모델 평가, 튜닝, 선정

모델이 out-of-sample 데이터에 대해 잘 해야 할텐데...

- 어떻게 미리 알 수 있을까?
- 좋은 방법이나 절차는?
- Regression과 Classification때 다를까?
- 더 나은 모델을 선택하던가 만들려면 어떻게 해야 하지?

- **out-of-sample 데이터**: Training Set에 있지 않은 데이터
  (예, Test Set & Validation Set)

용어

- 문맥으로 Training 인지 Test 인지 명확하면 굳이 Training/Test를 붙이지 않음

■ **Training Set** : 모델을 학습시킬 때 사용한 데이터

- **Training Performance** : Training set로 모델을 평가했을 때의 성능
  - Training Accuracy, Training Error Rate, Training Sensitivity 등

■ **Validation Set** : Test Set이 특별히 준비되지 않았을 때 Test Set 역할을 하도록 하여 Test Performance를 추정하고, 이를 이용해 hyper parameter를 튜닝하거나 , 모델을 선택할 때 쓰는 데이터 (자주 사용하는 용어 아님)

■ **Test Set** : 학습된 모델을 평가/시험할 때 사용하는 out-of-sample 데이터

- **Test Performance** : Test set로 모델을 평가했을 때의 성능
  - Test Accuracy, Test Error Rate, Test Sensitivity 등
- **Field Performance** : 실제 사용할 때의 성능

우리가 알고 있는 것을 정리하면,

- Regression의 response는 숫자. 모델이 예측하는 response도 숫자
- C_p, AIC, BIC, Adjusted R-squared 등으로 Test 성능(에러)을 가늠해 본다지만, 이들은 기본적으로 Training 성능을 바탕으로 복잡한 모델에게 마이너스를 주어 (이 외 가정을 추가) 보정해 간접적으로 Test 성능을 추정. 그리 믿음 안감.
  - C_p, AIC, BIC : 작을 수록 좋음. Observation 개수 n이 클수록 작아지고 (좋은 모델), 사용한 predictor 개수 d, RSS, variance_of_error가 커질수록 증가 (나쁜 모델). n<p 이면 사용 못함
  - BIC는 C_p , AIC와 비슷하나, BIC가 AIC보다 predictor가 많은 모델에 대해 더 마이너스를 많이 주어 BIC가 상대적으로 C_p , AIC 보다 predictor가 작은 모델을 더 선호함
  - Adjusted R^2 : 클수록 좋음. variance_of_error 가 필요하지 않고, 간단하고, 이론적으로 n<p 일 때도 적용가능하나 실제로는 효용성이 의심됨
  - C_p, AIC, BIC, Adjusted R-squared : 모두 n , d와 RSS가 있어야 함. 없거나 모르면 문제
- Classification의 response는 클래스_래이블. 모델의 response를 클래스_래이블로 할 수도 또는 클래스_확률로 할 수도 있다.
- Classifier의 response를 확률로 가져오면, 판별임계값 (discriminating threshold)를 적용해서 클래스 래이블 response를 튜닝할 수도 있다.

좋은 방법이 생각났다 – (실은, 선사시대부터 쓴 방법)

1. 잘 모르는 물건이 있으면 일단 두들겨본다

2. 다양하게 두들겨본다. 화끈하게 두들기면 화끈하게 자기의 본 바탕을 다 드러낼지 모르고 (Impulse response, 아니면 죽거나), 요리 두들기면 저런 반응을 보일 지도. 여하간 배울 것이 있다.

3. 우린 지금 (학습된) 모델이 앞으로 out-of-sample 데이터에 대해 얼마나 적절한 response를 낼 것인가를 짐작하고 싶다.

4. **우리에겐 labeled data (True response)가 있다. 이 데이터를 training에 다 쓰지 말고, 일부를 남겨 (hold-out) 이 것들을 test 데이터라 간주해 이 것에 대해 모델을 시험하자** – 예쁜 공식이나 방법이 없으면(이런 경우가 더 많음) 현재 갖고 있는 자원을 활용해 최대한 실제 사용 환경과 비슷하게 만들고 돌려본다. 그리고 모델이 뱉은 response와 진짜 response가 어떻게 다른가 본다. **단순한 시뮬레이션**

5. 갖고 있는 데이터를 최대한 이용해 다양한 시뮬레이션을 해보자 - Training/Test set 나누는 것을 다양하게 만들어 돌려봐 필드에서 진짜 당황스런 일이 생기지 않도록 노력한다.

Regressor 평가 방법

책에서는 Validation Set 방법이라 함.
보통은 Validation Set도 그냥 Test Set이라 함

1. **Train/Test Split** : 데이터를 겹치지 않는 두 집합으로 나눈다. 하나는 training에 쓸 training set. 다른 하나는 test set. Test set은 training set으로 학습된 모델을 평가할 때에 쓸 것으로 모델을 training/학습 시킬 때 전혀 관여하지 않는다. (Test set의 역할이 학습된 모델이 앞으로 보게 될 out-of-sample 데이터 이기에)

2. **Cross Validation :** Train/Test split 평가는 어떤 데이터가 Train/Test set에 속하게 되었는가에 따라 달라진다. 이에 따른 variance를 낮추려면 **겹치지 않는** Train/Test split 평가를 여러 번 하면 되겠다. 이 개념이 Cross Validation

- **Regression 평가 지표 (metric)** : Train/Test Split 이나 Cross validation을 할 때 모델의 성능/정확성이 어떤 가를 측정하는 정량적인 지표가 필요하다. 이 때 **MAE** (Mean Absolute Error), **MSE** (Mean Squared Error), **RMSE** (Root Mean Squared Error) 등이 쓰인다

Classifier 평가 방법

1. **평가 방법/절차는 Regressor와 같다** : 역시 Train/Test split 및 Cross-Validation 흔히 활용

- **Classifier 평가 지표 (metric)** : Classifier의 최종 response는 카테고리명 같은 레이블(Label) 이어야 함.

  뒤에 언급되는 Confusion Matrix, 이를 이용한 Classification Accuracy, Error Rate, Sensitivity, ROC 커브, AUC 등 다양한 지표들이 사용됨

Confusion Matrix

- Binary Classification 평가에 활용
- 포맷이 다양함 (예는 Python)

		예측(Predicted)상황		
		예측이 **Negative**	예측이 **Positive**	
실제 (True) 상황	실제가 **Negative** (즉, 0/No)	**True Negative (TN)**	**False Positive (FP)** Type I error	Specificity = $\frac{TN}{\text{실제 Negative}}$
	실제가 **Positive** (즉, 1/Yes)	**False Negative (FN)** Type II error	**True Positive (TP)**	Sensitivy = $\frac{TP}{\text{실제 Positive}}$
		Negative Predictive Value= $\frac{TN}{\text{예측 Negative}}$	Precision = $\frac{TP}{\text{예측 Positive}}$	Accuracy = $\frac{TP + TN}{\text{Total population}}$

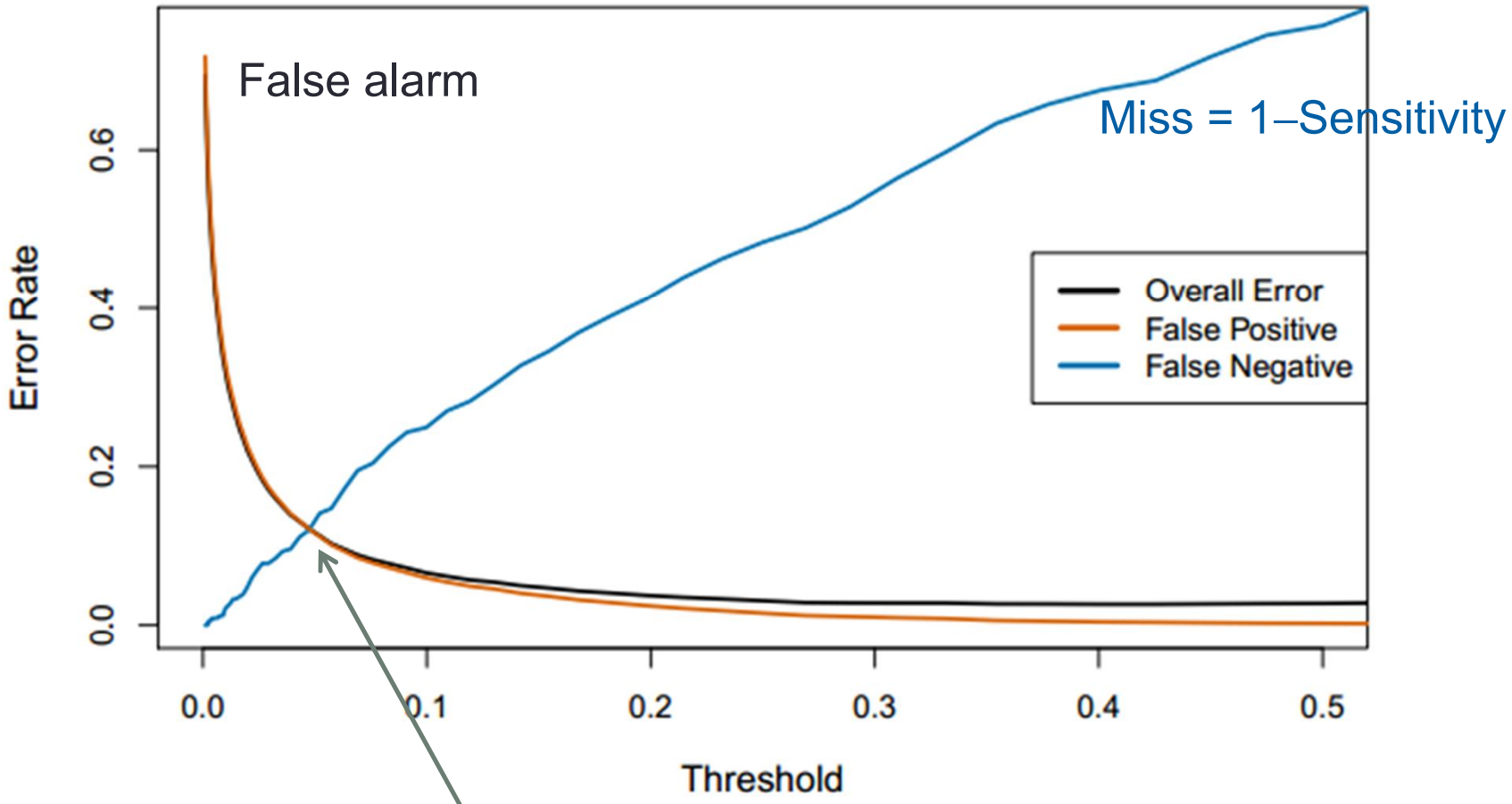
보통 식별하고 싶은 (관심이 있는) 클래스를 Positive, 1로 삼음

- **예를 들어, 암 진단, 비행기 탐지, 채무 불이행자 예측 시 암발견, 비행기 식별, 채무 불이행자로 판단을 Positive 1로**

- **TP** : 실제로 암이고 이를 암이라고 **맞게** 예측한 경우
- **TN** : 실제로 암이 아니고 이를 암이 아니라고 **맞게** 예측한 경우
- **FP** : 실제는 암이 아닌데 이를 암이라고 **틀리게** 예측한 경우
- **FN** : 실제로는 암인데 이를 암이 아니라고 **틀리게** 예측한 경우

❖ **Accuracy (정확도)** : 예측이 맞은 비율 (전체 예측 중에 TP와 TN 합의 비율)
❖ **Sensitivity (민감도**, True Positive Rate, Recall): 실제 암을 얼마나 예측이 이를 맞추었는지 (탐지했는지)
❖ **Precision** : 암이라고 (Positive) 예측을 한 것 중 실제로 **맞은 비율 (Positive 예측이 얼마나 정확한지?)**
❖ **Specificity (특이도, True Negative Rate) :** 실제로 암이 **아닐 때**, 얼마나 이를 **맞게** 예측한 비율
❖ **False Positive Rate (FPR)** : 실제 암이 아닌 것 중 (Negative), 암이라고 (Positive) **틀리게** 예측한 비율 :
FPR = 1 – Specificity = FP/(FP + TN)

Threshold가 작아질수록 miss는 줄고, false alarm은 증가한다. 특히, 어느 시점 (0.04) 부터 false alarm이 급격히 증가함을 볼 수 있다. Threshold를 어디로 잡을 지는 두 타입의 error가 발생 시 상대적 피해가 어떤 쪽이 얼마나 더 큰 지 같은 도메인 지식이 필요



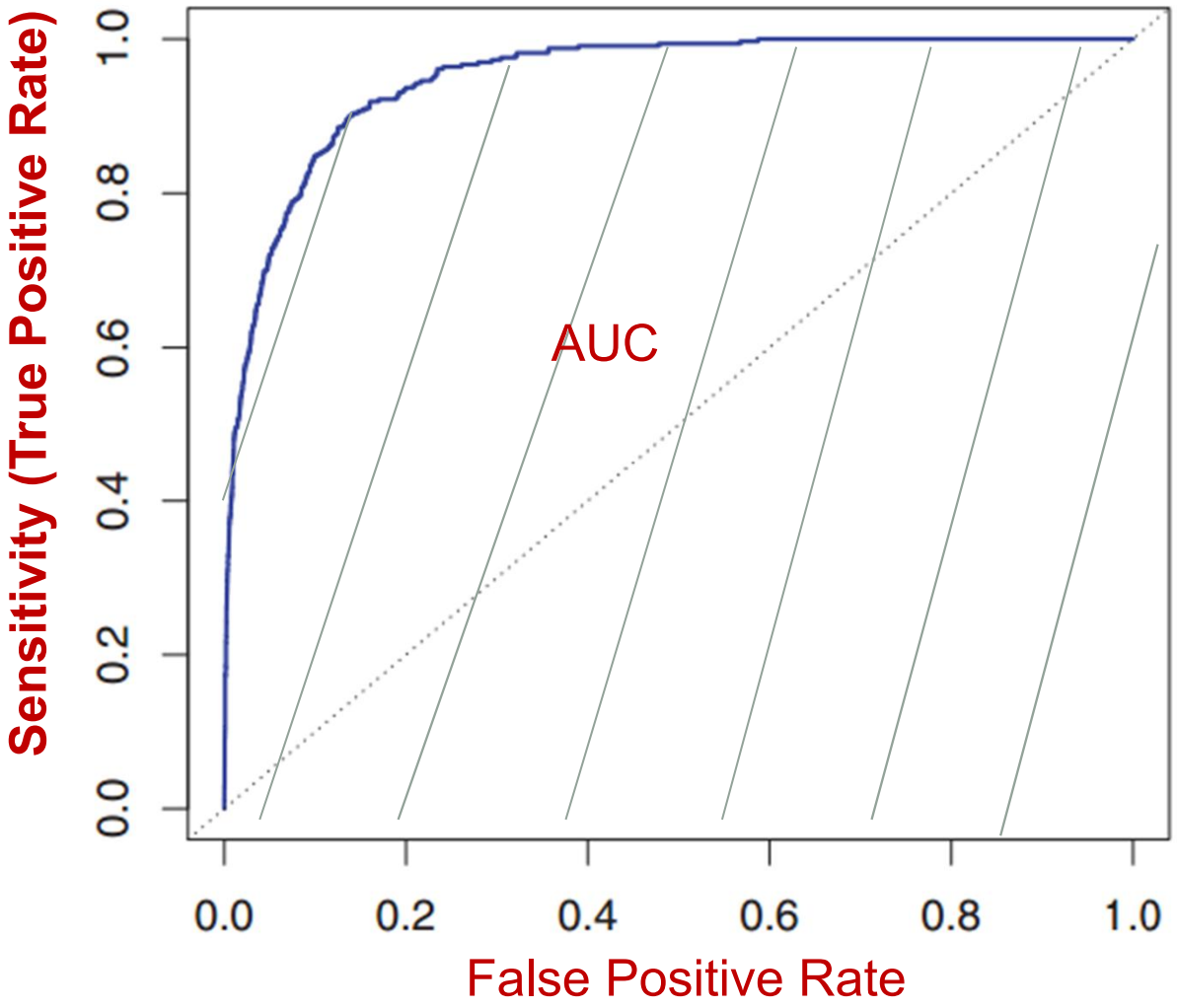
False Negative (Miss) 비율을 낮추는 것이 특히 중요하면 (즉, Sensitivity를 높이는 것이 중요하면) threshold를 0.04 정도까지 낮추는 결정을 할 수도 있다. 이 지점까지는 아직 **False alarm**이 크게 증가하지는 않는다.

ROC (Receiver Operating Characteristic) 커브와 AUC(Area Under an ROC Curve) 역사

- 2차 세계대전 중 레이더 엔지니어들의 고민 중 하나가 레이더 수신장치로 들어온 신호가 구름을, 새를, 적기를, 아군기를, UFO를 잡은 것인지 아니면 그냥 잡음인지 판단하는 것
- 즉, 신호처리/분석을 스마트하게 해서 target 탐지(detection)을 잘 하는 문제. 이 문제를 Classification으로 생각하는 법을 "detection as a classification problem" 이라고.
- 이 사람들은 그 당시까지의 estimation/detection theory, measurement 에 대한 이론/실무적용에 대한 전문가들
- Confusion matrix 같은 개념은 선사시대 동굴에 살던 우리 선조들도 잘 알고 있었고 실생활에 적용했음. 실은 ***개구리***도 이의 대가 : 위협인 새를 놓치지 않고 잘 탐지해야 한 번에 가지 않고, 한편 너무 예민해 나비도 새로 잘 못 판단하면 굶어 죽음
- Target의 식별/탐지가 매우 중요한데, 가령 타깃이 우리 몸의 암, 우리 머리에 폭탄 떨어뜨리려는 적 비행기 라면 이 것 놓치면 우리 건강과 복지에 매우 안 좋기에 가능한 놓치면 안됨.
- 기본적으로, 목표물(Target, Positive Response) 탐지에 대한 예민함을 높이려고 (즉, 타깃을 놓치지 않으려고) 시스템이 Positive라 판단하는 기준을 낮추면 (즉, Positive에 대한 threshold를 낮추면), 대부분의 경우 False Positive (타깃이 실제론 아닌데 타깃이라고 잘 못 예측/판단)도 같이 증가.
- FPR이 높으면 그 시스템에 대한 신뢰 저하. 가령, 암진단 기계가 실제로 암이 아닌데 암이라고 자주 진단하면 그 장치 잘 안 믿을 것임. 이 문제는 공중에 있는 물체 중 적기인 비율, X선 사진 중 암이 있는 사진일 비율과 같이 실제 타깃 비율이 매우 작을 때 (즉, 보통은 negative 비율이 매우 높을 때) 두드러짐.
- 레이다 엔지니어의 고민은 최대한 타깃은 놓치지 않으면서 (Sensitivity가 높음), FPR이 낮은 시스템을 개발하던가, 또는 어떤 시스템에서 Sensitivity와 FPR 간에 적절한 포인트를 찾는 것
- ROC 커브는 어떤 튜닝 패러미터 (여기서는 threshold)를 변화하면서, 그 때 마다 sensitivity와 FPR을 그래프상에 표시한 것으로, 레이더 기술자들이 알고싶은 시스템 성능을 한 눈에 보기 쉽게 표시한 것
- Threshold가 작아지면서 TPR(sensitivity)과 FPR은 monotonically 증가함
- ROC 커브가 빨리 좌상단에 접근함은, threshold를 조금만 낮추어도 sensitivity가 급격히 높아지고 (좋아지고), FPR은 아직 그리 높아지지 않는 그런 threshold 포인트들이 있다는 것. 이 얘기는 해당 시스템이 좋다는 말
- AUC는 ROC 커브 모양을 하나의 숫자로 표현한 것으로, ROC 모양이 좋으면 (좌상단에 붙으면) 커브 아래의 면적이 1에 가까와지므로, 곧 AUC가 1에 근접할 수록 시스템의 성능이 좋다는 말. Binary Classifier의 성능을 간단히 나타낼 때 흔히 사용

ROC (Receiver Operating Characteristic) Curve

-Threshold 를 변화함에 따라 TRP과 FPR 점들을 그린 것. 커브가 좌상단 모서리에 붙을수록 좋다



ROC :
Binary Classifier의 성능을 간략하게 나타낼 수 있는 지표로서, positive 예측의 threshold(임계치)를 변화시킴에 따른 sensitivity와 False Positive Rate의 변화를 그린 곡선

AUC (Area Under the Curve) :
ROC 아래부분 면적
- 일반적으로, 0.5 < AUC < 1.0

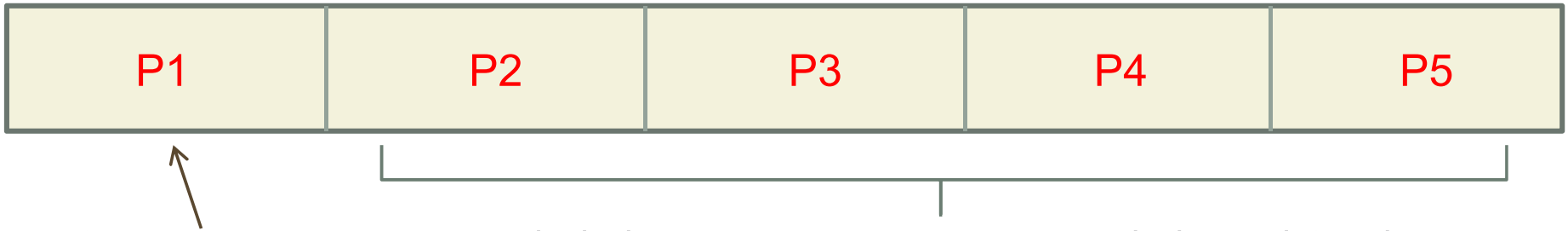
• **False Positive Rate** : 실제 negative 중에서 FP로 잘 못 식별한 비율 (1 – specificity)

Regressor나 Classifier의 평가에서 적절한 metric을 사용함이 **중요**. 적절한 잣대를 사용하지 않았으면 그 모델 엉망. MSE, AUC 같은 것은 기본 **기성품**. 자신에게 맞는 metric들을 고안해야 할 경우 생김

■ K-fold Cross Validation을 이용한 test error rate 추정

- *Widely used approach* for estimating test error.
- Estimates can be used to select best model, and to give an idea of the test error of the final chosen model.
- Idea is to randomly divide the data into K equal-sized parts. We leave out part k, fit the model to the other $K-1$ parts (combined), and then obtain predictions for the left-out kth part.
- This is done in turn for each part $k = 1, 2, \ldots K$, and then the results are combined.

- Validation set을 이용한 test error rate 추정은 데이터를 일단 2 파트로 나누면서 진행되었다. 모델 학습용 training set, 그리고 평가용 validation set.
- K-CV는 데이터를 K개의 같은 크기의 Fold들로 원 데이터를 나누면서 시작된다. 가령, 원 데이터에 샘플/observation이 1000개이고 K=5 이면 5개의 fold에 각각 200개의 observation들이 들어간다. 5개의 fold들을 P1, P2, … P5 라 해보자.



- P1을 validation set으로, 나머지 P2… P4를 training set 삼아 모델을 만들고 P1에 모델을 적용해 보아 test 성능 추정치 (가령 MSE_1)을 구한다.
- 비슷하게, P2를 validation set으로, 나머지 P1, P3, P4, P5를 training set 삼아 모델을 만들고 MSE_2 를 구한다. 이렇게 MSE_5 까지 구한다.
- MSE_1, MSE_2 , … MSE_5 의 평균을 구하면 5-CV의 (평균) MSE

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i$$

Validation vs. LOOCV vs. K-CV

- LOOCV나 K-CV를 해서는 너무 시간이 많이 걸려 문제가 되면 Validation Set (Training/Test Split) 방법을.

시간과 컴퓨팅 자원이 허락하면,

- **LOOCV**는 K-CV에 비해 별 장점이 없다

- **가능한 K-CV를 사용**. K=5~10이 적당. 각 fold가 정확히 똑 같은 개수로 나누어지지 않아도 괜찮다. Classification에서 Class Imbalance가 심하면 stratified K-CV 활용

결론 :

Test 성능을 추정하고 싶으면 C_p,
AIC 이런 것 생각하지 말고
가능한 **Cross-Validation** 하자

* 더 확실한 것은 **Field Test.**