

데이터마이닝 Term Project 보고서

건국대학교
기계항공공학부
201610841유재성
2021-06-23

1. 분석 목적

연합뉴스 PICK | 14시간 전 | 네이버뉴스

공급 절벽에 매물 잠김까지...뛰고 또 뛰는 집값

한은 "공급 부족, 이례적 저금리로 집값 급등" 전문가들 "하방 요인 적어 강보합세 이어질 듯" 서울 아파트... 씬 없이 뛰는 집값...천장이 뚫렸다 한국부동산원 주택 ...

뛰고 또 뛰는 집값...문 대통령... 디지털타임스 PICK | 12시간 전 | 네이버뉴스
집값, 한 주도 거르지 않고 '뛰고 또 뛰어'... MBN PICK | 9시간 전 | 네이버뉴스



연합뉴스 PICK | 1일 전 | 네이버뉴스

"집값의 6~16%만으로"...與, 인천 안산 파주 등 6곳에 1만가구(종...

송영길표 '누구나집' 시범사업 부지 발표...내년 초 분양 목표 고상민 홍규빈 기자 =
집값의 6~16%만 내면... 송영길 대표가 인천시장 재직 시절 시범 도입한 정책으...

집값 6% 내고 10년 거주 '누구나집' 1... 한겨레 PICK | 21시간 전 | 네이버뉴스
'집값 10%'만 내고 입주 가능...인천·시흥 등 6... 매일경제 | 1일 전 | 네이버뉴스
與, 집값 6~16%내고 거주하는 '누구나집' 시... 국민일보 | 1일 전 | 네이버뉴스
"집값 6~16%면 입주"...'누구나 집' 두고 시장 반응 '분분' 뉴데일리 | 1일 전

관련뉴스 7건 전체보기 >



- 최근 집 값에 대한 관심이 뜨거움
- 실거래가에 영향을 주는 영향을 주는 요인 분석 및 모델링을 통한 거래가를 예측
- 모델의 예측력이 높다면 실제 투자 및 구매에도 참고할 수 있을 것으로 기대

2. 데이터

- 데이콘 '아파트 실거래가 예측' 데이터 이용

2-1. 데이터 구조

Aa 컬럼명	: 태그	≡ 정보	≡ 데이터타입	≡ 비고
<u>Transaction_id</u>		거래 ID	수치형	
<u>Apartment_id</u>		아파트 ID	수치형	
<u>City</u>		도시명	문자형	서울특별시, 부산광역시 2곳
<u>Dong</u>		동	문자형	
<u>Jibun</u>		지번	문자형	
<u>Apt</u>		아파트명	문자형	
<u>Add_kr</u>		주소	문자형	

Aa 컬럼명	:≡ 태그	≡ 정보	≡ 데이터타입	≡ 비고
<u>Exclusive_use_area</u>		전용면적	수치형	
<u>Year_of_completion</u>		완공일	수치형	
<u>Transaction_year_month</u>		거래연월	수치형	
<u>transaction_date</u>		거래일	문자형	1~10, 11~20, 21~31로 구성
<u>Floor</u>		층	수치형	
<u>Transaction_real_price</u>		실거래가	수치형	예측변수(단위: 만원)

2-1. 데이터 크기

- 처음부터 train 데이터와 test데이터가 분리되어 있음
 - train data: 1216552행 13열로 구성
 - test data: 5463행 12열로 구성 (예측변수 'transaction_real_price'는 제외됨)

2-2. 데이터 구조

- train 데이터의 상위 5개 행

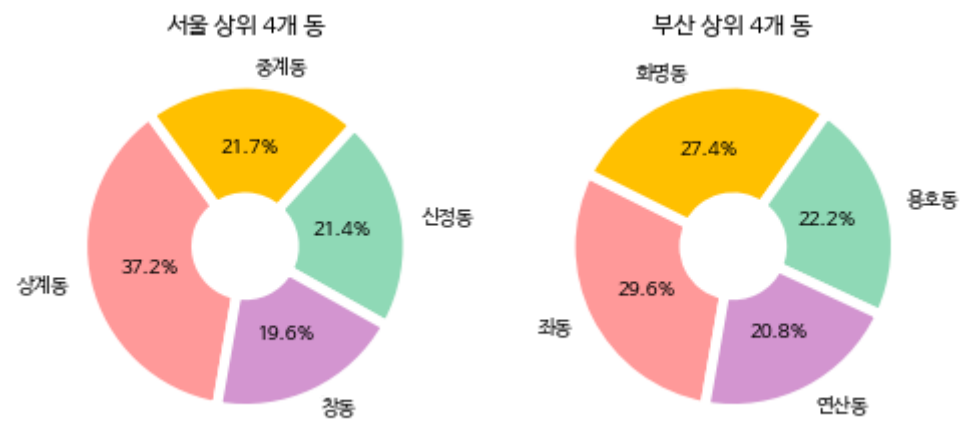
	transaction_id	apartment_id	city	dong	jibun	apt	addr_kr
0	0	7622	서울특별시	신교동	6-13	신현(101동)	신교동 6-13 신현(101동)
1	1	5399	서울특별시	필운동	142	사직파크맨션	필운동 142 사직파크맨션
2	2	3578	서울특별시	필운동	174-1	두레엘리시안	필운동 174-1 두레엘리시안
3	3	10957	서울특별시	내수동	95	파크팰리스	내수동 95 파크팰리스
4	4	10639	서울특별시	내수동	110-15	킹스매너	내수동 110-15 킹스매너

exclusive_use_area	year_of_completion	transaction_year_month	transaction_date	floor	transaction_real_price
84.82	2002	200801	21~31	2	37500
99.17	1973	200801	1~10	6	20000
84.74	2007	200801	1~10	6	38500
146.39	2003	200801	11~20	15	118000
194.43	2004	200801	21~31	3	120000

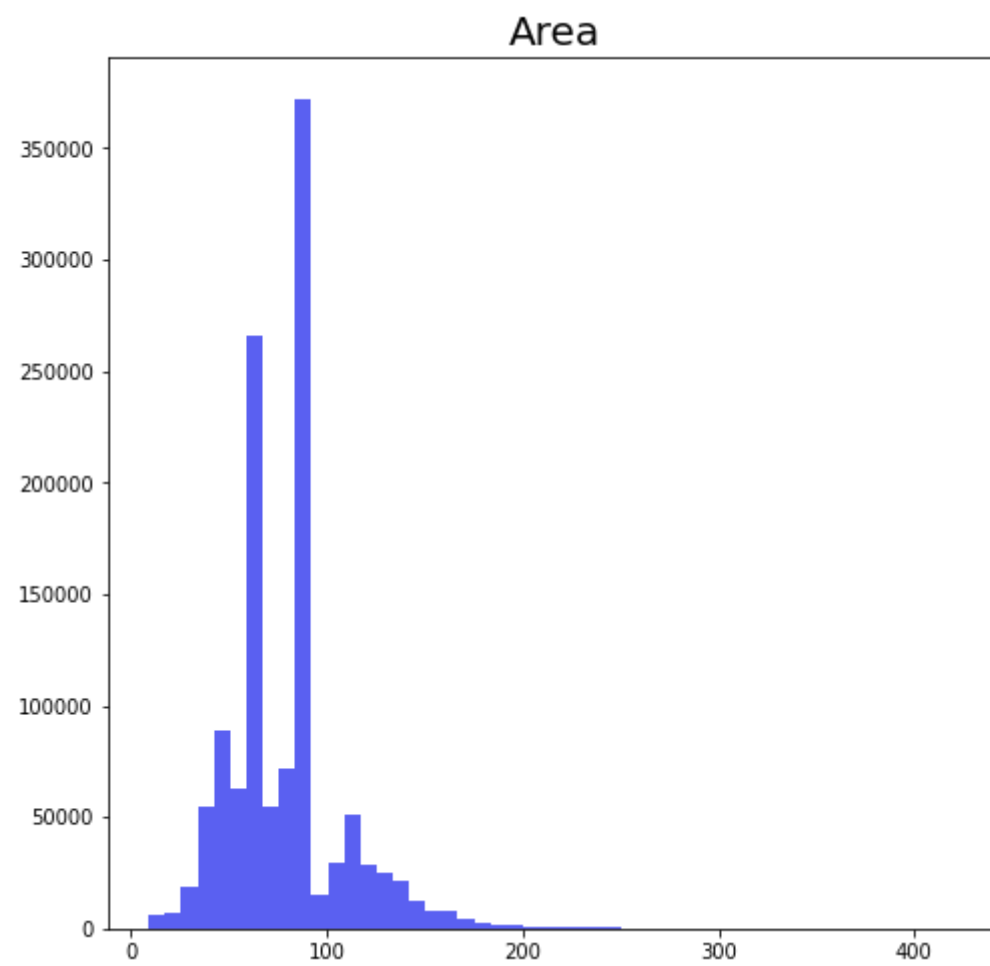
2-3. 변수 설명

test도 유사하기 때문에 train 데이터를 대표로 설명

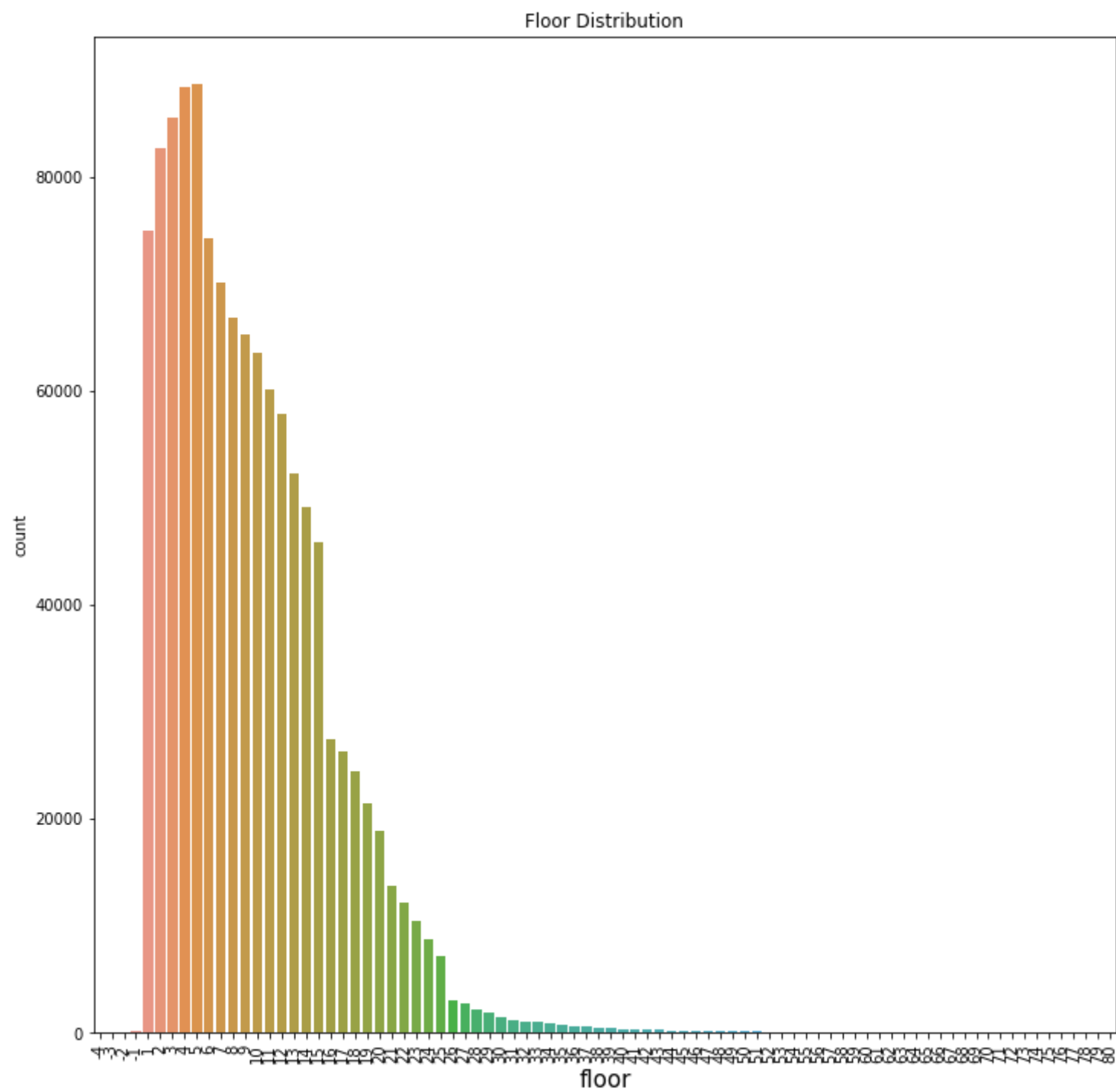
- **transaction_id**
 - 0부터 1,234,827까지 존재
 - 행의 개수인 1,216,553보다 많은 이유는 **transaction_id**가 연속적이지 않고 중간에 건너뛰는 숫자가 존재하기 때문
 - 중간에 건너뛰는 id에 해당하는 데이터는 test데이터에 들어가 있음
- **apartment_id**
 - 0부터 12,658까지 존재
- **city**
 - 서울특별시, 부산광역시 두가지만 존재
 - 서울은 742,285행, 부산은 474,268행 존재
- **dong**
 - 473개의 동이 존재
 - 거래가 가장 많았던 도시별 상위 4개의 동



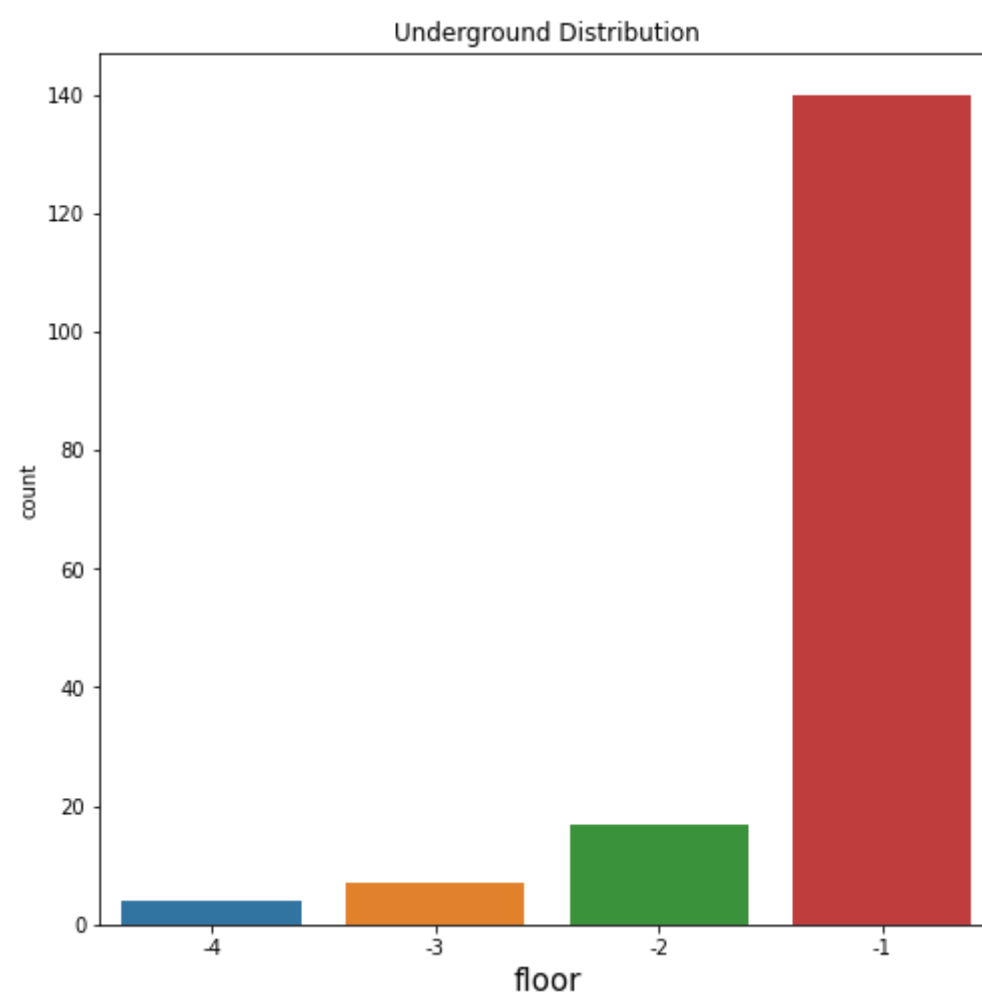
- `apt`
 - 10,440개의 이름이 다른 아파트가 존재
- `exclusive_use_area` (단위: m^2)



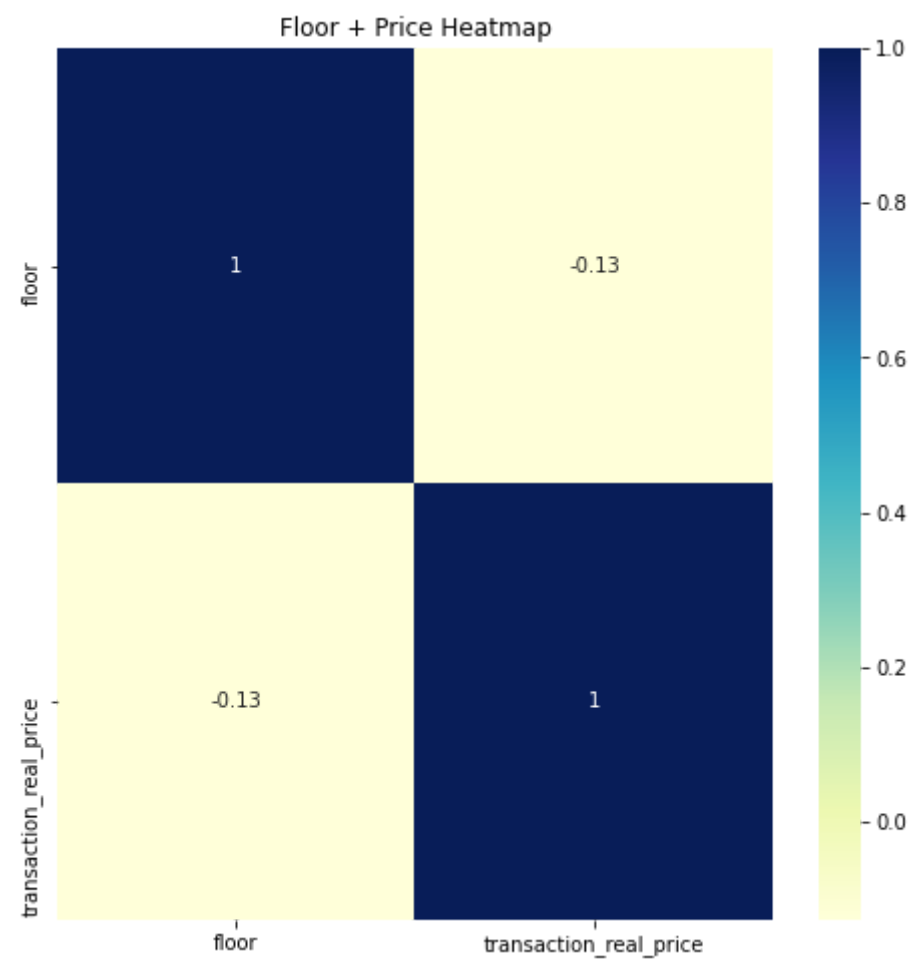
- `year_of_completion`
 - 최소연도: 1961년
 - 최대연도: 2017년
- `transaction_year_month`
 - 최소연도: 2008년 1월
 - 최대연도: 2017년 11월
- `transaction_date`
 - 1~10, 11~20, 21~30 세 가지 카테고리
 - 거래 일자 는 무의미하다고 판단하여 삭제
- `floor`
 - -4층부터 80층까지 존재



- 지하(-4~-1층)의 경우 168개 존재



- 층이 지하일 때, 가격과의 상관관계가 -0.13



- `transaction_real_price` (단위: 만원)
 - 최소 실거래가: 100만 원

transaction_id	apartment_id	city	dong	jibun	apt	addr_kr
722888	6225	부산광역시	범전동	263-5	서면	범전동 263-5 서면

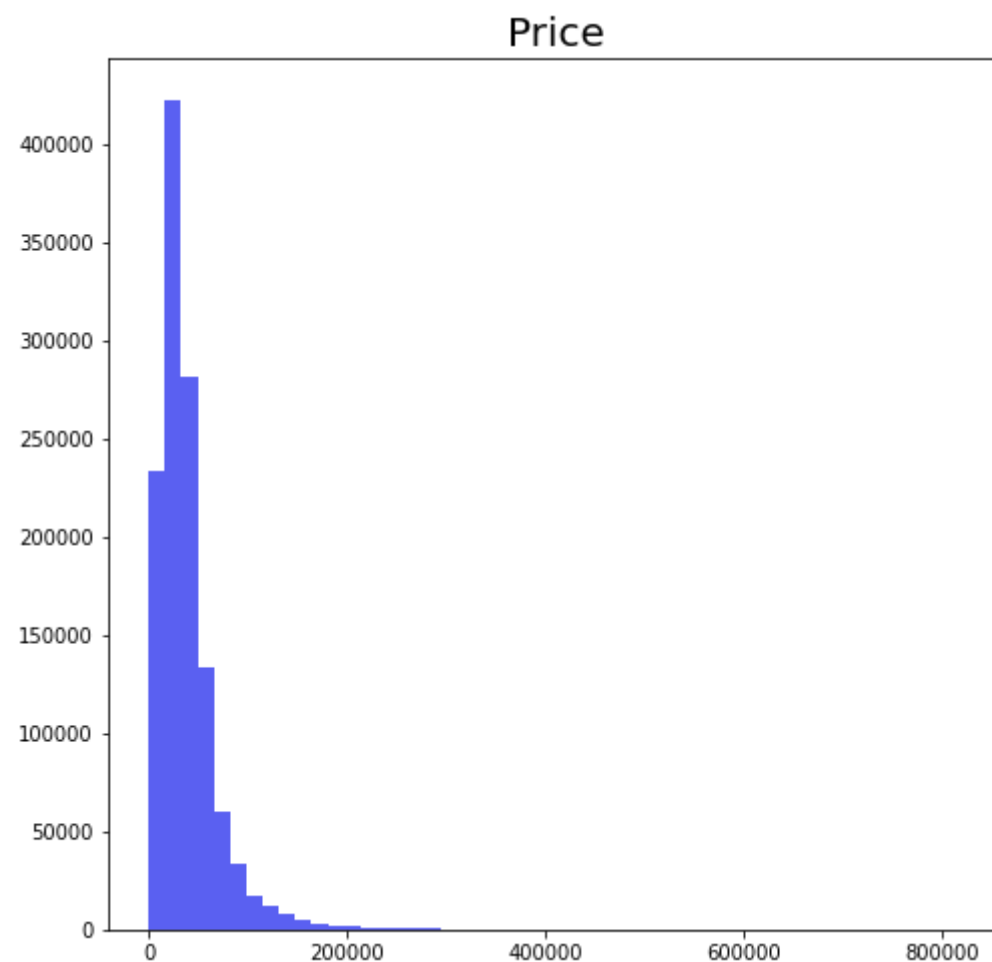
exclusive_use_area	year_of_completion	transaction_year_month	transaction_date	floor	transaction_real_price
138.25	1974	201606	21~30	-1	100

- 최대 실거래가: 82억 원

transaction_id	apartment_id	city	dong	jibun	apt	addr_kr
23603	11320	서울특별시	한남동	810	한남더힐	한남동 810 한남더힐

exclusive_use_area	year_of_completion	transaction_year_month	transaction_date	floor	transaction_real_price
244.749	2011	201612	21~31	3	820000

- 가격 분포



- 좌측으로 편향된 분포를 확인할 수 있음

3. 데이터 전처리(Preprocessing)

3-1. 도시 더미변수화

- `city` 변수는 `서울특별시`, `부산광역시` 2가지의 카테고리로 이루어져 있음
- 문자열로 처리하지 않고 **더미변수로 변경함**
 - `서울특별시` = 0
 - `부산광역시` = 1

3-2. 주소 관련 변수

1. 주소(`addr_kr`)

- 아파트 주소의 경우 아파트(`apt`), 지번(`jibun`), 동(`dong`)에 관한 정보가 중복
 - 예) `addr_kr`: 신교동 6-13 신현(101동) ↔ `dong`: 신교동, `jibun`: 6-13, `apt`: 신현(101동)

<code>dong</code>	<code>jibun</code>	<code>apt</code>	<code>addr_kr</code>
신교동	6-13	신현(101동)	신교동 6-13 신현(101동)

- 따라서 **포괄적인 정보가 담긴 `addr_kr` 제거**

2. 번지수(`jibun`)

- 아파트가 몇 번지에 속하는 지에 대한 정보를 처리하기 어렵다고 판단
- 해당 **변수(`jibun`) 제거**

3. 동(`dong`)

- 서울과 부산에 중복되는 이름을 가진 동을 접두사를 붙여 분리
 - 중복되는 동: `사직동`, `부암동`, `중동`, `송정동` 총 4개
 - 예) 사직동: `서울사직동`, `부산사직동`
- 같은 동이어도 세부 동이 다를 경우 가격차이가 상당히 존재

dong	
장충동1가	269888.888889
장충동2가	6628.421053

- 따로 숫자를 통일하지 않고 다른 동처럼 취급하기로 결정
- 동 별 실거래가를 내림차 순으로 정렬하여 순서형 범주로 변경
 - 실거래가 상위 10개 동

장충동1가	269888.888889
압구정동	164534.722914
청암동	161403.700000
용산동5가	153497.331633
회현동2가	139906.140351
반포동	132489.395651
한남동	122593.293264
서빙고동	116547.239777
대치동	116320.538909
남대문로5가	113153.604651

- 순서형 범주 변경 예) 장충동1가 → 0, 압구정동 → 1, 청암동 → 2, 용산동5가 → 3, ...

3-3. 아파트

1. 아파트명에 존재하는 괄호 제거

- 괄호내에 아파트 호 수 또는 번지수가 포함되어있음
- 동일한 아파트 수를 세는 apt_count 생성을 위해 괄호와 괄호 내 존재하는 내용 제거
- 정규표현식 사용: "\(.*\)|\s-\s.*"
- 괄호 제거 예시: 신현(101동) → 신현

transaction_id	apartment_id	city	dong	jibun	apt
0	0	서울 특별시	신교 동	6-13	신현 (101동)

transaction_id	apartment_id	city	dong	jibun	apt
0	0	서울특 별시	신교 동	6-13	신현

2. 상위 10개의 시공사 여부를 나타내는 top10 더미변수 추가

- 아파트 브랜드가 거래가에 유의미한 영향을 미친다고 판단
- 상위 10개 시공사 목록: 자이, 푸르지오, 더샵, 롯데캐슬, 이편한, 힐스테이트, 아이파크, 래미안, sk, 데시앙
- 아파트 명에 해당 시공사 단어가 포함돼있으면 1, 없으면 0
- 예) 반포자이: top10 = 1, 태전성원아파트: top10 = 0

3. 상위10개 시공사 + 가장 많이 등장한 아파트 이름 20개만 남기고 나머지 이름 통일

- 다른 이름의 아파트가 약 1만개정도 되서 수치형변수나 더미변수 처리가 어렵기 때문에 상위 30개만 고려
- 데이터셋에 가장 많이 포함된 아파트 상위 20개 리스트

현대	17716
한신	10135
삼성	6771
대우	6390
신동아	6386
두산	5801
우성	5781
주공2	5669
삼성래미안	5483
벽산	4651
대림	4582
동원로얄듀크	4430
경남	4030
삼환	3896
극동	3771
삼익	3583
롯데캐슬	3570
쌍용	3409
코오롱	3320
한양	3231

- 예) '삼성래미안' → '래미안'(상위10 시공사이름)으로 통일,
'OO우성아파트' → '우성'으로 통일
상위20개에 포함되지 않은 아파트명은 **others** 로 통일
- 변환 후 아파트 개수 리스트

others	653503
현대	81824
주공	65937
래미안	32743
한신	27481
벽산	26968
우성	24524
롯데캐슬	24053
삼성	20067
sk SK 에스케이	19709
두산	19501
삼익	18708
푸르지오	18416
쌍용	18256
대림	18194
대우	17786
이편한 e편한 e-편한	14337
신동아	14049
힐스테이트	11985
자이	11719
엘지	10938
경남	10875
코오롱	8602
아이파크	7989
성원	7601
더샵	6977

- 아파트 별 실거래가의 평균을 내림차 순으로 정렬하여 **dong** 변수를 처리할 때와 동일하게 **순서형 범주 처리**
 - 예) **잠실 → 0, 파크리오 → 1, 자이 → 2, 래미안 → 3, ...**
 - 상위 5개 아파트 변환 예시

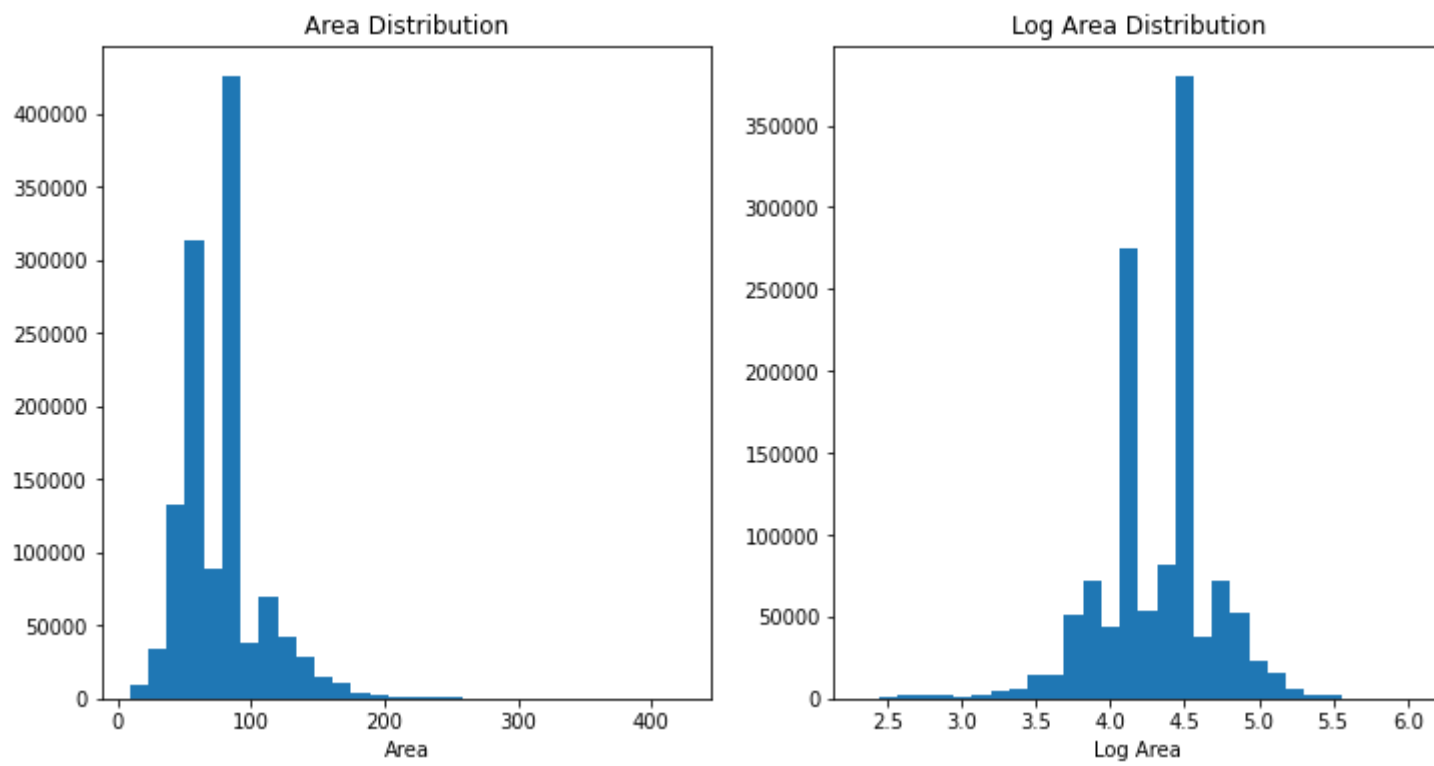
```

apt
잠실      91273.525326
파크리오  88739.646736
자이      79838.112467
래미안    64184.531289
아이파크  63115.982726
Name: transaction_real_price, dtype: float64
apt
0      91273.525326
1      88739.646736
2      79838.112467
3      64184.531289
4      63115.982726
Name: transaction_real_price, dtype: float64

```

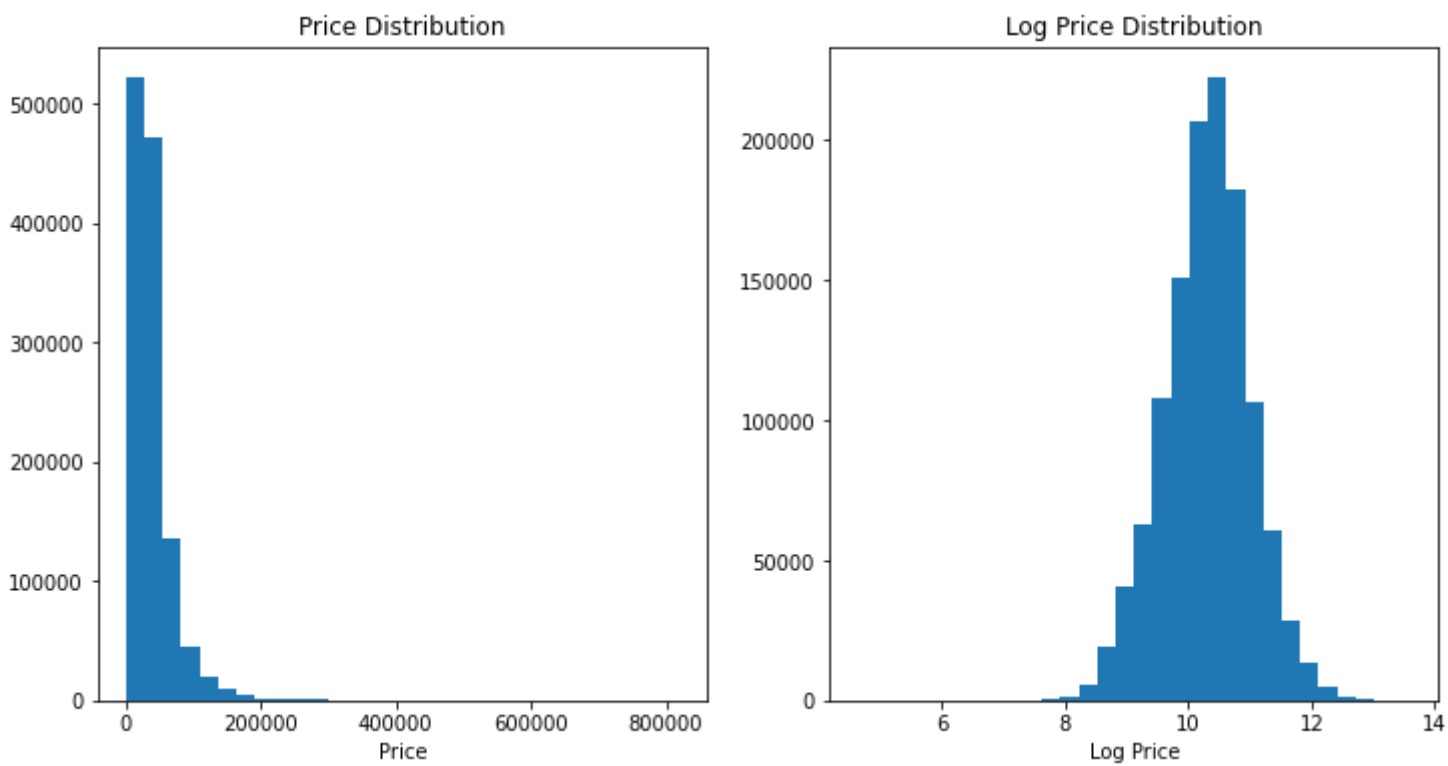

3-4. 면적

- 좌측으로 편향된 데이터를 **log 변환**을 통해 정규분포와 유사한 형태로 변환



3-5. 실거래가

- 면적과 동일한 방법으로 변환
- 면적보다 더 정규분포처럼 보임



3-6. 완공연도 및 거래연월

1. 완공연도

- 연도가 증가하는 순으로 순서형 범주 처리
- 예) 1961 → 0, 1962 → 1, ..., 2017 → 56

변환전
[2002 1973 2007 2003 2004]
변환후
[41 12 46 42 43]

2. 거래연월

- 마찬가지로 연월이 증가하는 순으로 순서형 범주 처리
- 예) 200801 → 0, 200802 → 1, ..., 201712 → 119

```

변환전
[200801 200802 200803 200804 200805]
변환후
[0 1 2 3 4]

```

3-7. 층 수

- 최소 층 이 -4층이므로 모든 층에 4를 더해서 음수를 없애고 순서형 범주 처리
- 예) `-4 → 0, -3 → 1, ..., 80 → 84`

```

변환전
[ 2 6 6 15 3]
변환후
[ 6 10 10 19 7]

```

3-8. 그 외 변수 제거

- ID 정보는 학습 시 모델이 의미있는 값으로 해석할 수 있기 때문에 제거: `transaction_id`, `apartment_id`
- 더미변수로 변환했기 때문에 원래 변수인 `city` 제거

3-9. 최종 데이터 프레임 예시

- 기존 형태

	transaction_id	apartment_id	city	dong	jibun	apt	addr_kr
0	0	7622	서울특별시	신교동	6-13	신현(101동)	신교동 6-13 신현(101동)
1	1	5399	서울특별시	필운동	142	사직파크맨션	필운동 142 사직파크맨션
2	2	3578	서울특별시	필운동	174-1	두레엘리시안	필운동 174-1 두레엘리시안
3	3	10957	서울특별시	내수동	95	파크팰리스	내수동 95 파크팰리스
4	4	10639	서울특별시	내수동	110-15	킹스매너	내수동 110-15 킹스매너

exclusive_use_area	year_of_completion	transaction_year_month	transaction_date	floor	transaction_real_price
84.82	2002	200801	21~31	2	37500
99.17	1973	200801	1~10	6	20000
84.74	2007	200801	1~10	6	38500
146.39	2003	200801	11~20	15	118000
194.43	2004	200801	21~31	3	120000

- 최종 형태

	dong	apt	year_of_completion	transaction_year_month	floor	top10	log_price	log_area	city_부산광역시	city_서울특별시
0	138	23	41	0	6	0	10.532123	4.452252	0	1
1	65	23	12	0	10	0	9.903538	4.606869	0	1
2	65	23	46	0	10	0	10.558439	4.451319	0	1
3	13	23	42	0	19	0	11.678448	4.993082	0	1
4	13	23	43	0	7	0	11.695255	5.275202	0	1

- 13열에서 10열로 변경

4. 모델링

4-1. 평가 지표

$$RMSLE = \sqrt{\frac{1}{N} \sum_{i=1}^N (\log(p_i + 1) - \log(a_i + 1))^2}$$

$p = Predicted, a = Actual$

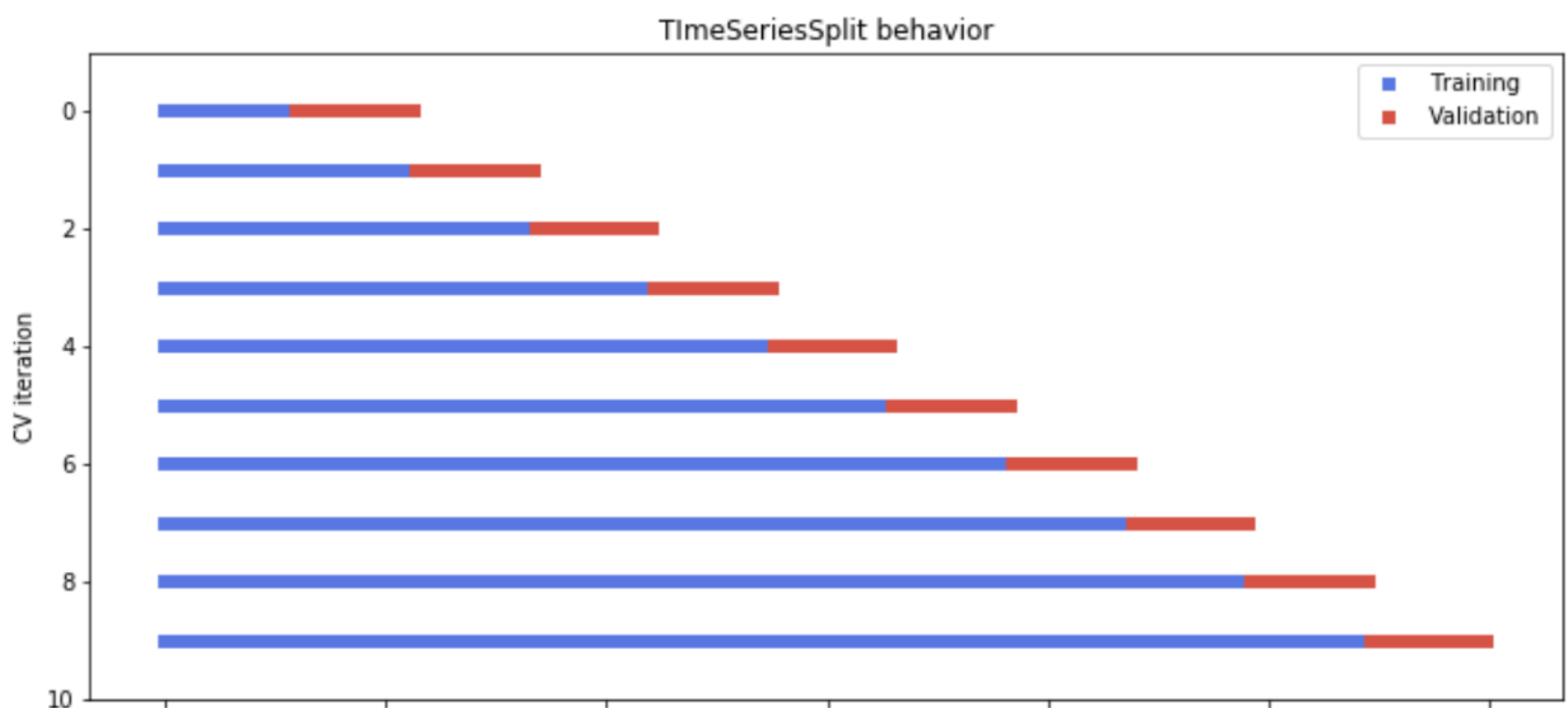
- 타겟 값이 좌측으로 편향되었기 때문에 RMSE대신 정규화의 역할을 수행하는 **RMSLE 사용**
- 타겟 값에 그냥 log를 씌울 경우 값이 0일 때 발산할 수 있으므로, 1을 더한 값에 log를 씌움
- log값으로 보정되기 때문에 이상치에도 Robust한 결과를 가져올 수 있음

$$\log(p_i + 1) - \log(a_i + 1) = \log\left(\frac{p_i + 1}{a_i + 1}\right)$$

- 상대적 오차를 측정
 - 예측값이 실제값보다 작은 경우(Under Estimation) 더 큰 페널티를 부여함(이 경우 로그값이 음수가 되는데 음수쪽 기울기가 양수쪽 기울기보다 크기 때문)

4-2. 교차 검증

- 해당 자료는 시계열 자료라고 볼 수 있음
 - 일반 Kfold를 사용할 경우 섞인 값 중에 미래의 값으로 과거의 값을 예측할 가능성이 있음
 - 성능은 좋게 나올 수 있으나, 실제로는 미래의 값을 사용할 수 없기 때문에 부적절하다고 판단
 - 시계열자료에서 사용하는 **TimeSeriesSplit 기법을 사용, 10folds 적용**
 - 총 10개의 폴드로 구분하여 앞에서 부터 끊어서 학습 및 검증 진행
 - 다음 폴드에선 이전의 폴드 데이터까지 이용하여 반복적으로 교차 검증
 - 이 경우 미래의 값은 무조건 검증에 이용됨(미래의 값이 학습에 이용되지 않음)

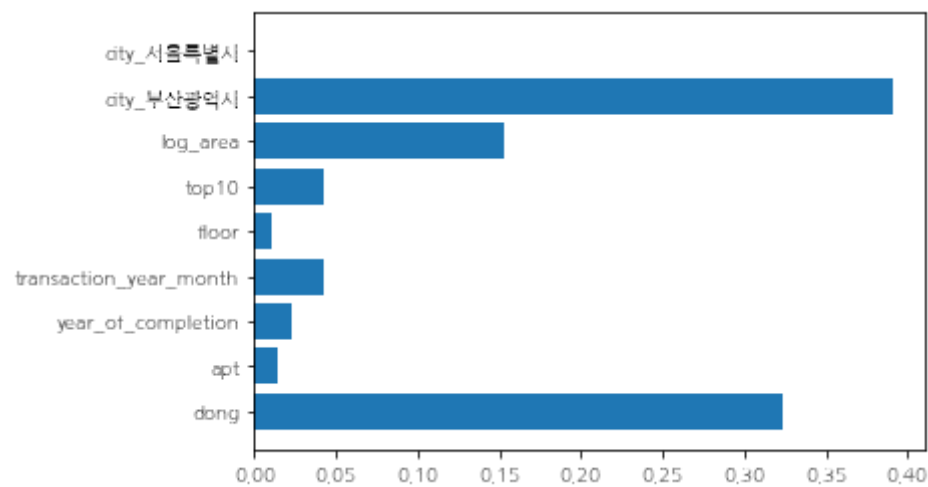


- 데이터가 충분히 많다고 판단하여 **Randomization**을 적용하지 않음

4-3. 모델 비교

- 4-2에서 정의한 10 Folds TimeSeriesCV를 통해 평균 RMSLE를 측정

- 평균 RMSLE Score
 - Linear Regression: 0.2888
 - Ridge Regression: 0.2888
 - Lasso Regression: 0.2990
 - Elasticnet Regression: 0.2960
 - Decision Tree (Regression): 0.3092
 - Random Forest: 0.2749
 - XGBoost: **0.2390**
 - LightGBM: 0.2419
- 앙상블 모델과 부스팅 모델의 점수가 낮게 나오는 걸 확인
- 이 중 가장 낮은 RMSLE를 달성한 **XGBoost를 최종 모델로 선정**
- Feature Importance
 - XGBoost로 Feature Importance를 측정



- apt와 floor 변수가 다른 변수에 비해 중요도가 낮게 나옴
- 둘다 제거하는 대신 처리 비용이 높은 apt만 제거 후 동일한 방식으로 분석

4-4. 최종 모델 하이퍼파라미터 튜닝

1. 데이터 분할

- 최종학습을 위해 학습에는 train의 80%(973,242행), 검증에는 train의 20%(243,311행)을 사용
- 위에서 설명했던 것처럼 과거의 값을 미래 값으로 예측하지 않도록 학습데이터는 앞에서 80%, 검증 데이터는 뒤에서 20%을 이용

2. 하이퍼파라미터 튜닝

- pruning을 통해 시간 감축, 약 2시간 소요

XGBoost Hyperparameter

Aa 파라미터명	≡ 설명	:: 값
<u>booster</u>	부스터 구조	gbtree
<u>lambda</u>	L2 정규화 정도	2.27418e-5
<u>alpha</u>	L1 정규화 정도	0.000292
<u>max_depth</u>	트리의 최대 깊이	9
<u>eta</u>	학습율	0.055985
<u>gamma</u>	분할 수행 시 필요한 최소 손실 감소	0.005501
<u>n_estimators</u>	반복 수행 횟수	1181

Aa 파라미터명	≡ 설명	≡ 값
<u>min_child_weight</u>	최소 가중치 합	10
<u>subsample</u>	트리의 관측 데이터 샘플링 비율	0.415083

3. 튜닝 후 RMSLE Score: **0.2001**

- 가장 낮은 RMSLE을 달성

5. 예측

5-1. Test 데이터 예측

- 4-4에서 얻은 하이퍼 파라미터를 바탕으로 Test 데이터 예측
- 데이터 분할
 - 학습에는 **train 데이터 전체** 사용(1,216,553행)
 - 테스트는 **test 데이터 전체** 사용(5,463행)
 - test 데이터의 거래연월은 train 데이터 보다 늦으므로 이 경우 미래의 값으로 과거의 값을 예측하지 않음

5-2. 예측 결과

- Test 데이터에는 타겟 값이 없기 때문에 Dacon에서 직접 제출해서 평가
- Dacon 대회의 평가지표는 RMSE기 때문에 예측값에 다시 exp을 취하고 -1을 뺀(log를 취할 때 1을 더했기 때문)

제출한 예측 결과물은 Root Mean Squared Error(RMSE)로 평가합니다(평가는 게시된 데이터셋만을 기준으로 합니다).

- Public Score 1등 달성

PUBLIC

PRIVATE








순위기준

WINNER

1%

4%

10%

#	팀	팀 멤버	점수	제출수	등록일
1	유재성 KADE		4795.7994	7	2시간 전
1	유재성 KADE		4,795.7994	7	2시간 전
2	통계청김웅곤+양경성+민성욱EDA김현우	   	5,787.87666	139	일 년 전
3	최정명		8,049.69405	8	7달 전

5-3. 추가 데이터 예측

- 2018년의 공공데이터를 추가로 사용해서 실거래가를 예측
 - 데이터 형태가 달라 전처리에 어려움이 있어, 11개의 데이터만 사용
 - 데이터 목록

city	dong	apt	exclusive_use_area	transaction_year	transaction_real_price	floor	year_of_completion
서울특별시	개포동	개포6차우성아파트1동~8동	79.97	120	1,300,000	4	1987
서울특별시	개포동	성원대치2단지아파트	39.53	121	884,000	10	1992
서울특별시	대치동	한보미도맨션2	115.05	123	2,280,000	13	1985
서울특별시	대치동	한티	102.43	125	967,000	4	2004
서울특별시	대치동	해암프리존	204.38	120	1,400,000	4	1996
서울특별시	대치동	현대1	84.27	122	1,175,000	4	1990
서울특별시	대치동	현대1	84.27	127	1,480,000	8	1990
서울특별시	대치동	현대1	84.27	128	1,520,000	2	1990
서울특별시	대치동	현대썬앤빌 테헤란	49.24	121	650,000	7	2016
서울특별시	대치동	현대썬앤빌 테헤란	33.09	123	482,000	6	2016
서울특별시	대치동	현대썬앤빌 테헤란	49.24	124	679,190	7	2016

- 전처리후 데이터프레임

	dong	transaction_year	floor	year_of_completion	top10	city_서울특별시	city_부산광역시	log_price	log_area
0	41	120	8	26	0	1	0	11.775297	4.394079
1	41	121	14	31	0	1	0	11.389639	3.702042
2	8	123	17	24	0	1	0	12.337105	4.754021
3	8	125	8	43	0	1	0	11.479379	4.638895
4	8	120	8	35	0	1	0	11.849405	5.324862
5	8	122	8	29	0	1	0	11.674202	4.445823
6	8	127	12	29	0	1	0	11.904974	4.445823
7	8	128	6	29	0	1	0	11.931642	4.445823
8	8	121	11	55	0	1	0	11.082158	3.916812
9	8	123	10	55	0	1	0	10.783135	3.529004
10	8	124	11	55	0	1	0	11.126086	3.916812

- RMSLE: 1.89205
- 상당히 오차가 크게 나왔다.

6. 토의 및 후기

test 데이터로 성능을 평가했을 때는 만족스러운 성과가 나왔지만, 실제로 추가로 수집한 데이터를 활용하여 예측했을 때 상당히 모델이 예측하지 못했다. 특히 Underestimation이 발생했는데, 왜 발생하였는지에 대해 몇 가지 원인을 생각해 봤다.

- XGBoost가 과적합의 이슈가 있음
- 하이퍼 파라미터 튜닝을 2017년도 까지의 데이터로 진행했기 때문에, 2018년의 데이터를 잘 예측하지 못했음(과적합)

이에 대한 몇가지 대책은 다음과 같다.

- XGBoost대신 LightGBM을 사용(XGBoost보다 빠르고 과적합에 대해 비교적 안전하다고 알려짐)
- 하이퍼 파라미터 튜닝에서 과적합을 방지하는 파라미터를 강하게 적용
- 하이퍼 파라미터 튜닝에서 epochs을 더 작게 설정
- 새로운 연도의 데이터를 사용할 때 해당 연도의 데이터에 대해 어느정도 훈련이 필요

막연히 사용했던 여러 알고리즘과 모델들이 정교한 통계적 이론 위에 설립되어 있었음을 알게 된 수업이었습니다. 이번 프로젝트를 진행하면서 지난 대회이지만 Public Score 1등을 할 수 있었습니다. 그러나 아직 해결하지 못한 새로운 데이터에 대한 문제 해결 전략 수립, 배포 및 유지 보수를 위한 코드 리팩토링 등 아직 부족한 점은 많습니다. 해당 프로젝트를 제출한 후에도 계속 보완을 해 볼 예정입니다. 제가 이런 경험을 할 수 있게 기회를 제공해주신 교수님께 감사드립니다.

7. 참고문헌 + 코드

데이콘. 아파트 실거래가 예측, <https://dacon.io/competitions/official/21265/data>

국토교통부 실거래가 공개시스템
<http://rt.molit.go.kr/>

권성훈 교수님의 데이터마이닝 교안

Tianqi Chen. XGBoost: A Scalable Tree Boosting System, 2016

Guolin Ke. LightGBM: A Highly Efficient Gradient Boosting Decision Tree, 2017

코드
<https://github.com/Haebuk/dataminingTP>