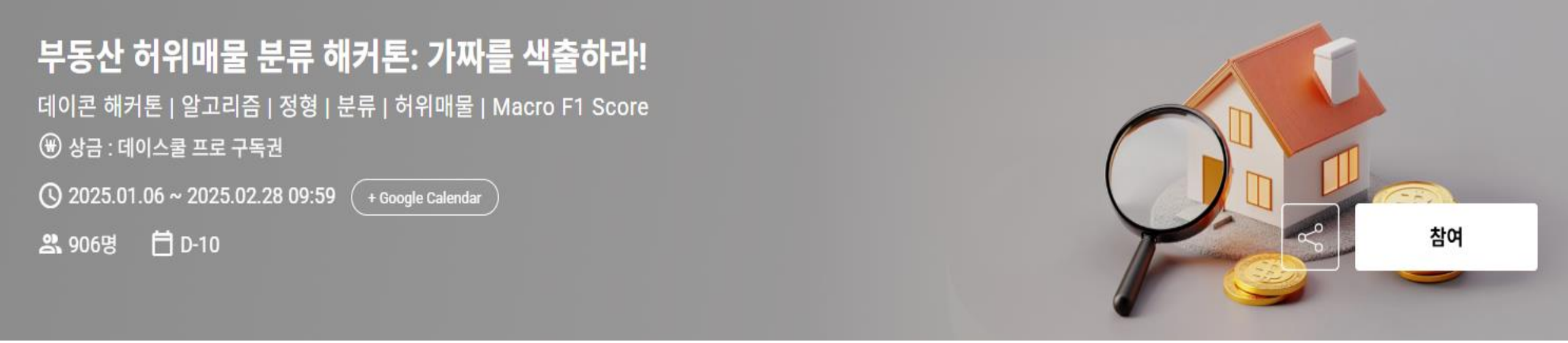


데이콘 – 부동산 허위매물 분류 해커톤

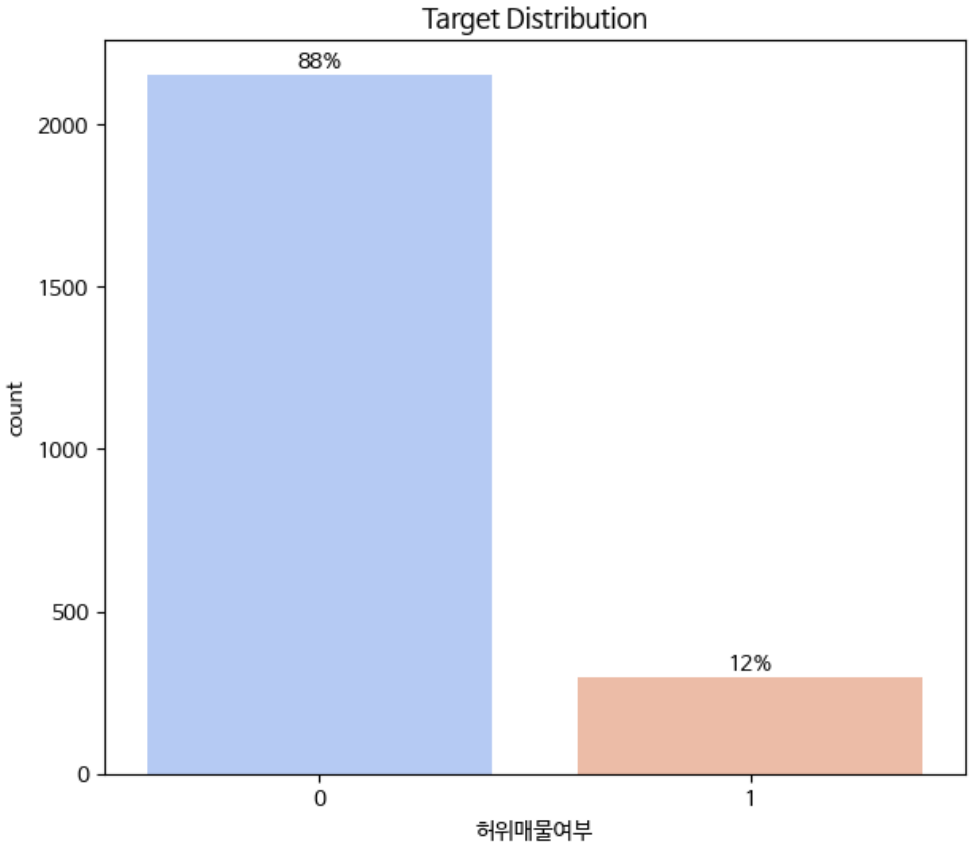
대회 목적 : 허위매물 detecting
허위 매물 여부로 되어 있고 허위 매물을 찾는것이 목적
평가 방식 : f1-score

Dataset (2452 x 17) 16개의 feature와 [0,1]의 target

	ID	매물확인방식	보증금	월세	전용면적	해당층	총층	방향	방수	욕실수	주차가능여부	총주차대수	관리비	중개사무소	제공플랫폼	게재일	허위매물여부
0	TRAIN_0000	현장확인	402500000.0	470000	NaN	NaN	15.0	서향	1.0	1.0	가능	40.0	96	t93Nt6I2I0	B플랫폼	2024-10-09	0
1	TRAIN_0001	현장확인	170500000.0	200000	NaN	3.0	4.0	남동향	2.0	1.0	불가능	NaN	0	q39iV5J4E6	D플랫폼	2024-12-26	0
2	TRAIN_0002	전화확인	114000000.0	380000	NaN	2.0	3.0	동향	1.0	1.0	불가능	NaN	0	b03oE4G3F6	A플랫폼	2024-11-28	0
3	TRAIN_0003	현장확인	163500000.0	30000	36.30	3.0	9.0	남동향	2.0	1.0	가능	13.0	10	G52Iz8V2B9	A플랫폼	2024-11-26	0
4	TRAIN_0004	현장확인	346000000.0	530000	NaN	3.0	3.0	동향	2.0	1.0	불가능	NaN	0	N45gM0M7R0	B플랫폼	2024-06-25	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2447	TRAIN_2447	서류확인	159000000.0	550000	48.95	3.0	3.0	남향	2.0	1.0	불가능	NaN	0	d22DX4Y4P8	B플랫폼	2024-11-16	0
2448	TRAIN_2448	서류확인	158500000.0	750000	NaN	2.0	4.0	남향	1.0	1.0	불가능	NaN	2	g99sy3I3R8	A플랫폼	2024-10-06	0
2449	TRAIN_2449	전화확인	329000000.0	610000	17.50	8.0	20.0	남서향	1.0	1.0	가능	29.0	10	G52Iz8V2B9	B플랫폼	2024-05-15	0
2450	TRAIN_2450	현장확인	31000000.0	400000	22.87	8.0	9.0	남동향	2.0	1.0	가능	NaN	8	m69GM9O9B3	B플랫폼	2024-08-06	0
2451	TRAIN_2451	전화확인	126000000.0	340000	29.89	4.0	6.0	북향	2.0	1.0	불가능	8.0	7	w94Qb4G0K5	B플랫폼	2024-03-23	0

2452 rows × 17 columns

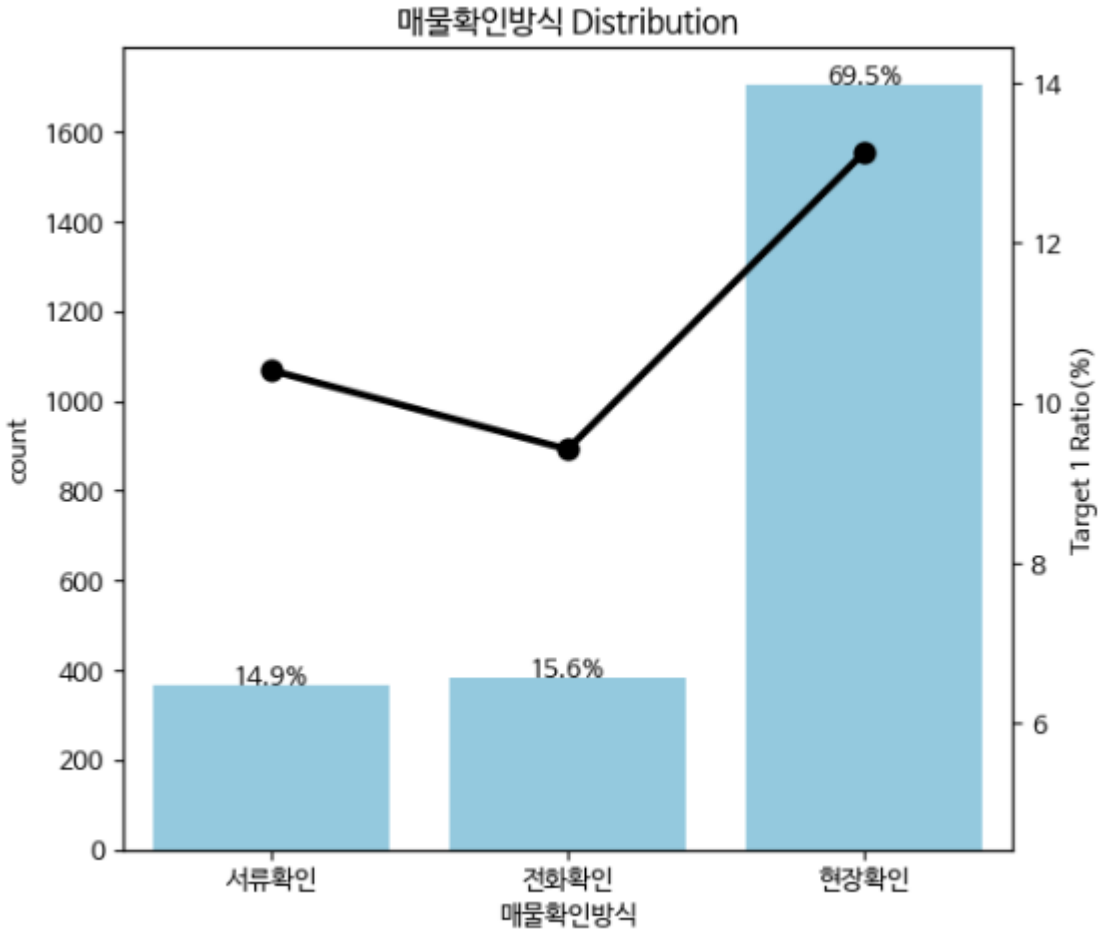
Target [허위매물여부]



Target 값이 88 : 12 로 데이터 불균형 상태
이대로 학습하면 0값을 많이 출력하는 문제가 있을 수 있음

대안 : 데이터 증강(SMOTE), 모델 가중치 조정

categorical [**매물 확인 방식** , 방향, 주차가능 여부, 중개사무소 , 제공플랫폼]



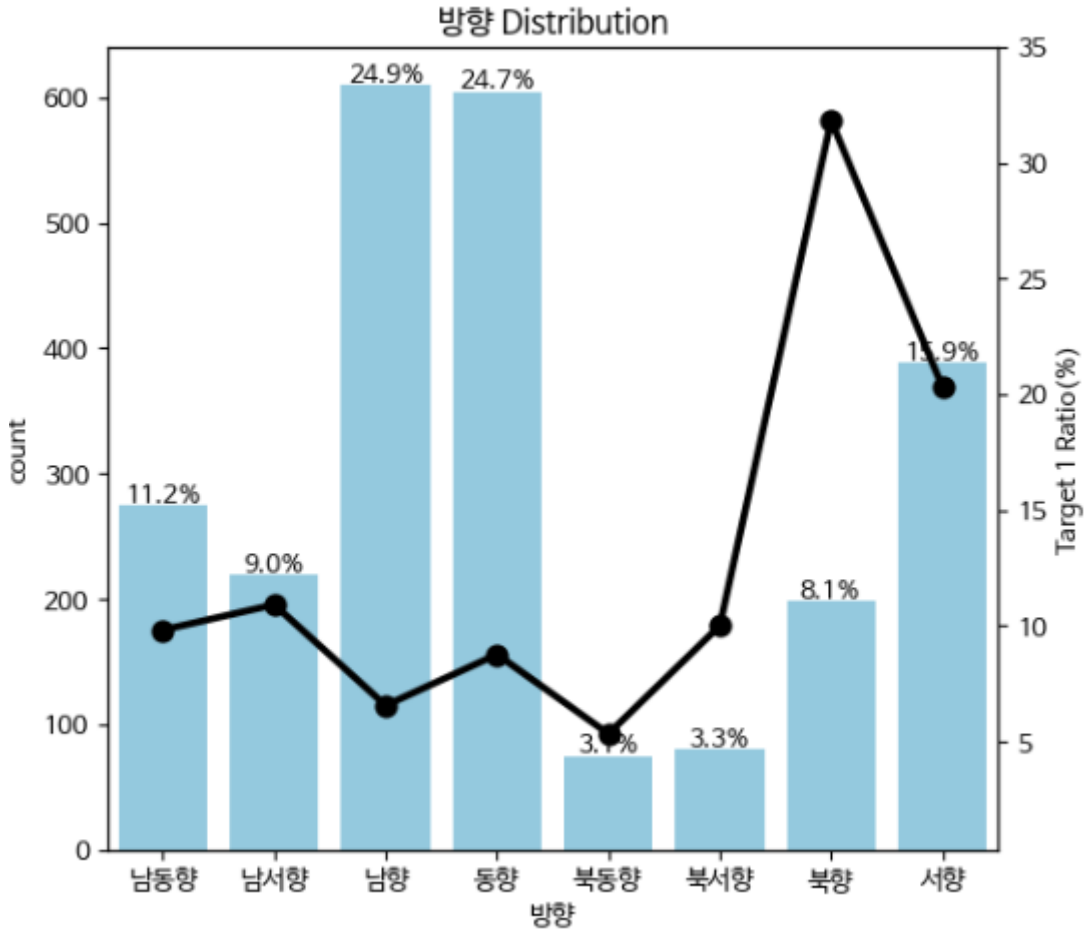
특이사항: 결측치 없음

카운트 플롯 (선) 의 경우 0과 1중 1의 비율
즉 서류확인일때 1인 비율 10% 0인 비율 90

실제 target의 경우 1인 비율 10% 0인 비율 90프로 임

따라서 매물 확인 방식의 경우
서류확인 11% 전화확인 9% 현장 13%로 크게 벗어나지 않음을 확인 할 수 있음

categorical [매물 확인 방식 , **방향**, 주차가능 여부, 중개사무소 , 제공플랫폼]

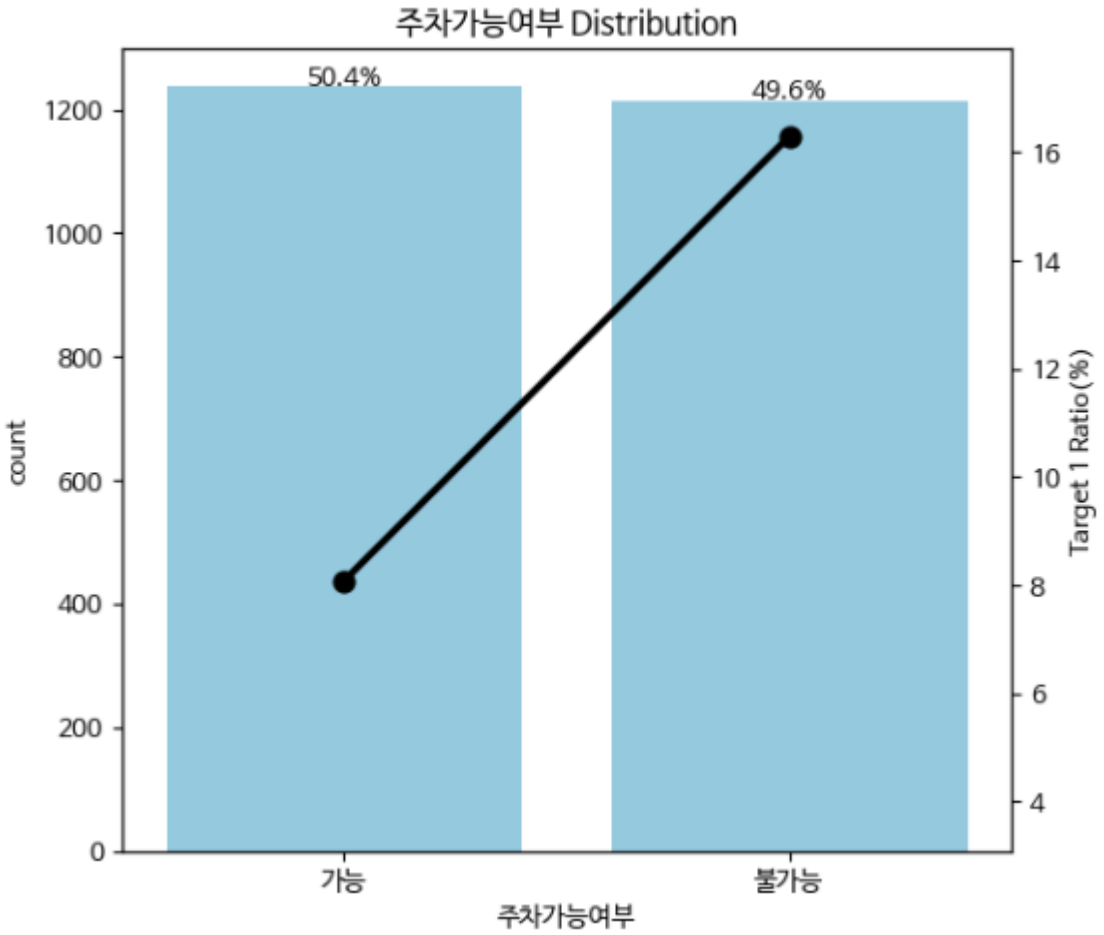


특이사항: 결측치 없음

남향 동향이 24.7프로로 가장 많음에도 불고 하고 실제 0인 비율이 적음

북향과 서향의 경우 33%와 20%로 기존의 9:1 비율에서 많이 벗어나 있음을 확인 할 수 있음

categorical [매물 확인 방식 , 방향, **주차가능** 여부, 중개사무소 , 제공플랫폼]

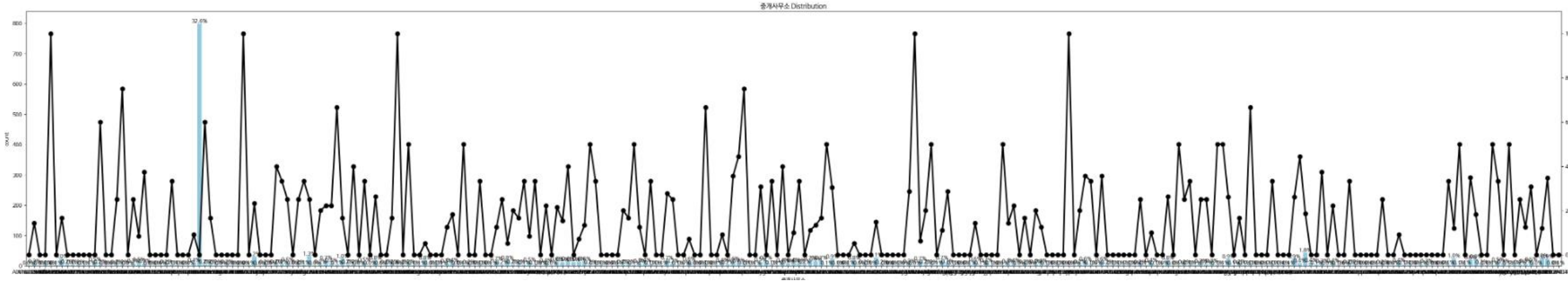


특이사항: 결측치 없음

주차 가능 여부의 경우 이진데이터로 절반씩 가지고 있음

주차가 불가능일때 16프로로 가능보다 약 8% 허위여부일 가능성이 높았음

categorical [매물 확인 방식 , 방향, 주차가능 여부, **중개사무소** , 제공플랫폼]



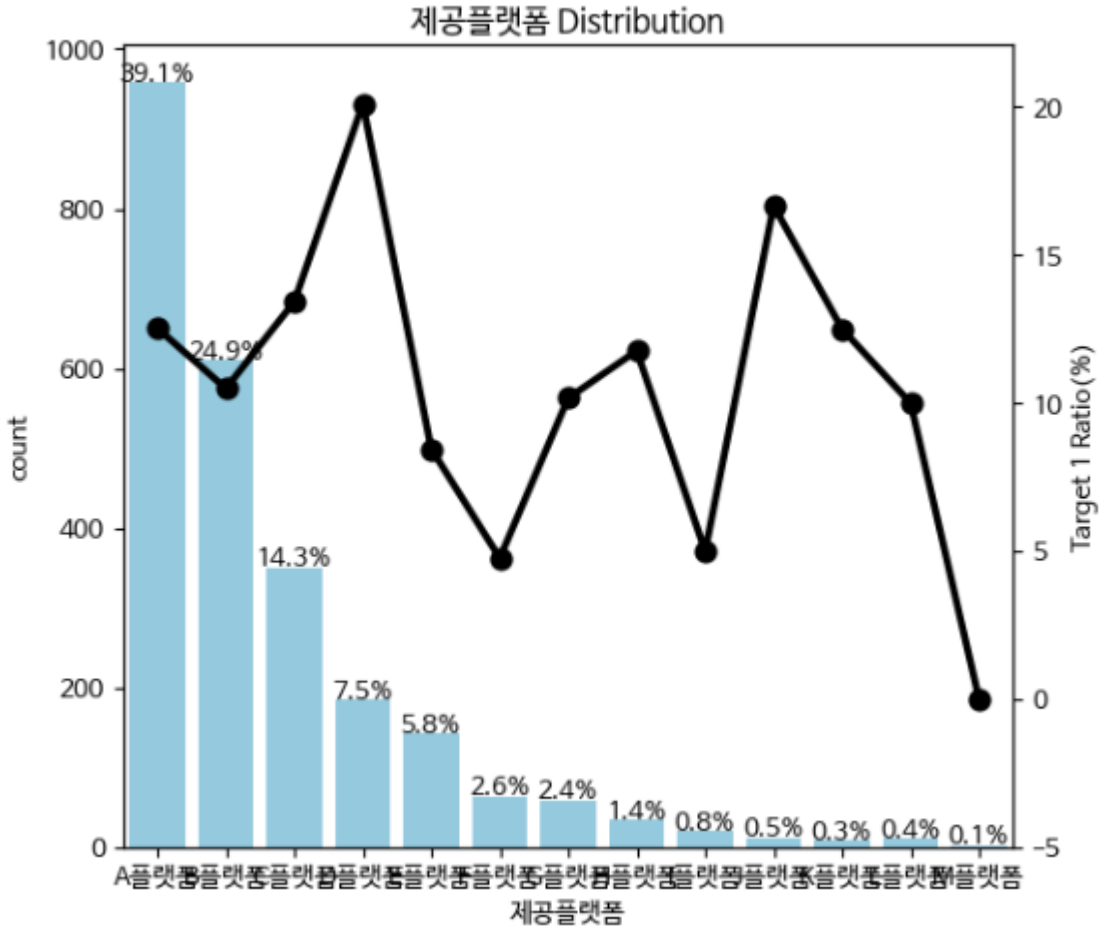
특이사항: 결측치 없음

중개사무소의 경우 279개가 존재

한곳 빼고 골고루 양분하고 그렇기 때문에 적은 개수의 매물이 허위매물이여서 그 곳은 허위매물이다 라고 판단하기 어렵다고 생각

그렇기 때문에 이 열은 빼는게 나을 수도 있다고 판단 됌 -> 이열은 일반화 성능을 떨어뜨린다 라고 판단 중

categorical [매물 확인 방식 , 방향, 주차가능 여부, 중개사무소 , **제공플랫폼**]



특이사항: 결측치 없음

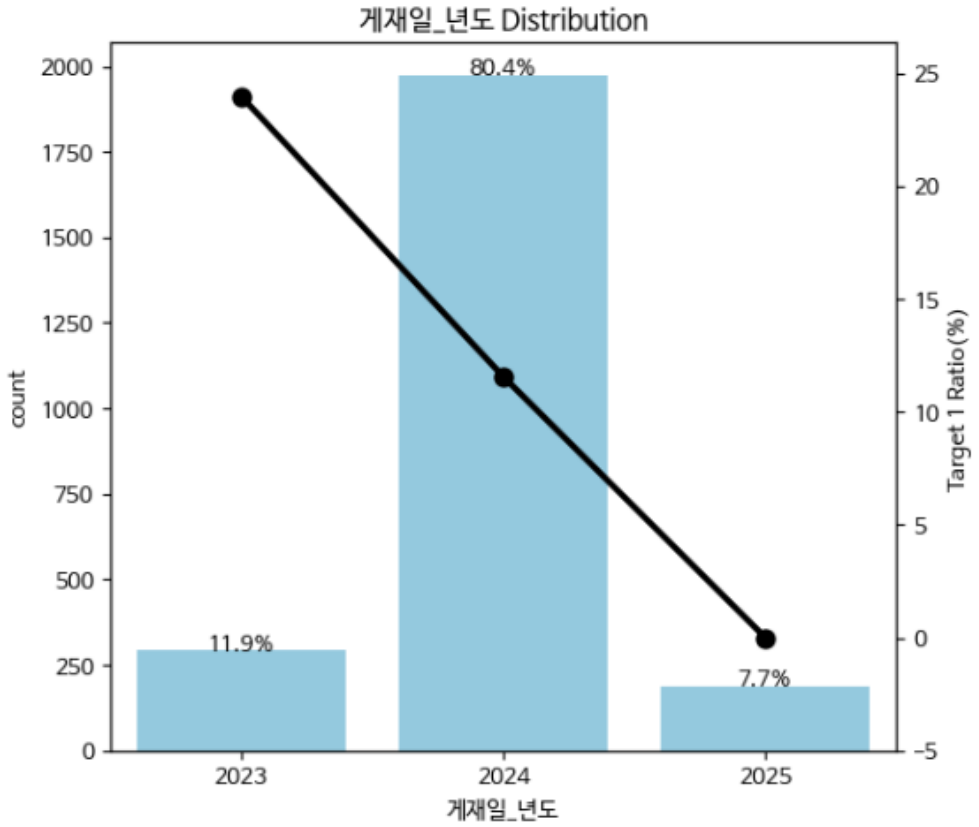
한 플랫폼에서 약 40프로의 데이터 불균형을 가지고 있음

A플랫폼의 경우 10%로 허위 매물이 있는 반면 D플랫폼의 경우 20%로 2배정도의 허위매물이 있음 확인 할 수 있음

Feature [**게재일**] 실제 게재일이 굉장히 주요 변수로 나와서 좀 쪼개서 볼 필요가 있음

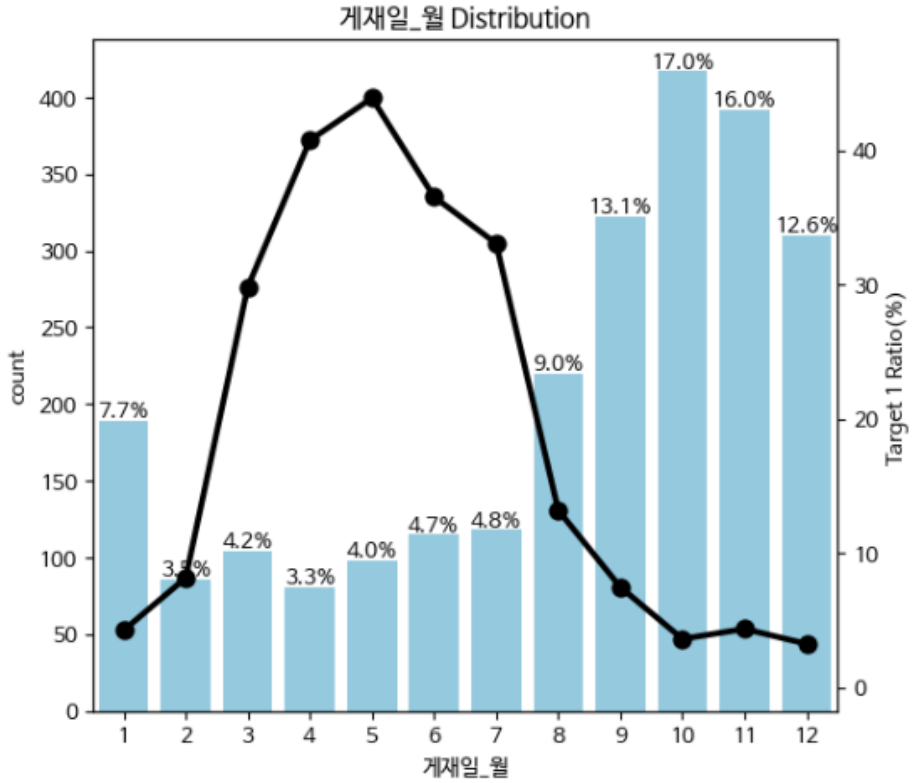
2025년도는 아예 사기가 없다고 모델이 판단할 가능성이 큼

2023의 경우 허위매물의 비율이 높음



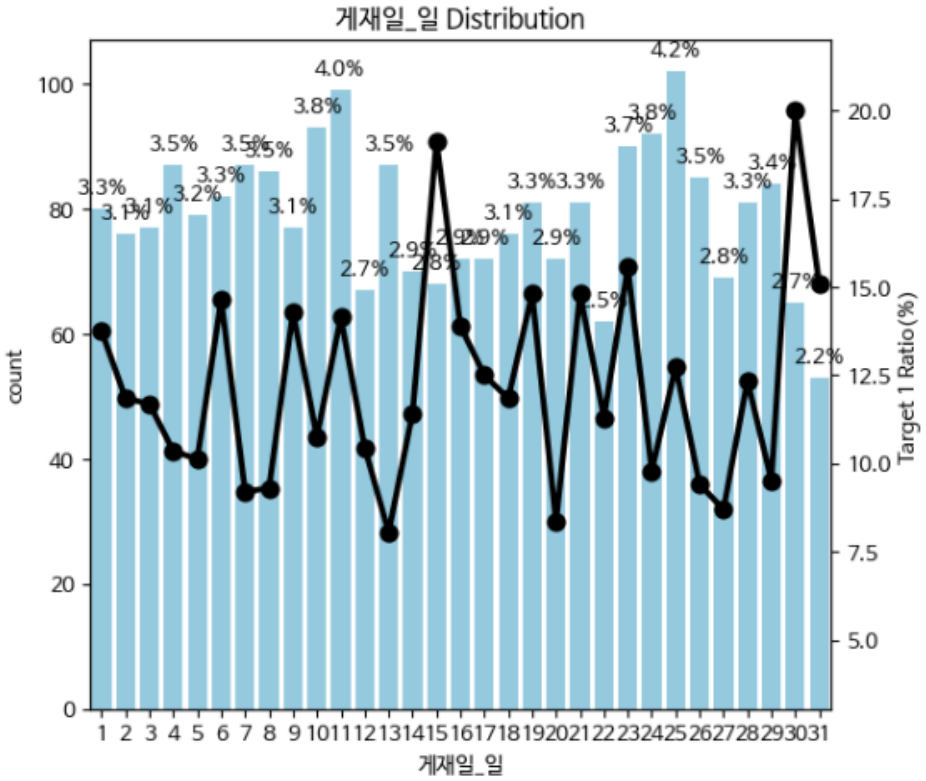
3~7월이 허위 매물일 때가 가장 많았음

실제 매물이 없을 시기에 허위매물일 가능성이 높다 판단 가능

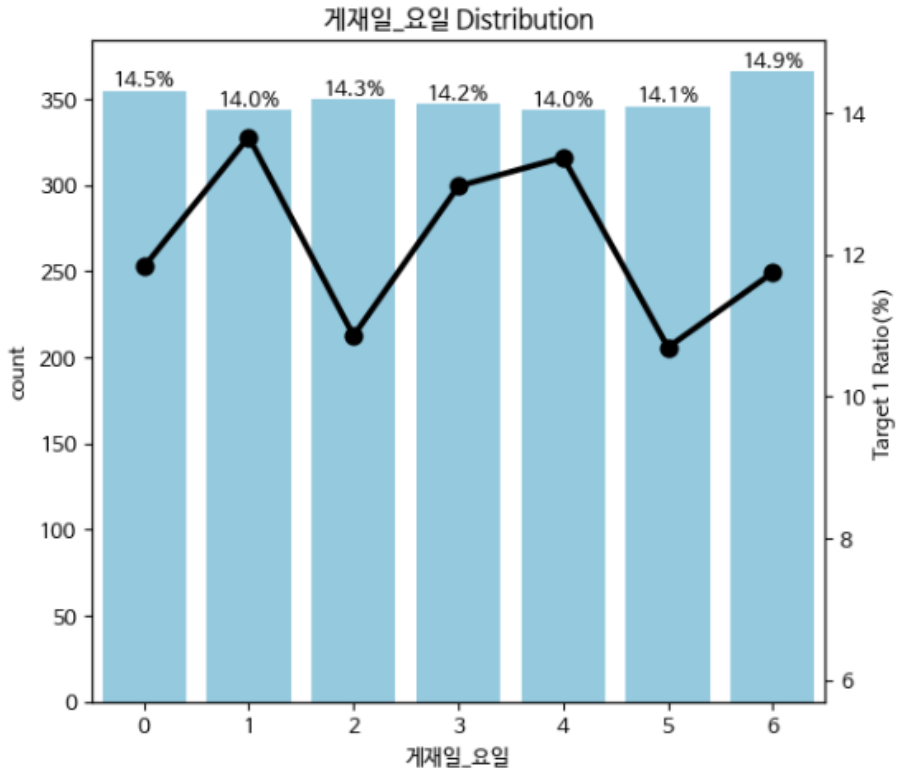


Feature [**게재일**] 실제 날짜가 굉장히 주요 변수로 나와서 좀 쪼개서 볼 필요가 있음

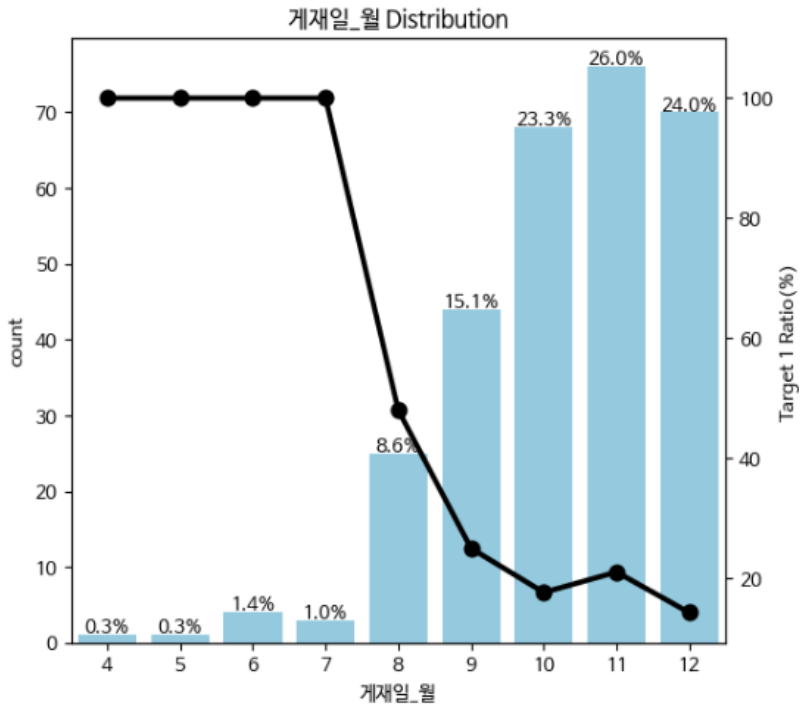
중순과 말에 비율이 높아짐
-> 말은 다음달과 관련이 있나?



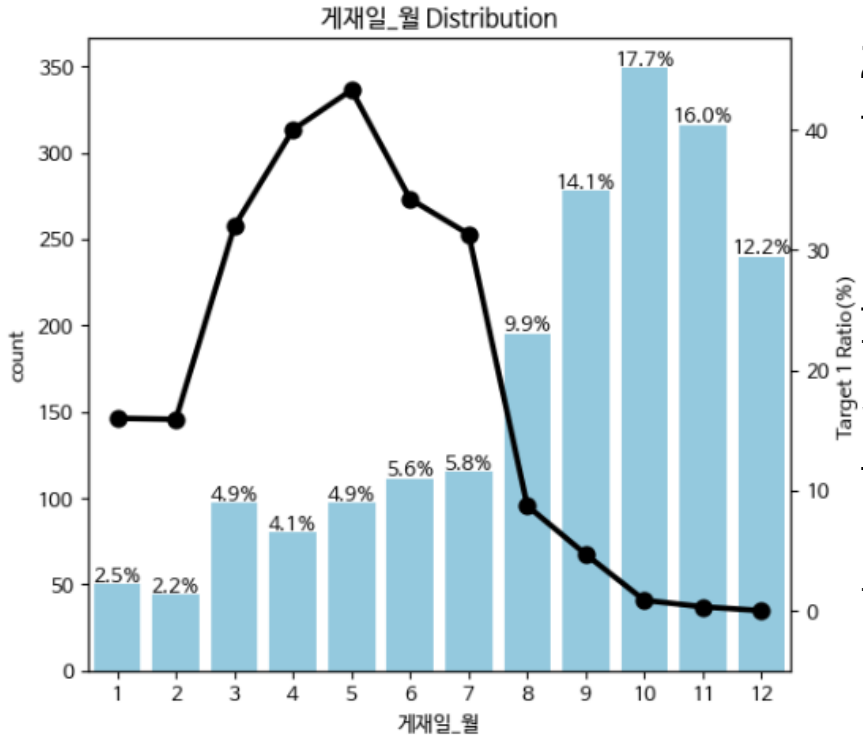
요일은 크게 벗어난 비율이 보이지 않음



Feature [**게재일**]



2023



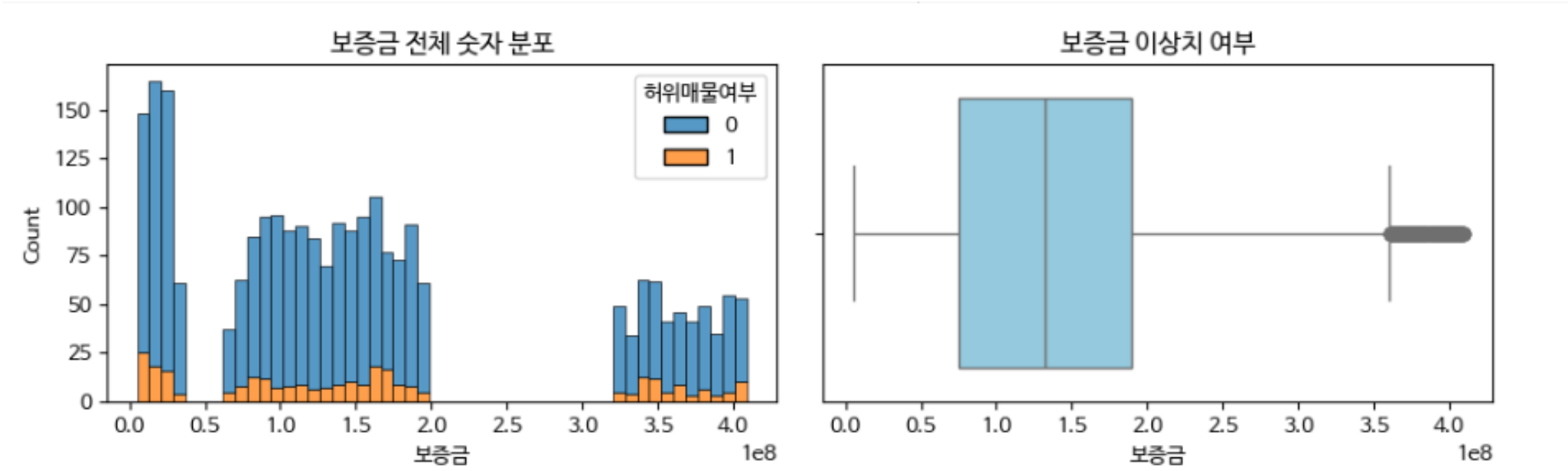
2024

2025의 경우 1,2,3월데이터만 있음
즉 1,2,3의 경우 2025년에 허위매물이 없음

그렇다면 4~7월에 가중을 해서 봐야하지 않을까? 왜 월로 했을때 모델 성능이 안좋았을까?

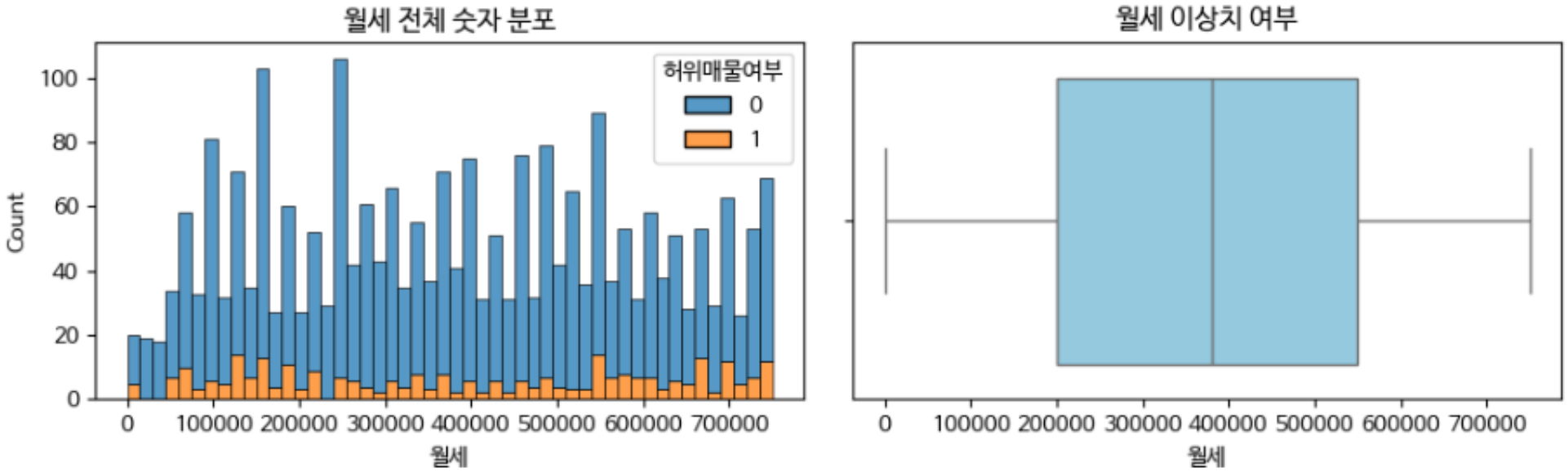
-> 라벨인코딩을 해서?

Numerical [**보증금** , 월세 , 전용면적, 해당층 , 총층 , 총차대수, 관리비]



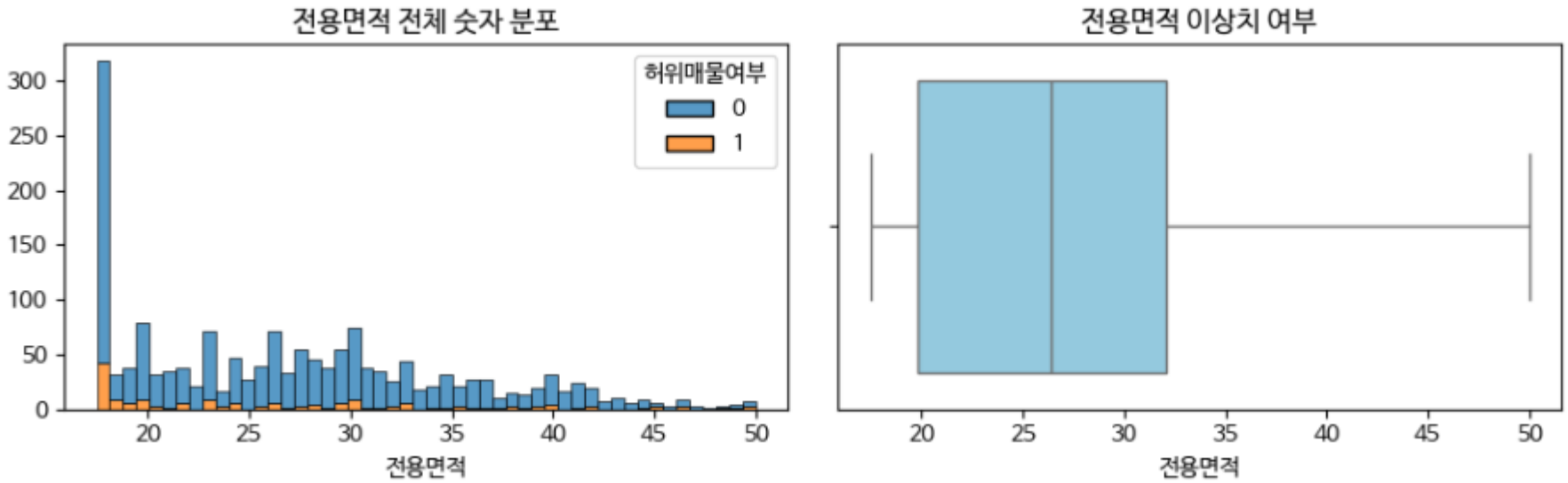
특이사항 : 결측치 없음
보증금이 높다고 무조건 허위 매물은 아님, 보증금 가격별로 허위 매물 존재
따라서 보증금이 높다고 무조건 이상치 처리 하기는 어렵다고 판단

Numerical [보증금 , **월세** , 전용면적, 해당층 , 총층 , 총차대수, 관리비]



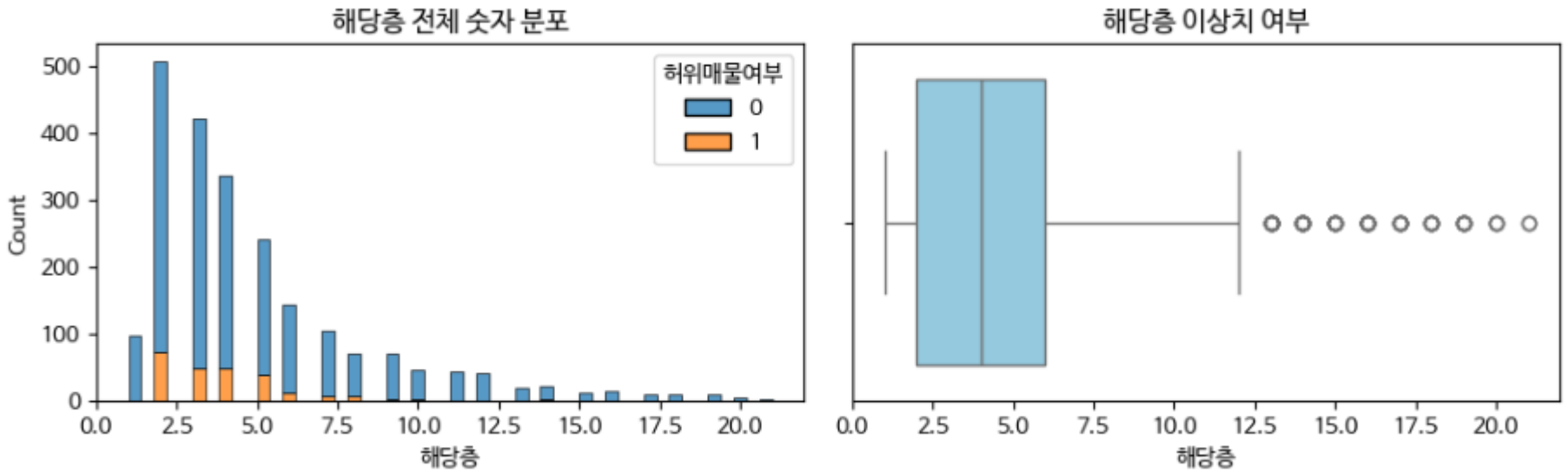
특이사항 : 결측치 없음
월세의 박스 플롯 또한 문제가 없는 플롯으로 보임

Numerical [보증금 , 월세 , **전용면적**, 해당층 , 총층 , 총차대수, 관리비]



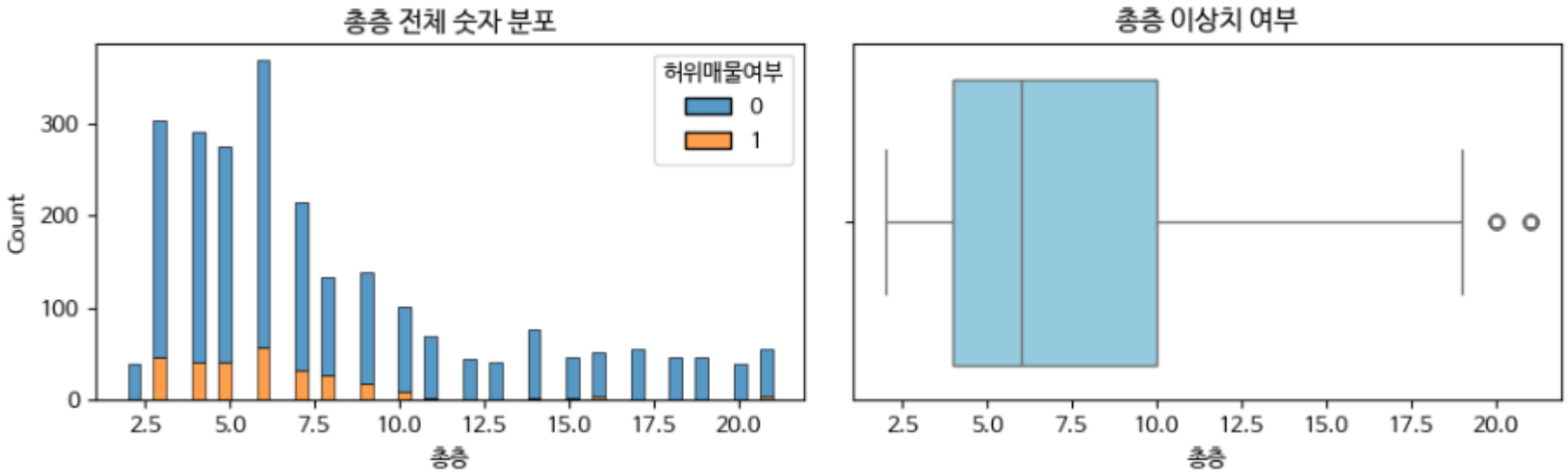
특이사항 : 결측치 있음 (797개)
20 아래일때 매물이 제일 많음 , 이상치가 존재하지 않음

Numerical [보증금 , 월세 , 전용면적, **해당층** , 총층 , 총차대수, 관리비]



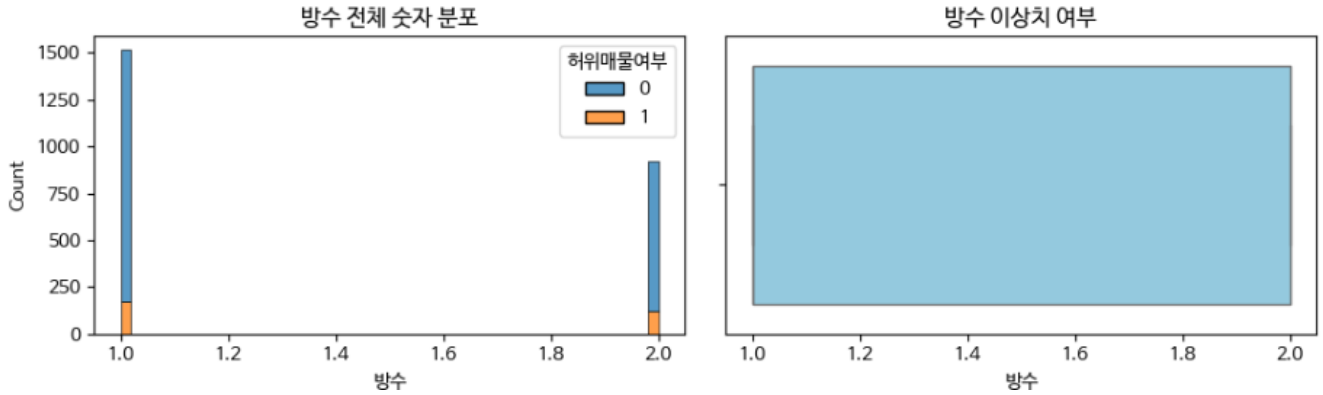
특이사항 : 결측치 있음(229개)
박스 플롯에는 이상치가 많이 나옴
하지만 방 면적이 작은 곳이 많다 -> 원룸일 가능성이 높다. 따라서 많은 매물들이
작은 층 수에 있을 가능성이 높다 라고 해석가능 또한 층수가 높을때 허위매물이 적어보임

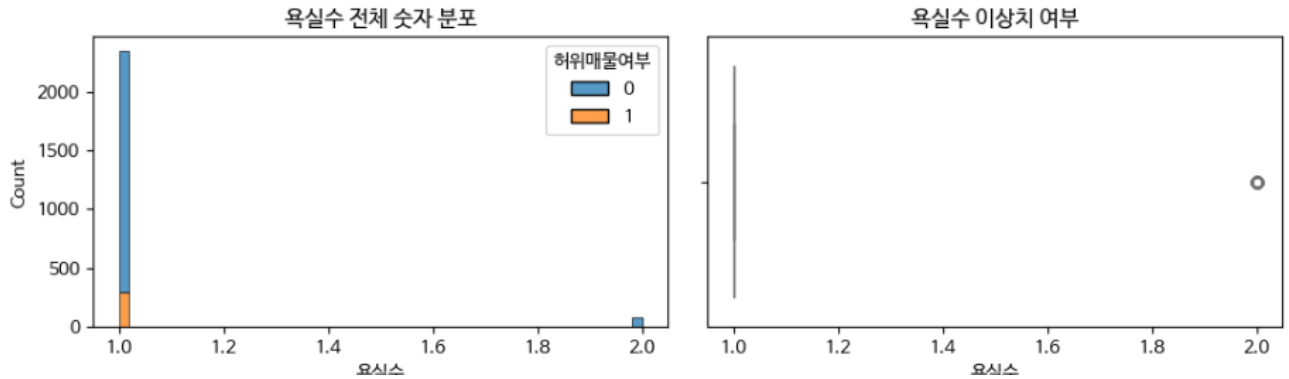
Numerical [보증금 , 월세 , 전용면적, 해당층 , **총층** , 총차대수, 관리비]

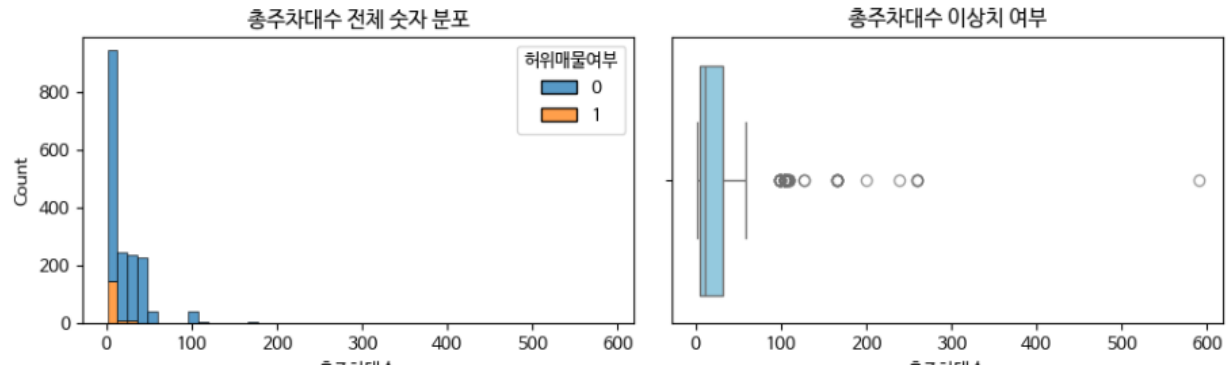


특이사항 : 결측치 있음(16개)
5~8층 사이가 매물이 제일 많음 -> 원룸 비율이 높다
또한 원룸일때 허위 매물일 가능성이 높다 해석가능

Numerical [보증금 , 월세 , 전용면적, 해당층 , 총층 , **방수, 욕실수, 총차대수**, 관리비]







특이사항 :방수 결측치(16개)

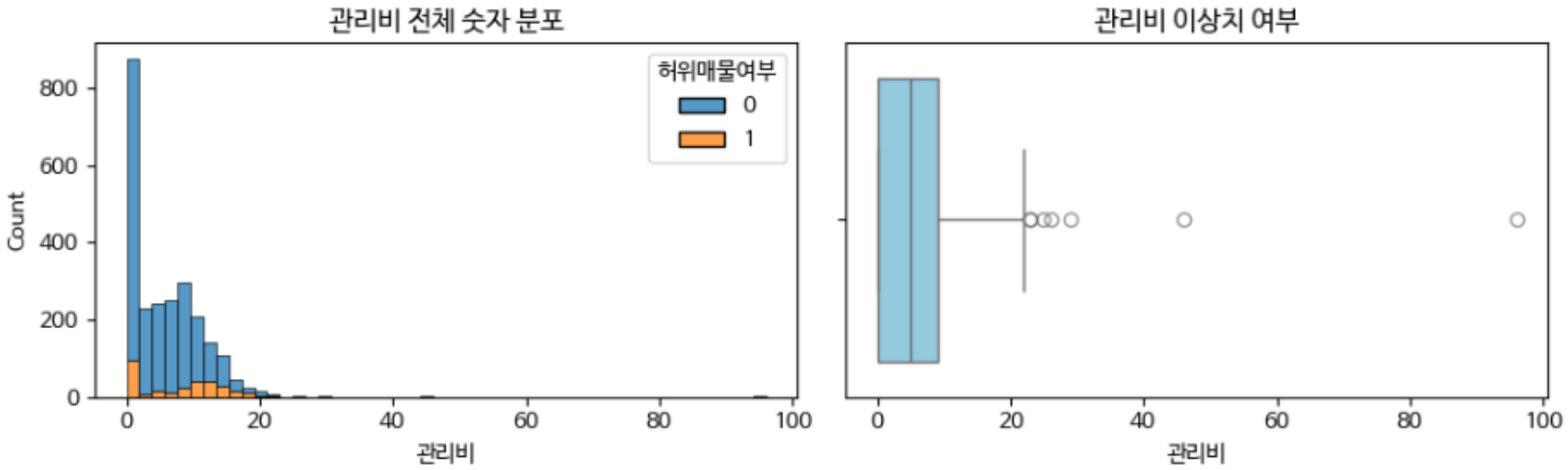
특이사항 :욕실수 결측치(18개)

특이사항 :주차대수 결측치(696개)

욕실수는 학습에 도움이 안될 가능성이 높음 -> 욕실수가 적음 근데 평수 큰 애들욕실은 2개보다 더 많지 않나?

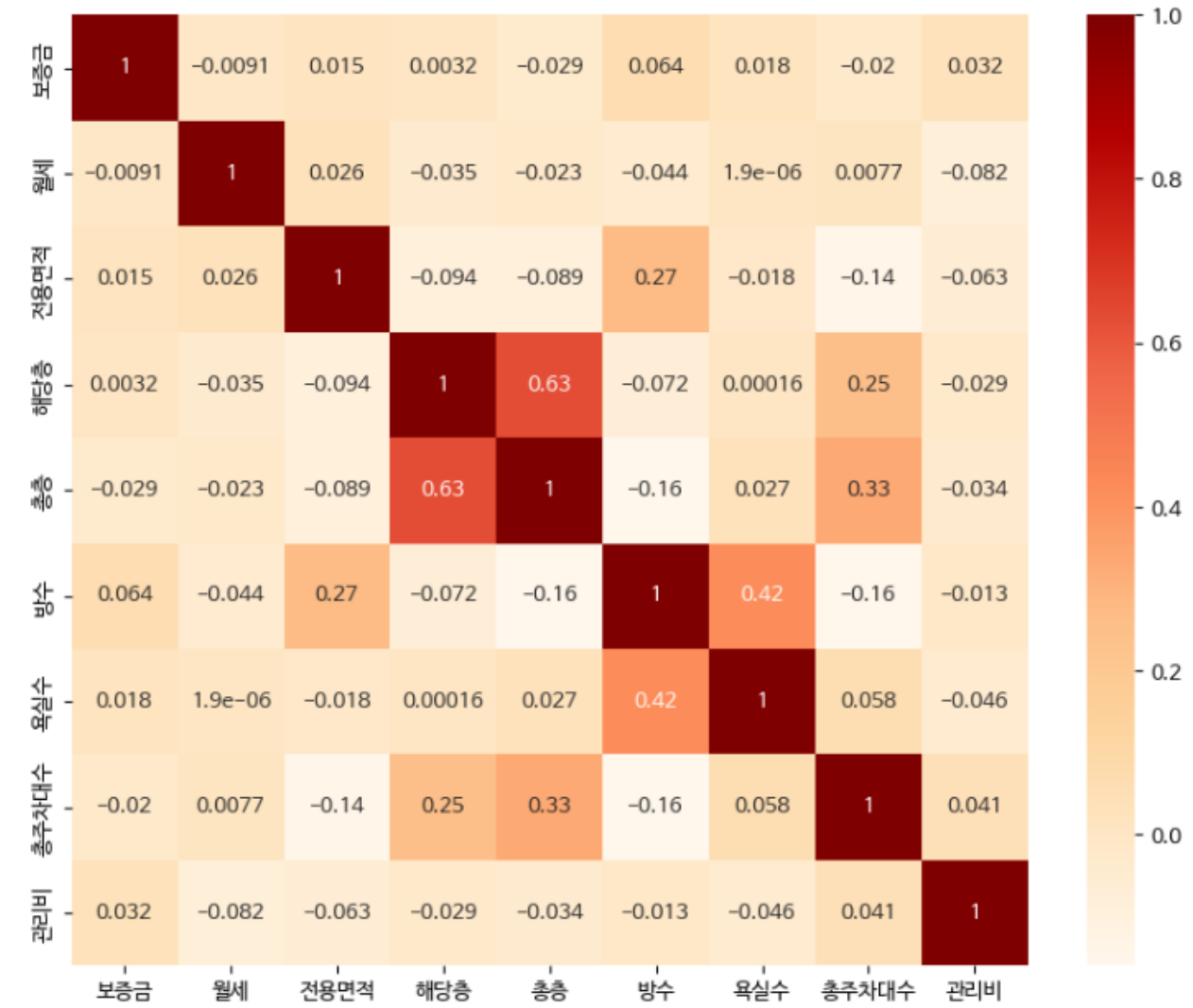
결측치 처리할때 중앙값으로 해야 한다는 판단 들게함 box플롯에서 봤을때
아니면 MICE? -> 평균은 안될 듯

Numerical [보증금 , 월세 , 전용면적, 해당층 , 총층 , 방수, 욕실수, 총차대수, **관리비**]



특이사항 : 결측치 없음
관리비 또한 높은 금액이 거의 존재 x
그리고 모든 숫자형 그래프에서 아웃라이어 하나가 크게 돋아짐 저거 하나 삭제하고 학습 하는게 도움될듯

상관관계 분석



해당층과 총층 서로 상관관계가 강함
둘중 하나만 사용 or 둘다 삭제 검증 해봐야할듯

결측데이터

전용면적(787개), 총주차대수(696개), 해당층(229개), 총층(16개), 방수(16개), 욕실수(16개)

모두 numerical 데이터임
어떤 방법으로 결측치를 선택할 것인지 판단

1. 중앙값 대치 -> 일단 평균 값은 하면 안됨 -> 정규분포가 아니기 때문에 이상치에 민감함

2. regression imputation -> 빠름 , 결측치 5%미만일때, 평균에 가까워질 수 있음

3. MICE -> 느림 , regression 의 보완(여러 번 반복해서 최적의 값 찾음), 데이터 간의 관계 찾을 수 있음

실험을 다 해봐야 하긴 하겠지만 MICE에 마음이 더감

Feature 엔지니어링을 하지 않은 상태에서 모델이 얼마나 더 좋게 나오는지 실험 해봐야 함

결측치 처리 실험

Validation set -> 5-cross fold 교차 검증으로 진행 rmse , mse로 확인

성능평가

실험 1 regression (Numerical)

실험 2 regression (Numerical + categorical)

실험 3 Mice (Numerical)

실험 4 Mice (Numerical + categorical)

실험 1 regression (Numerical)

🔹 전용면적 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 6.51, MAE: 5.16

🔹 층주차대수 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 20.18, MAE: 14.44

🔹 해당층 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 4.53, MAE: 3.05

🔹 욕실수 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 0.24, MAE: 0.08

🔹 총층 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 4.74, MAE: 3.37

🔹 방수 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 0.43, MAE: 0.28

실험 3 Mice (Numerical)

🔹 전용면적 MICE 기반 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 6.55, MAE: 5.11

🔹 층주차대수 MICE 기반 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 20.59, MAE: 15.03

🔹 해당층 MICE 기반 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 3.95, MAE: 2.83

🔹 욕실수 MICE 기반 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 0.24, MAE: 0.08

🔹 총층 MICE 기반 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 4.20, MAE: 3.05

🔹 방수 MICE 기반 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 0.41, MAE: 0.27

실험 2 regression (Numerical + cate)

🔹 전용면적 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 6.06, MAE: 4.96

🔹 층주차대수 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 14.92, MAE: 11.13

🔹 해당층 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 4.48, MAE: 3.05

🔹 욕실수 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 0.22, MAE: 0.08

🔹 총층 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 4.07, MAE: 2.70

🔹 방수 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 0.42, MAE: 0.26

실험 4 Mice (Numerical + cate)

🔹 전용면적 MICE 기반 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 6.25, MAE: 4.90

🔹 층주차대수 MICE 기반 결측치 대체 성능 비교
결측치 대체 후 RMSE: 14.74, MAE: 11.00

🔹 해당층 MICE 기반 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 3.96, MAE: 2.89

🔹 욕실수 MICE 기반 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 0.22, MAE: 0.08

🔹 총층 MICE 기반 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 3.83, MAE: 2.57

🔹 방수 MICE 기반 결측치 대체 성능 비교 🔹
결측치 대체 후 RMSE: 0.45, MAE: 0.29

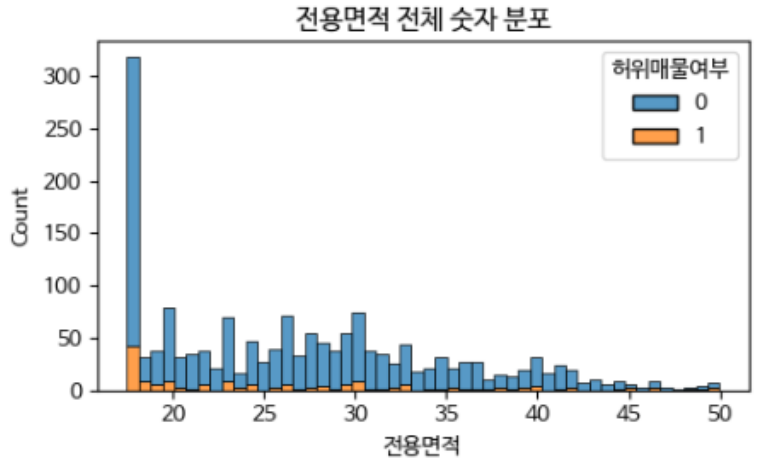
결측치 처리 실험

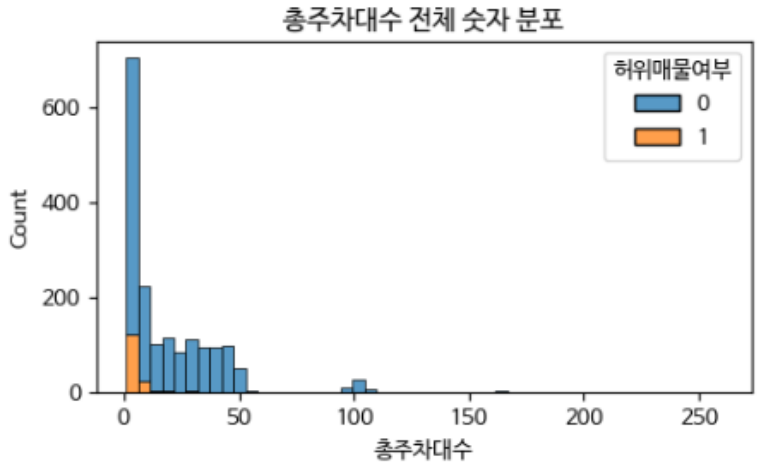
	Regression(num)		Regression(num+cat)		Mice(num)		Mice(num+cat)		Median	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
전용면적	6.51	5.16	**6.06**	4.96	6.55	5.11	6.25	**4.90**	6.16	5.05
주차대수	20.18	14.44	14.92	11.13	20.59	15.03	**14.74**	**11.00**	23.88	18.27
해당층	4.53	3.05	4.48	3.05	**3.95**	**2.83**	3.96	2.89	5.29	3.69
욕실수	0.24	0.08	**0.22**	0.08	0.24	0.08	**0.22**	0.08	0.24	**0.06**
총층	4.74	3.37	4.07	2.70	4.20	3.05	**3.83**	**2.57**	5.71	4.47
방수	0.43	0.28	0.42	**0.26**	**0.41**	0.27	0.45	0.29	0.56	0.31
평균	6.10	4.39	5.02	3.69	5.98	4.39	**4.90**	**3.62**	6.97	5.30

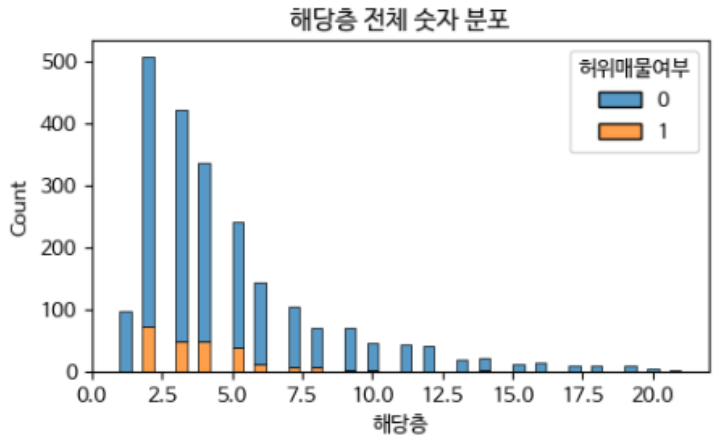
평균적인 성능상 MICE (num + cat) 성능이 제일 좋았으나

평균적인 성능상 MICE (num + cat) 성능이 제일 좋았으나,
모든 결측치에서 좋은 성능을 보인 것은 아니기 때문에 고려대상 또한 주차대수의 경우 값차이가 크기 때문에
Mice와 Regression(num+cat) 둘다 모델의 고려사항

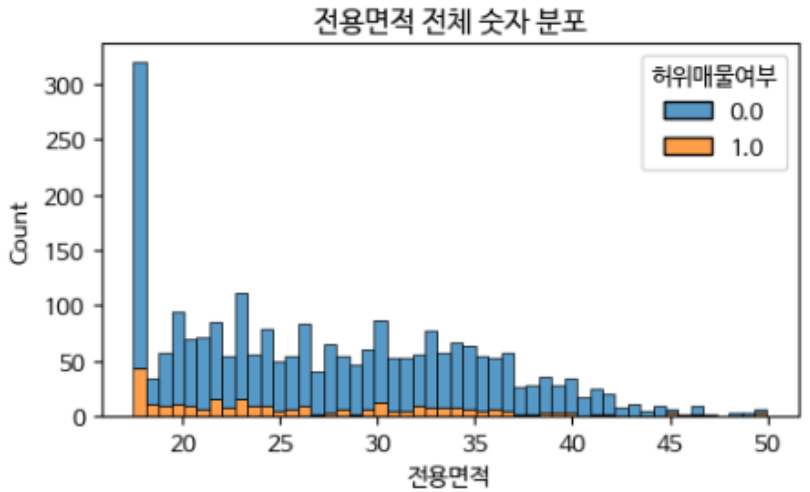
Before

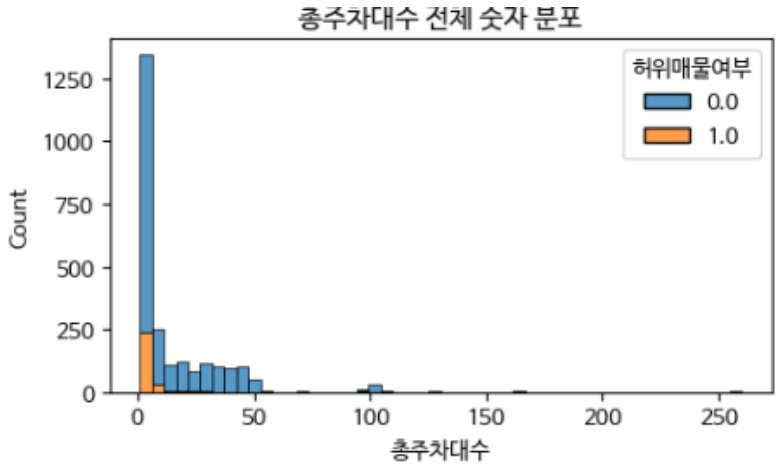


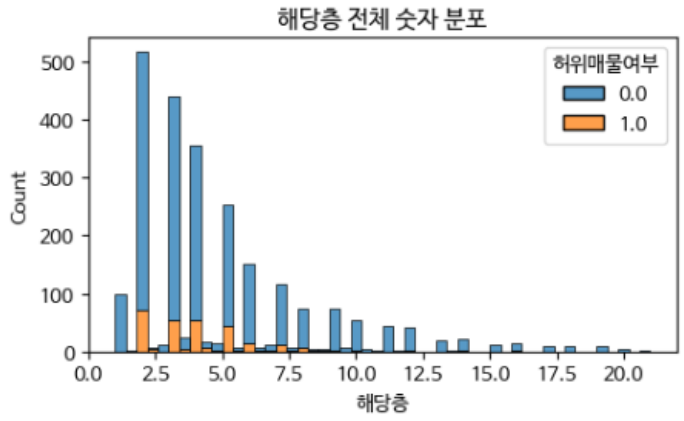




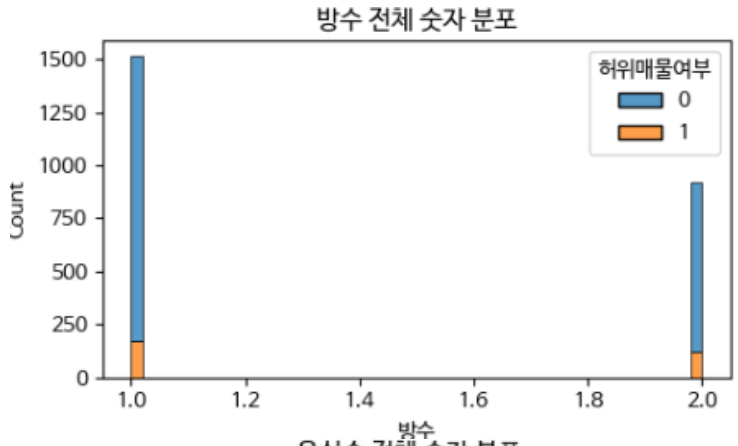
After

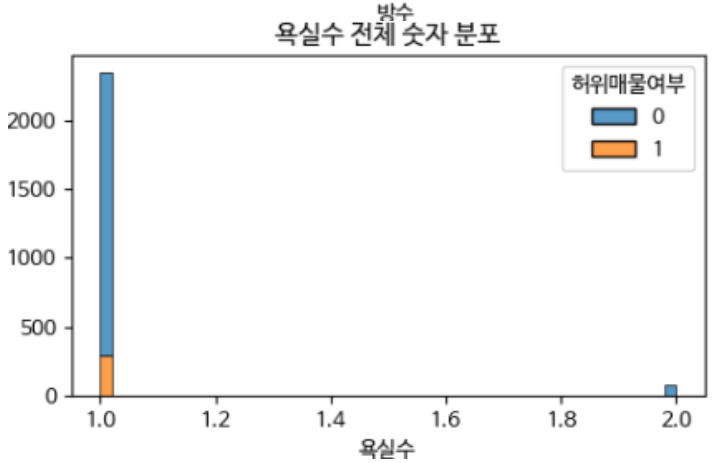


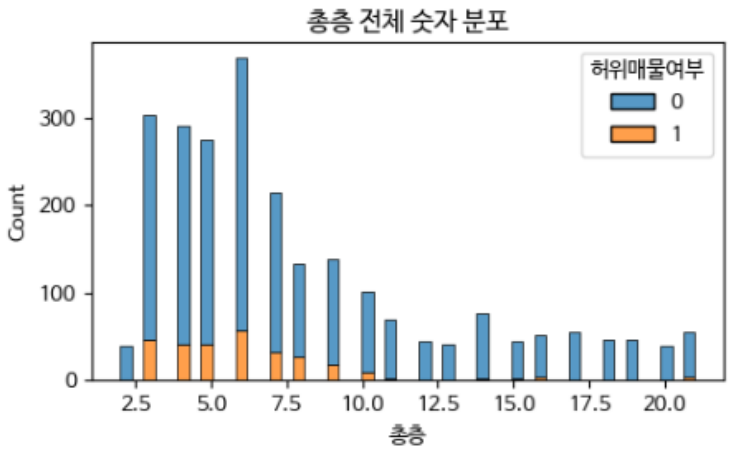




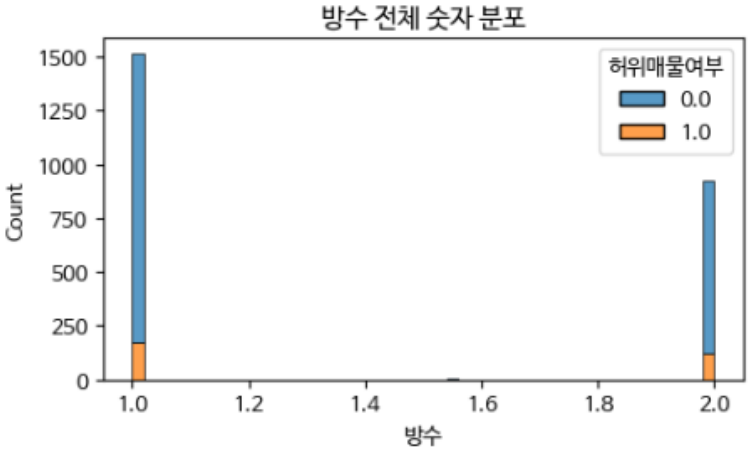
Before

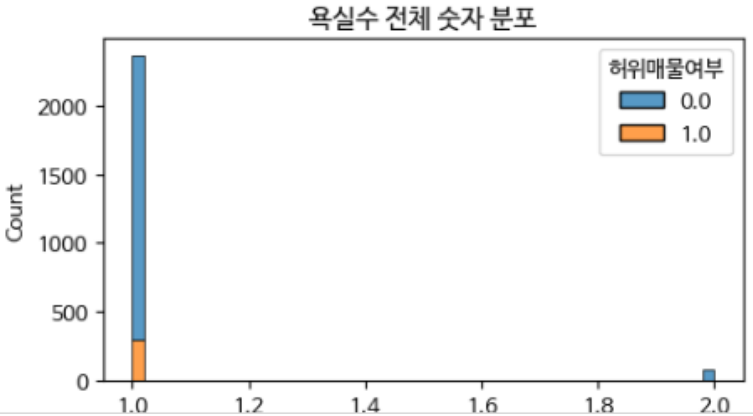


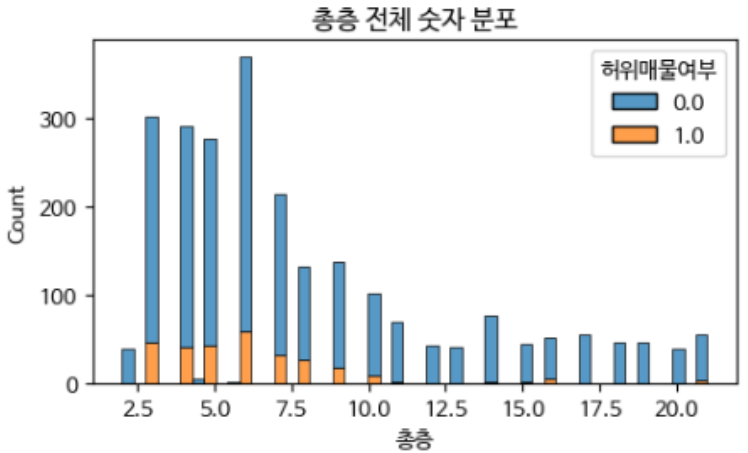




After







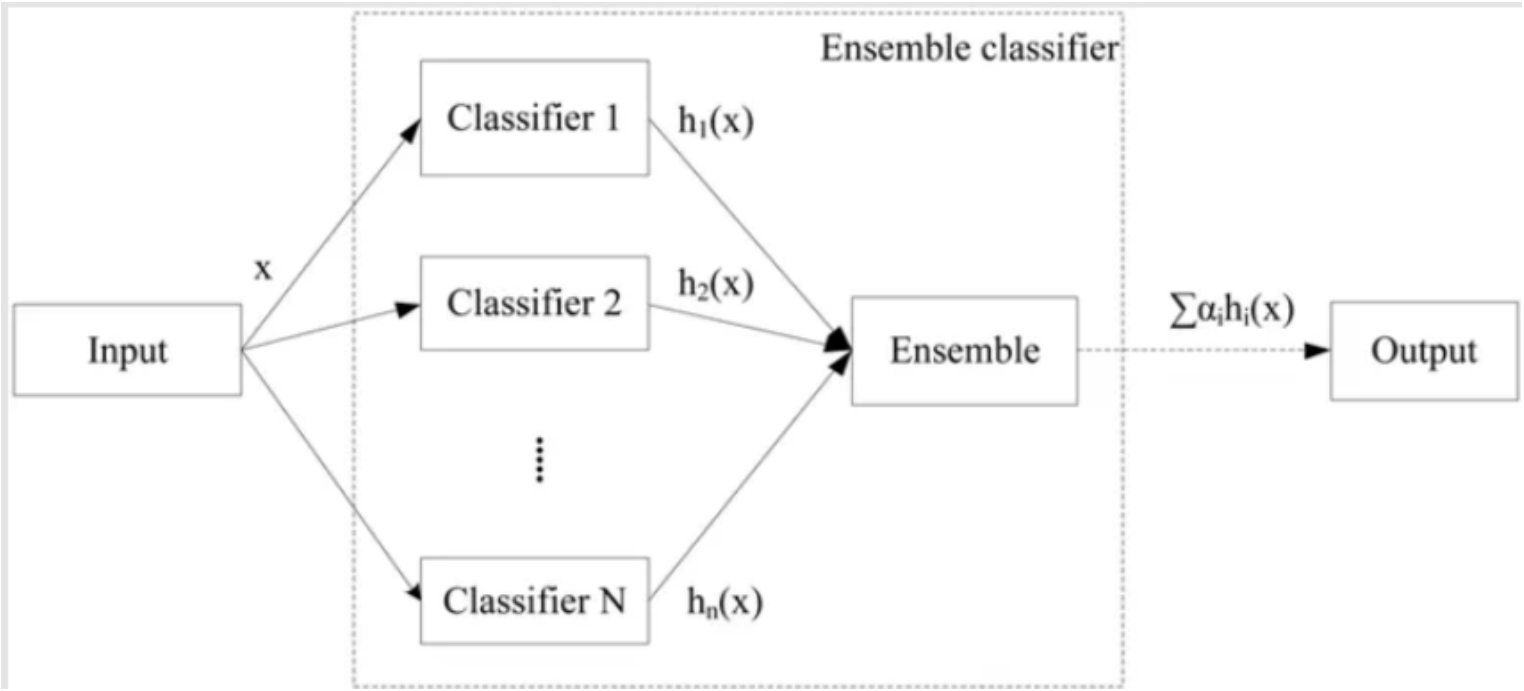
모델선택

랜덤포레스트 vs Xgboost

현재 데이터 셋이 너무 작아 딥러닝은 성능이 떨어질 가능성이 굉장히 큼
따라서 머신러닝 모델을 선택해서 확인하는 것이 좋음

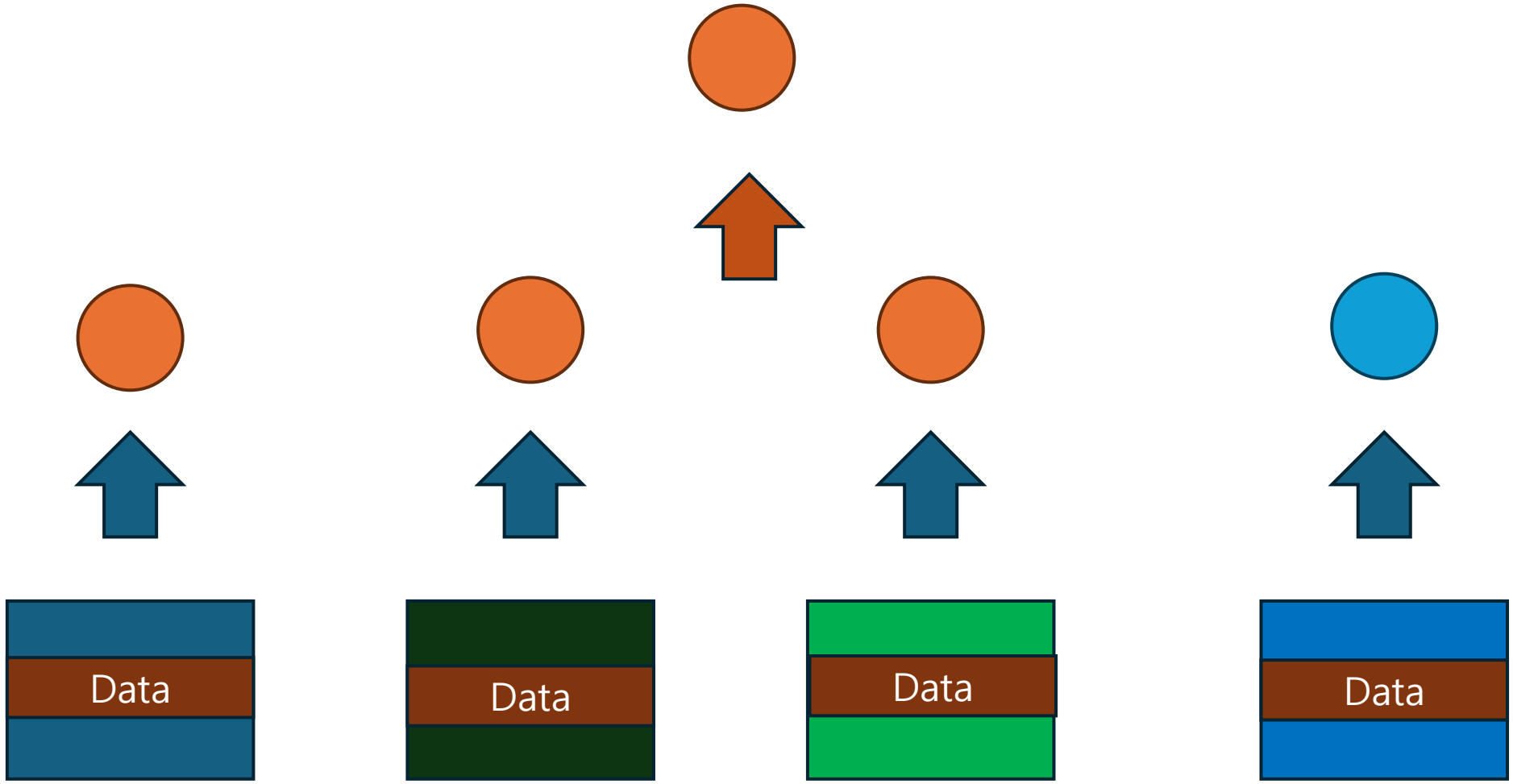
비교 항목	랜덤 포레스트 (Random Forest)	XGBoost
기반 기법	**배깅(Bagging)**	**부스팅(Boosting)**
트리 생성 방식	여러 개의 결정 트리를 독립적으로 학습	이전 트리의 오류를 보완하면서 학습
과적합 방지	비교적 강함 (다양한 트리를 결합)	매우 강함 (정규화 및 가중치 조절)
훈련 속도	상대적으로 빠름 (병렬 처리 가능)	느림 (트리를 순차적으로 학습)
예측 속도	빠름 (병렬 처리 가능)	상대적으로 느림
성능	중간 (XGBoost보다 낮을 수 있음)	**일반적으로 더 높은 성능**
하이퍼파라미터 튜닝	상대적으로 간단함	튜닝이 복잡하지만 최적화 시 강력한 성능
해석 가능성	상대적으로 쉬움 (특성 중요도 확인 가능)	어려움 (트리와 가중치가 많아 복잡)
메모리 사용량	많음 (여러 개의 트리를 저장해야 함)	상대적으로 적음 (트리를 효율적으로 활용)
결측값 처리	직접 처리해야 함	자동으로 처리 가능
적합한 데이터 유형	중소형 데이터셋 (구조적 데이터)	대규모 데이터셋 (특히 정형 데이터)
추천 사용 사례	특성(Feature) 중요도를 알고 싶을 때	대량의 데이터에서 높은 성능이 필요할 때

앙상블이란?

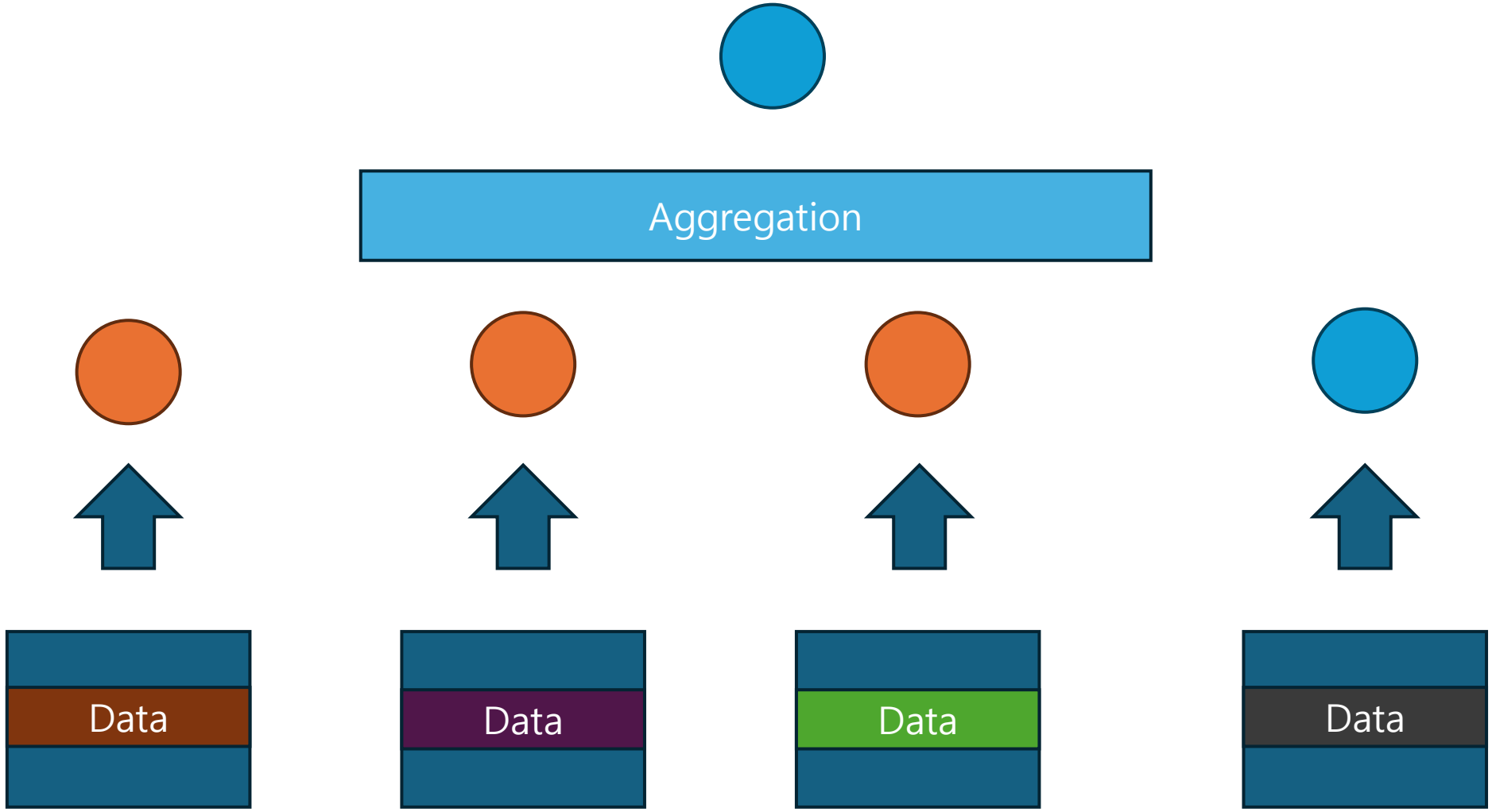


1. voting(보팅)
2. boosting(부스팅)
3. bagging(배깅)
4. stacking(스태킹)

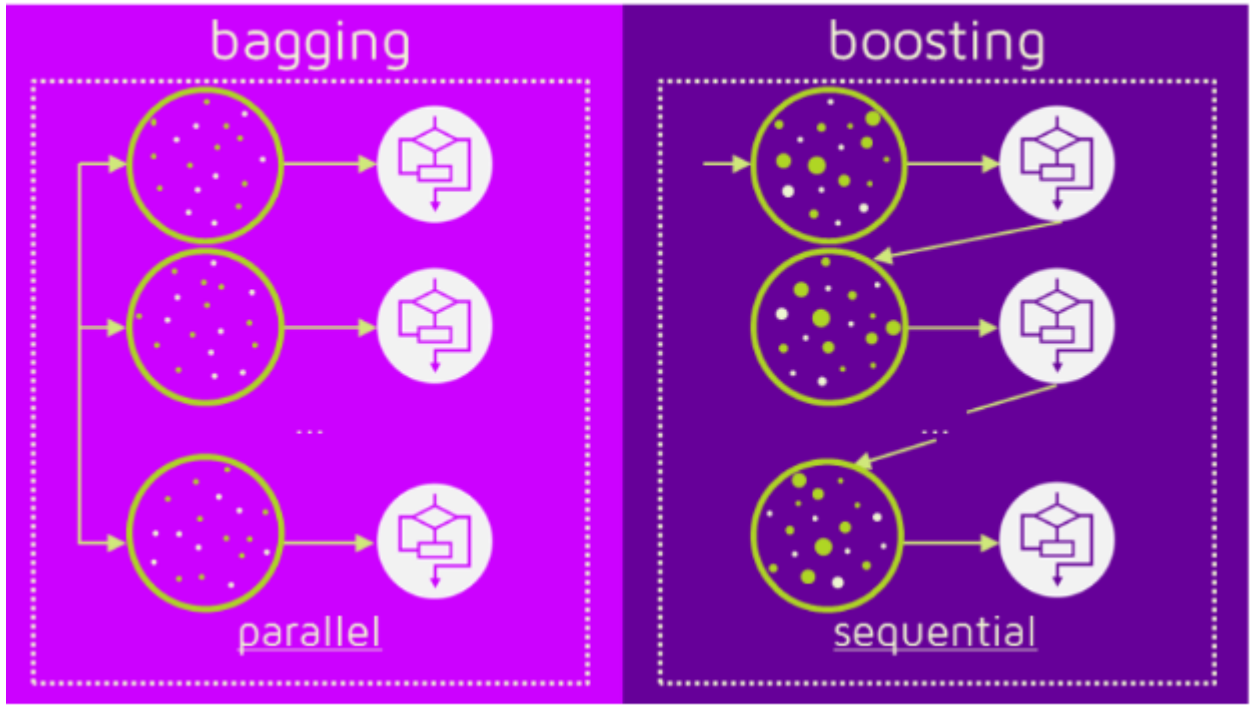
Voting : 여러 개의 모델의 결과 값을 투표로 선정



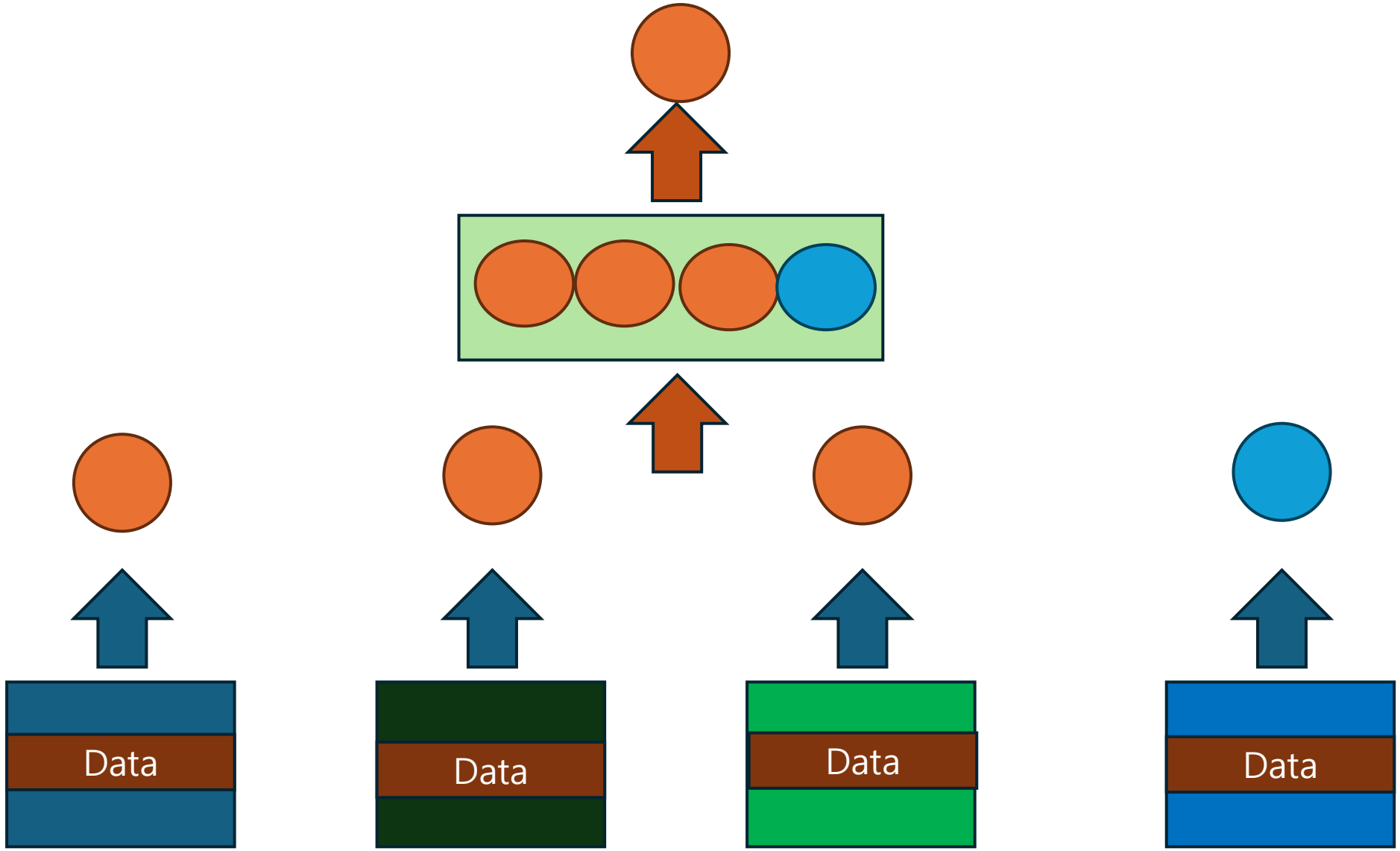
bagging : 복원 추출한 데이터를 한개의 모델 여러개로 결과 값을 aggregation



Boosting : 병렬로 학습하는 것이 아닌 순차적으로 학습함



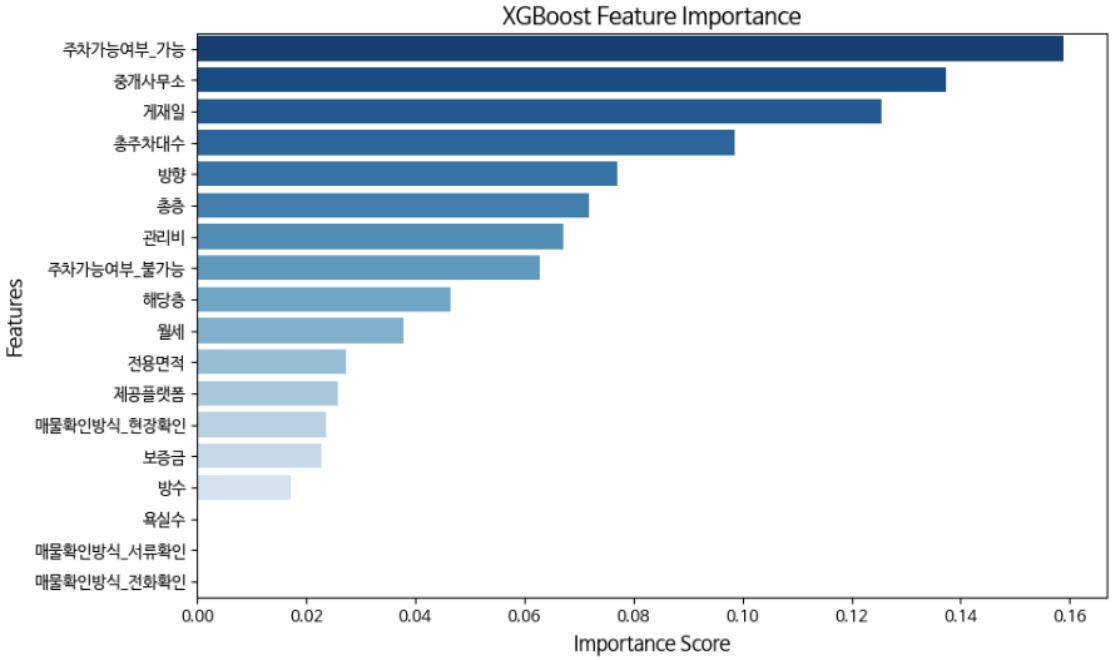
stacking : 여러 개의 모델의 결과 값으로 새로운 모델을 만들어 결과를 산출함



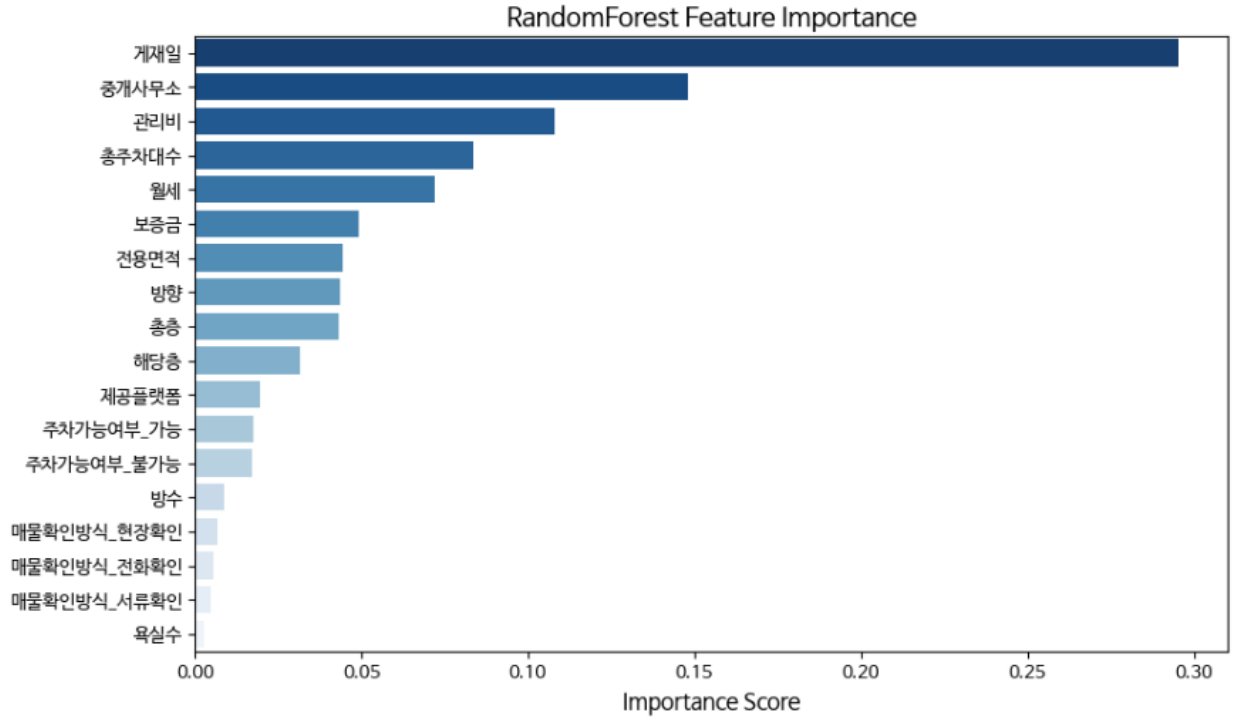
앙상블 기법	대표 모델
보팅 (Voting)	VotingClassifier
배깅 (Bagging)	RandomForestClassifier, BaggingClassifier
부스팅 (Boosting)	XGBoost, LightGBM, CatBoost, GradientBoostingClassifier
스태킹 (Stacking)	StackingClassifier, Custom Stacking

간단한 성능 비교

Mice + XGboost 0.8471528472



Mice + 랜덤포레스트 0.8152173913

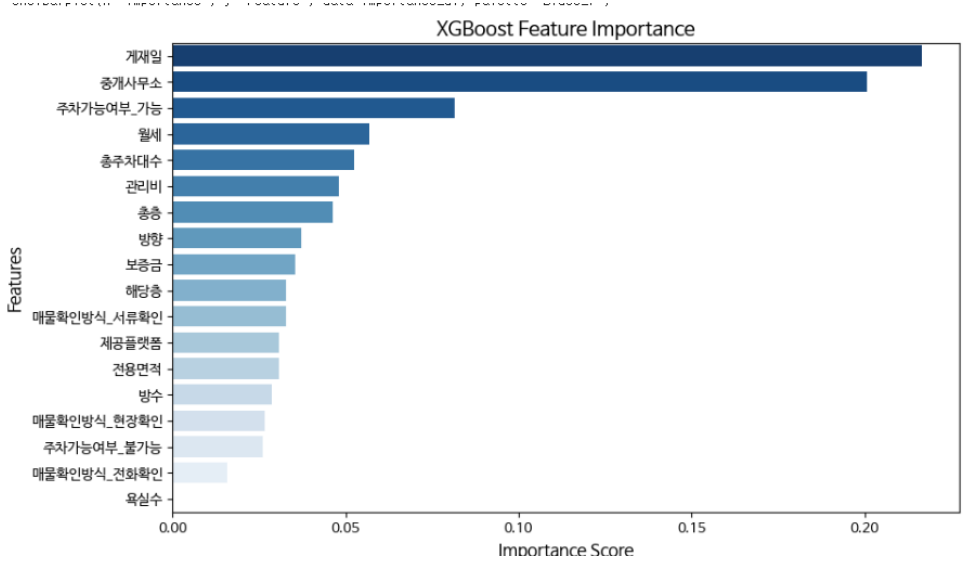


간단한 성능 비교

재밌는 건 media으로 imputation
한거랑 Mice로 한거랑 모델 정확도가 비슷함

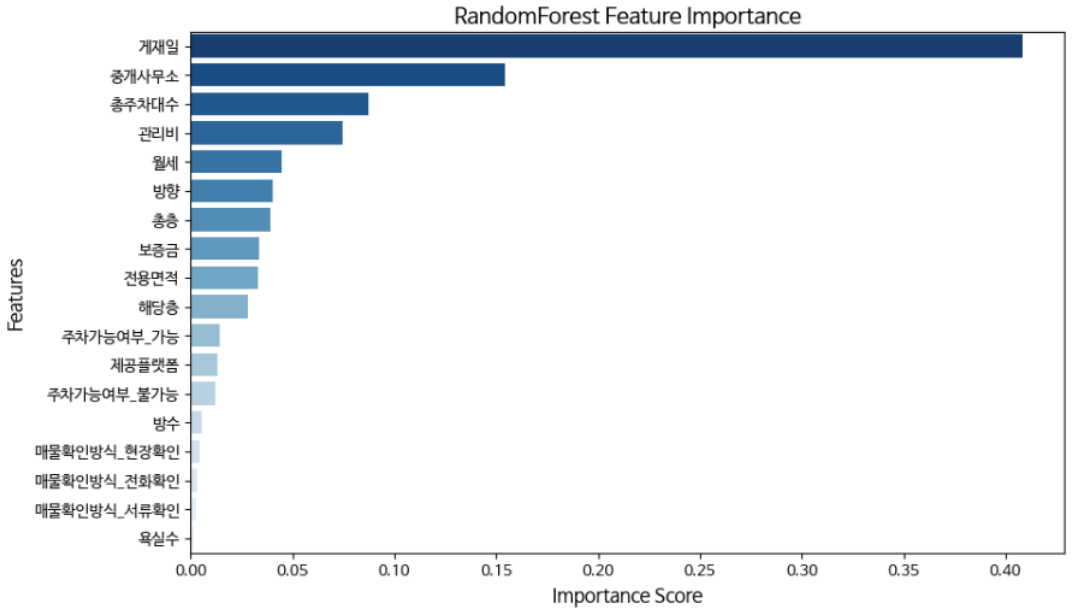
Mice + Xgboost+ 가중치

0.8677916995



Mice + RandomForest+ 가중치

0.824017527



```
{'n_estimators': 872,
 'max_depth': 10,
 'learning_rate': 0.025315626753467668,
 'subsample': 0.8312445600776525,
 'colsample_bytree': 0.8455336430028378,
 'gamma': 0.41026833548824077,
 'min_child_weight': 5,
 'scale_pos_weight': 3.036502102793806}
```

2시간 동안 최적화 시킨 Xgboost

0.8677916995

{'n_estimators': 872, 'max_depth': 10, 'learning_rate': 0.025315626753467668, 'subsample': 0.8312445600776525, 'colsample_bytree': 0.8455336430028378, 'gamma': 0.41026833548824077, 'min_child_weight': 5, 'scale_pos_weight': 3.036502102793806}

XGBClassifier

```
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.8455336430028378, device=None,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric='logloss', feature_types=None,
              gamma=0.41026833548824077, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.025315626753467668,
              max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=10, max_leaves=None,
              min_child_weight=5, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=872, n_jobs=None,
              num_parallel_tree=None, random_state=42, ...)
```

보완점

결론적으로 0.87에서 더 이상 성능을 높이지 못했음

feature engineering을 통해 예측치를 더 높이고 싶었지만 시간상 못했음

게재일에 너무 눈이돌아가서 이것만 살펴봤는데 다시 생각해보면
이렇게 높은 feature는 내비두고 다른 애들의 성능을 높이는 방향으로 설계하는게 맞았을꺼 같음

NEXT

제출 XP 획득!

건설공사 사고 예방 및 대응책 생성 : 한솔데코 시즌3 AI 경진대회

알고리즘 | NLP | 생성형 AI | LLM | MLOps | 유사도

₩ 상금 : 1,000만 원

2025.02.17 ~ 2025.03.24 09:59 + Google Calendar

762명 D-26

참여

제출 XP 획득!

채무 불이행 여부 예측 해커톤: 불이행의 징후를 찾아라!

데이콘 해커톤 | 알고리즘 | 정형 | 분류 | 금융 | ROC-AUC

₩ 상금 : 데이스쿨 프로 구독권

2025.02.03 ~ 2025.03.31 09:59 + Google Calendar

533명 D-33

참여