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

Logistics

- In-class quiz on Oct. 20 (next week)
- Arrangement for Lecture 7 (Oct. 20):
  - Introduction of Course Project, Q&A (6:30pm – 7:00pm)
  - Quiz (7:10pm – 8:40pm)
  - Review of quiz sample solutions (8:50pm – 9:20pm)
- Lab 2: Machine Learning with scikit-learn on Oct. 15
- Next Office Hour: Oct. 18 (next Monday)
- Course Project Instructions will be posted in Moodle by Oct. 18 (next Monday)

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

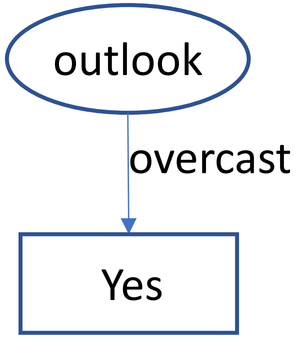
purity before split *purity after split*

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
  $X \leftarrow$ attribute with highest information gain
  Label(T) = X
  For each value x of X
    $\mathbf{D}_x \leftarrow$ instances in **D** with $X = x$
    If $\mathbf{D}_x$ is empty
      Let T_x be a new tree
      Label(T_x) = most common class in **D**
    Else
      T_x = ID3($\mathbf{D}_x$, $\mathbf{X} - \{X\}$)
    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
  $X \leftarrow$ attribute with highest information gain
  Label(T) = X
  For each value x of X
    $\mathbf{D}_x \leftarrow$ instances in **D** with $X = x$
    If $\mathbf{D}_x$ is empty
      Let T_x be a new tree
      Label(T_x) = most common class in **D**
    Else
      $T_x = \text{ID3}(\mathbf{D}_x, \mathbf{X} - \{X\})$
    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 $\text{Gain}(D, \text{“Color”})$

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

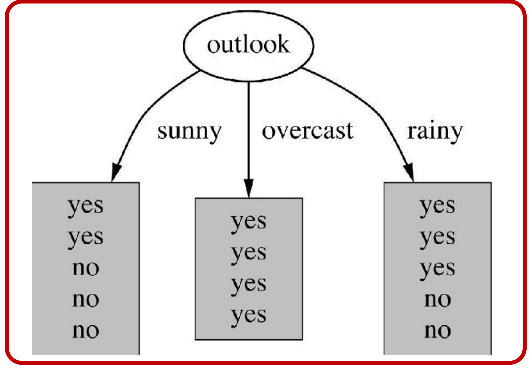
Do we need to split D in ID3 algorithm?

No: no attribute has positive information gain

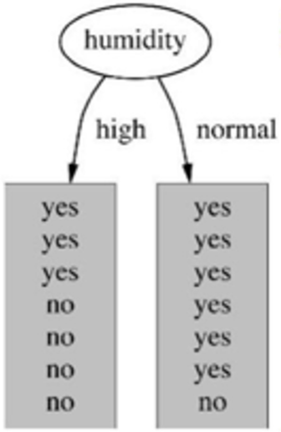
Recap: ID3 Algorithm

```
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
```

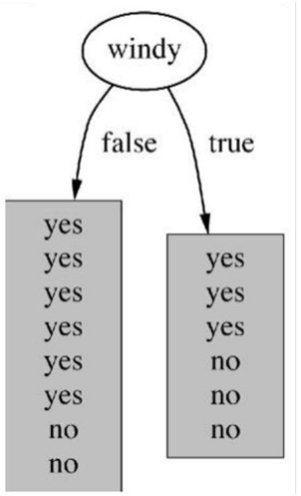
highest



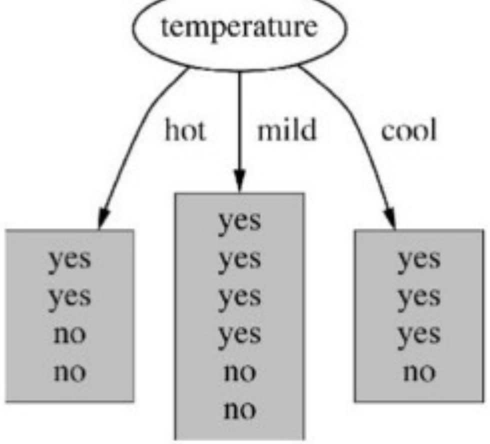
$\text{Gain}(D, \text{“outlook”}) = 0.246$



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



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



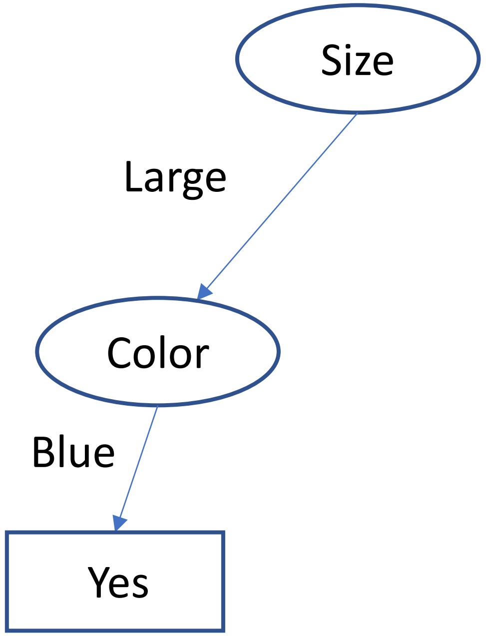
$\text{Gain}(D, \text{“temperature”}) = 0.029$

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 Options: Blue, Red

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



D_x is empty when x = “Blue”

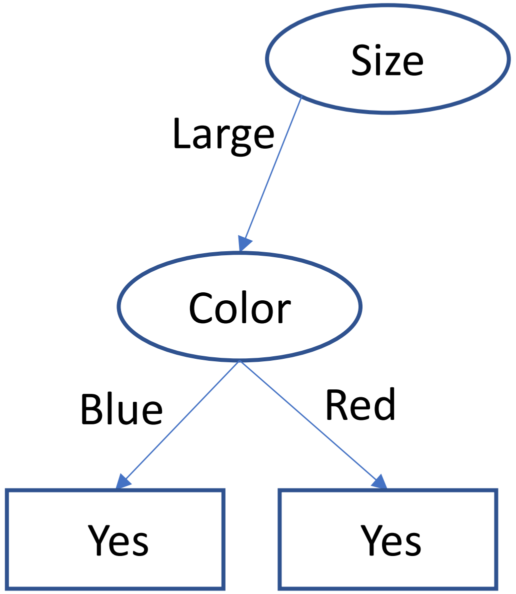
Recap: ID3 Algorithm

ID3(**D**,**X**) =
  Let T be a new tree
  If all instances in **D** have same class c
    Label(T) = c; Return T
2 If **X** = ∅ or no attribute has positive information gain
    Label(T) = most common class in **D**; return T
  $X \leftarrow$ attribute with highest information gain
  Label(T) = X
  For each value x of X
    $\mathbf{D}_x \leftarrow$ instances in **D** with $X = x$
    If $\mathbf{D}_x$ is empty
      Let T_x be a new tree
      Label(T_x) = most common class in **D**
    Else
1     $T_x = \text{ID3}(\mathbf{D}_x, \mathbf{X} - \{X\})$
    Add a branch from T to T_x labeled by x
  Return T

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

Color: Blue, Red
Size: Large, Small

		Purchase?
		Yes
		Yes
		No



D_{x} is not empty when x = "Red"

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

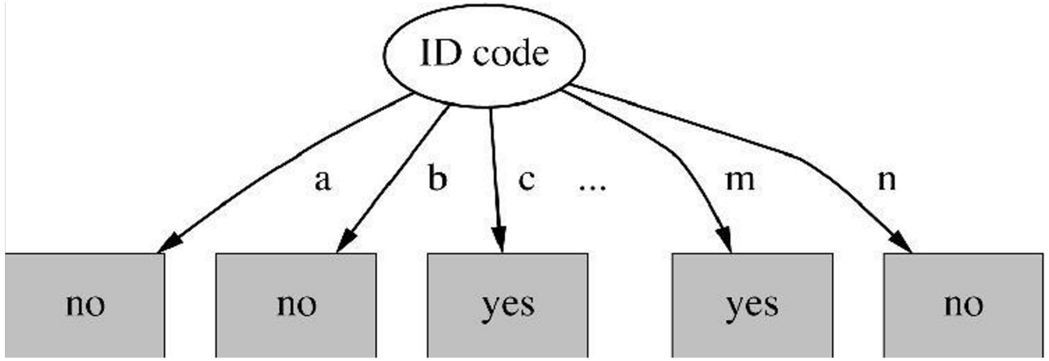
Exercise

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

$$
\begin{aligned}
&\text{Gain}(D, \text{“ID”}) \\
&= 0.971 - 14 * \frac{1}{14}(-1 * \log_2 1) \\
&= 0.971 - 0 \\
&= 0.971
\end{aligned}
$$



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:

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

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

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

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

$\text{IV}(a)$: 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 <u>into two parts</u> 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:

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

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

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

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

1) t = 23.5:

D_t^+ = {No, No, Yes, Yes, Yes, Yes, No, Yes, Yes, Yes, Yes, Yes, No} $\mathrm{Ent}(D_t^+) = 0.891$

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

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

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

3. Compute the information gain by:

for continuous attributes

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

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

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

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

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

Compute for all possible thresholds:

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

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

threshold: $t = 32$

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

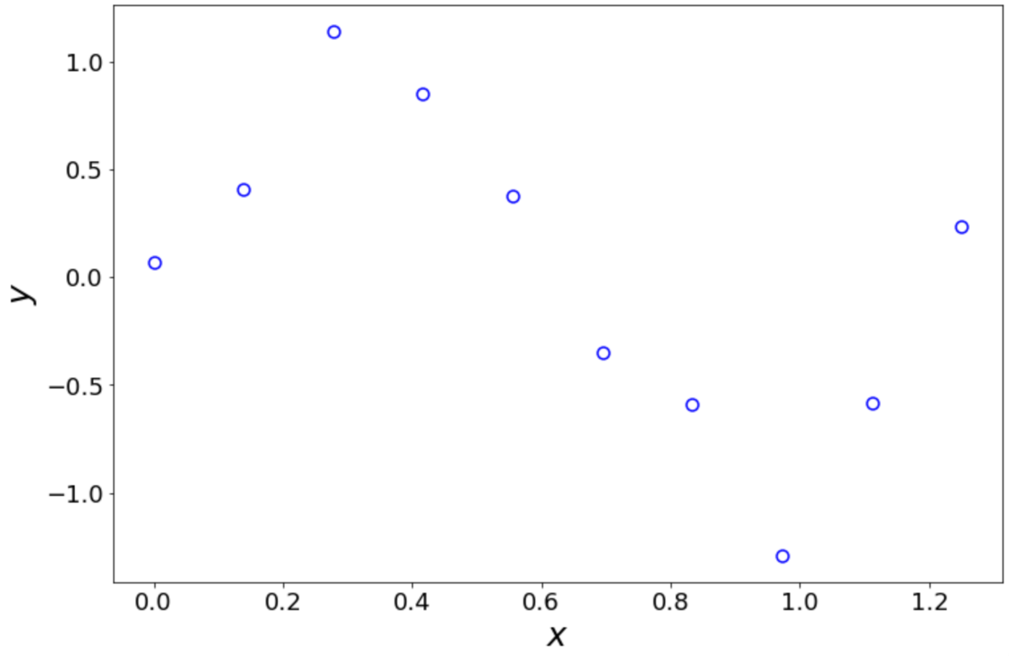
Example: Regression

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

Linear regression: $y = wx + b$

p-order polynomial regression:

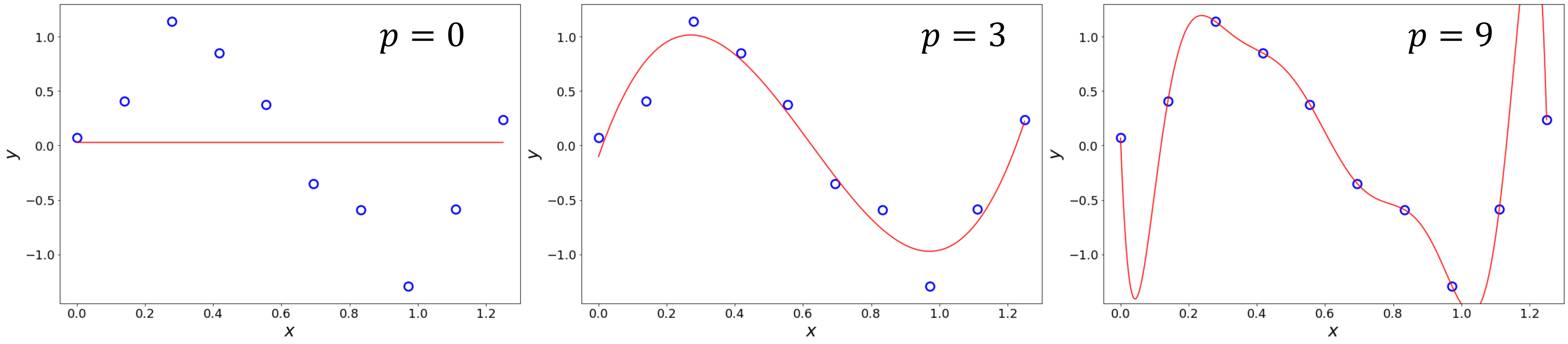
$$y = w_1 x + w_2 x^2 + \cdots + w_p x^p + b$$



some data points to fit the model

Example: Regression

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



MSE = 0.487 MSE = 0.035 MSE = 0

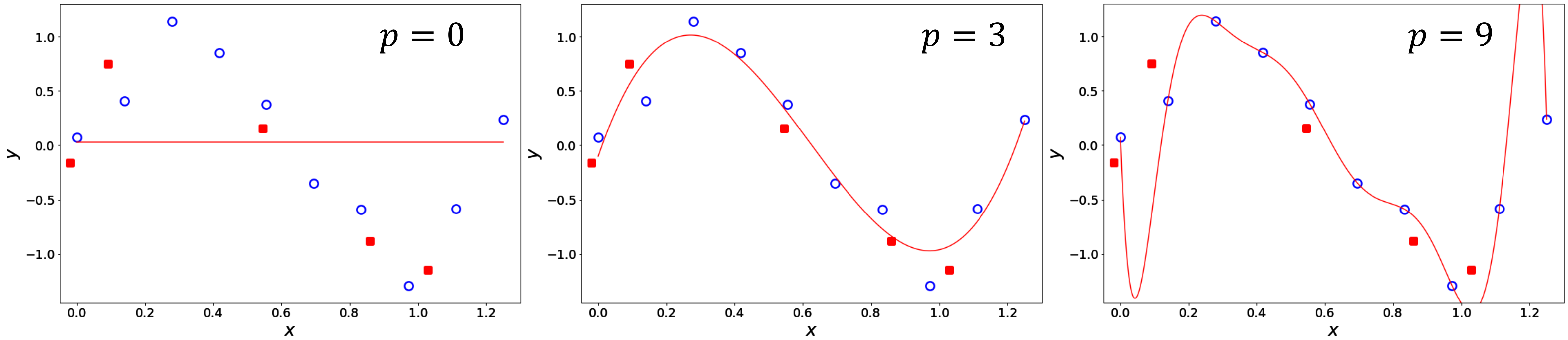
Which is a better model? It seems $p = 9$ is the best: the lowest MSE.

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

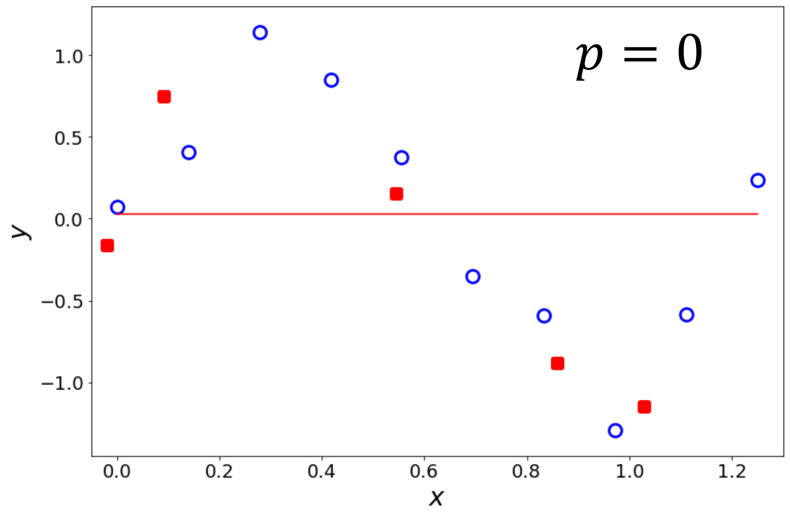
Is $p = 9$ a good model?

Zero loss when fitting the model;

Huge loss when using the model for new data.

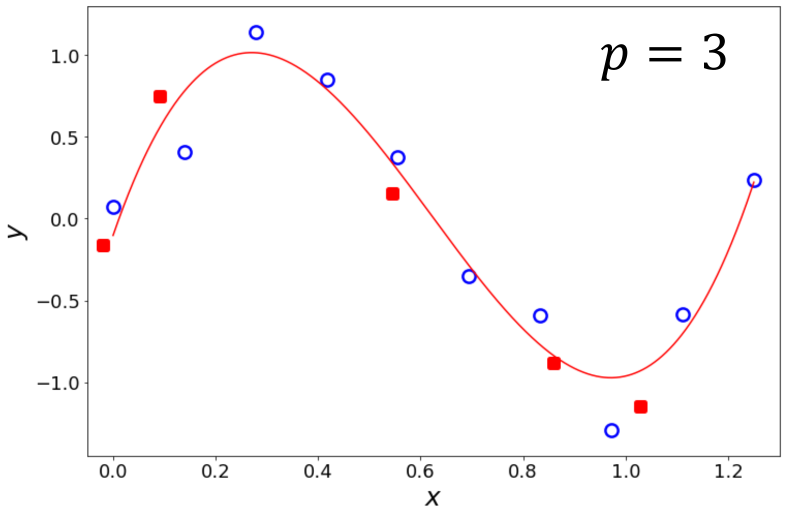
Example: Regression

Terminologies: Underfitting and overfitting



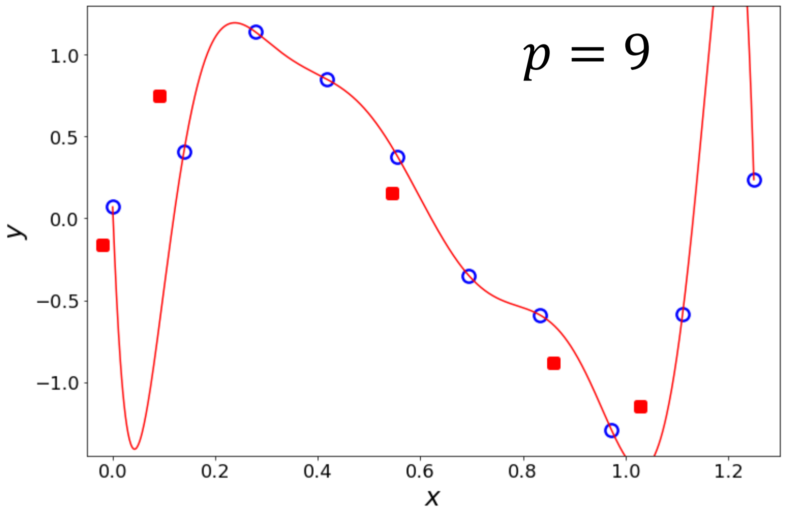
MSE(original data) = 0.487

MSE(new data) = 0.473



MSE (original data) = 0.035

MSE (new data) = 0.135



MSE (original data) = 0

MSE (new data) = 136.76

Underfitting

Large error for training & testing

Overfitting

Small training error, large testing error

Terminologies

- 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.

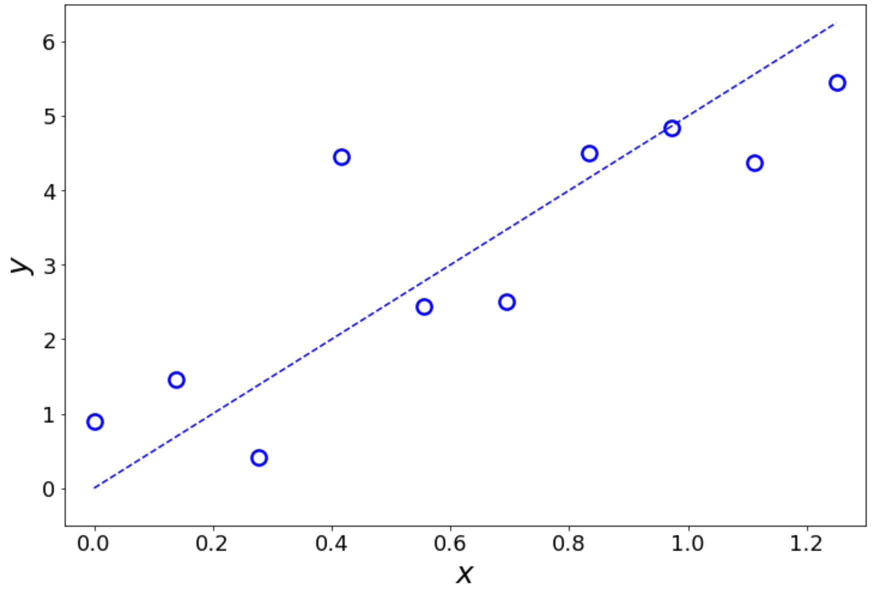
Question: can we train a heart failure prediction model using data collected from elderlies and directly apply that to the young?

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

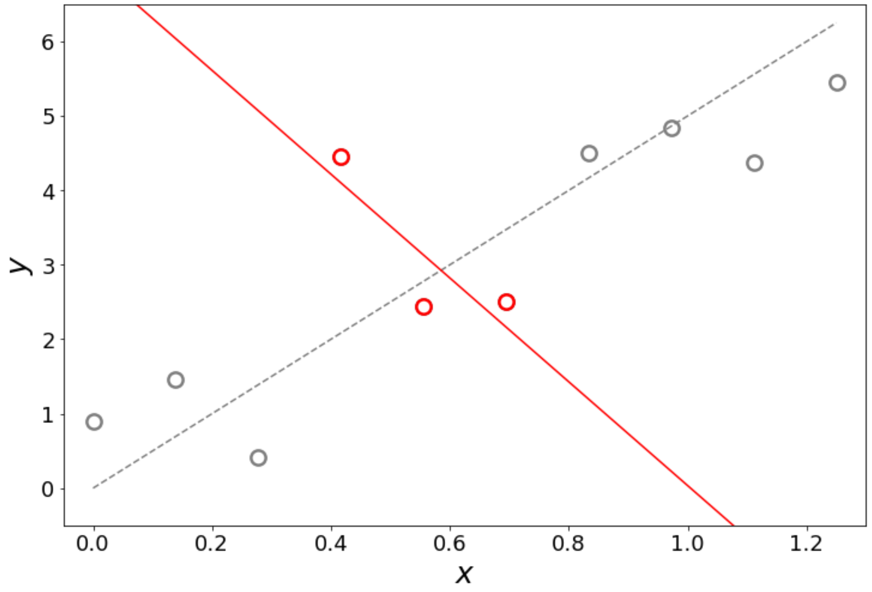
(Advanced methods exist to allow such application,
but direct application could lead to serious outcomes)

Reasons of Poor Generalization

Data is not enough or not representative



Blue line: A well fitted linear model

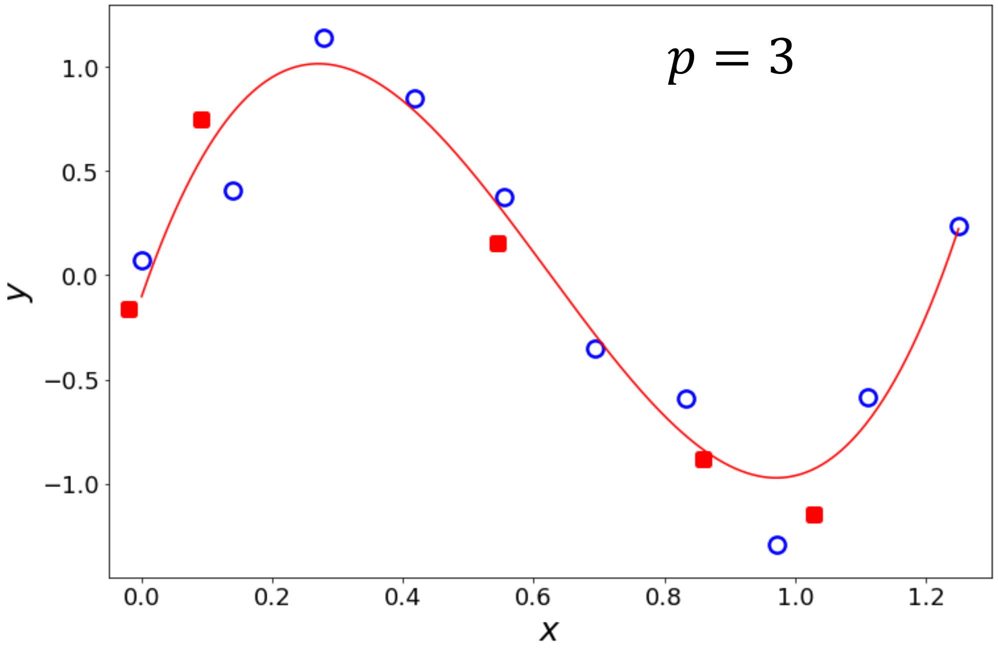


If we only observe three data samples that are not representative

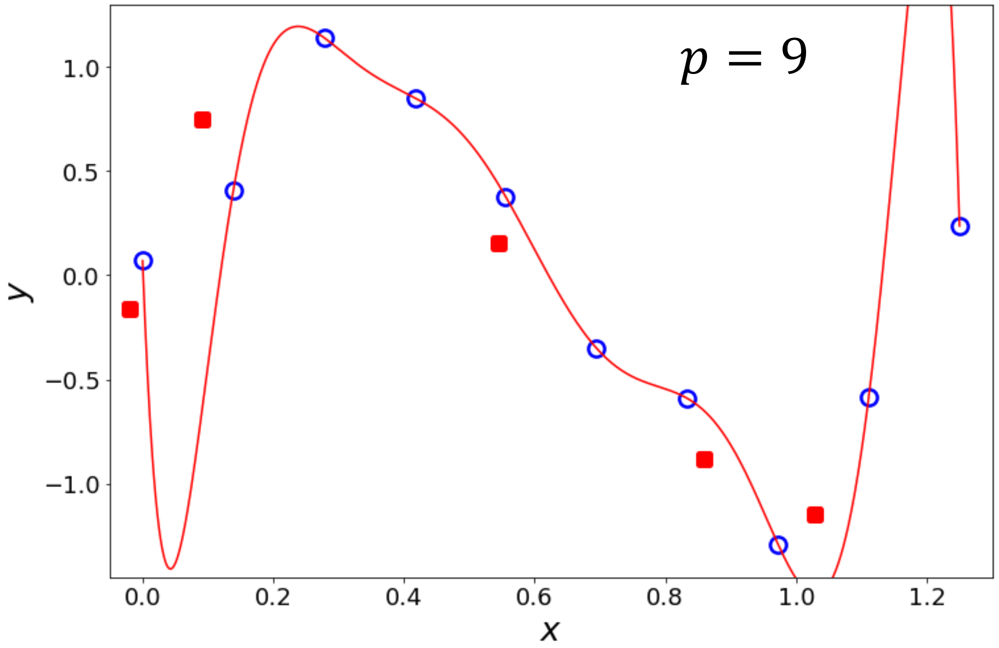
Overfitting to the observed (red) data!

Reasons of Poor Generalization

The model is too complex



Less complex model

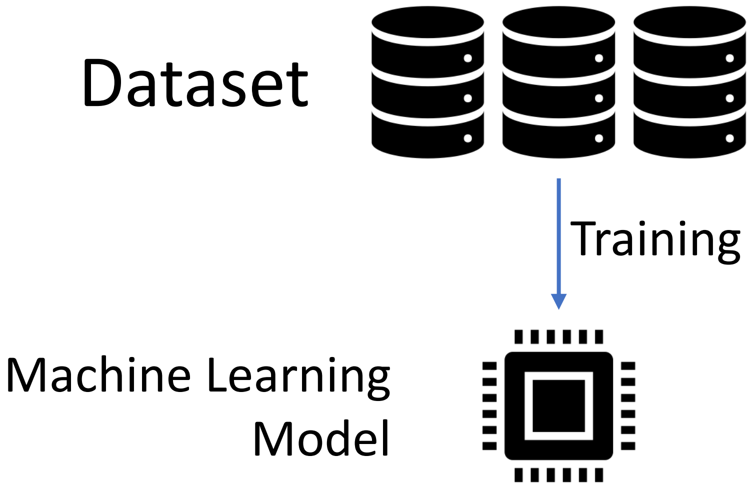


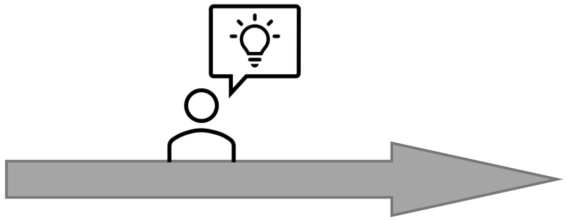
Very complex model

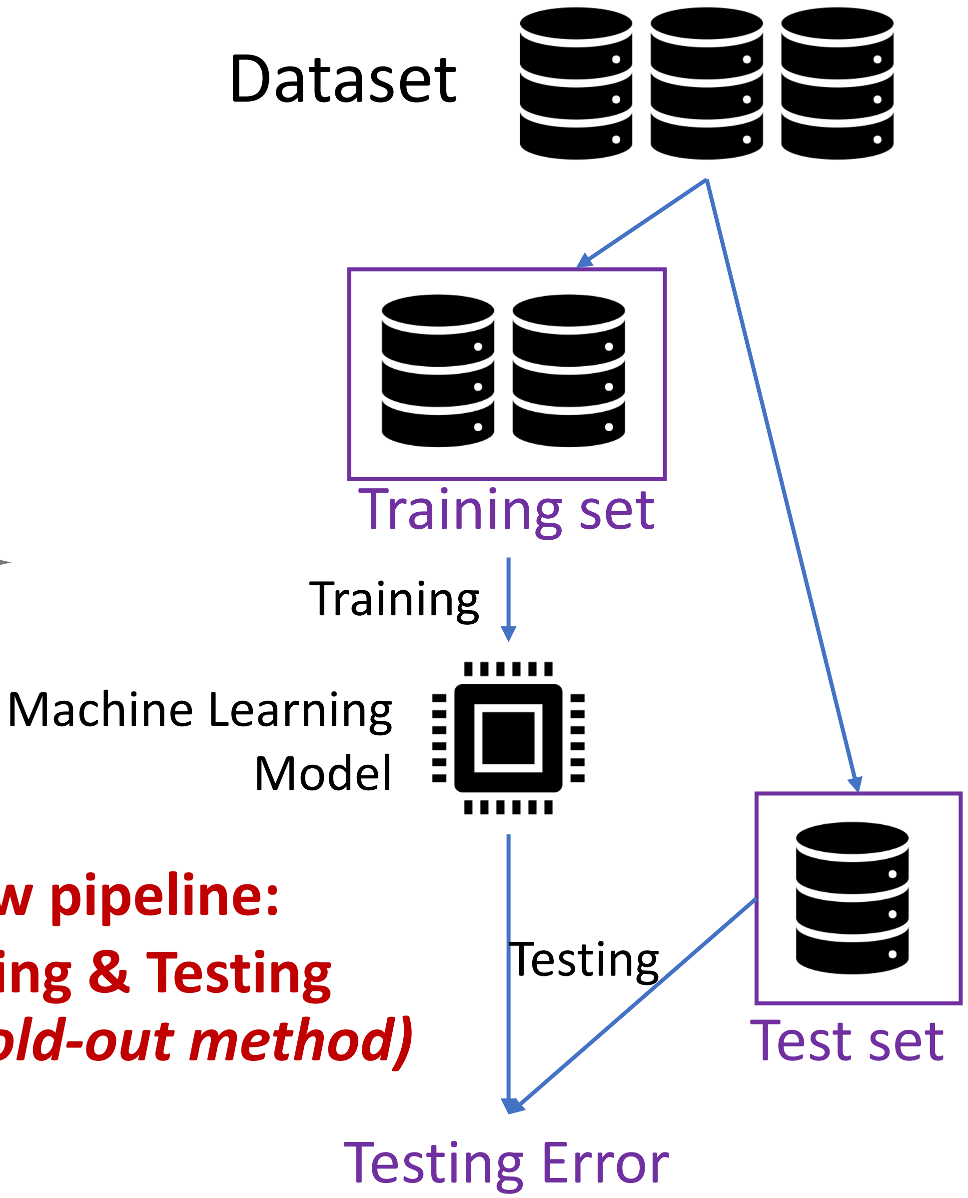
Complex models are more expressive but is more prone to overfitting!

How to Measure Generalization?

(How do we know if the model overfits?)



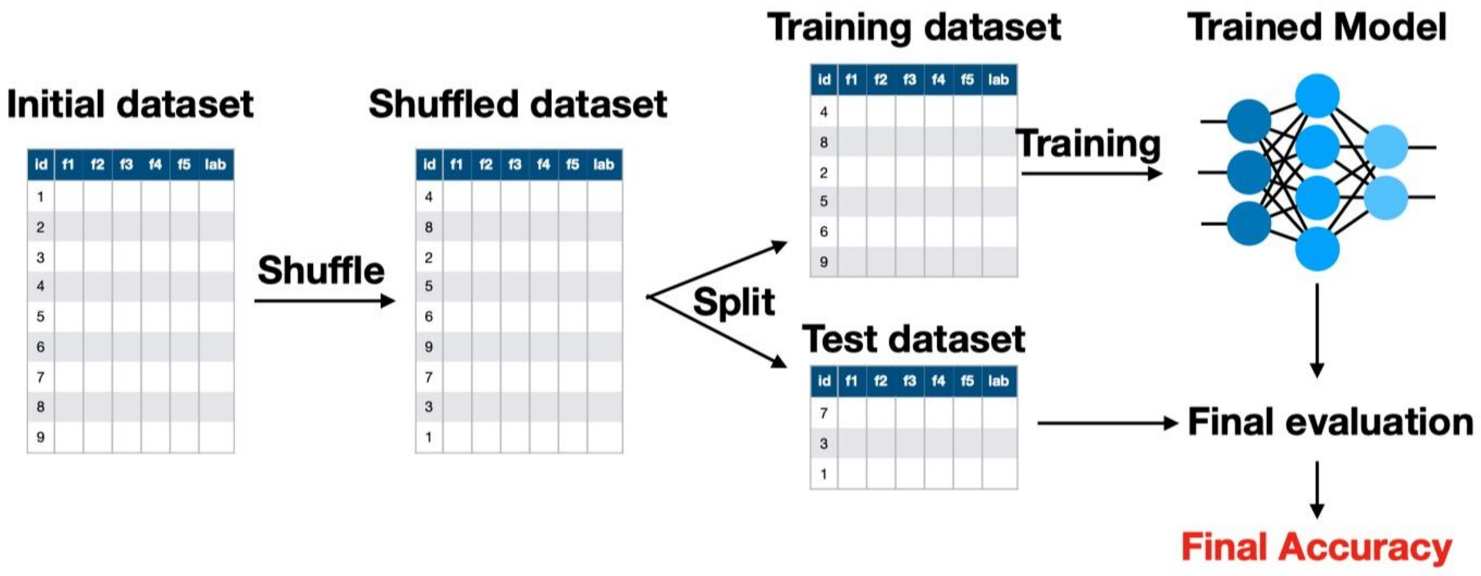




Our current pipeline:
Cannot know how good it generalizes

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

Hold-out Method for Performance Evaluation



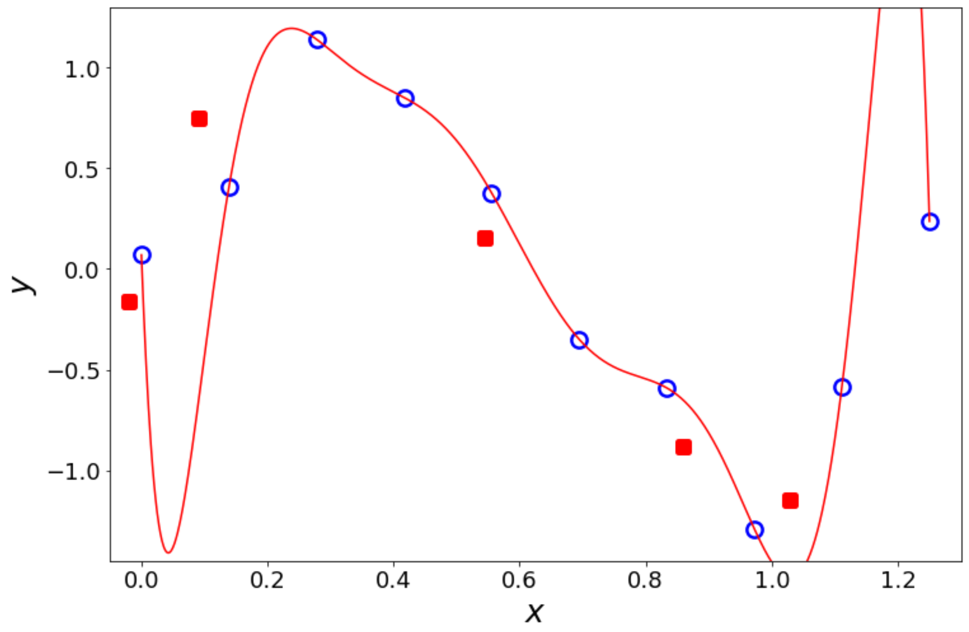
Example of Stratified sampling:

D has 500 positive & 500 negative samples

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

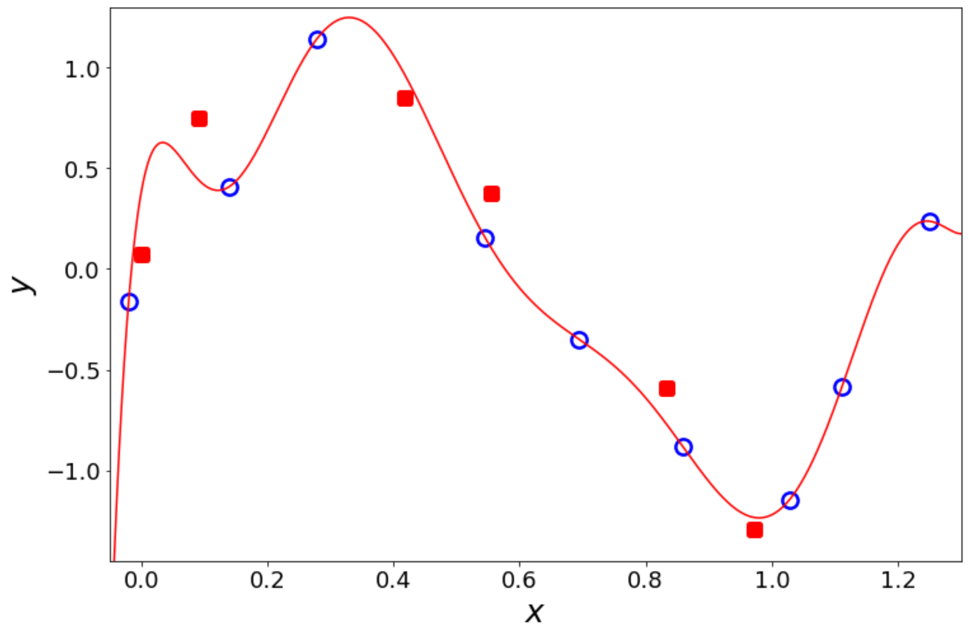
- Randomly split data into training and testing sets (e.g., 70% for training and 30% for testing)
- The training/test split should preserve a consistent distribution (use stratified sampling).
- Performance of a model must be evaluated in a held-out testing set.
- The test dataset should NEVER be used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

Limitations of Hold-out Method



MSE (training) = 0

MSE (testing) = 136.76



MSE (training) = 0

MSE (test) = 0.0537

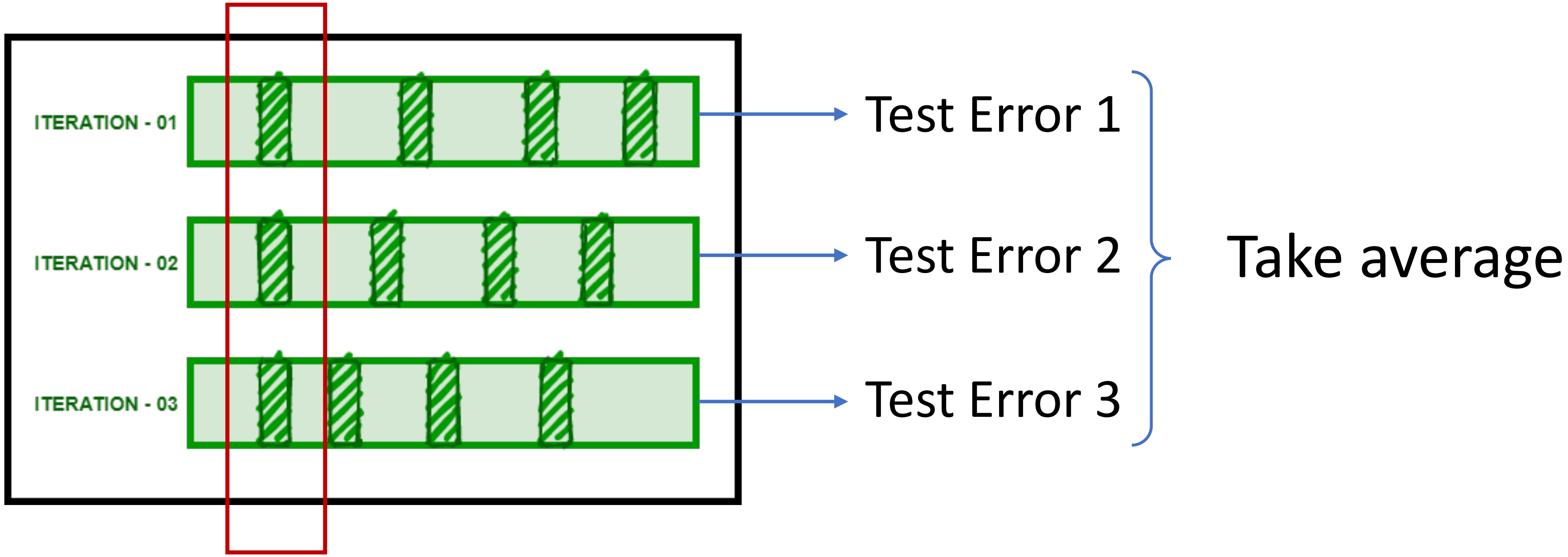
Same data points with two different data split

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.

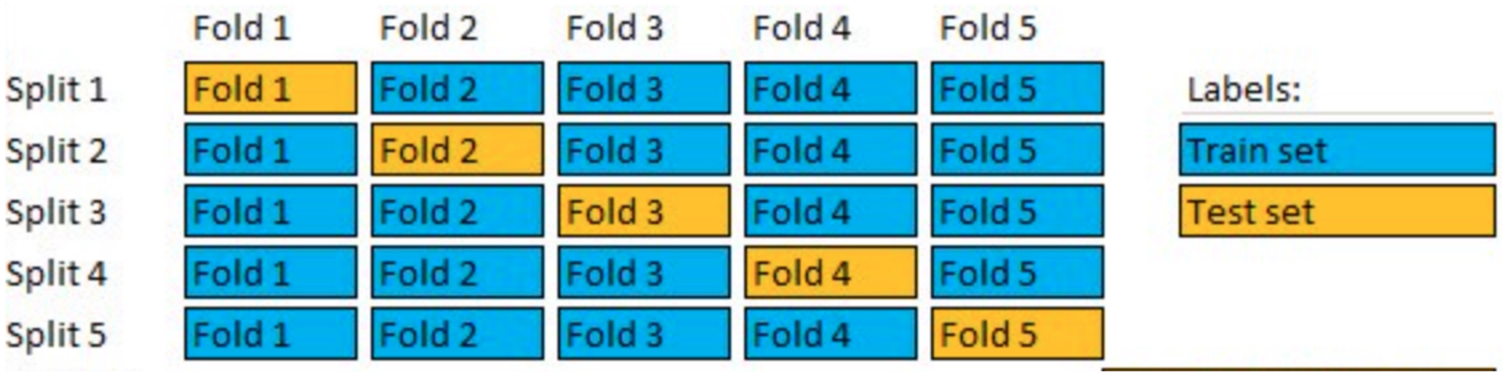


Possible to have overlaps

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

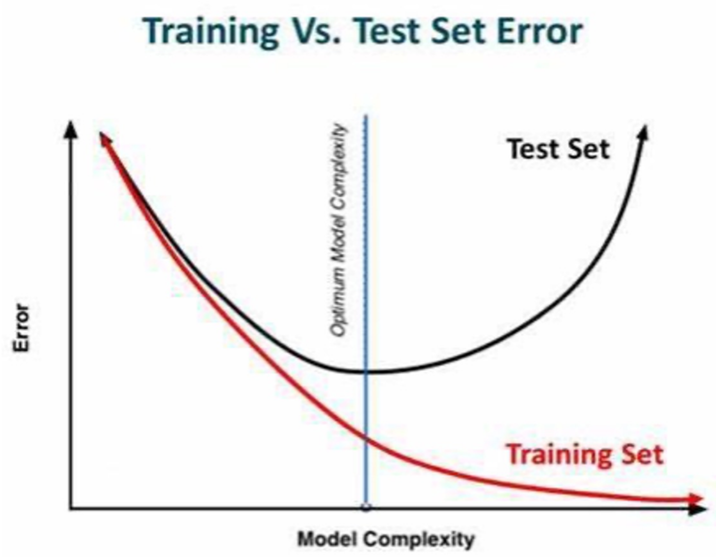
Use a smaller T

Algorithm: gradient descent

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

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

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



Performance Measures

How to compare the performance of two models?

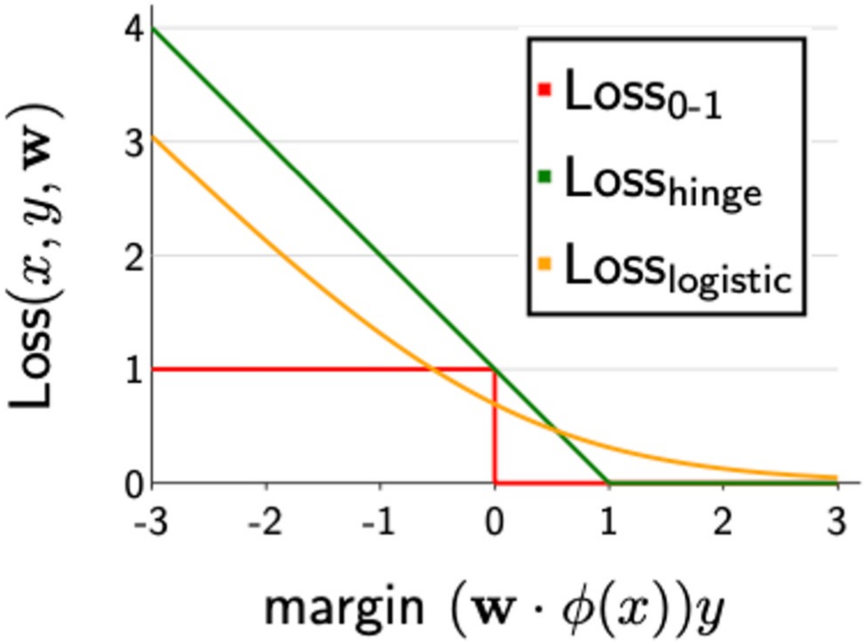
So far, we only used training loss/error and test loss/error.

For regression: we can use the MSE to measure the performance of different models.

How about classification?

- Logistic regression: logistic loss
- SVM: hinge loss
- Decision tree

We need standard performance metrics



Confusion Matrix for Classification Problems

For each model, we can construct a confusion matrix:

	Predicted Positive	Predicted Negative
Ground-truth Positive	TP	FN
Ground-truth Negative	FP	TN

Confusion matrix for binary classification

TP: number of true positives
TN: number of true negatives

FP: number of false positives
FN: number of false negatives

$N = TP + TN + FP + FN$: total number of data samples

Confusion Matrix for Classification Problems

Example: Construct a fusion matrix given the following test set and the model predictions

Outlook	Other features	Play?	Prediction	
sunny	…	No	Yes	*false positive*
sunny	…	No	No	*true negative*
overcast	…	Yes	Yes	*true positive*
rain	…	Yes	No	*false negative*
rain	…	Yes	No	*false negative*
rain	…	No	Yes	*false positive*
overcast	…	Yes	Yes	*true positive*
sunny	…	No	No	*true negative*
sunny	…	Yes	No	*false negative*
rain	…	Yes	Yes	*true positive*

	Predicted Positive	Predicted Negative
Ground-truth Positive	$TP = 3$	$FN = 3$
Ground-truth Negative	$FP = 2$	$TN = 2$

$$N = TP + TN + FP + FN = 10$$

Evaluation Metrics for Binary Classification Problems

Given this confusion matrix, we can define some evaluation metrics:

	Predicted Positive	Predicted Negative
Ground-truth Positive	$\boxed{TP}$	FN
Ground-truth Negative	FP	$\boxed{TN}$

Number of samples that are correctly predicted

1. **Accuracy**:

$$\mathrm{acc}(f; D) = \frac{\boxed{TP + TN}}{N}$$

The ratio of correct samples

Evaluation Metrics for Binary Classification Problems

Given this confusion matrix, we can define some evaluation metrics:

	Predicted Positive	Predicted Negative
Ground-truth Positive	TP	FN
Ground-truth Negative	FP	TN

Number of samples that are predicted to be positive

Number of samples that are really positive (ground-truth)

2. Precision:

$$\text{Precision}(f; D) = \frac{TP}{\boxed{TP + FP}}$$

Out of the predicted positives, how many are really positive?

3. Recall:

$$\text{Recall}(f; D) = \frac{TP}{\boxed{TP + FN}}$$

Out of the positive samples, how many are predicted positive?

Evaluation Metrics for Binary Classification Problems

Precision and Recall are contradictory

Generally, when the recall is high, the precision is often low, and vise versa.

4. F1 Score:

$$\text{Precision}(f;D) = \frac{TP}{TP+FP} \qquad \text{Recall}(f;D) = \frac{TP}{TP+FN}$$

$$\text{F1}(f;D) = \frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}} = \frac{2\times TP}{N+TP-TN}$$

A combination of precision and recall

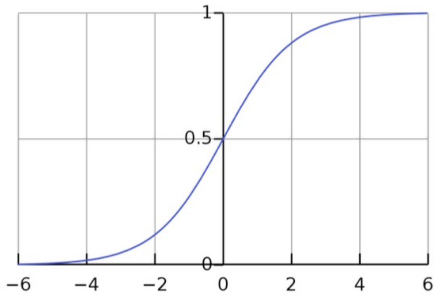
F1 is the harmonic mean of precision and recall: $\frac{1}{F1} = \frac{1}{2}\left(\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}\right)$

Evaluation Metrics for Binary Classification Problems

However, the outcome of our predictors are continuous "scores"

Outlook	Other features	Play?	Score	Prediction
sunny	…	No	0.96	Yes
sunny	…	No	0.11	No
overcast	…	Yes	0.76	Yes
rain	…	Yes	0.37	No
rain	…	Yes	0.42	No
rain	…	No	0.28	Yes
overcast	…	Yes	0.77	Yes
sunny	…	No	0.03	No
sunny	…	Yes	0.39	No
rain	…	Yes	0.56	Yes

Thresholding



E.g., in logistic regression:
Score $\geq$ 0.5: positive
Score $<$ 0.5: negative

- Different thresholds give different evaluation results.
- 0.5 is often a good choice but may not always be the case.

Evaluation Metrics for Binary Classification Problems

In practice, we sort the predicted "scores" in descending order

#	Score
1	0.96
2	0.11
3	0.76
4	0.37
5	0.42
6	0.28
7	0.77
8	0.03
9	0.39
10	0.56

sorting

Samples more likely to be positive

Samples more likely to be negative

#	Score
1	0.96
7	0.77
3	0.76
10	0.56
5	0.42
9	0.39
4	0.37
6	0.28
2	0.11
8	0.03

cut point

Before cut point: Positive predictions

After cut point: Negative predictions

Evaluation Metrics for Binary Classification Problems

True Positive Rate and False Positive Rate

#	Label	Score	
1	Yes	0.96	+
7	No	0.77	+
3	Yes	0.76	+
10	No	0.56	-
5	Yes	0.42	-
9	Yes	0.39	-
4	No	0.37	-
6	No	0.28	-
2	No	0.11	-
8	No	0.03	-

cut point

sorted predictions

	Predicted Positive	Predicted Negative
Ground-truth Positive	TP	FN
Ground-truth Negative	FP	TN

True Positive Rate: $\mathrm{TPR} = \frac{TP}{TP+FN}$

False Positive Rate: $\mathrm{FPR} = \frac{FP}{FP+TN}$

Exercise: Compute TPR and FPR for the example shown in left

TP=2, FN=2, FP=1, TN=5. Therefore, TPR=2/4 and FPR=1/6

Evaluation Metrics for Binary Classification Problems

Receiver Operating Characteristic (ROC) Curve:

#	Label	Score	
1	Yes	0.96	-
7	No	0.77	-
3	Yes	0.76	-
10	No	0.56	-
5	Yes	0.42	-
9	Yes	0.39	-
4	No	0.37	-
6	No	0.28	-
2	No	0.11	-
8	No	0.03	-

cut point

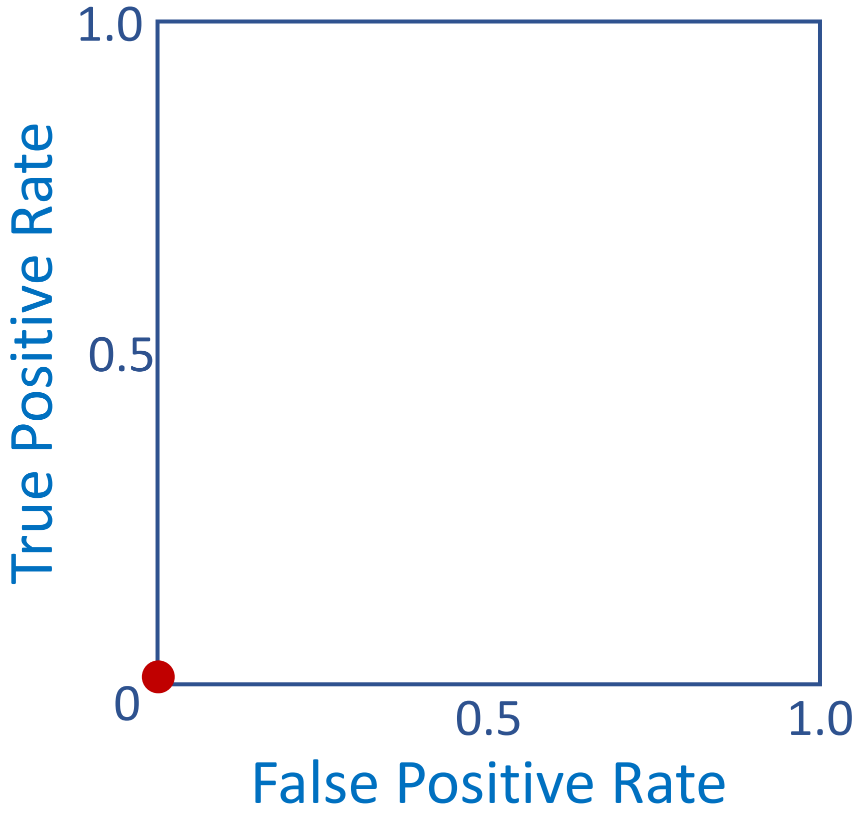
sorted predictions

For each cut point, we can plot its corresponding TPR and FPR in a figure:

TP=0; FN=4
FP=0; TN=6

$$\text{TPR} = \frac{TP}{TP+FN} = 0$$

$$\text{FPR} = \frac{FP}{FP+TN} = 0$$



Evaluation Metrics for Binary Classification Problems

Receiver Operating Characteristic (ROC) Curve:

#	Label	Score	
1	Yes	0.96	+
7	No	0.77	-
3	Yes	0.76	-
10	No	0.56	-
5	Yes	0.42	-
9	Yes	0.39	-
4	No	0.37	-
6	No	0.28	-
2	No	0.11	-
8	No	0.03	-

cut point

sorted predictions

TP=1; FN=3

FP=0; TN=6

$$\mathrm{TPR} = \frac{TP}{TP + FN} = 0.25$$

$$\mathrm{FPR} = \frac{FP}{FP + TN} = 0$$

For each cut point, we can plot its corresponding TPR and FPR in a figure:

Evaluation Metrics for Binary Classification Problems

Receiver Operating Characteristic (ROC) Curve:

#	Label	Score	
1	Yes	0.96	+
7	No	0.77	+
3	Yes	0.76	-
10	No	0.56	-
5	Yes	0.42	-
9	Yes	0.39	-
4	No	0.37	-
6	No	0.28	-
2	No	0.11	-
8	No	0.03	-

cut point

sorted predictions

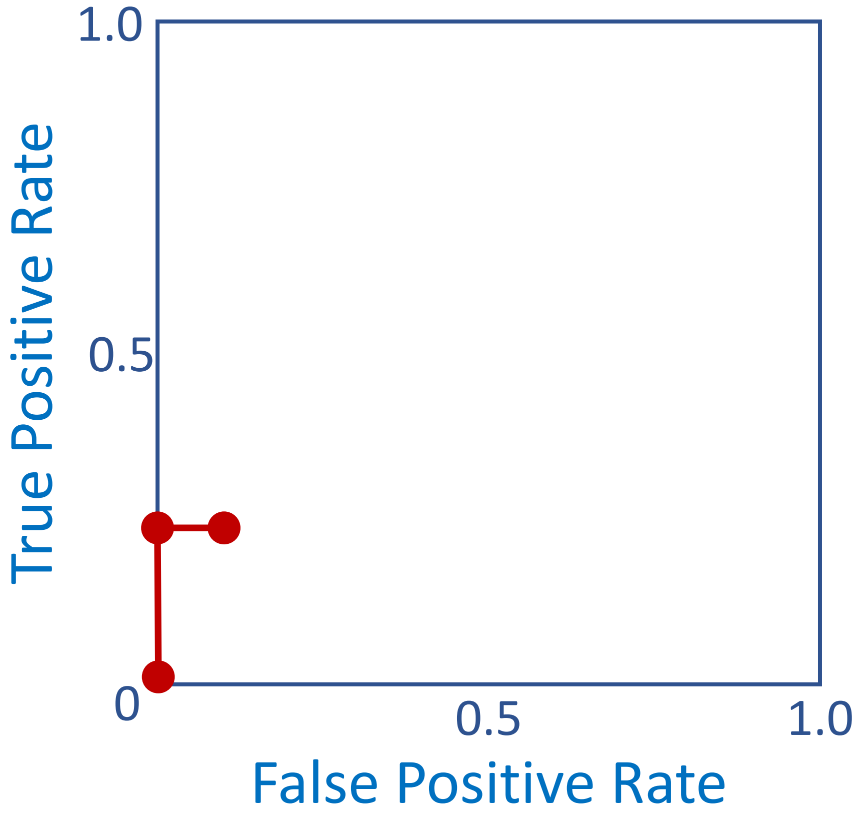
For each cut point, we can plot its corresponding TPR and FPR in a figure:

TP=1; FN=3

FP=1; TN=5

$$\mathrm{TPR} = \frac{TP}{TP + FN} = 0.25$$

$$\mathrm{FPR} = \frac{FP}{FP + TN} = 0.17$$



Evaluation Metrics for Binary Classification Problems

Receiver Operating Characteristic (ROC) Curve:

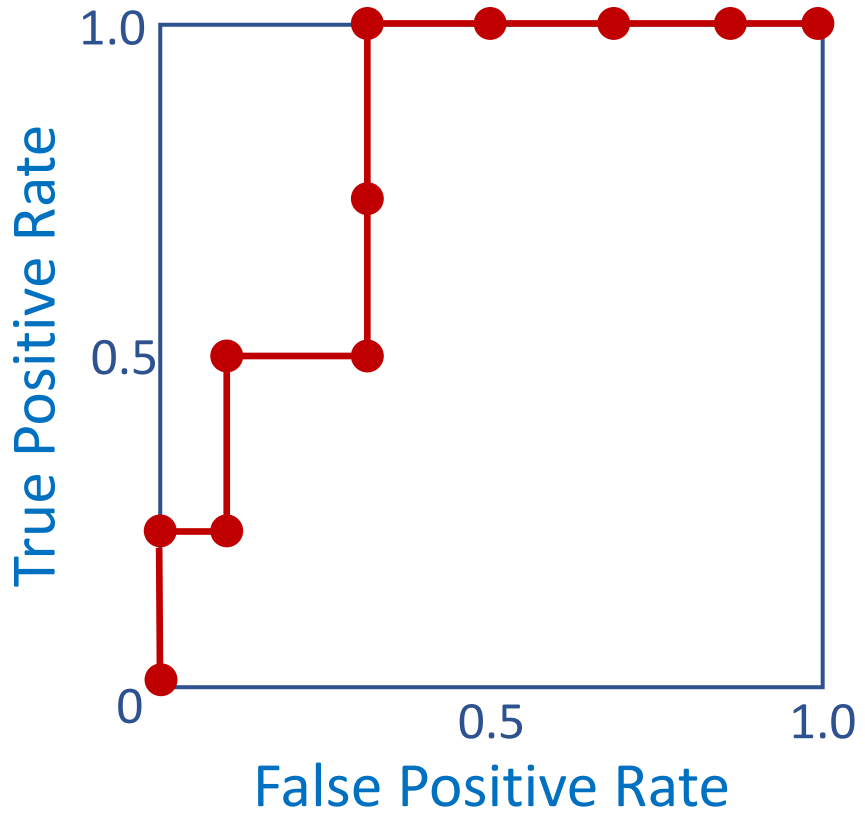
#	Label	Score	
1	Yes	0.96	+
7	No	0.77	+
3	Yes	0.76	+
10	No	0.56	+
5	Yes	0.42	+
9	Yes	0.39	+
4	No	0.37	+
6	No	0.28	+
2	No	0.11	+
8	No	0.03	+

cut point

sorted predictions

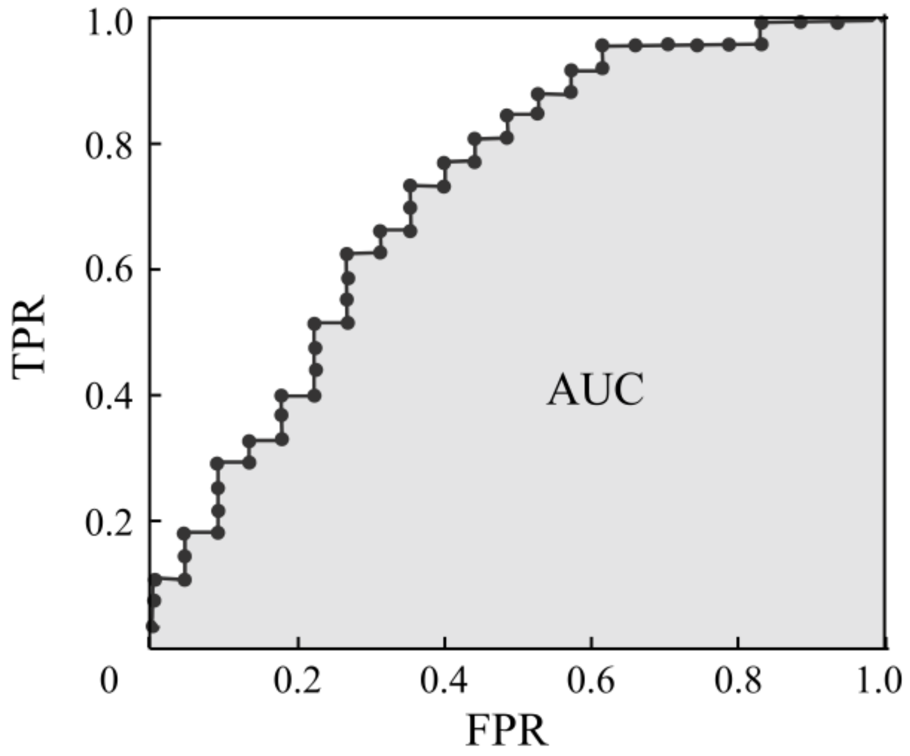
For each cut point, we can plot its corresponding TPR and FPR in a figure:

Continue this process:

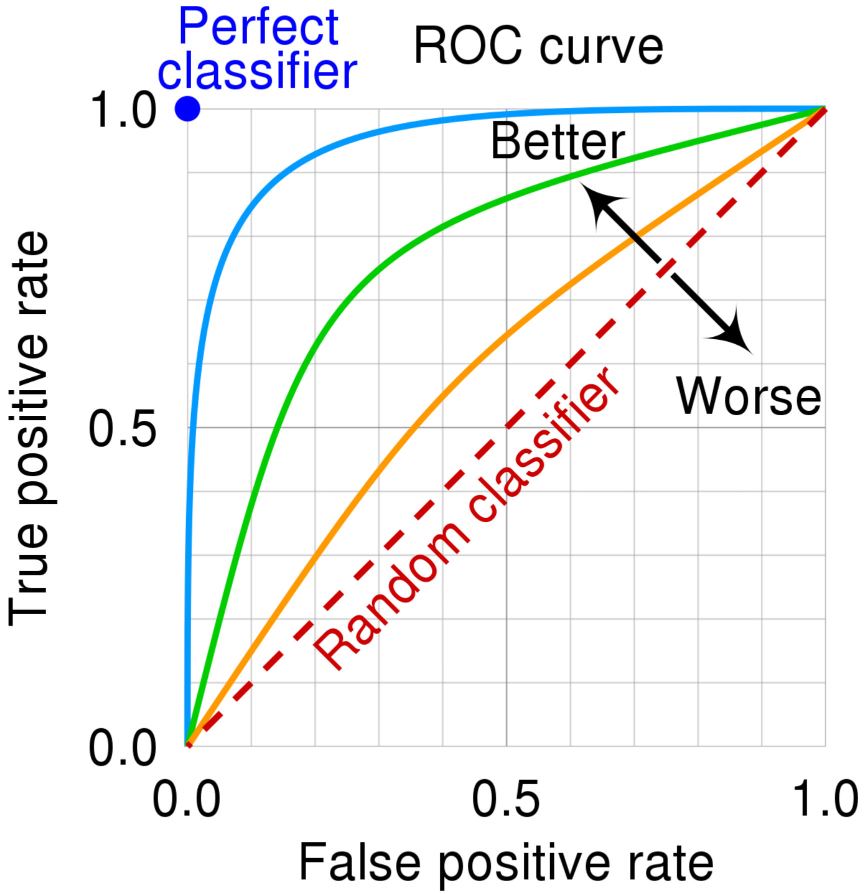


Evaluation Metrics for Binary Classification Problems

ROC and AUC Score



We can get a smother curve with more samples



AUC Score: Area under the ROC Curve

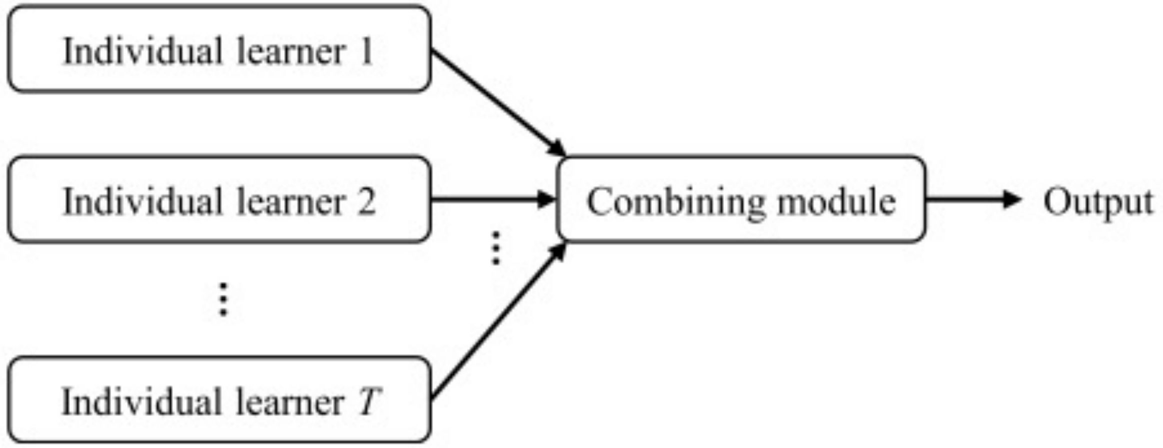
AUC: The larger, the better

Ensemble Learning and Random Forest

Ensemble Learning

Predictive power of an individual model might be limited.
How about we train multiple models and let them vote?

Base model



Ensemble Learning

Why and when could ensemble learning help?

	Testing sample 1	Testing sample 2	Testing sample 3
h_1	✓	✓	✗
h_2	✗	✓	✓
h_3	✓	✗	✓
Ensemble	✓	✓	✓

(a) Ensemble helps.

	Testing sample 1	Testing sample 2	Testing sample 3
h_1	✓	✓	✗
h_2	✓	✓	✗
h_3	✓	✓	✗
Ensemble	✓	✓	✗

(b) Ensemble doesn't help.

	Testing sample 1	Testing sample 2	Testing sample 3
h_1	✓	✗	✗
h_2	✗	✓	✗
h_3	✗	✗	✓
Ensemble	✗	✗	✗

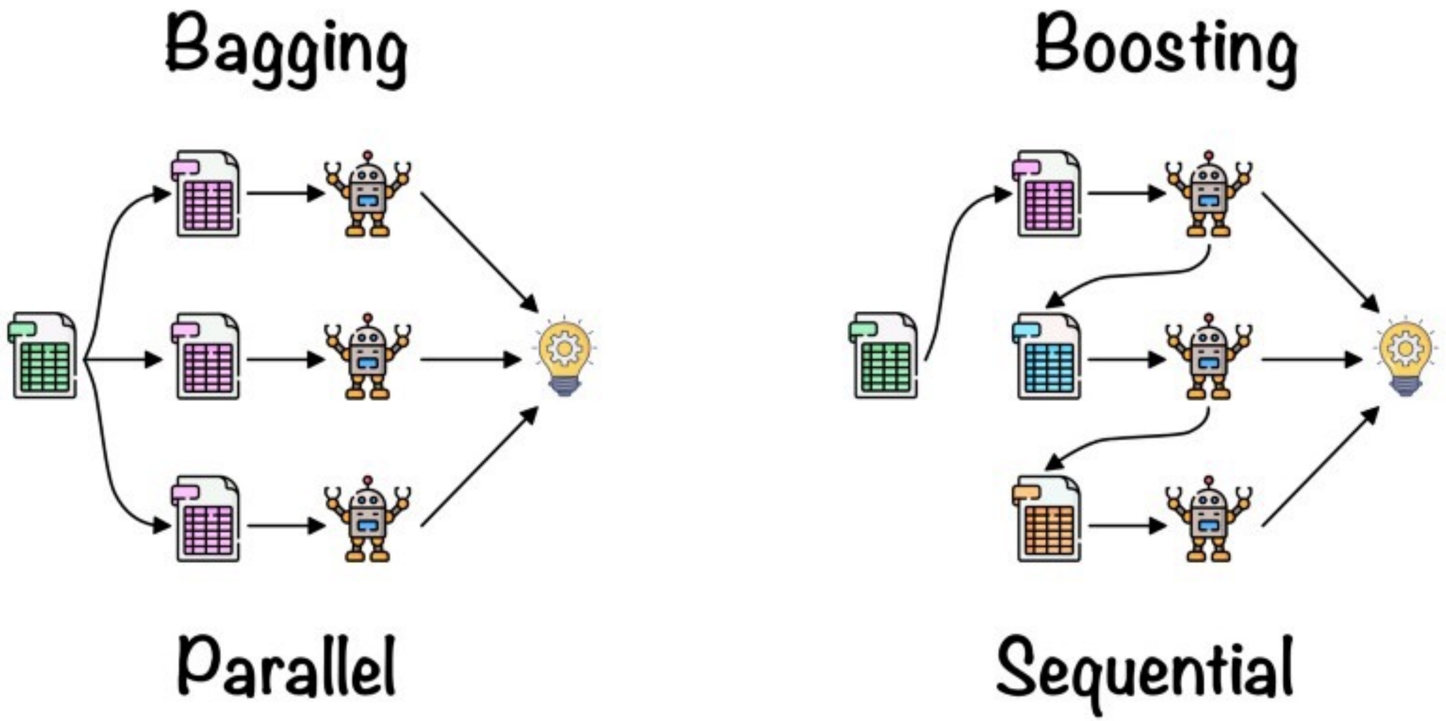
(c) Ensemble hurts.

In (a), every classifier only has an accuracy of 66.6%, but the ensemble achieves an accuracy of 100%

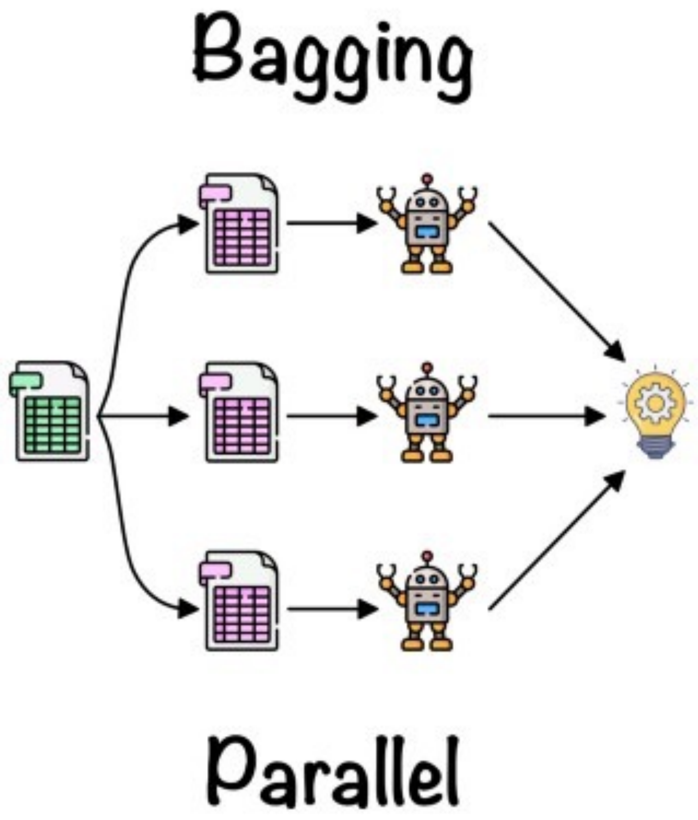
Base models should be <u>better than a random one (>50%)</u> and are <u>diverse</u>

Ensemble Learning

Two categories of ensemble learning



Bagging



Algorithm 8.2 Bagging

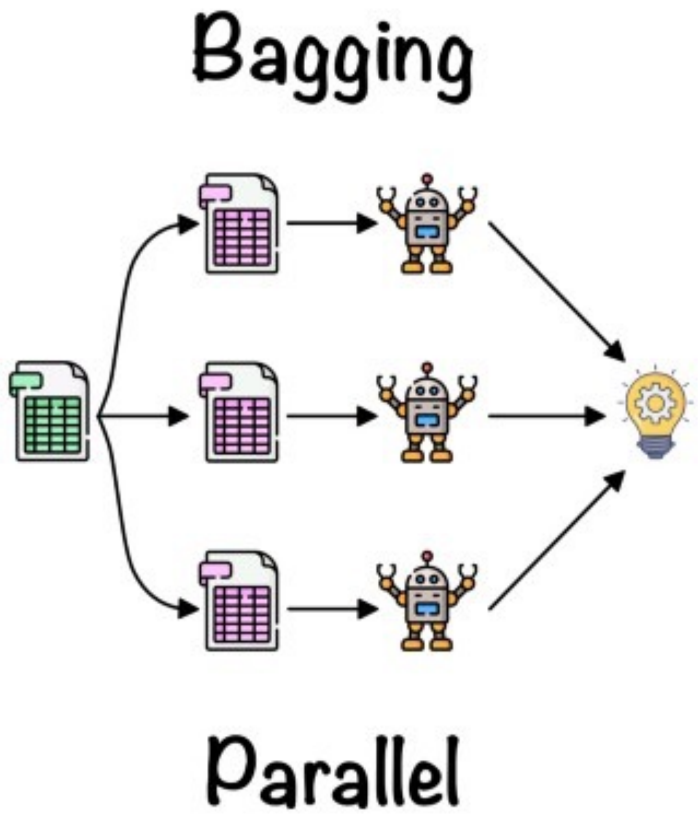
Input: Training set: $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Base learning algorithm $\mathcal{L}$;
Number of training rounds T.

Process:

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $\quad h_t = \mathcal{L}(D, \mathcal{D}_{bs})$.
3: **end for**

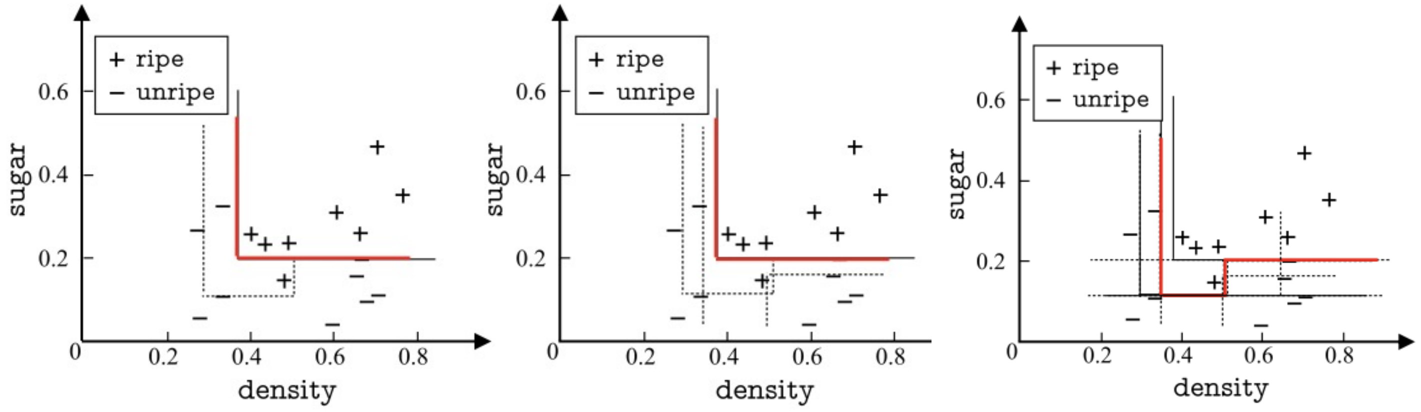
Output: $H(\boldsymbol{x}) = \underset{y \in \mathcal{Y}}{\arg\max} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y)$.

Random Forest: Bagging with Decision Trees



Random: randomly select a subset of data and a subset of attributes to train the base models

Forest: make predictions with multiple trees



(a) 3 base learners. (b) 5 base learners. (c) 11 base learners.

Machine Learning Summary

- Loss minimization framework
- Regression: mean squared loss
- Classification
  - Zero-one loss, hinge loss, and logistic loss
  - Decision Tree Algorithm
- Generalization and Model Selection
- Performance Metrics
- Ensemble Learnings