

# EE2211 Introduction to Machine Learning

## Lecture 10

Wang Xinchao  
[xinchao@nus.edu.sg](mailto:xinchao@nus.edu.sg)

# Course Contents

- Introduction and Preliminaries (Xinchao)
    - Introduction
    - Data Engineering
    - Introduction to Linear Algebra, Probability and Statistics
  - Fundamental Machine Learning Algorithms I (Yueming)
    - Systems of linear equations
    - Least squares, Linear regression
    - Ridge regression, Polynomial regression
  - Fundamental Machine Learning Algorithms II (Yueming)
    - Over-fitting, bias/variance trade-off
    - Optimization, Gradient descent
    - Decision Trees, Random Forest
  - Performance and More Algorithms (Xinchao)
    - Performance Issues
    - K-means Clustering
    - Neural Networks
- [Important] In the Final, no coding questions for Xinchao's part!
- Despite you will see some in the tutorial, they won't be tested.

# EE2211: Learning Outcome

## A Summary of Module Content

- **I am able to understand the formulation of a machine learning task**
  - Lecture 1 (feature extraction + classification)
  - Lecture 4 to Lecture 9 (regression and classification)
  - Lecture 11 and Lecture 12 (clustering and neural network)
- **I am able to relate the fundamentals of linear algebra and probability to machine learning**
  - Lecture 2 (recap of probability and linear algebra)
  - Lecture 4 to Lecture 8 (regression and classification)
  - Lecture 12 (neural network)
- **I am able to prepare the data for supervised learning and unsupervised learning**
  - Lecture 1 (feature extraction) [For supervised and unsupervised]
  - Lecture 2 (data wrangling) [For supervised and unsupervised]
  - Lecture 10 (Training/Validation/Test) [For supervised]
  - Programming Exercises in tutorials
- **I am able to evaluate the performance of a machine learning algorithm**
  - Lecture 5 to Lecture 9 (evaluate the difference between labels and predictions)
  - Lecture 10 (evaluation metrics)
- **I am able to implement regression and classification algorithms**
  - Lecture 5 to Lecture 9

# Outline

- Dataset Partition:
  - Training/Validation/Testing
- Cross Validation
- • Evaluation Metrics
  - Evaluating the quality of a trained machine learning system

**We will talk about many metrics:  
It is OK you can't memorize them all  
But intuition is important!**

# A Real-world Scenario

- We would like to train a Random Forest for face classification (i.e., to tell an image is a human face or not)



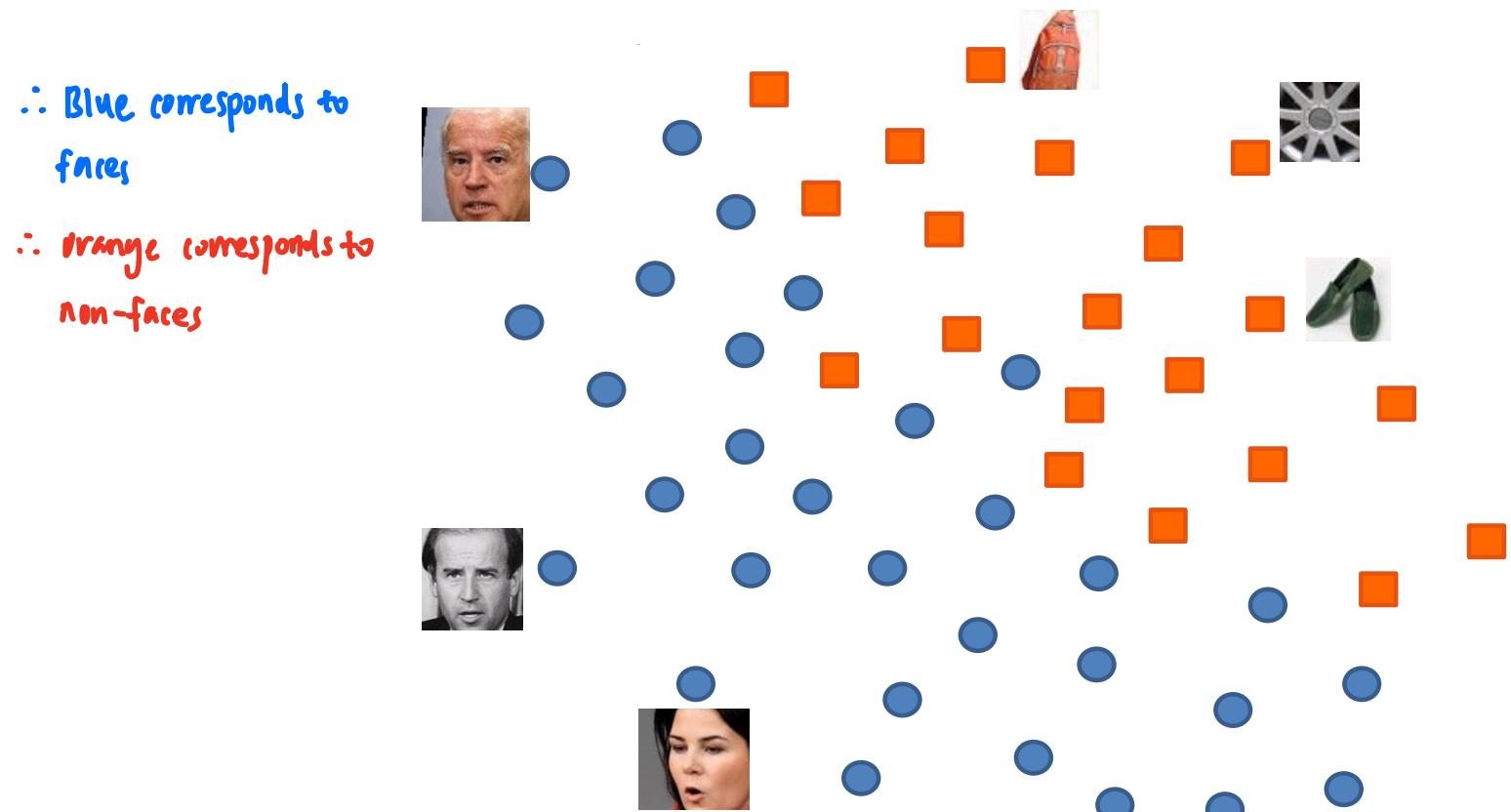
Faces



Non-faces

# A Real-world Scenario

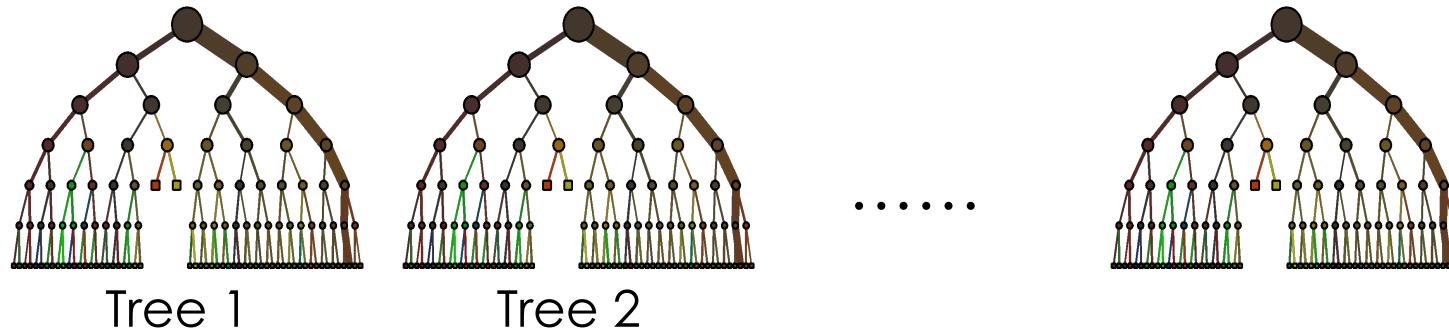
- We would like to train a Random Forest for face classification (i.e., to tell an image is a human face or not)
  - We will have one dataset to train the Random Forest



# A Real-world Scenario

- We would like to train a Random Forest for face classification (i.e., to tell an image is a human face or not)
  - We will have **one dataset** to **train** the Random Forest
  - We will have **tunable (hyper)parameters** for the Random Forest.  
For example, ***the number of trees*** in the Random Forest
    - Shall we **use 100 trees?**
    - Shall we **use 200 trees?**
    - ...

We need to decide on the parameter



# A Real-world Scenario

- We would like to train a Random Forest for face classification (i.e., to tell an image is a human face or not)
  - We will have **one dataset** to **train** the Random Forest
  - We will have tunable **(hyper)parameters** for the Random Forest.  
For example, ***the number of trees*** in the Random Forest
    - Shall we use 100 trees?
    - Shall we use 200 trees?
    - ...

We need to decide on the parameter

- ① Train
  - ② Select parameters
  - ③ Apply test on unseen data
- ↳ see how well it works on unseen data

- Once we decide the number of trees, we will apply the Random Forest with the **selected parameter** on **unseen** test data.

## Test Data



Yes!



No!

# Training, Validation, and Test

- In **real-world application**,
  - We **don't have test data**, since they are **unseen**
  - Imagine you develop a face detector app, you don't know whom you will test on
- In **lab practice**, *↑  
but we only have one dataset*
  - We divide the **dataset** into three parts



For training the ML models

(1)

For validation:  
choosing the  
parameter or  
model

(2)

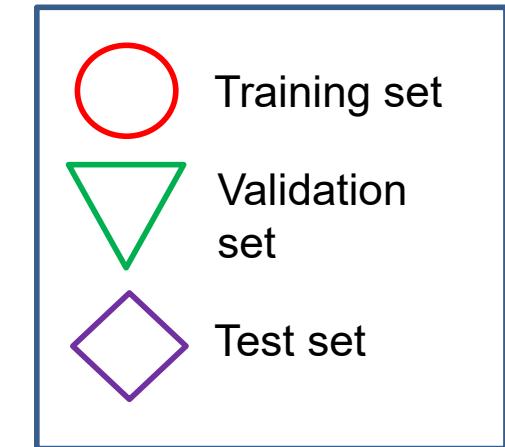
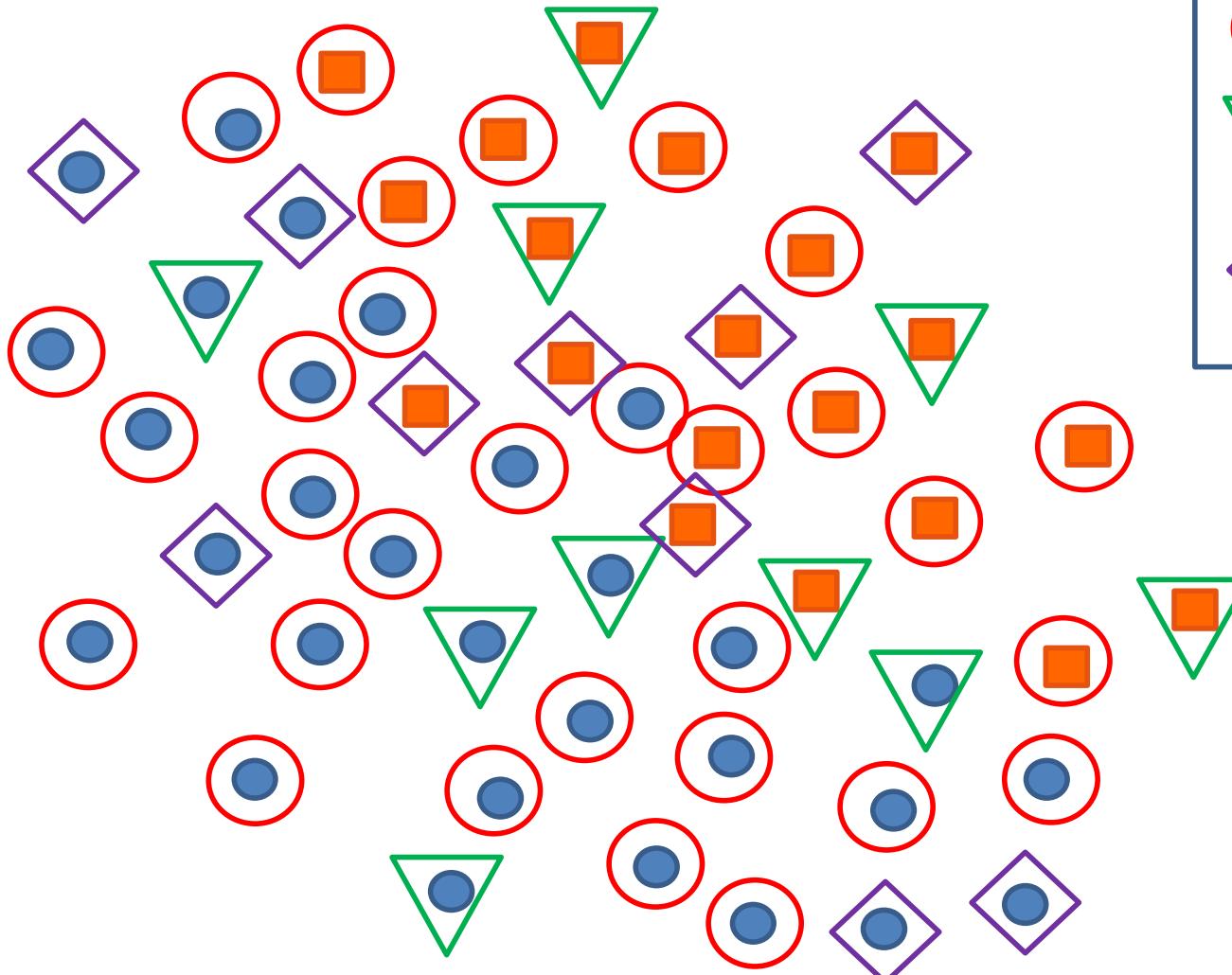
For testing the “real”  
performance and  
generalization

(3)

\*\*\*\*\*

– **NEVER touch test data during training/validation!!!**

# Training, Validation, and Test



# Training, Validation, and Test



Training set	Validation set	Test set
--------------	----------------	----------

For training the ML models

Validation set

Test set

For validation:  
choosing the  
parameter or  
model

For testing the “real”  
performance and  
generalization

\*\*\*\*  
Once best parameter found,  
can combine training and  
validation set and train  
another RF with best  
parameter (100trees)  
↓  
train more powerful RF  
as more data

Example: Assume I want to build a Random Forest. I have a parameter to decide: shall I have

- 100 Trees?
- 200 Trees?

What we do next is to use the training set to train two classifiers,

1)  $C_1$ : Random Forest with 100 trees, and 2)  $C_2$ : Random Forest with 200 trees

They have the following accuracy:

1.  $C_1$ : Random Forest with 100 trees: validation accuracy 90%
2.  $C_2$ : Random Forest with 200 trees: validation accuracy 88%

↳ apply  $C_1$  and  $C_2$  on validation set

∴ check validation  
accuracy to select best  
parameter

Which one to choose for real application, i.e., testing?

The one with higher validation accuracy, i.e., Random Forest with 100 trees!  $C_1$

# Python Demo: lec10.ipynb



For training the ML models      For validation: choosing the parameter or model      For testing the “real” performance and generalization

Load Dataset Split it into Train:Val:Test = 100:25:25

```
[76]: ##-- load data from scikit ---#
import numpy as np
import pandas as pd
print("pandas version: {}".format(pd.__version__))
import sklearn
print("scikit-learn version: {}".format(sklearn.__version__))
from sklearn.datasets import load_iris

## Set Seed
seed = 20

## Load dataset
iris_dataset = load_iris()
X = np.array(iris_dataset['data'])
y = np.array(iris_dataset['target'])

## one-hot encoding
Y = list()
for i in y:
    letter = [0, 0, 0]
    letter[i] = 1
    Y.append(letter)
Y = np.array(Y)

## Random shuffle data and train-test split
test_Idx = np.random.RandomState(seed=seed).permutation(Y.shape[0])
X_test = X[test_Idx[:25]]
Y_test = Y[test_Idx[:25]]
X = X[test_Idx[25:]]
Y = Y[test_Idx[25:]]
```

## • Problem Setup

- Dataset used: IRISdataset
  - Link: [https://scikit-learn.org/stable/datasets/toy\\_dataset.html#iris-dataset](https://scikit-learn.org/stable/datasets/toy_dataset.html#iris-dataset)
- Training/Validation/Test: 100/25/25
- Machine Learning Task and Model: Polynomial regression
- Parameters to select: Order 1 to 10

In the Final, no coding questions for Xinchao’s part!

→  $k^{\text{th}}$  integer (user set)

# k-fold Cross Validation [cv]

- In practice, we do the k-fold cross validation

Test

4-fold cross validation

\*\*

Step 1: take out **test set** from the dataset

# k-fold Cross Validation

- In practice, we do the **k-fold cross validation**

Test

4-fold cross validation



Step 2: We partition the *remaining part of the dataset* (after taking out the test set), into  $k$  equal parts (equal in terms of number of samples).

# k-fold Cross Validation

- In practice, we do the **k-fold cross validation**

Test

4-fold cross validation

Fold 1	Train	Train	Train	Validation
Fold 2	Train	Train	Validation	Train
Fold 3	Train	Validation	Train	Train
Fold 4	Validation	Train	Train	Train

Order of the samples are kept the same across all folds.

Step 3: We run ***k folds*** (i.e.,  $k$  times) of experiments.

Within each fold, we use ***one part*** as ***validation set***, and the ***k-1 remaining parts*** as ***training set***. We use different validation sets for different folds.

$C_i \rightarrow$  Fold (k) #  
 $C_j \rightarrow$  parameter #

Classifiers  
Trained

Test	4-fold cross validation				
Fold 1	Train	Train	Train	Validation	$C_1^1 \ C_2^1$
Fold 2	Train	Train	Validation	Train	$C_1^2 \ C_2^2$
Fold 3	Train	Validation	Train	Train	$C_1^3 \ C_2^3$
Fold 4	Validation	Train	Train	Train	$C_1^4 \ C_2^4$

- 1)  $C_1$ : Random Forest with 100 trees
- 2)  $C_2$ : Random Forest with 200 trees

$\downarrow$   
 parameters

Step 3.1: Within each fold, if we have  $n$  parameter/model candidates, we will train  $n$  models, and we check their validation performance.

# k-fold Cross Validation

- In practice, we do the **k-fold cross validation**

Test	4-fold cross validation				Classifiers Trained
	Fold 1	Train	Train	Train	
Fold 2	Train	Train	Validation	Train	$C_1^2 \ C_2^2$
Fold 3	Train	Validation	Train	Train	$C_1^3 \ C_2^3$
Fold 4	Validation	Train	Train	Train	$C_1^4 \ C_2^4$

Example: which one to select for test?

	Fold 1 Accuracy on Validation Set 1	Fold 2 Accuracy on Validation Set 2	Fold 3 Accuracy on Validation Set 3	Fold 4 Accuracy on Validation Set 4	Average Accuracy on All Validation Sets
Classifier with Param1 (e.g. 100 trees)	88% $C_1^1$	89% $C_1^2$	93% $C_1^3$	92% $C_1^4$	90.5%
Classifier with Param2 (e.g. 200 trees)	90% $C_2^1$	88% $C_2^2$	91% $C_2^3$	91% $C_2^4$	90%

Step 4: We select the parameter/model with best average validation performance over k folds.

# k-fold Cross Validation

**Other common partitioning:**

- 10-Fold CV
- 5-Fold CV
- 3-Fold CV



We may decide on the size of the test set, for example, 15%, 20%, 30% of the whole dataset, the rest for training/validation.

# k-fold Cross Validation

- The **test set** contains the examples that the learning algorithm has **never seen before**,
- So **test performance** shows how well our model **generalizes**.

\*\*\*\*\*

## Example:

Xinchao uses k-fold cross validation to obtain an optimal parameter for his model (e.g., decision tree). This parameter, on the test set, achieves accuracy of 0.8.

Yueming uses k-fold cross validation to obtain an optimal parameter for her model (e.g., random forest). This parameter, on the test set, achieves accuracy of 0.9.

We can say, Yueming's model **generalizes** better than Xinchao's model.

## Take home message:

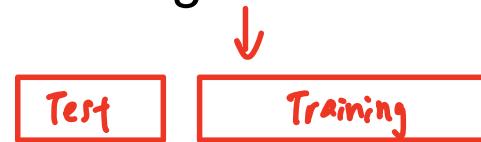
\*\*\*\*\*

**Validation** performance -> **Selecting parameters!**

**Test** performance -> The “real” performance of a model with selected parameter!

# Training, Validation, and Test

- Validation is however not always used:
  - Validation is used when you need to pick parameters or models
  - If you have no models or parameters to compare, you may consider partition the data into only training and test



# Outline

- Dataset Partition:
  - Training/Validation/Testing
- Cross Validation
- Evaluation Metrics
  - Evaluating the quality of a trained machine learning system

**We will talk about many metrics:  
It is OK you can't memorize them all  
But intuition is important!**

# Evaluation Metrics

## Regression

### 1) Mean Square Error

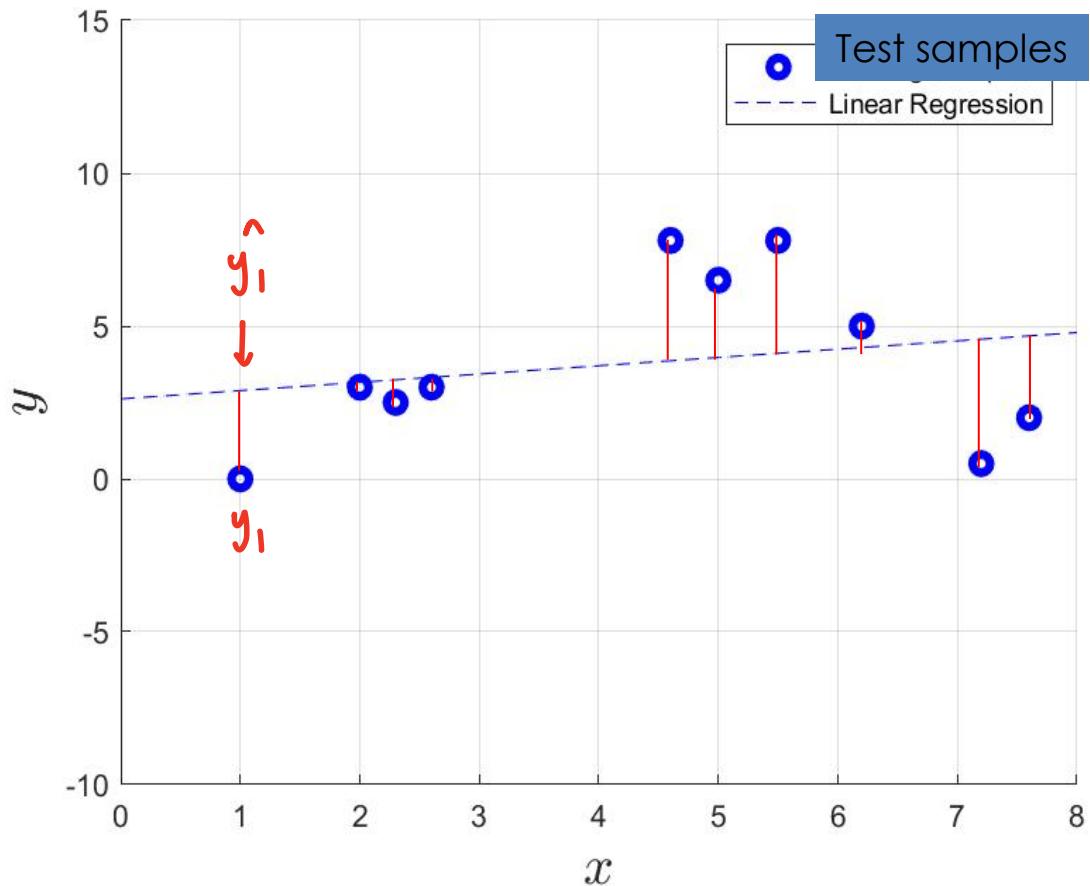
$$(MSE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n})$$

↓ origin      ↓ predicted

### 2) Mean Absolute Error

$$(MAE = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n})$$

where  $y_i$  denotes the target output and  $\hat{y}_i$  denotes the predicted output for sample  $i$ .



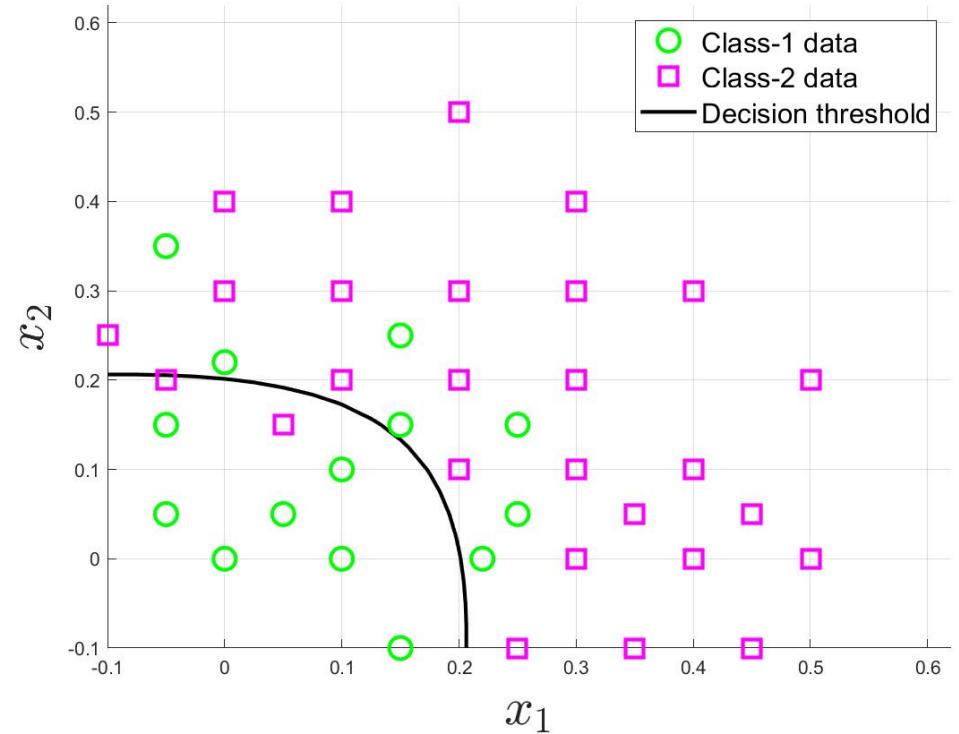
# Evaluation Metrics

## Classification

Class-1: Positive Class  
Class-2: Negative Class

Confusion Matrix

	Class-1 (predicted)	Class-2 (predicted)
Class-1 (actual)	7 (TP)	7 (FN)
Class-2 (actual)	2 (FP)	25 (TN)



TP: True Positive  
FN: False Negative (i.e., **Type II Error**)  
FP: False Positive (i.e., **Type I Error**)  
TN: True Negative

# Evaluation Metrics

## Classification

- How many samples in the dataset have the real label of Class-2? [27]

$$25 + 2 = 27$$

- How many samples are there in total? [41]

$$25 + 7 + 7 + 2 = 41$$

- How many sample are correctly classified? How many are incorrectly classified?

$$7 + 25 = 32 \quad [32]$$

$$[9]$$

	Class-1 (predicted)	Class-2 (predicted)
Class-1 (actual)	7 (TP)	7 (FN)
Class-2 (actual)	2 (FP)	25 (TN)

# Evaluation Metrics

## Classification

### Confusion Matrix for Binary Classification

	$\hat{P}$ (predicted)	$\hat{N}$ (predicted)	
P (actual)	TP	FN	Recall $TP/(TP+FN)$
N (actual)	FP	TN	
	Precision $TP/(TP+FP)$	Accuracy $(TP+TN)/(TP+TN+FP+FN)$	

correctly classified      total

# Evaluation Metrics

## Classification

### Cost Matrix for Binary Classification

	$\hat{P}$ (predicted)	$\hat{N}$ (predicted)
$P$ (actual)	$C_{p,p} * TP$	$C_{p,n} * FN$
$N$ (actual)	$C_{n,p} * FP$	$C_{n,n} * TN$

∴ Usually,  $C_{p,p}$  and  $C_{n,n}$  are set to 0 because the classification is correct, no need to impose penalty

#### Total cost:

$$O = C_{p,p} * TP + C_{p,n} * FN + C_{n,p} * FP + C_{n,n} * TN$$

$$= C_{p,n} * FN + C_{n,p} * FP$$

Main Idea: To assign different **penalties** for different entries. Higher penalties for more severe results. Smaller costs are preferred.

Usually,  $C_{p,p}$  and  $C_{n,n}$  are set to 0;  $C_{n,p}$  and  $C_{p,n}$  may and may not equal

# Evaluation Metrics

- Example of cost matrix
  - Assume we would like to develop a self-driving car system
  - We have an ML system that detects the pedestrians using camera, by conducting a binary classification
    - When it **detects a person (positive class)**, the car should stop
    - When **no person is detected (negative class)**, the car keeps going

True Positive (cost  $C_{p,p}$ )

There is person, ML detects person and car stops



Credit: automotiveworld.com

True Negative (cost  $C_{n,n}$ )

There is no person, car keeps going



False Positive (cost  $C_{n,p}$ )

There is **no person**, ML detects person and car stops

False Negative (cost  $C_{p,n}$ )

There is **person**, ML fails to detect person and car keeps going

$$C_{n,p} \stackrel{?}{\textcolor{red}{[<]}} C_{p,n} \quad (>, <, \text{ or } =)$$

# Evaluation Metrics

∴ Using just confusion matrix,  
accuracy not sufficient

- For unbalanced data...

- Assume we have 1000 samples, of which 10 are positive and 990 are negative

$$\text{Accuracy} = 990/1000 = 0.99!$$

Very high number! → does not reflect

- Yet, half of the Class-1 are Classified to Class-2!

	Class-1 (predicted)	Class-2 (predicted)
Class-1 (actual)	5 (TP)	5 (FN)
Class-2 (actual)	5 (FP)	985 (TN)

The goal is to highlight the problems of the results!

In this case, we shall

- 1) Use cost matrix, assign different costs for each entry
- 2) Use Precision and Recall! Precision = 0.5 and Recall = 0.5

# Evaluation Metrics

## Classification

\*\*\*\*\*

(True Positive Rate)  $TPR = TP/(TP+FN)$  **Recall**

(False Negative Rate)  $FNR = FN/(TP+FN)$

(True Negative Rate)  $TNR = TN/(FP+TN)$

(False Positive Rate)  $FPR = FP/(FP+TN)$

$TPR + FNR = 1$  (100% of positive-class data)

$TNR + FPR = 1$  (100% of negative-class data)

$\therefore$  Two class classification

$\hat{P}$ (predicted)	$\hat{N}$ (predicted)	
P (actual)	TP	FN
N (actual)	FP	TN

# Evaluation Metrics

## Classification

(predict) target  
input  
Prediction function  $y = f(x)$

sample	N1	N2	P1	N3	P2	P3
input x	-4	-3	-2.5	-2	-1.5	-0.5
Prediction y	-1.1 $\therefore y = f(-4)$	-0.5 $\therefore y = f(-3)$	-0.1	0.2	0.6	0.9
Actual Label	-1	-1	1	-1	1	1

If threshold set to be  $y=0$ ,  
N3, P2, P3 will be taken as +1  
P1, N2, N1 will be taken as -1



$\hat{P}$ (predicted)	$\hat{N}$ (predicted)	
P (actual)	TP = 2	FN = 1
N (actual)	FP = 1	TN = 2

# Evaluation Metrics

## Classification

Prediction function  $y = f(x)$

\*\*

We can change the threshold!

sample	<b>N1</b>	<b>N2</b>	<b>P1</b>	<b>N3</b>		<b>P2</b>	<b>P3</b>
input x	-4	-3	-2.5	-2		-1.5	-0.5
Prediction y	-1.1	-0.5	-0.1	0.2		0.6	0.9
Actual Label	<b>-1</b>	<b>-1</b>	<b>1</b>	<b>-1</b>		<b>1</b>	<b>1</b>

$y=0.4$

$\leftarrow$   $\begin{matrix} N \\ (-ve) \end{matrix}$        $\begin{matrix} P \\ (+ve) \end{matrix} \rightarrow$

If threshold set to be  $y=0.4$ ,  
P2, P3 will be taken as +1  
N3, P1, N2, N1 will be taken as -1

\* Results in different confusion matrix

$\hat{P}$ (predicted)	$\hat{N}$ (predicted)
P (actual)	N (actual)
$P_2, P_3$ $TP = 2$	$P_1$ $FN = 1$
	$N_1, N_2, N_3$ $TN = 3$
$FP = 0$	

# Evaluation Metrics

## Classification:

\*\*\*\*

TP, FP, FN, TN will change wrt thresholds!

If threshold set to be y=0,  
 N3, P2, P3 will be taken as +1  
 P1, N2, N1 will be taken as -1

If threshold set to be y=0.4,  
 P2, P3 will be taken as +1  
 N3, P1, N2, N1 will be taken as -1

	$\hat{P}$ (predicted)	$\hat{N}$ (predicted)
P (actual)	$TP = 2$	$FN = 1$
N (actual)	$FP = 1$	$TN = 2$

	$\hat{P}$ (predicted)	$\hat{N}$ (predicted)
P (actual)	$TP = 2$	$FN = 1$
N (actual)	$FP = 0$	$TN = 3$

∴ The confusion matrix is a function of thresholds, if  
 the threshold changes, confusion matrix also changes.

# Evaluation Metrics

## Classification

### Confusion Matrix for Multicategory Classification

	$P_{\widehat{1}}$ (predicted)	$P_{\widehat{2}}$ (predicted)	...	$P_{\widehat{C}}$ (predicted)
$P_1$ (actual)	$P_{1,\widehat{1}}$	$P_{1,\widehat{2}}$	...	$P_{1,\widehat{C}}$
$P_2$ (actual)	$P_{2,\widehat{1}}$	$P_{2,\widehat{2}}$	...	$P_{2,\widehat{C}}$
:	:	:	...	:
$P_C$ (actual)	$P_{C,\widehat{1}}$	$P_{C,\widehat{2}}$		$P_{C,\widehat{C}}$

A red oval highlights the diagonal elements of the matrix, labeled "rightly classified samples".

# Other Issues

- Computational speed and memory consumptions are also important factors
  - Especially for mobile or edge devices
- Other factors
  - Parallelable, Modularity, Maintainability
- Not focus of this module

# Practice Question

Suppose we have a dataset of 550 samples. We take out  $n$  samples as test set, and run k-fold cross validation on the remaining samples.

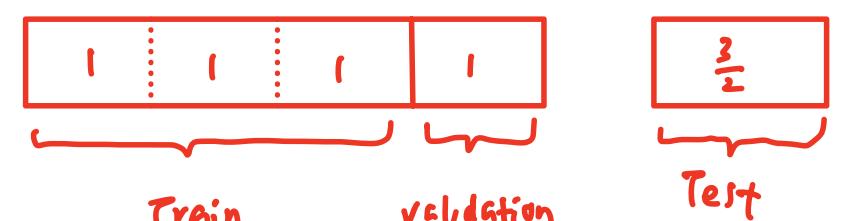


Within each fold, we know that, the number of training samples is three times as large as the number of validation samples, and two times as large as the number of test samples.

1. What is  $k$ ? [4]

2. What is  $n$ ? [150]

$$\begin{aligned}\therefore \text{Total boxes } x: \\ 3x + 1.5x + 1x = 5.5x \\ \text{One box } x: \frac{550}{5.5} \\ &= 100 \\ \text{Test} &= 100 \times 1.5 \\ &= 150\end{aligned}$$



$\therefore$  Since training must be 3 times validation, use  $k=4$

$$\begin{aligned}\text{Train set} &= 2 \text{ Test set} \\ 3 &= 2 \text{ Test set} \\ \text{Test set} &= \frac{3}{2}\end{aligned}$$

