양재 AI School 인공지능 캠프

Lecture 4

Cross Validation: K-fold, Stratified-fold

박상기 선생님

Lecture 4

Cross Validation:
K-fold, Stratified-fold

수업 목표

1 Cross Validation을 이해한다

적은 데이터로 모델을 학습 시킬 수 있는 여러가지
Cross validation의 종류에 대해 알아봅시다

2 모델에서 Dataset을 나눠 쓰는 이유를 이해한다

데이터셋을 Train/Test/Validation 셋 으로 나누는 이유와
그 역할을 알아봅시다

Data Set?

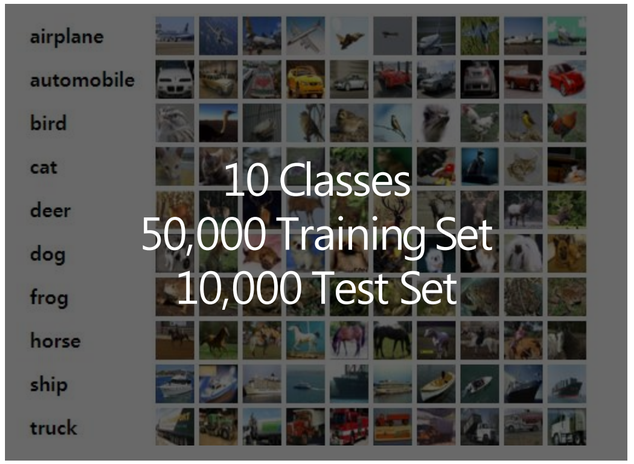
- Machine Learning 모델을 만들기 위해 필요한 데이터의 모임

- 또는 machine learning을 통해 해석이 필요한 데이터의 모임

Data Sets

- **Vision**에서 가장 많이 쓰이는 데이터셋

MINST

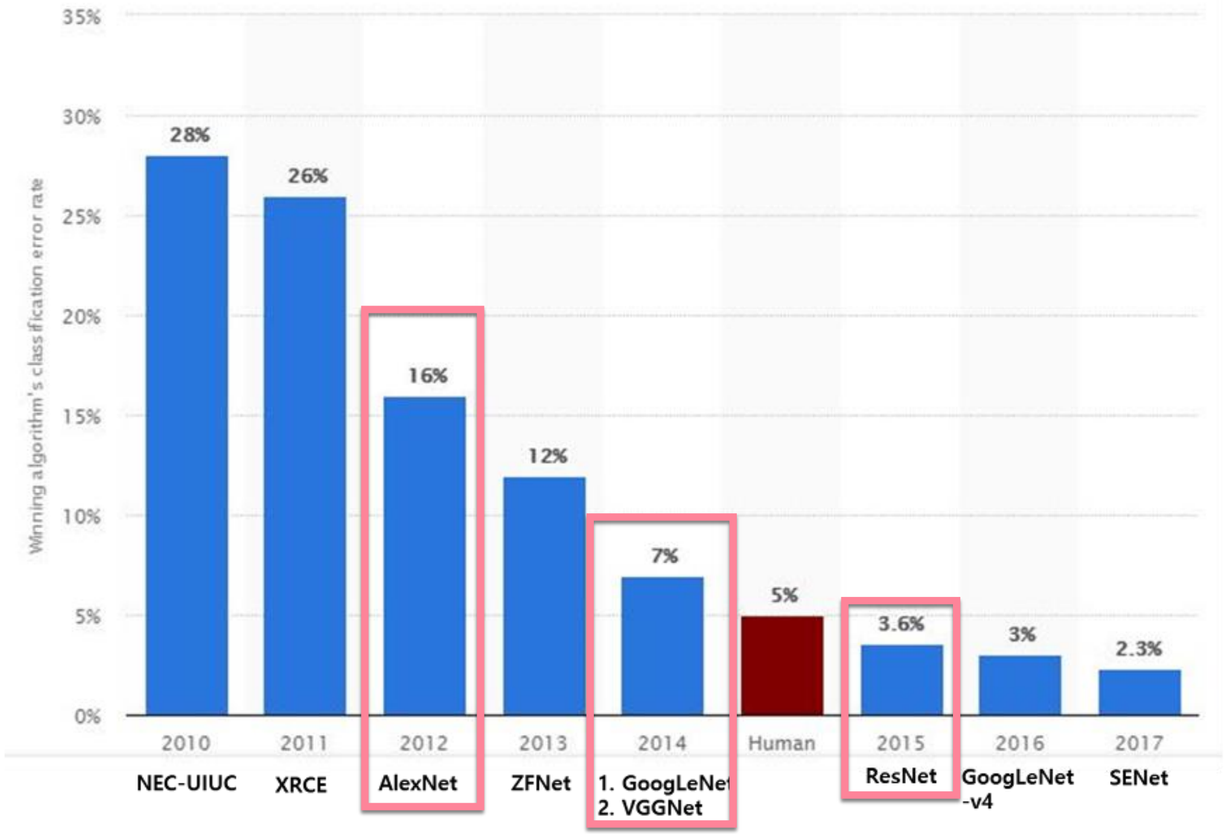


CIFAR-10

ImageNet

Data Sets

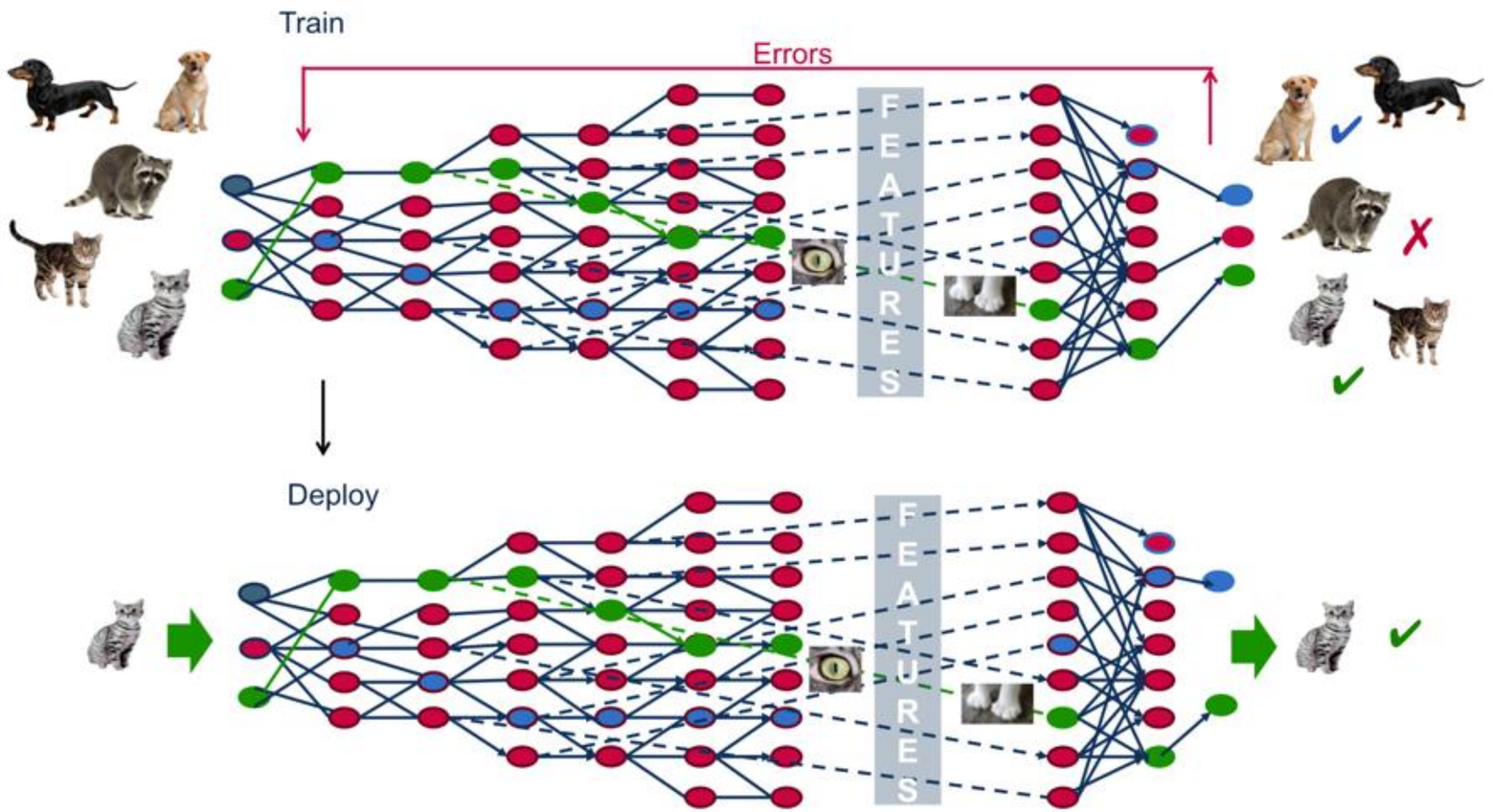
- **Vision**에서 가장 많이 쓰이는 데이터셋



Data Set

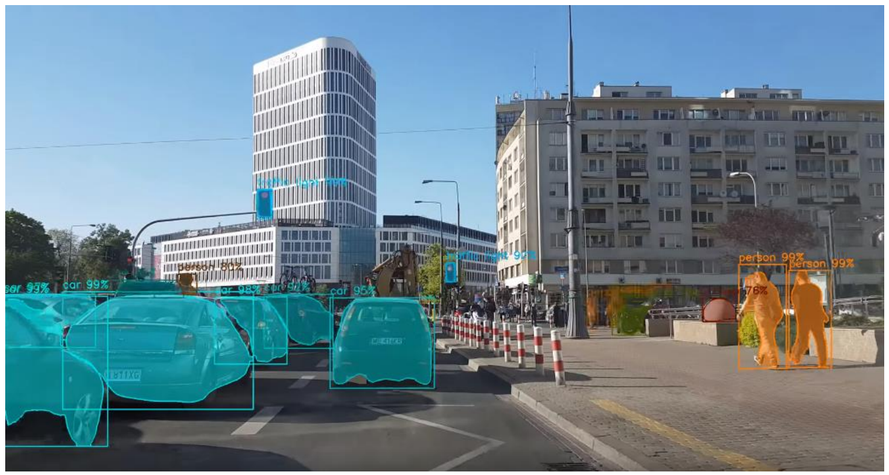
- **Language Model** 에 많이 사용되는 데이터셋
  - Penn Tree Bank (PTB): 929k training, 73k valid words
  - WikiText-2 / WikiText-103: 2M words in total / 267k unique words
  - 1B Words: 829,250,940 tokens, 793,471 unique words

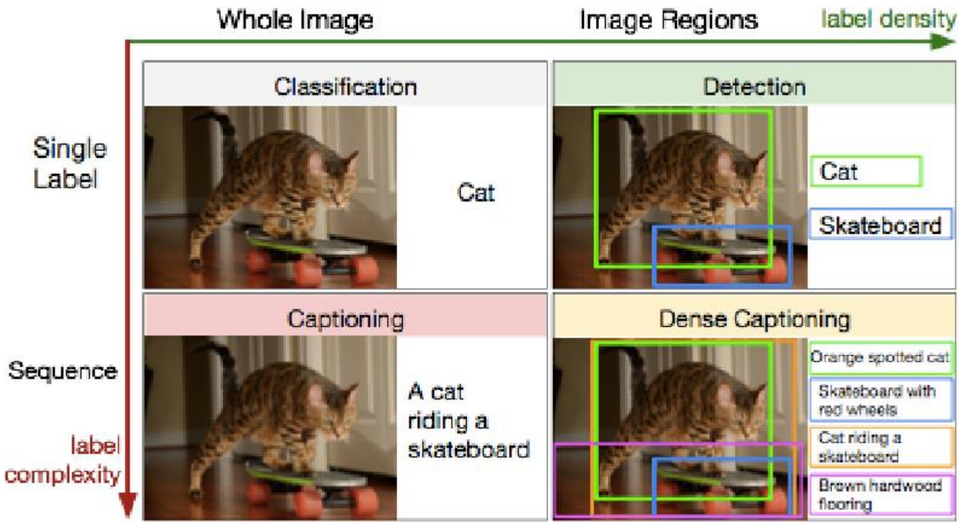
Training? Inference?



Training? Inference?

- 훈련 – 얼마나 예측 잘하는 모델을 만들 것인가
- 추론 – 처음 보는 데이터에 대해 얼마나 정확하게 맞출 수 있는가 (+ 얼마나 빠르게)



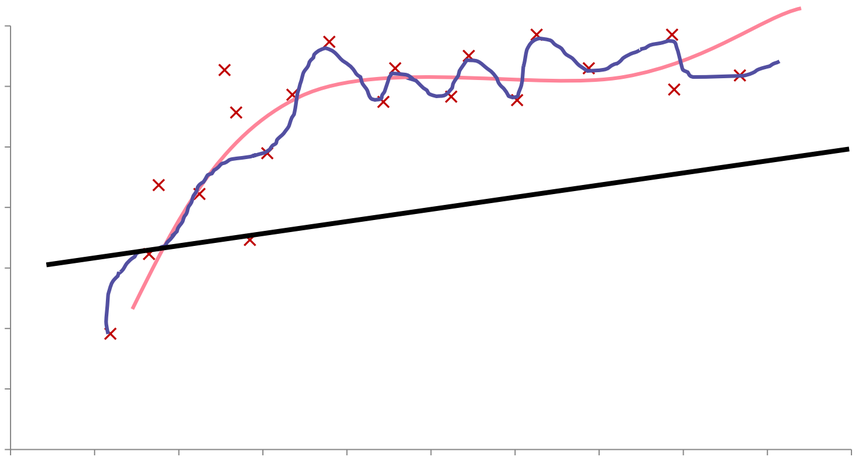


Dataset 나누기

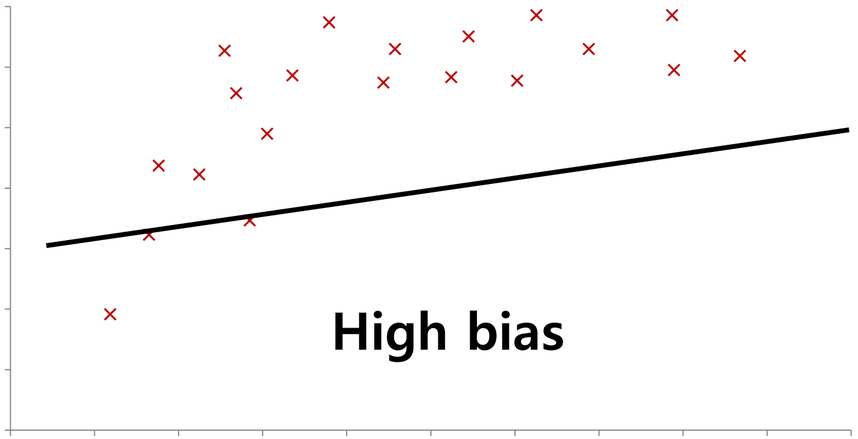
- Dataset을 **Train** 용과 **Test** 용을 따로 나눈다
- **Training** 용 데이터 중 일부분을 **Validation** 용으로 나눈다
- **Validation** set은 모델의 성능을 평가, 및 조정

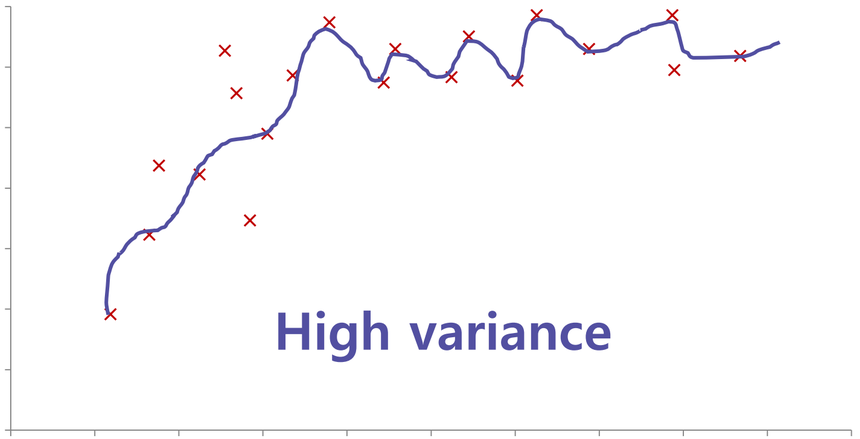
Train a model

- 우리가 해야 할 일은 **데이터 예측**을 잘 할 수 있는 어떤 **수학적 모델**을 만드는 것
- 주어진 훈련 데이터 이외의 데이터도 예측을 잘 해야 좋은 모델



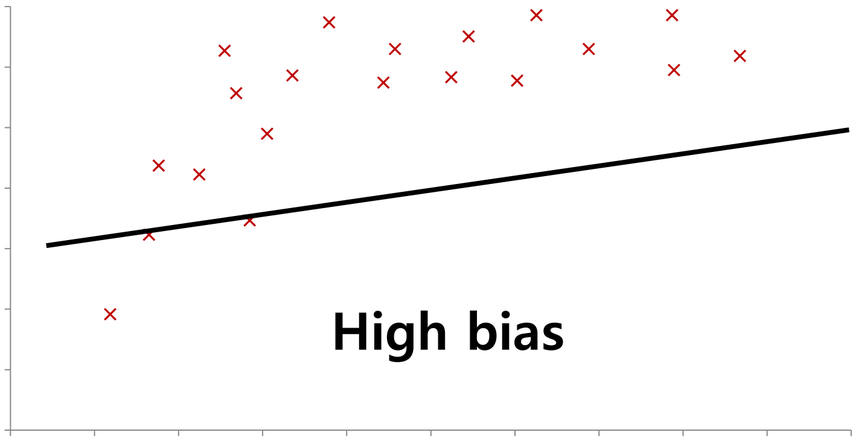
High Bias & High Variance

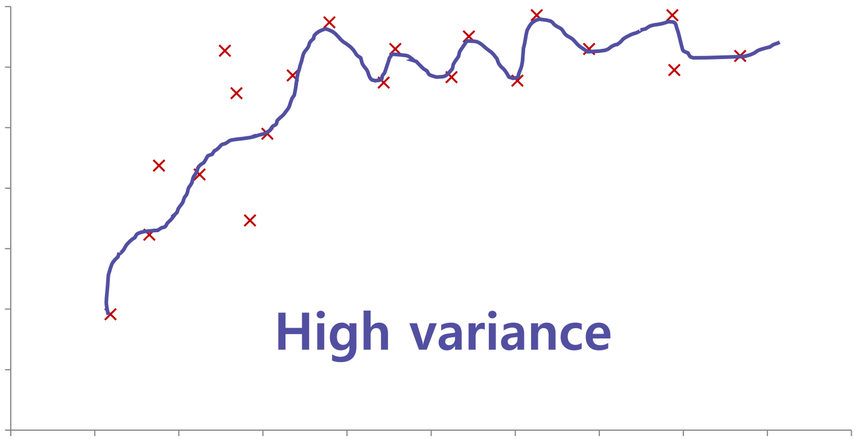




- 모델의 **Hyper-parameter**를 올바르게 정해주어야 한다
  - Learning rate, batch size, L2 regularization 계수 …
- **Bias**: 모델의 **training data**에 대한 예측값과 실제값의 오차
- **Variance**: 모델의 **test data**에 대한 예측값과 실제값의 오차

High Bias & High Variance





- **High bias:** 훈련 데이터를 제대로 표현하지 못함. **Underfit**
- **High variance:** 훈련 데이터를 지나치게 자세하게 표현. 새로운 데이터를 오히려 제대로 예측하지 못함. **Overfit**
- **Validation** 데이터는 train 셋으로 학습한 모델이 “**Sample Bias**” 되지 않았는지 확인

Validation

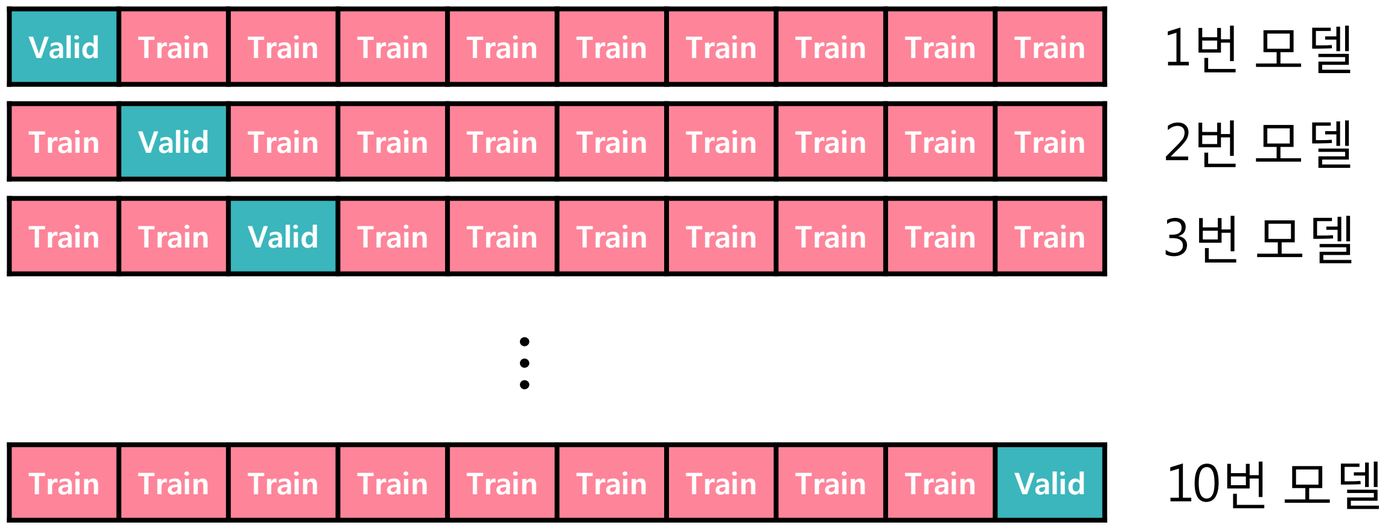
- Training 정확도는 높은데 validation 정확도는 낮다?
  - **Sample bias:** 훈련에 사용하지 않은 데이터에서 낮은 정확도를 보임
- 데이터 셋이 작다면 train/valid/test 세 개로 분류하기 어려움
  - Train 데이터가 적어도 성능 미달
- 데이터가 적을 때는 **Cross validation** 사용
  - Train / test 두 개로 나눔
  - 작은 데이터셋에서도 좋은 정확도
  - **K-Fold / Stratified K-Fold / Leave One Out / Shuffle-split / Bootstrap**

Hold Out

- **Hold Out**: train/test 두 개로 나눔
- +) 간단
- -) Test set을 훈련에 사용하지 못하므로 데이터셋이 작으면 성능의 손실
- -) train/test set을 어떻게 나누냐에 따라 결과가 달라짐

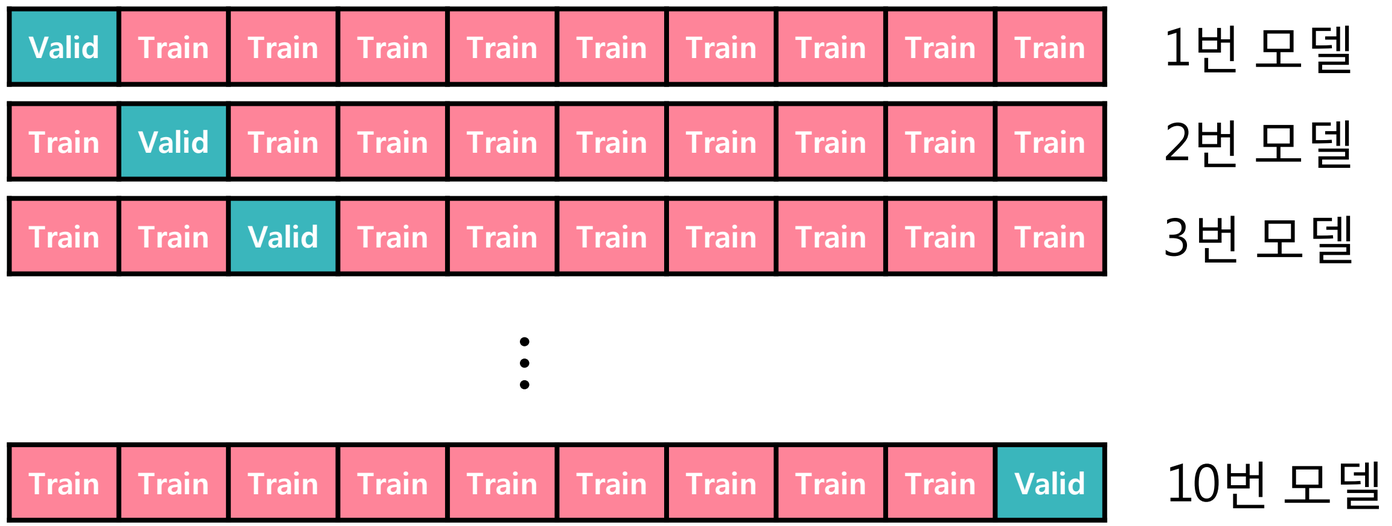
K-Fold

- Train 데이터셋을 K개로 나눈다
- K개 중 한 개를 valid, 나머지를 training 용으로 사용하여 학습
- K개의 모델의 Hyper-parameter의 평균을 최종 결과로 사용한다



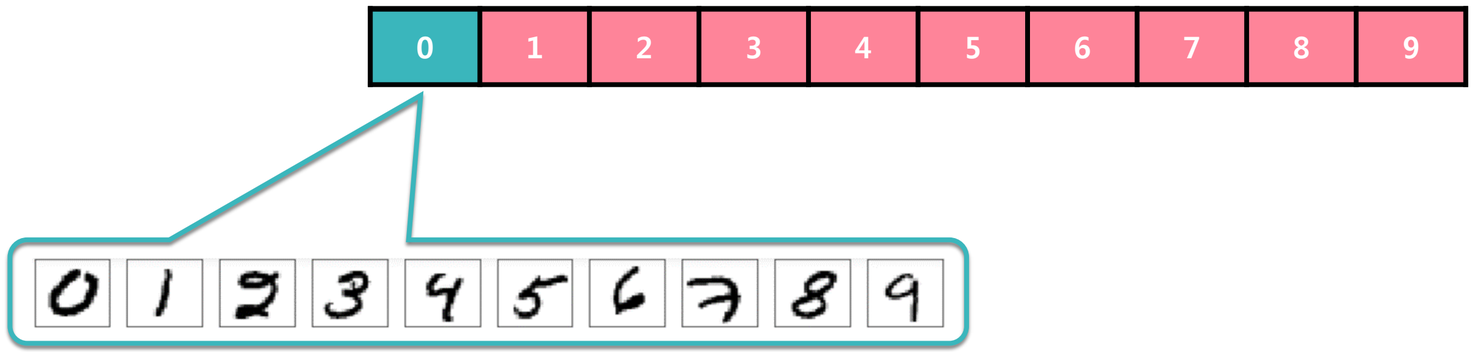
K-Fold

- +) 모든 데이터를 train/valid 용으로 사용 가능
- +) 적은 데이터로도 높은 정확도
- -) 느림



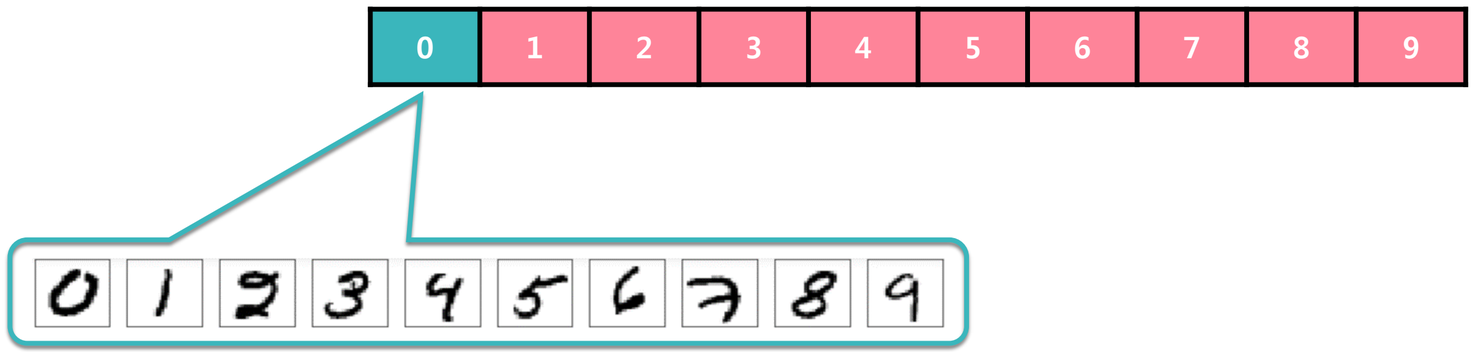
Stratified K-Fold

- 각 fold가 특수하게 나뉜 경우에는 K-Fold 정확도가 좋지 않음
- **Stratified K-Fold**: 각 fold의 데이터 클래스의 분포가 같도록 만들어 준다
  - MNIST: 0~9의 숫자가 동일한 비율로 있음 -> 각 fold의 데이터도 동일한 비율로 들어가도록 설정



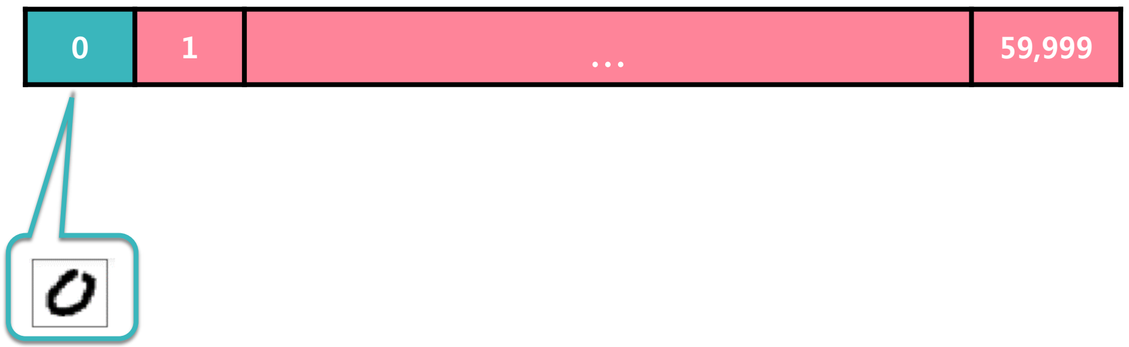
Stratified K-Fold

- +) 각 fold 가 전체 데이터셋을 잘 대표
- +) 모델을 학습시킬 때 편향되지 않게 학습 가능



Leave One Out (LOO)

- K-Fold의 특수한 경우
- K = 데이터의 총 개수, 각 fold에는 데이터 한 개 만
- +) 작은 데이터셋에서 좋은 결과
- -) 데이터셋이 커질수록 매우 느려짐



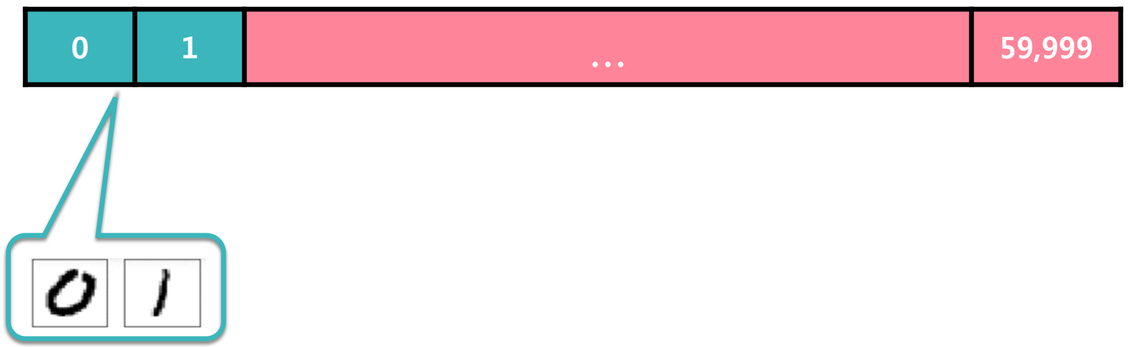
Leave One Out (LOO)

```
import numpy as np
from sklearn.model_selection import LeaveOneOut
X = np. array([[1, 2], [3, 4], [5,6]])
loo = LeaveOneOut()
loo.get_n_splits(X)

for  train_index, test_index in loo.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    print(X_train, X_test)
"""
TRAIN: [1] TEST: [0] [2]
[[3 4]]   [[1 2]] [[5 6]]
TRAIN: [0] TEST: [1] [2]
[[1 2]]   [[3 4]] [[5 6]]
...
"""
```

Leave P Out

- LOO와 비슷하지만, 여기서는 p개 만큼 제외하고 train/test 셋을 만든다
- 총 N개의 데이터에 대해 train_size=N-p, valid_size=p
- 총 N개의 데이터에 대해 $_nC_p$ 만큼 연산. LOO보다 더 느리다.



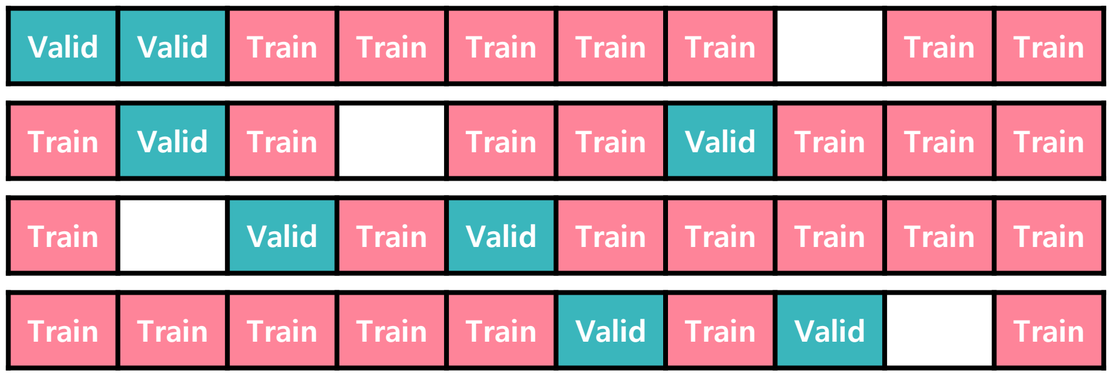
Leave P Out (LPO)

```
import numpy as np
from sklearn.model_selection import LeavePOut
X = np. array([[1, 2], [3, 4], [5,6]])
lpo = LeavePOut(2)
lpo.get_n_splits(X)

for  train_index, test_index in lpo.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    print(X_train, X_test)
"""
TRAIN: [1] TEST: [0] [2]
[[3 4]]   [[1 2]] [[5 6]]
TRAIN: [0] TEST: [1] [2]
[[1 2]]   [[3 4]] [[5 6]]
...
"""
```

Shuffle-Split

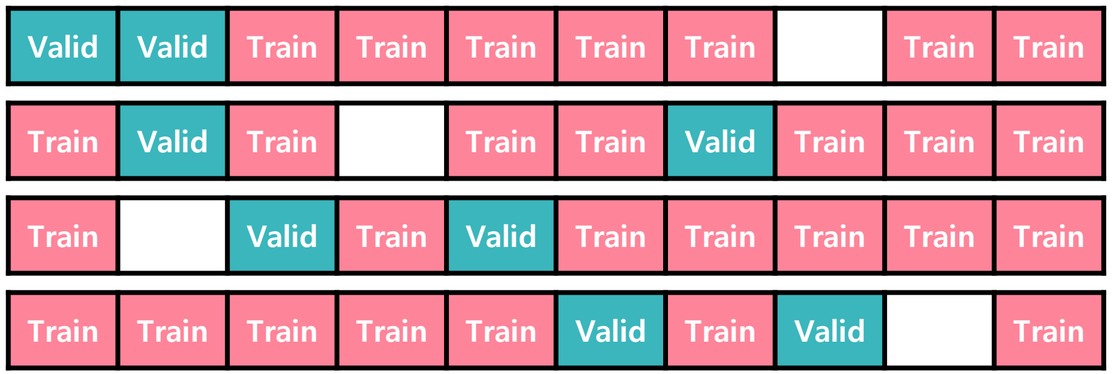
- 반복 횟수를 train fold 수, valid fold 개수와 독립적으로 조절
- train/valid fold의 합을 전체 fold 수와 다르게 설정도 가능



ShuffleSplit with 10 points, train_set=7, valid_set=2, n_split=4

Shuffle-Split

```
import numpy as np
from sklearn.model_selection import ShuffleSplit
X = get_data()
rs = ShuffleSplit(n_split=4, test_size=7, train_size=2)
rs.get_n_splits(X)
```



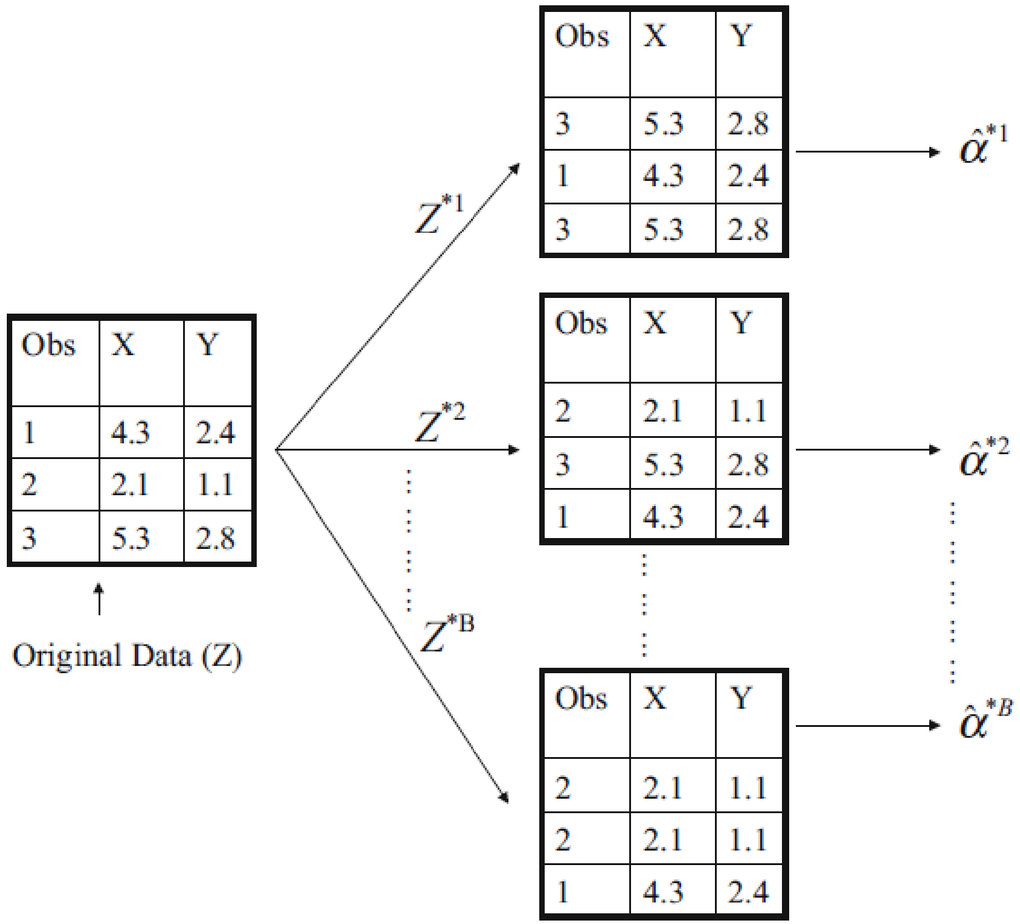
ShuffleSplit with 10 points, train_set=7, valid_set=2, n_split=4

Bootstrap

- 계산하기 어려운, 데이터들의 분산,평균,편차 등을 구하는 통계기법
- 모집단의 분포도를 알 수 없을 때 사용
- 데이터 양이 적을 때, 모델의 통계적 신뢰도를 높이기 위함

- 데이터 셋 내에서 **복원추출** 방법을 이용해 **새로운 샘플**을 만든다
- 충분히 많은 샘플이 생긴다면 모집단의 통계치 추정이 가능

Bootstrap


Obs X Y
1 4.3 2.4
2 2.1 1.1
3 5.3 2.8
Original Data (Z)
Z^{*1}
Z^{*2}
Z^{*B}
Obs X Y
3 5.3 2.8
1 4.3 2.4
3 5.3 2.8
Obs X Y
2 2.1 1.1
3 5.3 2.8
1 4.3 2.4
Obs X Y
2 2.1 1.1
2 2.1 1.1
1 4.3 2.4
$\hat{\alpha}^{*1}$
$\hat{\alpha}^{*2}$
$\hat{\alpha}^{*B}$

Bootstrap

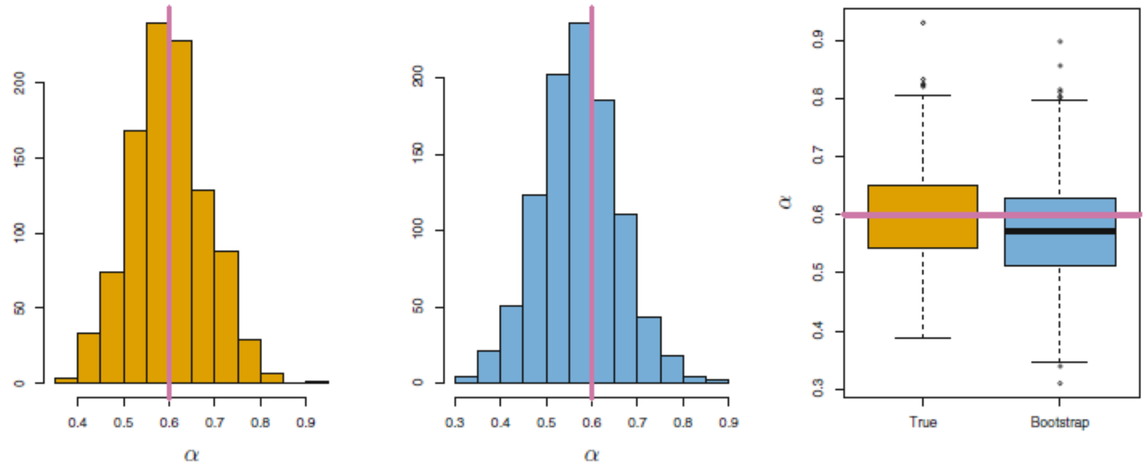


FIGURE 5.10. Left: *A histogram of the estimates of* α *obtained by generating 1,000 simulated data sets from the true population.* Center: *A histogram of the estimates of* α *obtained from 1,000 bootstrap samples from a single data set.* Right: *The estimates of* α *displayed in the left and center panels are shown as boxplots. In each panel, the pink line indicates the true value of* α.

- 노란색: 한 세트당 100개의 데이터가 있는 1000 세트로 얻은 결과
- 파란색: 한 개의 세트에서 bootstrap을 이용해 1000개의 샘플로 추정한 결과

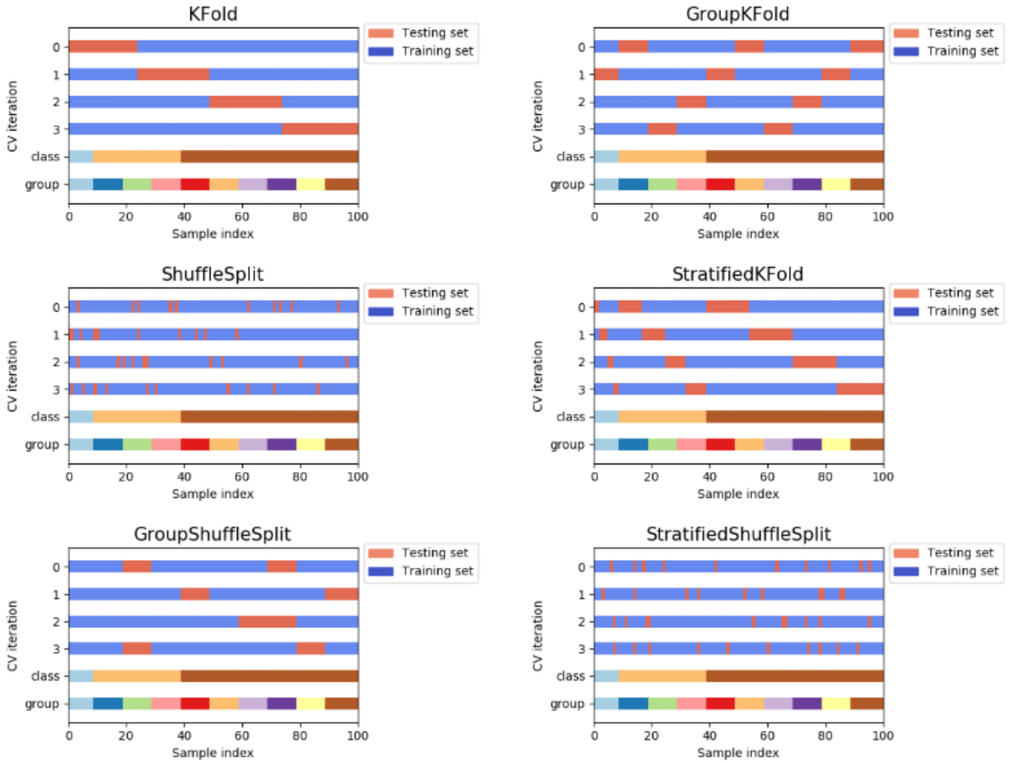
Bootstrap

```
import numpy as np
from sklearn.model_selection import cross_validation
Bs = cross_validation.Bootstrap(n=9, n_bootstrap=3, n_train=5, n_test=4)
“””
TRAIN: [1 8 7 7 8] TEST: [0 3 0 5]
TRAIN: [5 4 2 4 2] TEST: [6 7 1 0]
TRAIN: [4 7 0 1 1] TEST: [5 3 6 5]
“””
```

- #n: 전체 데이터 개수
- #n_bootstrap: 반복횟수. 이 수 만큼 샘플이 생긴다
- #n_train / n_test: train/test 한 세트마다 들어갈 데이터의 갯수

Visualizing

- Visualizing cross-validation behavior in scikit-learn



Visualizing

- 코드에 다음 내용 추가

```
from elice_utils import ElicUtils
elice_utils = EliceUtils()
…
i = 0
for cv in cvs:
        …
        file_name = str(i)
        fig.savefig(file_name + “.png”)
        elice_utils.send_image(file_name + “.png”)
        i += 1
```