로지스틱 & KNN
분석 과정 및 결과

이세은

로지스틱 회귀 모델

- 모델 적합 후 성능 평가까지 소요시간
  - 센서 1개 조합: 1초
  - 센서 2개 조합: 19초
  - 센서 3개 조합: 5분 46초
  - 센서 4개 조합: 1시간 12분 57초
  - 센서 5개 조합: 13시간 19분 54초 (예상)

=> 센서 6개 조합, 센서 7개 조합, ... 센서 60개 조합까지 모든 조합들의 경우의 수에 대한 모델 성능을 평가하려고 했으나, 시간이 너무너무너무x10000000 많이 걸림.
=> 이런 방식은 불가능. 방식을 바꾸기로 결정.

현재까지 센서 1개 조합부터 센서 4개 조합까지에 대해 모델 성능을 평가해본 상태임. 이 결과를 바탕으로, 어떤 센서를 택해야 할지 후보군을 추려보자!

Test Accuracy가 최대인 경우를 뽑아봤을 때 다음과 같다. 6가지 경우 모두, test accuracy가 0.88로 높은 편이며, FNR은 0.1로 매우 낮은 편이다.(사진 3번, 4번의 값 분포를 보면 알 수 있다.)

Sensors	Accuracy	FNR
(10, 11, 36, 44)	0.880952	0.111111
(10, 26, 36, 45)	0.880952	0.111111
(10, 36, 42, 45)	0.880952	0.111111
(12, 17, 34, 43)	0.880952	0.111111
(12, 33, 43, 46)	0.880952	0.111111
(12, 35, 43, 48)	0.880952	0.111111

```
11100 23400 8100 6900
11100 3000 8100 4200
11100 8100 9000 4200
28500 7500 9900 7800
28500 15900 7800 14700
28500 6900 7800 21900
```

Price

```
df_no['Accuracy'].describe()

count    523685.000000
mean          0.612918
std           0.082826
min           0.333333
25%           0.547619
50%           0.619048
75%           0.666667
max           0.880952
Name: Accuracy, dtype: float64
```

```
df_no['FNR'].describe()

count    523685.000000
mean          0.387858
std           0.099883
min           0.000000
25%           0.333333
50%           0.407407
75%           0.444444
max           0.703704
Name: FNR, dtype: float64
```

=> 10, 11, 36, 44, 26, 45, 42, 12, 17, 34, 43, 33, 46, 35, 48 센서에 대해서만 조합을 만들어 모델 성능을 평가해보자.

모든 조합에 대해 모델을 만든 후, Test Accuracy가 최대인 경우를 뽑아봤을 때 다음과 같다. 2가지 경우 모두, test accuracy가 0.90으로 전보다 높아졌다. FNR은 첫번째 경우가 0.07로 전보다 낮아졌고, 두번째 경우는 전과 동일하다.

Sensors	Accuracy	FNR
(44, 42, 12, 17, 34)	0.904762	0.074074
(10, 11, 36, 44, 45, 42, 12, 17, 34, 33)	0.904762	0.111111

```
6900 9000 28500 7500 9900
11100 23400 8100 6900 4200 9000 28500 7500 9900 15900
```

분석 결과와 가격을 고려하여, 최적의 LR 모델 3가지를 선정해보았다. (하이퍼파라미터 튜닝은 진행하지 않았으나, 로지스틱 회귀에서 하이퍼파라미터 때문에 성능이 엄청 크게 변동하지는 않을 것이므로, 기본 모델을 기준으로 하였다. 또한, 성능이 튜닝 시 상승할 가능성을 고려하여 가격에 더 초점을 맞춰 선정하였다. 토요일 회의에서 비용에 대한 어느 정도 가닥이 잡힌 후에, 선택된 모델에 대해서 튜닝을 진행하려고 한다.)

1. 정확도: 0.905 / FNR: 0.074 / 비용: 제일 비싼 센서 1개, 평균 미만 가격 4개
센서: (12, 17, 34, 42, 44) / 가격: (28500, 7500, 9900, 9000, 6900)

2. 정확도: 0.881 / FNR: 0.111 / 비용: 평균 가격 1개, 평균 미만 가격 3개
센서: (10, 26, 36, 45) / 가격: (11100, 3000, 8100, 4200)

3. 정확도: 0.881 / FNR: 0.111 / 비용: 평균 가격 1개, 평균 미만 가격 3개
센서: (10, 36, 42, 45) / 가격: (11100, 7500, 9900, 7800)

```
price.describe()

count       60.000000
mean     11640.000000
std       5309.064842
min       2700.000000
25%       8100.000000
50%      11100.000000
75%      14400.000000
max      28500.000000
Name: 가격, dtype: float64
```

*제일 비싼 센서 12의 정확도가 가장 높음(0.762). 비싸지만 가장 높은 정확도를 가진 센서를 사용해서 정확도를 안정적으로 높일 것인지, 또는 평균 가격 이하의 센서들을 사용해서 가격의 효율성을 높일 것인지 생각해봐야 함! (개인적인 생각으론, 저렴한 센서들만 사용했을 때도 성능이 매우 높게 나오고 있으므로, 굳이 젤 비싼 센서를 사용할 필요는 없을 것 같음!)

하이퍼파라미터 튜닝

	Sensors	Accuracy_train	Accuracy_test	FNR	FPR	AUC	Best_Params
0	[12, 17, 34, 42, 44]	0.7108	0.9048	0.0741	0.0741	0.8716	{'C': 2.6037523690193916, 'max_iter': 9899}
1	[10, 26, 36, 45]	0.7590	0.8333	0.1481	0.1154	0.8519	{'C': 15.030492863292057, 'max_iter': 10000}
2	[10, 36, 42, 45]	0.7590	0.8571	0.1111	0.1111	0.8543	{'C': 1.7810870787131148, 'max_iter': 3123}

오히려 성능이 낮아짐!

KNN 모델

센서 후보 선택 과정

1. 센서 1개 조합에서 정확도가 0.5 이하인 센서는 버린다.

- 정확도가 0.5라는 것은 0, 1 중 하나를 랜덤하게 선택할 확률과 같다.

놀랍게도, lr에서 뽑은 최적의 센서들은 여기에 없다.

버릴 센서	정확도
3	0.452381
18	0.476190
19	0.476190
21	0.5
22	0.5
23	0.5
29	0.5
30	0.5
31	0.476190
33	0.5
35	0.476190
39	0.5
41	0.380952
53	0.476190
54	0.380952
56	0.476190
57	0.452381
60	0.5

센서 후보 선택 과정

2. 센서 1개 조합에서 FNR이 0.5 이상인 센서는 버린다.

-FNR은 실제로 1인데 0으로 예측한 비율을 의미한다. 즉, 실제로 누수가 있는데 누수가 없다고 판단한 비율을 말한다.

=>

버릴 센서	fnr
14	0.555556
16	0.518519
28	0.555556
45	0.518519

lr에서 뽑은 최적의 센서인 45가 버려진다.

=> 버리지 않은 센서들에 대해서 모델 적합을 해보자!

센서 후보 선택 과정

3. 센서 2개 조합 중, 정확도가 0.5 이하, FNR이 0.5 이상인 조합은 버린다.

=> 버릴 센서: 1, 2, 4, 5, 6, 7, 8, 9, 15, 20, 24, 25, 26, 27, 32, 34, 38, 40, 42, 43, 44, 47, 50, 51, 52, 55, 58, 59

모든 조합에 대해 모델을 돌려봤을 때, test accuracy가 0.881(로지스틱 회귀에서의 정확도) 이상인 경우는 다음과 같았다.

Sensors	Accuracy	FNR	K
(10, 11, 12, 37, 49)	0.928571	0.074074	{'n_neighbors': 3}
(10, 13, 17, 37, 48)	0.904762	0.111111	{'n_neighbors': 5}
(10, 11, 12, 37, 48, 49)	0.904762	0.074074	{'n_neighbors': 3}
(10, 13, 17, 37, 48, 49)	0.904762	0.111111	{'n_neighbors': 5}

```
11100 23400 28500 2700 11100
11100 10500 7500 2700 21900
11100 23400 28500 2700 21900 11100
11100 10500 7500 2700 21900 11100
```

LR에서의 최적의 모델 3가지에 비해 정확도가 높고, FNR도 낮다. 하지만 모든 조합에서 20000원 이상의 비싼 센서가 한 개 이상 포함되어 있고, 이를 배제했을 때에도 가격대가 전체적으로 LR에 비해서 높은 편이다.

분석 결과와 가격을 고려하여, 최적의 KNN 모델 2가지를 선정해보았다. (하이퍼파라미터 튜닝은 n_neighbors에 대해서만 했다. 또한, 성능이 튜닝 시 상승할 가능성을 고려하여 가격에 더 초점을 맞춰 선정하였다. 토요일 회의에서 비용에 대한 어느 정도 가닥이 잡힌 후에, 선택된 모델에 대해서 튜닝을 진행하려고 한다.)

1. 정확도: 0.905 / FNR: 0.111 / 비용: 비싼 센서 1개, 평균 이하 가격 3개, 가장 저렴한 센서 1개)
센서: (10, 13, 17, 37, 48) / 가격: (11100, 10500, 7500, 2700, 21900)

2. 정확도: 0.905 / FNR: 0.111 / 비용: 비싼 센서 1개, 평균 이하 가격 4개, 가장 저렴한 센서 1개)
센서: (10, 13, 17, 37, 48, 49) / 가격: (11100, 10500, 7500, 2700, 21900, 11100)

*37번 센서가 가격이 제일 싼데도 불구하고 정확도가 0.67로 매우 높게 나왔다.

```
price.describe()
```

```
count       60.000000
mean     11640.000000
std       5309.064842
min       2700.000000
25%       8100.000000
50%      11100.000000
75%      14400.000000
max      28500.000000
Name: 가격, dtype: float64
```