

# Data Science

Lecture 2-2: Data Science Fundamentals (Modeling)



Lecturer: Yen-Chia Hsu

Date: Feb 2024

This lecture recaps **classification** and **regression** techniques for modeling data.

Classification

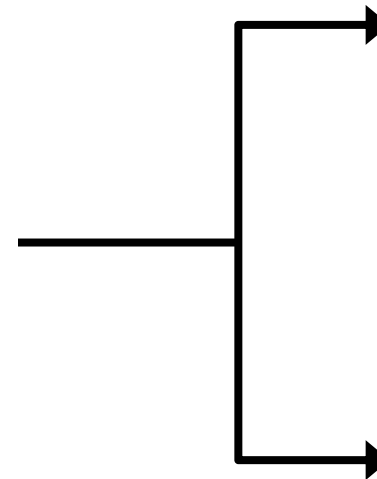
For this lecture, let us now use the following text classification task as an example: identifying **whether a text message is spam or ham (non-spam)**.

◆◆◆◆◆◆◆◆◆◆ PRIVATE!  
Your 2020 Account won \$1,000,000  
lottery! ◆◆◆◆◆◆◆◆◆◆ To claim  
call 08719180248 ◆◆◆◆◆◆◆◆◆◆

Hi Yen-Chia, may we have our  
meeting on 5/15 by just email  
update to buy some time? if not,  
zero worries if you need to talk.



Mail



Spam



Ham  
(Non-spam)

Classification

To classify spam messages, we need examples: a **dataset** with **observations** (messages) and **labels** (spam or non-spam).

◆◆◆◆◆◆◆◆◆◆ PRIVATE! Your 2020 Account won \$1,000,000  
lottery! ◆◆◆◆◆◆◆◆◆◆ To claim call 08719180248 ◆◆◆◆◆◆◆◆◆◆



Spam

Hi Yen-Chia, may we have our meeting on 5/15 by just email update to  
buy some time? if not, zero worries if you need to talk.



Ham

Would you be willing to meet with me on 3/26 Thursday when I was in  
TU Delft after (or before) giving the guest lecture (10:35am-11:50am)?



Ham



Observations



Labels

## Classification

We can extract **features (information)** using human knowledge, which can help distinguish spam and ham messages.

◆◆◆◆◆◆◆◆◆◆ PRIVATE! Your 2020 Account won \$1,000,000  
lottery! ◆◆◆◆◆◆◆◆◆◆ To claim call 08719180248 ◆◆◆◆◆◆◆◆◆◆



Spam

Number of special characters = 34

Number of digits = 22

Hi Yen-Chia, may we have our meeting on 5/15 by just email update to  
buy some time? if not, zero worries if you need to talk.



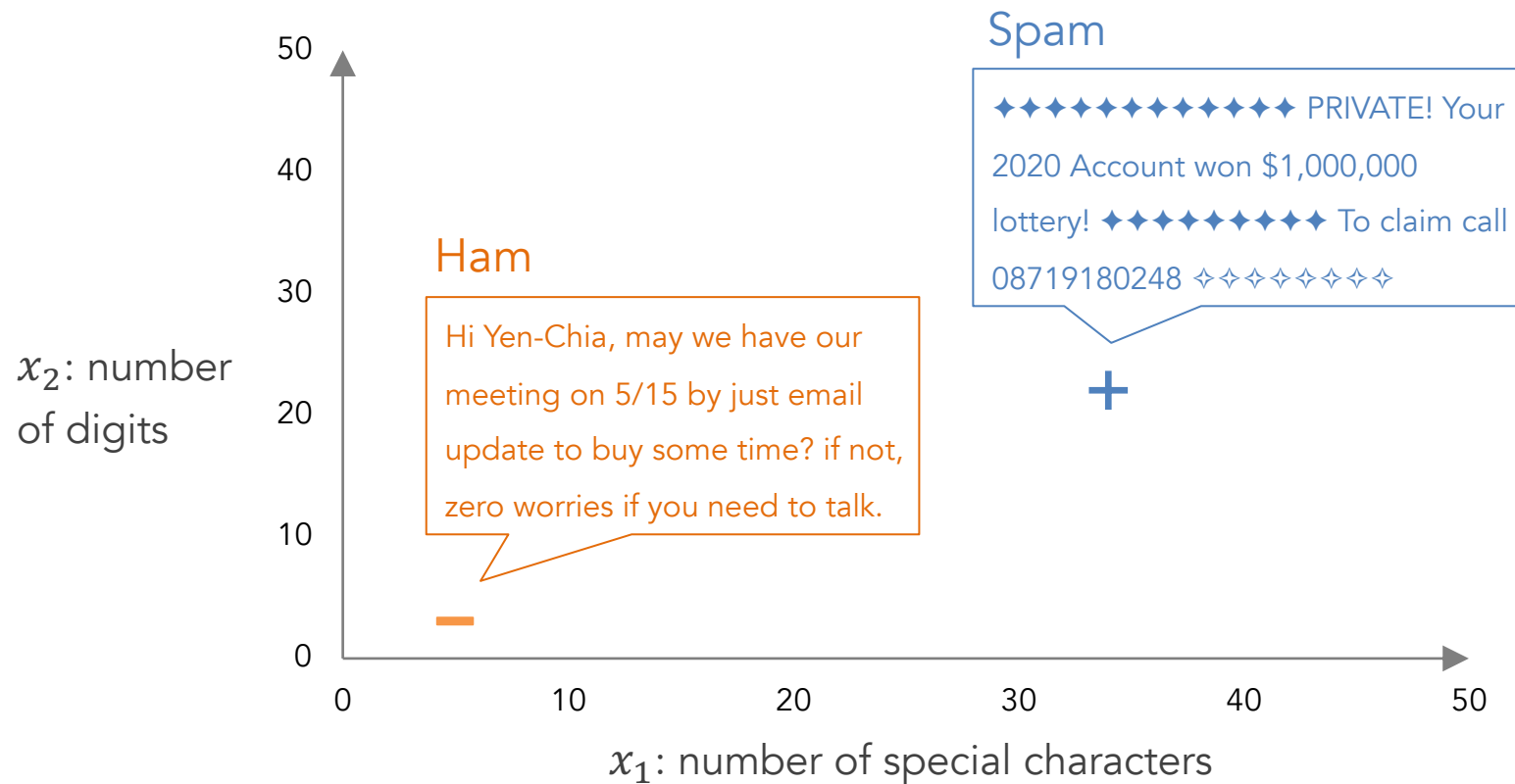
Ham

Number of special characters = 5

Number of digits = 3

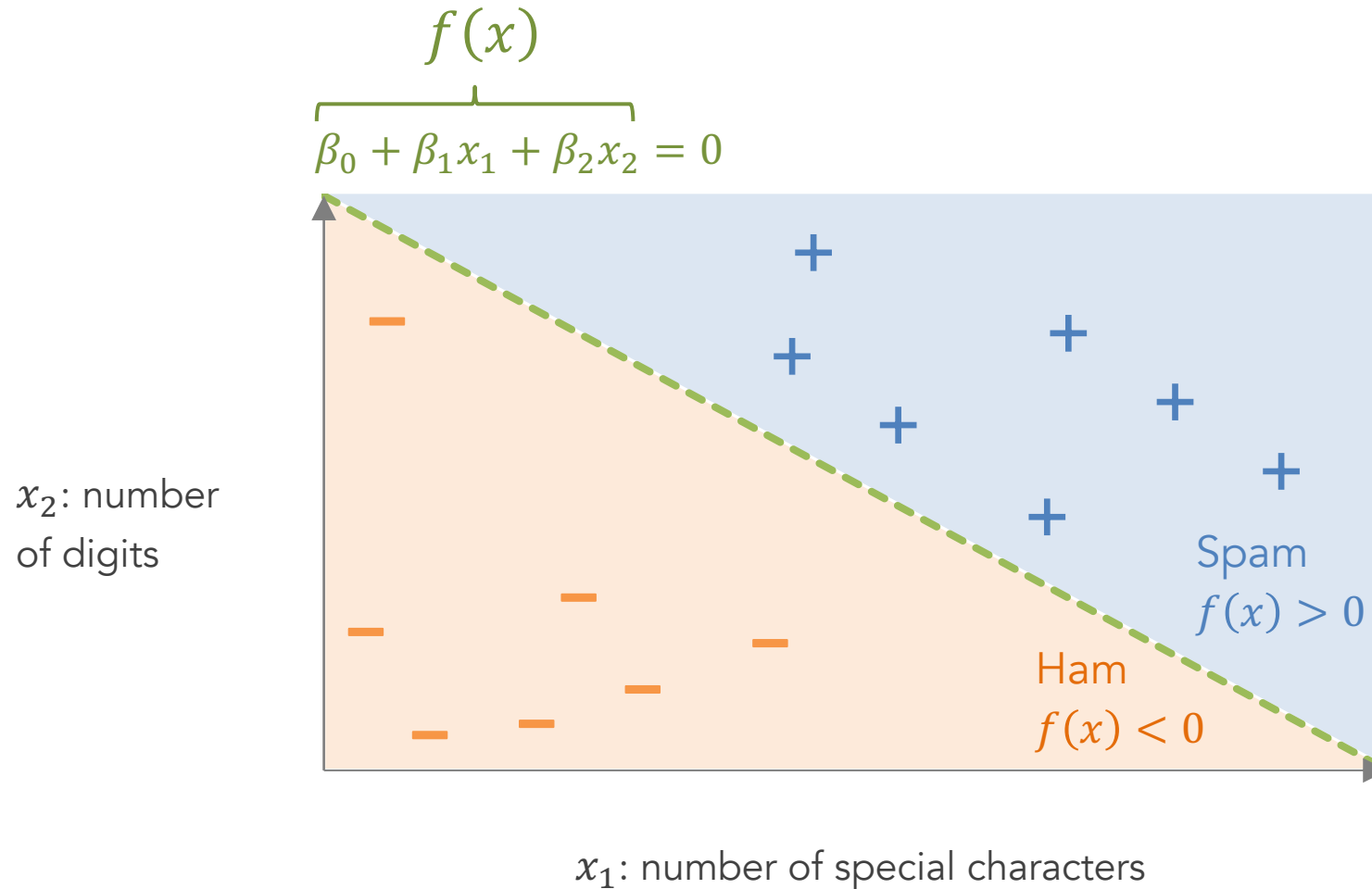
## Classification

Using **features**  $x$  (which contains  $x_1$  and  $x_2$ ), we can represent each message **as one data point** on an  $p$ -dimensional space ( $p = 2$  in this case).



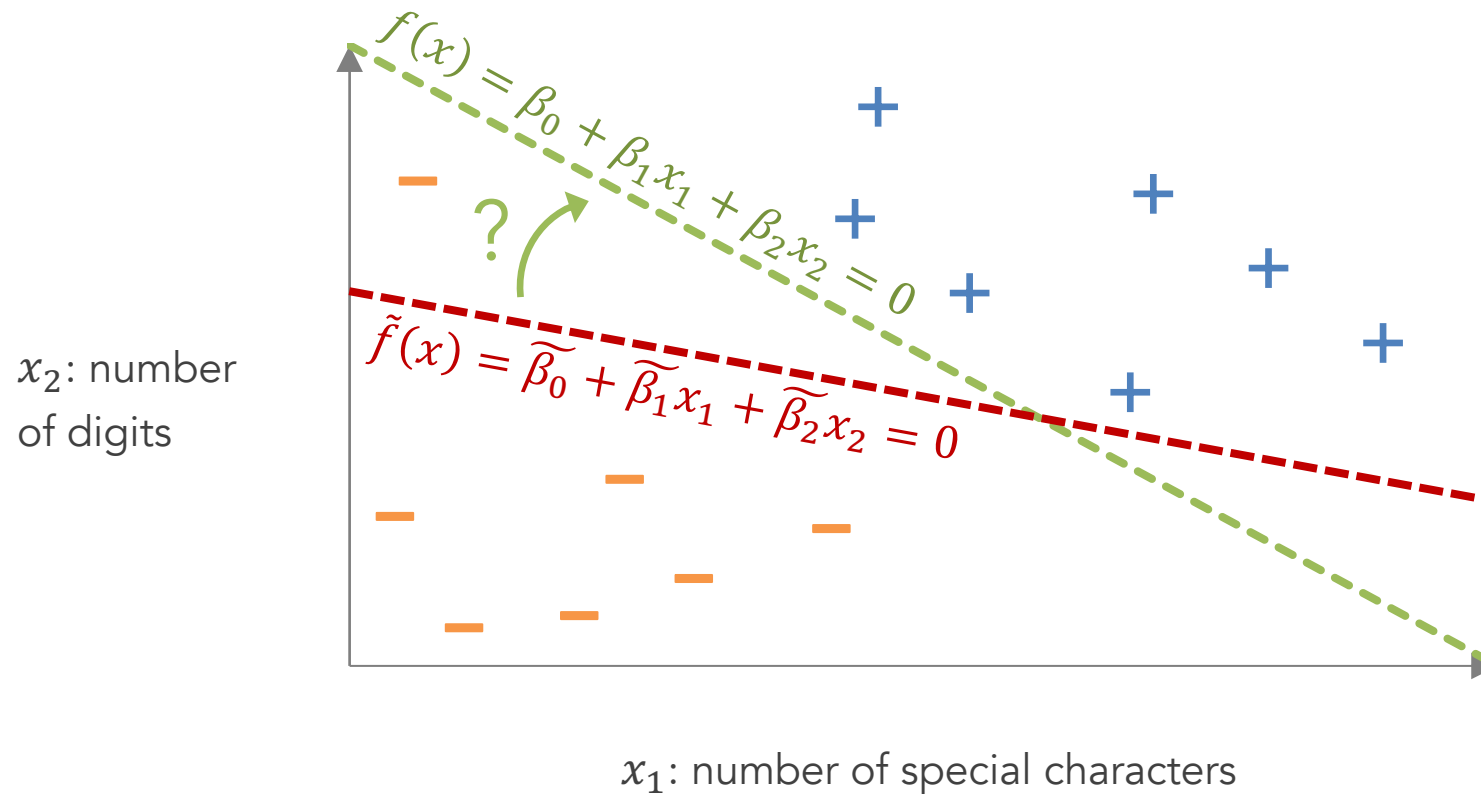
Classification

We can think of the model as a function  $f$  that can separate the observations into groups (i.e., class labels  $y$ ) according to their features  $x = \{x_1, x_2\}$ .



## Classification

To find a good function  $f$ , we start from some  $f$  and **train it** until satisfied. We need something to tell us **which direction and magnitude** to update.

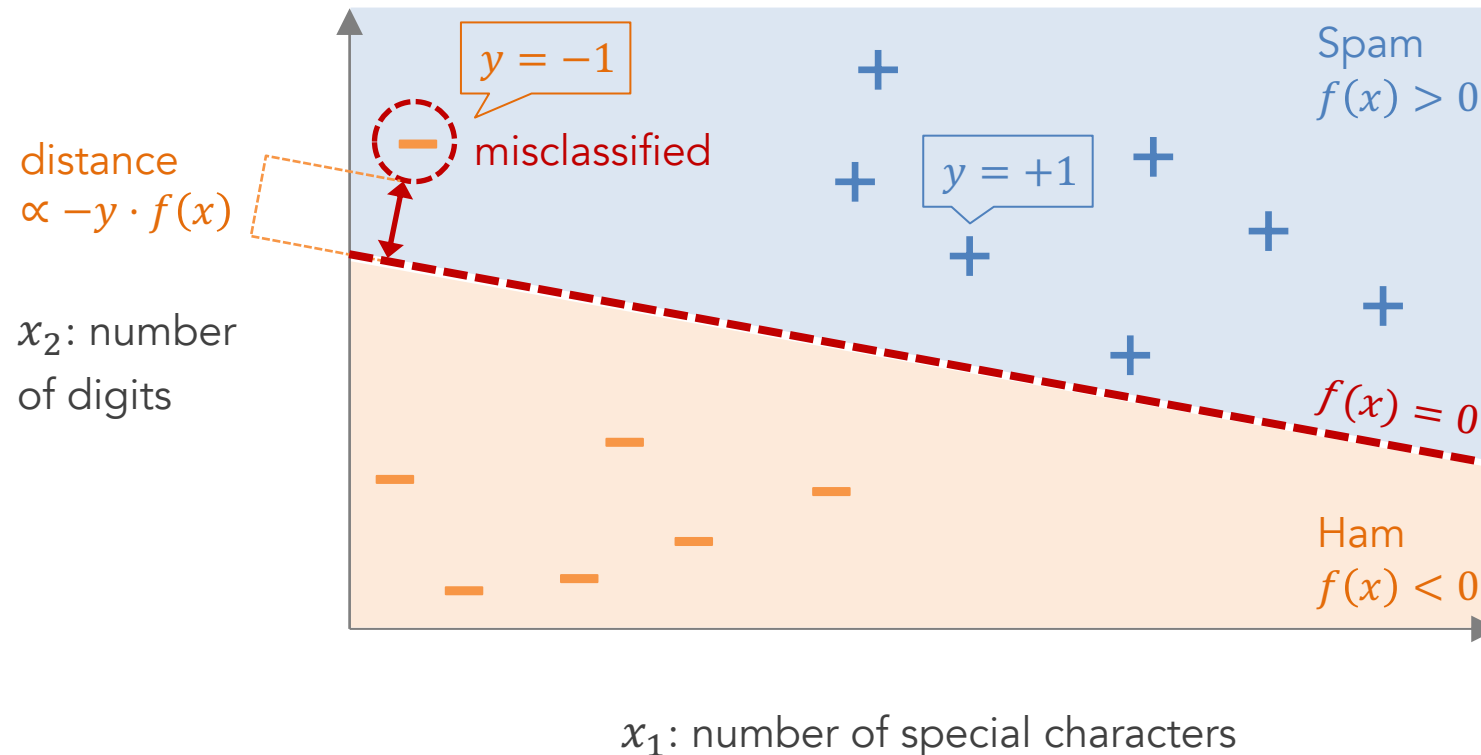




## Classification

First, we need an error metric (i.e., cost or objective function). For example, we can use the **sum of distances** between the misclassified points and line  $f$ .

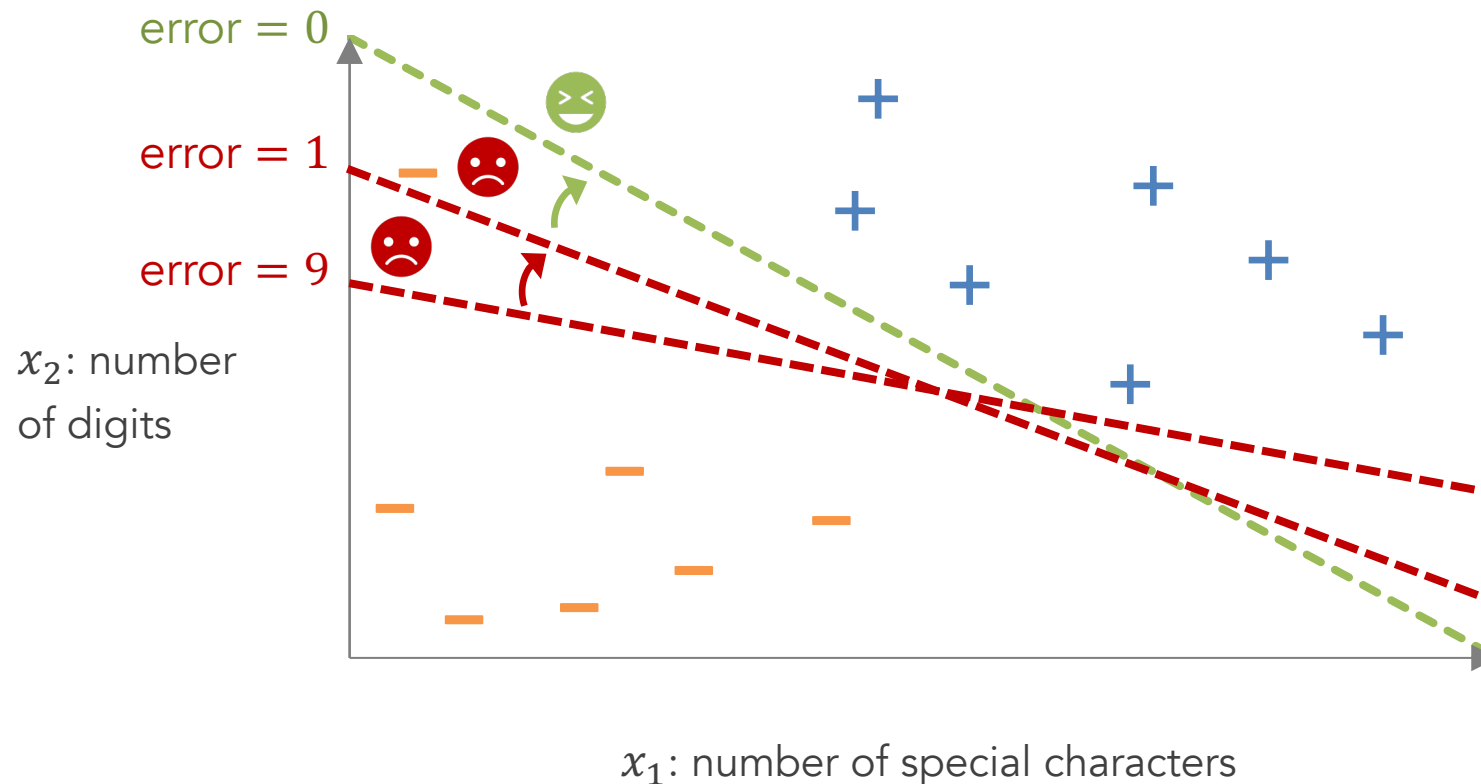
$$\text{error} = \sum -y \cdot f(x) \quad \text{for each misclassified point } x = \{x_1, x_2\}$$



## Classification

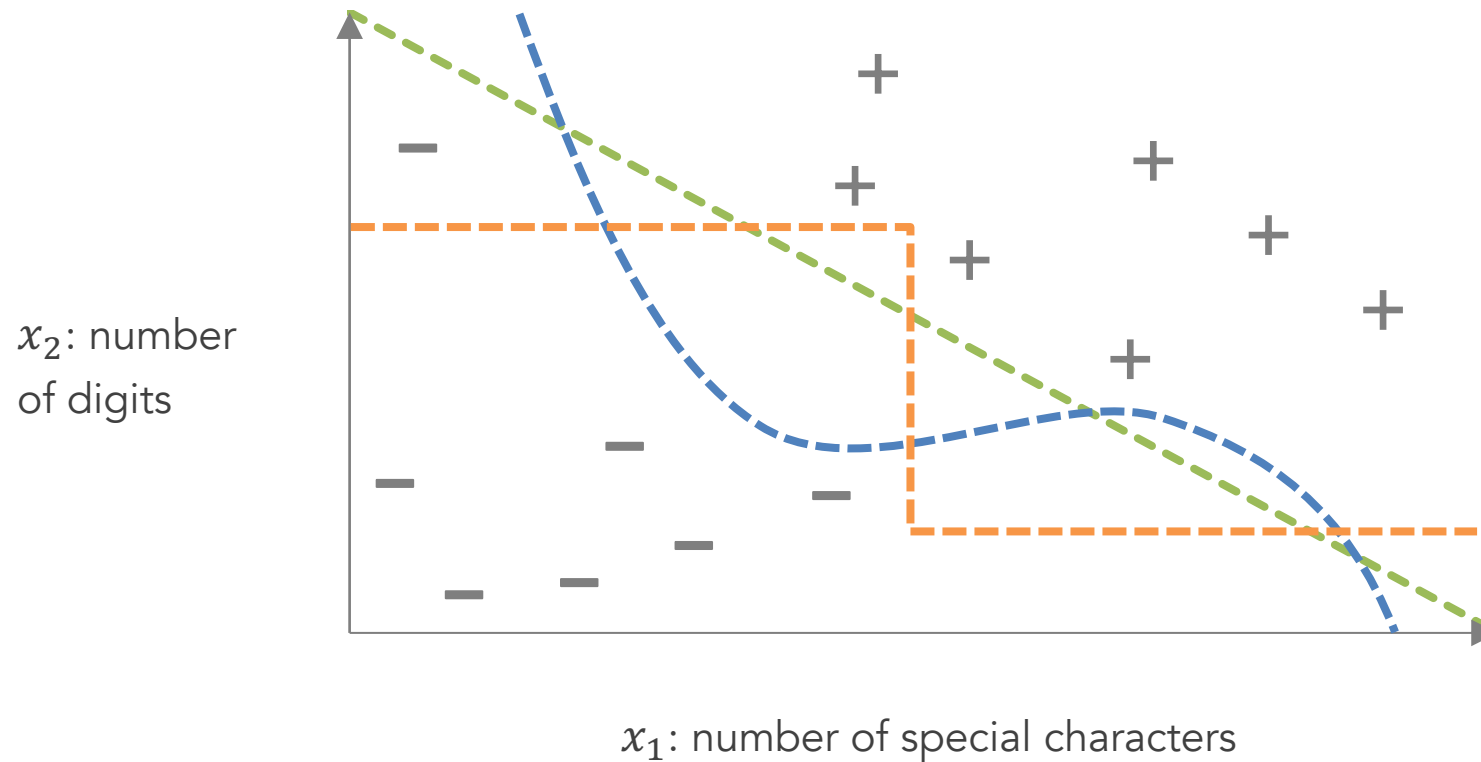
We can use gradient descent (an optimization algorithm) to **minimize the error** to train the model  $f$  iteratively. This example is the Perceptron algorithm.

$$\text{minimize error} = \sum -y \cdot f(x) \text{ for each misclassified point } x = \{x_1, x_2\}$$



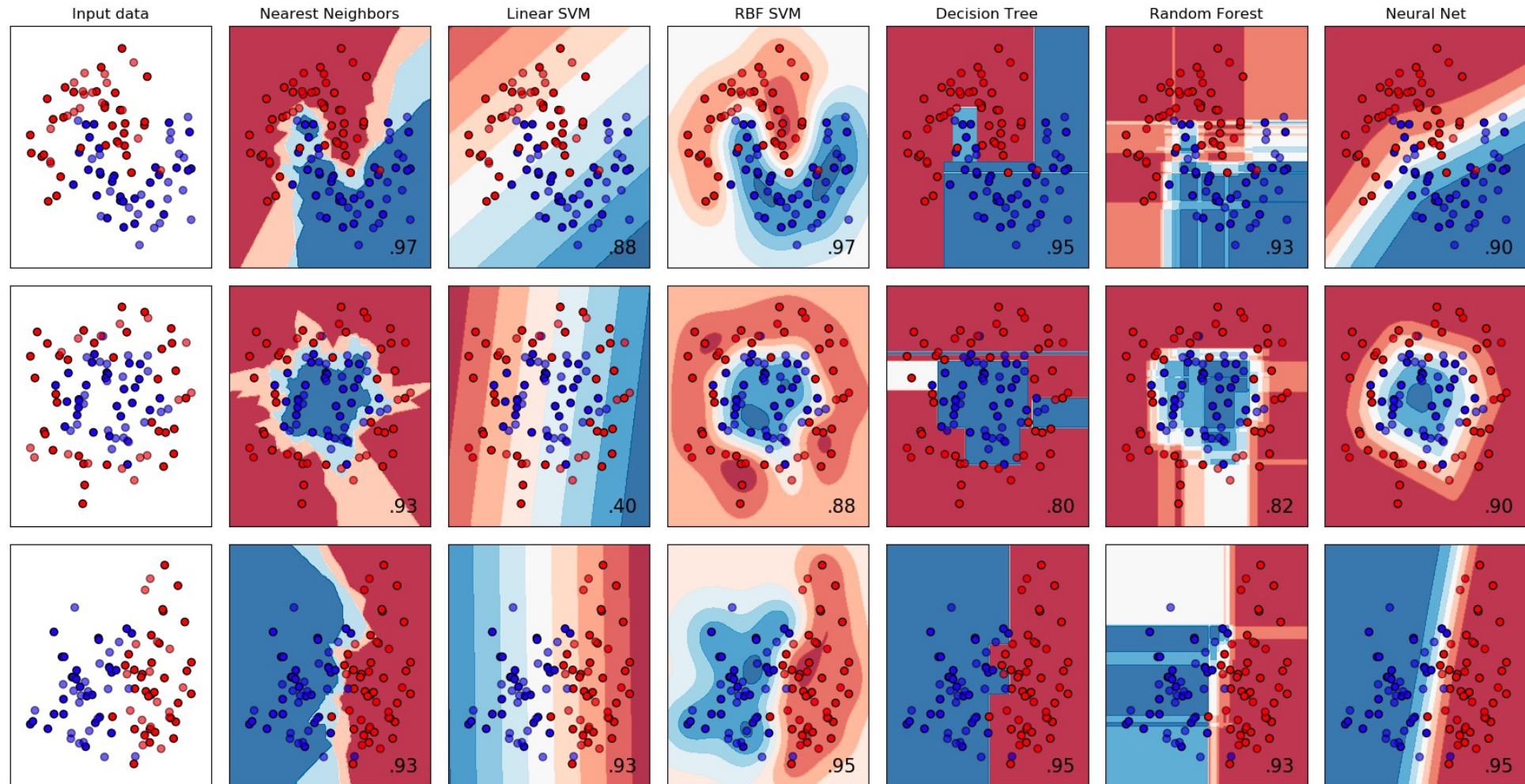
## Classification

Depending on the needs, we can train **different models** (using different loss functions) with various shapes of decision boundaries.



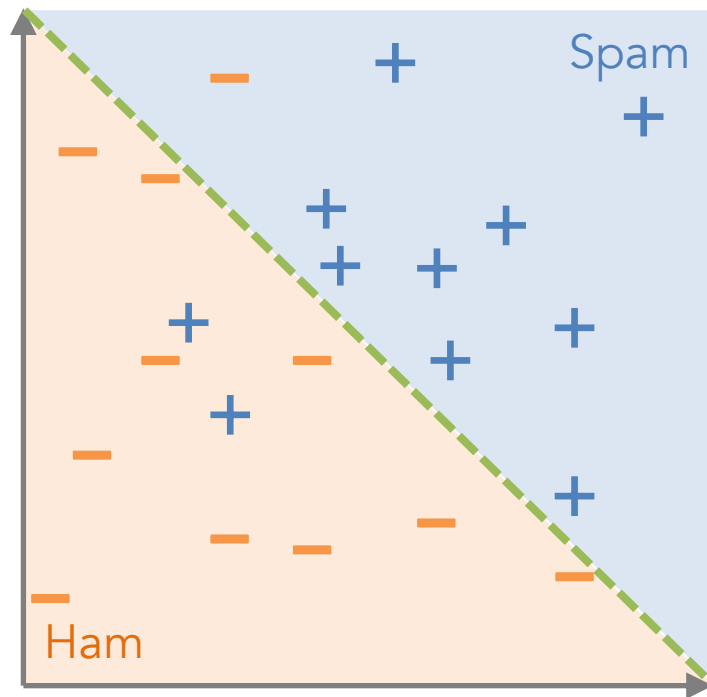
## Classification

Depending on the needs, we can train **different models** (using different loss functions) with various shapes of decision boundaries.



## Classification

To evaluate our classification model, we need to compute **evaluation metrics** to measure and quantify model performance, such as the **accuracy** of all data.



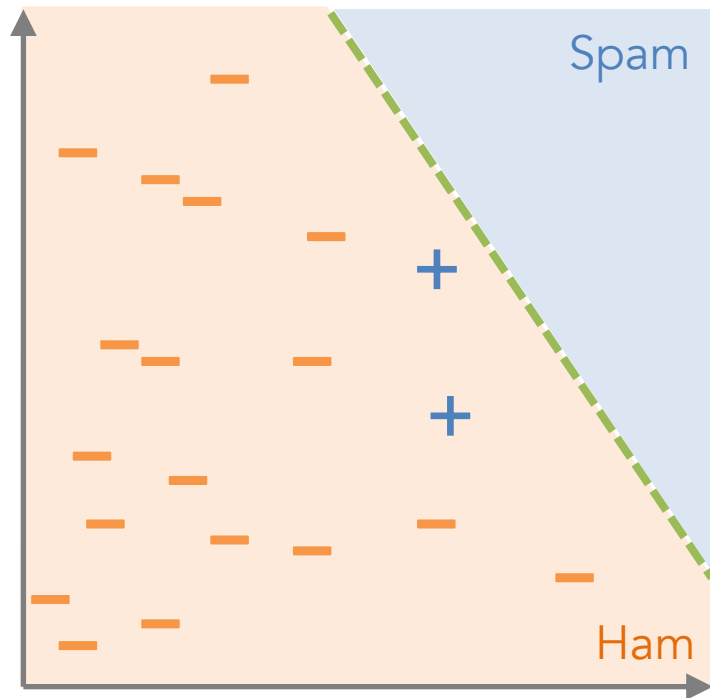
Accuracy for all data

$$= \frac{\text{\# of correctly classified points}}{\text{\# of all points}}$$

$$= \frac{19}{22} = 0.86$$

Classification

But what if **the dataset is imbalanced** (i.e., some classes have far less data)? In this case, the accuracy of all data is a bad evaluation metric.



Classify all data as non-spam

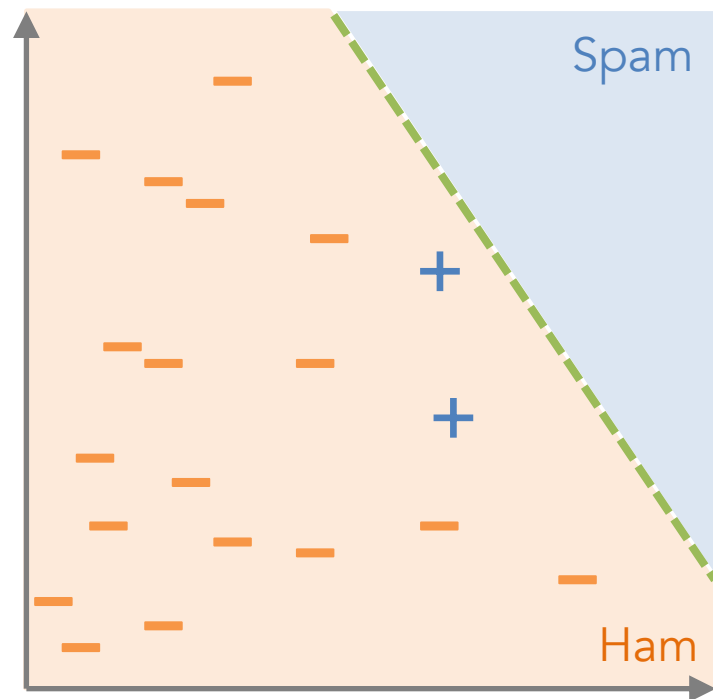
Accuracy for all data

$$= \frac{\text{\# of correctly classified points}}{\text{\# of all points}}$$

$$= \frac{18}{20} = 0.9 \quad \text{☹️}$$

## Classification

Instead of computing the accuracy for all the data, we can compute **accuracy for each class**, which allows us to see the performance of different labels.



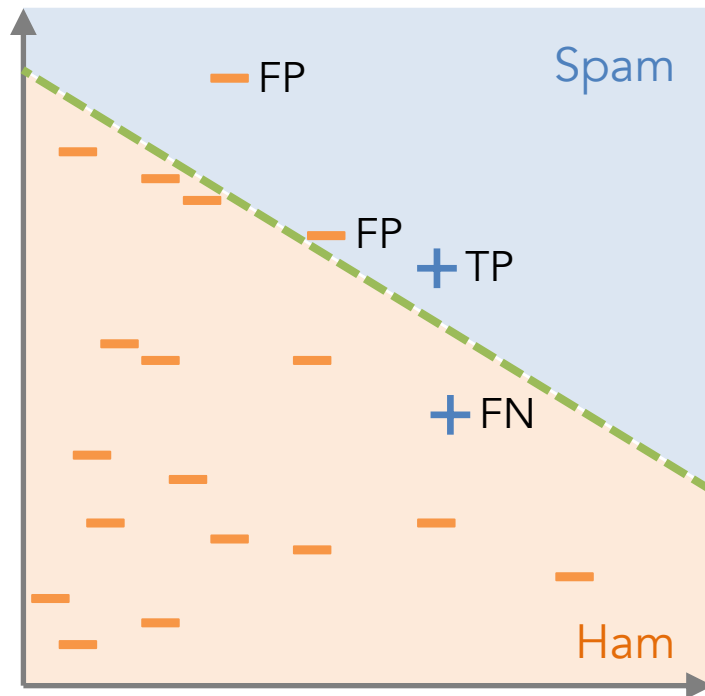
$$\text{Accuracy for spam} = \frac{0}{2} = 0$$

(true positive rate, recall, sensitivity)

$$\text{Accuracy for ham} = \frac{18}{18} = 1$$

(true negative rate, specificity)

If we care more about the positive class (e.g., spam), we can use **precision** and **recall**, with its best value at 1 and the worst value at 0.



TP = 1 (True Positive)

FP = 2 (False Positive)

FN = 1 (False Negative)

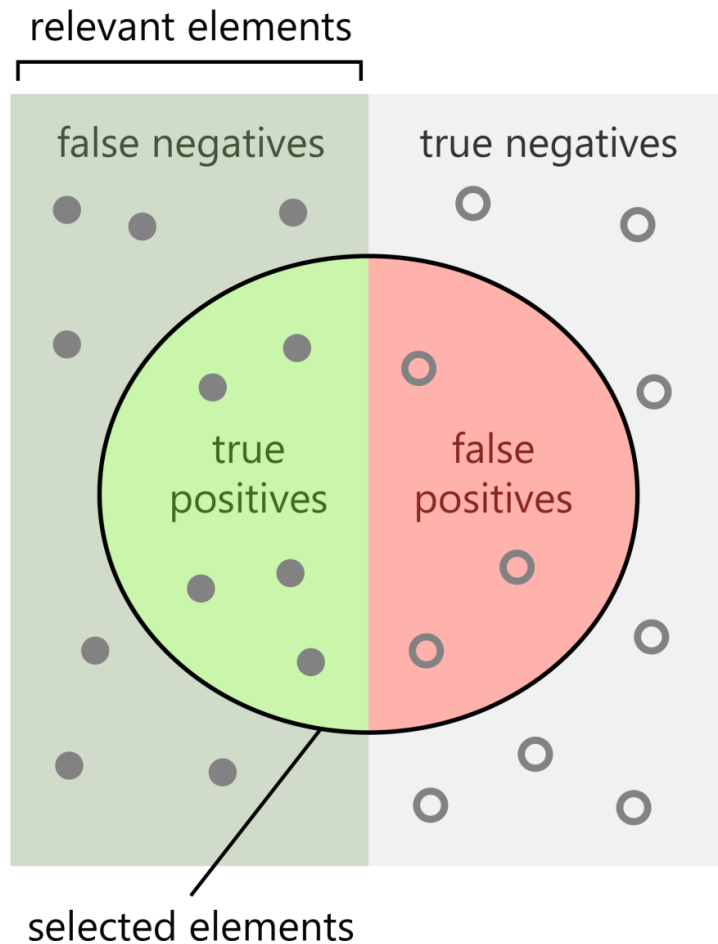
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = 0.33$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = 0.5$$



## Classification

Precision and recall can be aggregated into **F-score** as a general model performance, with its best value at 1 and worst value at 0.



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

$$\text{F-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

**Exercise 2.1:** Suppose that we fit a binary classification model in identifying spam and ham (i.e., non-spam). Spam is the positive label, and ham is the negative label.

- 40 samples are predicted as spam, and they are indeed spam in reality
- 20 samples are predicted as spam, but it turns out that they are not spam in reality
- 60 samples are predicted as ham, but it turns out that they are spam in reality
- 80 samples are predicted as ham, and they are indeed ham in reality

What are the precision, recall, and f-score (F) of the model?

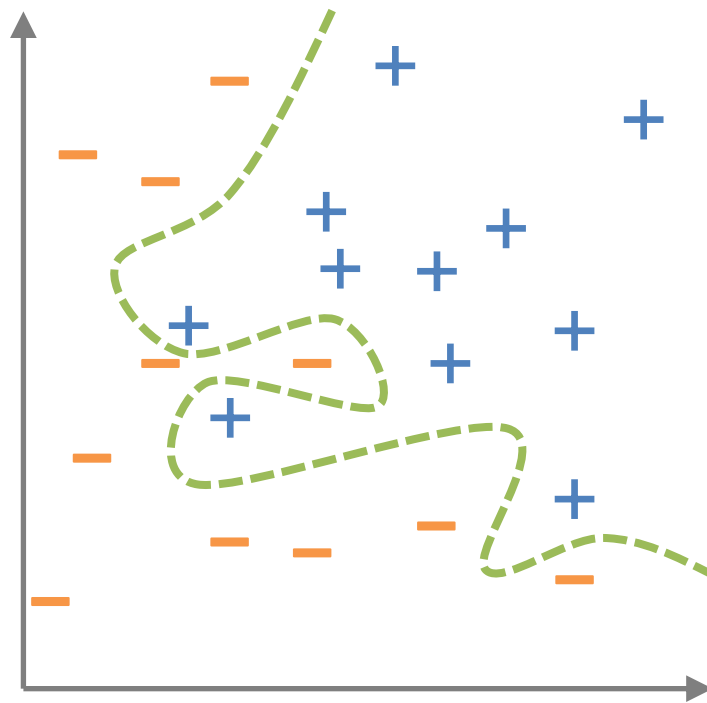
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

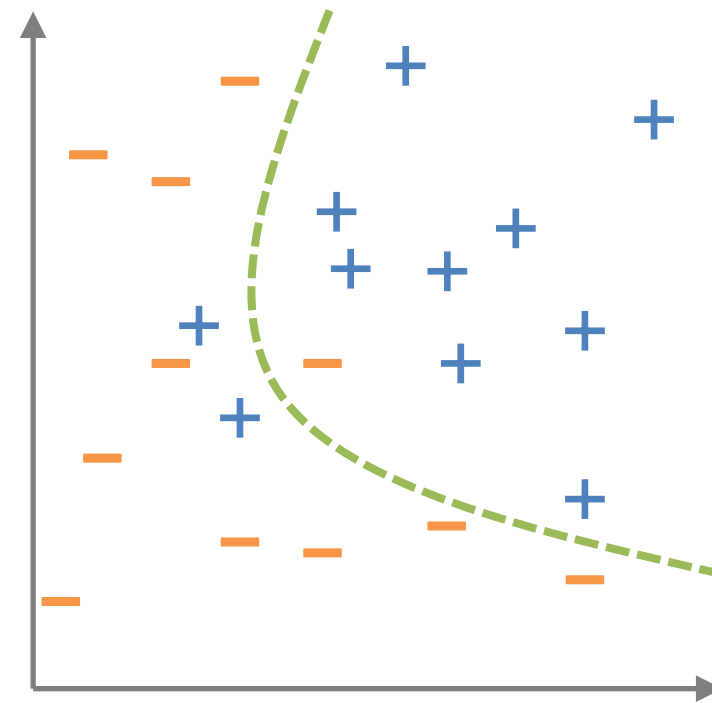
$$F = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Classification

We can train different types of models. But how do we know **which one is better**? Can we just pick an evaluation metric to determine which model is good?



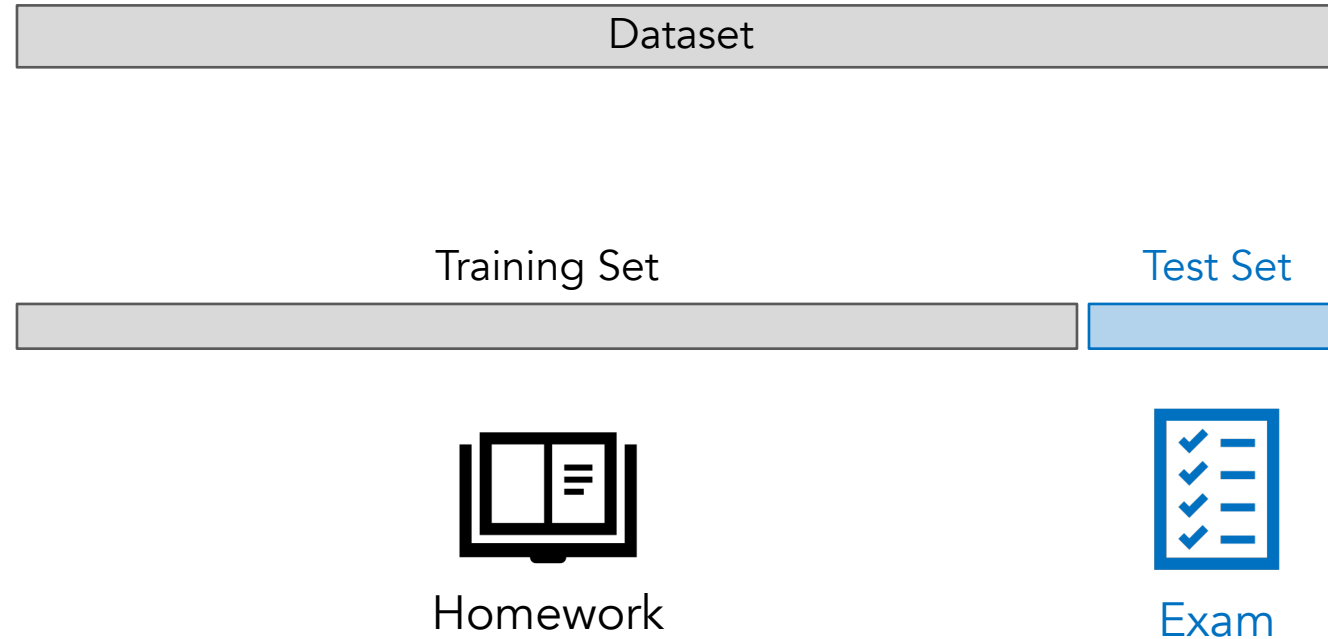
Model A



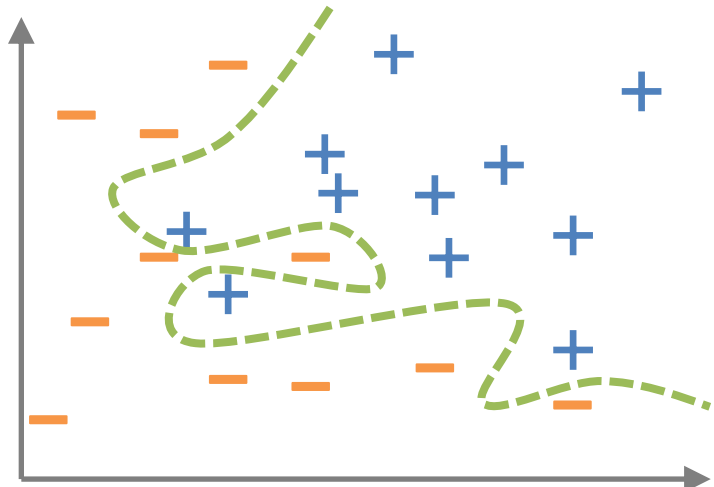
Model B

## Classification

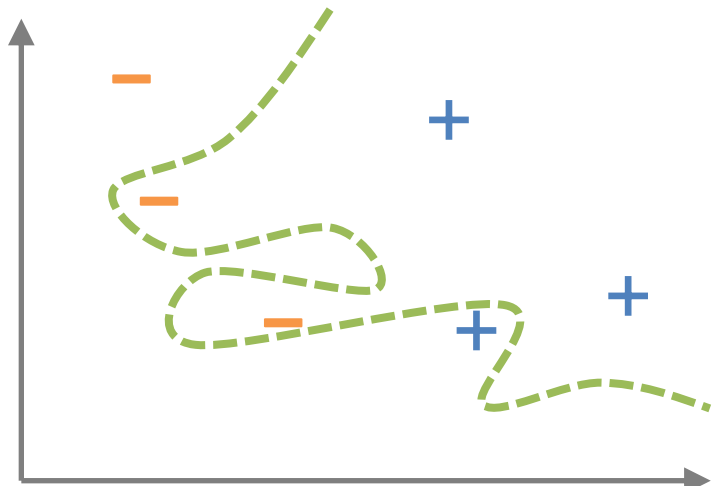
To choose models, we need a **test set**, which contains data that the models **have not yet seen before** during the training phase.



☹️ Model A

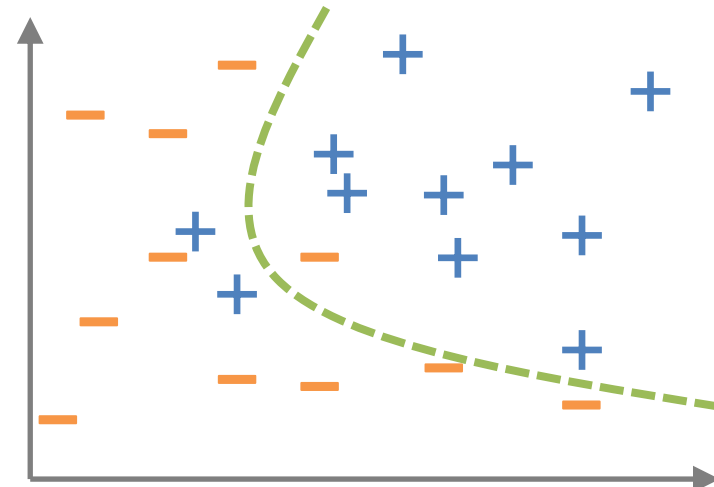


Training Accuracy = 1

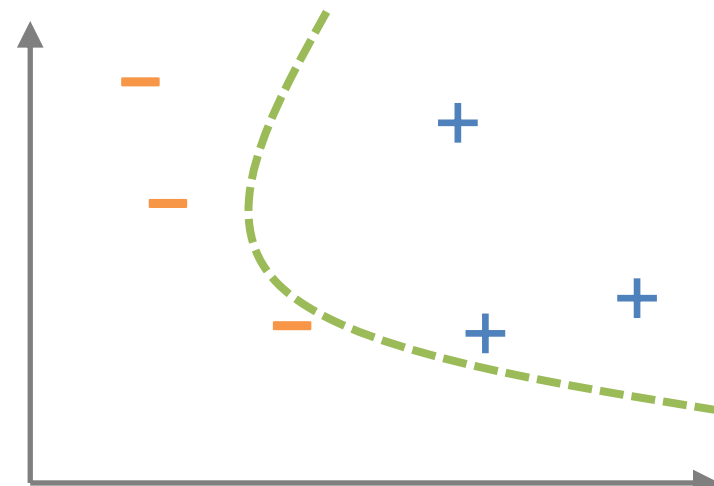


Testing Accuracy = 0.5

😊 Model B



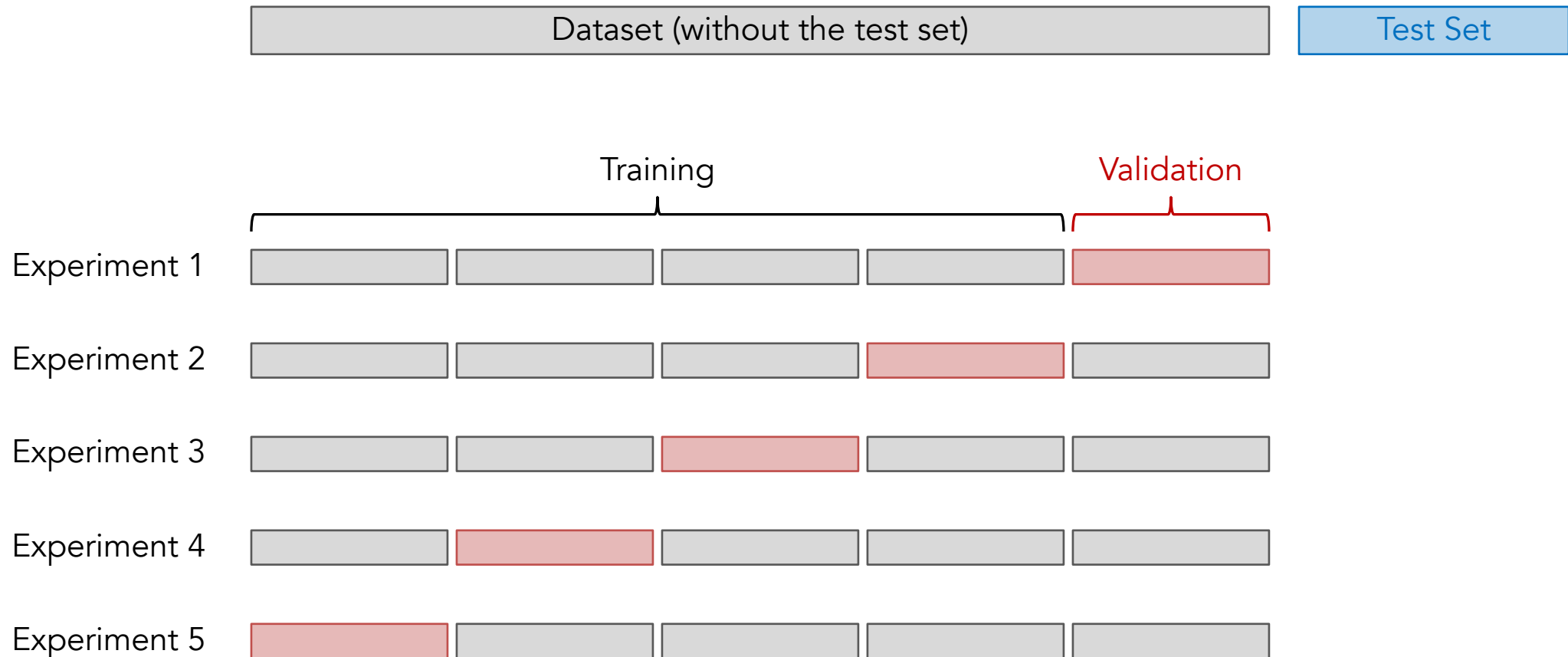
Training Accuracy = 0.86



Testing Accuracy = 1

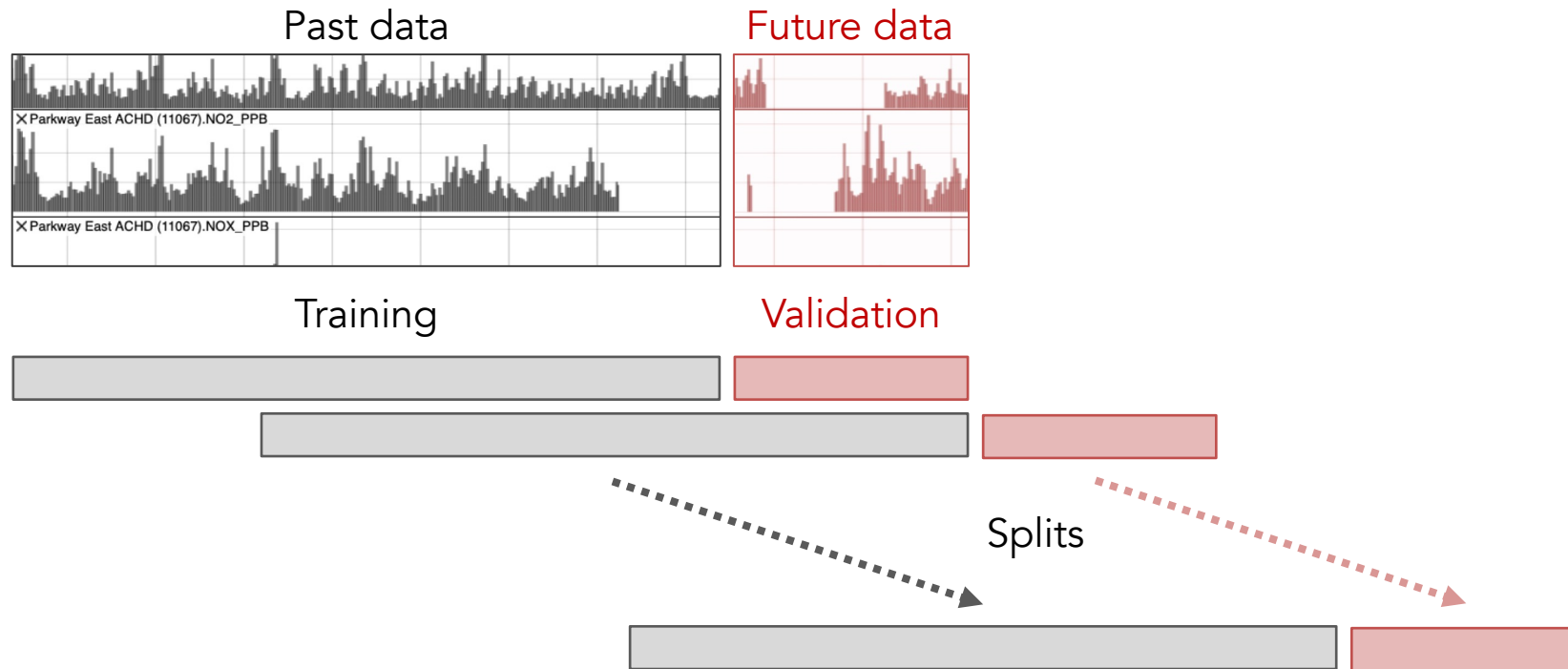
## Classification

To tune hyper-parameters for a model, we use cross-validation to divide the dataset into folds and use each fold for validation.



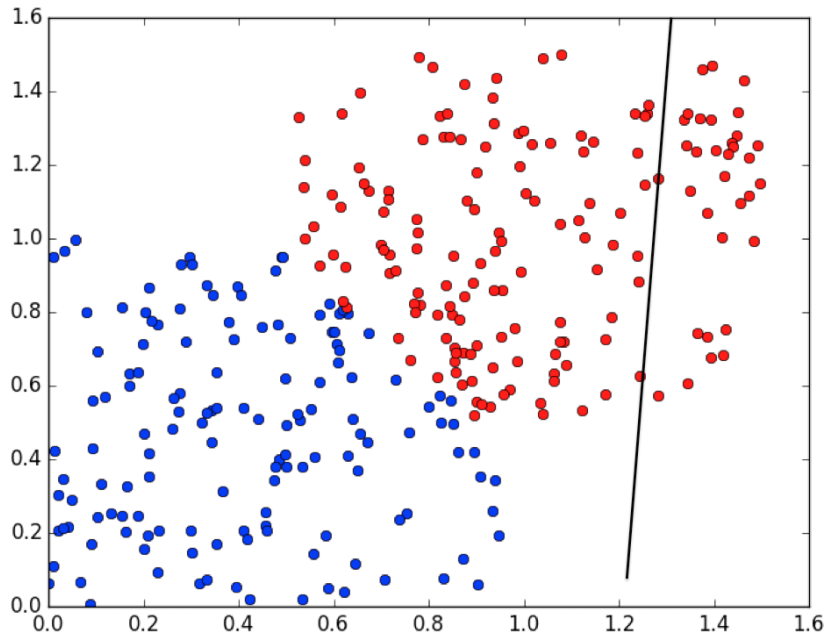
## Classification

For time-series data, it is better to do the split for cross-validation based on the order of time intervals, which means we only use data in the past to predict the future, but not the other way around.

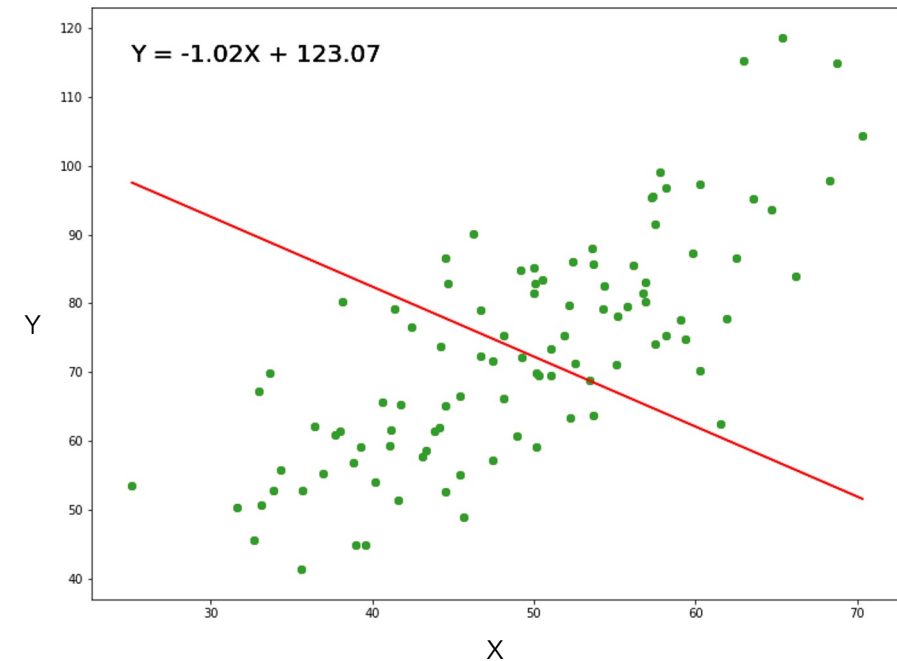


## Regression

Unlike classification (which separates data into categories), **regression** fits a function that maps features  $x$  to a **continuous variable  $y$**  (i.e., the response).



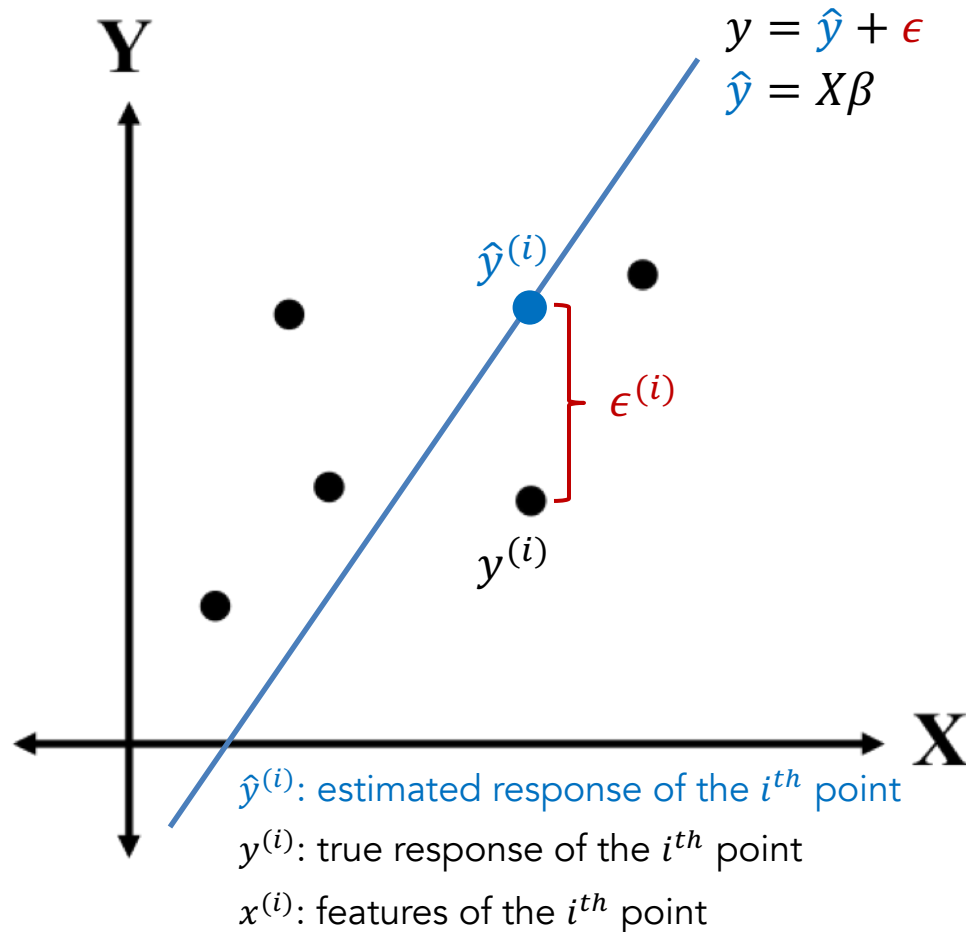
- [Classification] How can we fit a function that separates data points into different groups?



- [Regression] How can we fit a function that maps features (input) to a continuous variable (output)?



**Linear regression** fits a linear function  $f$  that maps  $x$  to  $y$  using some **error metric**, which can best describe the linear relationship between variables  $x$  and  $y$ .



$y$ : true response (in vector form)

$\hat{y}$ : estimated response (in vector form)

$X$ : predictors/features (in matrix form)

$\beta$ : coefficient (in vector form)

$\epsilon$ : error/noise/residual (in vector form)

$X = [\mathbf{1} \quad x_1] = \begin{bmatrix} 1 & x_1^{(1)} \\ \vdots & \vdots \\ 1 & x_1^{(n)} \end{bmatrix}$

$\beta_0$ : intercept

$\beta_1$ : slope

$x_1$ : first predictor

$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$

$\hat{y} = X\beta = \beta_0 + \beta_1 x_1 = \begin{bmatrix} \hat{y}^{(1)} \\ \vdots \\ \hat{y}^{(n)} \end{bmatrix}$

We use the vector and matrix forms to simplify equations.

$$X = [\mathbf{1} \quad \boxed{x_1} \quad \cdots \quad x_p] = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_p^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{(n)} & \cdots & x_p^{(n)} \end{bmatrix} \quad \beta = \begin{bmatrix} \boxed{\beta_0} \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$$

Vector                      Matrix                      Scalar

$$\hat{y} = f(x) = X\beta = [\mathbf{1} \quad x_1 \quad \cdots \quad x_p] \cdot \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix} = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p = \begin{bmatrix} \hat{y}^{(1)} \\ \vdots \\ \hat{y}^{(n)} \end{bmatrix}$$

Vector

# Regression

We can map vector and matrix forms to data directly.

Sum of smell ratings

...
25
40
45
...

Vector  $\begin{bmatrix} \hat{y}^{(1)} \\ \vdots \\ \hat{y}^{(n)} \end{bmatrix}$

H2S in PPM

SO2 in PPM

...

Wind direction in DEG

Wind speed in MPH

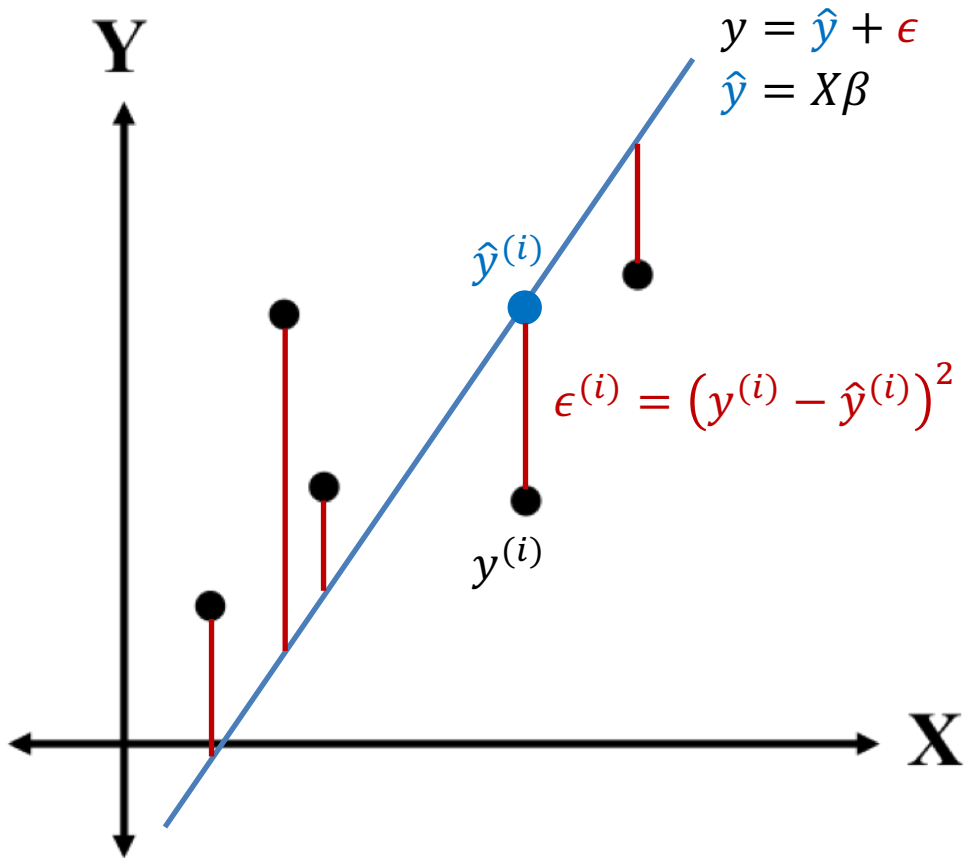
...	...	...	...	...
0.019	0.020	...	215.0	3,2
0.130	0.033	...	199.0	3.4
0.095	0.044	...	184.0	2.9
...	...	...	...	...

Vector  $x_1$

Matrix  $\begin{bmatrix} x_1^{(1)} & \cdots & x_p^{(1)} \\ \vdots & \vdots & \vdots \\ x_1^{(n)} & \cdots & x_p^{(n)} \end{bmatrix}$

## Regression

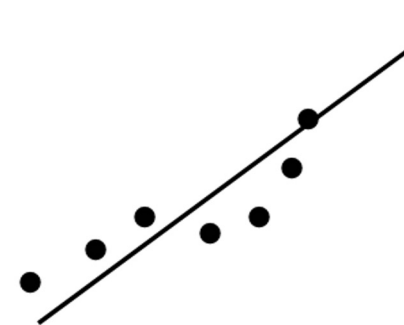
Usually, we assume that the error  $\epsilon$  is IID (independent and identically distributed) and follows a normal distribution with zero mean and some variance  $\sigma^2$ .



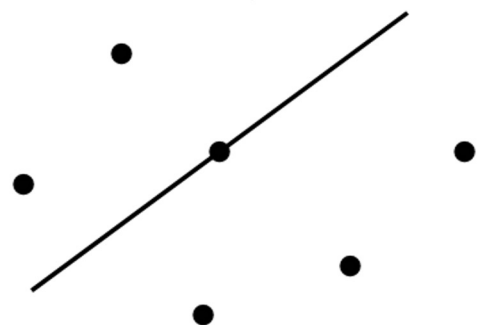
$$\epsilon \sim \text{iid } N(0, \sigma^2)$$

$$\text{total errors} = \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2$$

Small  $\sigma^2$

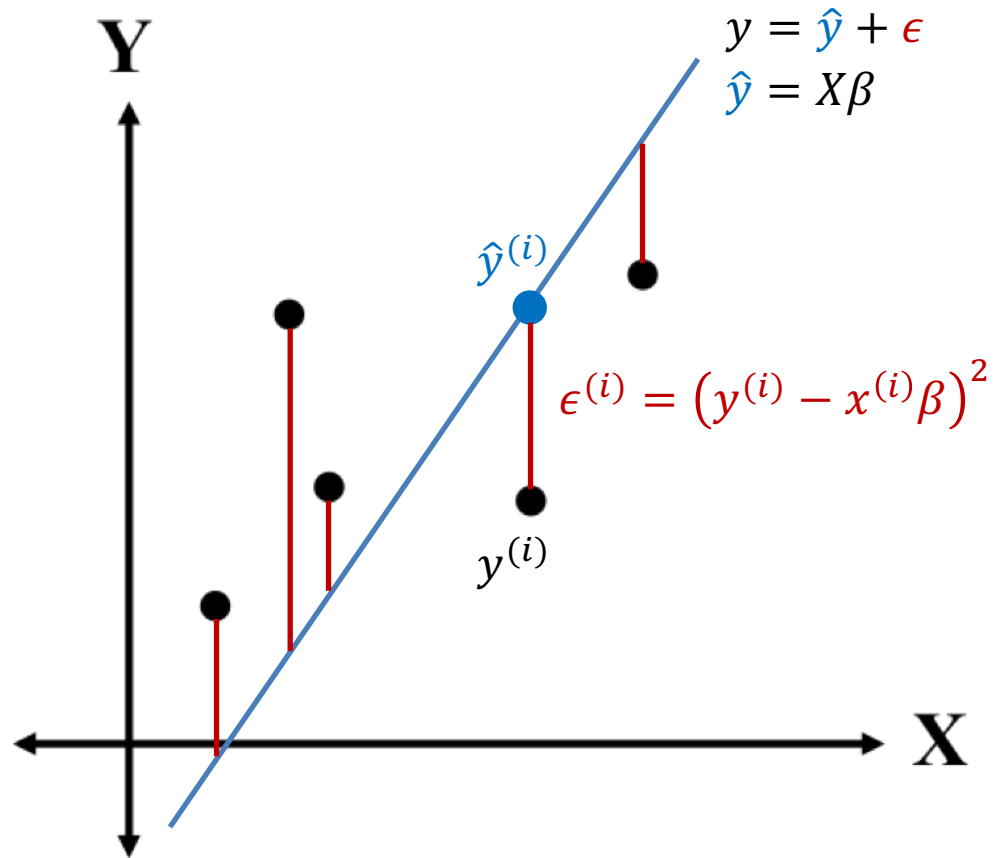


Large  $\sigma^2$

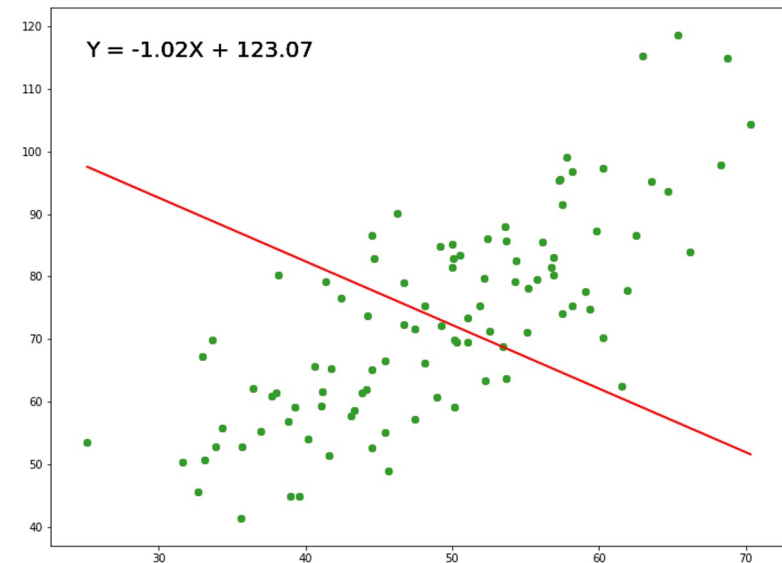


## Regression

To find the optimal coefficient  $\beta$ , we need to **minimize the error** (the sum of squared errors) using gradient descent or taking the derivative of its matrix form.

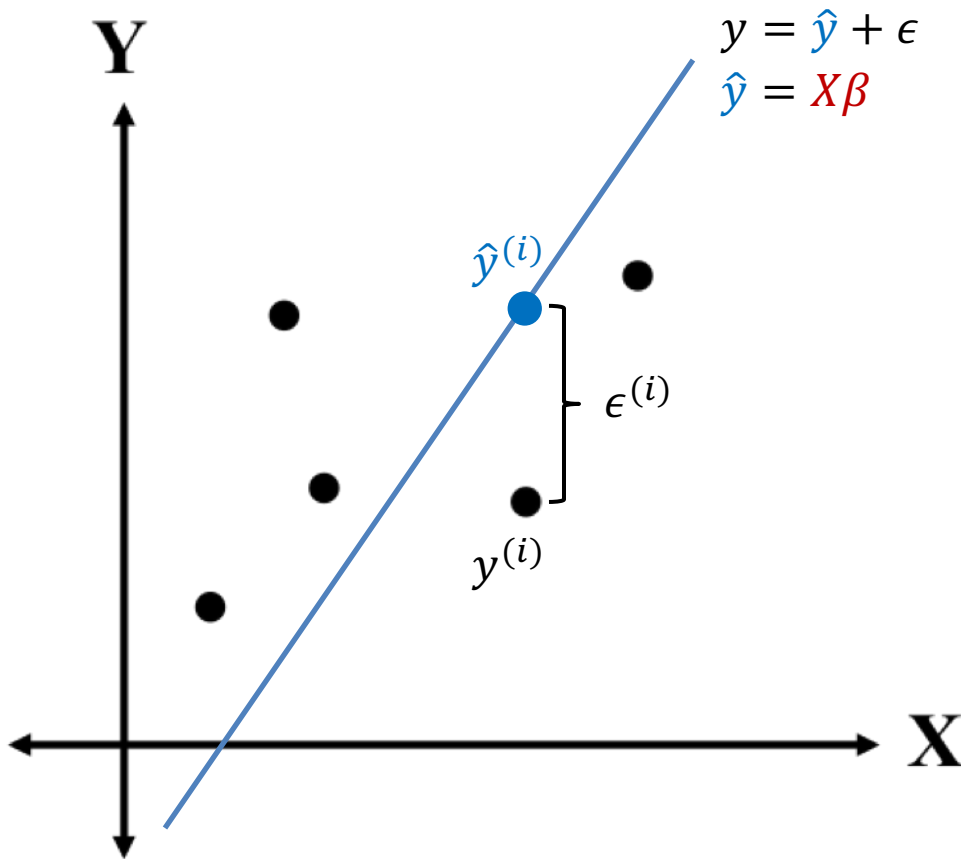


$$\begin{aligned}\min_{\beta} \sum_{i=1}^n \epsilon^{(i)} &= \min_{\beta} \sum_{i=1}^n (y^{(i)} - x^{(i)}\beta)^2 \\ &= \min_{\beta} (y - X\beta)^T (y - X\beta)\end{aligned}$$



## Regression

We can generalize linear regression to have **multiple predictors** (i.e., multiple linear regression) and keep the original mathematical representation.



$x_j^{(i)}$ : the  $j^{\text{th}}$  predictor of the  $i^{\text{th}}$  data point

$y^{(i)}$ : response of the  $i^{\text{th}}$  data point

$\hat{y}^{(i)}$ : estimated response of the  $i^{\text{th}}$  data point

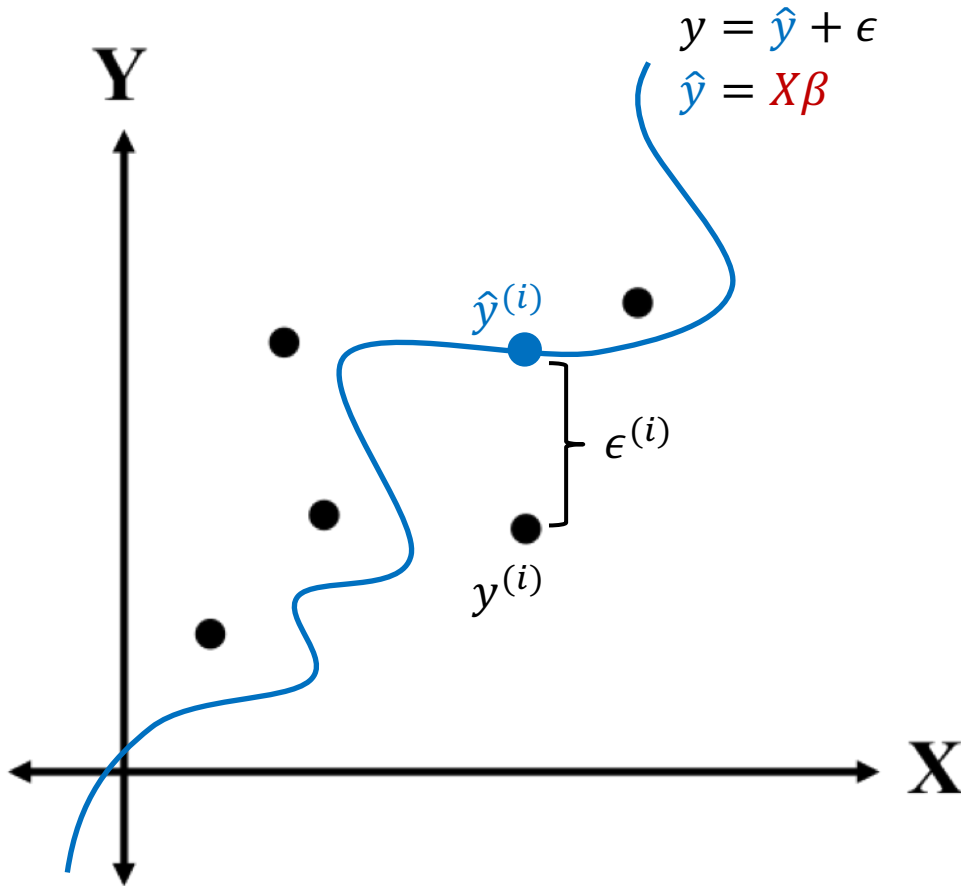
$$X = [\mathbf{1} \quad x_1 \quad \cdots \quad x_p] = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_p^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{(n)} & \cdots & x_p^{(n)} \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$$

$$\hat{y} = X\beta = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p = \begin{bmatrix} \hat{y}^{(1)} \\ \vdots \\ \hat{y}^{(n)} \end{bmatrix}$$

## Regression

We can model a **non-linear relationship** using polynomial functions with degree  $k$ . The example below uses one predictor  $x_1$ .



$y$ : true response (in vector form)

$\hat{y}$ : estimated response (in vector form)

$X$ : predictors/features (in matrix form)

$\beta$ : coefficient (in vector form)

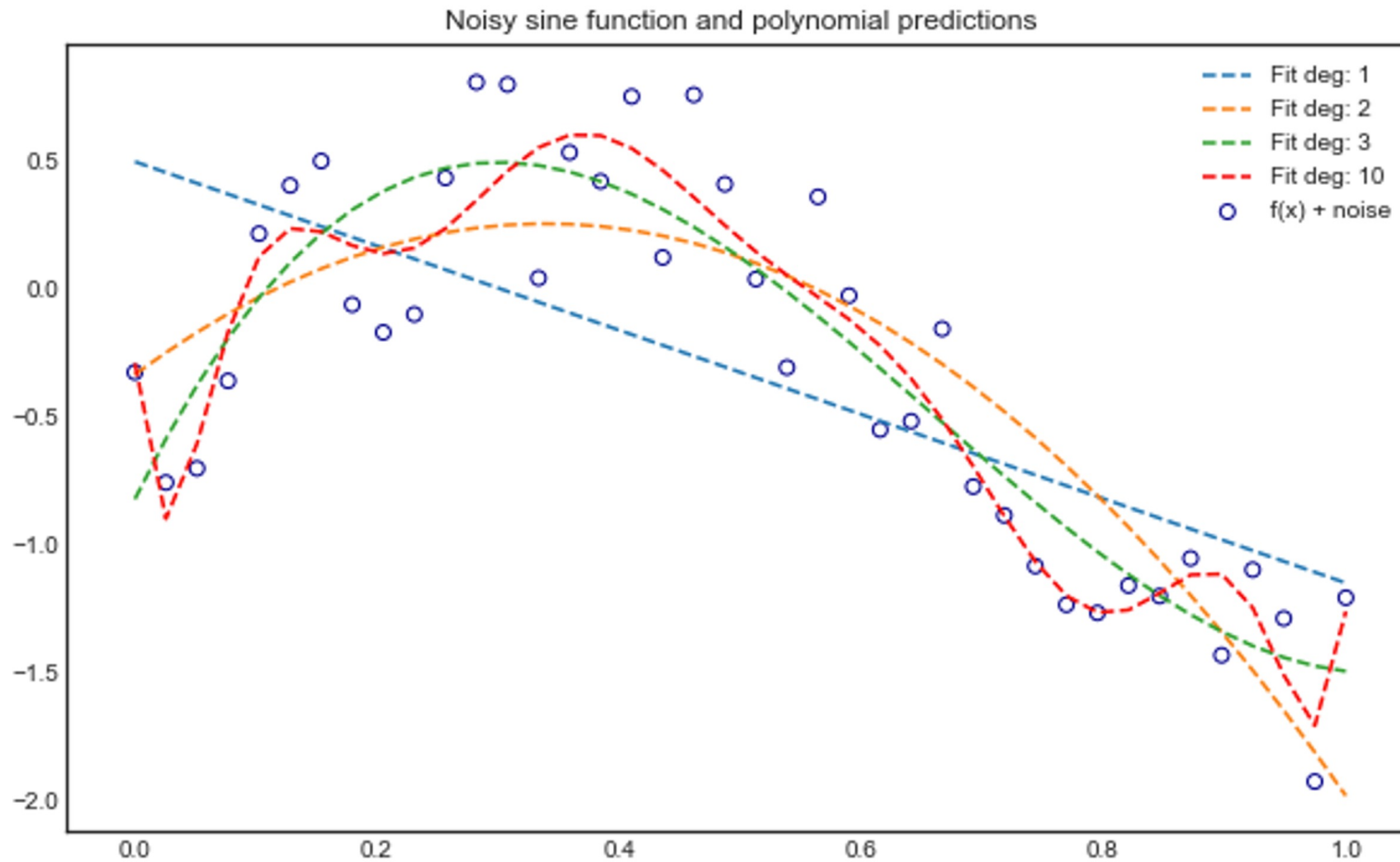
$$X = [\mathbf{1} \quad x_1 \quad (x_1)^2 \quad \cdots \quad (x_1)^k]$$

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}$$

$$\hat{y} = X\beta = \beta_0 + \beta_1 x_1 + \beta_2 (x_1)^2 + \cdots + \beta_k (x_1)^k$$

## Regression

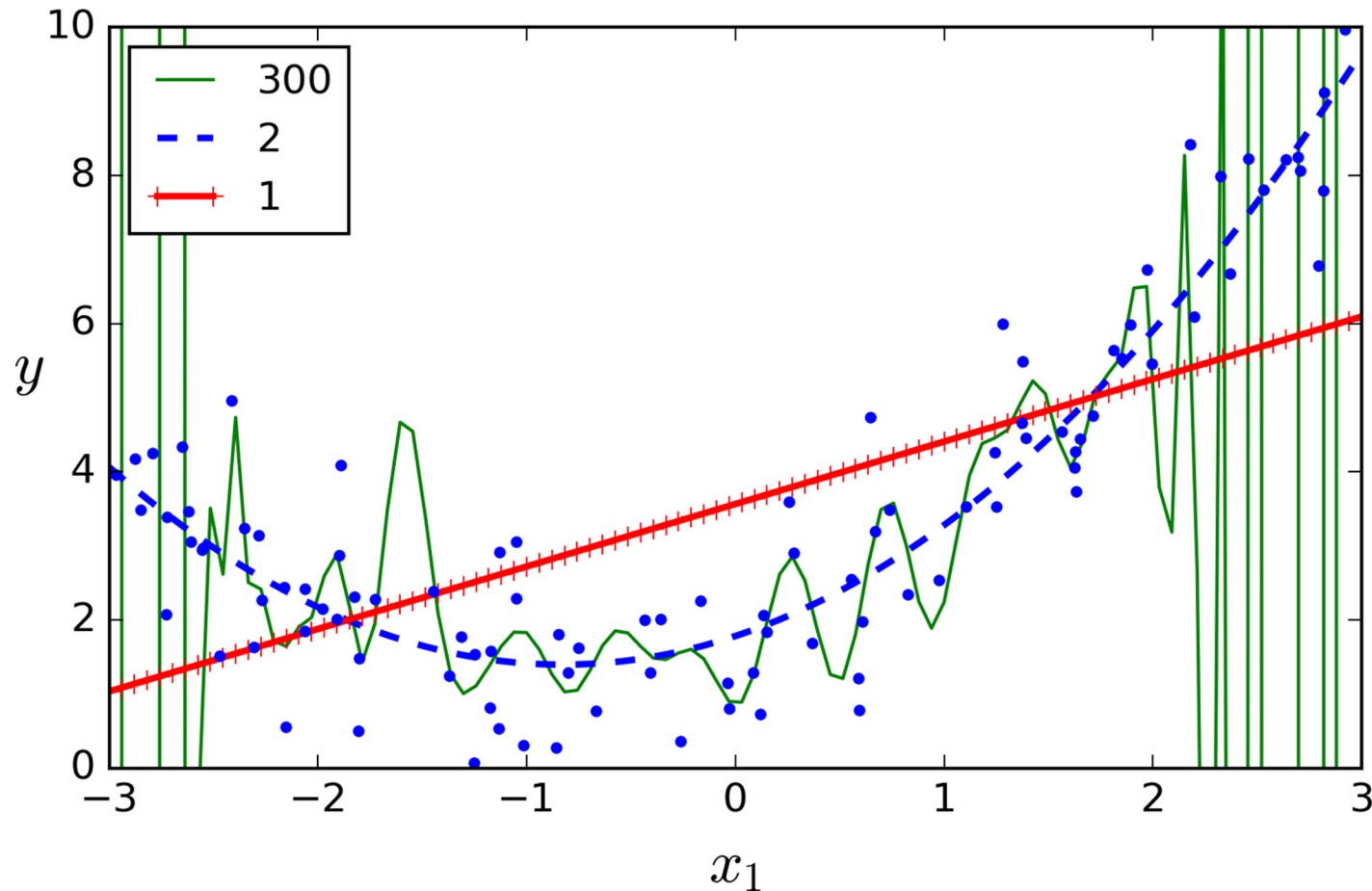
Here is an example of applying linear and polynomial regression to the data that is created using a sine function with some random noise.





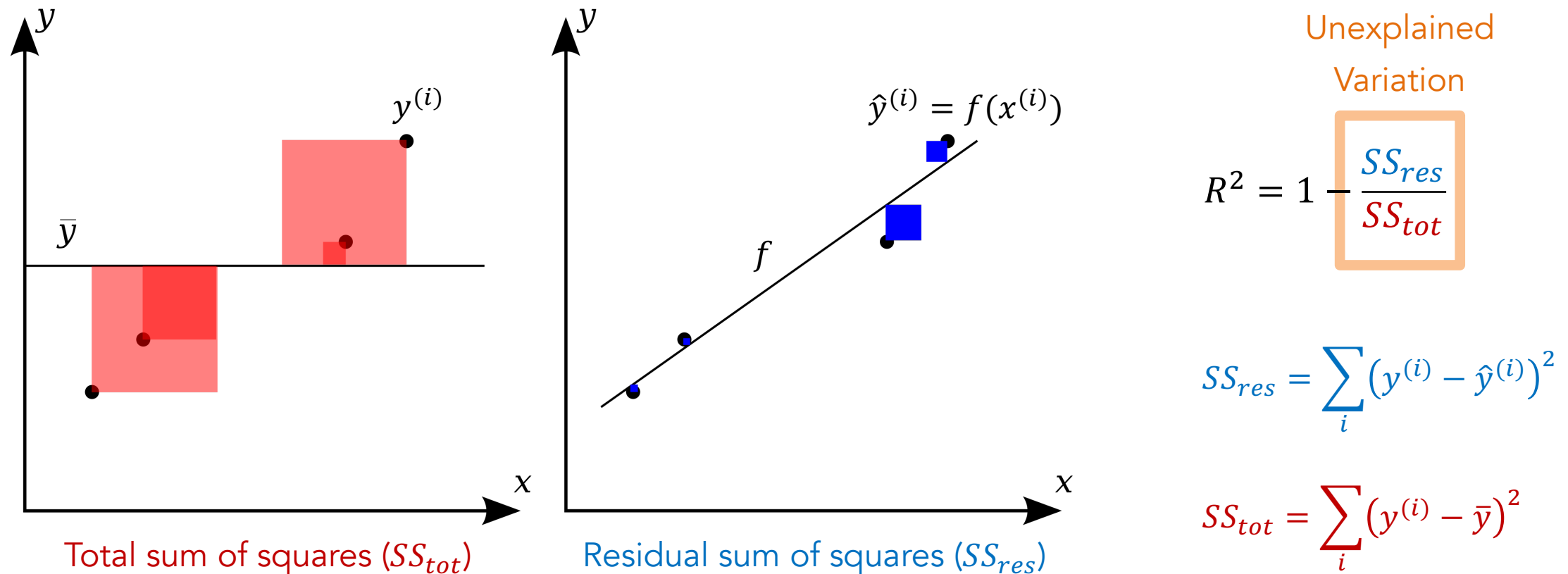
## Regression

Using too complex/simple models can lead to **overfitting/underfitting**, which means the model fits the training set well but generalizes poorly on the test set.



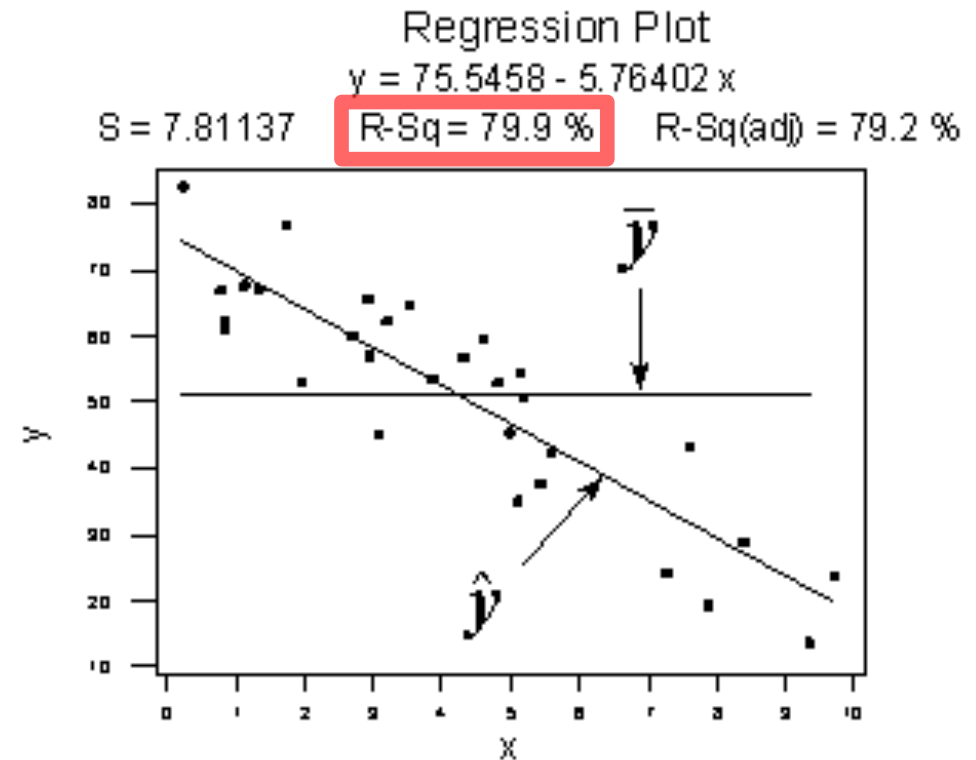
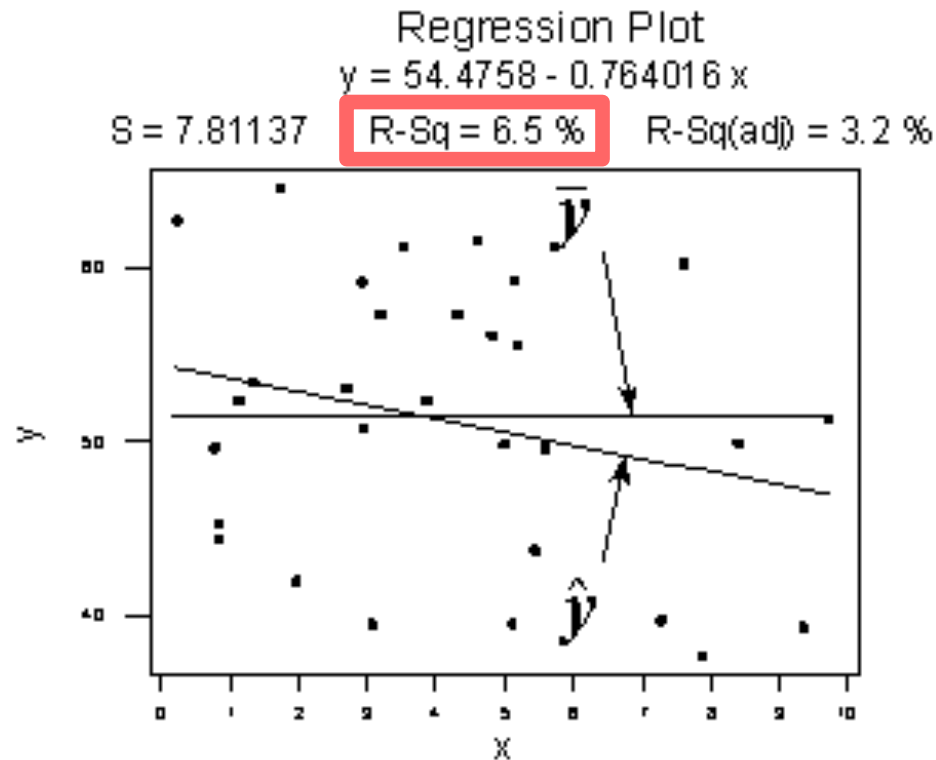
## Regression

To evaluate regression models, one common metric is the **coefficient of determination (R-squared,  $R^2$ )**. There exist other metrics such as AIC (Akaike's Information Criterion) that is based on likelihood, which is not covered in this lecture.



## Regression

For simple/multiple linear regression,  $R^2$  equals the square of Pearson correlation coefficient  $r$  between the true  $y$  and the estimated  $\hat{y} = f(X)$ .



## Regression

$R^2$  increases as we add more predictors (because the optimization always want to decrease the residual sum of squares) and thus is **not a good metric for model selection**. We need the adjusted  $R^2$ , which considers the number of predictors.

$$R_{adj}^2 = 1 - \frac{SS_{res}/df_{res}}{SS_{tot}/df_{tot}}$$

$p$ : number of features/predictors

$R_{adj}^2$ : adjusted value of  $R^2$

$$df_{res} = n - p - 1$$

$df_{res}$ : residual degree of freedom

$$df_{tot} = n - 1$$

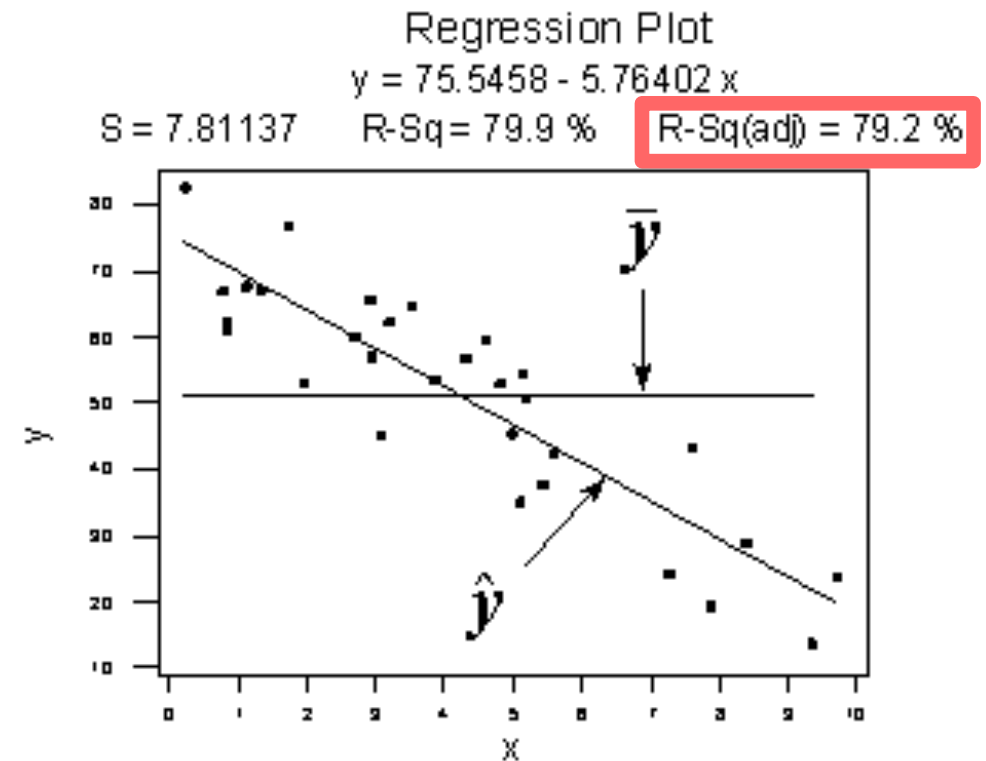
$df_{tot}$ : total degree of freedom

$$SS_{res} = \sum_i (y^{(i)} - \hat{y}^{(i)})^2$$

$SS_{res}$ : residual sum of squares

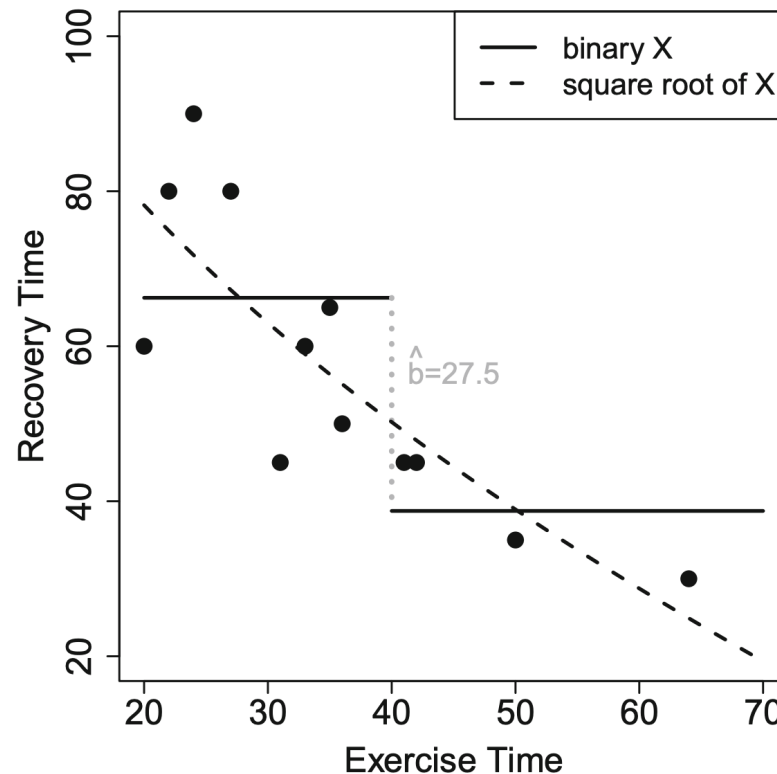
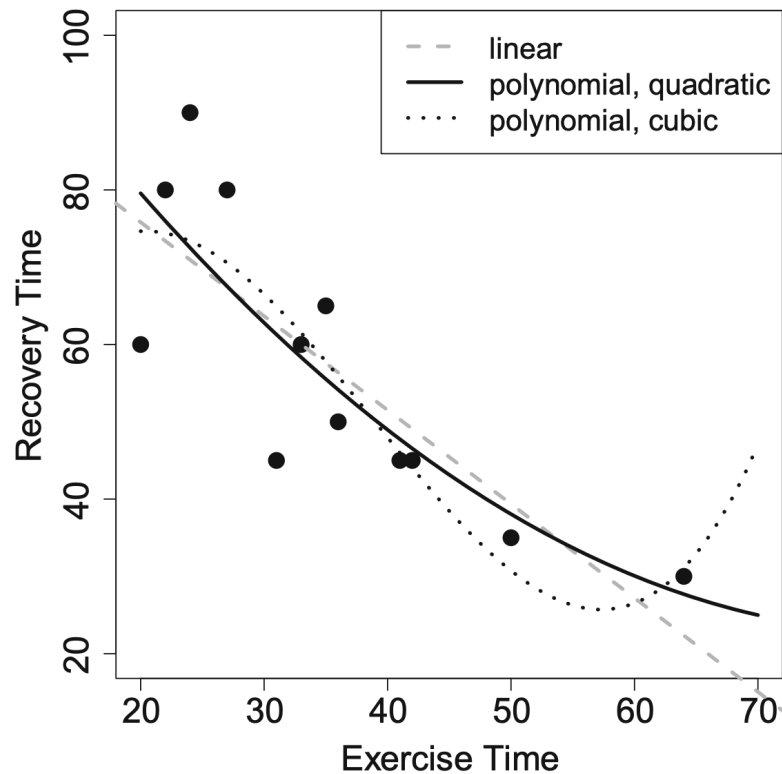
$$SS_{tot} = \sum_i (y^{(i)} - \bar{y})^2$$

$SS_{tot}$ : total sum of squares



## Regression

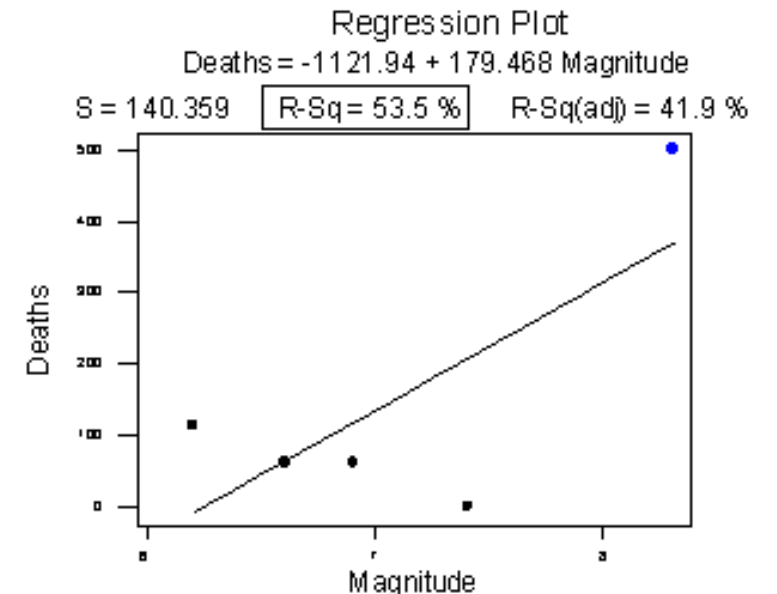
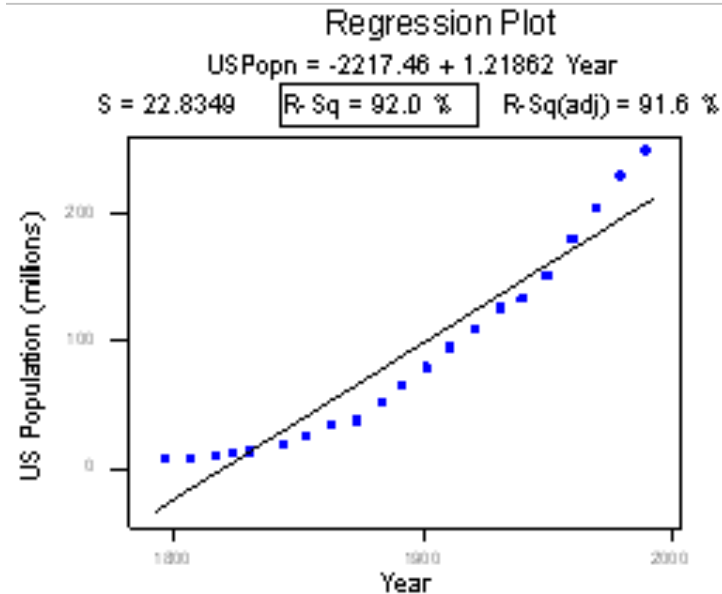
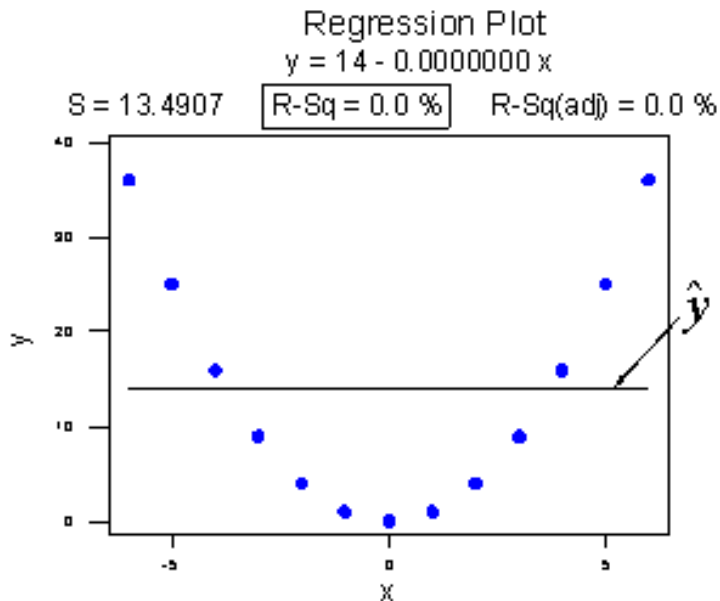
In the example below,  $R^2$  is larger for the model with more predictors (i.e., the **cubic model** that has three predictors). The **adjusted  $R^2$** , which considers the number of predictors (model complexity), favors the **square-root model**.



	$R^2$	$R^2_{adj}$
Linear	0.6584	0.6243
Quadratic	0.6787	0.6074
Cubic	0.7151	0.6083
Square root	0.6694	0.6363

## Regression

Be careful when using and explaining  $R^2$  in your findings. A bad  $R^2$  does not always mean no pattern in the data. A good  $R^2$  does not always mean that the function fits the data well. And  $R^2$  can be greatly affected by outliers.



# Take-Away Messages

- Classification outputs discrete labels, while regression outputs continuous values.
- Precision, recall, and F-score are common metrics for evaluating classification models.
- R-squared is a common evaluation metric for regression models.
- Feature engineering is an important step for models that do not use deep learning techniques.
- To train and update a model iteratively, you need a loss function to measure errors.
- Generally, it is a good practice to divide datasets into different parts for model training and testing.
- A model can perform extremely well on the training set but badly on the test set (i.e., overfitting).
- Cross-validation is a good technique to prevent overfitting.



Questions?