머신러닝2 과제 2025

(ver 1.2)

25. 5.20

학번 : 1958015
이름 : 박시형

Preview

>이전의 작업 요약

(ver 1.1)

1. **데이터 전처리**
   - process1_result.csv 사용, 피처: Sex, SibSp, Parch, Embarked, TicketNumeric
   - 결측치·인코딩 처리: Age 그룹 중앙값, Embarked 최빈값, Cabin 제외

---

2. **KNN 튜닝**
   - n-fold 교차검증 단계 제거
   - NumPy 브로드캐스트로 k 탐색 범위 확장(1→100)
   - 최적 k = 20 → Test1: 80.47%, Test2: 77.03%

---

3. **Logistic Regression 개선**
   - 배치 학습 → 인스턴스(온라인) SGD로 변환
   - 하이퍼파라미터: learning_rate=0.01, iterations=20, ε=1e-15
   - Loss 수렴(0.5024) 후 Test1: 79.12%, Test2: 77.51%

---

4. **Decision Tree 깊이 탐색**
   - Gini 불순도 기반 분할, max_depth 후보 탐색(3→20)
   - 최적 depth = 15 → Test1: 94.95%, Test2: 72.25%

---

5. **모델 성능 비교**
   - **Baseline (Threshold)**: Test1 72.95%, Test2 72.49%
   - **KNN**: +7.52% (Test1), +4.55% (Test2)
   - **Logistic**: +6.17% (Test1), +5.02% (Test2)
   - **Decision Tree**: +22.00% (Test1), +0.24% (Test2)

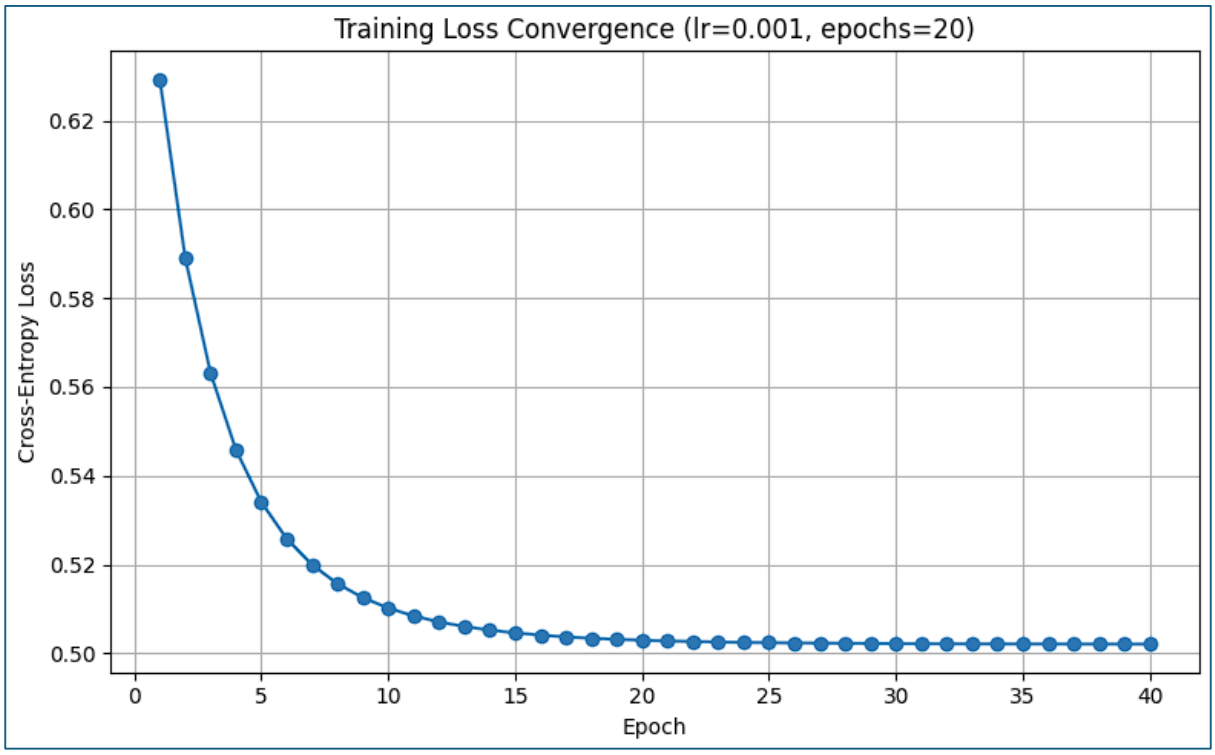
[교차검증]

> 하이퍼 파라미터를 안정적으로 선택, 모델의 일반화 성능을 추정하기 위해 사용한다.

전체데이터에 대해 k개로 나눈 fold를 만들고, 여기에 모델을 돌려 나온 결과에 대해 k개로 나눈다. (검증을 하기 위함.)

(로지스틱) – early stopping

교차검증을 진행하기 전에, 기존 로지스틱 회귀에서 과적합 되고 있던 현상을 수정하기 위해 변화율이 가장 작아지는 부분에서 학습을 멈춰야 한다.



테스트 1에 대한 로지스틱 결과

(학습률 : 0.001, 에폭스 : ~40)

과적합을 피하기 위해, 변화율이 급격히 작아지는 부분에서 멈춰야 한다.

변화율에 대해 표를 만들어보면

Epoch	Loss	$\Delta_n = Loss_{n-1} - Loss_n$
10	0.5102	–
11	0.5084	0.0018
12	0.5071	0.0013
13	0.5060	0.0011
14	0.5052	0.0008
15	0.5046	0.0006
16	0.5041	0.0005
17	0.5037	0.0004
18	0.5034	0.0003
19	0.5031	0.0003
20	0.5029	0.0002
21	0.5028	0.0001
...	...	<0.0001

이런식으로 만들어지고,

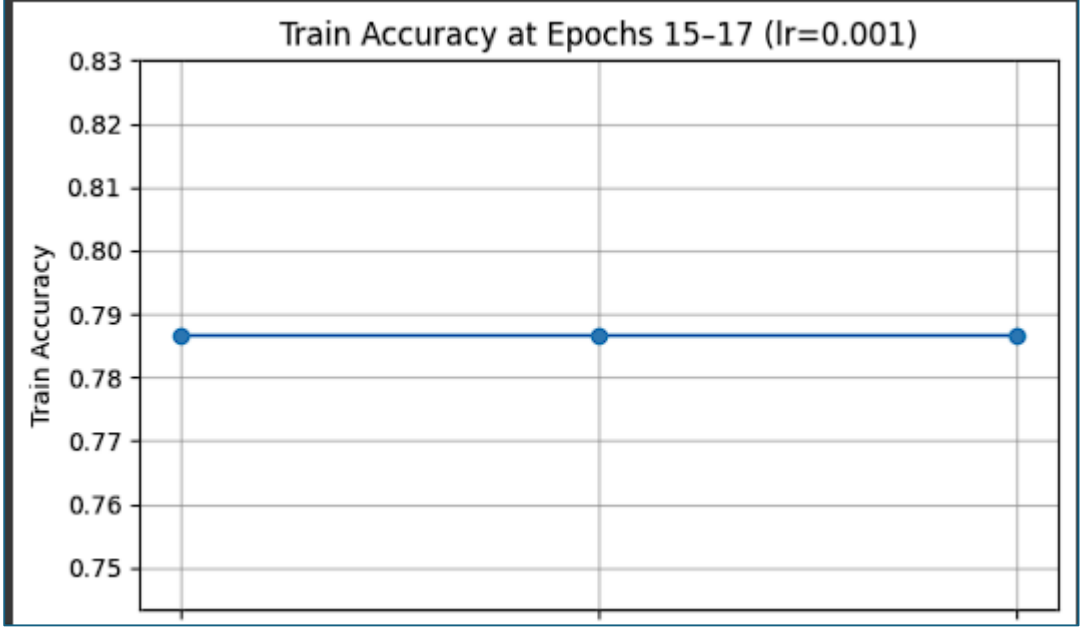
14 에폭 이후 : 0.001 이내,

17 에폭 이후 : 0.0005 이내

로 급격히 떨어진다.

따라서 15~17 에폭 구간에서 학습을 멈춰야 한다.

```
Epoch  1: Train ACC = 0.7823
Epoch  2: Train ACC = 0.7823
Epoch  3: Train ACC = 0.7823
Epoch  4: Train ACC = 0.7823
Epoch  5: Train ACC = 0.7823
Epoch  6: Train ACC = 0.7834
Epoch  7: Train ACC = 0.7834
Epoch  8: Train ACC = 0.7834
Epoch  9: Train ACC = 0.7868
Epoch 10: Train ACC = 0.7868
Epoch 11: Train ACC = 0.7868
Epoch 12: Train ACC = 0.7868
Epoch 13: Train ACC = 0.7868
Epoch 14: Train ACC = 0.7868
Epoch 15: Train ACC = 0.7868
Epoch 16: Train ACC = 0.7868
Epoch 17: Train ACC = 0.7868
```



결과는 15~17구간 정확도가 모두 같고,

에폭이 9부터 결과가 같음을 볼 수 있다.

초기의 목적이 과적합을 피하는 것이기 때문에, 에폭스는 9로 지정한다.

이후 교차검증을 진행한다.

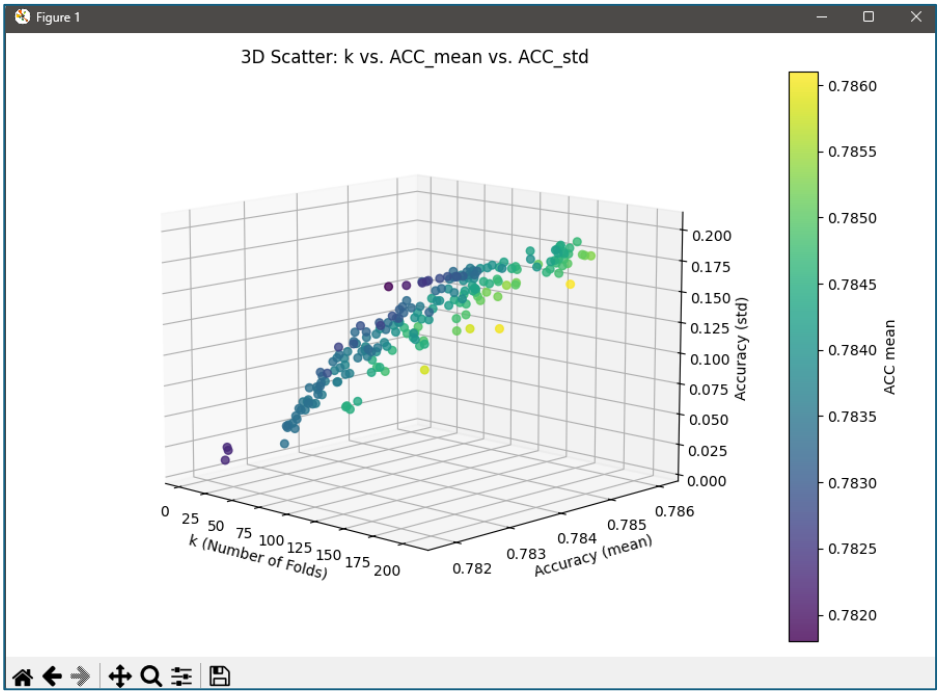
적절한 k 값을 찾기 위해 k=2부터 200까지 돌린다.

대략적인 결과는

```
k =  2 → ACC = 0.7823 ± 0.0087,  F1 = 0.7017 ± 0.0223
k =  3 → ACC = 0.7834 ± 0.0114,  F1 = 0.7027 ± 0.0253
k =  4 → ACC = 0.7823 ± 0.0199,  F1 = 0.7007 ± 0.0408
k =  5 → ACC = 0.7823 ± 0.0175,  F1 = 0.7014 ± 0.0313
k =  6 → ACC = 0.7834 ± 0.0257,  F1 = 0.7023 ± 0.0428
k =  7 → ACC = 0.7834 ± 0.0282,  F1 = 0.7026 ± 0.0437
k =  8 → ACC = 0.7845 ± 0.0329,  F1 = 0.7014 ± 0.0591
k =  9 → ACC = 0.7845 ± 0.0340,  F1 = 0.7026 ± 0.0552
k = 10 → ACC = 0.7833 ± 0.0288,  F1 = 0.7010 ± 0.0513
k = 11 → ACC = 0.7834 ± 0.0304,  F1 = 0.7021 ± 0.0451
k = 12 → ACC = 0.7845 ± 0.0316,  F1 = 0.7015 ± 0.0570
k = 13 → ACC = 0.7834 ± 0.0262,  F1 = 0.7014 ± 0.0462
k = 14 → ACC = 0.7834 ± 0.0371,  F1 = 0.7031 ± 0.0496
k = 15 → ACC = 0.7834 ± 0.0346,  F1 = 0.7025 ± 0.0489
k = 16 → ACC = 0.7835 ± 0.0484,  F1 = 0.6983 ± 0.0878
k = 17 → ACC = 0.7834 ± 0.0439,  F1 = 0.7014 ± 0.0678
k = 18 → ACC = 0.7835 ± 0.0420,  F1 = 0.7007 ± 0.0668
```

이런식으로 나온다. (+-는 표준편차를 의미)

이를 3d matrix로 시각화해봤다.



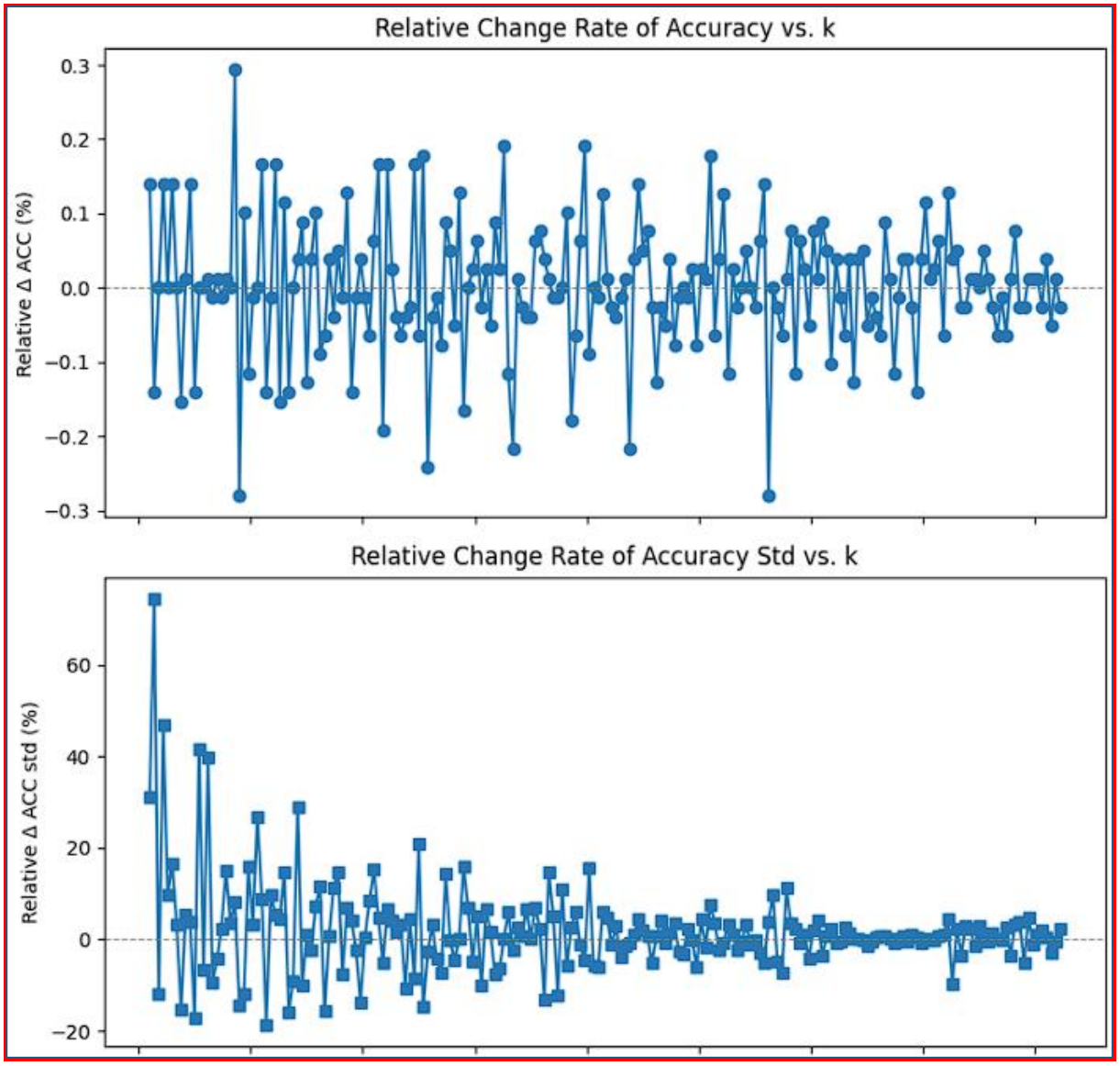
이런식으로 나오는데, 결국 적절한 k를 찾는 것은,

Acc가 높으면서, 표준편차가 크지 않은 값이다.

따라서 변화율에 대해서 한번 더 시각화를 해야하는데,

상단 : 평균 정확도(ACC mean)의 **상대 변화율**(ΔACC/ACC 이전×100%)

하단 : 정확도 표준편차(ACC std)의 **상대 변화율**(Δstd/std 이전×100%)

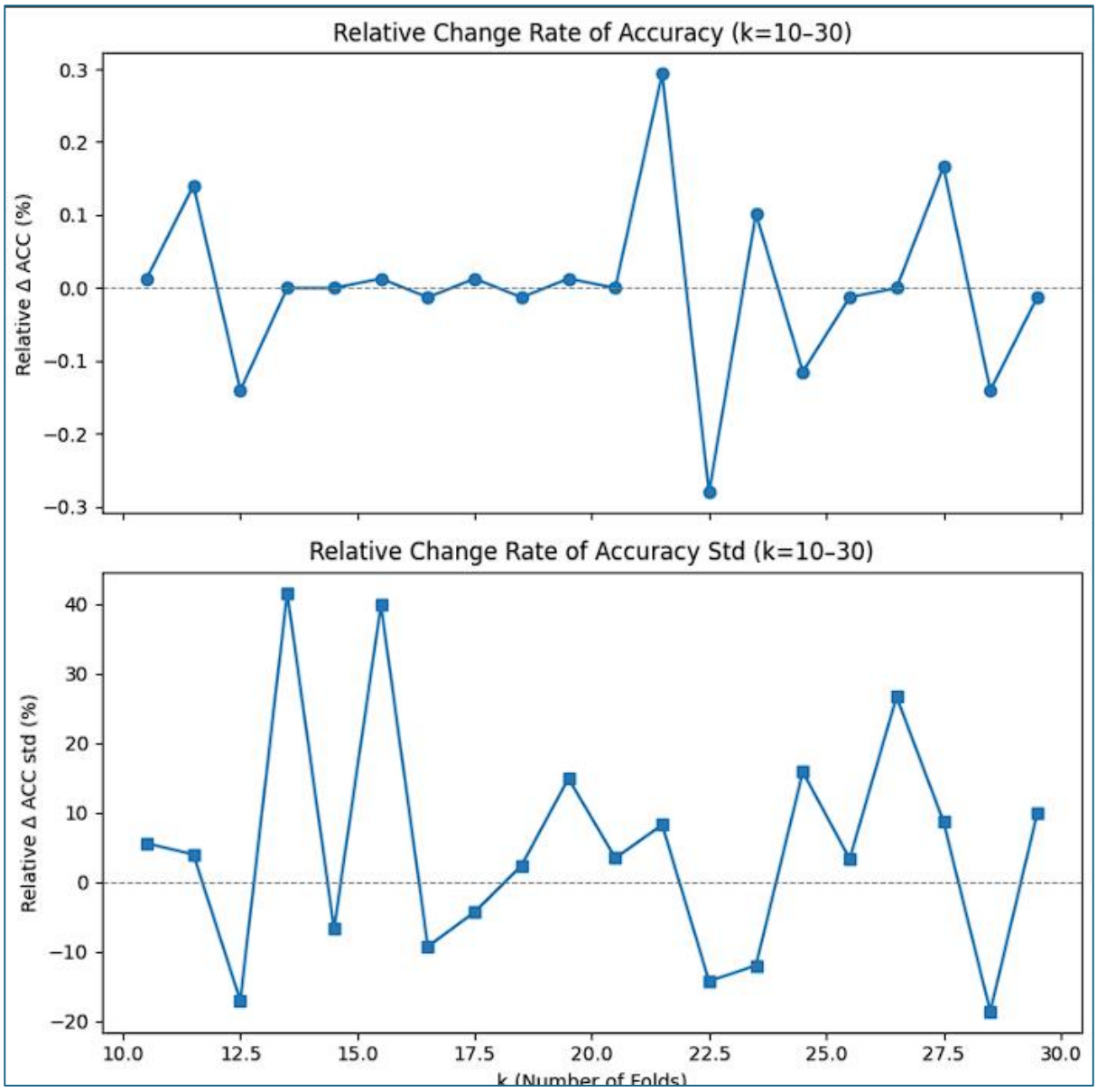


이전에 값에 대한 퍼센테이지 변화를 k 200 (x축) 까지 시각화 한 그림이다.

초기에 정확도 변화가 크다가, k=10 ~ 30 구간에 수렴하는 부분이 있고, 이 때 표준편차도 어느정도 작아지는 모습을 보인다.

K가 30 이상인 구간은 의미가 없음을 그래프로 확인할 수 있으므로,

K의 값을 10~30으로 지정하기로 한다.



위의 그림은 10~30으로 범위를 축소하여 이전 값에 대한 변화율을 그린 그래프이다.

표준편차 변화율이 작고, acc의 변화율이 어느정도 수렴하는 값인 18로 k를 지정한다.

K = 18,

Test1에 대한 acc는

```
Fold  1 — ACC: 0.8200, F1: 0.7692
Fold  2 — ACC: 0.8600, F1: 0.8000
Fold  3 — ACC: 0.7200, F1: 0.5882
Fold  4 — ACC: 0.7400, F1: 0.6667
Fold  5 — ACC: 0.8000, F1: 0.7059
Fold  6 — ACC: 0.7800, F1: 0.6857
Fold  7 — ACC: 0.7600, F1: 0.6667
Fold  8 — ACC: 0.7600, F1: 0.6000
Fold  9 — ACC: 0.7400, F1: 0.6061
Fold 10 — ACC: 0.8163, F1: 0.7568
Fold 11 — ACC: 0.7755, F1: 0.6667
Fold 12 — ACC: 0.7551, F1: 0.7000
Fold 13 — ACC: 0.7551, F1: 0.6842
Fold 14 — ACC: 0.7755, F1: 0.6857
Fold 15 — ACC: 0.7551, F1: 0.7143
Fold 16 — ACC: 0.8776, F1: 0.8500
Fold 17 — ACC: 0.8367, F1: 0.7647
Fold 18 — ACC: 0.7755, F1: 0.7027

=== Cross-Validation Results (k=18) ===
ACC: 0.7835 ± 0.0420
F1 : 0.7007 ± 0.0668
```

으로 나온다.

폴드를 나눈 뒤에, 1개의 폴드 안에서 서로 비교하고, 평균을 낸다. 이를 k=2부터 반복해서 18이라는 optimal 값을 찾아냈고, 이 18이라는 fold를 전체 데이터에 대해 18개로 분할하여 평균을 낸 값이 최종적인 test1에 대한 정확도인 것

최종 결과

(Test1)

Acc : 78.35

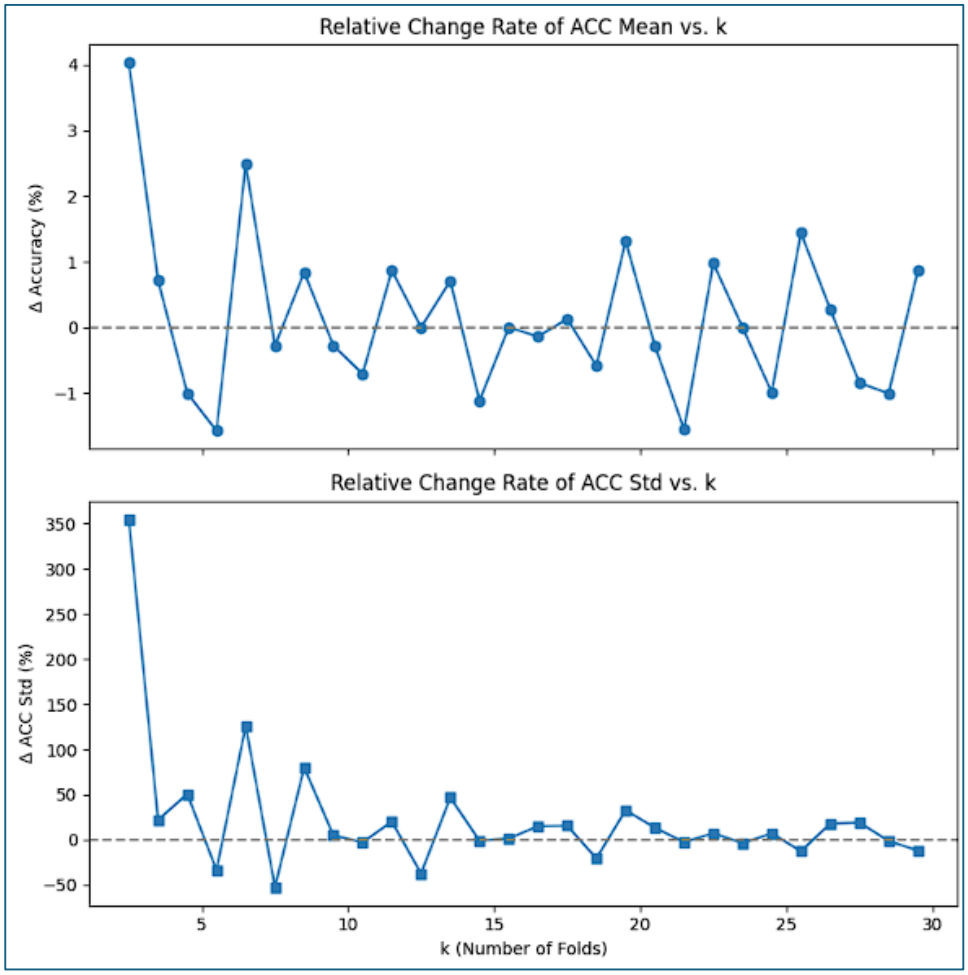
F1 : 70.07

(Decision Tree) – 교차검증

기존의 코드에서 k=2~30까지 돌려본 결과

```
k= 2 → ACC=0.7520±0.0031, F1=0.6730±0.0165
k= 3 → ACC=0.7823±0.0141, F1=0.6949±0.0287
k= 4 → ACC=0.7879±0.0172, F1=0.7104±0.0319
k= 5 → ACC=0.7800±0.0259, F1=0.6978±0.0380
k= 6 → ACC=0.7677±0.0172, F1=0.6849±0.0322
k= 7 → ACC=0.7868±0.0389, F1=0.7098±0.0508
k= 8 → ACC=0.7846±0.0185, F1=0.7036±0.0315
k= 9 → ACC=0.7912±0.0333, F1=0.7111±0.0542
k=10 → ACC=0.7890±0.0349, F1=0.7104±0.0542
k=11 → ACC=0.7834±0.0339, F1=0.7021±0.0588
k=12 → ACC=0.7902±0.0407, F1=0.7089±0.0623
k=13 → ACC=0.7902±0.0255, F1=0.7056±0.0491
k=14 → ACC=0.7958±0.0374, F1=0.7180±0.0620
k=15 → ACC=0.7869±0.0370, F1=0.7061±0.0556
k=16 → ACC=0.7869±0.0375, F1=0.7064±0.0600
```

이런식으로 나오고, 변화율에 대해 시각화해보면



이런식으로 나온다. 따라서 k는 16으로 지정.

```
--- 16-Fold Cross-Validation (Decision Tree) ---
Fold  1 — ACC: 0.8036, F1: 0.7442
Fold  2 — ACC: 0.8036, F1: 0.7442
Fold  3 — ACC: 0.7143, F1: 0.5789
Fold  4 — ACC: 0.7500, F1: 0.6818
Fold  5 — ACC: 0.7857, F1: 0.6842
Fold  6 — ACC: 0.7500, F1: 0.6667
Fold  7 — ACC: 0.7321, F1: 0.5946
Fold  8 — ACC: 0.8036, F1: 0.7179
Fold  9 — ACC: 0.8214, F1: 0.7727
Fold 10 — ACC: 0.8036, F1: 0.6857
Fold 11 — ACC: 0.8036, F1: 0.7317
Fold 12 — ACC: 0.8182, F1: 0.7500
Fold 13 — ACC: 0.8182, F1: 0.7619
Fold 14 — ACC: 0.8364, F1: 0.8000
Fold 15 — ACC: 0.8182, F1: 0.7368
Fold 16 — ACC: 0.7273, F1: 0.6512

=== Cross-Validation Results (k=16) ===
ACC: 0.7869 ± 0.0375
F1 : 0.7064 ± 0.0600
```

위가 결과이고, 따라서

DT에서 K=16으로 지정하여 나온 최종 결과는

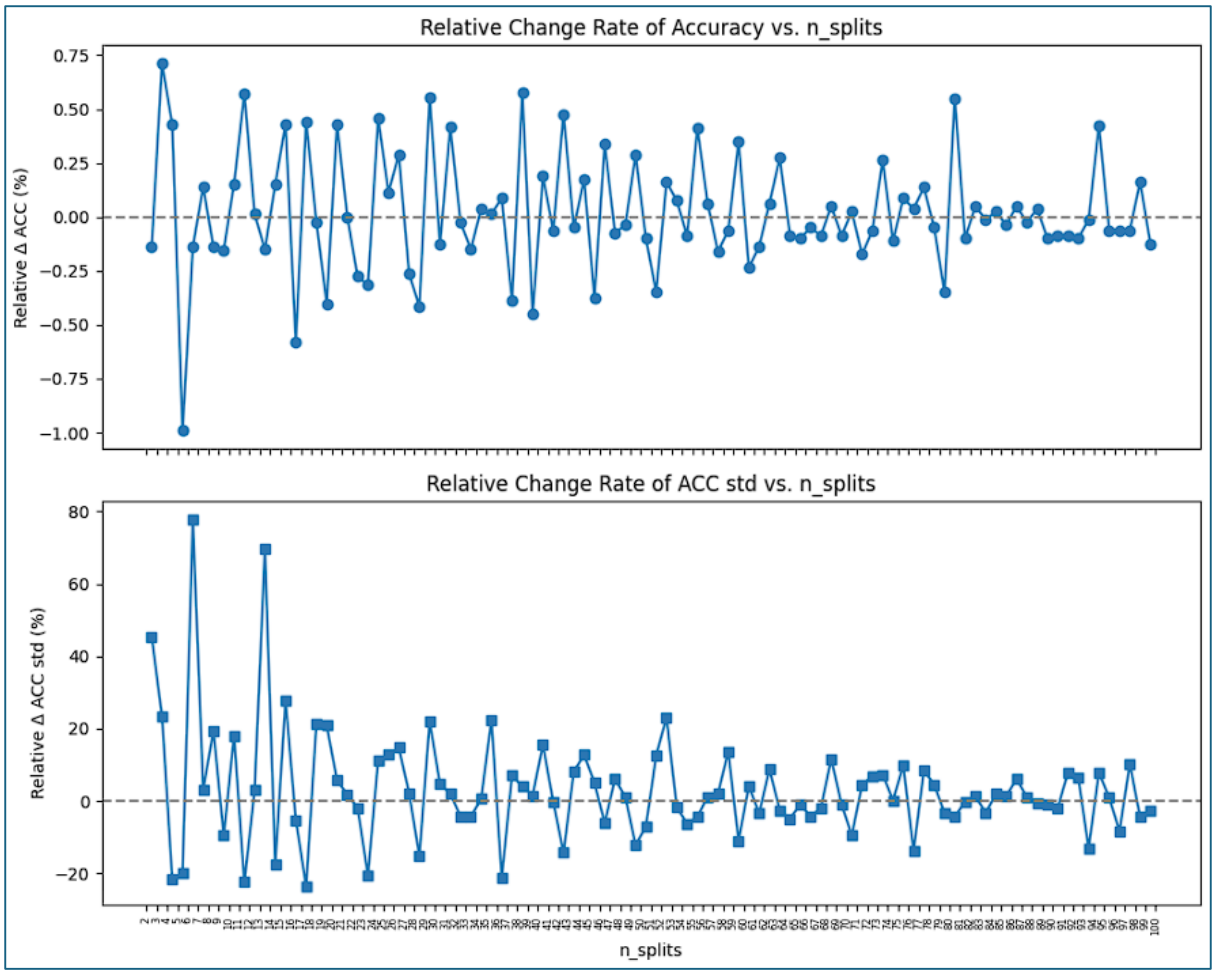
Acc : 78.69

F1 : 70.64

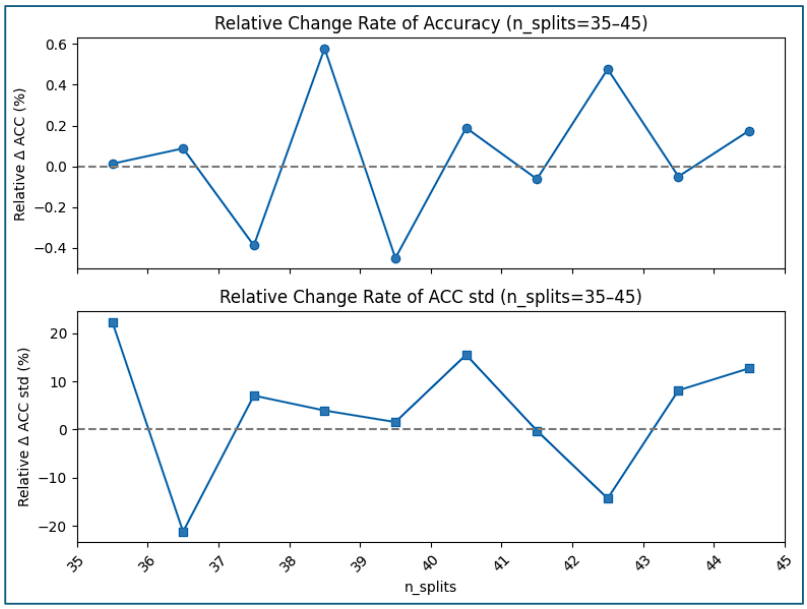
이다

(knn)

n_split : optimal k



범위 35~45로 축소



따라서 n_split은 37로 지정한다.

최종결과

```
Fold 32 — ACC: 0.7917, F1: 0.7368
Fold 33 — ACC: 0.9167, F1: 0.9000
Fold 34 — ACC: 0.8333, F1: 0.7778
Fold 35 — ACC: 0.8750, F1: 0.8235
Fold 36 — ACC: 0.7917, F1: 0.7368
Fold 37 — ACC: 0.7500, F1: 0.6250

=== Cross-Validation Results ===
ACC: 0.7979 ± 0.0639
F1 :  0.6970 ± 0.1219
```

Acc : 79.79

F1 : 69.70