香港浸會大學
HONG KONG BAPTIST UNIVERSITY

DEPARTMENT OF
COMPUTER SCIENCE
計算機科學系

COMP7015 Artificial Intelligence

Lecture 6: Machine Learning II

Instructor: Dr. Kejing Yin

October 13, 2022

Recap: Entropy & Information Gain

- Entropy of dataset D: $\boxed{\mathrm{Ent}(D) = -\sum_{k=1}^{K} p_k \log_2 p_k}$

  K: number of classes p_k is the frequency of the k-th class

- The smaller $\mathrm{Ent}(D)$, the purer D.

- Information Gain:

$$\mathrm{Gain}(D, a) = \boxed{\mathrm{Ent}(D)} - \boxed{\sum_{v=1}^{V} \frac{|D^v|}{|D|} \mathrm{Ent}(D^v)}$$

D^v: dataset that has value of v in D

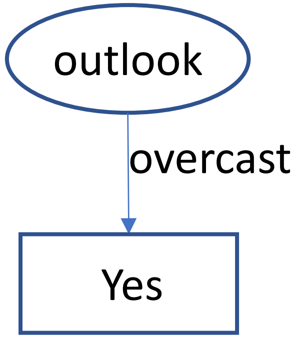
purity before split *purity after split*

Recap: ID3 Algorithm

```
ID3(D,X) =
  Let T be a new tree
  If all instances in D have same class c
    Label(T) = c; Return T
  If X = ∅ or no attribute has positive information gain
    Label(T) = most common class in D; return T
  X ← attribute with highest information gain
  Label(T) = X
  For each value x of X
    D_x ← instances in D with X = x
    If D_x is empty
      Let T_x be a new tree
      Label(T_x) = most common class in D
    Else
      T_x = ID3(D_x, X – { X })
    Add a branch from T to T_x labeled by x
  Return T
```

Outlook	Temperature	Humidity	Windy	Play?
overcast	hot	high	false	Yes
overcast	cool	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes

Same class



Recap: ID3 Algorithm

```
ID3(D,X) =
  Let T be a new tree
  If all instances in D have same class c
    Label(T) = c; Return T
  If X = ∅ or no attribute has positive information gain
    Label(T) = most common class in D; return T
  X ← attribute with highest information gain
  Label(T) = X
  For each value x of X
    D_x ← instances in D with X = x
    If D_x is empty
      Let T_x be a new tree
      Label(T_x) = most common class in D
    Else
      T_x = ID3(D_x, X – { X })
    Add a branch from T to T_x labeled by x
  Return T
```

Color	Purchase?
Red	Yes
Red	Yes
Red	No
Blue	Yes
Blue	Yes
Blue	No

$\log_2 3 \approx 1.585$

Compute $\text{Gain}(D, \text{“Color”})$

$$\text{Gain}(D, \text{“Color”}) = Ent(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} \text{Ent}(D^v)$$
$$= 0.918 - \left(\frac{1}{2} * 0.918 + \frac{1}{2} * 0.918\right)$$
$$= 0$$

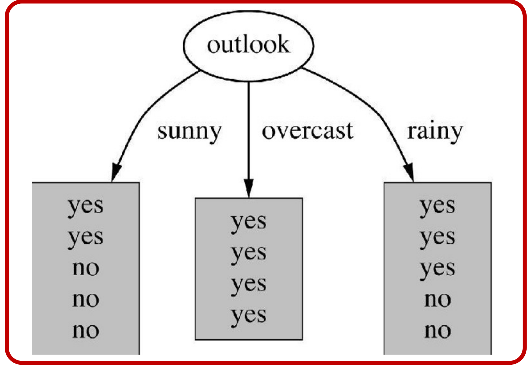
Do we need to split D in ID3 algorithm?

No: no attribute has positive information gain

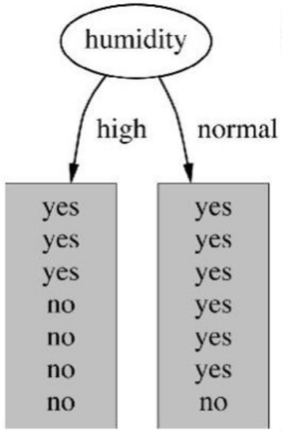
Recap: ID3 Algorithm

```
ID3(D,X) =
  Let T be a new tree
  If all instances in D have same class c
    Label(T) = c; Return T
  If X = ∅ or no attribute has positive information gain
    Label(T) = most common class in D; return T
  X ← attribute with highest information gain
  Label(T) = X
  For each value x of X
    D_x ← instances in D with X = x
    If D_x is empty
      Let T_x be a new tree
      Label(T_x) = most common class in D
    Else
      T_x = ID3(D_x, X – { X })
    Add a branch from T to T_x labeled by x
  Return T
```

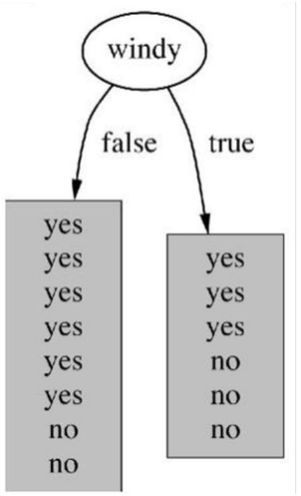
highest



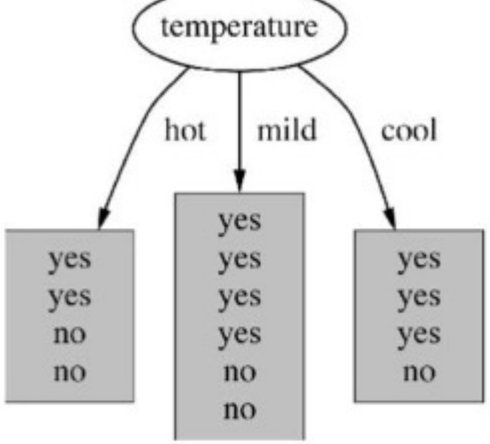
$\text{Gain}(D, \text{"outlook"}) = 0.246$



$\text{Gain}(D, \text{"humidity"}) = 0.152$



$\text{Gain}(D, \text{"windy"}) = 0.048$



$\text{Gain}(D, \text{"temperature"}) = 0.029$

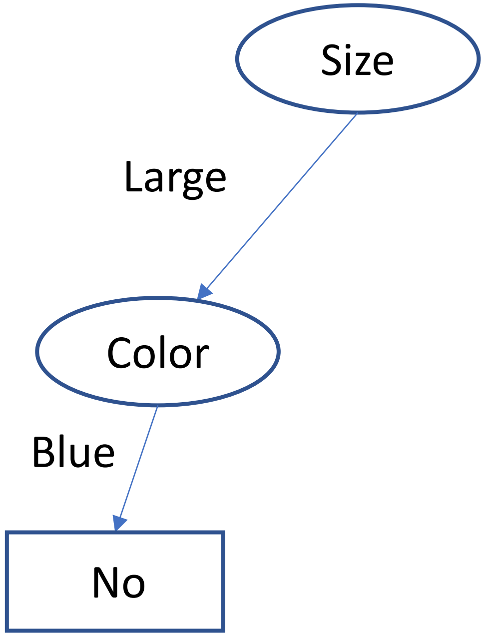
Recap: ID3 Algorithm

```
ID3(D,X) =
    Let T be a new tree
    If all instances in D have same class c
        Label(T) = c; Return T
    If X = ∅ or no attribute has positive information gain
        Label(T) = most common class in D; return T
    X ← attribute with highest information gain
    Label(T) = X
    For each value x of X
        D_x ← instances in D with X = x
        If D_x is empty
            Let T_x be a new tree
            Label(T_x) = most common class in D
        Else
            T_x = ID3(D_x, X – { X })
        Add a branch from T to T_x labeled by x
    Return T
```

Color: Blue, Red

Size: Large, Small

Size	Color	Purchase?
Large	Red	Yes
Large	Red	Yes
Large	Red	No
		No
		No
		No



D_x is empty when x = "Blue"

Recap: ID3 Algorithm

```
ID3(D,X) =
  Let T be a new tree
  If all instances in D have same class c
    Label(T) = c; Return T
  If X = ∅ or no attribute has positive information gain
    Label(T) = most common class in D; return T
  X ← attribute with highest information gain
  Label(T) = X
  For each value x of X
    D_x ← instances in D with X = x
    If D_x is empty
      Let T_x be a new tree
      Label(T_x) = most common class in D
    Else
      T_x = ID3(D_x, X – { X })
    Add a branch from T to T_x labeled by x
  Return T
```

2

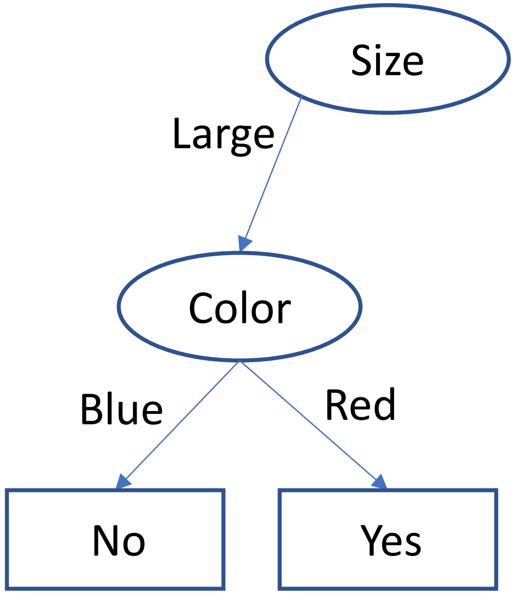
1

Recursively call the ID3 algorithm (as if this is a brand new dataset)

Color: Blue, Red

Size: Large, Small

Size	Color	Purchase?
Large	Red	Yes
Large	Red	Yes
Large	Red	No
Small	Red	No
Small	Blue	No
Small	Blue	No



D_x is not empty when x = “Red”

$$T_x = ID3(D_{x=\text{“Red”}}, \emptyset)$$

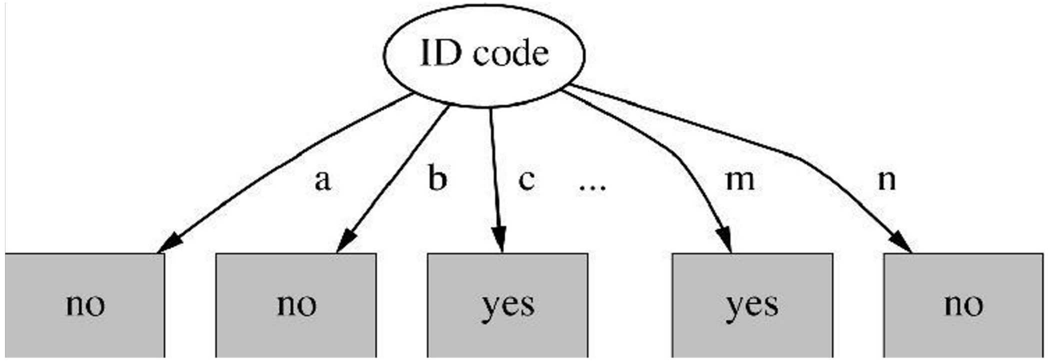
Exercise

Fever	Cough	Breathing Issues	Infected
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	Yes	Yes
Yes	Yes	Yes	Yes
No	Yes	No	No
Yes	No	Yes	Yes
Yes	No	Yes	Yes
No	Yes	Yes	Yes
Yes	Yes	No	Yes
No	Yes	No	No
No	Yes	Yes	Yes
No	Yes	Yes	No
Yes	Yes	No	No

Construct a decision tree for this COVID-19 infection dataset

Suppose we have an ID column

$$\text{Gain}(D, \text{“ID”})$$
$$= 0.971 - 14 * \frac{1}{14}\left(-\frac{1}{14} * \log_2 \frac{1}{14}\right)$$
$$= 0.971 - 0.272$$
$$= 0.699$$



Useless: a new instance with ID=“p”?

Information gain favours attributes with a larger number of possible values (V)

ID	Outlook	Temperature	Humidity	Windy	Play?
a	sunny	hot	high	false	No
b	sunny	hot	high	true	No
c	overcast	hot	high	false	Yes
d	rain	mild	high	false	Yes
e	rain	cool	normal	false	Yes
f	rain	cool	normal	true	No
g	overcast	cool	normal	true	Yes
h	sunny	mild	high	false	No
i	sunny	cool	normal	false	Yes
j	rain	mild	normal	false	Yes
k	sunny	mild	normal	true	Yes
l	overcast	mild	high	true	Yes
m	overcast	hot	normal	false	Yes
n	rain	mild	high	true	No

An Alternative: Information Gain Ratio

- For attribute a, it has V possible values.
  E.g., a="outlook", $V = 3$

- If we divide the data using a, the information gain ratio is:

$$\mathrm{Gain\_ratio}(D, a) = \frac{\mathrm{Gain}(D, a)}{\mathrm{IV}(a)}$$

$$\mathrm{Gain}(D, a) = \mathrm{Ent}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} \mathrm{Ent}(D^v)$$

D^v: dataset that has value of v in D

$$\mathrm{IV}(a) = -\sum_{v=1}^{V} \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|}$$

$\mathrm{IV}(a)$: intrinsic value of attribute a

An Alternative: Information Gain Ratio

$$\text{Gain\_ratio}(D, a) = \frac{\text{Gain}(D, a)}{\text{IV}(a)}$$

$$\text{Gain}(D, a) = \text{Ent}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} \text{Ent}(D^v)$$

$$\text{IV}(a) = -\sum_{v=1}^{V} \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|}$$

Example: Compute $\text{Gain\_ratio}(D, \text{“ID”})$

$\text{Gain}(D, \text{“ID”}) = 0.699 \quad \text{IV}(\text{“ID”}) = 3.8073$

$$\text{Gain\_ratio}(D, \text{“ID”}) = 0.1836$$

Information gain ratio favours attributes with a smaller number of possible values (V)

ID	Outlook	Temperature	Humidity	Windy	Play?
a	sunny	hot	high	false	No
b	sunny	hot	high	true	No
c	overcast	hot	high	false	Yes
d	rain	mild	high	false	Yes
e	rain	cool	normal	false	Yes
f	rain	cool	normal	true	No
g	overcast	cool	normal	true	Yes
h	sunny	mild	high	false	No
i	sunny	cool	normal	false	Yes
j	rain	mild	normal	false	Yes
k	sunny	mild	normal	true	Yes
l	overcast	mild	high	true	Yes
m	overcast	hot	normal	false	Yes
n	rain	mild	high	true	No

Continuous Attributes in Decision Tree Algorithm

Outlook	Temperature	Humidity	Windy	Play?
sunny	34	high	false	No
sunny	33	high	true	No
overcast	31	high	false	Yes
rain	28	high	false	Yes
rain	24	normal	false	Yes
rain	23	normal	true	No
overcast	25	normal	true	Yes
sunny	27	high	false	No
sunny	25	normal	false	Yes
rain	27	normal	false	Yes
sunny	28	normal	true	Yes
overcast	28	high	true	Yes
overcast	29	normal	false	Yes
rain	27	high	true	No

Continuous attributes (features) is common in real datasets.

Number of possible values of a continuous attribute is infinite.

A simplest approach: discretization

E.g., divide into ranges:

“hot”: $T \geq 29$

“mild”: $29 > T \geq 25$

“cool”: $T > 25$

Can algorithm automatically do this?

Bi-Partition for Continuous Attributes in Decision Tree

Idea: divide the dataset D into two parts by threshold t for the continuous attribute “a”:

D_t^+: data samples which has the value of “a” greater than t;

D_t^-: data samples which has the value of “a” less than t.

The temperatures:

Temperature	34	33	31	28	24	23	25	27	25	27	28	28	29	27
Play?	No	No	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes	No

1. Sort the unique values ascendingly: $\{a^1, a^2, \ldots, a^n\}$

E.g., {23, 24, 25, 27, 28, 29, 31, 33, 34}

What are the possible thresholds? Thresholds between two values have the same effect

2. Consider the midpoint of the intervals:

$$T_a = \left\{\frac{a^i + a^{i+1}}{2} \mid 1 \leq i \leq n-1\right\}$$

E.g., $T_{\text{Temperature}} = \{23.5, 24.5, 26, 27.5, 28.5, 30, 32, 33.5\}$

Bi-Partition for Continuous Attributes in Decision Tree

The temperatures:

Temperature	34	33	31	28	24	23	25	27	25	27	28	28	29	27
Play?	No	No	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes	No

3. Compute the information gain by:

for continuous attributes

$$\mathrm{Gain}(D,a) = \max_{t \in \mathrm{T}_a} \mathrm{Gain}(D,a,t) = \boxed{\max_{t \in \mathrm{T}_a} \left[\mathrm{Ent}(D) - \sum_{\lambda \in \{-,+\}} \frac{|D_t^\lambda|}{|D|} \mathrm{Ent}(D_t^\lambda) \right]}$$

$$T_{\mathrm{Temperature}} = \{23.5, 24.5, 26, 27.5, 28.5, 30, 32, 33.5\} \qquad \mathrm{Ent}(D) = 0.94$$

1) t = 23.5: D_t^+ = {No, No, Yes, Yes, Yes, Yes, No, Yes, Yes, Yes, Yes, Yes, No} $\quad \mathrm{Ent}(D_t^+) = 0.891$

D_t^- = {No} $\quad \mathrm{Ent}(D_t^-) = 0$

$$\mathrm{Gain}(D, \text{“Temperature”}, 23.5) = 0.94 - \frac{13}{14} * 0.89 - 0 \approx 0.113$$

Bi-Partition for Continuous Attributes in Decision Tree

The temperatures:

Temperature	34	33	31	28	24	23	25	27	25	27	28	28	29	27
Play?	No	No	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes	No

for continuous attributes

3. Compute the information gain by:

$$\mathrm{Gain}(D,a)=\max_{t\in \mathrm{T}_a}\mathrm{Gain}(D,a,t)=\boxed{\max_{t\in \mathrm{T}_a}\left[\mathrm{Ent}(D)-\sum_{\lambda\in\{-,+\}}\frac{|D_t^\lambda|}{|D|}\mathrm{Ent}(D_t^\lambda)\right]}$$

$$T_{\mathrm{Temperature}}=\{23.5,24.5,26,27.5,28.5,30,32,33.5\} \qquad \mathrm{Ent}(D)=0.94$$

2) t = 24.5: D_t^+ = {No, No, Yes, Yes, Yes, No, Yes, Yes, Yes, Yes, Yes, No} $\quad \mathrm{Ent}(D_t^+)=0.918$

D_t^- = {Yes, No} $\quad \mathrm{Ent}(D_t^-)=1$

$$\mathrm{Gain}(D,\text{“Temperature”},23.5)=0.94-\frac{12}{14}*0.92-\frac{2}{14}*1\approx 0.01$$

Bi-Partition for Continuous Attributes in Decision Tree

The temperatures:

Temperature	34	33	31	28	24	23	25	27	25	27	28	28	29	27
Play?	No	No	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes	No

3. Compute the information gain by:

for continuous attributes

$$\mathrm{Gain}(D,a) = \max_{t\in \mathrm{T}_a} \mathrm{Gain}(D,a,t) = \boxed{\max_{t\in \mathrm{T}_a} \left[\mathrm{Ent}(D) - \sum_{\lambda\in\{-,+\}} \frac{|D_t^\lambda|}{|D|}\mathrm{Ent}(D_t^\lambda)\right]}$$

$$T_{\mathrm{Temperature}} = \{23.5, 24.5, 26, 27.5, 28.5, 30, 32, 33.5\} \qquad \mathrm{Ent}(D) = 0.94$$

Compute for all possible thresholds:

$$Gain(D, \text{“Temperature”}) = 0.245$$

$$\mathrm{Gain}(D, \text{“Temperature”}, 23.5) \approx 0.113$$
$$\mathrm{Gain}(D, \text{“Temperature”}, 24.5) \approx 0.01$$
$$\mathrm{Gain}(D, \text{“Temperature”}, 26) \approx 0.015$$
$$\mathrm{Gain}(D, \text{“Temperature”}, 27.5) \approx 0.016$$
$$\mathrm{Gain}(D, \text{“Temperature”}, 28.5) \approx 0.025$$
$$\mathrm{Gain}(D, \text{“Temperature”}, 30) \approx 0.079$$
$$\boxed{\mathrm{Gain}(D, \text{“Temperature”}, 32) \approx 0.245}$$
$$\mathrm{Gain}(D, \text{“Temperature”}, 33.5) \approx 0.113$$

threshold: $t = 32$

Bi-Partition for Continuous Attributes in Decision Tree

The temperatures:

Temperature	34	33	31	28	24	23	25	27	25	27	28	28	29	27
Play?	No	No	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes	No

3. Compute the information gain by:

for continuous attributes

$$\mathrm{Gain}(D,a) = \max_{t \in \mathrm{T}_a} \mathrm{Gain}(D,a,t) = \boxed{\max_{t \in \mathrm{T}_a} \left[\mathrm{Ent}(D) - \sum_{\lambda \in \{-,+\}} \frac{|D_t^\lambda|}{|D|} \mathrm{Ent}(D_t^\lambda) \right]}$$

$$T_{\text{Temperature}} = \{23.5, 24.5, 26, 27.5, 28.5, 30, 32, 33.5\} \qquad \mathrm{Ent}(D) = 0.94$$

Compute for all possible thresholds:

$$Gain(D, \text{“Temperature”}) = 0.245$$

$$\mathrm{Gain}(D, \text{“Temperature”}, 23.5) \approx 0.113 \qquad \mathrm{Gain}(D, \text{“Temperature”}, 28.5) \approx 0.025$$

$$\mathrm{Gain}(D, \text{“Temperature”}, 24.5) \approx 0.01 \qquad \mathrm{Gain}(D, \text{“Temperature”}, 30) \approx 0.079$$

$$\mathrm{Gain}(D, \text{“Temperature”}, 26) \approx 0.015 \qquad \boxed{\mathrm{Gain}(D, \text{“Temperature”}, 32) \approx 0.245}$$

$$\mathrm{Gain}(D, \text{“Temperature”}, 27.5) \approx 0.016 \qquad \mathrm{Gain}(D, \text{“Temperature”}, 33.5) \approx 0.113$$

threshold:

$t = 32$

Representative Decision Tree Algorithms

- **ID3** (Quinlan, 1979, 1986)
  - Uses Information gain to select attributes.
- **C4.5** (Quinlan, 1993)
  - Uses information gain ratio to select attributes.
  - First find attributes having information gain above average, then select the one with highest information gain ratio.
  - Handles continuous attributes.
  - Does pruning
- **CART** (Breiman et al., 1984)
  - Can do regression as well.

Quinlan, J.R. (1979) **Discovering Rules by Induction** from **Large Collections of Examples**. *Expert Systems in the Micro Electronic Age*.
Quinlan, J.R. (1986) **Induction of decision trees**. *Machine Learning*.
Quinlan, J.R. (1993) *C4.5: Programs for Machine Learning.*
Breiman et al. (1984) *Classification and Regression Trees*.

Generalization and Model Selection

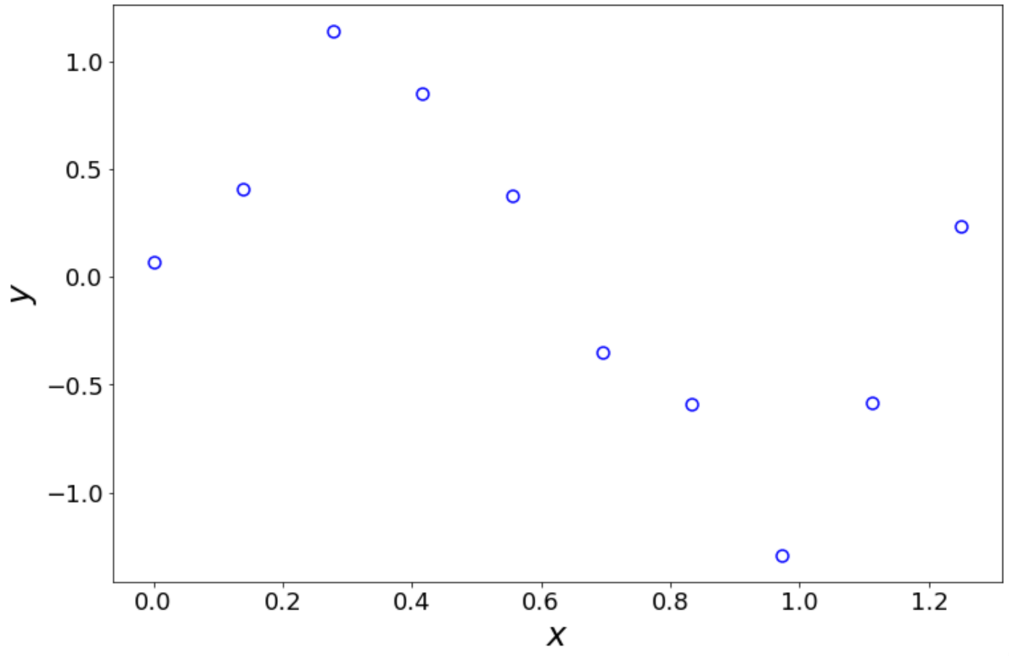
Example: Regression

Let's consider a simple polynomial extension of linear regression:

Linear regression: $y = wx + b$

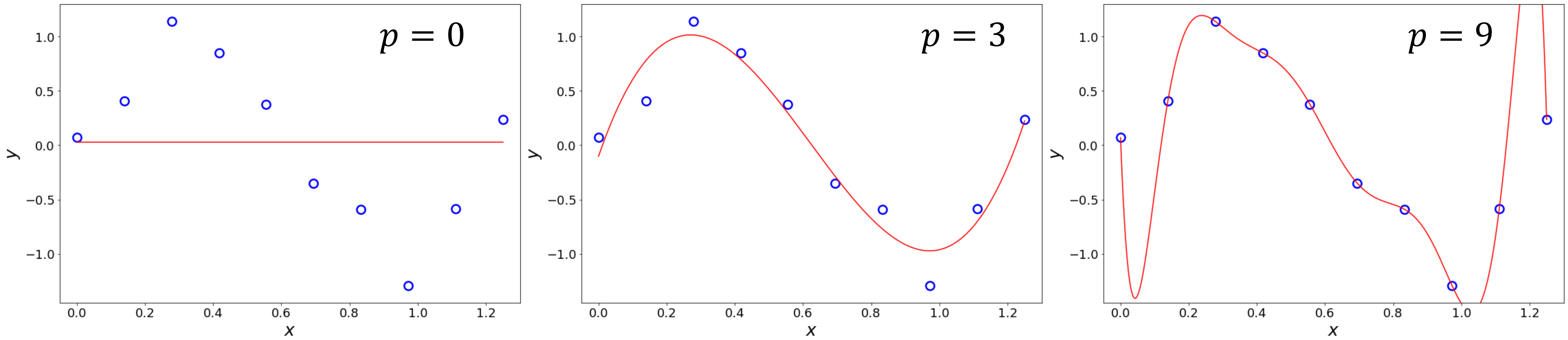
p-order polynomial regression:

$$y = w_1 x + w_2 x^2 + \cdots + w_p x^p + b$$



Example: Regression

Let's consider a simple polynomial extension of linear regression:



MSE = 0.487 MSE = 0.035 MSE = 0

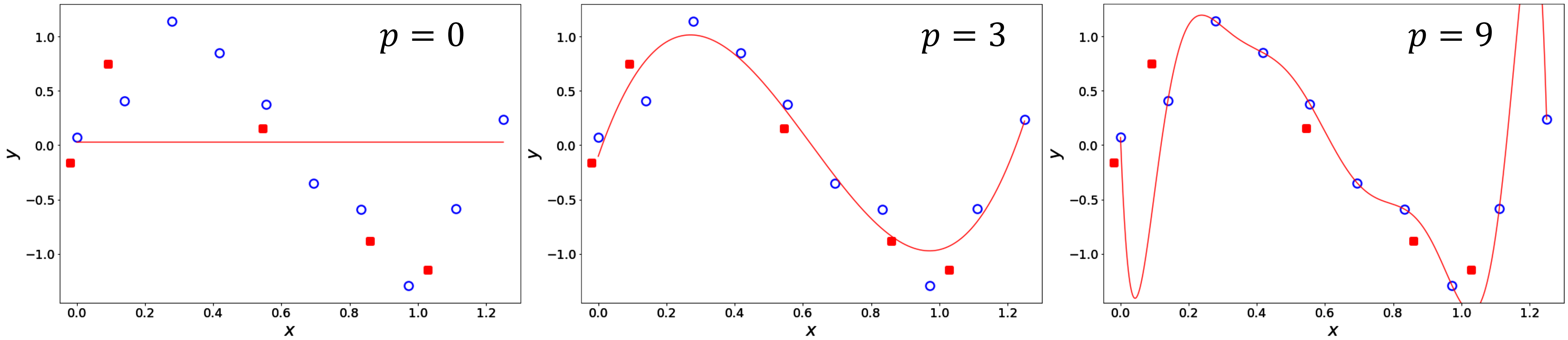
Which is a better model? It seems $p = 9$ is the best: the lowest MSE.

Hold on… What is the goal of doing machine learning?

(Intuition) Making predictions for new data!

Example: Regression

Let's add in a few new data points (red squares):



MSE(original data) = 0.487
MSE(new data) = 0.473

MSE (original data) = 0.035
MSE (new data) = 0.135

MSE (original data) = 0
MSE (new data) = 136.76

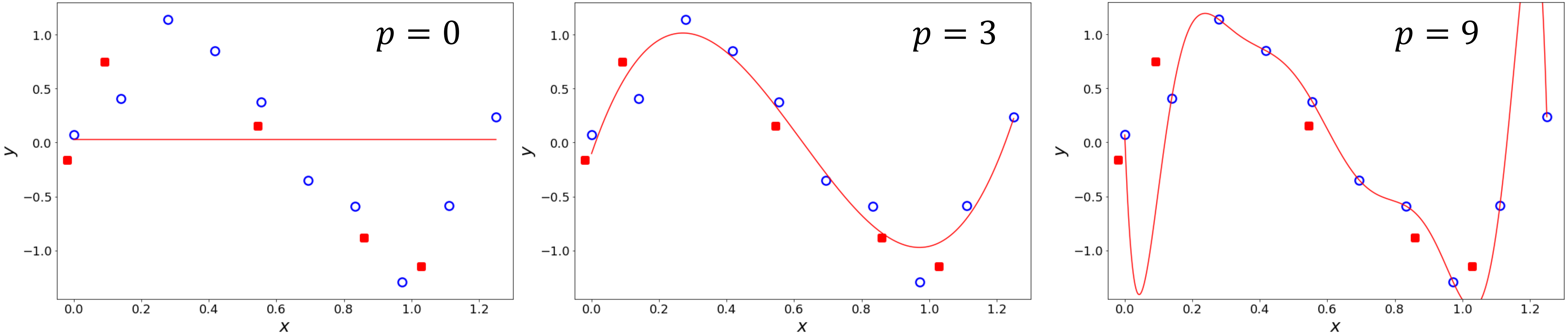
Is $p = 9$ a good model?

Zero loss when fitting the model;

Huge loss when using the model for new data.

Example: Regression

Terminologies: Underfitting and overfitting



MSE(original data) = 0.487	MSE (original data) = 0.035	MSE (original data) = 0
MSE(new data) = 0.473	MSE (new data) = 0.135	MSE (new data) = 136.76

Underfitting

Large error for training & testing

Overfitting

Small training error, large testing error

Terminologies

Generalization

- Generalization: the ability of a machine learning model to adapt to new and previously unseen data. (*a central task in ML*)
- We train a ML model on some data (called training data) and want to apply it to some new data (called testing data).
- Underfitting: when a model has large error in both training data and testing data.
- Overfitting: when a model has small error in training data, but large error in testing data.

Terminologies

Why can we expect good generalization?

- Fundamental assumption in machine learning: data are independent and identically distributed (i.i.d. / IID).
- Intuition: it allows the patterns learned to be applied to new data.

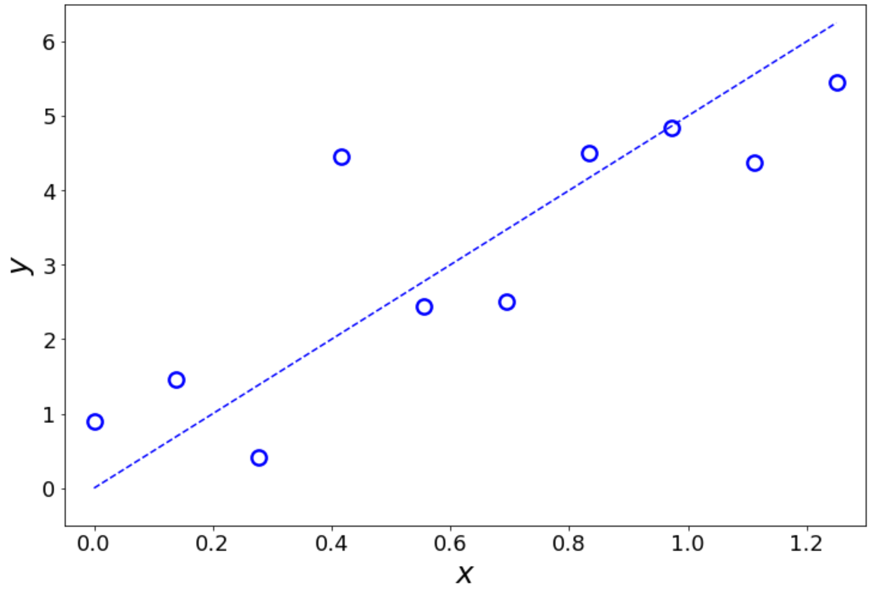
Question: can we train a heart failure prediction model using data collected from elderlies and directly apply that to the young?

No! Elderlies and the young have different distribution in their health conditions.

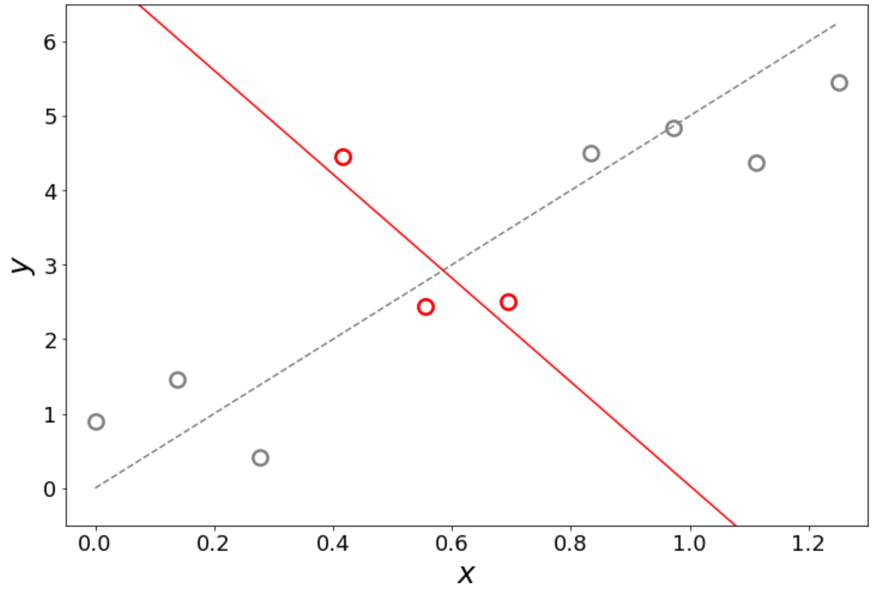
(Advanced methods exist to allow such application,
but direct application could lead to serious outcomes)

Reasons of Poor Generalization

Data is not enough or not representative



Blue line: A well fitted linear model

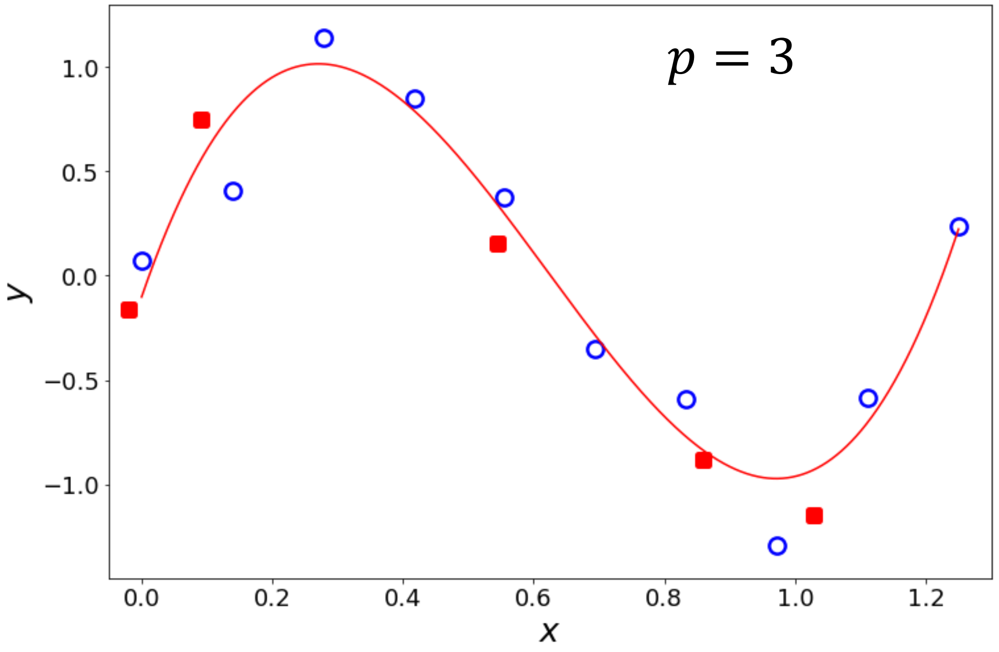


If we only observe three data samples that are not representative

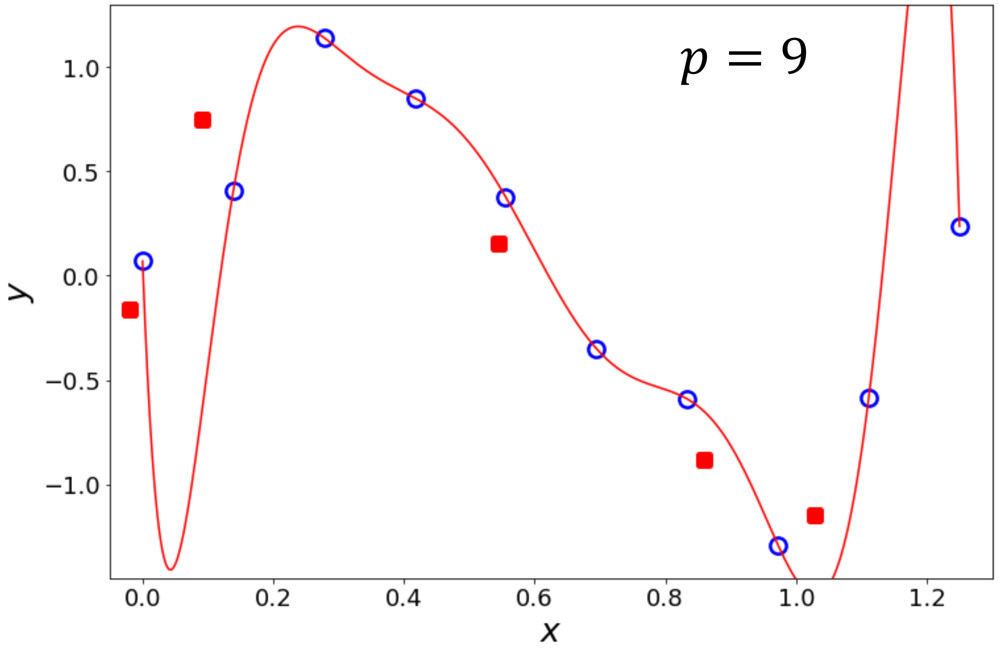
Overfitting to the observed (red) data!

Reasons of Poor Generalization

The model is too complex



Less complex model

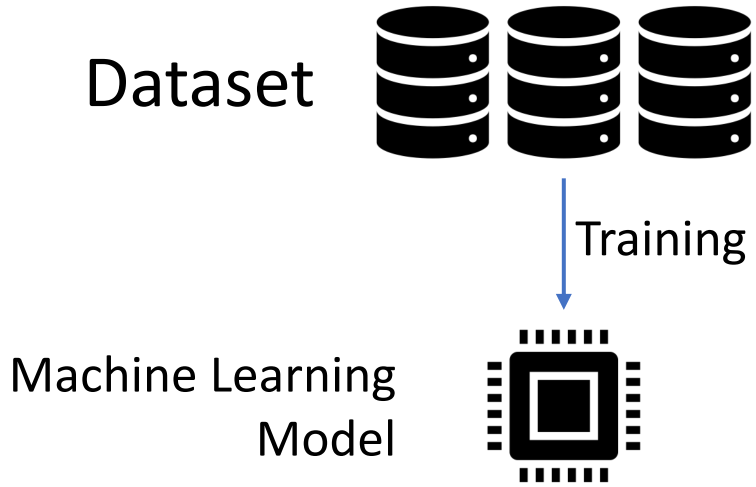


Very complex model

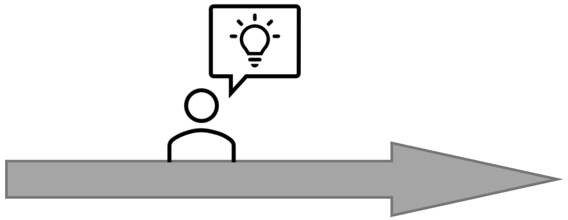
Complex models are more expressive but is more prone to overfitting!

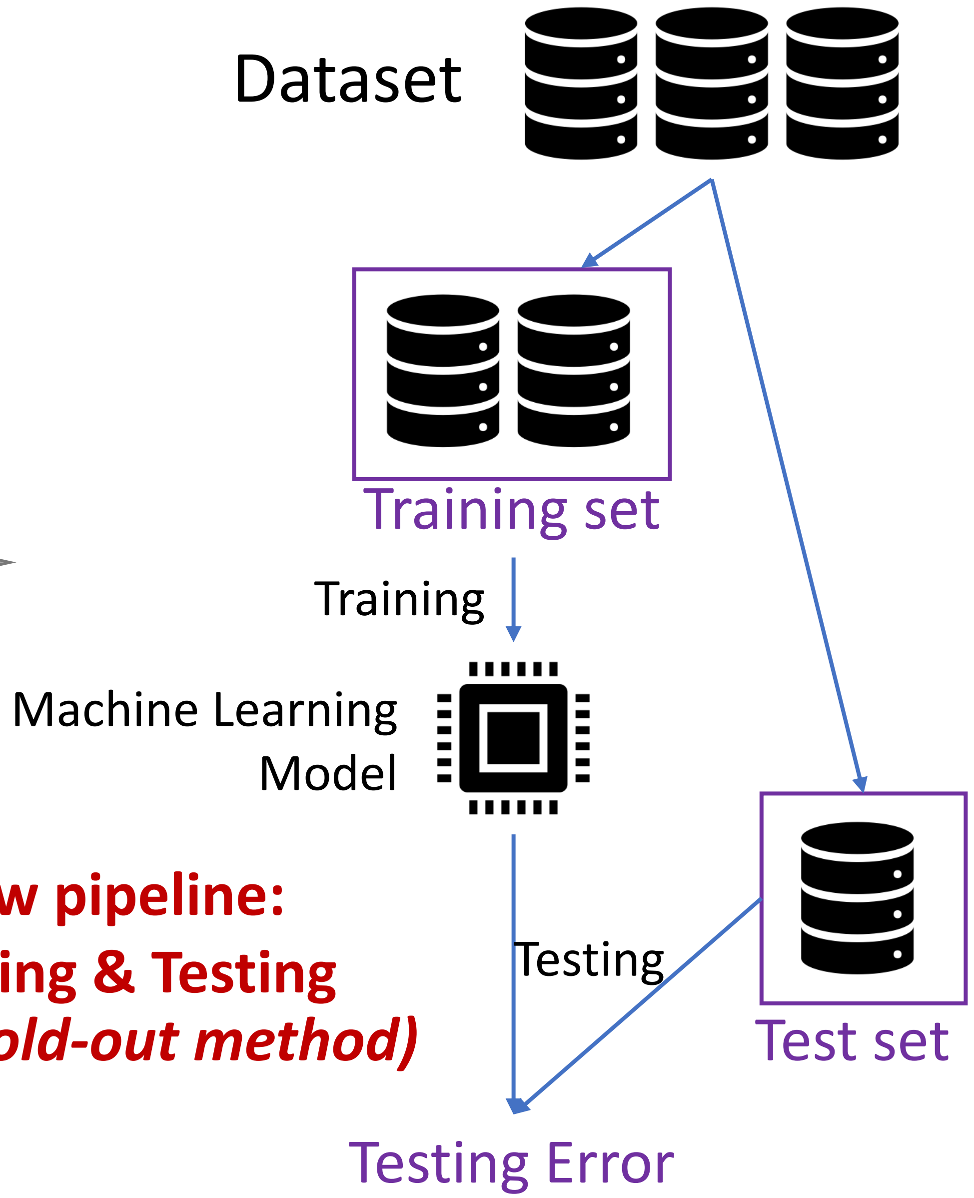
How to Measure Generalization?

(How do we know if the model overfits?)



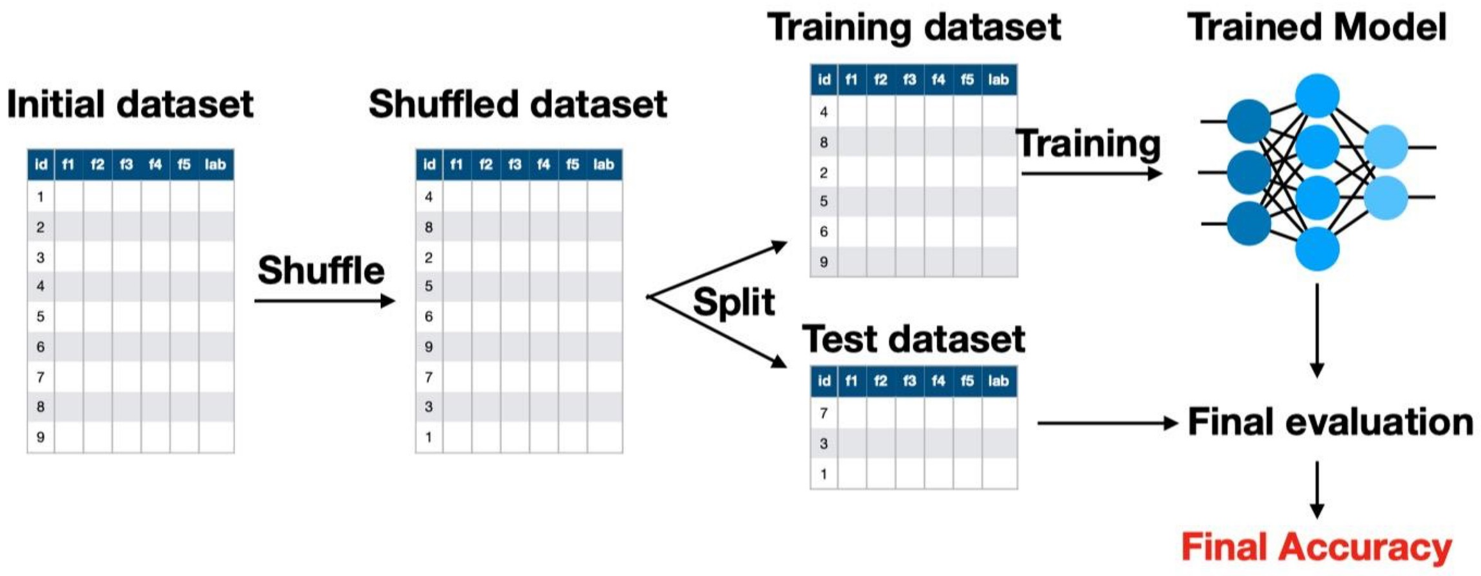
Our current pipeline:
Cannot know how good it generalizes





New pipeline:
Training & Testing
(formally: hold-out method)

Hold-out Method for Performance Evaluation



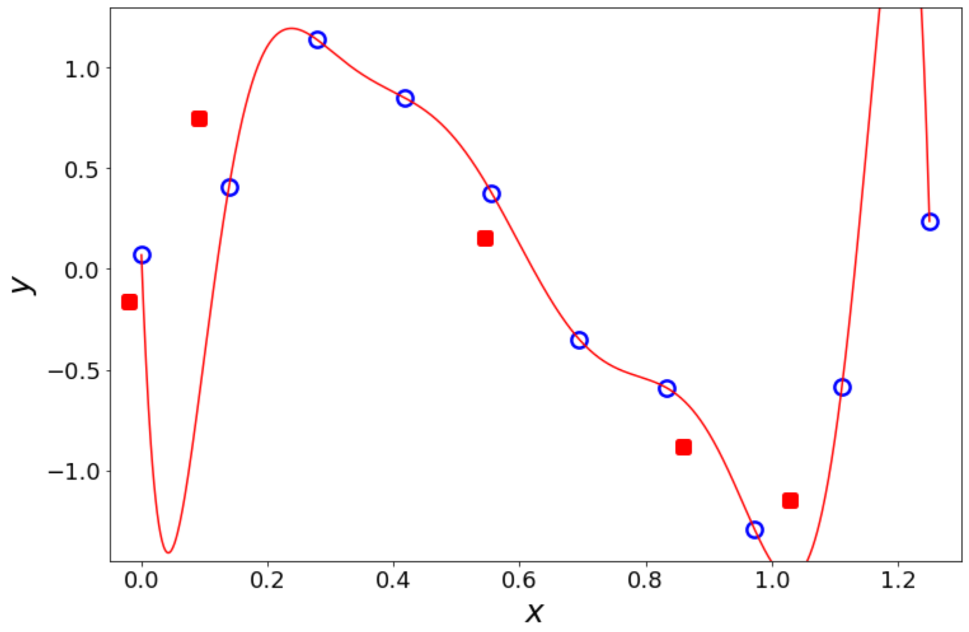
Example of Stratified sampling:

D has 500 positive &
500 negative samples

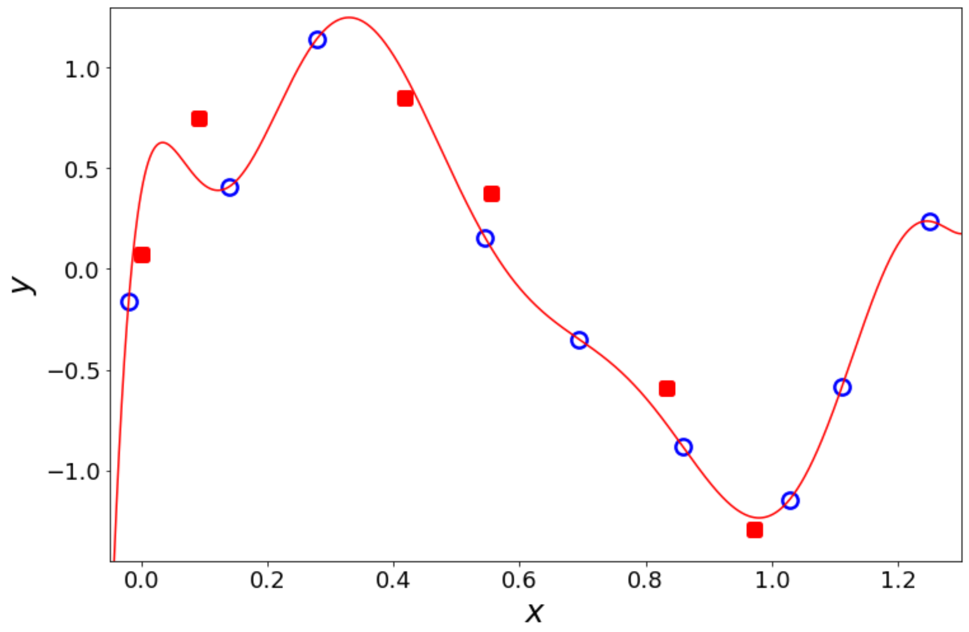
70% for training:
sample 350 positive &
350 negative for D_{train}

- Randomly split data into training and testing sets (e.g., 70% for training and 30% for testing)
- The training/test split should preserve a consistent distribution (use stratified sampling).
- Performance of a model must be evaluated in a held-out testing set.
- The test dataset should NEVER be used to train the model.

Limitations of Hold-out Method



MSE (training) = 0
MSE (testing) = 136.76



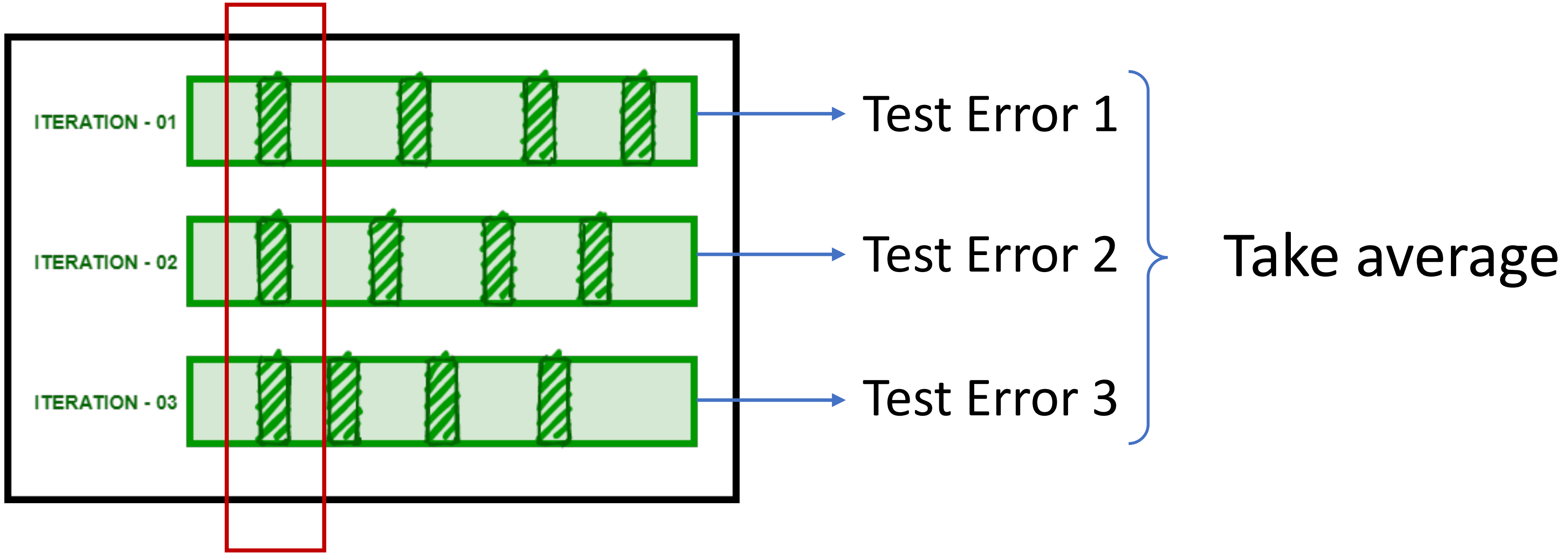
MSE (training) = 0
MSE (test) = 0.0537

Same data points with two different data split:

By chance, we could get small testing error even for a model that overfits in a particular training/test split.

Repeated Hold-out Method for Performance Evaluation

Idea: Repeat the pipeline for K times,
then take average of the performance over the test set.

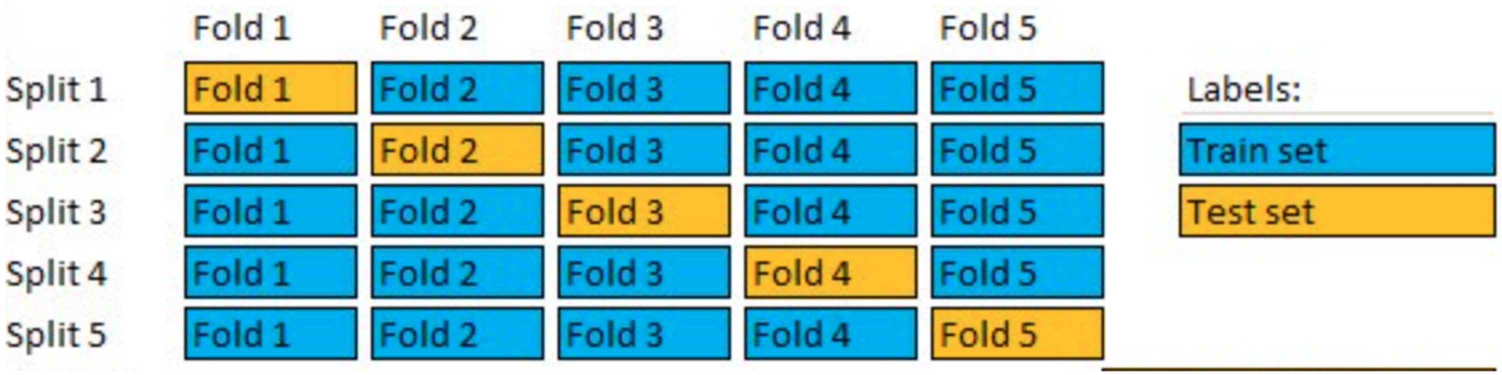


Possible to have overlaps

Even the model overfits to a particular training/test split, it affects little the final average performance.

K-Fold Cross Validation for Performance Evaluation

Idea: Split the data into K subsets of equal size, use one subset for testing and others for training in each iteration.
(commonly used K: 5, 10, 20)



$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

Avoids overlapping test set.

A more systematical way of performance evaluation

A Special Case: Leave-One-Out Method

Idea: When $K = |D_{train}|$ for the K-fold cross validation, we call it leave-one-out method. (each subset only contains one sample)

ID	Outlook
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	

ID	Outlook
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	

ID	Outlook
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	

ID	Outlook	Temperature	Humidity	Windy	Play?
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					
28					
29					
30					

Iterations: 1 2 i N

Iteration 1: Train model with $N-1$ data, compute test error E_1 with the remaining one

Iteration i: Train model with $N-1$ data, compute test error E_i with the remaining one

Final error: $$E = \frac{1}{N}\sum_{i=1}^{N} E_i$$

A Special Case: Leave-One-Out Method

Idea: When $K = N$ (size of D) for the K-fold cross validation, we call it leave-one-out method. (each subset only contains one sample)

- Advantage:
  - Model is trained using $N - 1$ data points, close to that trained using D. *(our goal: evaluating how good the model is trained using D)*
  - Therefore, it is more accurate measurement of the performance.
- Disadvantage:
  - Computationally expensive! Need to train the model for N times. *(N can be greater than 1 million for large datasets)*

How to Prevent Overfitting?

Recall the two major reasons of overfitting:

- Data is not enough or is not representative. → *Collect more data.*
- The model is too complex. → *Control the complexity of the model.*

- Some common methods to control the model complexity

Regularization

$$\min_{\mathbf{w}} \text{TrainLoss}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

Algorithm: gradient descent

Initialize $\mathbf{w} = [0, \ldots, 0]$

For $t = 1, \ldots, T$:

$\quad \mathbf{w} \leftarrow \mathbf{w} - \eta(\nabla_{\mathbf{w}} [\text{TrainLoss}(\mathbf{w})] + \lambda \mathbf{w})$

Early stopping

Use a smaller T

Algorithm: gradient descent

Initialize $\mathbf{w} = [0, \ldots, 0]$

For $t = 1, \ldots, T$:

$\quad \mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})$