EIS SoH Prediction 모델 개발

MLP

Index

1. 개요

2. EIS(임피던스) 기반 SoH 회귀분석

3. SoH 시계열 기반 RUL 회귀

4. EOL(End-of-Life) 비교

5. 결론 및 향후 연구

1. 개요

▪ **파이프라인 설계**

EIS(임피던스) 기반 SoH 회귀 → SoH 시계열 기반 RUL 회귀 → EOL(End-of-Life) 비교

▪ **EIS(임피던스) 기반 SoH 회귀**

- Re, Rct등의 EIS를 입력으로 하여 MLP Regressor를 사용해 SoH를 회귀분석

▪ **SoH 시계열 기반 RUL 회귀**

- SoH 예측 시계열에서 얻은 현재 SoH, 이동평균, 기울기를 입력으로 RUL을 회귀분석

▪ **EOL(End-of-Life) 비교**

- 예측된 SoH를 Median rolling으로 smoothing해서 SoH ≤ 80%를 만족하는 사이클을 EOL로 판정

1. 개요

▪ 파이프라인 설계

EIS(임피던스) 기반 SoH 회귀 → SoH 시계열 기반 RUL 회귀 → EOL(End-of-Life) 비교

▪ 전처리

- 랜덤으로 60:20:20으로 배터리셋 분리 (seed: 42)
- metadata 스키마 정규화
- 각 배터리의 discharge(방전) 파일을 start_time → file_num 순으로 정렬하고순서대로 cycle_index = 0, 1, 2, … 부여
- 임피던스·충전 데이터를 가장 가까운 방전 사이클과 시간 기준(Δt ≤ 6시간) 으로 매핑
- 초기 3사이클 중앙값으로 Q0, SoH 물리 클리핑(0.4~1.25)

2. EIS(임피던스) 기반 SoH 회귀

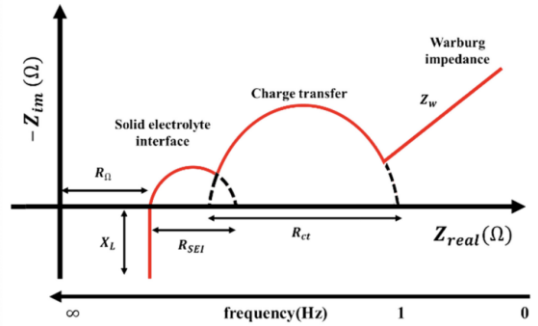
▪ EIS 기반 SoH 회귀분석

초반 3사이클로 base_re, rct를 지정해서 배터리 자체의 고유 저항값이 아닌 해당 배터리에서의 기준을 잡고 비교

```python
def add_rel(g: pd.DataFrame) -> pd.DataFrame:
    g = g.sort_values("cycle_index").copy()
    base_re  = np.nanmedian(g["Re_norm"].iloc[:max(1, min(3, len(g)))])
    base_rct = np.nanmedian(g["Rct_norm"].iloc[:max(1, min(3, len(g)))])
    if not np.isfinite(base_re):  base_re  = 1.0
    if not np.isfinite(base_rct): base_rct = 1.0
    g["Re_rel"]  = g["Re_norm"]  / base_re
    g["Rct_rel"] = g["Rct_norm"] / base_rct
    # 차분
    g["dRe"]  = g["Re_norm"].diff()
    g["dRct"] = g["Rct_norm"].diff()
    g["dRe"]  = winsorize_series(g["dRe"],  0.01)
    g["dRct"] = winsorize_series(g["dRct"], 0.01)
    # 롤링
    w = 5
    for col in ["Re_norm","Rct_norm"]:
        g[f"{col}_roll_std"]   = g[col].rolling(w, min_periods=1).std()
        g[f"{col}_roll_delta"] = g[col] - g[col].rolling(w, min_periods=1).mean()
    return g

feat = feat0.groupby("battery_id", group_keys=False).apply(add_rel).reset_index(drop=True)
```

극단값을 제외하기 위해 winsorize하였음.[1)]

- 처음엔 아예 제외하려고 하였으나, EIS 데이터는 이상치와 진짜 갑자기 커지는 경우도 있기 때문에 값은 남기되, weight를 줄이는 방식으로 결정하였음.



1) Hampel, F. R. (1974). *The influence curve and its role in robust estimation. Journal of the American Statistical Association, 69*(346), 383–393. https://doi.org/10.1080/01621459.1974.10482962

2. EIS(임피던스) 기반 SoH 회귀

- **EIS 기반 SoH 회귀분석**

초반 3사이클로 base_re, rct를 지정해서 배터리 자체의 고유 저항값이 아닌 해당 배터리에서의 기준을 잡고 비교

```python
def add_rel(g: pd.DataFrame) -> pd.DataFrame:
    g = g.sort_values("cycle_index").copy()
    base_re  = np.nanmedian(g["Re_norm"].iloc[:max(1, min(3, len(g)))])
    base_rct = np.nanmedian(g["Rct_norm"].iloc[:max(1, min(3, len(g)))])
    if not np.isfinite(base_re):  base_re  = 1.0
    if not np.isfinite(base_rct): base_rct = 1.0
    g["Re_rel"]  = g["Re_norm"]  / base_re
    g["Rct_rel"] = g["Rct_norm"] / base_rct
    # 차분
    g["dRe"]  = g["Re_norm"].diff()
    g["dRct"] = g["Rct_norm"].diff()
    g["dRe"]  = winsorize_series(g["dRe"],  0.01)
    g["dRct"] = winsorize_series(g["dRct"], 0.01)
    # 롤링
    w = 5
    for col in ["Re_norm","Rct_norm"]:
        g[f"{col}_roll_std"]   = g[col].rolling(w, min_periods=1).std()
        g[f"{col}_roll_delta"] = g[col] - g[col].rolling(w, min_periods=1).mean()
    return g

feat = feat0.groupby("battery_id", group_keys=False).apply(add_rel).reset_index(drop=True)
```

최근 5사이클 동안 얼마나 데이터가 출렁이는지 확인

Re_norm이나 Rct_norm의 표준 편차를 비교하여 특정 값이 넘어가면 불안정한 것으로 확인.

단기 추세를 반영하고, 노이즈를 제거하기 위함

ex) roll_delta가 높아지면, 최근 평균보다 값이 상승한 것

> 고장 근접

2. EIS(임피던스) 기반 SoH 회귀

- EIS 기반 SoH 회귀분석

```
Xcols = [
    "Re_norm","Rct_norm","Rct_over_Re","log_Re","log_Rct",
    "Re_rel","Rct_rel","dRe","dRct",
    "Re_norm_roll_std","Rct_norm_roll_std",
    "Re_norm_roll_delta","Rct_norm_roll_delta",
    "delta_T","cycle_norm"
]
ycol = "SoH"

keep = [c for c in Xcols if c in df.columns]
df = df.dropna(subset=keep+[ycol]).copy()

counts = df.groupby("split").size().to_dict()
print("[INFO] rows per split:", counts)

train = df[df["split"].eq("train")].copy()
val   = df[df["split"].eq("val")].copy()
test  = df[df["split"].eq("test")].copy()
if len(train) == 0:
    raise RuntimeError("Train split이 비었습니다. 분할/매핑/라벨을 확인하세요.")
```

입력 스케일 표준화(StandardScaler) 와 2층 MLP 회귀 모델 (64, 32 노드) 을 하나의 파이프라인으로 묶어 학습.

훈련 데이터만으로 모델을 fit() 하고, 조기 종료(early stopping)와 내부 검증(validation_fraction = 0.12) 으로 과적합을 방지한다.

훈련이 완료되면 train·val·test 세 구간 각각에 대해 SoH 예측값(SoH_hat_raw) 을 계산한다.

이 예측값은 이후 캘리브레이션과 EOL 평가 단계의 입력으로 사용된다.

2. EIS(임피던스) 기반 SoH 회귀

- EIS 기반 SoH 회귀분석

```
pipe = Pipeline([
    ("scaler", StandardScaler()),
    ("mlp", MLPRegressor(
        hidden_layer_sizes=(64,32),
        activation="relu",
        alpha=1e-3,
        learning_rate_init=5e-4,
        max_iter=300,
        random_state=42,
        early_stopping=True,
        n_iter_no_change=20,
        validation_fraction=0.12,
        verbose=False
    ))
])

pipe.fit(train[keep], train[ycol])

# 예측: 각 분할에 직접 컬럼 추가 + df에도 반영
yhat_train = pipe.predict(train[keep]); train["SoH_hat_raw"] = yhat_train
yhat_val   = pipe.predict(val[keep])  if len(val)>0 else np.array([])
if len(val)>0: val["SoH_hat_raw"] = yhat_val
yhat_test  = pipe.predict(test[keep]) if len(test)>0 else np.array([])
if len(test)>0: test["SoH_hat_raw"] = yhat_test

df.loc[train.index, "SoH_hat_raw"] = yhat_train
if len(val)>0:  df.loc[val.index,   "SoH_hat_raw"] = yhat_val
if len(test)>0: df.loc[test.index,  "SoH_hat_raw"] = yhat_test
```

- 입력 스케일 표준화(StandardScaler) 와 2층 MLP 회귀 모델 (64, 32 노드) 을 하나의 파이프라인으로 묶어 학습
- 훈련 데이터만으로 모델을 fit() 하고, 조기 종료(early stopping)와 내부 검증(validation_fraction = 0.12) 으로 과적합을 방지
- 훈련이 완료되면 train·val·test 세 구간 각각에 대해 SoH 예측값(SoH_hat_raw) 을 계산
- 이 예측값은 이후 캘리브레이션과 EOL 평가 단계의 입력으로 사용된다.

2. EIS(임피던스) 기반 SoH 회귀

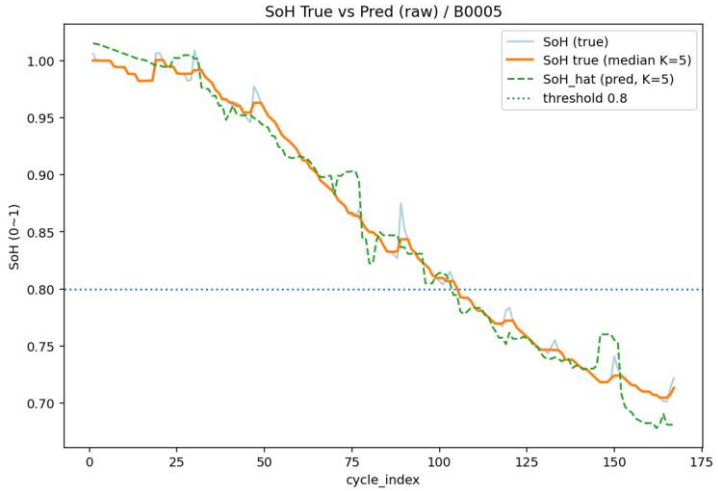
- EIS 기반 SoH 회귀분석

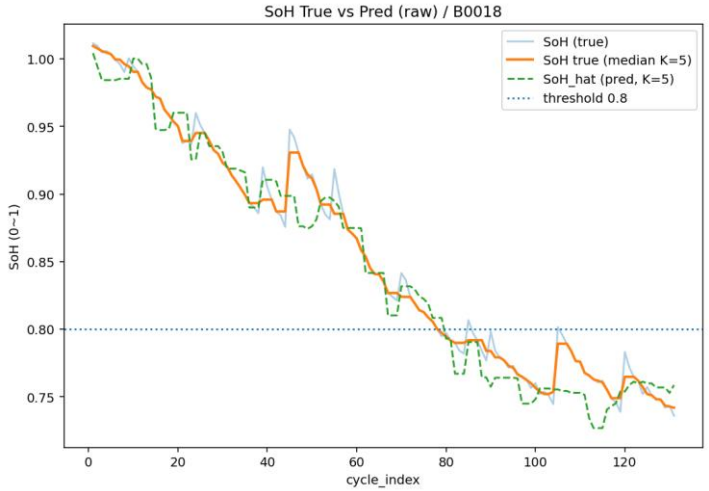
```
pipe = Pipeline([
    ("scaler", StandardScaler()),
    ("mlp", MLPRegressor(
        hidden_layer_sizes=(64,32),
        activation="relu",
        alpha=1e-3,
        learning_rate_init=5e-4,
        max_iter=300,
        random_state=42,
        early_stopping=True,
        n_iter_no_change=20,
        validation_fraction=0.12,
        verbose=False
    ))
])

pipe.fit(train[keep], train[ycol])

# 예측: 각 분할에 직접 컬럼 추가 + df에도 반영
yhat_train = pipe.predict(train[keep]); train["SoH_hat_raw"] = yhat_train
yhat_val   = pipe.predict(val[keep])  if len(val)>0 else np.array([])
if len(val)>0: val["SoH_hat_raw"] = yhat_val
yhat_test  = pipe.predict(test[keep]) if len(test)>0 else np.array([])
if len(test)>0: test["SoH_hat_raw"] = yhat_test

df.loc[train.index, "SoH_hat_raw"] = yhat_train
if len(val)>0:  df.loc[val.index,   "SoH_hat_raw"] = yhat_val
if len(test)>0: df.loc[test.index,  "SoH_hat_raw"] = yhat_test
```

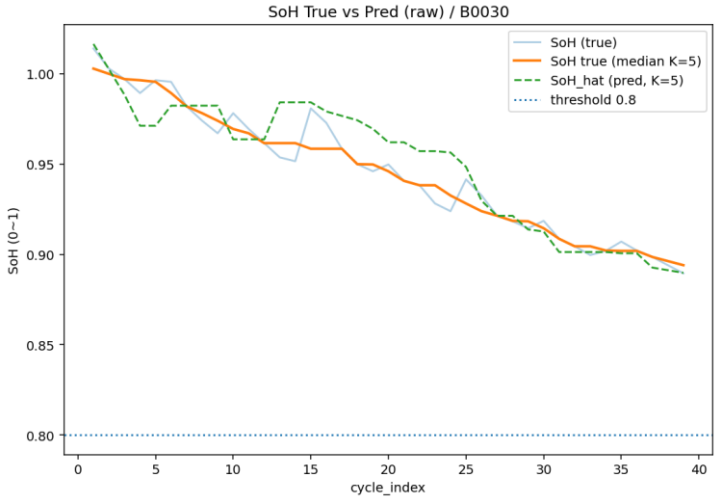
- 입력 스케일 표준화(StandardScaler) 와 2층 MLP 회귀 모델 (64, 32 노드) 을 하나의 파이프라인으로 묶어 학습
- 훈련 데이터만으로 모델을 fit() 하고, 조기 종료(early stopping)와 내부 검증(validation_fraction = 0.12) 으로 과적합을 방지
- 훈련이 완료되면 train·val·test 세 구간 각각에 대해 SoH 예측값(SoH_hat_raw) 을 계산
- 이 예측값은 이후 캘리브레이션과 EOL 평가 단계의 입력으로 사용된다.

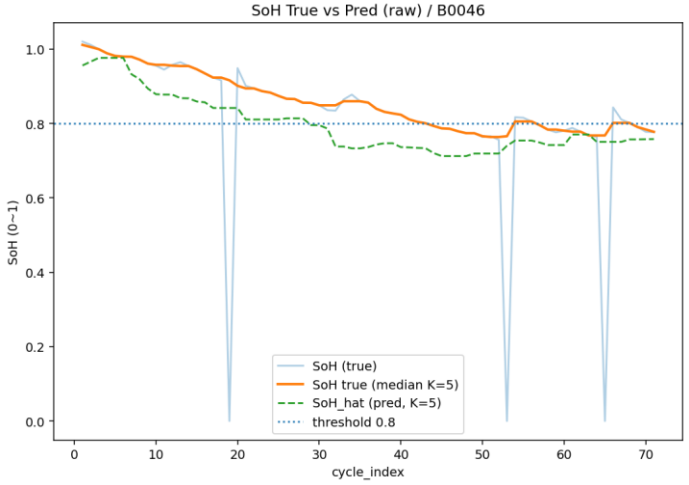
2. EIS(임피던스) 기반 SoH 회귀

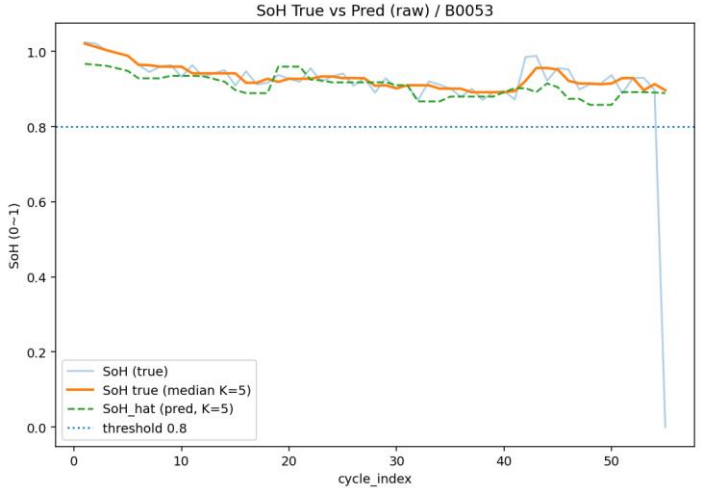
- 결과









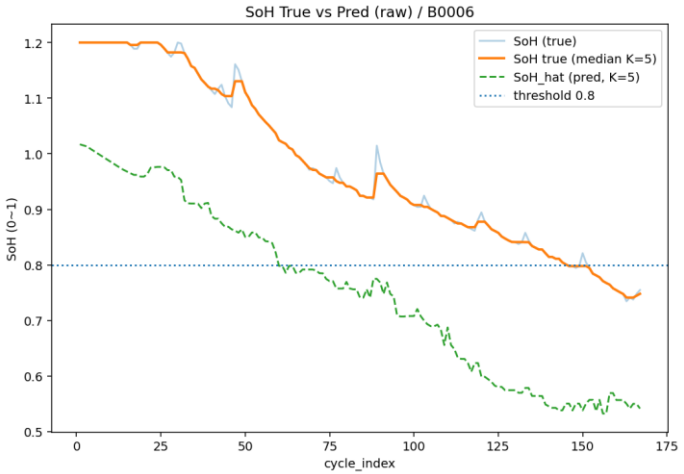


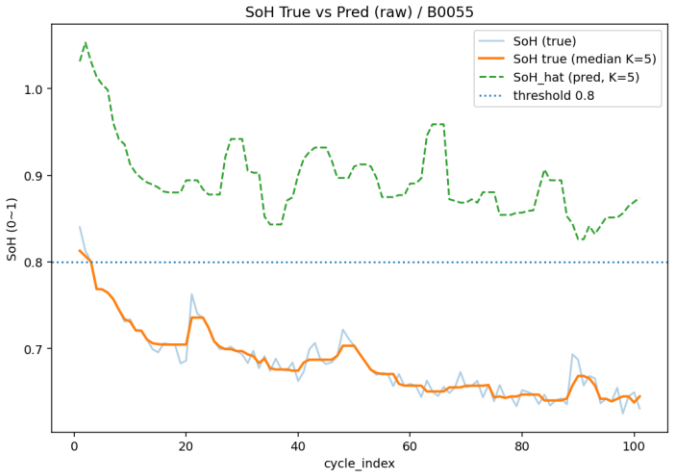
사이클 수도 어느 정도 있고, 기존 SoH가 자연스럽게 우하향하는 경우 추세를 어느정도는 따라감.

2. EIS(임피던스) 기반 SoH 회귀

- 결과

하지만 특정 배터리셋에서는 offset이 다르게 나타나고 추세만 어느정도 맞춰가는 현상 발생

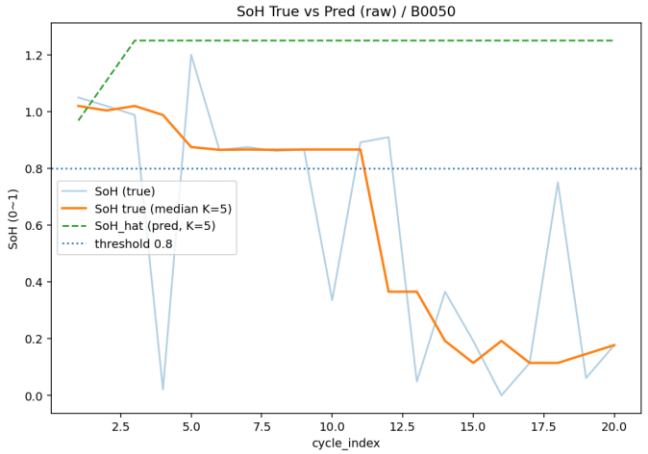


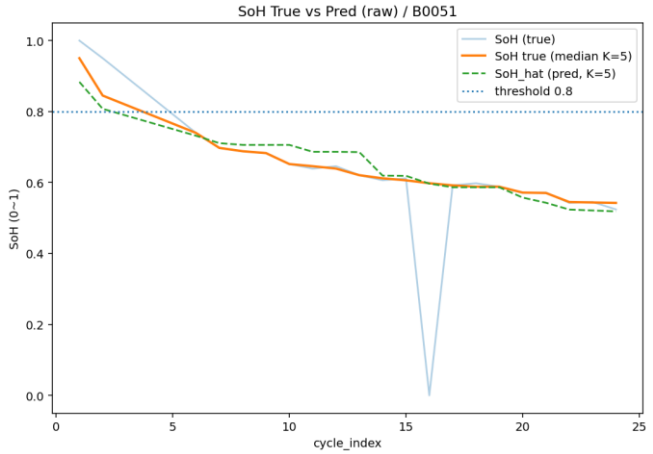


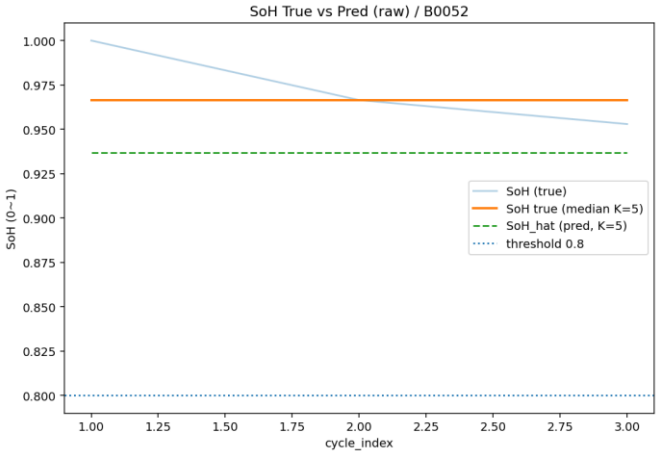
2. EIS(임피던스) 기반 SoH 회귀

- 결과

심지어 사이클 수가 너무 적은 경우에는 아예 헤메는 모습을 보였음.







2. EIS(임피던스) 기반 SoH 회귀

- 결과

추후 개선사항

1. offset 현상 제거.

- 배터리별 임피던스 레벨(Re, Rct)의 초기 기준점(Q_0) 이 달라서 생긴 문제로 추정
- 학습 후 각 배터리의 (SoH_true − SoH_hat) 평균값을 계산해 보정

2. 매핑 문제 해결하여 터무니없이 적은 사이클을 가진 배터리 문제 해결

- 임피던스–방전 매핑 조건($\Delta t \le 6h$ 또는 ±2 ord)이 너무 좁아EIS 횟수가 적은 배터리는 매칭 실패 → 사이클이 3개 등으로 줄어듦.
- 보간이나 윈도우 확대 등으로 사이클 매핑 실패를 줄일 예정

3. 전반적인 성능 개선

- 노이즈 개선 등을 위해 사용된 보정이 너무 많은 것 같아 이를 줄이고, 상관이 낮은 피처를 제거해보기도 할 예정

3. SoH 시계열 기반 RUL 회귀

- **EIS 회귀분석을 통해 산출된 SoH를 통해 RUL을 회귀**

- 20_train_rul_mlp.py에서는 SoH 예측 결과를 기반으로 최근 5사이클의 평균 및 기울기를 계산, 남은 수명(RUL)을 회귀 모델로 추정하였다.
- 2층 MLP(32,16, ReLU) 구조를 사용해 열화 추세를 학습하였으며, RUL은 사이클 진행에 따라 반드시 감소하도록 단조성 보정을 적용했다.
- 검증 및 테스트 세트에 대해 MAE를 계산하고, 최종 결과(rul_mlp_preds.csv)와 메타정보(model_card_rul.json)를 함께 저장하였다.


```
{
  "val_MAE": 9.470552697842814,
  "test_MAE": 36.19127368116006
}
```


- RUL-MLP 모델은 검증셋 기준 평균 9.47 사이클의 오차로 안정적인 예측 성능을 보였으나,테스트셋에서는 36.19 사이클로 오차가 증가하였다.
- SoH_hat의 예측 정확도로 비롯한 오차도 있겠으나, 자세한 것은 더 살펴보아야할 것 같다.

4. EOL(End-of-Life) 비교

▪ **예측된 SoH 시계열(SoH_hat) 로부터EOL(SoH=0.8 교차 시점) 을 추정하고,실제 EOL(라벨 기준)과 비교하여 모델의 수명 예측 정확도를 평가**

- 해당 단계에서는 SoH 회귀 결과를 기반으로임계값 SoH=0.8을 기준으로 한 EOL(End-of-Life) 시점을 산출하였다.
- 배터리별 SoH 예측 곡선을 5사이클 이동중앙값으로 평활화한 뒤, 임계선과 최초로 교차하는 사이클을 예측 EOL로 정의하고실제 EOL(라벨 기준)과의 차이로 MAE, Median AE, Hit@±5/±10을 평가하였다.
- 이를 통해 모델의 수명 예측 일관성 및 운영 신뢰성을 정량적으로 검증하였다.

battery_id	split	EOL_cycle	pred_eol	error
B0006	test	146	167	21
B0007	test	167	120	47
B0025	test	27	27	0
B0029	test	39	39	0
B0034	test	0	0	0
B0054	test	0	100	100
B0055	test	0	100	100
B0005	train	106	82	24
B0018	train	79	129	50
B0026	train	5	27	22
B0027	train	27	0	27
B0028	train	27	27	0
B0030	train	39	39	0
B0031	train	39	39	0

① EOL_cycle=0으로 기록된 배터리
② EIS 매핑이 안 되어 사이클이 거의 없는 배터리
③ SoH가 0.8에 도달하지 않은 배터리

에서 오차가 발생하고 있는 것을 알 수 있다.
이를 해결하기 위해,
- SoH 보정 캘리브레이션 점검, 평활화 윈도우를 줄이거나 교차 보건 등을 적용하여 EOL 위상차 맞춤.
- EIS 사이클 부족의 경우 매핑 허용폭을 확장하거나 보간하여 매핑 실패를 줄임.

근본적으로 SoH 예측의 정확도를 올린 다음에야 진행가능해보임.

5. 결론 및 향후 연구

결과

▪ EIS(임피던스) 기반 SoH 회귀 → SoH 시계열 기반 RUL 회귀 → EOL 비교의 세 단계로 구성된MLP 기반 ESS 배터리 고장 진단·수명 예측 파이프라인을 구축하였다.

▪ EIS 기반 SoH 회귀 단계

- SoH_hat은 전체 추세를 일정 수준 재현하였으나, 배터리별 offset 현상과 임피던스 매핑 실패에 따른 데이터 불균형이 잔존하였다.

▪ SoH 시계열 기반 RUL 회귀 단계

- 최근 5사이클 SoH 평균 및 기울기를 이용해 남은 수명(RUL)을 예측하였다.
- 검증셋 MAE는 약 9.47사이클로 양호했으나, 테스트셋 MAE는 36.19사이클로 증가하여 배터리 간 분포 차이(domain shift)에 따른 일반화 한계가 나타났다.

▪ EOL(End-of-Life) 비교 단계

- SoH 예측 곡선을 5사이클 이동중앙값으로 평활화한 뒤, SoH=0.8 임계치와의 교차 시점을 EOL로 산출하였다.
- 다수의 배터리에서 ±10사이클 이내 정확도를 보였으나, SoH 미도달 배터리(B0054, B0055 등) 및 EIS 데이터 부족 배터리에서

큰 오차가 발생하였다.

5. 결론 및 향후 연구

추후 개선사항

▪ SoH 회귀 단계 개선(우선순위 1)

- 배터리별 offset 보정(Bias Calibration):각 배터리의 (SoH_true − SoH_hat) 평균을 계산해 보정하거나,battery_id 임베딩(one-hot → dense)으로 모델이 자체 학습하도록 설계.

- EIS–Cycle 매핑 보완:Δt 허용폭을 ±12h 또는 ±3 order로 확장하고,인접 임피던스 간 시간가중 보간(temporal interpolation) 으로 결측 cycle 보강.

- Feature Selection 재검토:상관이 낮거나 중복된 피처(log_Re, Rct_over_Re 등) 제거 후 재학습.

▪ RUL 회귀 단계 고도화 (우선순위 2)

- 상대 RUL 정규화:RUL을 절대 cycle 수가 아니라 EOL 대비 비율(%)로 학습 → 배터리 간 수명 차이 보정.

- 가변 윈도우(K=3,5,10) 피처 확장:단기/중기 추세를 함께 반영해 열화 속도 변화를 더 잘 포착.

▪ EOL 평가 단계 개선 (우선순위 3)

- 전 단계들의 개선 이후 살펴볼 예정

Thank You