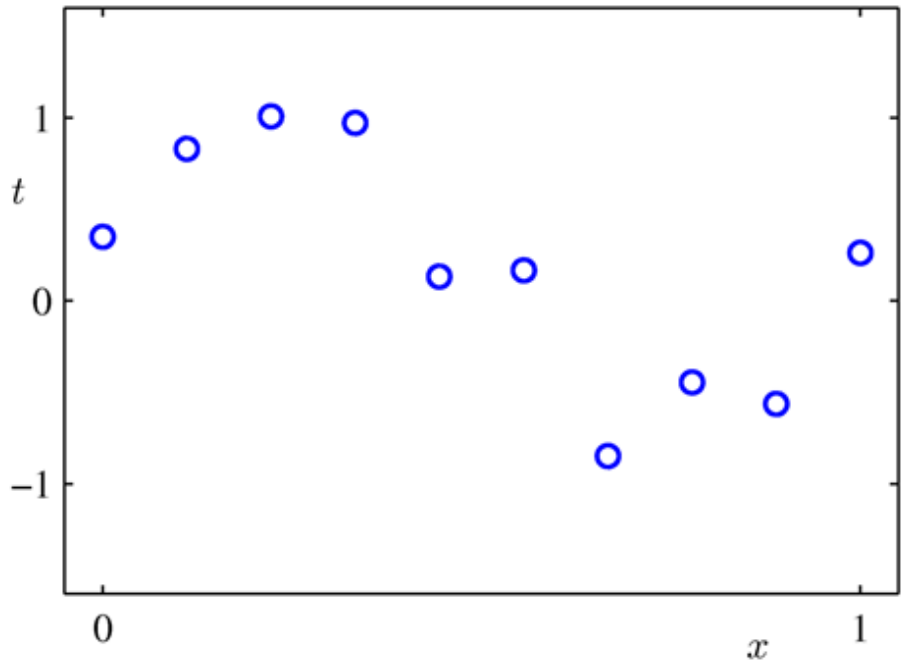
Overfitting,Generalization,Cross-Validation

Overfitting vs Generalization

of Which order polynomial will be best for the data?

- The model which has the least error as much as possible

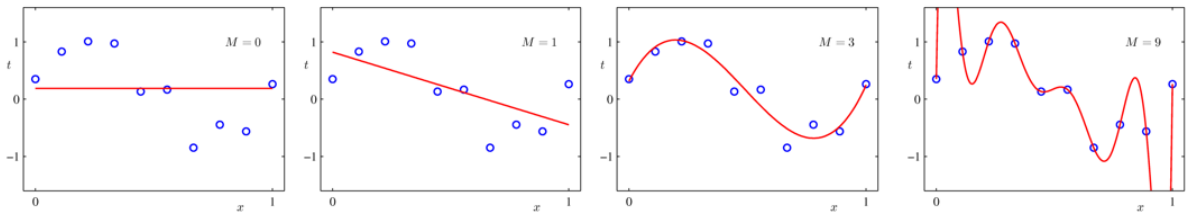


과연 에러가 적은게 가장 좋은것일까?

Learning the given data as exactly as possible

vs

Predict the unknown data as exactly as possible based on the given data



Then, Which one looks best?

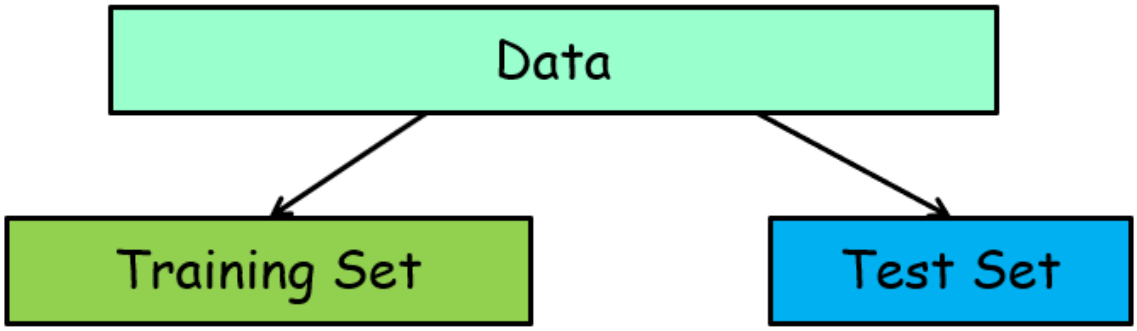
- As M increases,
  - the complexity of model increases
- As the complexity of model increases,
  - the model can more **exactly learn the given data**
  - However, the **prediction accuracy does not necessarily increase**

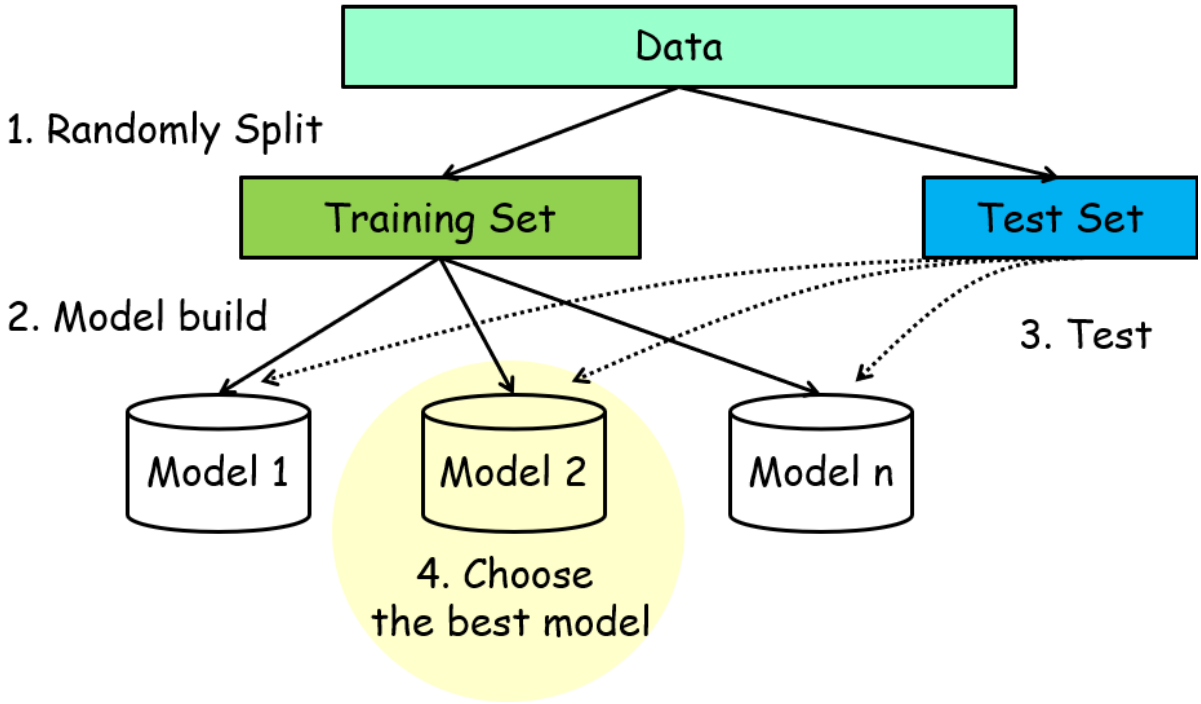
머신러닝,딥러닝의 목적은 주어진 데이터를 잘맞는것도 중요하지만 결국 목적은 unknown data를 잘 예측하는것이다. 따라서 너무 주어진 데이터에만 잘 맞추면(즉, overfitting) unknown data를 잘 맞추지 못한다.

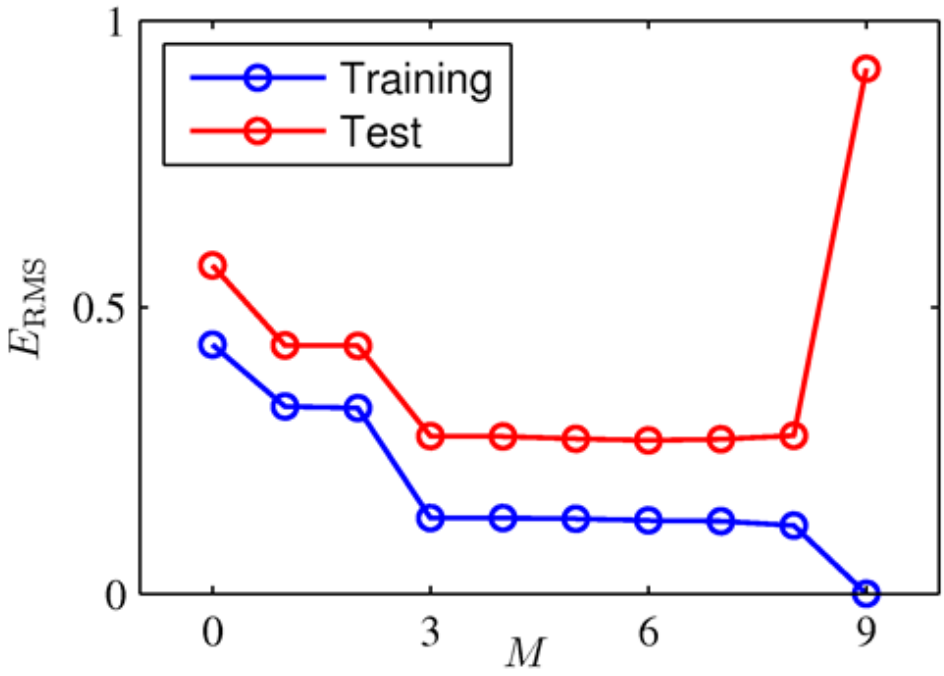
Training Set and Test Set

How to choose a good model

- Divide the given data into TRAINING set and TEST set
  - Training set and Test set should NOT overlap each other!!
  - Both need to be **independent as much as possible**
- With Training set, build various models
- With Test set, test each model
- Choose the model which shows the best performance with Test set


Data
Training Set
Test Set


Data
1. Randomly Split
Training Set
Test Set
2. Model build
3. Test
Model 1
Model 2
Model n
4. Choose
the best model



Advantage

- Simple and easy

Disadvantage

- Test set is not used for modeling building. Waste of data
- Data is randomly split
  - Ebaluation can be significantly different depending on data split

—>Any good idea?>>cross validation

Cross Validation

교차 검증이 필요한 이유

여기에 우리가 사용할 데이터가 있고, 그 데이터는 label이 있는 train, test set으로 구성되어 있다.

이 경우에 만약 'train set을 다시 train set + validation set으로 분리하지 않는다'라고 가정하면, 우리는 모델 검증을 위해서 test set을 사용하여야 할 것이다. 사실상 test set이 아닌 valdiation set인 셈인데, 여기에 한 가지 약점이 존재한다. 고정된 test set을 가지고 모델의 성능을 확인하고 파라미터를 수정하고, 이 과정을 반복하면 결국 내가 만든 모델은 test set에만 잘 동작하는 모델이 된다. 이 경우에는 **test set에 과적합(overfitting)**되어 다른 실제 데이터를 가지고 예측을 수행하면 엉망인 결과가 나와버리게 된다.

이렇듯 고정된 train set과 test set으로 평가를 하고, 반복적으로 모델을 튜닝하다보면 test set에만 과적합되어버리는 결과가 생긴다. 이를 해결하고자 하는 것이 바로 **교차 검증(cross validation)이다**

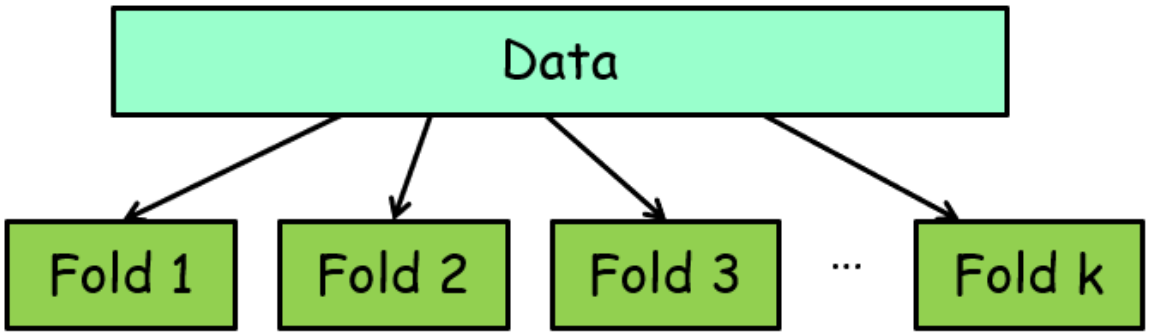
자, 잘 생각해 보면 'test set에 과적합 되는 문제'는 test set이 데이터 중 일부분으로 고정되어 있고, 이 일부분의 데이터 셋에 대하여 성능이 잘 나오도록 파라미터를 반복적으로 튜닝하기 때문에 발생한다. 교차 검증은 데이터의 모든 부분을 사용하여 모델을 검증하고, test set을 하나로 고정하지 않는다.

In order to reduce statistical variance

- Usually, K-fold cross validation is widely used

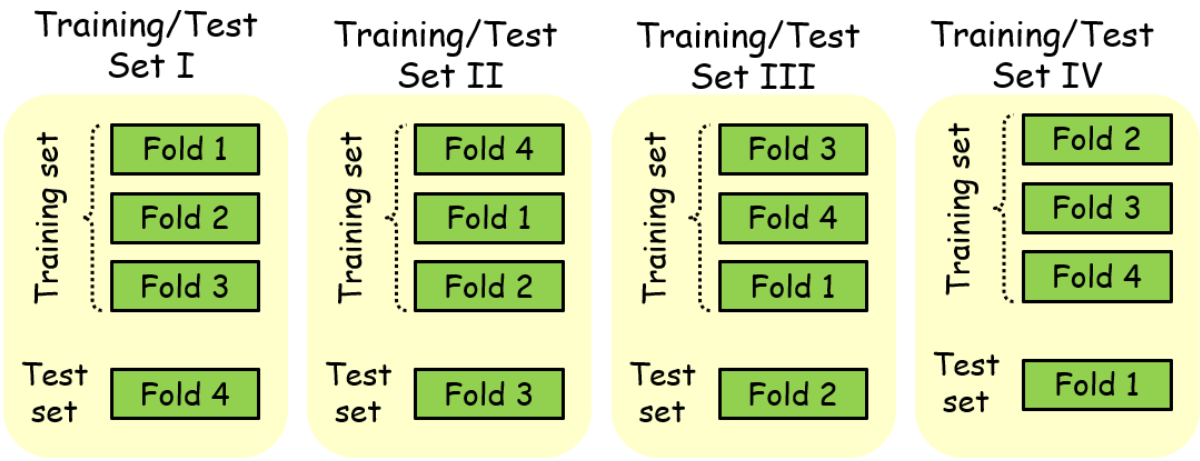
K-fold Cross validation

- Split given data into K folds
- Folds should not overlap with each other

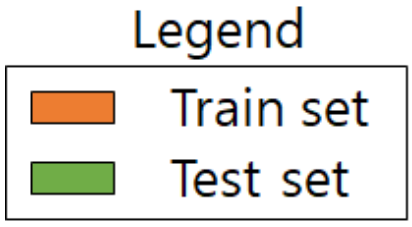


우선 제일 많이쓰는 k-fold cross validation을

말해보자



Choose a model by the average performance of 4 sets



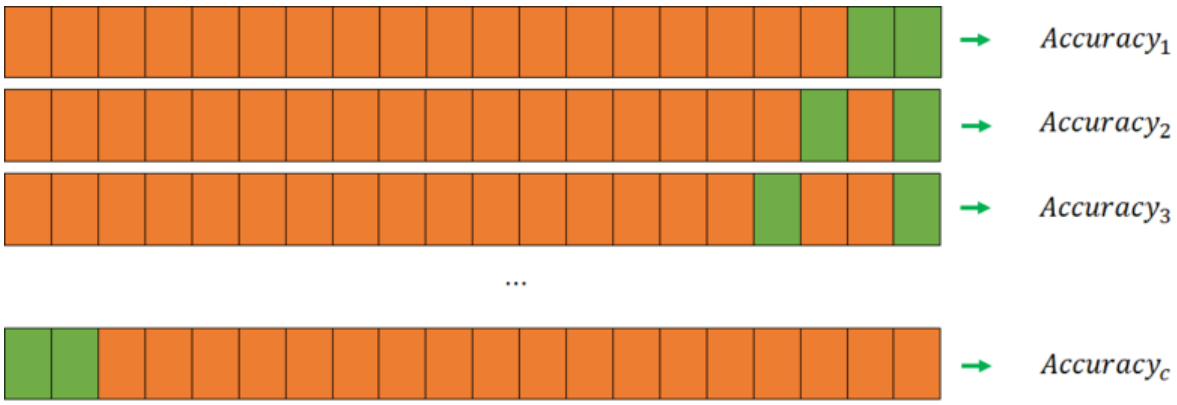
Holdout method(홀드아웃 방법)

Holdout method 또는 Holdout cross validation이라고 불린다. 이 방법이 교차 검증인지 아닌지에 대해서는 애매한 부분이 존재하지만, 위키피디아 [3]에서는 교차 검증 범주에 속해져있기 때문에, 소개하도록 한다.

이 방법은 주어진 train set을 다시 임의의 비율로 train set과 test set으로 분할하여 사용한다. 이 때 train : test = 9 : 1 또는 7 : 3 비율이 가장 자주 쓰이며, Iteration(훈련 및 검증)을 한 번만 하기 때문에 계산 시간에 대한 부담이 적은 것이 장점이다. 반면에, test set에 관한 검증 결과 확인 후 모델 파라미터 튜닝을 하는 작업을 반복하게 되면 모델이 test set에 대해 overfit 될 가능성이 높다는 것이 단점이다.

흔히 아는 그냥 train,test set분할하고 하는방법

Leave -p-out cross validation



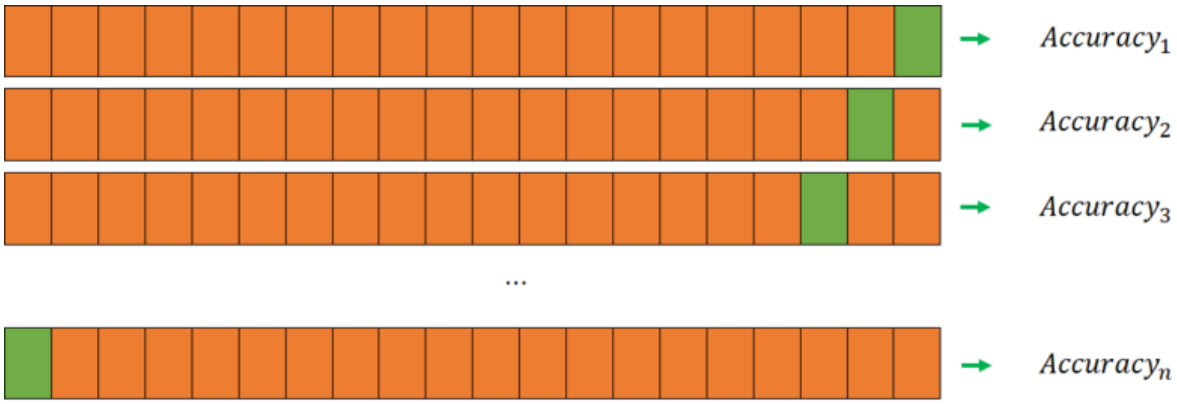
$$Accuracy = Average(Accuracy_1, \cdots, Accuracy_c),$$
where $c = \#\ of\ combinations\ of\ data\ sample$

전체 데이터(서로 다른 데이터 샘플들) 중에서 p개의 샘플을 선택하여 그것을 모델 검증에 사용하는 방법이다. 따라서, test set을 구성할 수 있는 경우의 수(=훈련 및 검증에 소요되는 Iteration 횟수)는 아래와 같다.

$$_nC_p$$

k-겹 교차 검증 방법과 마찬가지로, 각 데이터 폴드 세트에 대해서 나온 검증 결과들을 평균내어 최종적인 검증 결과를 도출하는 것이 일반적이다. 이 교차 검증 방법은 구성할 수 있는 데이터 폴드 세트의 경우의 수가 매우 크기 때문에, 계산 시간에 대한 부담이 매우 큰 방법이다.

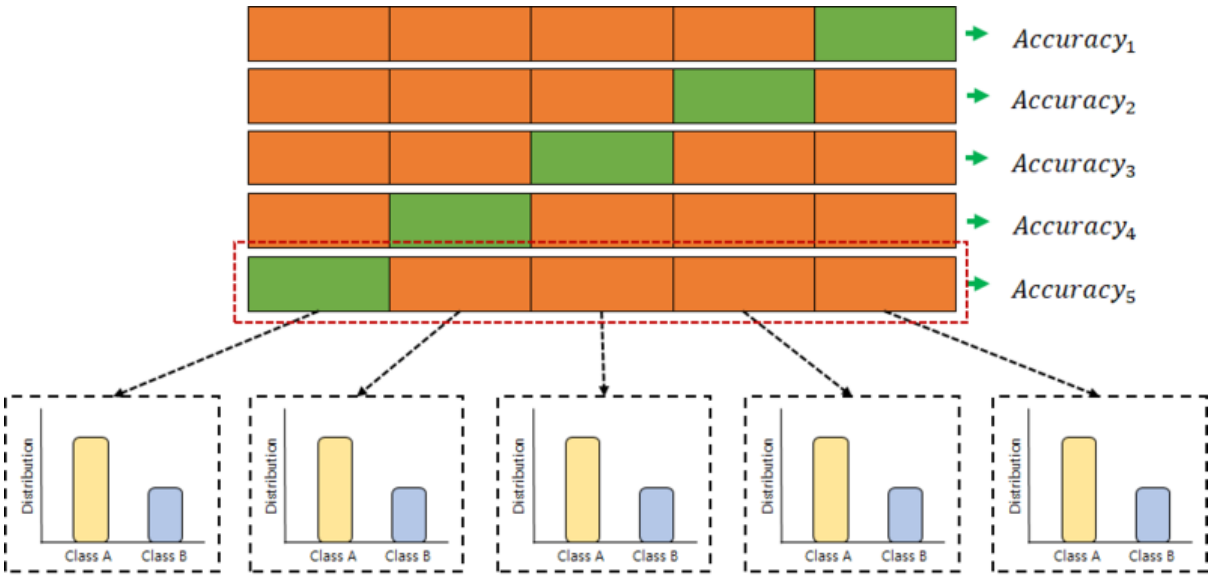
Leave-one-out cross validation



$$Accuracy = Average(Accuracy_1, \cdots, Accuracy_n), \quad \text{where } n = \#\ of\ data\ sample$$

Leave-one-out cross validation은 줄여서 LOOCV라고도 불리우며, 앞서 언급했던 leave-p-out cross validation에서 p=1일 때의 경우를 말한다. leave-p-out cross validation 보다 계산 시간에 대한 부담은 줄어들고, 더 좋은 결과를 얻을 수 있기 때문에 더욱 선호된다. 검증에 사용되는 test set의 갯수가 적은 만큼 모델 훈련에 사용되는 데이터의 갯수는 늘어난다. 모델 검증에 희생되는 데이터의 갯수가 단 하나이기 때문에, 나머지 모든 데이터를 모델 훈련에 사용할 수 있다는 것이 장점이다. K-fold cross validation에서 data size=n이라면 k=n으로 하는 것과 같은 맥락이다.

Stratified K-fold cross validation



$$Accuracy = Average(Accuracy_1, \cdots, Accuracy_k)$$

주로 **Classification** 문제에서 사용되며**, label의 분포가 각 클래스별로 불균형(편향)**을 이룰 때 유용하게 사용된다. label의 분포가 불균형한 상황에서 sample의 index 순으로 데이터 폴드 세트를 구성하는 것은 데이터를 검증하는데 치명적인 오류를 야기할 수 있다. Stratified k-fold cross validation은 이러한 데이터 **label의 분포까지 고려해 주어서**(그렇기 때문에 데이터 폴드 세트를 구성해주는 함수에서 데이터의 label 값이 요구됨), 각 훈련 또는 검증 폴드의 분포가 전체 데이터셋이 가지고 있는 분포에 근사하게 된다.

Pros&cons about Cross Validation

장점 :

1. 모든 데이터 셋을 평가에 활용할 수 있다.

- 평가에 사용되는 데이터 편중을 막을 수 있다.

(특정 평가 데이터 셋에 overfit 되는 것을 방지할 수 있다.)

- 평가 결과에 따라 좀 더 일반화된 모델을 만들 수 있다.

2. 모든 데이터 셋을 훈련에 활용할 수 있다.

- 정확도를 향상시킬 수 있다.

- 데이터 부족으로 인한 underfitting을 방지할 수 있다.

단점 : Iteration 횟수가 많기 때문에 모델 훈련/평가 시간이 오래 걸린다.

Reference

https://m.blog.naver.com/ckdgus1433/221599517834