

# Model Selection

อ. ปรัชญ์ ปิยะวงศ์วิศาล

Pratch Piyawongwisal

# Today

- Recap – kNN
- Hyperparameter Tuning
- Evaluation Metrics
- Cross-Validation
- Model Complexity
- Bias-Variance Tradeoff
- Lab: Cross-Validation for kNN

# KNN Summary

```
def train(train_images, labels):  
    # memorize training data  
    return model  
  
def predict(model, test_images):  
    # compute distance and find  
    # nearest neighbors  
    return test_labels
```

## Training Phase

แค่จำข้อมูล training data ทั้งหมดไว้

## Inference Phase

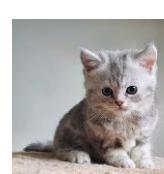


?

Test Image

Find top k matches

หาภาพที่มีหน้าตาใกล้เคียงที่สุด  $k$  ภาพ  
(คำนวณ L1/L2 distance)



CAT



CAT



DOG



CAT

แล้วทำนาย label ตาม:

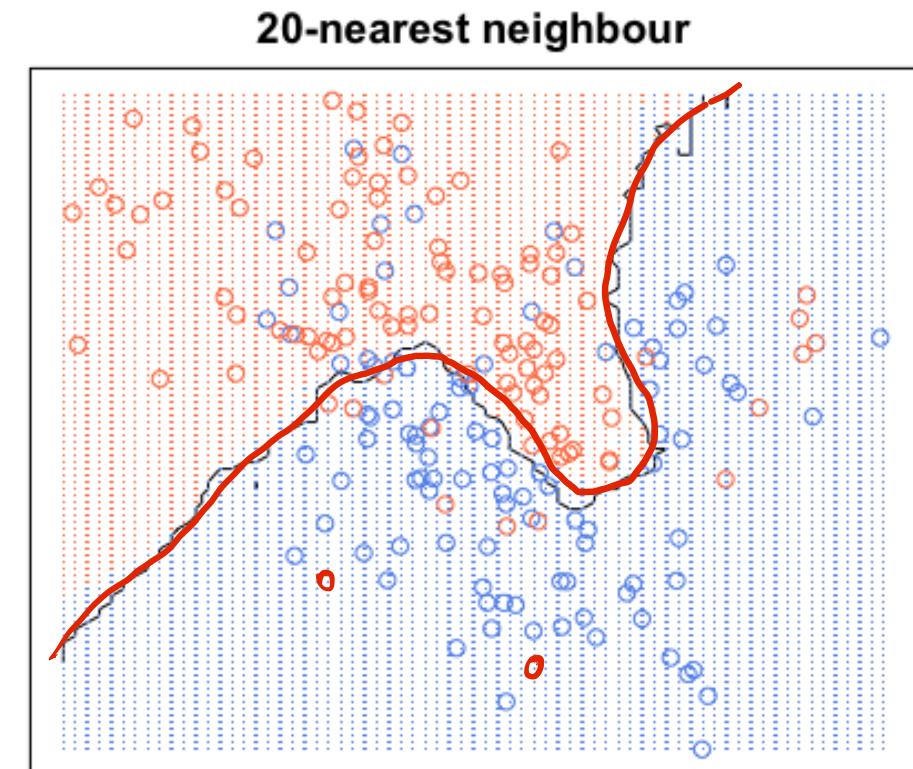
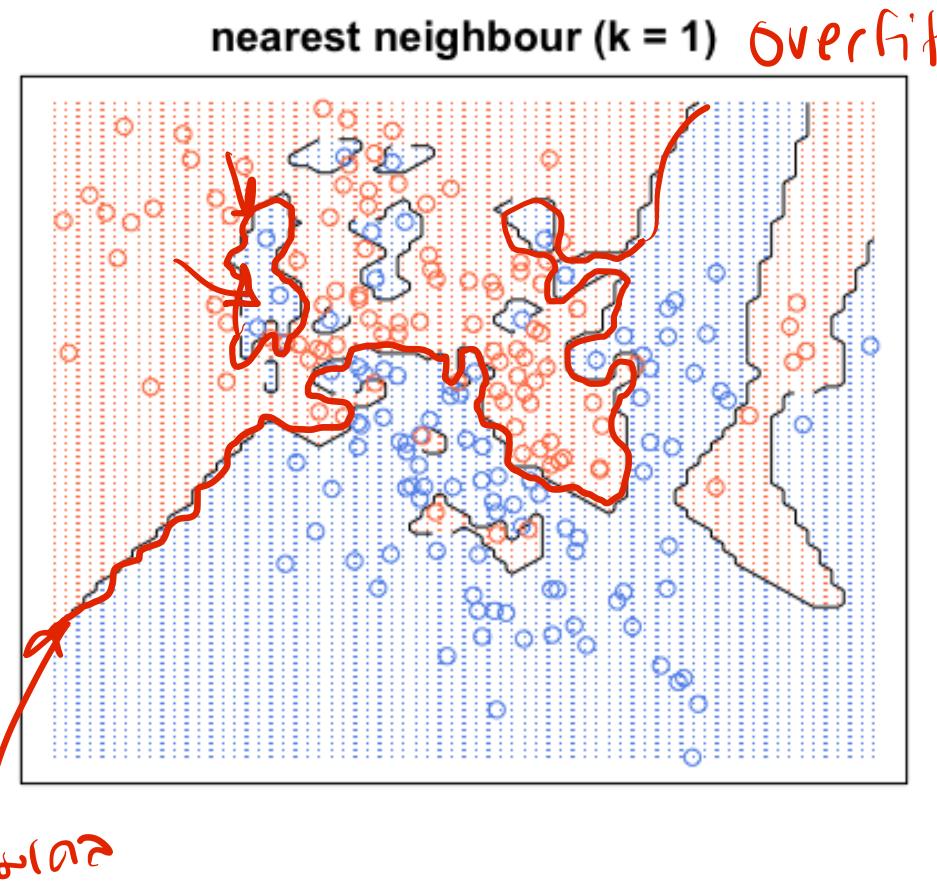
- เสียงส่วนใหญ่ หรือ
- ค่าเฉลี่ยถ่วงน้ำหนัก \*

→ Predict: CAT

\* เช่น ใช้ระยะทางเป็น weight เพื่อให้เพื่อนบ้านที่ใกล้มือทิพมากกว่าเพื่อนบ้านที่ไกล

# Hyperparameter Tuning: How to choose the best k?

- ค่าของ  $k$  เป็น hyperparameter ที่เราเลือกปรับได้ แต่เราจะเลือกค่าที่ดีสุดได้อย่างไร?



# Evaluation Metrics

- ตัวชี้วัดประสิทธิภาพการทำงานของ Machine Learning algorithm
  - Accuracy
  - Training Error
  - Testing Error
  - Confusion Matrix
  - Precision, Recall, F1 score
  - Logarithmic Loss (ค่าบล็อกฯ ไป)
  - Mean squared error
  - ROC/AUC

# Evaluation Metrics

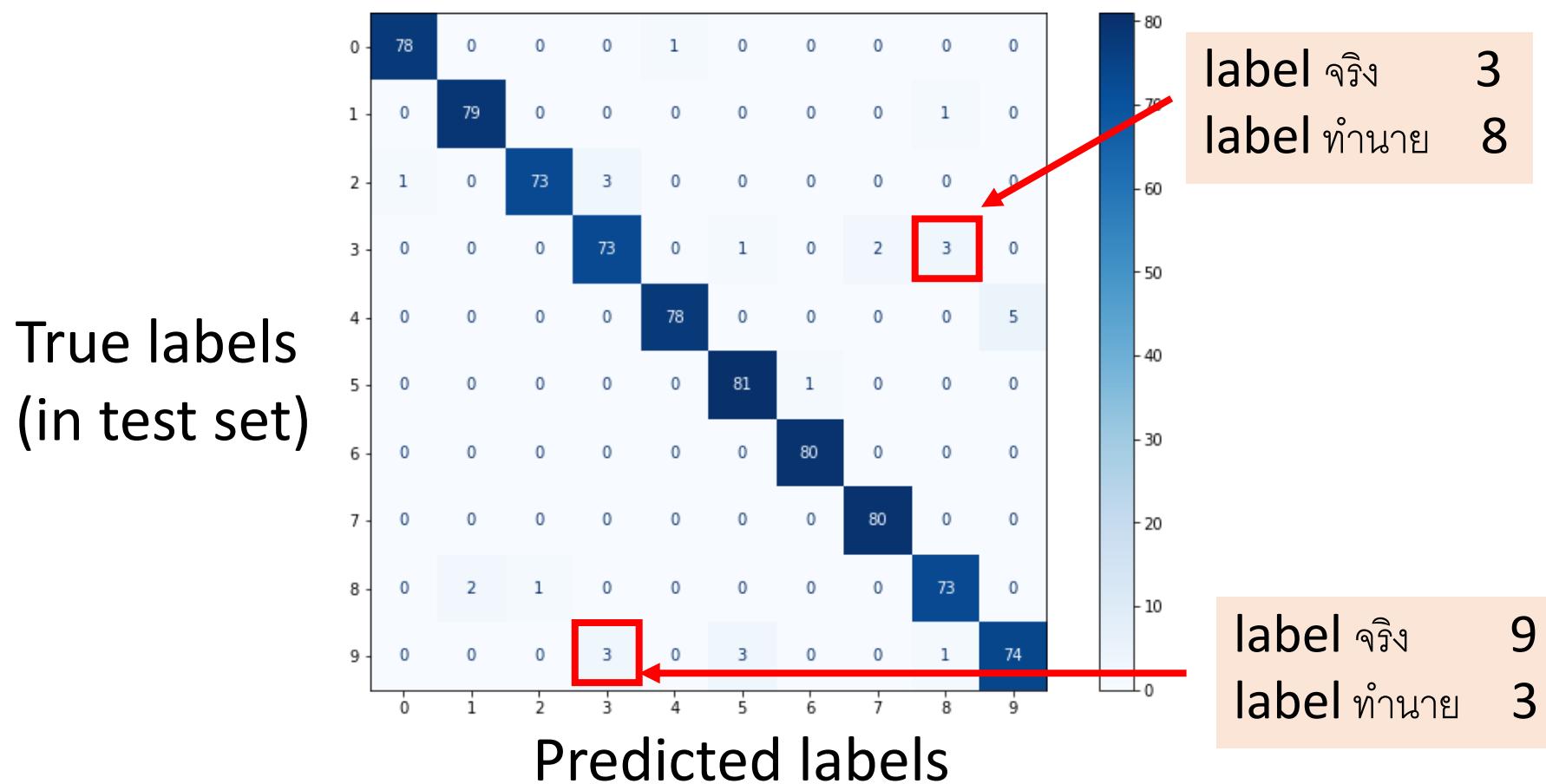
- ตัวชี้วัดประสิทธิภาพการทำงานของ **Machine Learning algorithm**
  - Accuracy =  $\# \text{ครั้งที่预言ถูก} / \# \text{ครั้งที่预言ทั้งหมด}$  (ตรงข้ามกับ error)
  - Training Error =  $\# \text{ข้อมูลชุด train ที่预言ผิด} / \# \text{ข้อมูลชุด train} \times 100\%$
  - Testing Error =  $\# \text{ข้อมูลชุด test ที่预言ผิด} / \# \text{ข้อมูลชุด test} \times 100\%$

**\*\*Important\*\***

เราให้ความสำคัญกับ training error หรือ testing error มากกว่ากัน?

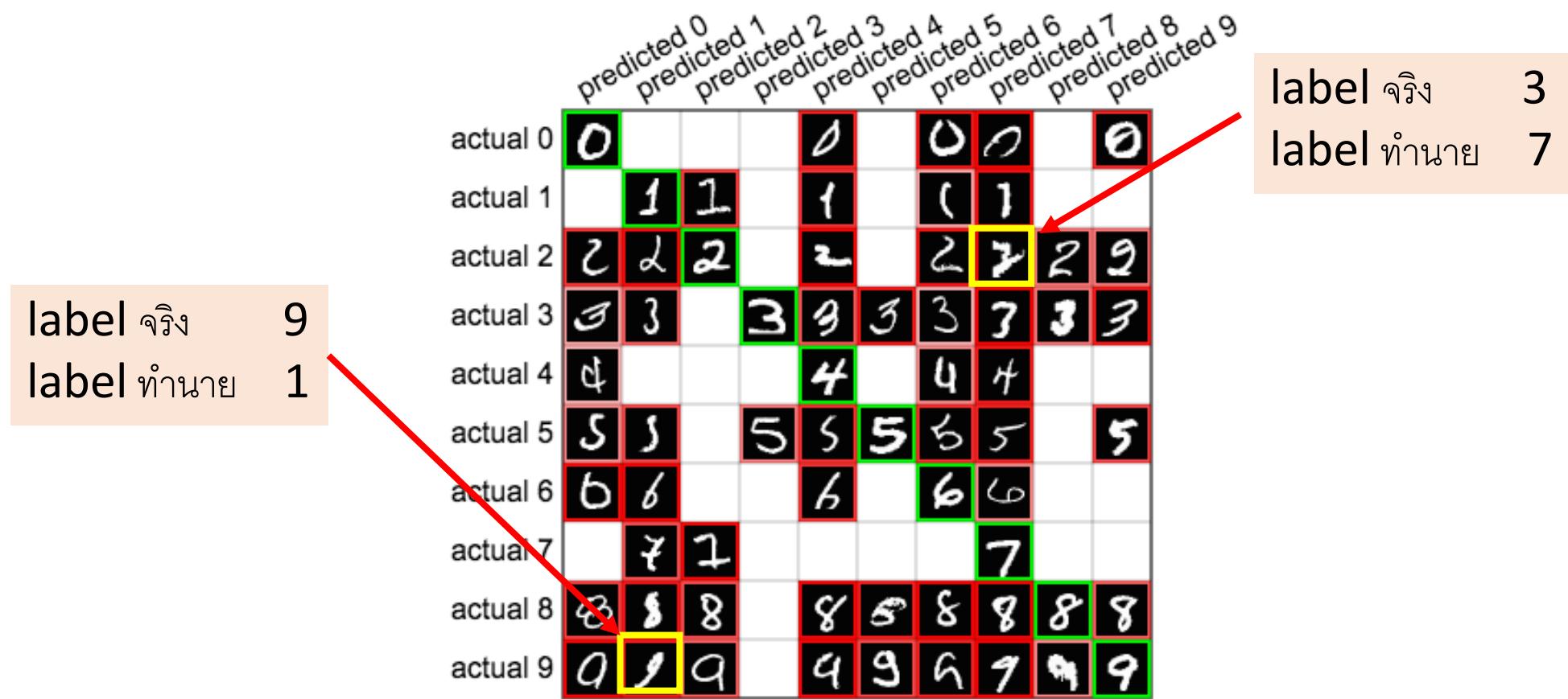
# Confusion Matrix

- ใช้แสดงอัตราการทํานายถูกผิดของแต่ละค่า **label** ระหว่างทุกคู่ **prediction** กับ **ground truth**



# Confusion Matrix

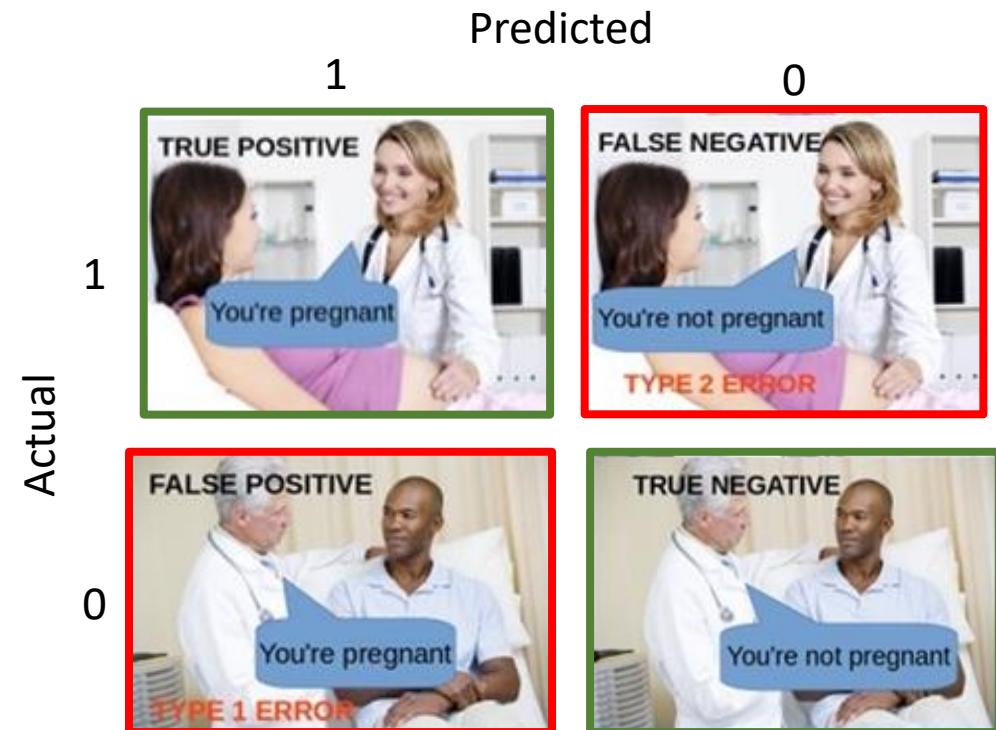
- ตัวอย่างภาพจาก MNIST dataset



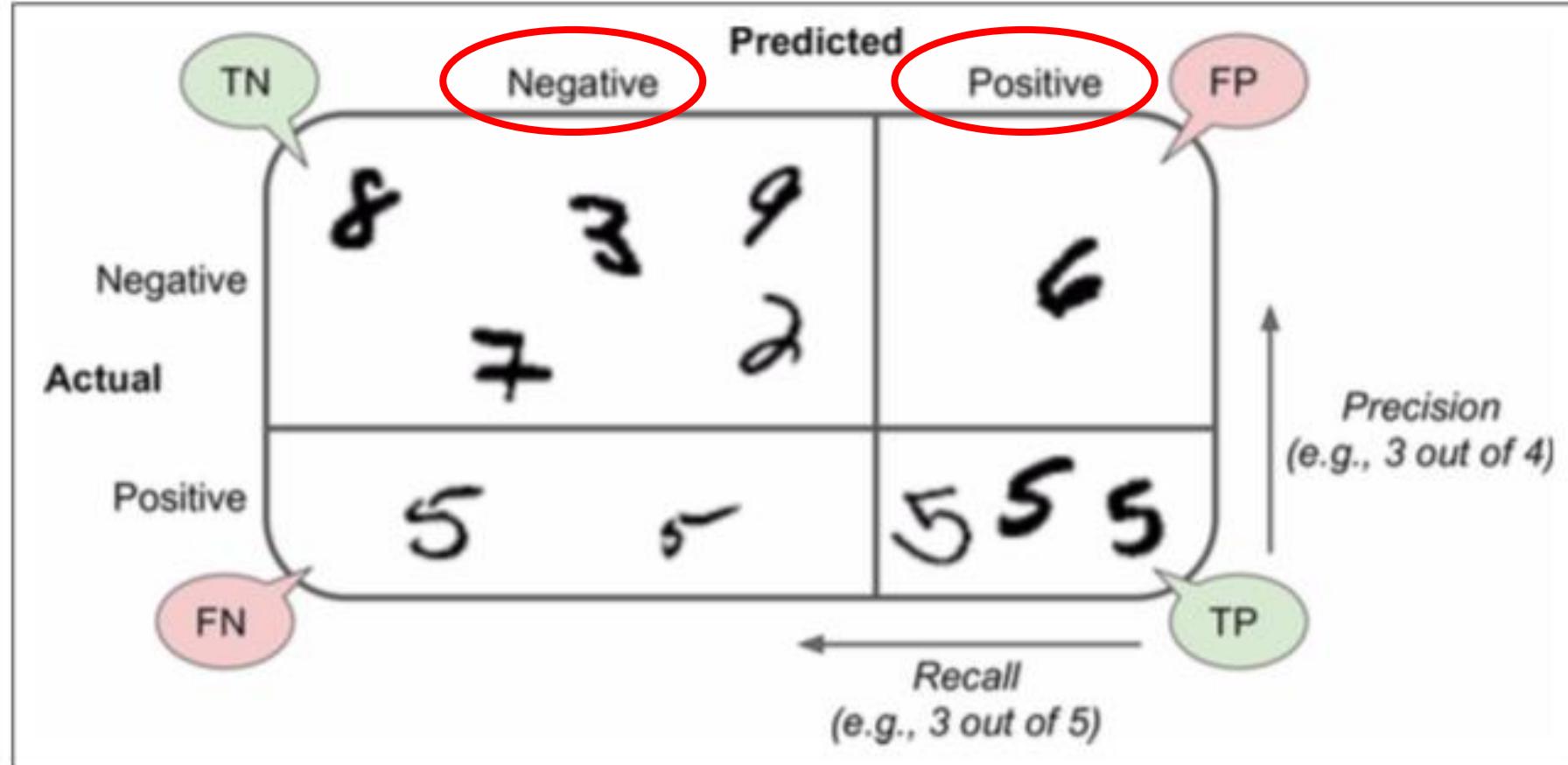
# Confusion Matrix (Binary Classification)

- พิจารณา confusion matrix สำหรับการทำนายที่มีเพียง 2 คลาส ( เช่น มะเร็ง / ไม่มะเร็ง )

		Predicted Class	
		Positive	Negative
Actual Class	Positive	True Positives (TP)	False Negatives (FN)
	Negative	False Positives (FP)	True Negatives (TN)



ระวัง: บางเจ้าอาจจะเขียน confusion matrix โดยให้ negative มา ก่อน positive ได้ ดังรูปนี้



# Precision, Recall, Accuracy

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

ในบรรดา sample ที่ระบบเรา  
ตรวจจับ เป็น + จริงมากแค่ไหน ?

		Predicted Class	
		Positive	Negative
Actual Class	Positive	True Positives (TP)	False Negatives (FN)
	Negative	False Positives (FP)	True Negatives (TN)

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

ในบรรดา sample ที่เฉลยเป็น +  
ระบบเราตรวจจับเจอมากแค่ไหน ?

$$Accuracy = \frac{TP + TN}{Total}$$

# Exercise: Precision, Recall

- พิจารณา **confusion matrix** ของระบบตรวจจับวัตถุระเบิดที่สนามบิน
  - จำนวนหา Precision, Recall, Accuracy
  - ระบบดังกล่าวเหมาะสมที่จะนำไปใช้งานจริงหรือไม่?

ทบทวน

	Predicted Y +	Predicted N
Actual Y	150 <b>TP</b>	100 <b>FN</b>
Actual N	1 <b>FP</b>	249 <b>TN</b>

$$Precision = \frac{TP}{TP + FP} = \frac{150}{150+1} = 0.993$$

$$Recall = \frac{TP}{TP + FN} = \frac{150}{150+100} = 0.6$$

$$Accuracy = \frac{TP + TN}{Total} = \frac{150+249}{500} = 0.798$$

# Exercise: Precision, Recall

- พิจารณา **confusion matrix** ของระบบตรวจจับวัตถุระเบิดที่สนามบิน
  - จำนวนหา Precision, Recall, Accuracy
  - ระบบดังกล่าวเหมาะสมที่จะนำไปใช้งานจริงหรือไม่?

	Predicted Y	Predicted N
Actual Y	150	100
Actual N	1	249

$$Precision = \frac{150}{150 + 1} = 0.993$$

$$Recall = \frac{150}{150 + \underline{100}} = 0.6$$

$$Accuracy = \frac{150 + 249}{150 + 100 + 1 + 249} = 0.798$$

# Exercise: Precision, Recall

- พิจารณา **confusion matrix** ของระบบตรวจจับวัตถุระเบิดที่สนามบิน
  - จำนวนมาตรา **Precision, Recall, Accuracy**
  - ระบบดังกล่าวเหมาะสมที่จะนำไปใช้งานจริงหรือไม่?

	Predicted Y	Predicted N ทำหง่าวไม่
Actual Y ระเบิดจริง	150	100
Actual N	1	249

$$Precision = \frac{150}{150 + 1} = 0.993$$

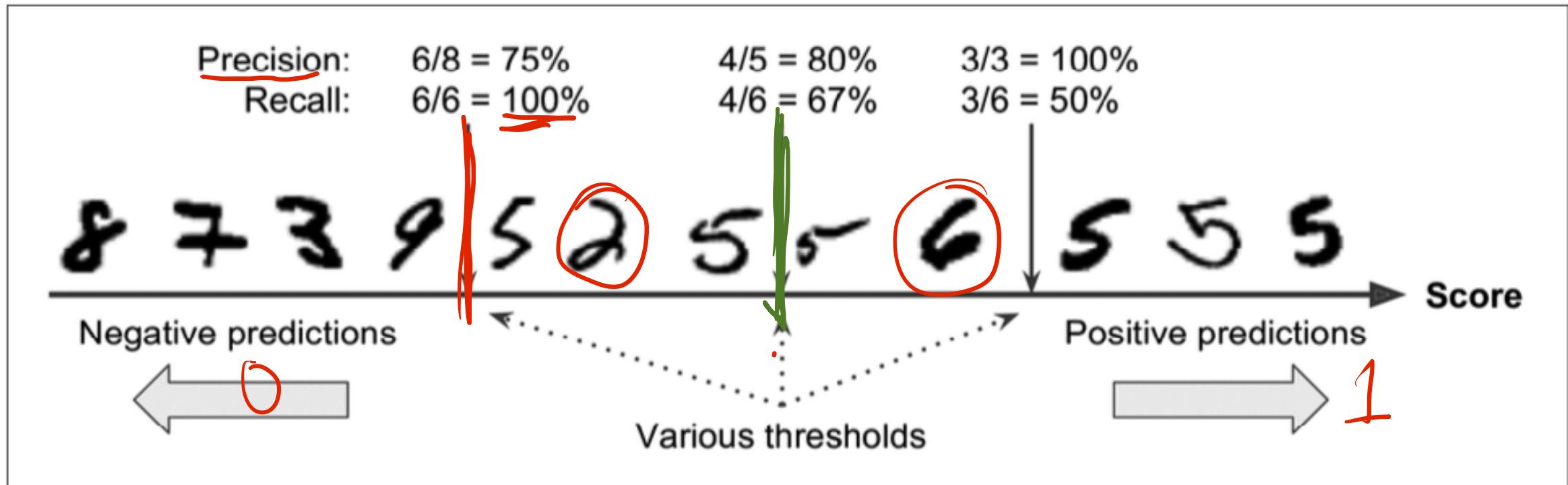
$$Recall = \frac{150}{150 + 100} = 0.6$$

$$Accuracy = \frac{150 + 249}{150 + 100 + 1 + 249} = 0.798$$

$$F_1 = \frac{2(0.993)(0.6)}{(0.993 + 0.6)}$$

ไม่สมควรนำไปใช้งานจริง เพราะ **Recall** ต่ำ -> ระเบิดหลุดลอดเข้าไปได้เป็นจำนวนมาก

# Precision vs Recall trade-off



โดยปกติ precision กับ recall มักจะสวนทางกัน

- เช่น หากปรับระบบให้ได้ precision ที่สูงไป อาจจะกระทบกับ recall (ดัง threshold ขวามาก)

# F1-Score

- เนื่องจากเราต้องการระบบที่มีทั้งค่า Precision และ Recall สูง
- จึงมีแนวคิดที่จะรวมทั้งสอง metric ให้เป็น F1-score ค่าเดียว
- F1-score เป็น harmonic mean ระหว่าง Precision, Recall ดังสมการ

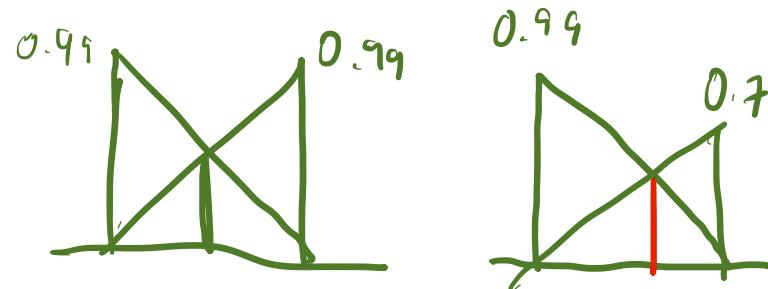
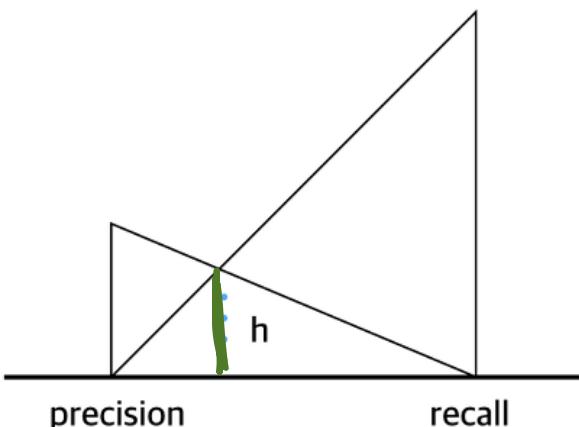
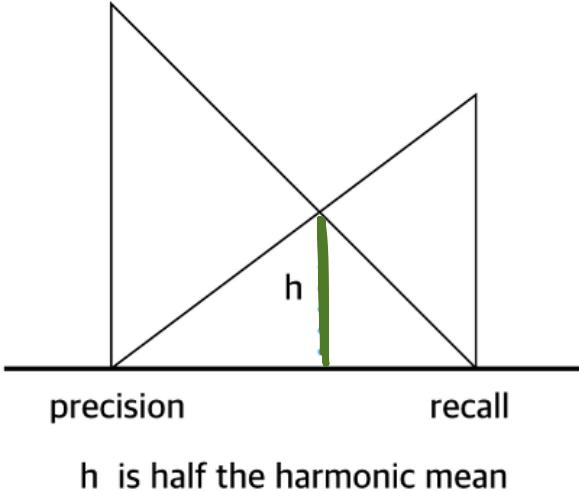
80	1
6	90

$$\begin{array}{ccc} P & r & F1 \\ 0.7 & 0.8 & \\ 0.75 & 0.75 & \end{array}$$

$$F1 = \frac{1}{\frac{1}{2} \left( \frac{1}{Recall} + \frac{1}{Precision} \right)} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

$$\frac{1}{2} \left( \frac{1}{Recall} + \frac{1}{Precision} \right) = \frac{1}{F1}$$

# Visualizing F1-Score



หาก Recall หรือ Precision ค่าใดค่าหนึ่งน้อย  
ก็จะฉุดให้ F1 ต่ำลงไป แม้วิค่าจะสูงก็ตาม

# Metrics for Multiclass Classification (ไม่ออกสอบ)

- การคำนวณ precision, recall, F1 ที่ผ่านมา สามารถใช้ได้กับ binary classifier เท่านั้น
- สำหรับ multiclass classifier (เช่น MNIST, CIFAR-10) *dog, cat, airplane, ...*
  - เราจะต้องหา TP/TN/FP/FN ของทีละคลาส เช่น dog vs NOT dog, cat vs NOT cat, ...
  - แล้วจึงนำมาคำนวณหา precision, recall รวมทุกคลาส ซึ่งทำได้ 2 วิธี คือ
    - Macro-average: นำ precision, recall ของแต่ละคลาスマเข้าด้วยกัน
$$\text{Precision}_{\text{tot}} = \frac{\text{Pre}_{\text{dog}} + \text{Pre}_{\text{cat}} + \dots}{10}$$
• ให้ความสำคัญกับแต่ละคลาสเท่าๆ กัน เหมาะกับ imbalanced data
    - Micro-average: คำนวณ precision, recall รวมโดยนำ TP/TN/FP/FN ของแคล์คลาスマบวกกันก่อนเข้าสูตร
      - ให้ความสำคัญกับแต่ละ data sample เท่าๆ กัน
$$\text{Precision}_{\text{tot}} = \frac{\sum_{c} \text{TP}_c}{\sum_{c} \text{TP} + \sum_{c} \text{FP}}$$
- ตัวอย่างการคำนวณโดยละเอียด
  - <https://www.youtube.com/watch?v=HBi-P5j0Kec>

# Hyperparameter Tuning: KNN

1, 2, 7, ..., 10

- **goal:** เรายังต้องการหาค่า  $k$  ที่ทำให้ test error ต่ำสุด

- **solution?:**

ทำการ **train, test** กับข้อมูลชุดเดิมซ้ำๆ โดยเปลี่ยนค่า  $k$  ไปเรื่อยๆ แล้วหา  $k$  ที่ทำให้ test error ต่ำสุด

Problem?

```
# ลอง k ตั้งแต่ 1...10  
for k in range(1,11):  
    classifier.train(X_train, y_train)  
    dists = classifier.compute_distances(X_test)  
    y_test_pred = classifier.predict_labels(dists, k)  
    accuracy = np.sum(y_test_pred == y_test)/len(y_test)  
    print(accuracy) # เลือก k ที่ให้ค่า test accuracy สูงสุด
```

$k=1$       0.9

$k=2$       0.93 ✓  
⋮

# Hyperparameter Tuning: KNN

KNN  $\rightarrow$  train err 0%

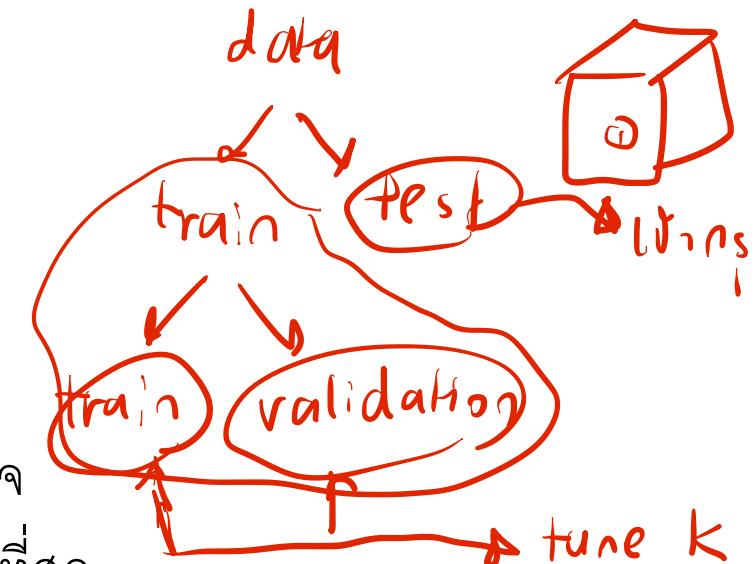
- **goal:** เรายังต้องการหาค่า  $k$  ที่ทำให้ **test error** ต่ำสุด
  - **solution?:**  
ทำการ **train, test** กับ **ข้อมูลชุดเดิม** ซ้ำๆ โดยเปลี่ยนค่า  $k$  ไปเรื่อยๆ แล้วหา  $k$  ที่ทำให้ **test error** ต่ำสุด
  - **problem:** โมเดลเราอาจจะ **overfit** กับ **ข้อมูลชุด test** นั้น
- ❖ Overfitting คือการที่โมเดล **fit** กับ **ข้อมูลชุดหนึ่งมากเกินไป** ทำให้ **generalize** กับ **ข้อมูลชุดอื่นๆ** ไม่ได้

KNN

# Cross-Validation

- better solution:

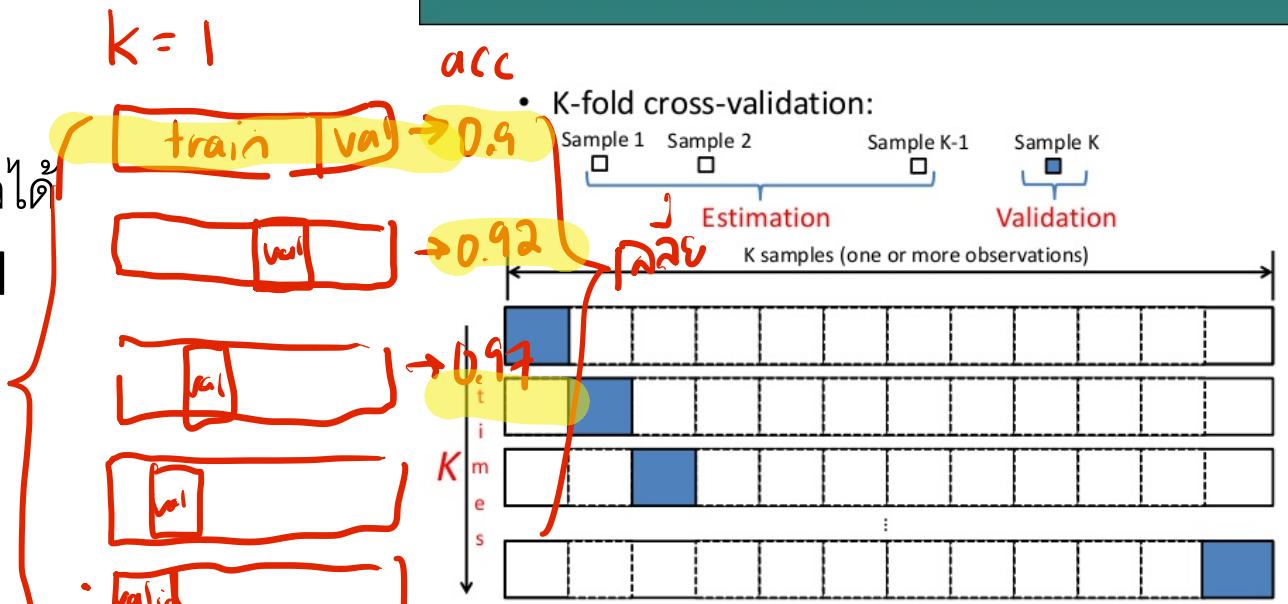
- แบ่งข้อมูลเป็น 3 ส่วน คือ train, validate, test
- แยกชุด test ออกไปเก็บไว้ในกรุ ห้ามใช้จันกว่าจะ train เสร็จ
- ใช้ชุด train, validate ใน การหาค่า hyperparam ที่ดีที่สุด
- ใช้ชุด test ในการประเมินสุดท้าย



- สามารถเลือกจำนวน fold ในการแบ่งข้อมูลได้
  - เช่น leave-one-out, 3-fold, 5-fold

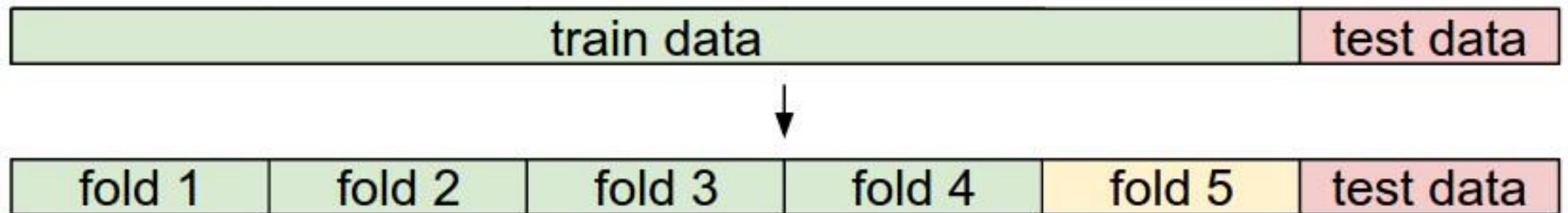
5-fold  
cross validation

## Cross-validation: How it works?



# K-fold Cross-Validation

- 5-fold example

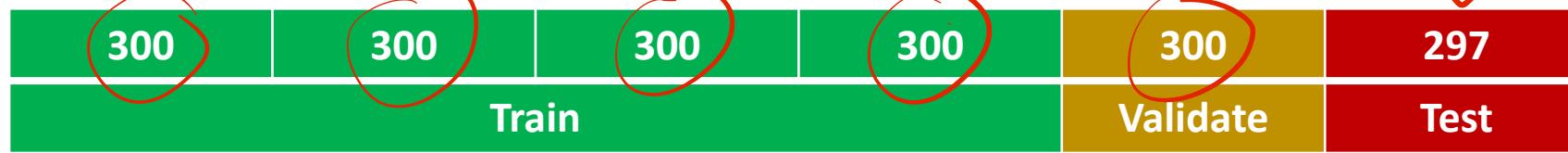


Fold 1-5 ผลักกันเป็น validate set (สีเหลือง)

# ตัวอย่าง: วิธีทำ 5-fold CV กับ MNIST, k-NN

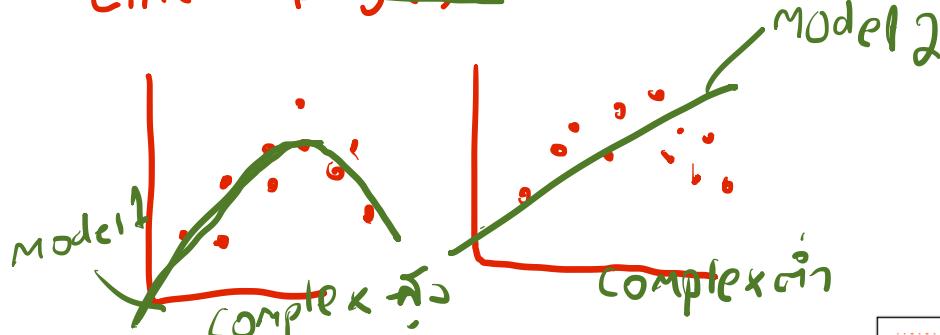
- ชุดข้อมูล digits ใน sklearn ประกอบด้วยภาพ + label จำนวน 1797 ภาพ
  - แยกข้อมูล 297 ภาพไปเป็น test set (เก็บเข้ากรุใช้ทดสอบครั้งสุดท้ายครั้งเดียว)
  - นำข้อมูลที่เหลือ 1500 ภาพไปใช้ train k-NN
- กระบวนการ train k-NN โดยละเอียด
  - วนลูปค่า k ทีละค่า เช่น จากเซ็ต {1, 3, 5, 7, 9, 11}
  - เริ่มจาก  $k = 1$ 
    - ตัดแบ่งข้อมูล 1500 ภาพเป็น 5 ชุด (fold) ชุดละ 300 ภาพ
    - เลือก 1 ชุด (300 ภาพ) เป็น validation set และให้ที่เหลืออีก 4 ชุด (1200 ภาพ) เป็น train set
    - หาค่า accuracy จากการใช้ k-NN นำนายชุด validation set
    - ทำซ้ำโดยลดให้แต่ละชุดได้เป็น validation set ==> จะได้ accuracy มา 5 ค่า
    - หากค่า accuracy เฉลี่ย
  - ทำซ้ำสำหรับ  $k=3, 5, \dots$  ต่อไปจนครบ แล้วเราจะเลือก k ที่ให้ค่า mean accuracy สูงที่สุด
  - ใช้ค่า k นั้นในการทำนาย test set (เปิดกรุ) และหา accuracy ท้ายสุด

เปิดกรุ



# Model Complexity

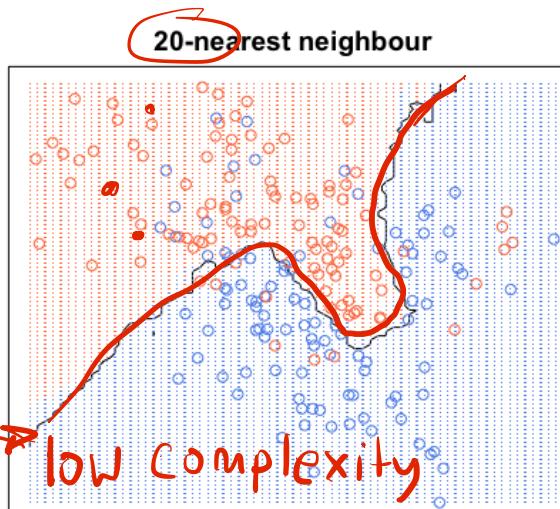
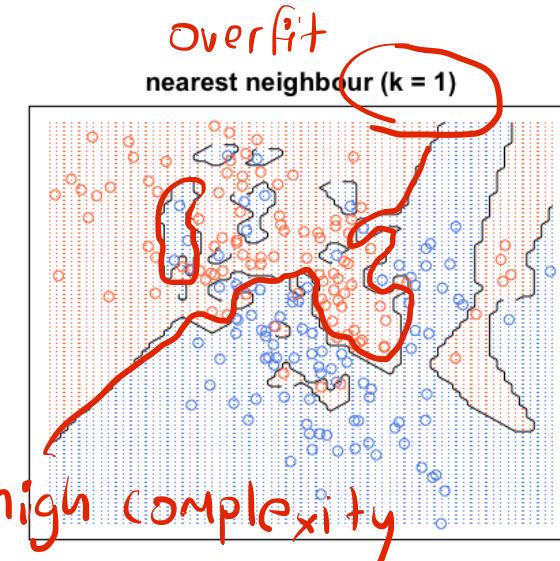
## Linear Regression



- การปรับจูนค่า hyperparameter อาจส่งผลต่อ model complexity
  - model ที่มี complexity สูง จะมี assumption เกี่ยวกับข้อมูลที่มาก
    - เช่น 1-NN
  - model ที่มี complexity ต่ำ จะมี assumption ที่ผ่อนปรนกว่า
    - เช่น 20-NN

❖ สร้างเกตได้จากการความซับซ้อนยุ่งเหยิงของ **decision boundary**

❖ **model complexity** มีผลต่อความสามารถในการ **generalize** ของโมเดล



# Bias-Variance Tradeoff

笔记: my bias variance  
in OneNote 15:00

- Assuming true relationship is  $Y = f(x) + \varepsilon$ 
  - where noise  $\varepsilon \sim N(0, \sigma^2)$
- We want to make a model  $\hat{f}(x; D)$  of  $f$  using  $D$  as the training samples
- Expected squared test error is  $Err(x) = E_D[(Y - \hat{f}(x))^2]$
- which can be decomposed as

$$Err(x) = \left( E_D[\hat{f}(x)] - f(x) \right)^2 + E_D \left[ (\hat{f}(x) - E[\hat{f}(x)])^2 \right] + \sigma^2$$

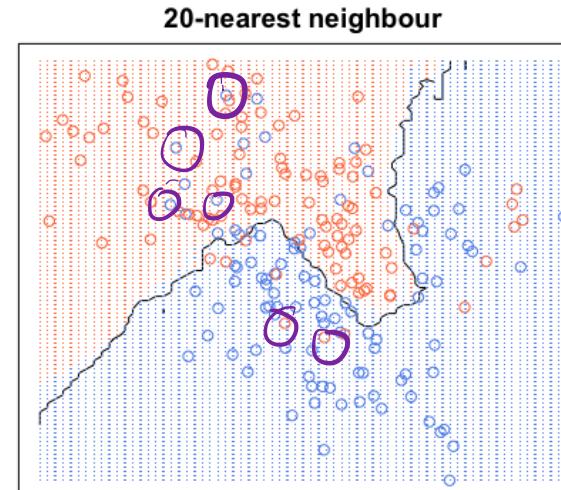
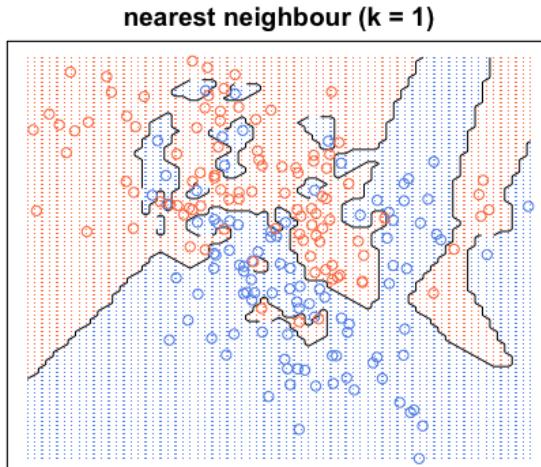
Bias

Variance

Irreducible  
Error

# Bias-Variance Tradeoff (อธิบาย)

- ด้วยทฤษฎีทางสถิติ พบว่าเราสามารถตัด generalization error (expected test error) ของโมเดลออกเป็นจาก 3 แหล่ง ได้แก่
  - bias = ทำนายผิด เพราะ model มี assumption เกี่ยวกับข้อมูลที่ผิด เรียบง่ายไป (complexity ต่ำไป) ทำให้เกิดการ underfit (ภาพขาว)
  - variance = ทำนายผิด เพราะ model อ่อนไหวต่อ variation ในข้อมูลเกินไป จึงเกิด overfit (ภาพซ้ำ)
  - irreducible error = ทำนายผิด เพราะ noise ในข้อมูลตามธรรมชาติ จะปรับโมเดลอย่างไรก็ลดไม่ได้



ห์นๆ นๆ 20:  
high bias 😞  
low variance ☺

key takeaway : រួចកែវា hyperparam នៃការងារ នូវការ/ដំឡើង

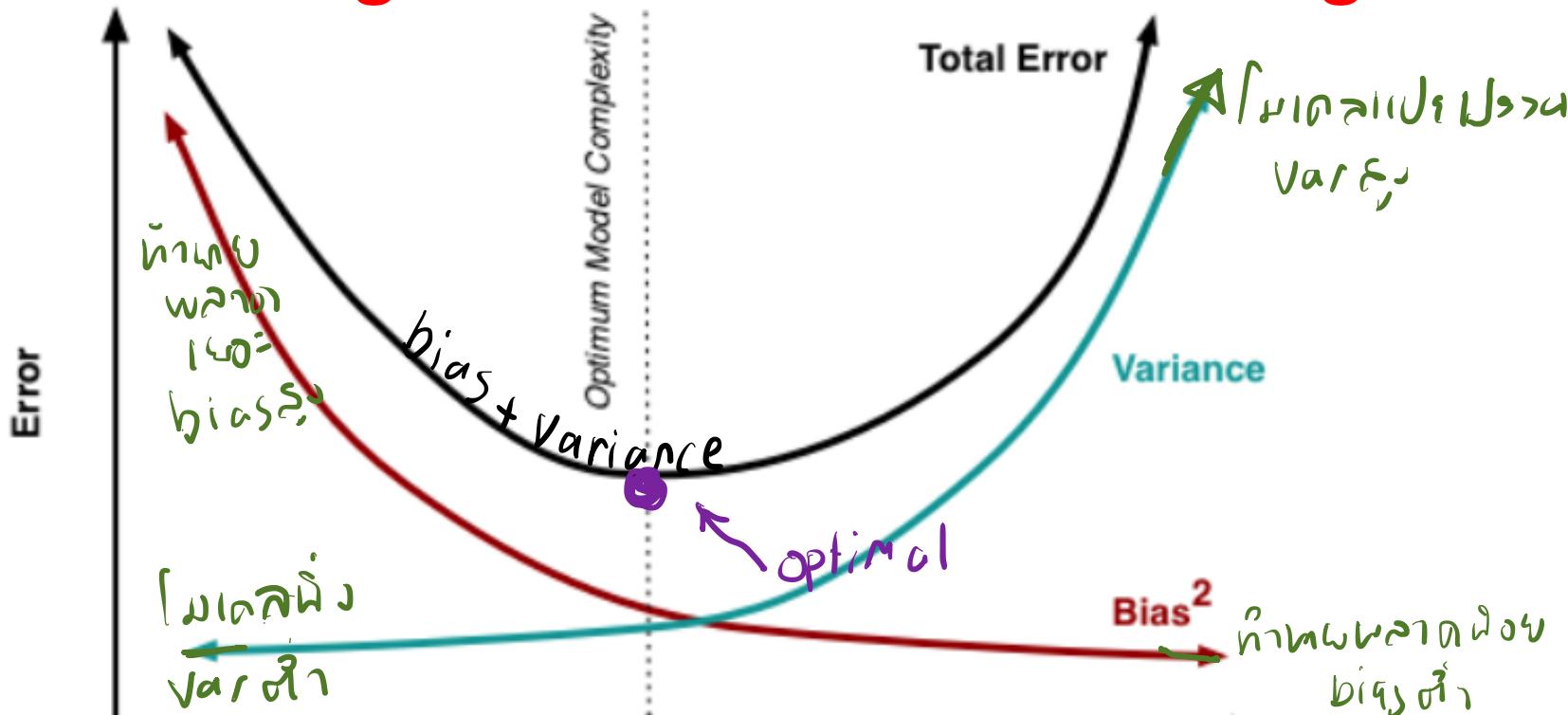
# Bias-Variance Tradeoff

$k$  ត្រឹម → underfit (varធំ / Biasធ្មោ)

$k$  ខ្សោយ → overfit (biasធ្មោ / Varធំ)

underfitting

overfitting



| 20-NN is here

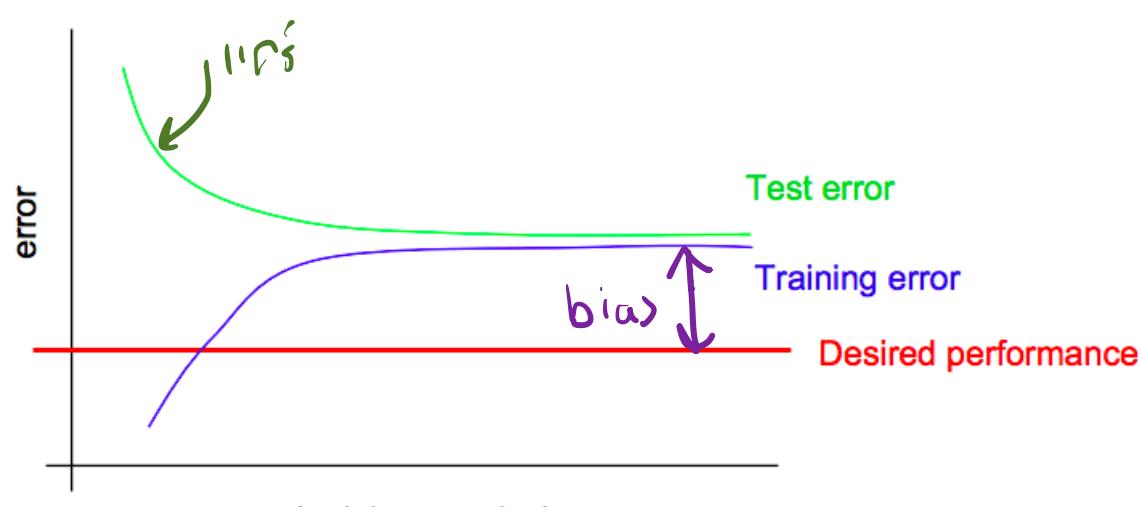
←  
 $k$  មាត្រា (kNN)

1-NN is here

# Learning curve

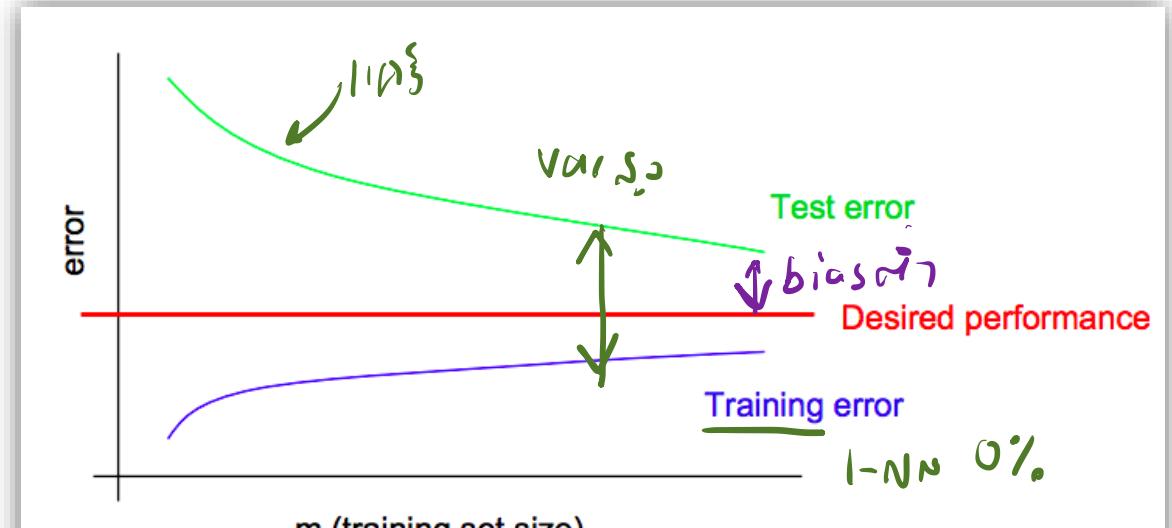
- Accuracy vs Training size plot

underfit

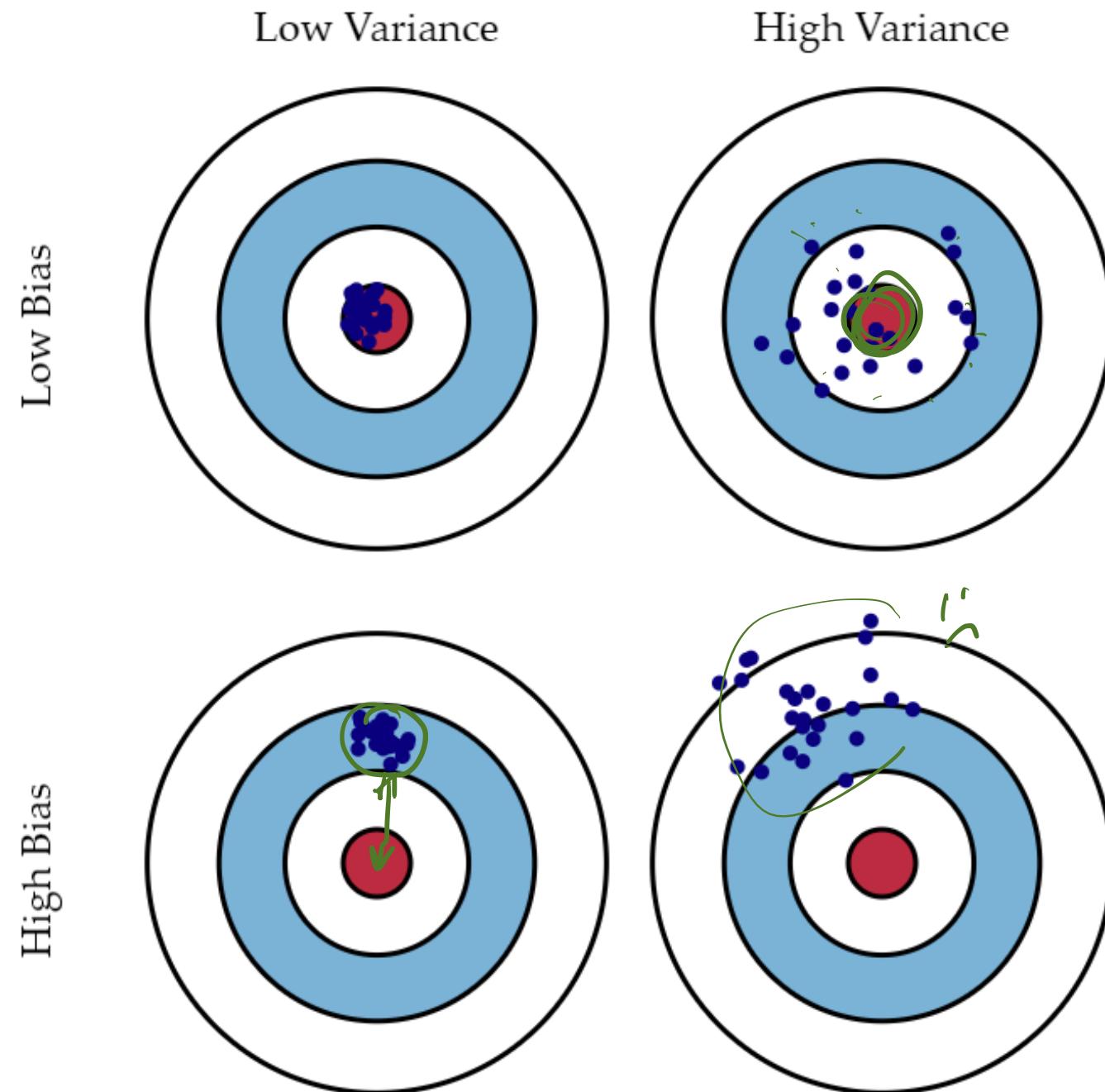


High Bias

Overfit



High Variance



# Lab: Cross-validation for kNN

- ทำการ implement cross-validation เพื่อหาค่า  $k$  ที่เหมาะสม
- ลอง  $k = 1, 3, 5, 8, 10, 12, 15, 20, 50, 100$
- กำหนด `num_folds` เป็นจำนวน fold
- ให้ `num_folds = 5`

300	300	300	300	300	297
Train			Validate	Test	

# Extra: การ implement kNN ด้วย scikit-learn

- โหลดข้อมูล MNIST ซึ่งประกอบด้วย
  - images ภาพตัวเลข
  - target เฉลย
- แบ่งข้อมูลเป็นชุด train ชุด test
- สร้าง KNeighborsClassifier
- ทำการ train โดยเรียก `classifier.fit(ชุด train)`
- ทำการ test โดยเรียก `classifier.predict(ชุด test)`
- เปรียบเทียบผลการ `predict` กับเฉลยใน `target` และประเมินประสิทธิภาพ
  - $\text{accuracy} = \frac{\text{จำนวนชุด test ที่ถูกต้อง}}{\text{จำนวนชุด test}} / \text{จำนวนชุด test}$
  - ใช้ `metrics.confusion_matrix(เฉลย, ทาย)` เพื่อหาว่าทำนายผิดอย่างไร

# Summary - 1

`gridsearchCV( )`

- ใน ML algo เราสามารถปรับค่า **hyperparam** เช่น ถ้าเป็น k-NN ก็จะมี k (1,2,3,...) หรือ distance (L1, L2)
- เวลาเขียนโค้ด เพื่อลองแต่ละค่า k เราจะทดลอง predict กับข้อมูลชุด validate set แล้วคำนวณหา eval metric คือตัวชี้วัดความดี เช่น accuracy, precision, recall, F1
  - Recall สูง = สามารถตรวจจับคลาสนั้นเจอกีบหมด ไม่มีหลุดลอด (False Negative ต่ำ)
    - เช่น ภาพ dog มี 100 ภาพ เรา classify เป็น dog 98 ภาพ
  - Precision สูง = การตรวจจับคลาสนั้น ไม่ค่อยพลาดไปหมายภาพคลาสอื่น (False Positive ต่ำ)
    - เช่น เรา classify 110 ภาพเป็น dog ในนั้นเป็น dog จริงๆ อยู่ 98 ภาพ
  - F1 สูง = ทั้ง Precision และ Recall สูง

# Summary - 2

- ในการทดลอง ควรทำ **k-fold cross validation** เพื่อป้องกันไม่ให้ผลที่ได้ขึ้นกับข้อมูลชุดใดชุดหนึ่ง
  - โดยผลัดให้ทุกส่วนของข้อมูล **train** ได้มีโอกาสเป็น **validate set**
- ปกตินิยมทำ **5 fold** คือ ตัดข้อมูลเป็น **fold** ละจำนวนเท่าๆ กัน และให้ **1 fold** เป็น **validate set** และอีก **4 fold** ที่เหลือเป็น **train set**
  - ผลัดกันเป็น **validate set** ได้ **5** แบบ เกิดเป็นข้อมูล **5** ชุด
  - **predict** กับข้อมูล **5** ชุดนี้ ก็จะได้ผล **accuracy** มา **5** ค่า
  - นำค่า **accuracy** **5** ค่า มาเฉลี่ยกัน

# Summary - 3

- ค่า  $k$  มีผลต่อ **model complexity** (ความซับซ้อนของเส้นแบ่งแทน)
  - 1 NN => เส้นแบ่งแทนดูซับซ้อนสูง => พยายามบิดไปตามข้อมูล **train** (100% accuracy on train) => overfitting => ใช้กับข้อมูลชุดอื่นแล้วไม่ดี (low accuracy on test)
  - 20 NN => เส้นแบ่งแทนดูเรียบง่ายกว่า => train accuracy จะต่ำกว่า 1-NN แต่เรามีเครื่อง => ใช้กับข้อมูลชุดอื่นได้ดีกว่า (higher accuracy on test) คือดี => แต่  $k$  มากไปก็อาจทำให้ underfit ก็ไม่ดี
- สรุป ควรเลือกค่า  $k$  ที่ไม่มาก ไม่น้อยไป

decision tree : hyperparam :  
- <sup>max</sup> depth 2, 3, 5, ... overfit  
- criterion GINI, Entropy

neural net : hyperparam - # of layers, # parameters, dropout

# Next week

- Linear Regression
- Gradient Descent
- Logistic Regression