스마트 공장 제품 품질 상태 분류 솔루션

우리 Gram 써요

백지수, 가상은, 서유진, 오정민, 이경원

스마트 공장이란?

- 제품의 기획부터 판매까지 모든 생산과정을 ICT(정보통신) 기술로 통합하여 최소 비용과 시간으로 고객 맞춤형 제품을 생산하는 사람 중심의 첨단 지능형 공장을 말한다.

그림 출처: https://sjtp.or.kr/bbs/center/center.php?bo_content=center02

스마트 공장 발전의 필요 조건

스마트 공장을 구성하고 발전시킴에 있어서 꼭 필요한 5가지 조건은 다음과 같다.

4M+1E의 디지털화

4M+1E의 요소(Man, Machin-ery, Material, Method, Environ-ment)들을 실시간으로
디지털 값을 인지하고, 측정 가능한 정보를 제공해야 하면 통신을 통한 대화가 가능해야 한다.

지능화

알고리즘 또는 인공지능 등의 솔루션을 이용하여, 최적해 또는 예측 가능한 해를 제공해야 한다.

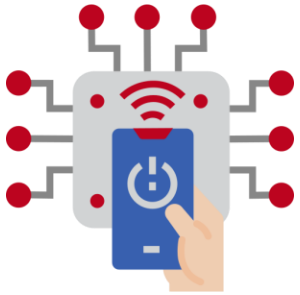
스마트 공장 발전의 필요 조건

통합

사회망과 가치사슬을 통해 단대단(End-to-end)의 정보 교류가 이뤄지도록 하는 수평적 통합과 최하위 수준인 기계장치부터 기업 비즈니스 수준까지 수직적 통합을 지향한다.

엔지니어링 지식의 창출

지속해서 정보를 확보하고 저장한 후, 이를 바탕으로 자동화를 위한 제조 지식을 점진적으로 창출할 수 있어야 한다.



스마트 시스템과의 연결

향후에 발전할 스마트 제품들과 통신 표준에 의거해 연결이 가능해야 한다.

코드 설명

```python
import pandas as pd
import random
import os
import numpy as np

from sklearn.preprocessing import LabelEncoder

def seed_everything(seed):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)
seed_everything(37) # Seed 고정

# 데이터 불러오기
train_df = pd.read_csv('./train.csv')
test_df = pd.read_csv('./test.csv')

train_x = train_df.drop(columns=['PRODUCT_ID', 'TIMESTAMP', 'Y_Class', 'Y_Quality'])
train_y = train_df['Y_Class']

test_x = test_df.drop(columns=['PRODUCT_ID', 'TIMESTAMP'])

# 데이터 전처리
train_x = train_x.fillna(0)
test_x = test_x.fillna(0)
```

- seed_everything() 함수는 난수를 생성할 때 발생하는 시드 값을 고정해준다.
- 시드 값은 37로 설정하였다.
- 같은 시드 값을 사용해서 난수를 생성하면 같은 결과를 얻을 수 있다.

- pd.read_csv() 함수를 이용하여 csv 파일 데이터를 읽어온다.
- train.csv 파일에서 읽어온 train_x, train_y 데이터에서 필요한 열만 남기고 삭제한다.
- test.csv 파일에서 읽어온 test_x 데이터에서도 필요한 열만 남기고 삭제한다.

- train_x, test_x데이터에서 누락된 값이 있을 경우 0으로 채운다.

```
27  # qualitative to quantitative
28  qual_col = ['LINE', 'PRODUCT_CODE']
29
30  for i in qual_col:
31      le = LabelEncoder()
32      le = le.fit(train_x[i])
33      train_x[i] = le.transform(train_x[i])
34
35      for label in np.unique(test_x[i]):
36          if label not in le.classes_:
37              le.classes_ = np.append(le.classes_, label)
38      test_x[i] = le.transform(test_x[i])
39  print('Done.')
```

- 'LINE'과 'PRODUCT_CODE'를 변수 qual_col에 저장한 후 숫자형으로 변환해주는 코드이다.
- for문을 사용해 qual_col리스트에 있는 변수들에 LabelEncoder을 적용한다.
  LabelEncoder는 문자열 형태의 데이터를 숫자 형태로 변환하는 작업을 수행하는 scikit-learn의 클래스이다.
- train_x 데이터에 fit()을 사용하여 LabelEncoder 객체에 대한 학습을 한다.
  그 다음, train_x 데이터에 대해 transform()을 사용해 문자열을 숫자형으로 변환한다.
  test_x 데이터에 LabelEncoder를 재사용하기 위해 classes_를 적용한다.
  그 후, test_x 데이터에 대해 transform()을 사용해 문자열을 숫자형으로 변환한다.

```python
# Classification Model Import
from sklearn.ensemble import RandomForestClassifier #예시코드
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import VotingClassifier

# Classification Model Fit
RF = RandomForestClassifier(random_state=37)
GBC = GradientBoostingClassifier(random_state=37)
VLD = VotingClassifier(estimators=[('RF', RF), ('GBC', GBC)], voting='soft')
VLD.fit(train_x, train_y)
```

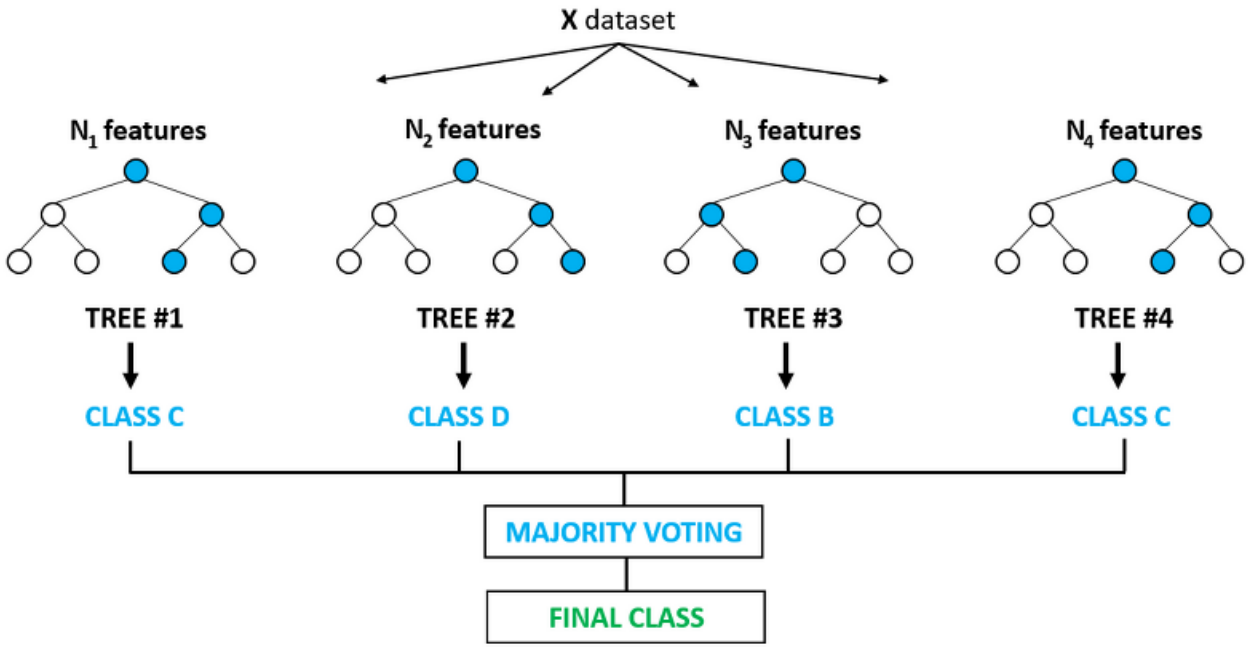
- RandomForestClassifier와 GradientBoosting Classifier를 이용하여 VotingClassifier 모델을 학습한다.
- VotingClassifier는 'soft'방식을 사용하여 RandomForestClassifier와GradientBoosting Classifier의 예측 결과에 대한 확률값을 이용하여 다수결 투표를 통해 최종 예측 결과를 출력한다.

```python
# Test 데이터에 대한 분류 전, Train 데이터에 대해 학습이 잘 되었는지 확인
print(VLD.score(train_x, train_y))
print('Done.')

# Test 데이터 분류
preds= VLD.predict(test_x)
print('Done.')

# 제출
submit2 = pd.read_csv('./sample_submission.csv')
submit2['Y_Class'] = preds
submit2.to_csv('./baseline_submission4.csv', index=False)
```

- score()를 사용해서 모델(VLD)의 예측 정확도를 계산한다.
- predict()를 사용해서 test데이터를 예측하고, 예측값을 저장한다.
- to_csv()로 데이터를 csv파일로 저장한다.

Random Forest

- 분류, 회귀 분석 등에 사용되는 앙상블 학습 방법의 일종
- 다수의 결정 트리로부터 분류 또는 평균 예측지를 출력



다수의 결정 트리

Random Forest Classifier -> 변수까지 특정 개수로 무작위 추출을 하는 방법으로 학습을 진행한다

```
① from sklearn.ensemble import RandomForestClassifier #예시코드
  from sklearn.ensemble import GradientBoostingClassifier
  from sklearn.ensemble import VotingClassifier

  # Classification Model Fit
② RF = RandomForestClassifier(random_state=37)
  GBC = GradientBoostingClassifier(random_state=37)
③ VLD = VotingClassifier(estimators=[('RF',RF), ('GBC',GBC)], voting='soft')
  VLD.fit(train_x, train_y)
```

① from sklearn.ensemble import RandomForestClassifier

- sklearn.ensemble 패키지(앙상블 알고리즘 패키지)에서 RandomForestClassifier 함수를 가져온다.

② RF=RandomForestClassifier(random_state=37)

- RF라는 이름으로 RandomForestClassifier 모델을 선언한다.
- random_state는 호출할 때마다 데이터 세트를 생성하기 위해 주어지는 난수 값이다.
- n_estimators 파라미터의 기본 값은 100이다.

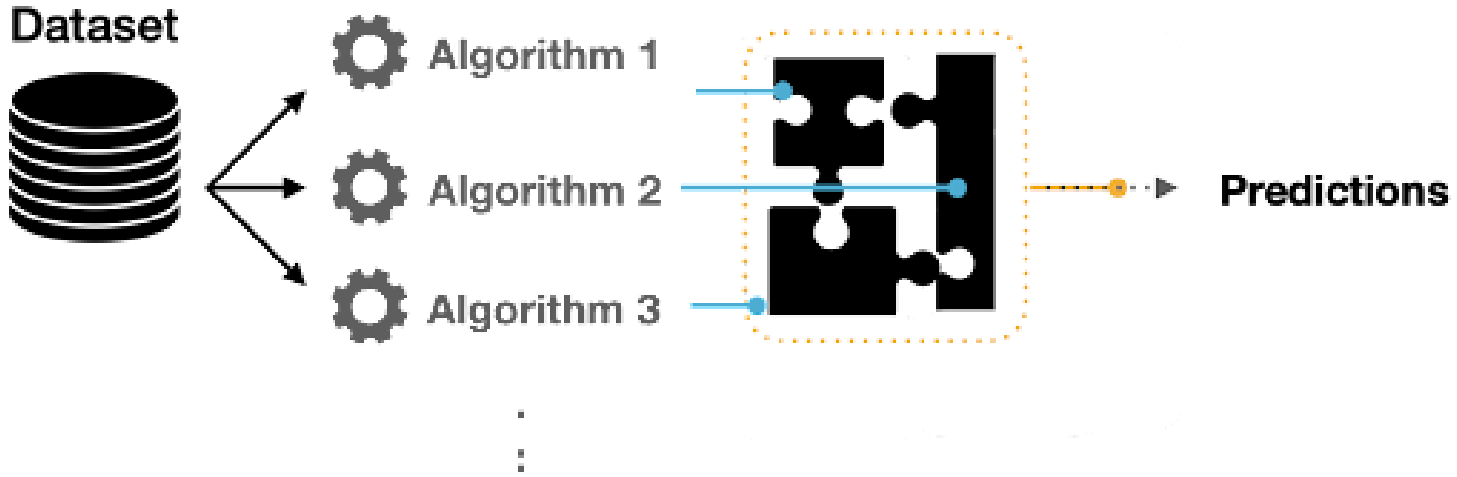
③ VLD=VotingClassifier(estimators=[('RF',RF),('GBC',GBC)],voting='soft

- 랜덤 포레스트 RF가 Voting 알고리즘의 데이터셋에 적용된다.

Ensemble

- 여러 개의 분류기를 생성하고, 그 예측을 결합함으로써 보다 정확한 예측을 도출하는 머신러닝의 대표적인 기법 중 하나이다.

- 약한 모델 여러 개를 조합하여 더 정확한 예측에 도움을 주는 방식이다.



Ensemble에는 크게 4가지(Voting, Bagging, Boosting, Stacking) 기법이 존재하는데,
그중에서 우리는 Boosting과 Voting 기법을 활용하였다!

그림 출처: https://builtin.com/machine-learning/ensemble-model

Boosting

- 약 분류기를 순차적(Sequential)으로 학습하는 앙상블 기법

- 예측을 반복하면서 잘못 예측한 데이터에 더 큰 가중치를 부여하여 오류를 개선한다.

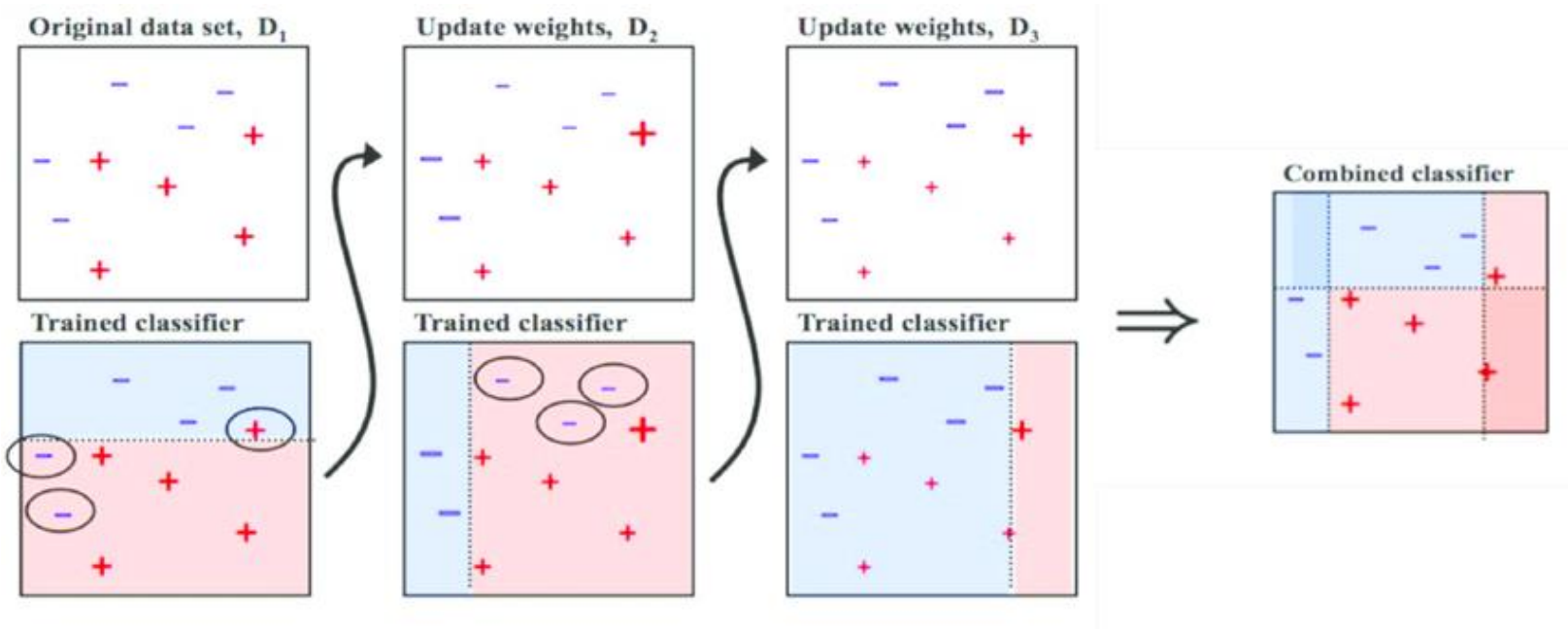


그림 출처: https://bkshin.tistory.com/entry/%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D-11-%EC%95%99%EC%83%81%EB%B8%94-%ED%95%99%EC%8A%B5-Ensemble-Learning-%EB%B0%B0%EA%B9%85Bagging%EA%B3%BC-%EB%B6%80%EC%8A%A4%ED%8C%85Boosting

Gradient Boosting

- Boosting 기법을 활용한 앙상블 모델 중 하나이다.

- 이전 모델의 residual(잔차)를 가지고 모델을 강화시키는 방법이며 residual을 예측하여 발전하는 weak learner 모델이다. (weak learner : 나쁜 모델로 성장해 나가는 것)

- 모델들은 계속해서 잔차(실제값과 예측 값의 차이)를 줄여가는 방향으로 학습한다.

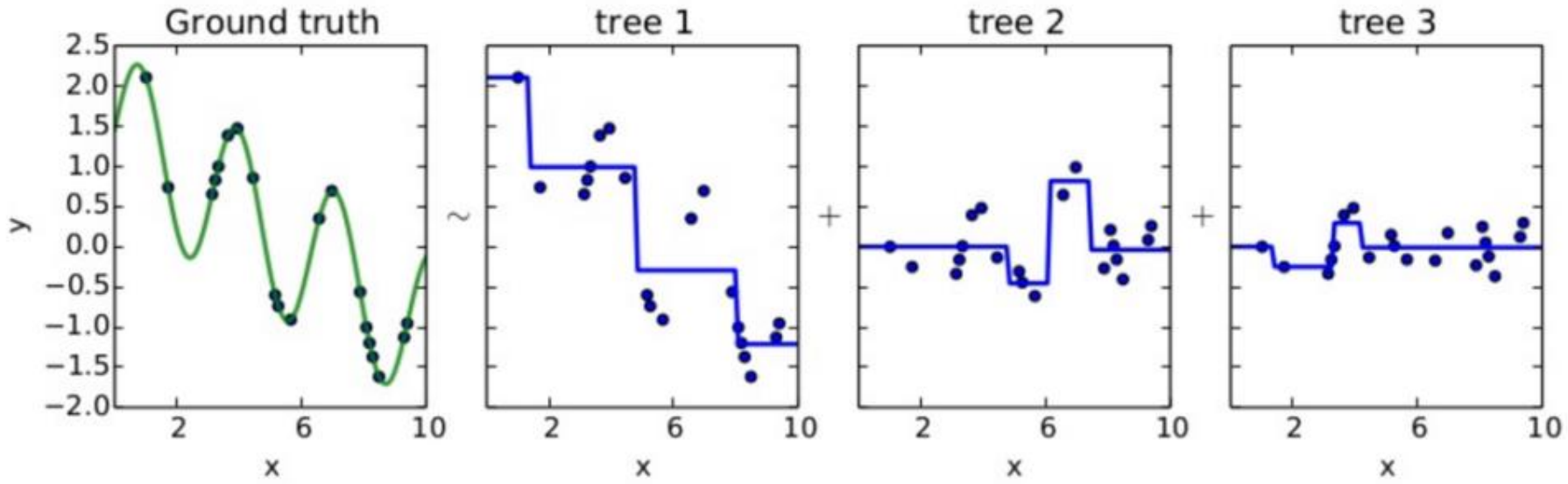
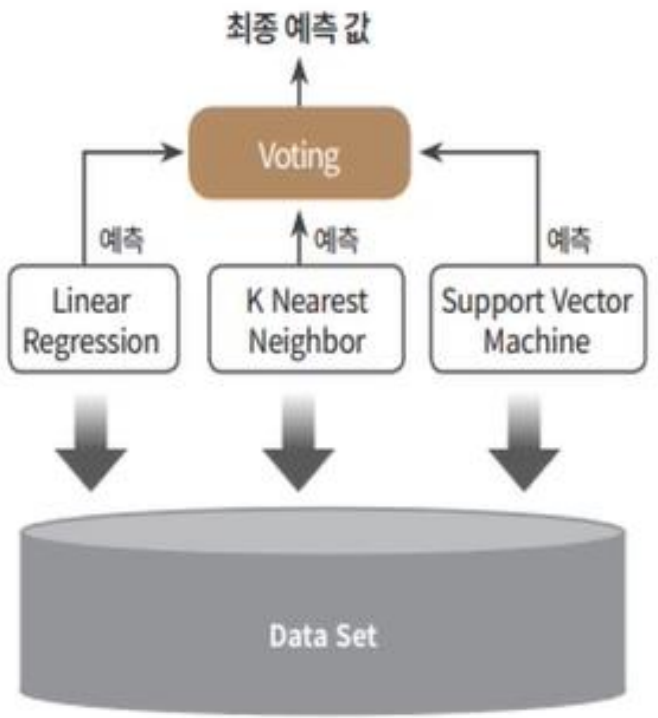


그림 출처: https://3months.tistory.com/368

Voting



Voting은 왼쪽 그림과 같이,
서로 다른 estimator들을 데이터셋에 적용해 예측값을 받고,
이를 합산하여 최종 예측값을 산출해내는 방식을 말한다.

(estimator: Ensemble 학습을 진행할 다양한 분류 모델)

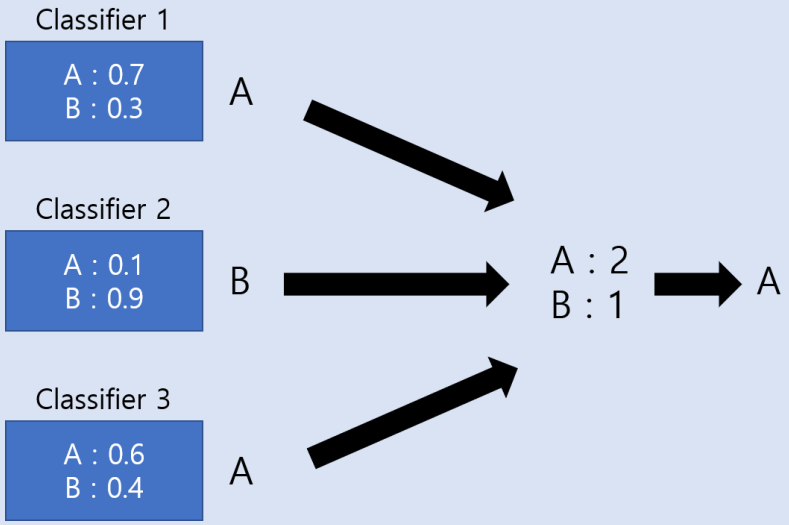
Voting의 종류에는 Hard Voting과 Soft Voting이 있다.

그림 출처: https://tbr74.tistory.com/entry/ensemble-learning-voting

Hard Voting vs **Soft Voting**

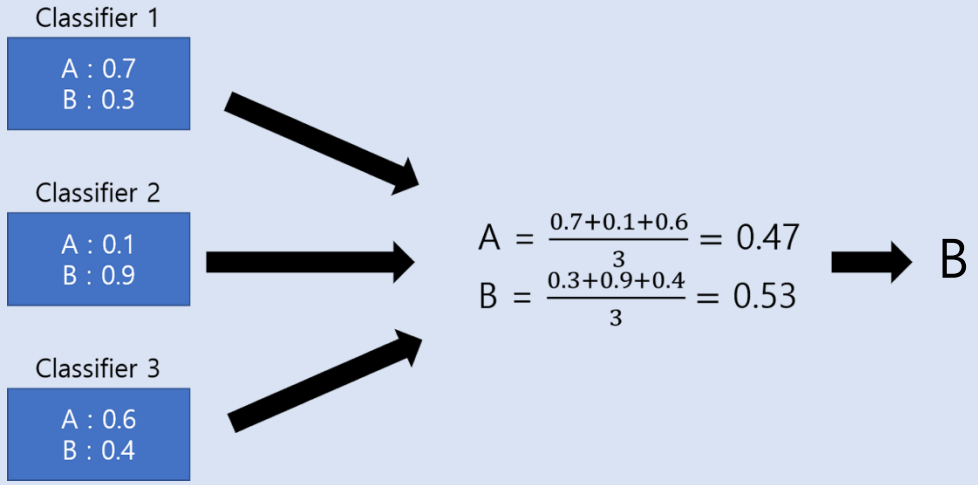
성능이 더 좋은
Soft Voting 방식
채택

Hard Voting



각각의 모델들이 결과를 예측하면
가장 많은 표를 얻은 결과를 선택하는 방식

Soft Voting



**각 class별로 모델들이 예측한 probability를
합산해서 가장 높은 class를 선택하는 방식**

그림 출처: https://devkor.tistory.com/entry/Soft-Voting-%EA%B3%BC-Hard-Voting