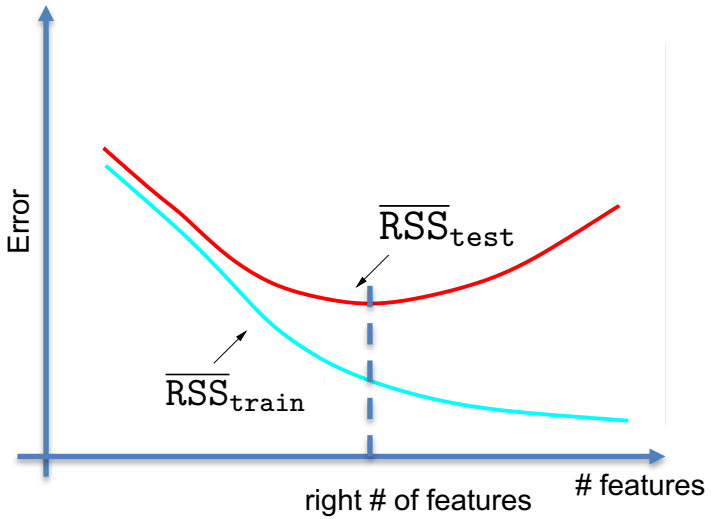
Cross-validation error:

$$\overline{\text{RSS}} = \frac{1}{k} \sum_{\ell=1}^{k} \text{RSS}_{\texttt{test}_\ell}$$

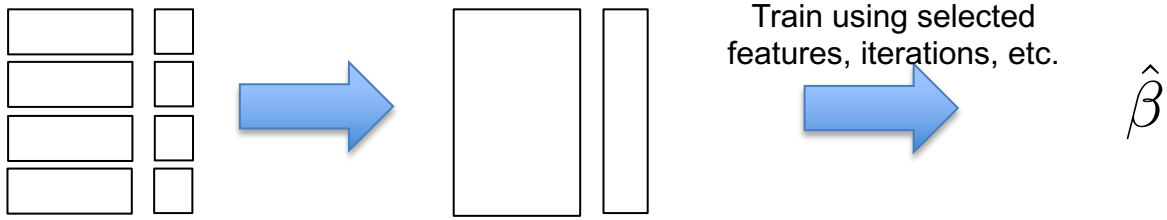
THIS IS AN EXTREMELY IMPORTANT TOPIC!!! IF YOU ONLY REMEMBER TWO THINGS FROM ENTIRE CLASS, PLEASE REMEMBER TO DOCSTRING AND CROSS-VALIDATE!!!

Finding the Right Features

- Use k-fold CV to find right problem parameters:
  - Features
  - Iterations
  - Regularization parameters (coming up)…



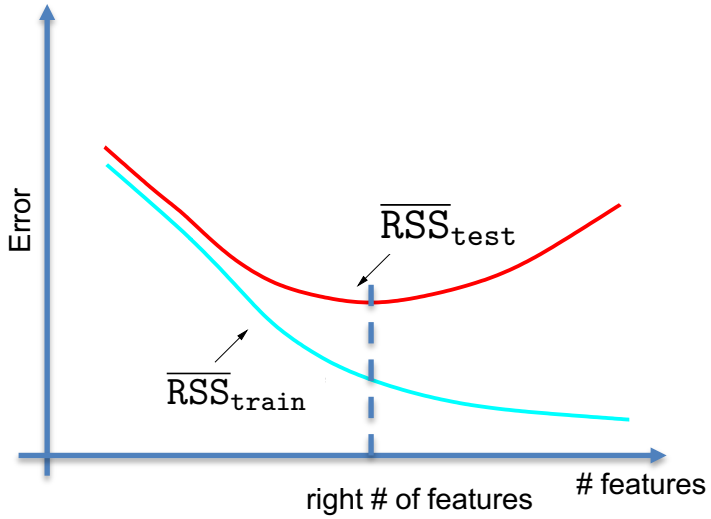
- One model $\hat{\beta}_\ell$ per fold $\ell = 1, \ldots, k$.
  - Fix these parameters and then retrain model **over entire dataset**



Feature Selection

We actually need two things:

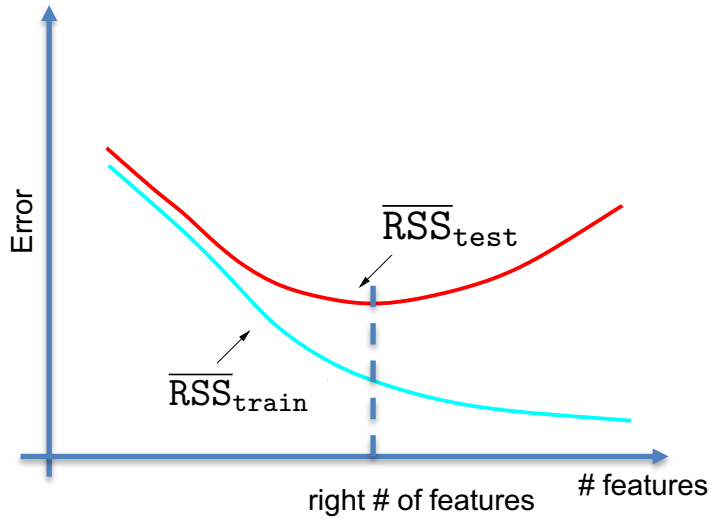
- ❑ A procedure for selecting features
- ❑ A way of measuring whether this selection is good



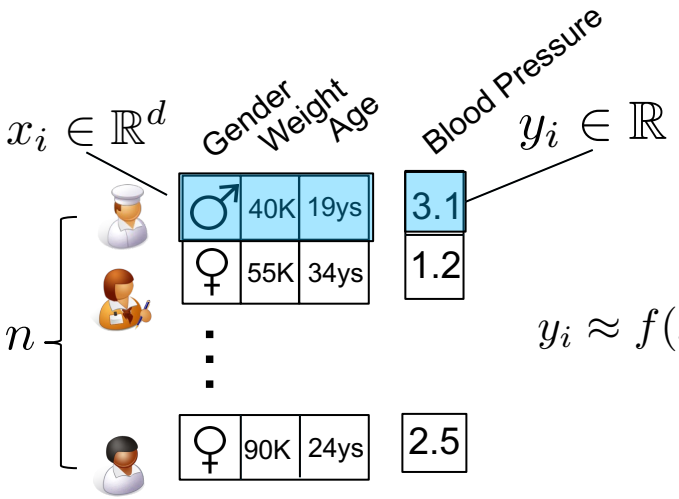
Feature Selection

We actually need two things:

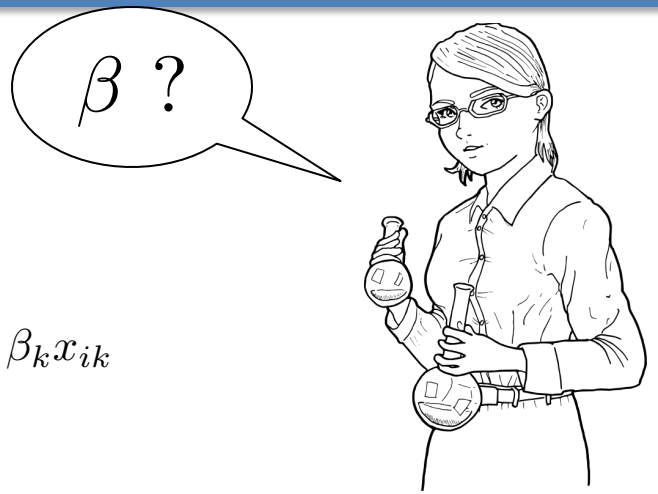
- ❑ **A procedure for selecting features**
- ❑ A way of measuring whether this selection is good



Shrinkage/Regularization Methods



$$y_i \approx f(x_i) = \beta^\top x_i = \sum_{k=1}^{d} \beta_k x_{ik}$$



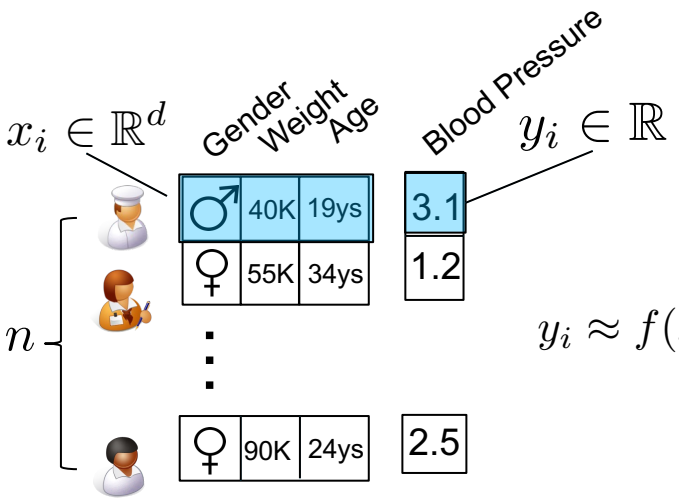
$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} (y_i - \beta^\top x_i)^2 + c(\beta)$$

RSS(β)

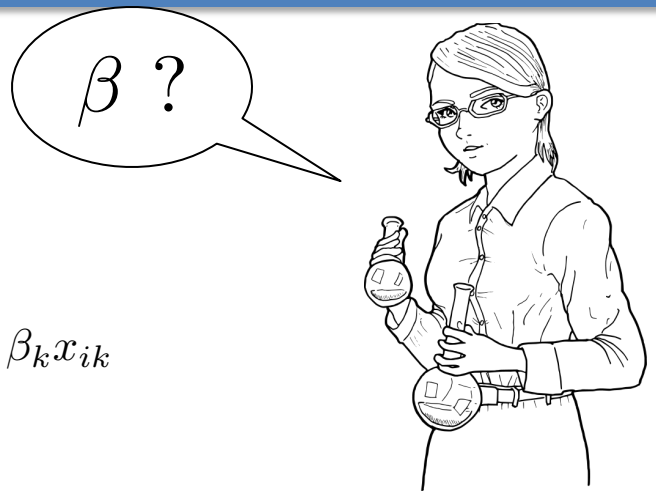
Penalty if β is "complicated"

Occam's razor: among two solutions that produce **the same RSS**, we prefer the one that has **smaller complexity**

Shrinkage/Regularization Methods



$$y_i \approx f(x_i) = \beta^\top x_i = \sum_{k=1}^{d} \beta_k x_{ik}$$

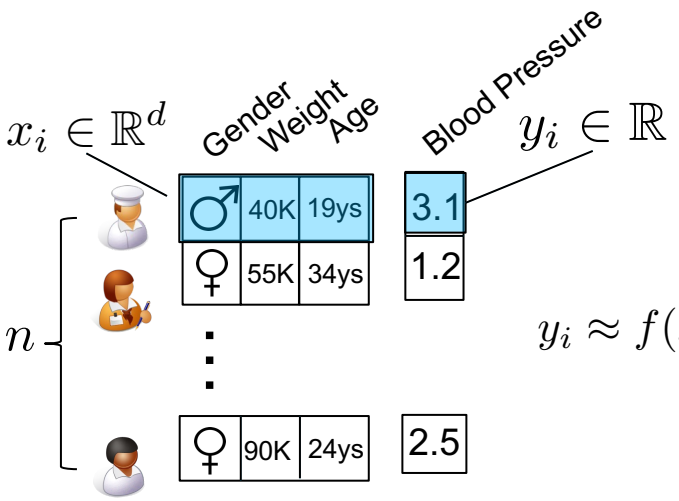


$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} (y_i - \beta^\top x_i)^2 + \lambda \|\beta\|_0 \text{ , for some } \lambda > 0$$

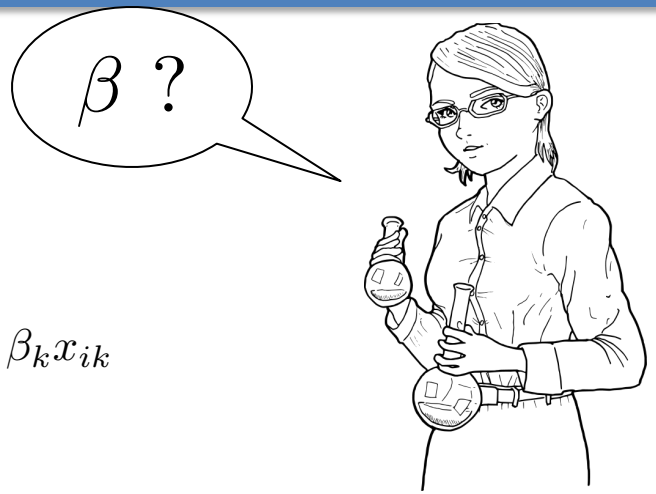
$\|\beta\|_0 =$ # of non-zero elements of β (i.e, size of β's **support**)

Occam's razor: Between two β with the same RSS, we prefer the one that is **sparser**.

Shrinkage/Regularization Methods



$$y_i \approx f(x_i) = \beta^\top x_i = \sum_{k=1}^{d} \beta_k x_{ik}$$



$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} (y_i - \beta^\top x_i)^2 + \lambda \|\beta\|_0 \text{ , for some } \lambda > 0$$

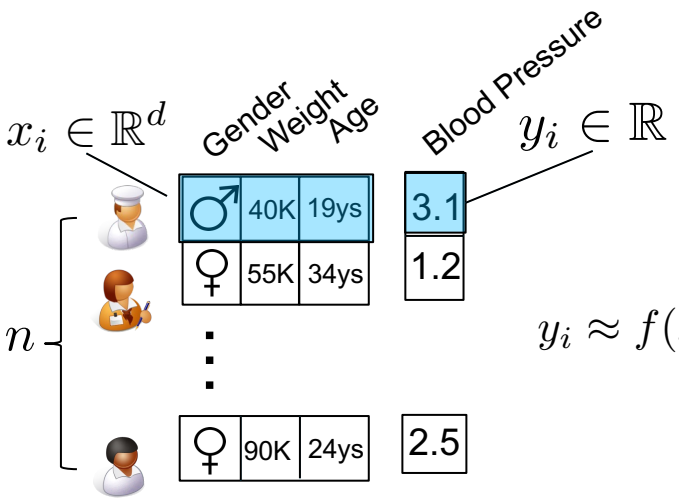
$\|\beta\|_0 =$ # of non-zero elements of β (i.e, size of β's **support**)

$\lambda \gg 0$: optimal solution has very few non-zeros

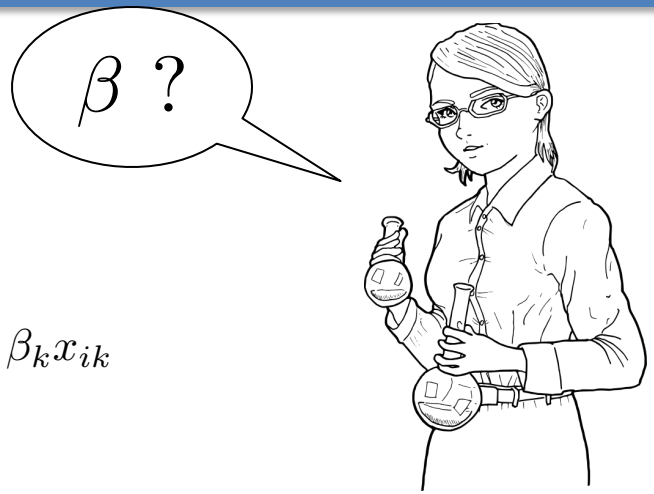
$\lambda = 0$: linear regression

Varying λ can be used for feature selection!

Shrinkage/Regularization Methods



$$y_i \approx f(x_i) = \beta^\top x_i = \sum_{k=1}^{d} \beta_k x_{ik}$$

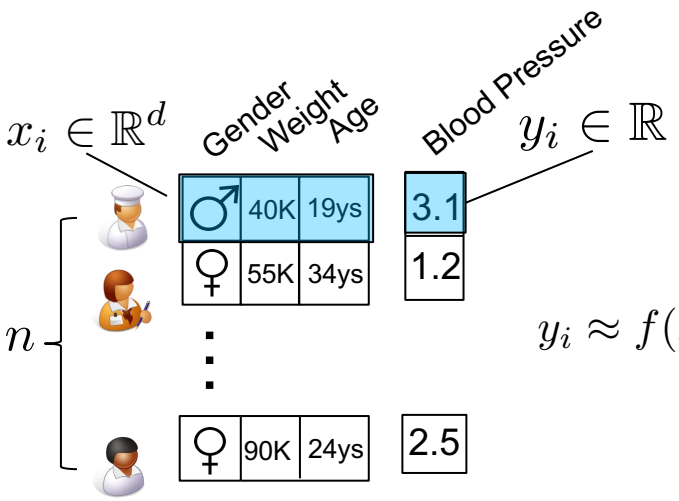


$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} (y_i - \beta^\top x_i)^2 + \lambda \|\beta\|_0 \text{ , for some } \lambda > 0$$

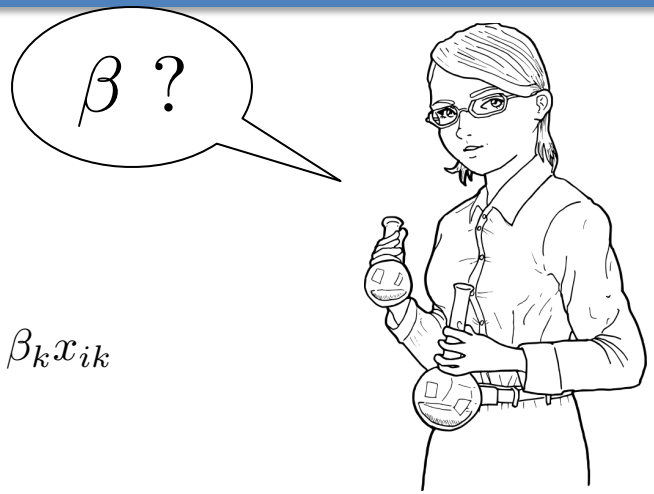
Alas, this is **not a convex objective!**

We replace it with **convex relaxations**

Ridge Regression



$$y_i \approx f(x_i) = \beta^\top x_i = \sum_{k=1}^{d} \beta_k x_{ik}$$

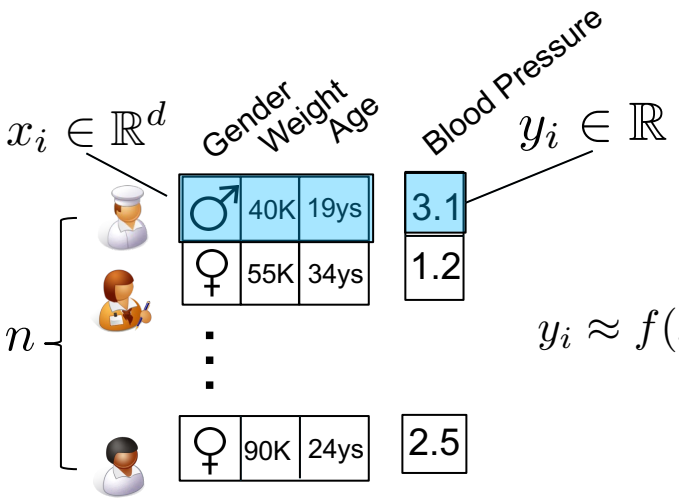


$$\hat{\beta}^{\texttt{ridge}} = \arg\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} (y_i - \beta^\top x_i)^2 + \lambda \|\beta\|_2^2, \text{ for some } \lambda > 0$$

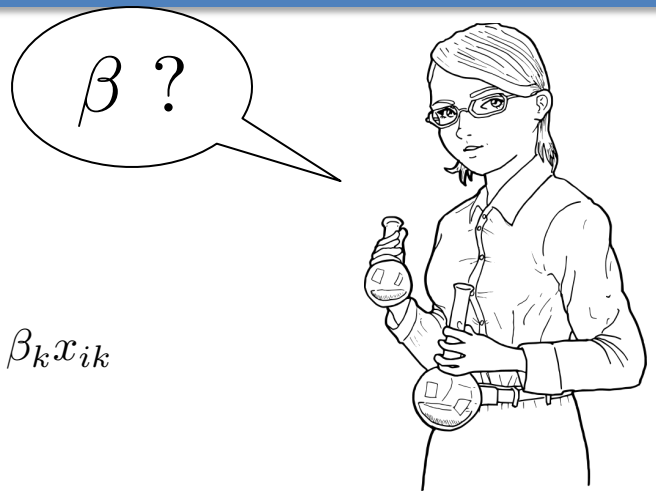
where $\|\beta\|_2^2 = \beta^\top \beta = \sum_{k=1}^{d} \beta_k^2$

Strongly Convex!!!!

Lasso Regression



$$y_i \approx f(x_i) = \beta^\top x_i = \sum_{k=1}^{d} \beta_k x_{ik}$$

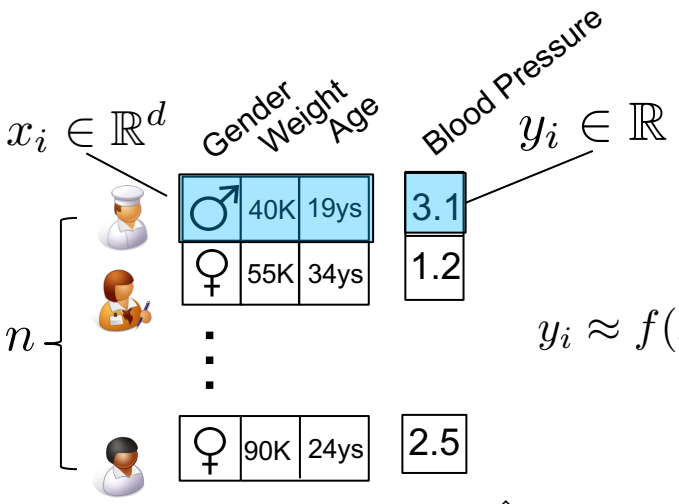


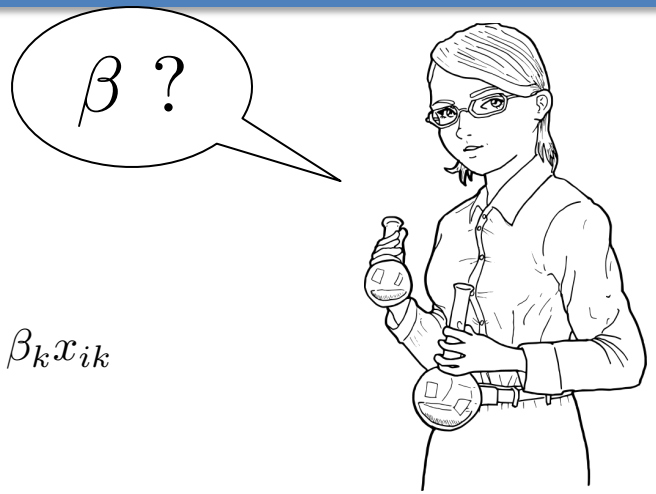
$$\hat{\beta}^{\texttt{lasso}} = \arg\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} (y_i - \beta^\top x_i)^2 + \lambda \|\beta\|_1, \quad \text{for some } \lambda > 0$$

$$\text{where } \|\beta\|_1 = \sum_{k=1}^{d} |\beta_k|$$

Convex!
(not differentiable)

Ridge Regression



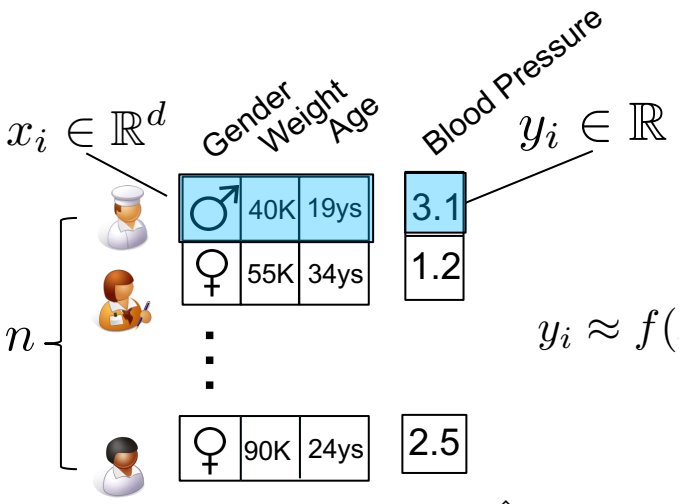


$$y_i \approx f(x_i) = \beta^\top x_i = \sum_{k=1}^{d} \beta_k x_{ik}$$

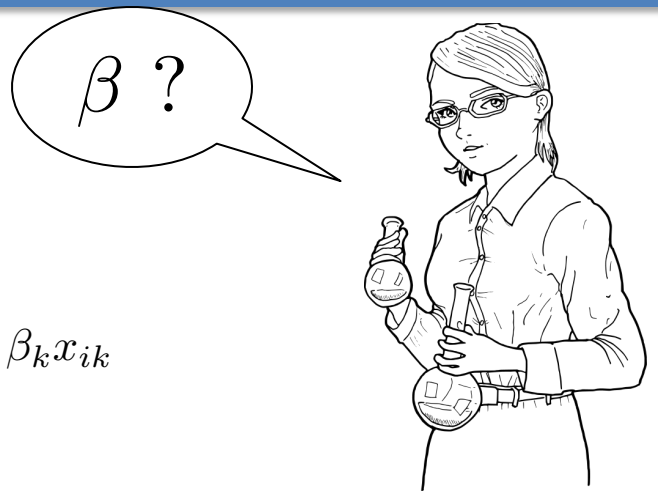
$$\hat{\beta}^{\texttt{ridge}} = \arg\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} (y_i - \beta^\top x_i)^2 + \lambda \|\beta\|_2^2 \quad , \text{ for some } \lambda > 0$$

l2-penalty,
ridge penalty,
regularization term,

Ridge Regression



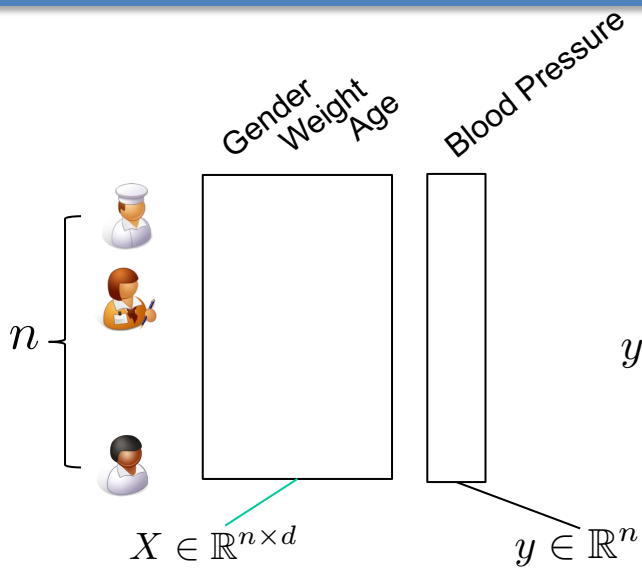
$$y_i \approx f(x_i) = \beta^\top x_i = \sum_{k=1}^{d} \beta_k x_{ik}$$

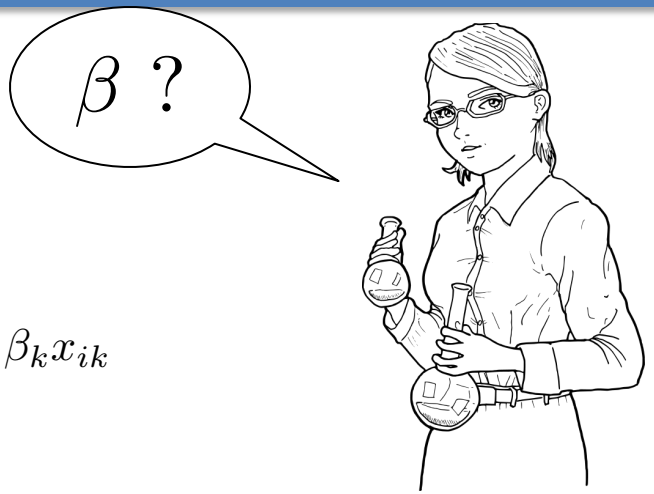


$$\hat{\beta}^{\texttt{ridge}} = \arg\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} (y_i - \beta^\top x_i)^2 + \lambda \|\beta\|_2^2 \quad , \text{ for some } \lambda > 0$$

regularization parameter

Ridge Regression





$$y_i \approx f(x_i) = \beta^\top x_i = \sum_{k=1}^{d} \beta_k x_{ik}$$

$$F(\beta) = \sum_{i=1}^{n} (y_i - \beta^\top x_i)^2 + \lambda \|\beta\|_2^2 \quad \text{, for some } \lambda > 0$$

$$= \|X\beta - y\|_2^2 + \lambda \|\beta\|_2^2$$

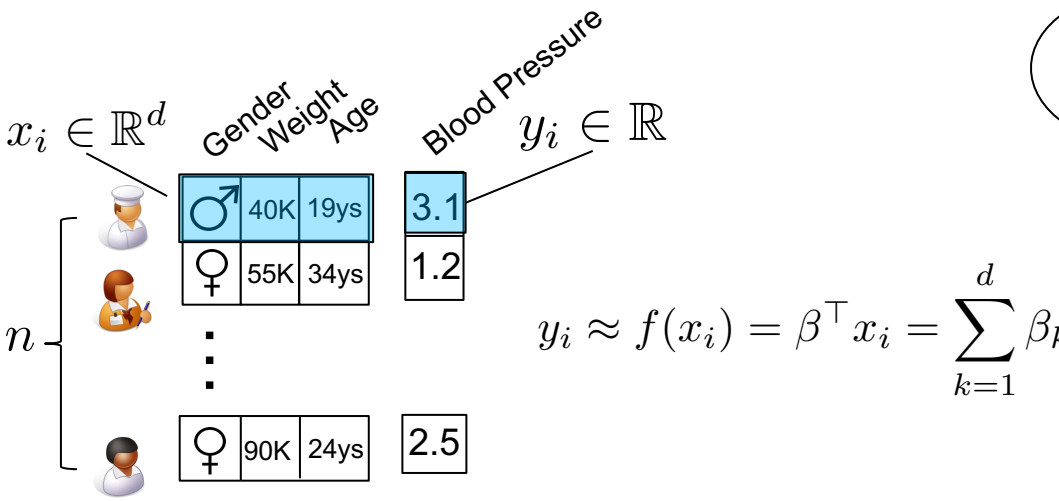
$$\nabla F(\beta) = 2X^\top X\beta - 2X^\top y + 2\lambda\beta$$

$$\nabla^2 F(\beta) = 2(X^\top X + \lambda I) \succ 0$$

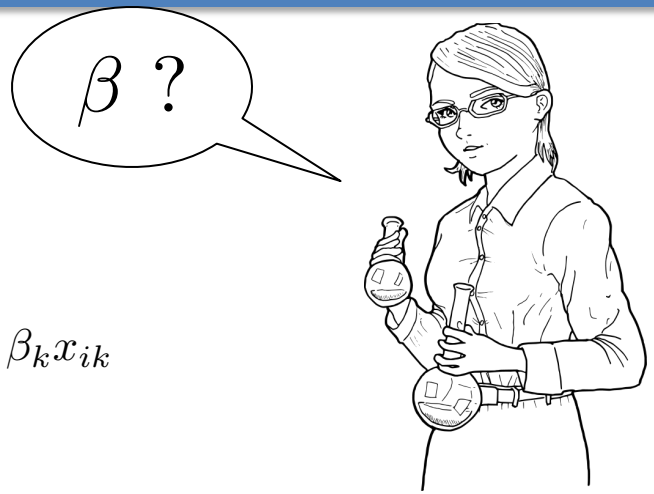
$$\nabla F(\hat{\beta}) = 0 \Leftrightarrow \hat{\beta} = \underbrace{(X^\top X + \lambda I)}^{-1} X^\top y$$

Invertible for $\lambda > 0$!

Ridge Regression: Intuition



$$y_i \approx f(x_i) = \beta^\top x_i = \sum_{k=1}^{d} \beta_k x_{ik}$$



For every $\lambda \geq 0$, there exists a $t \geq 0$ such that the two problems produce **the same solution**

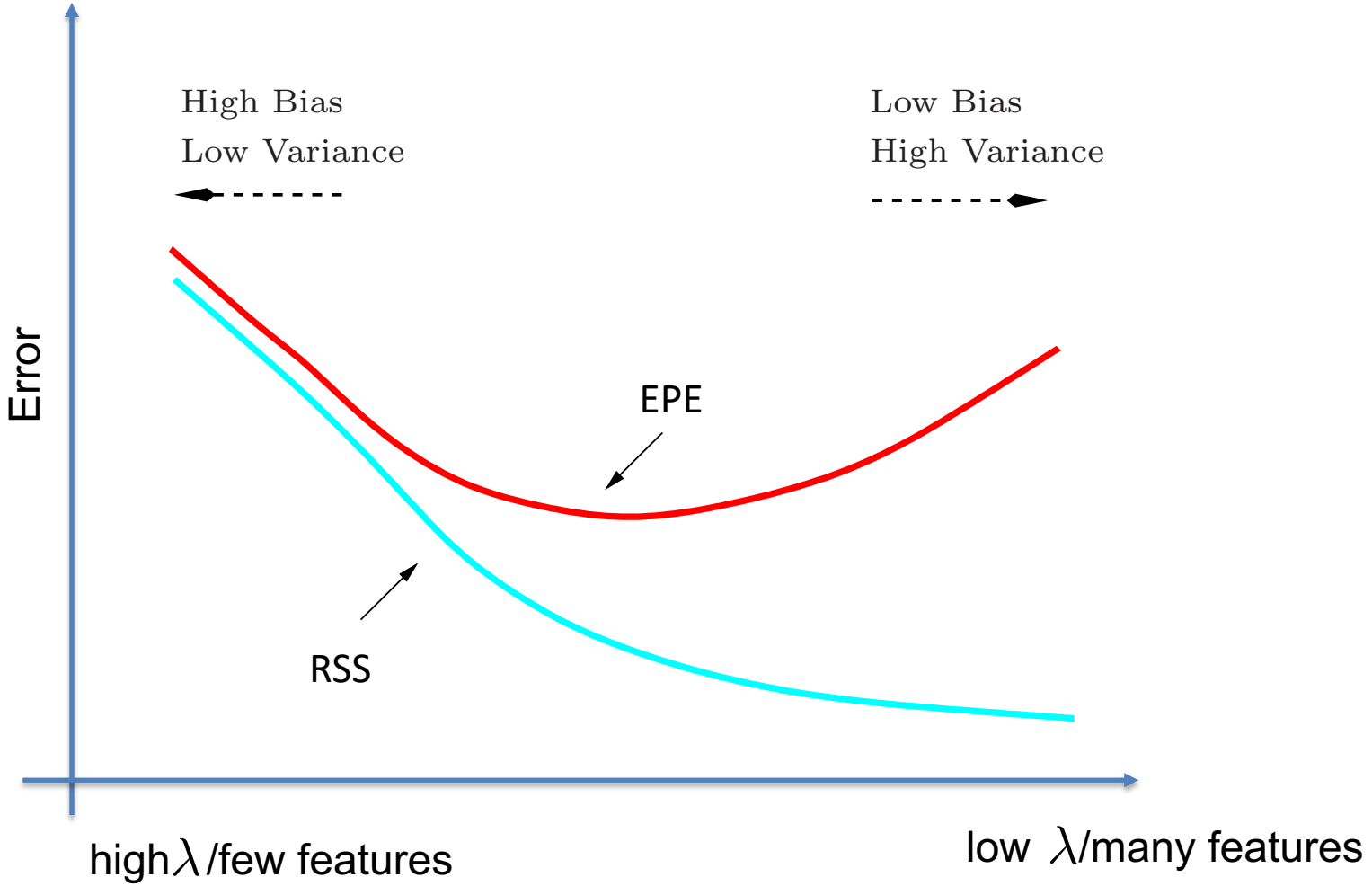
Minimize: $\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 + \lambda\|\beta\|_2^2$

subject to: $\beta \in \mathbb{R}^d$

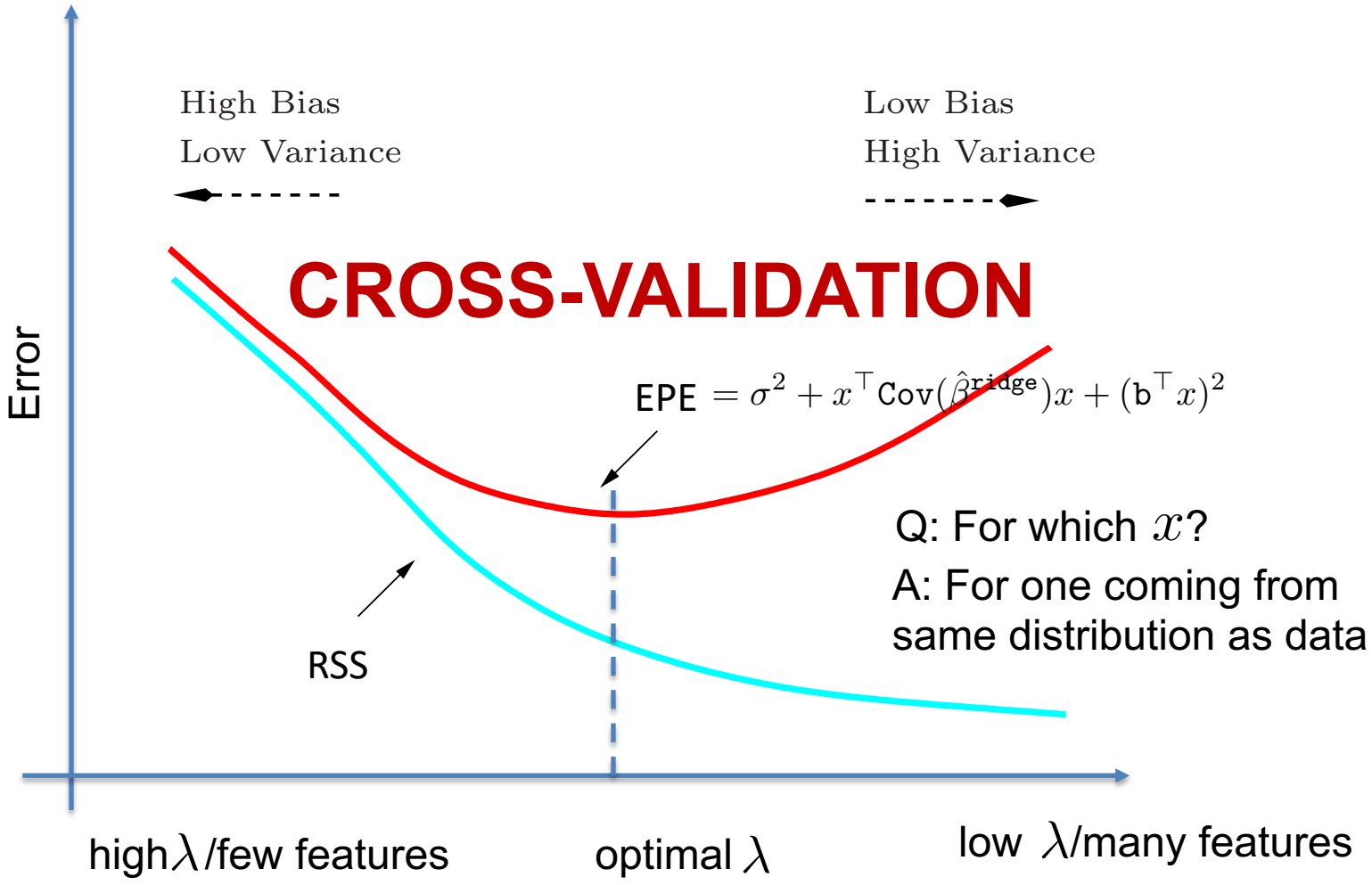
Minimize: $\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2$

subject to: $\|\beta\|_2^2 \leq t$

Ridge Regression: Intuition

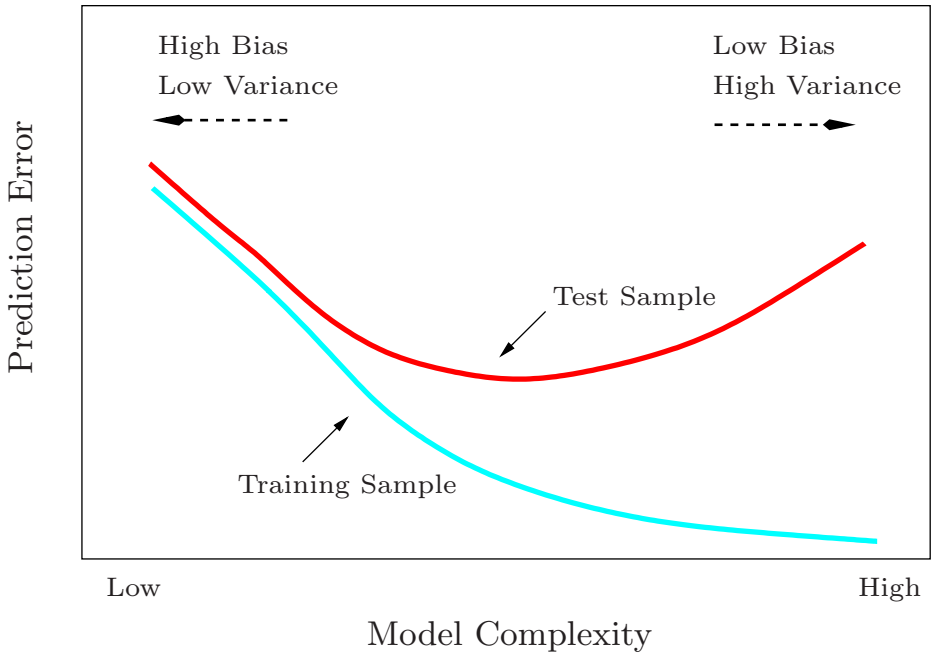


Ridge Regression: Intuition

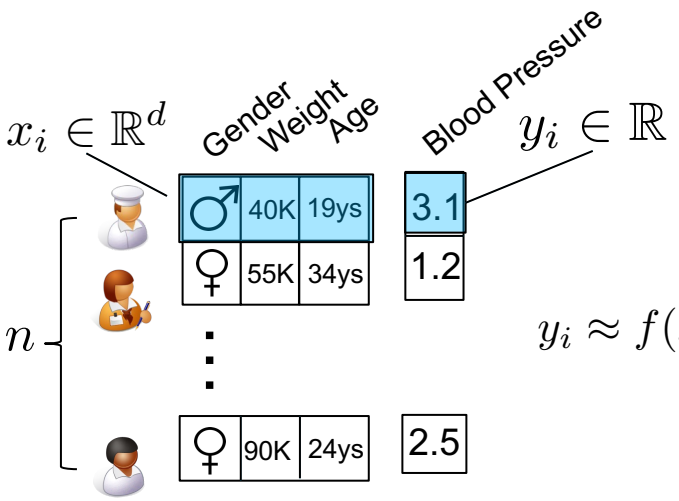


Intuition Has a Universal, General Interpretation

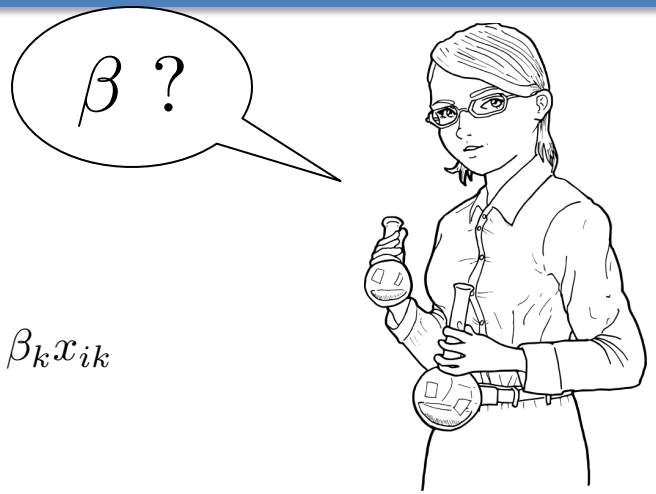
- Cross-Validation minimizes EPE, assuming new data comes from **same distribution** as existing data
- When varying **model complexity,** we are establishing a tradeoff between **variance** and **bias**



Lasso Regression



$$y_i \approx f(x_i) = \beta^\top x_i = \sum_{k=1}^{d} \beta_k x_{ik}$$

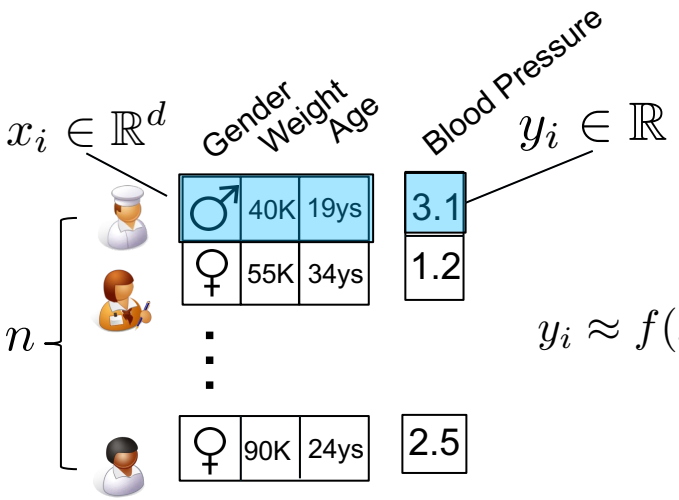


$$\hat{\beta}^{\texttt{lasso}} = \arg\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} (y_i - \beta^\top x_i)^2 + \lambda \|\beta\|_1, \quad \text{for some } \lambda > 0$$

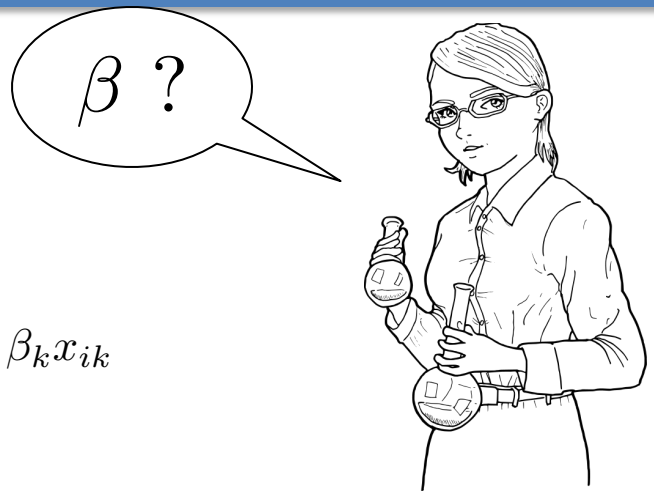
$$\text{where } \|\beta\|_1 = \sum_{k=1}^{d} |\beta_k|$$

Convex!
(not differentiable)

Lasso Regression



$$y_i \approx f(x_i) = \beta^\top x_i = \sum_{k=1}^{d} \beta_k x_{ik}$$



For every $\lambda \geq 0$, there exists a $t \geq 0$ such that the two problems produce **the same solution**

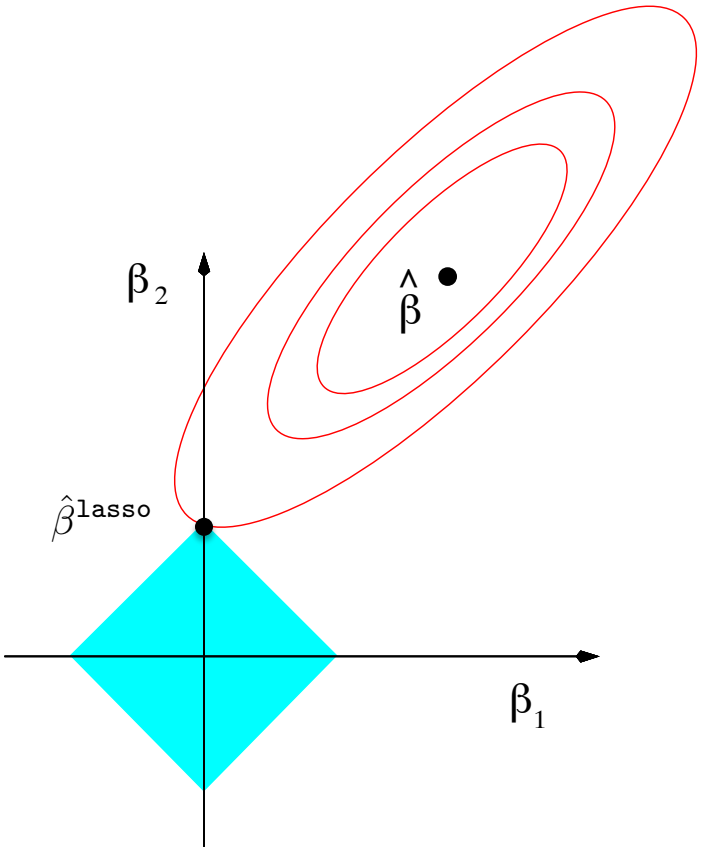
Minimize: $\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 + \lambda\|\beta\|_2^2$

subject to: $\beta \in \mathbb{R}^d$

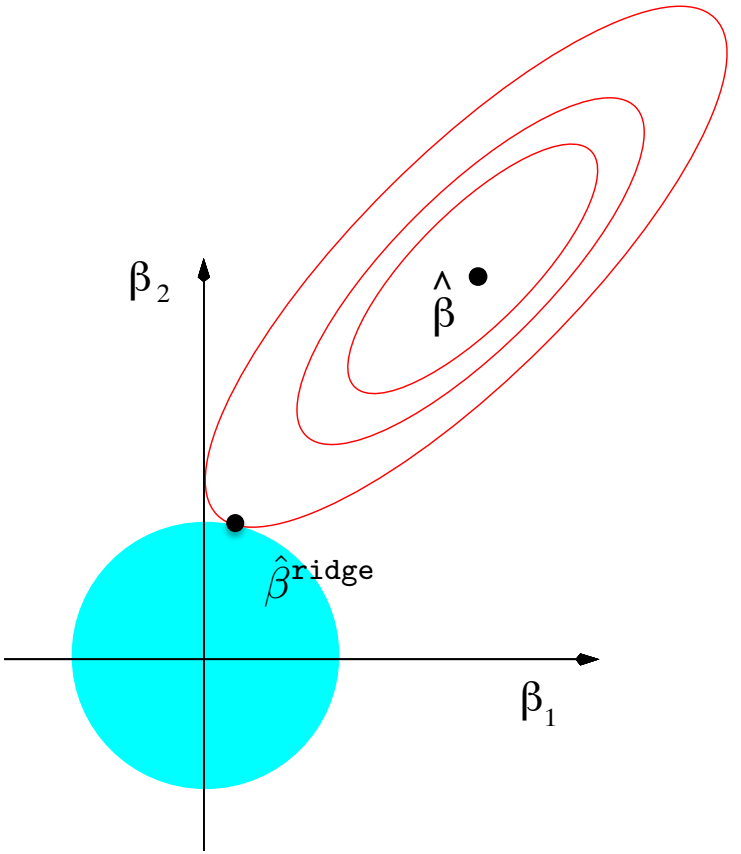
Minimize: $\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2$

subject to: $\|\beta\|_1 \leq t$

Lasso Regression vs. Ridge Regression



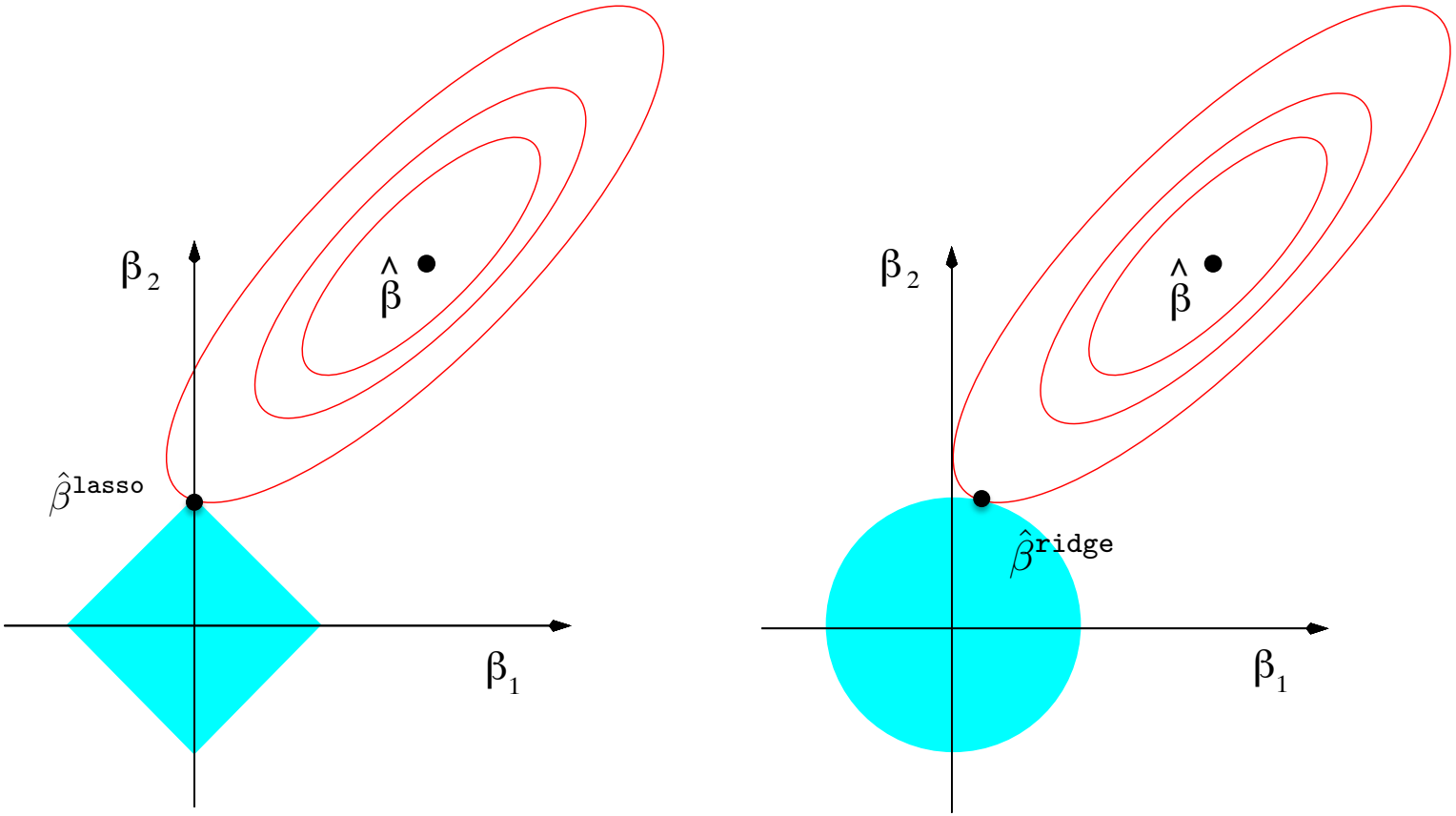
$arg(min_{\beta i}(MSE))$ *given that* $\sum |\beta_i| < t$



$arg(min_{\beta i}(MSE))$ *given that* $\sum \beta_i^2 < t$

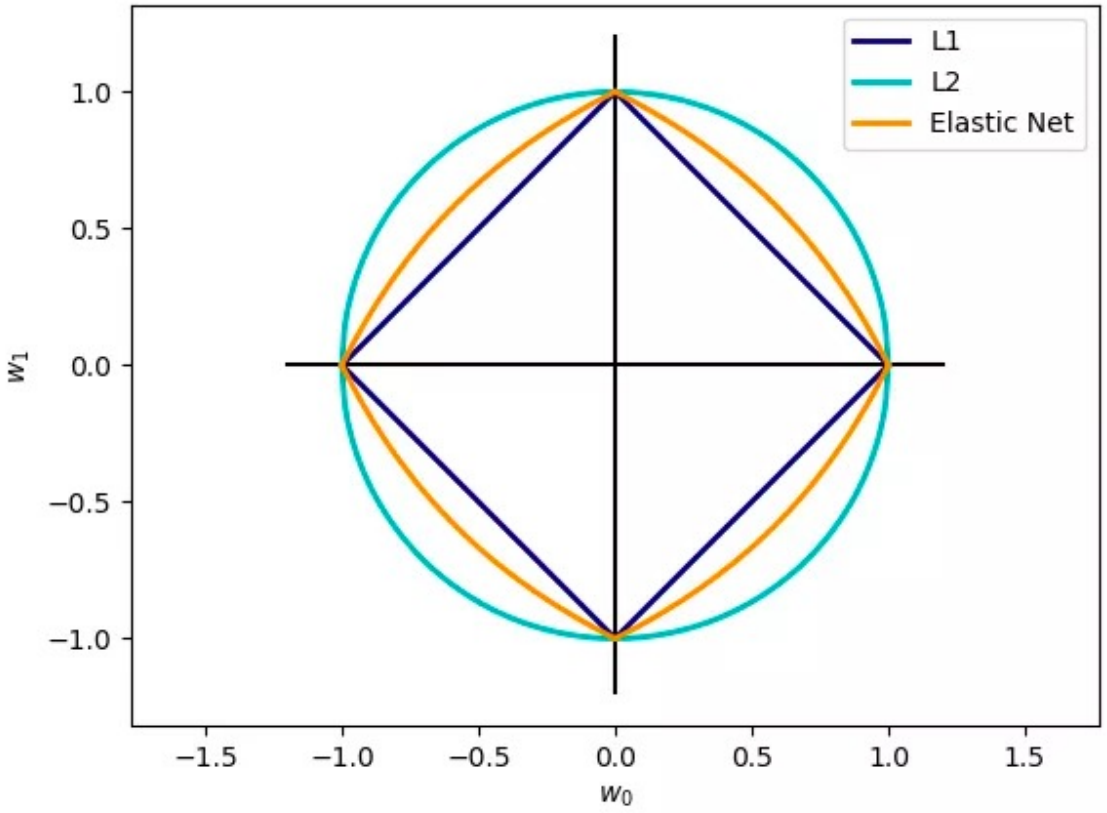
Lasso is more prone to sparse solutions

Lasso Regression vs. Ridge Regression



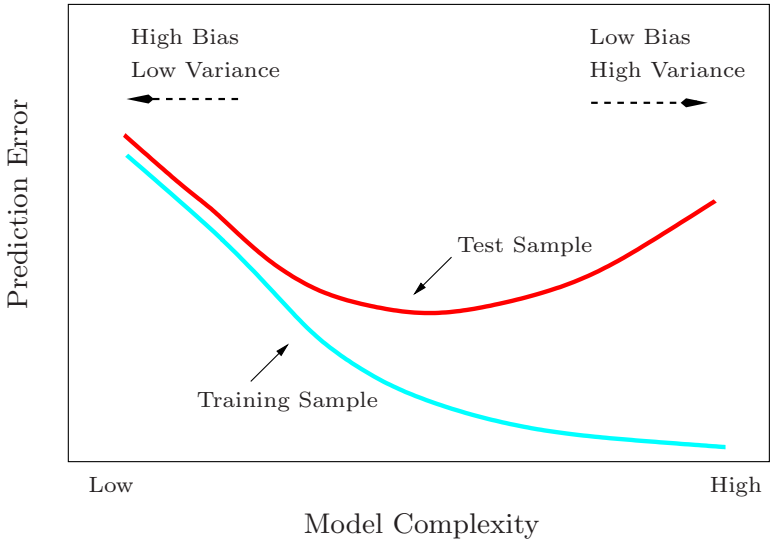
Lasso vs. Ridge vs. Elastic Regression

The Lasso, Ridge and Elastic-net regression can also be viewed as a constraint added to the optimization process.



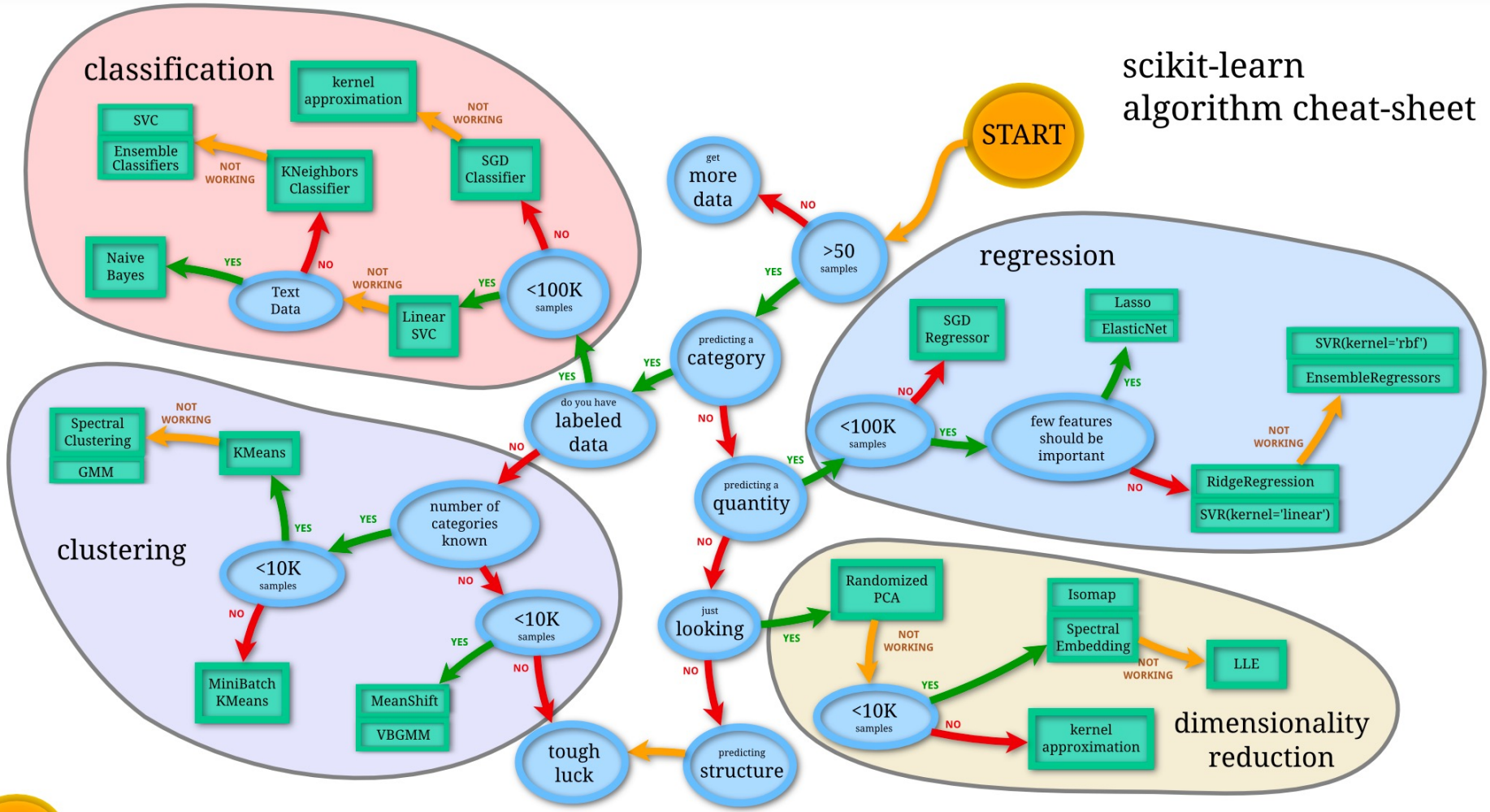
Conclusions

- Learning f ⇒ learning right **features**
- Fitting is not enough! **Cross Validation**



- Model complexity ⇔ Bias-Variance Tradeoff

SKLEARN


scikit-learn
algorithm cheat-sheet
START
classification
kernel approximation
NOT WORKING
SVC
Ensemble Classifiers
NOT WORKING
KNeighbors Classifier
SGD Classifier
NO
Naive Bayes
YES
NO
NOT WORKING
YES
Text Data
Linear SVC
<100K samples
get more data
NO
>50 samples
YES
regression
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
predicting a category
YES
NO
<100K samples
NO
YES
few features should be important
YES
NOT WORKING
RidgeRegression
NO
SVR(kernel='linear')
do you have labeled data
YES
NO
predicting a quantity
YES
clustering
Spectral Clustering
NOT WORKING
GMM
KMeans
number of categories known
YES
NO
<10K samples
YES
NO
<10K samples
NO
MiniBatch KMeans
YES
MeanShift
VBGMM
NO
just looking
YES
NO
Randomized PCA
NOT WORKING
Isomap
Spectral Embedding
NOT WORKING
LLE
YES
<10K samples
NO
kernel approximation
dimensionality reduction
tough luck
predicting structure
Back
scikit learn