KIPS Trans. Softw. and Data Eng.
Vol.12, No.2 pp.59~76
ISSN: 2287-5905 (Print), ISSN: 2734-0503 (Online)

https://doi.org/10.3745/KTSDE.2023.12.2.59

Apartment Price Prediction Using Deep Learning and Machine Learning

Hakhyun Kim† · Hwankyu Yoo†† · Hayoung Oh†††

ABSTRACT

Since the COVID-19 era, the rise in apartment prices has been unconventional. In this uncertain real estate market, price prediction research is very important. In this paper, a model is created to predict the actual transaction price of future apartments after building a vast data set of 870,000 from 2015 to 2020 through data collection and crawling on various real estate sites and collecting as many variables as possible. This study first solved the multicollinearity problem by removing and combining variables. After that, a total of five variable selection algorithms were used to extract meaningful independent variables, such as Forward Selection, Backward Elimination, Stepwise Selection, L1 Regulation, and Principal Component Analysis(PCA). In addition, a total of four machine learning and deep learning algorithms were used for deep neural network(DNN), XGBoost, CatBoost, and Linear Regression to learn the model after hyperparameter optimization and compare predictive power between models. In the additional experiment, the experiment was conducted while changing the number of nodes and layers of the DNN to find the most appropriate number of nodes and layers. In conclusion, as a model with the best performance, the actual transaction price of apartments in 2021 was predicted and compared with the actual data in 2021. Through this, I am confident that machine learning and deep learning will help investors make the right decisions when purchasing homes in various economic situations.

Keywords : Real Estate, Regression, DNN, XGBoost, CatBoost, Map Visualization

딥러닝과 머신러닝을 이용한 아파트 실거래가 예측

김 학 현† · 유 환 규†† · 오 하 영†††

요 약

코로나 시대 이후 아파트 가격 상승은 비상식적이었다. 이러한 불확실한 부동산 시장에서 가격 예측 연구는 매우 중요하다. 본 논문에서는 다양한 부동산 사이트에서 자료 수집 및 크롤링을 통해 2015년부터 2020년까지 87만개의 방대한 데이터셋을 구축하고 다양한 아파트 정보와 경제지표 등 가능한 많은 변수를 모은 뒤 미래 아파트 매매실거래가격을 예측하는 모델을 만든다. 해당 연구는 먼저 다중 공선성 문제를 변수 제거 및 결합으로 해결하였다. 이후 의미있는 독립변수들을 뽑아내는 전진선택법(Forward Selection), 후진소거법(Backward Elimination), 단계적선택법(Stepwise Selection), L1 Regularization, 주성분분석(PCA) 총 5개의 변수 선택 알고리즘을 사용했다. 또한 심층신경망(DNN), XGBoost, CatBoost, Linear Regression 총 4개의 머신러닝 및 딥러닝 알고리즘을 이용해 하이퍼파라미터 최적화 후 모델을 학습시키고 모형 간 예측력을 비교하였다. 추가 실험에서는 DNN의 node와 layer 수를 바꿔가면서 실험을 진행하여 가장 적절한 node와 layer 수를 찾고자 하였다. 결론적으로 가장 성능이 우수한 모델로 2021년의 아파트 매매실거래가격을 예측한 후 실제 2021년 데이터와 비교한 결과 훌륭한 성과를 보였다. 이를 통해 머신러닝과 딥러닝은 다양한 경제 상황 속에서 투자자들이 주택을 구매할 때 올바른 판단을 할 수 있도록 도움을 줄 수 있을 것이라 확신한다.

키워드 : 부동산, 회귀, DNN, XGBoost, CatBoost, 지도 시각화

1. 서 론

1.1 연구의 배경 및 목적

과거 서브프라임 모기지 사태 때 주택 가격이 급격하게 하락한 것과 달리 코로나 시대 이후 주택가격은 반대로 가파르게 오르고 있다. 이러한 집 값 상승은 전 세계적으로

발생하는 현상으로 파이낸셜타임스(FT)에 의하면 지난해 7월부터 올해 6월까지 55개국의 평균 집값이 9.2% 올랐다고 한다. 집 값 폭등의 원인은 '역사적으로 낮은 수준의 금리',

※ This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2022R1F1A1074696).
† 비 회 원 : 성균관대학교 전자전기공학부 학사과정
†† 비 회 원 : 한양대학교 기계공학과 학사과정
††† 종신회원 : 성균관대학교 글로벌융합학부 부교수
Manuscript Received : April 8, 2022
First Revision : June 7, 2022
Second Revision : July 15, 2022
Third Revision : August 29, 2022
Fourth Revision : October 12, 2022
Accepted : October 13, 2022
* Corresponding Author : Hayoung Oh(hyoh79@skku.edu)

'부족한 공급', '늘어난 재택근무' 등 다양하게 언급되고 있다.

이렇게 주택가격을 결정하고 변동을 발생시키는 다양한 요인들을 정확하게 이해하는 것은 중요하며 불확실한 미래에 대비할 수 있다. 본 연구의 목적은 최대한 많은 서울시 부동산 데이터를 모아서 데이터셋을 구축한 뒤 가장 성능이 좋은 모델을 만드는 것이다. 이는 불확실한 주택 시장에 향후 투자자들의 합리적인 의사결정을 도울 것이다.

1.2 선행 연구 검토

Nam[1]은 Linear Regression 기법을 활용하여 한국감정원과 국토교통부에서 제공하는 과거 부동산 매매 데이터를 학습해 미래 부동산 시세를 예측하는 프로그램을 개발하였다.

Hwang[2]은 서울시 공공데이터를 활용하여 주택 대출 금리, 자살율, 노인인구비율 등 다양한 외부 변수들을 활용해 지역별로 세분화하여 아파트 가격 지수를 머신러닝 알고리즘을 활용해 예측하였다.

Whieldon[3]은 미국 Maryland's Open Data Portal (ODP)에서 추출한 11,000개의 주택 매물 데이터를 이용해서 데이터 분석 및 Linear, Ridge, Lasso Regression를 사용해서 주택 가격을 예측하였다.

Jha[4]는 Florida's Volusia County Property Appraiser website에서 2015년부터 2019년까지의 부동산 데이터를 확보하였고 종속변수 그룹화를 하고 XGBoost, CatBoost, Random Forest 등 다양한 머신러닝 기법을 활용하여 주택 가격에 대한 예측 모델을 개발하였다.

Bae[5]는 DNN, LSTM과 기존의 시계열분석 방법인 ARIMA 모형을 이용하여 여러 가지 부동산 가격지수에 대한 예측을 시도하였으며 연구 결과 DNN의 예측력이 가장 우수하였다.

Bae[6]는 Support Vector Machine, Random Forest, Gradient Boosting Regression Tree, DNN, LSTM과 시계열분석 방법인 자기회귀이동평균모형, 벡터자기회귀모형, 베이지언 벡터자기회귀모형을 이용하여 아파트 매매실거래 가격지수를 예측하고 모형들의 예측력을 비교하였다. 연구 결과 머신러닝의 예측력이 시계열 분석 모형보다 우수하였으며 특히 외부 충격으로 시장이 급변하는 경우 시계열분석 방법은 시장 추세를 전혀 예측할 수 없었다.

Bae[7]는 공동주택 공시업무와 관련된 예산이 증가하고 있는 상황에서 다양한 기계 학습 방법들을 이용하여 공동주택 거래가격을 추정하여 모형들의 예측력을 비교하였으며 공동 주택공시가격 산정과 관련하여 기계 학습 방법의 활용 가능성을 검토하였다.

Lee[8]는 2006년 1월부터 2017년 10월까지의 서울 중대형 아파트의 월별 아파트 가격지수 수집하여 RNN 및 LSTM 모형을 이용하여 예측력 비교분석을 진행하였다.

1.3 선행 연구와의 차별성

본 연구의 차별성은 다음과 같다. 첫째 80만개 이상의 데이터셋을 구축했다. 부동산 가격은 다양한 변수들로 인하여 정해지는 복잡한 문제이다. 그러나 선행연구들은 5,000개 이하의 데이터를 사용해서 모델을 만들어서 사회의 복잡성을 설명하기에 부족했다. 반면 본 논문은 충분히 많은 데이터를 사용했다. 둘째, 합리적으로 변수를 가공하고 최적의 변수를 선택했다. 선행 연구들은 저자들이 임의로 독립변수를 정해서 학습을 했다. 반면 본 논문은 구할 수 있는 최대한 많은 변수들을 구한 뒤 다중공선성 문제를 해결하여 변수들을 합리적으로 가공했고 변수선택법(Feature Selection)이나 변수 추출법을 사용해서 최적의 변수를 선택하였다. 특히 변수를 수집하는 과정에서 선행 논문들에서는 볼 수 없었던 위도, 경도 변수들을 수집하여 부동산의 위치를 서초구, 강남구, 송파구(KBD), 여의도, 마포구(YBD), 종로구, 중구, 용산구(CBD)와 같이 광범위하게 표현할 뿐만 아니라 부동산의 위치를 정확한 좌표로도 구할 수 있었다. 셋째 본 논문의 모델은 선행 논문에 비하여 높은 수준의 예측력을 가진다. 우리는 방대한 데이터와 최적의 변수를 사용하였으며 다양한 범위의 하이퍼파라미터를 실험하였다.

1.4 연구 과정 도식화

본 연구는 총 5가지 과정을 거친다.

첫째, '국토교통부', '부동산 114', '네이트 부동산', '알동산', '통계청', '한국은행', 'e-나라지표'에서 데이터를 수집해서 2015년부터 2020년까지의 데이터셋을 구축한다. 둘째, 결측치 제거 및 다중공선성 해결 등 데이터 전처리를 한다. 셋째, Feature Selection, PCA, L1 Regularization을 통해

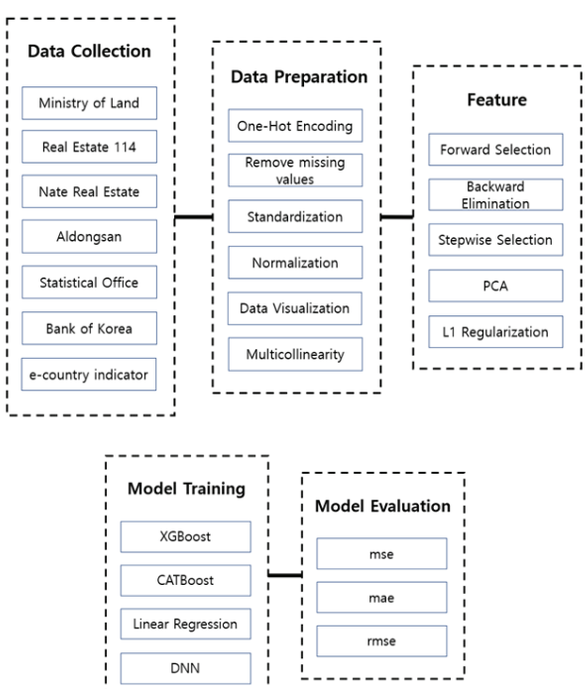

Fig. 1. Five Steps of the Thesis
Data Collection - Data Preparation - Feature - Model Training - Model Evaluation

의미있는 변수들을 각각 선정한다. 넷째, 선정된 변수들을 Linear Regression, XGBoost, CatBoost, DNN을 통해 학습시킨다. 다섯째, 학습시킨 모델들을 결정계수(R^2), 평균 제곱 오차(mse), 평균 절대 오차(mae), 평균 제곱근 편차(rmse) 4가지 평가지표를 평가하고 가장 성능이 좋은 모델로 2021년 아파트 매매 가격을 예측해보고 결과를 확인해본다.

2. 데이터셋 구축 및 전처리

2.1 데이터셋

데이터셋을 구축하기 위해 국토교통부, 통계청, 한국은행, e-나라지표, 부동산114, 네이트부동산, 알동산을 이용하였다.

국토교통부에서는 15년부터 20년까지 서울에 위치한 아파트 매매실거래가를 확보하였다. 통계청, 한국은행, e-나라지표에서는 년도별 경제지표를 얻었다. 부동산114와 네이트부동산에서 아파트 단지 정보를 얻었고 알동산에서는 아파트 단지별 위도와 경도를 수집하였다.

아파트 단지 정보를 얻는 과정에서는 셀레늄과 크롤링을 이용하였다. 부동산114 사이트에서는 url의 패턴을 파악해 법정구 코드를 리스트로 만들어서 구별로 위치한 아파트의 단지 정보를 추출할 준비를 마쳤다. 이후 크롤링을 통해 각 단지의 정보들을 뽑아서 딕셔너리의 형태로 저장하고 셀레늄을 통해 페이지 하단으로 스크롤한 뒤 다음 페이지로 넘기는 자동화 프로그램을 만들어 사용하였다. 네이트 부동산과 알동산도 비슷한 과정을 거쳐서 단지 정보들을 딕셔너리의 형태로 반환받았다.

2.2 데이터 전처리

매매가격 데이터(국토교통부), 경제지표(통계청, 한국은행, e-나라지표), 단지 데이터(부동산114, 네이트부동산, 알동산)를 모두 데이터프레임의 형태로 저장했다. 이후 수많은 독립변수를 변환, 병합, 제거를 통해 머신러닝과 딥러닝에서 사용할 수 있는 수치형 데이터로 변환하는 전처리 과정을 거쳤다. 예를 들어, '교통정보'는 수치형 데이터와 문자열이 같이 있어 수치형 데이터로 바꾸었고, '거래금액'이나 '국내총생산'과 같이 단위가 다른 변수들은 동일한 단위로 통일시켰다. 또한 '주차대수'는 '단순 주차대수'가 아닌 '가구당 주차대수'를, '건축년도'는 '(건축 후)경과일'로 바꾸었다. 마지막으로 '건설회사', '건축물 용도'와 같은 명목형 데이터는 원핫인코딩을 사용했고 '입주년도'나 '입주가능일'같이 필요 없는 데이터는 삭제했다.

데이터 결측치 탐색 결과 현관구조, 방향, 평수, 전용률, 방 수, 욕실 수에서 각각 1,117개, 교통정보에서 165,235개가 확인되었다. 총 데이터의 양과 비교했을 때 결측치가 적다고 판단되어 결측치가 있는 데이터는 모두 지웠다. 다음은 표준화 및 정규화 과정이다. 표준화를 통해 데이터의 평균은 0, 분산은 1로 만든 뒤 정규화를 통해 최솟값은 0, 최댓값은 1로 변환하였다. 일반적으로 정규화를 하기 전에 이상치 제거를 진행하지만 데이터셋이 실거래를 토대로 만들어진 데이터이므로 이상치 제거는 생략하였다.

완성된 데이터셋의 독립변수는 총 42개로 '가구 수', '주차대수', '단지 기본시설 수', '주변 교육시설 유무', '교통정보(가장 가까운 역까지의 거리)', '주변 생활편의시설 수', '전용률', '방 수', '욕실 수', '층', '아파트 경과일', '국내총생산', '회사채수익률', '금리', '환율', '통화량', '경기선행지수', '소비자물가지수', '실업률', '고용률', '종합주가지수', '위도', '경도', '지역더미 CBD', '지역더미 KBD', '지역더미 YBD', '지역더미 그 외', '건설회사 DL이앤씨(주)', '건설회사 GS건설(주)', '건설회사 삼성물산(주)', '건설회사 현대건설(주)', '건설회사 그 외', '난방방식 개별난방', '난방방식 중앙난방', '난방방식 지역난방', '난방방식 그 외', '정남향', '정북향', '방향 그 외', '계단식 현관구조', '복도식 현관구조', '복합식 현관구조'이고 종속변수는 '거래금액'을 '평수'로 나눈 '평당가격'이다. 데이터는 총 878,928개로 방대한 양의 데이터를 확보하였다.

2.3 데이터 시각화

회귀 분석 전에 데이터 시각화를 통해 데이터 분포 및 특징을 확인하여 pie chart를 통해 범주형 데이터의 분포를 확인하여 유사한 특징이면서 너무 적은 분포를 가진 데이터끼리 묶는 과정을 거쳤다. 따라서 Fig. 2와 같이 가장 큰 비율을 차지한 YBD, KBD, CBD와 그 외로 데이터를 정리하였고, 건설회사도 마찬가지로 Hyundai, Samsung, GS, DL을 제외한 데이터는 묶었다.

범주형 데이터는 pie chart(Fig. 2)를 통해 각각의 데이터가 어느 정도의 비율로 분포하는지 확인할 수 있다. Fig. 2에서 좌측은 지역 더미에 대한, 우측은 건설회사에 대한 pie chart이다.

숫자형 데이터는 box plot(Fig. 3)을 통해 확인하였다. 박스 내부의 가로선은 '중앙값'을, 맨 위의 선은 '중앙값 + 1.5 × IQR 보다 큰 데이터 중 가장 작은 값', 맨 아래의 선은 '중앙값 - 1.5 × IQR 중 가장 큰 값', 점은 '이상치'이다. 여기서 IQR은 '제3사분위수 - 제1사분위수'로 이상치를 탐색할 때 많이 쓰는 수치이다. Fig. 3의 좌측은 가구 수, 우측은 교통정보에 대한 box plot이다.

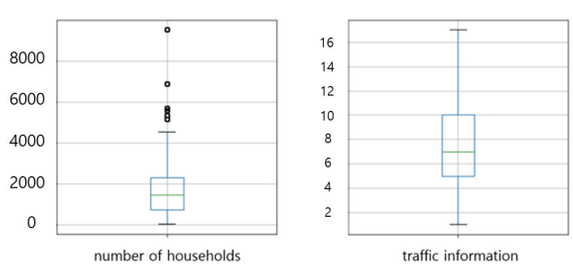

Fig. 2. Pie Chart
(Left : Local Pile, Right : Construction Company)

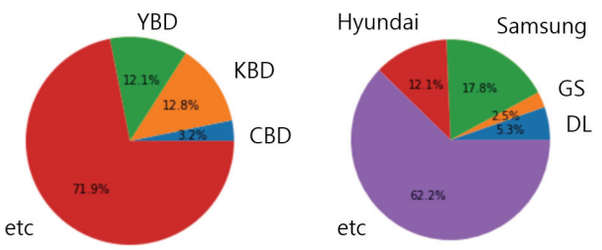

Fig. 3. Box Chart
(Left : Number of Households, Right : Traffic Information)

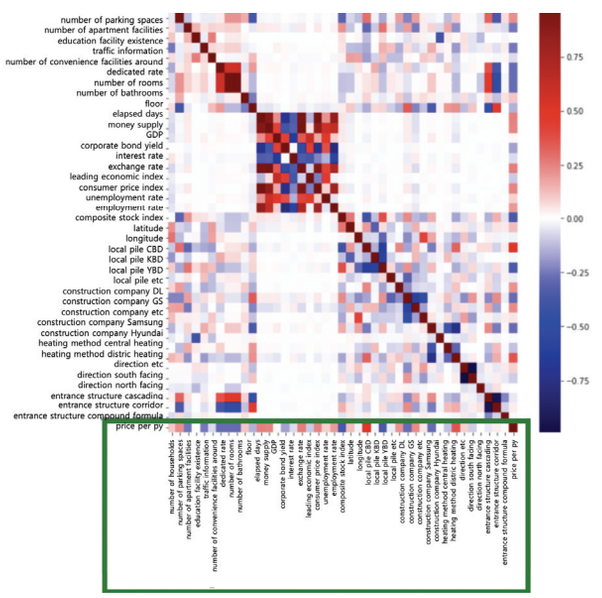

Fig. 4. Heat Map
Correlation Between Variables

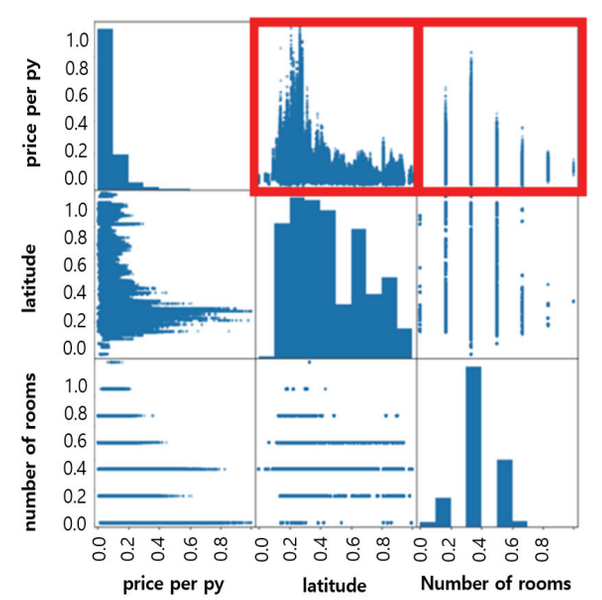

Fig. 5. Scatter Plot

Heat map(Fig. 4)과 Scatter plot(Fig. 5)을 통해 독립변수와 종속변수 간의 상관관계 정도를 확인할 수 있다. Fig. 4에서 '주차대수', '지역더미 KBD' 등의 변수들은 '평당 가격'과 양의 상관관계를 가지며 '교통정보', '아파트 경과일', '위도', '방 수' 등의 변수들은 음의 상관관계를 가졌다. Fig. 5에서는 '위도'와 '방 수' 모두 '평당 가격'과 음의 상관관계를 가지고 있는 것을 확인할 수 있다.

하지만 Fig. 4를 보면 일반적인 상식과 다르게 '단지 기본시설 수', '전용률', '방 수' 등이 '평당 가격'과 음의 상관관계를 가진다. 다음과 같은 결과는 다중공선성 문제로 인해 회귀결과가 왜곡되어 나왔음을 의미한다. 따라서 다중공선성 문제가 해결되어야 한다.

2.4 다중공선성 문제 해결

회귀 분석의 기본 전제는 독립변수간의 상관관계가 없어야 한다는 것이다. 다중공선성이란 하나의 독립변수가 다른 여러 개의 독립변수들로 예측되는 경우가 존재하여 데이터 분석에 부정적인 영향을 미치는 것이다. 즉, 다중공선성 문제는 회귀 분석의 기본 전제를 위반한다.

$$Y = \alpha + \beta_1 X_1 + \cdots + \beta_k X_k + \epsilon \quad (1)$$

$$Var(\hat{\beta}_j) = \frac{\sigma^2}{(1 - R_j^2)\sum (x_j - \overline{x})^2} \ (j = 1, 2, \cdots) \quad (2)$$

다중공선성 문제가 생길 경우, 독립변수 간의 결정계수인 R_j^2가 커지는데 이는 Equation (2)의 $Var(\hat{\beta}_j)$가 매우 커지게 만든다. 따라서 다중공선성이 존재하는 경우 추정량 $\hat{\beta}_j$가 매우 불안정하게 되어 Equation (1)에서의 β_j가 통계적으로 유의하지 않을 가능성이 높다.

다중공선성을 판단하는 지표로 Variance Inflation Factor(VIF)를 사용한다. VIF는 특정 독립변수를 나머지 독립변수로 적합했을 때의 성능을 나타내는 것으로 다른 변수에 의존할수록 값이 크게 나타난다. 정확한 VIF 분석을 위해서 bias 역할을 하는 const 항을 1로 해서 넣어주며 일반적으로 VIF값은 10을 기준으로 const항 제외 10 이상인 경우에는 다중공선성이 있다고 판단한다.

다중공선성을 해결하는 방법은 크게 다중공선성이 발생하는 변수 제거와 다중공선성이 발생하는 변수들을 결합하는 방법으로 나뉜다. Table 1은 42개의 각 독립변수별 VIF값을 표로 정리한 것으로 몇몇 독립변수들은 상당히 높은 VIF값을 보이고 있어 하나씩 해결하였다. 일단 방 수와 욕실 수에 대한 VIF값이 모두 높아 욕실 수를 제거했다. 경제 지표 변수들도 모두 VIF값이 높아 금리와 통화량을 제외한 모든 변수들을 제거했다. 범주형 변수들은 n개의 변수가 존재할 때, n-1개의 변수로 나머지 1개의 변수를 설명할 수 있으므로 n개의 변수를 모두 사용할 경우 다중공선성 문제가 발생한다. 따라서 범주형 변수에서도 변수 하나를 지우는 방식으로 다중공

Table 1. VIF Value Before Multicollinearity Resolution

	feature	vif factor
0	const	0
1	number of households	1.60671
2	number of parking spaces	1.49463
3	number of apartment facilities	1.34208
4	education facility existence	1.34007
5	traffic information	1.19418
6	number of convenience facilities around	1.3445
7	dedicated rate	2.17566
8	number of rooms	5.64801e07
9	number of bathrooms	5.64801e07
10	floor	1.10294
11	elapsed days	2.4897
12	money supply	2.76379e08
13	GDP	3.42709e08
14	corporate bond yield	1.08517e08
15	interest rate	8.55834e07
16	exchange rate	1.30405e08
17	leading economic index	3.16993e09
18	consumer price index	1.92641e09
19	unemployment rate	1.87903e09
20	employment rate	3.97855e08
21	composite stock index	8.36246e08
22	latitude	1.82523
23	longitude	1.80691
24	local pile CBD	3.01875e09
25	local pile KBD	1.23989e10
26	local pile YBD	1.4229e10
27	local pile etc	4.69166e07
28	construction company DL	6.10636e09
29	construction company GS	2.32534e10
30	construction company etc	1.40625e07
31	construction company Samsung	1.01602e10
32	consturction company Hyundai	1.74904e10
33	heating method individual heating	7.91829e09
34	heating method etc	1.78924e11
35	heating method central heating	1.91607e09
36	heating method district heating	2.78881e08
37	direction etc	2.62406e08
38	direction south facing	1.90031e08
39	direction north facing	2.68301e10
40	entrance structure cascading	9.74359e08
41	entrance structure corridor	2.79124e08
42	entrance structure copound formula	5.33905e10

Table 2. VIF Value After Multicollinearity Resolution

	feature	vif factor
0	const	171.605
1	number of households	1.30251
2	number of parking spaces	1.40753
3	number of apartment facilities	1.18719
4	education facility existence	1.14855
5	traffic information	1.13171
6	number of convenience facilities around	1.21192
7	dedicated rate	2.06504
8	number of rooms	1.82055
9	floor	1.08754
10	elapsed days	2.28491
11	money supply	4.7987
12	interest rate	5.3267
13	employment rate	2.7404
14	latitude	1.42608
15	longitude	1.31111
16	construction company top4	1.44854
17	heating method individual heating	1.30922
18	direction south facing	1.07309
19	direction noth facing	1.04101
20	entrance structure corridor	2.05438

선성을 해결했다. 즉, 방향에서는 '방향 그 외'를, 건설회사는 'DL', 'GS', '삼성물산', '현대건설'을 'top4'로 묶은 뒤 범주형 변수가 두 개이므로 나머지 하나인 '건설회사 그 외'를 제거하였고, 난방방식과 현관구조도 같은 방식으로 제거하였다. 가장 선호도가 높은 '개별 난방'을 제외한 나머지를 '난방 그 외'로 묶은 뒤 제거하였고, 현관구조는 가장 선호도가 낮은 '복도식'을 제외한 나머지를 '현관구조 그 외'로 묶은 뒤 제거하였다. 지역더미의 경우 위도, 경도와 관련성이 너무 높아 지역더미를 모두 지우고 더 정확하게 표현 가능한 위도 경도를 남기는 방식을 채택하였다. 언급한 모든 과정을 거친 뒤, 독립변수별 VIF값은 Table 2에서 확인할 수 있듯이 모든 값이 10 미만으로 다중공선성 문제가 해결되었다. 다중공선성 문제를 해결한 후 남은 독립변수의 개수는 20개로 '가구수', '주차대수', '단지 기본시설 수', '주변 교육시설 유무', '교통정보(가장 가까운 역까지의 거리)', '주변 생활편의시설수', '전용률', '방 수', '층', '아파트 경과일', '통화량', '금리', '고용률', '위도', '경도', '건설회사 top4', '난방방식 개별난방', '정남향', '정북향', '복도식 현관구조'이다.

Fig. 6은 statsmodel로 회귀 분석의 결과를 확인할 수 있는 표이다. 일반적으로 사회과학에서는 모든 변수를 모두 통제할 수 없다는 점에서 완벽한 연구가 불가능하다. 따라서 사회과학에서 모델의 설명력을 나타내는 R^2이 0.4 이상이면

OLS Regression Results

Dep. Variable:	price per py	R-squared:	0.505
Model:	OLS	Adj. R-squared:	0.505
Method:	Least Squares	F-statistic:	4.488e+04
Date:	Sun, 02 Jan 2022	Prob (F-statistic):	0.00
Time:	12:23:19	Log-Likelihood:	1.3945e+06
No. Observations:	878928	AIC:	-2.789e+06
Df Residuals:	878907	BIC:	-2.789e+06
Df Model:	20		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.1667	0.001	240.883	0.000	0.165	0.168
number of households	-0.0072	0.000	-21.514	0.000	-0.008	-0.007
number of parking spaces	0.2765	0.003	105.231	0.000	0.271	0.282
number of apartment facilities	-0.0240	0.000	-114.282	0.000	-0.024	-0.024
education facility existence	0.0293	0.000	61.942	0.000	0.028	0.030
traffic information	-0.0454	0.000	-171.079	0.000	-0.046	-0.045
number of convenience facilities around	-0.0132	0.000	-54.279	0.000	-0.014	-0.013
dedicated rate	-0.0517	0.001	-90.865	0.000	-0.053	-0.051
number of rooms	-0.1480	0.001	-247.346	0.000	-0.149	-0.147
floor	0.0139	0.001	22.973	0.000	0.013	0.015
elapsed days	-0.0635	0.000	-131.401	0.000	-0.064	-0.063
money supply	0.0725	0.000	190.782	0.000	0.072	0.073
interest rate	0.0092	0.000	22.655	0.000	0.008	0.010
employment rate	0.0011	0.000	4.822	0.000	0.001	0.002
latitude	-0.1040	0.000	-381.371	0.000	-0.105	-0.103
longitude	0.0539	0.000	228.583	0.000	0.053	0.054
construction company top4	0.0121	0.000	92.013	0.000	0.012	0.012
heating method individual heating	-0.0303	0.000	-250.901	0.000	-0.031	-0.030
direction south facing	-0.0042	0.000	-38.041	0.000	-0.004	-0.004
direction north facing	-0.0396	0.001	-35.568	0.000	-0.042	-0.037
entrance structure corridor	-0.0028	0.000	-16.068	0.000	-0.003	-0.002

Omnibus:	601821.995	Durbin-Watson:	0.434
Prob(Omnibus):	0.000	Jarque-Bera (JB):	17033962.706
Skew:	2.920	Prob(JB):	0.00
Kurtosis:	23.761	Cond. No.	113.

Fig. 6. Statsmodel
Regression Result Judgment Index

좋은 모델, 독립변수가 종속변수에 유의미하게 영향을 미치는지 확인할 수 있는 P>|t|(유의 확률)이 0.05보다 작으면 통계적으로 유의하다고 판단한다. 다중공선성 문제를 해결한 모델의 은 0.505, 모든 독립변수의 P>|t|는 모두 0에 수렴하므로 유의미한 결과가 나왔음을 확인할 수 있다.

3. 변수 선정

3.1 Feature Selection

과접합을 막기 위해 사용하는 Feature Selection에는 Forward Selection, Backward Selection, Stepwise Selection 3 종류가 있다. Fig. 7, 8, 9는 각각 Forward Selection, Backward Selection, Stepwise Selection을 한 결과이다.

Forward Selection이란 설명력이 높은 독립변수부터 순차적으로 모형에 추가하는 방식이며 유의미한 성능이 없을 때까지 이 과정을 실행한다. p-value가 가장 낮은 한 변수부터 시작해서 변수를 하나씩 더해 p-value가 0.00001보다 큰 변수가 나올 때까지 반복하였다. 각 단계마다 모델의 성능 비교는 조정된 결정계수(adjusted R^2)를 사용하였다. 총 14개의 단계를 거쳤으며 선택된 독립변수들은 '통화량', '위도', '경도', '가구 수', '방 수', '난방방식 개별난방', '아파트 경과일', '교통정보', '건설회사 top4', '단지 기본시설 수', '층', '정북향', '교육시설 유무', '주변 편의시설 수'로 총 14개다.

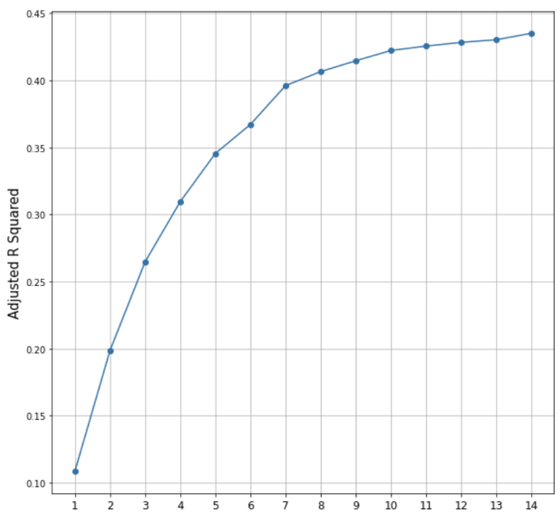

Fig. 7. Forward Selection
Adjusted R^2 of Determination
According to the Number of Variables

Backward Selection은 모든 독립변수를 이용한 후, 유의하지 않은 독립변수를 순차적으로 제외하는 방식이며 성능 저하가 나타날 때까지 이 과정을 반복한다. Full model에서 시작해서 p-value가 가장 높은 것부터 변수를 하나씩 지워나가며 남아있는 모든 변수의 p-value가 0.00001보다 낮아질 때까지 이 과정을 반복한다. 마찬가지로 각 단계마다 모델의 성능 비교는 adjusted R^2을 사용하였다. 제거된 독립변수는 Fig. 8에서 확인할 수 있듯이 6개이다. 나머지 14개의 변수는 '가구 수', '단지 기본시설', '교육시설 유무', '교통정보', '주변 편의시설 수', '방 수', '층', '아파트 경과일', '통화량', '위도', '경도', '건설회사 top4', '난방방식 개별난방', '정북향'이고 이는 Forward Selection에서 선택된 변수들과 모두 일치한다.

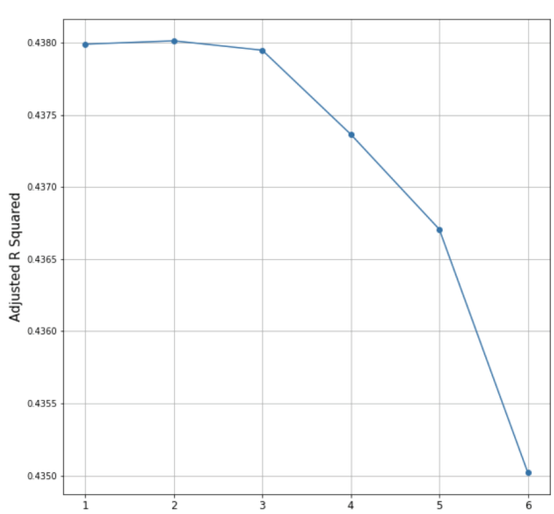

Fig. 8. Backward Elimination
Adjusted R^2 of Determination
According to the Number of Variables

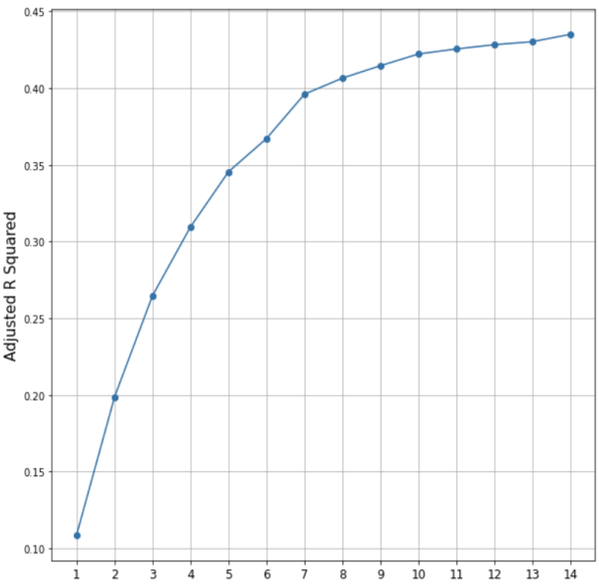

Fig. 9. Stepwise Selection
Adjusted R^2 of Determination
According to the Number of Variables

Stepwise Selection은 Forward Selection과 Backward Selection을 매 단계마다 번갈아가며 적용하는 방식으로 두 방식을 한 번씩 적용하여도 특성의 변화가 없을 때까지 반복한다. 마찬가지로 각 단계마다 모델의 성능 비교는 adjusted R^2을 사용하였다. Fig. 7과 9에서 알 수 있듯이 Stepwise Selection의 결과는 Forward Selection의 결과와 완벽히 일치한다. 따라서 '통화량', '위도', '경도', '가구 수', '방 수', '난방방식 개별난방', '아파트 경과일', '교통정보', '건설회사 top4', '단지 기본시설 수', '층', '정북향', '교육시설 유무', '주변 편의시설 수' 총 14개의 Forward Selection과 같은 독립변수가 선택되었다.

결론적으로 Forward Selection, Backward Selection, Stepwise Selection 모두 다른 과정을 거쳐 변수를 선정하였지만 동일한 14개의 독립변수를 선택하였다.

3.2 L1 Regularization

L1 Regularization은 과적합에 대한 솔루션으로 가장 먼저 떠오르는 해결책 중 하나이다. L1 Regularization은 기존의 loss function에 가중치의 절대값을 패널티로 추가해서 가중치를 0으로 만든다. 따라서 중요하지 않은 변수들은 loss function이 0이 되어 제거되므로 L1 Regularization은 변수 선택의 기능을 가진다. L1 Regularization의 결과 α=0.001일 때 mse=0.003(Fig. 10)으로 가장 좋은 결과가 나왔으며 이때 선택된 독립변수들은 총 9개로 '통화량', '경도', '건설회사 top4', '아파트 경과일', '전용률', '교통정보', '난방방식 개별난방', '방 수', '위도'이다. Fig. 11 선택된 독립변수들에 대한 가중치를 나타낸 것이다.

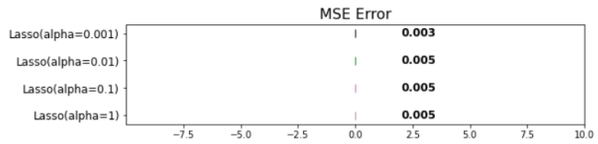

Fig. 10. L1 Regularization's α

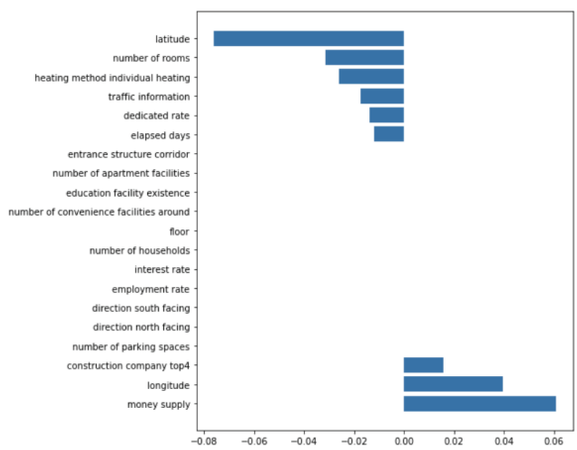

Fig. 11. Weight by Independent Variable

3.3 PCA

PCA는 대표적인 차원 축소 알고리즘이다. 큰 데이터의 차원(독립변수의 수)은 학습 속도를 느리게 할 뿐만 아니라 과적합의 위험이 커지므로 데이터의 차원을 축소하는 과정은 필수적이다. PCA의 원리는 학습 데이터셋에서 분산이 최대인 축을 찾고 앞선 축과 직교하면서 분산이 최대인 다음 축을 찾는 것이다. PCA는 이러한 방법으로 원하는 특성 수만큼 축을 찾아 그 축으로 데이터를 정사영 시키는 알고리즘이다. PCA를 주성분 분석이라고도 하는데 그 이유는 n번째 축을 정의하는 단위벡터를 n번째 주성분이라고 하기 때문이다.

PCA 분석 후, 기존의 독립변수들을 조합하여 만든 새로운 독립변수인 주성분의 수를 선정하기 위해서 Fig. 12와 같은 Scree Plot을 이용한다. 일반적으로 완만해지는 지점이 적절

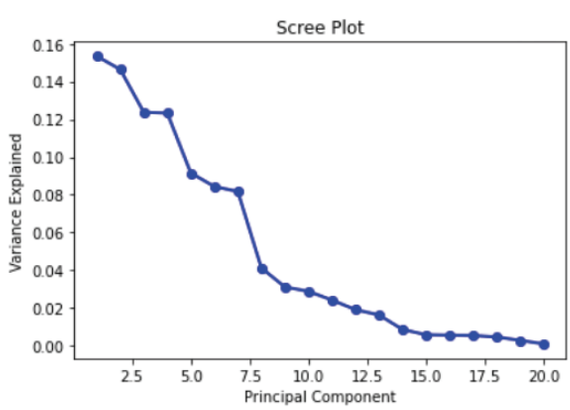

Fig. 12. Scree Plot
Eigenvalue-variance Change of Principal Component

한 주성분 개수를 나타내므로 Fig. 12에서 적절한 주성분 수는 15개라고 할 수 있다. 누적된 분산의 비율은 0.98로 원 데이터 분포의 98%를 보존한다는 의미이다. 이는 합리적인 분석에 적합한 주성분 개수 기준인 95%보다 크므로 15차원 데이터로 줄일 수 있다.

4. 분석 모형

4.1 DNN (심층신경망)

DNN은 다수의 hidden layer를 가지고 학습하는 머신러닝의 한 방식으로 데이터가 입력값으로 들어가면 초기 가중치 값(랜덤)을 바탕으로 모델을 학습시킨다. DNN(Fig. 13)은 역전파를 통해 각 layer들의 가중치(W)를 조정해서 최적의 모델을 찾는다.

$$W_{ji}(t+1) = W_{ji}(t) - \frac{\partial E}{\partial W_{ji}} \times lr \tag{3}$$

$$\frac{\partial E}{\partial W_{ji}} = \sum_{l,r,k} \frac{\partial E}{\partial o_l} \frac{\partial o_l}{\partial \sigma(z_k)} \frac{\partial \sigma(z_k)}{\partial z_k} \frac{\partial z_k}{\partial W_{ji}} \tag{4}$$

Equation (3)은 경사하강법을 나타내는 식이다. 경사하강법이란 예측값과 실제값의 오차를 함수로 표현한 loss function(E)의 최소값을 찾아주는 방법이다. 경사하강법에서 loss function을 가중치로 직접 편미분한 값은 찾기 어려워 Equation (4)과 같이 chain rule을 통해 구한다.

4.2 Boosting

Boosting은 약한 학습기를 순차적으로 나열하여 앞의 학습기가 잘못 학습한 데이터를 다음번 학습기가 올바르게 학습하도록 가중치를 부여하면서 학습을 진행한다. 즉, 한 가지 학습기로 학습을 하면 학습이 편향될 가능성이 높기 때문에 여러 가지 약한 학습기를 사용하여 강한 학습기를 만들어내는 원리이다.

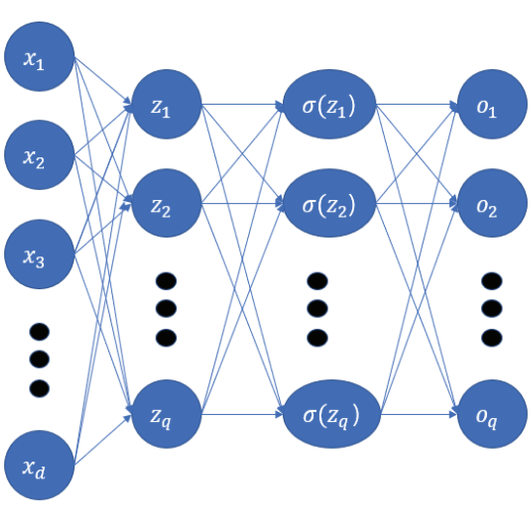

Fig. 13. DNN

1) Gradient Boost

Boosting은 분산-편차 트레이드오프 문제가 쉽게 발생한다. 편차는 모델의 예측값과 정답의 차이를 나타내고 분산은 예측값들이 서로 흩어진 정도를 나타내며 둘을 합쳐 loss라 한다. 분산-편차 트레이드오프 문제란 분산과 편향을 모두 최소화할 수 없다는 것이다.

Gradient Boost이란 Boosting 방식에 잔차를 추가한 방식으로 순차적으로 나열된 약한 학습기들로 잔차를 줄이면서 예측 모형을 만든다. Gradient Boost는 분산과 편차를 줄여 loss를 최소화할 수 있는 장점이 있지만, 과적합이 일어날 수 있다는 단점도 존재한다.

$$Loss = \frac{1}{2}(y_i - f(x_i))^2 \tag{5}$$

$$residual = \frac{\partial Loss}{\partial f(x_i)} \tag{6}$$

Equation (5)은 Gradient Boost의 loss값을 나타낸 것으로 mse와 동일함을 알 수 있다. Equation (6)은 Equation (5)에서 구한 loss값을 편미분 한 값이 잔차임을 나타낸다.

2) XGBoost

Gradient Boost 방식은 매우 높은 성능을 가지고 있지만 잔차를 계속 줄여가는 방식이므로 과적합이 발생할 가능성이 높다. XGBoost는 Gradient Boost에 잔차항을 추가해서 과적합 문제를 해결한다.

$$L = \sum_{i=1}^{n} l(y_i, \check{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{7}$$

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda W^2 \tag{8}$$

Equation (7)은 XGBoost의 loss값으로 첫째 항의 Gradient Boost의 loss 값과 둘째 항의 잔차항을 합한 것이다. Equation (8)은 Equation (7)의 잔차항을 나타내는 식으로 T는 각 약한 학습기의 node 수, W는 학습을 통해서 나오는 가중치를 의미한다. γ와 λ는 각각 T와 W에 패널티를 부여해 학습기가 복잡해지면서 과적합이 발생하는 것을 방지한다. 여기서 만약 γ, λ가 0이라면 기존의 Gradient Boosting과 동일해진다.

XGBoost를 학습할 때에는 '트리 최대 깊이', '각 트리의 깊이마다 사용할 컬럼 비율', '샘플링 비율', '학습률', '학습기 개수' 등의 하이퍼파라미터가 존재한다. '트리 최대 깊이'는 트리가 데이터를 최대 몇 번까지 데이터를 분리할 수 있는지를 의미하며 이 값이 클수록 데이터를 더 잘 분리하지만, 과적합이 일어나고 학습시간이 늘어난다. '각 트리의 깊이마다 사용할 컬럼 비율'은 각각의 트리의 깊이에서 컬럼의 개수를 얼마나 사용할지를 의미하며 과적합을 방지하기 위해 사용된다. '학습률'은 가중치를 업데이트 할 때 얼마나 많이 할 것인지를 의미한다. '학습률'은 클수록 학습 시간은 줄

어들지만 최적의 답을 못 찾을 가능성이 높아진다. '학습기 개수'가 클수록 더 큰 모델을 만들어서 더 많은 정보를 한번에 판단하고 처리할 수 있지만 과적합이 발생할 가능성이 높아진다. 'gamma'는 모델의 가지 수를 제어한다. 따라서 'gamma'가 클수록 모델은 가지를 만들지 않아 과적합이 발생하지 않는다.

3) CatBoost

XGBoost도 잔차를 줄여간다는 방식이 유지되므로 과적합 문제를 해결할 수는 없었다. 이러한 문제를 해결하고자 CatBoost는 Order Boosting 방식을 사용했다.

Fig. 14을 보면 CatBoost는 1번째부터 t-2번째 데이터를 가지고 모델을 생성한다. 이후 t-1번째 데이터로 모델의 성능을 평가하고 잔차를 계산하고 구한 잔차를 바탕으로 새로운 모델을 만든다. 새로운 모델의 성능은 t 번째 데이터로 평가한다. CatBoost는 이러한 과정을 지속적으로 반복한다. 기존 Boosting 모델은 모든 데이터를 대상으로 잔차를 계산했다면 Catboost는 일부 데이터를 가지고 모델을 만들고, 다음 데이터로 잔차를 구해서 모델의 성능을 향상시킨다. 또한 XGBoost와 달리 트리의 구조가 대칭적이여서 각 가지의 값을 이진화된 벡터로 저장할 수 있다는 장점을 가진다. 따라서 기존 Boosting 방식의 트리 구조에 비해 과적합을 방지할 수 있을 뿐만 아니라 메모리를 효율적으로 사용할 수 있다.

머신러닝을 학습할 때는 데이터의 독립변수 값이 동일한데 결과값이 다양할 경우, 그 결과값들의 평균으로 결과값을 바꾼다. 이러한 과정은 모델을 오직 훈련 데이터에만 적합하게 학습된다는 단점이 있다. CatBoost는 Target Encoding을 사용해서 이 문제를 해결한다. Target Encoding이란 전체 데이터의 평균으로 결과값을 바꾸는 것이 아니라 현재까지 학습한 데이터의 평균으로 결과값을 바꾸는 것이다. Target Encoding은 학습속도는 느리게 만들지만 효과적으로 과적합을 방지한다. 특히 Target Encoding은 범주형 데이터가 많을 때 좋은 성능을 보여준다. CatBoost의 하이퍼파라미터는 '트리 깊이', '학습률', 'L2 Regularization'이 있다.

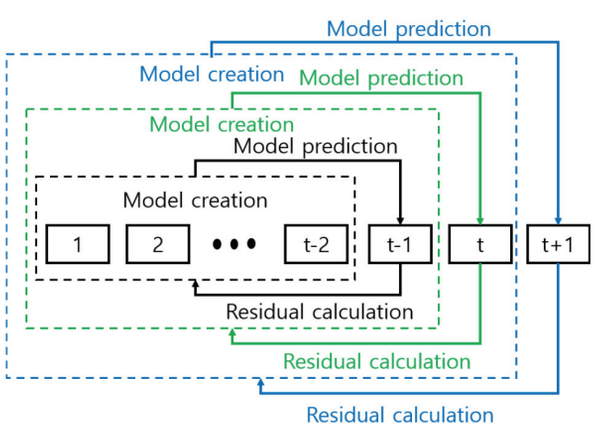

Fig. 14. CatBoost
How Catboost Works

4) Linear Regression

$$y = XB + \epsilon \tag{9}$$

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{pmatrix}, X = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ x_{n1} & \cdots & x_{np} \end{pmatrix}, B = \begin{pmatrix} B_1 \\ B_2 \\ \cdot \\ \cdot \\ \cdot \\ B_p \end{pmatrix}, \epsilon = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \cdot \\ \cdot \\ \cdot \\ \epsilon_n \end{pmatrix} \tag{10}$$

Linear Regression이란 독립변수와 종속변수의 관계를 선형으로 모델링하는 것이다. Equation (9)에서 y는 종속 변수, X는 독립변수, B는 회귀 계수, ϵ는 오차항을 의미한다. Equation (10)은 Equation (9)에 들어가는 변수를 행렬로 표현한 것이다. Linear Regression의 목표는 최적의 B와 ϵ를 찾아 오차가 가장 적은 Equation (9)을 찾는 것이다.

5. 모델 학습

다중공선성 문제를 해결해서 변수를 제거 한 후 또 다시 Forward Selection, Backward Selection, Stepwise Selection, PCA, L1 Regularization 총 5가지 방법으로 변수 선택 및 추출을 하여 변수의 개수를 줄였다(Table 3). 이를 통해 모델이 너무 많은 변수를 학습해서 과적합이 발생하게 되는 현상을 막도록 했다.

각 모델을 학습하기 전에 878928개의 데이터를 연도별로 나눈 뒤 각 연도별 데이터에서 위도와 경도가 동일한 데이터는 평당 가격이 유사하므로 하나의 데이터로 만들었다. 다음과 같은 작업을 함으로써 모델의 예측력을 더 높일 수 있었다. 이후 학습 데이터와 테스트 데이터를 무작위 추출 후 7:3으로 분리하였다. 이후 각 모델에 넣어서 부동산 가격을 예측했다.

DNN의 activation function은 'relu', optimizer는 'adam', batch size는 128, 64, 32를 시도하였다. 하지만 batch size가 128, 64일 때 학습이 너무 불안정해서 32로 고정하였다. 이후 layer의 수와 각 layer의 node 수를 바꿔가면서 학습을 진행하였다.

본 논문은 XGBoost의 최적의 하이퍼파라미터를 찾기 위해 GridSearchCV 알고리즘을 사용했다. 'cv'를 10 즉 학습기 수를 10개로 고정한 상태에서 처음에 '트리 최대 깊이'는 3, 5, 7, 10, 20, '학습률'은 0.025, 0.05, 0.1, 0.2, 0.4, 0.6, 0.8, 'gamma'는 0, 0.1, 0.5, 1.0, '샘플링 비율'은 0.1, 0.5, 0.7, 1.0, '각 트리의 깊이마다 사용할 컬럼 비율'은 0.25, 0.5, 1.0을 사용했다. 학습 결과 '각 트리의 깊이마다 사용할 컬럼 비율'과 '샘플링 비율'이 클수록 그리고 'gamma'는 작을수록 더 좋은 결과가 나왔다. 따라서 'gamma'는 최소값인 0으로, '각 가지의 최소 가중치'와 '샘플링 비율'은 최대값인 1로 모두 고정하고 '트리 최대 깊이'

Table 3. Variable Selection Result

feature	variables	total
Forward Selection	'money supply','latitude','longitude','number of housegolds','number of rooms','heating method individual heating','elapsed days','traffic information','construction company top4','number of facilities','floor','direction notrth facing','educational facilities exitence','number of convenience facilites around'	14
Backward Elimination	'money supply','latitude','longitude','number of housegolds','number of rooms','heating method individual heating','elapsed days','traffic information','construction company top4','number of facilities','floor','direction notrth facing','educational facilities exitence','number of convenience facilites around'	14
Stepwise Selection	'money supply','latitude','longitude','number of housegolds','number of rooms','heating method individual heating','elapsed days','traffic information','construction company top4','number of facilities','floor','direction notrth facing','educational facilities exitence','number of convenience facilites around'	14
PCA	pca1 ~ pca15	15
L1 Regularizatio n	'money supply','latitude','longitude','construction company top4','elapsed days','dedicated rate','traffic information,'heating method individual heating','number of rooms'	19??

와 '학습률' 두 가지 하이퍼파라미터만 변형하면서 학습을 진행하였다.

CatBoost도 GridSearchCV 알고리즘을 사용하였다. 'cv'를 10으로 고정하고 '트리 최대 깊이'는 3, 5, 7, 10, 20, 'L2 Regularization'는 5, 10, 20, '학습률'은 0.025, 0.05, 0.1, 0.2, 0.4, 0.6, 0.8을 사용하였다.

5.1 Feature Selection

Feature Selection(Forward Selection, Backward Selection, Stepwise Selection)은 모두 14개의 동일한 독립변수를 선택하였다. 선택된 독립변수들은 '통화량', '위도', '경도', '가구 수', '방 수', '난방방식 개별난방', '아파트 경과일', '교통정보', '건설회사 top4', '단지 기본시설 수', '층', '정북향', '교육시설 유무', '주변 편의시설 수'이다. 5.1에서는 다음 14개의 독립변수들을 가지고 모델을 학습했다.

1) DNN

DNN 모델을 생성할 때에는 hidden layer의 개수는 1개, 2개, 3개, node 수는 32개, 64개, 128개의 조합으로 만들었다. 위의 Table 4는 모델별로 train 데이터를 학습 후 test 데이터로 평가한 값이다. DNN에서 layer수가 3개일 때 가장 좋은 성능을 보였다. 그리고 layer가 3개인 모델 중에서 첫 hidden layer의 node가 많고 마지막 hidden layer의 node가 작을 때 더 좋은 성능을 보였다. Feature Selection으로 얻은 독립변수들로 학습한 DNN 모델 중 가장 성능이 좋은 모델은 layer가 3개이고 각 layer 당 node 조합이 128-32-32일 때로 이때 mae는 0.0179, rmse는 0.0294이다. 이는 layer가 1개일 때 가장 좋았던 값보다 mae와 rmse가 각각 약 20%, 10% 향상된 값이다.

Table 4. Variable Selection Method & DNN Result

Total hidden layer	Hidden layer node	mae	rmse
1 layer	32	0.023879	0.036030
	64	0.024150	0.034863
	128	0.022389	0.032161
2 layers	32-32	0.023405	0.038736
	64-32	0.022264	0.035598
	64-64	0.021839	0.034775
	128-32	0.021076	0.033142
	128-64	0.022579	0.034766
	128-128	0.022927	0.038002
3 layers	32-32-32	0.020873	0.032942
	64-32-32	0.021078	0.031945
	64-64-32	0.020315	0.032610
	64-64-64	0.019422	0.030988
	128-32-32	0.017902	0.029445
	128-64-32	0.019621	0.032508
	128-64-64	0.018869	0.032047
	128-128-32	0.018385	0.030592
	128-128-64	0.018532	0.031668
	128-128-128	0.019312	0.033084

2) XGBoost & CatBoost & Linear Regression

XGBoost를 GridSearch한 결과 최적의 하이퍼파라미터는 '학습률' 0.4, '트리 최대 깊이' 20이다(Tabel 5). '트리 최대 깊이'보다 '학습률'의 영향력이 컸으며 '학습률'은 0.025부터 0.4까지 커질수록 성능이 좋아졌고 0.6부터는 떨어졌다.

CatBoost도 Gridsearch를 한 결과 '트리 최대 깊이' 7, 'L2 Regularization' 5, '학습률' 0.1이 최적의 하이퍼파라미터였다(Table 6). '트리 최대 깊이'는 7까지 커질수록 성능이 좋아졌지만 그 이후로는 오히려 떨어졌다. '학습률'도 마찬가지로 0.2까지는 성능이 좋았다가 0.4부터는 성능이 떨어졌다. 'L2 Regularization'는 5, 10일 때는 큰 차이가 없었지만 20일 때는 급격히 성능이 떨어졌다.

Table 5. Variable Selection Method & XGBoost Top5 Hyperparameter Performance

	Variable selection		mae	rmse
	XGBoost Hyperparameter			
	learning_rate	max_depth		
1	0.4	20	0.0191	0.0303
2	0.4	10	0.0192	0.0304
3	0.6	10	0.0195	0.0311
4	0.6	20	0.0197	0.0314
5	0.4	7	0.0198	0.0307

Table 6. Variable Selection Method & CatBoost Top5 Hyperparameter Performance

	PCA			mae	rmse
	CatBoost Hyperparameter				
	depth	L2_leaf_reg	learning_rate		
1	7	5	0.1	0.0178	0.0290
2	7	10	0.1	0.0180	0.0292
3	7	20	0.2	0.0181	0.0291
4	7	5	0.2	0.0181	0.0294
5	7	10	0.2	0.0183	0.0298

Table 7. Variable Selection Method & Fit Result by Model

	model	r2_score	mean_squared_error	mean_absolute_error	rmse
Variable selection	Linear Regression	-0.4224	0.0026	0.0332	0.0515
	XGBoost	0.7965	0.0008	0.0182	0.0291
	CatBoost	0.8464	0.0006	0.0155	0.0255

Feature Selection으로 선정된 14개의 변수들을 토대로 Linear Regression, XGBoost, CatBoost를 적합하였으며 XGBoost와 CatBoost는 앞에서 선정된 최적의 하이퍼파라미터를 적용하였다. 그 결과 CatBoost가 mae 0.0155, rmse 0.0255로 가장 좋은 성능을 보였고 Linear Regression은 mae 0.0332, rmse 0.0515로 성능이 좋지 않았다(Table 7). Fig. 15, Fig. 16, Fig. 17은 각 모델의 성능을 시각한 그래프다. 그래프를 확인해보면 Linear Regression은 XGBoost, CatBoost와 비교하였을 때 급격한 변화에 반응하지 못한다.

5.2 L1 Regularization

L1 Regularization은 총 9개의 독립변수를 선택하였다. 선택된 독립변수들은 '통화량', '경도', '건설회사 top4', '아파트 경과일', '전용률', '교통정보', '난방방식 개별난방', '방수', '위도'이다. 5.2에서는 다음 9개의 독립변수들을 가지고 모델을 학습했다.

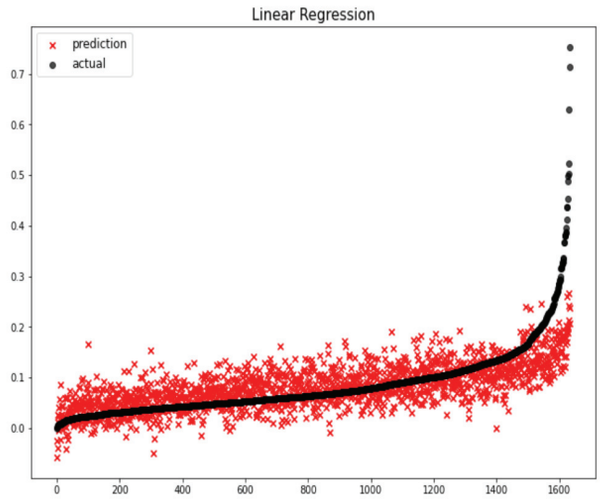

Fig. 15. Variable Selection Method & Linear Regression Plot

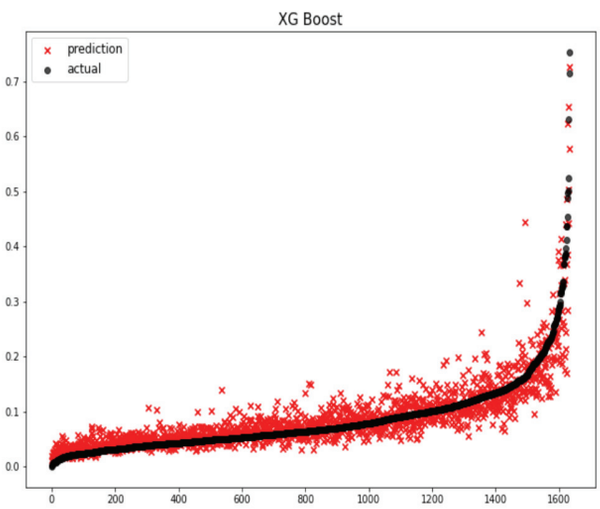

Fig. 16. Variable Selection Method & XGBoost Plot

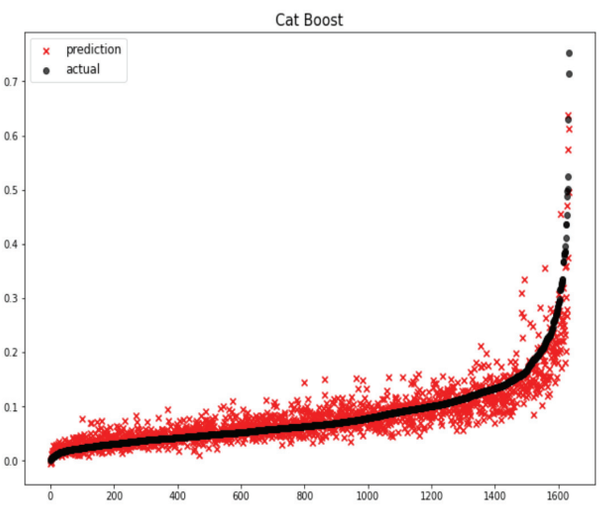

Fig. 17. Variable Selection Method & CatBoost Plot

Table 8. L1 Regularization & DNN

Total hidden layer	Hidden layer node	mae	rmse
1 layer	32	0.025712	0.042970
	64	0.024718	0.038688
	128	0.024781	0.035931
2 layers	32-32	0.025603	0.040184
	64-32	0.023505	0.035845
	64-64	0.023790	0.036236
	128-32	0.024657	0.036367
	128-64	0.023976	0.035872
	128-128	0.026494	0.042556
3 layers	32-32-32	0.022637	0.037110
	64-32-32	0.022820	0.036940
	64-64-32	0.022202	0.035752
	64-64-64	0.021751	0.036720
	128-32-32	0.020808	0.033928
	128-64-32	0.021711	0.034994
	128-64-64	0.021010	0.036734
	128-128-32	0.021444	0.038571
	128-128-64	0.020469	0.033863
	128-128-128	0.019849	0.036133

1) DNN

L1 Regularization을 통해 얻은 독립변수들로 DNN을 학습한 결과는 Feature Selection을 통해 얻은 독립변수들로 학습한 결과보다 좋지 않았다. 모델도 layer가 깊어질수록 더 좋은 성능을 보였고 hidden layer의 수가 3개면서 첫 layer의 수가 클수록 더 좋은 성능을 보였다. L1 Regularization을 통해 얻은 독립변수들로 학습한 DNN 모델 중 가장 성능이 좋았던 모델은 3 layer 중 128-128-128로 mae는 0.0198, rmse는 0.0361이다(Table 8).

2) XGBoost & CatBoost & Linear Regression

Table 9와 Table 10은 L1 Regulariztion을 사용했을 때 가장 좋은 성능을 보여준 하이퍼파라미터 조합 5개이다. XGBoost를 GridSearch한 결과 최적의 하이퍼파라미터는 '학습률' 0.4, '트리 최대 깊이' 20으로 Feature Selection에서의 결과와 동일하였다. 마찬가지로 '학습률'은 0.4, 0.6일 때 좋은 성능을 보였으며 너무 낮거나 높은 경우에는 성능이 떨어졌다. CatBoost를 Gridsearch한 결과 '트리 최대 깊이' 5, 'L2 Regularization' 5, '학습률' 0.2가 최적의 하이퍼파라미터였다. '트리 최대 깊이'는 Feature Selection에서와 다르게 5일 때, '학습률'은 0.2일 때가 가장 성능이 좋았다.

L1 Regularization으로 선정된 변수 9개를 토대로 Linear Regression, XGBoost, CatBoost를 적합하였으며 XGBoost와 CatBoost는 앞에서 선정된 최적의 하이퍼파라미터를 적용하였다. 그 결과 CatBoost가 mae 0.0157, rmse 0.0268로 가장 성능이 좋았고 Linear Regression은 mae 0.0334, rmse 0.0530로 성능이 좋지 않았다(Table 11).

Table 9. L1 Regularization & XGBoost Top5 Hyperparameter Performance

L1 Regularization	
XGBoost Hyperparameter	
learning_rate	max_depth
0.4	20
0.6	10
0.6	20
0.4	10
0.4	7

Table 10. L1 Regularization & CatBoost Top5 Hyperparameter Performance

L1 Regularization		
CatBoost Hyperparameter		
depth	L2_leaf_reg	learning_rate
5	5	0.2
5	10	0.2
7	10	0.2
7	5	0.2
7	5	0.1

Table 11. L1 Regularization & Fit Result by Model

	model	r2_score	mean_squared_error	mean_absolute_error	rmse
L1 Regularization	Linear Regression	-0.6698	0.0028	0.0334	0.0530
	XGBoost	0.7984	0.0008	0.0186	0.0296
	CatBoost	0.8273	0.0007	0.0157	0.0268

5.3 PCA

PCA는 총 15개의 독립변수를 추출하였다. 5.3에서는 다음 독립변수 15개를 가지고 모델을 학습했다.

1) DNN

PCA를 사용해서 선택된 15개의 독립변수를 가지고 학습을 했다. Table 12를 분석해보면 layer 수가 많을 때, hidden layer의 node 수가 클 때 성능이 높았다. hidden layer의 개수가 2개이면서 마지막 hidden layer의 개수가 적을 때 좋은 성능을 보였다. 또한 hidden layer 개수와 상관없이 첫 hidden layer의 node 수가 많을수록 좋은 성능을 보였다. PCA로 얻은 독립변수들로 학습한 DNN 모델 중 가장 성능이 좋은 모델은 3 layer 중 128-128-128로 mae는 0.0183, rmse는 0.0285이다.

2) XGBoost & CatBoost & Linear Regression

Table 13과 Table 14는 PCA를 사용했을 때 가장 좋은 결과를 보여주는 하이퍼 파라미터 조합 5개이다. XGBoost를 GridSearch한 결과 최적의 하이퍼파라미터는 '학습률' 0.4, '트리 최대 깊이' 10이였다. CatBoost를 Gridsearch한 결과 '트리 최대 깊이' 7, 'L2 Regularization' 5, '학습률' 0.1이 최적의 하이퍼파라미터였다. 하이퍼파라미터 성능 top 5의 '트리 최대 깊이'는 모두 7이였고 '학습률'은 0.1일 때가 가장 결과가 좋았다.

PCA로 선정된 변수 15개를 토대로 Linear Regression, XGBoost, CatBoost를 사용했으며 각 모델의 하이퍼파라

Table 12. PCA & DNN

Total hidden layer	Hidden layer node	mae	rmse
1 layer	32	0.024137	0.033693
	64	0.023786	0.033258
	128	0.023148	0.031761
2 layers	32-32	0.022474	0.031898
	64-32	0.022608	0.031492
	64-64	0.024179	0.033879
	128-32	0.020270	0.029140
	128-64	0.021261	0.029897
	128-128	0.023492	0.033251
3 layers	32-32-32	0.023052	0.034661
	64-32-32	0.020141	0.031623
	64-64-32	0.020365	0.029687
	64-64-64	0.020477	0.032148
	128-32-32	0.019895	0.029952
	128-64-32	0.019485	0.030074
	128-64-64	0.019616	0.029076
	128-128-32	0.018567	0.028794
	128-128-64	0.018649	0.028581
	128-128-128	0.018303	0.028503

Table 13. PCA & XGBoost Top5 Hyperparameter Performance

PCA	
XGBoost Hyperparameter	
learning_rate	max_depth
0.4	10
0.4	20
0.6	7
0.4	7
0.6	10

Table 14. PCA & CatBoost Top5 Hyperparameter Performance

PCA		
CatBoost Hyperparameter		
depth	L2_ leaf_reg	learning_ rate
7	5	0.1
7	10	0.1
7	20	0.2
7	5	0.2
7	10	0.2

Table 15. PCA & Fit Result by Model

	model	r2_score	mean_ squared_ error	mean_ absolute_ error	rmse
PCA	Linear Regression	-0.6059	0.0027	0.0338	0.0528
	XGBoost	0.5494	0.0014	0.0240	0.0375
	CatBoost	0.7453	0.0008	0.0184	0.0298

미터는 GridSearch 결과를 사용했다. CatBoost가 mae 0.0184, rmse 0.0298로 가장 좋은 성능을 보였으나 Linear Regression은 mae 0.0338, rmse 0.0528로 성능이 좋지 않았다(Table 15).

5.4 추가 실험

지금까지의 DNN 결과를 보면 layer 수가 많아질수록, hidden layer의 첫 node수가 클수록, 마지막 node 수가 작을수록 좋은 결과가 나왔다. 따라서 본 논문의 5.4에서 더 다양한 실험을 진행하고자 한다. 앞선 실험에서 가장 성능이 좋았던 모델과 동일하게 Feature Selection에서 선정된 독립변수들을 사용하였고 activation function은 'relu', optimizer는 'adam', batch size는 32를 적용하였다. 앞으로 진행될 추가 실험들은 위의 변수들과 조건들을 동일하게 유지한 상태에서 layer의 수와 layer별 node 수만 새롭게 변형하였다.

앞의 실험에서 3 layer일 때 성능이 좋았던 조합을 바탕으로 유사하게 4 layer 모델을 구성했다. Table 16과 같이 다양한 시도를 하였지만 앞선 실험에서 나온 결과인 3 layer의 128-32-32일 때가 가장 성능이 좋았다.

두 번째 실험은 layer 수를 6개 이상으로 만들었다. Table 17을 보면 layer 수가 늘어나면서 모델의 성능이 비슷하거나 낮아졌다. 특히 layer가 6개 이상일 때에는 hidden layer의 마지막 node의 개수가 8개일 때 가장 좋은 성능을 보였다.

세 번째 실험은 3 layer일 때 가장 성능이 좋으므로 3 layer로 고정한 상태에서 node의 수를 32개, 64개, 128개의 조합보다 더 다양하게 바꾸어 보았다. node 개수가 8개인 것부터 256개인 것까지 다양한 조합을 사용했다. Table 18에서 결과를 확인해보면 기존의 가장 성능이 좋았던 조합인 128-32-32에서 가장 좋은 결과 나왔다.

Table 16. Additional Experiment 1 - 4 Layer

Total hidden layer	Hidden layer node	mae	rmse
3 layers	128-32-32	0.017902	0.029445
	128-128-32	0.018385	0.030592
4 layers	128-32-32-32	0.023788	0.035477
	128-128-32-32	0.021641	0.032464
	128-128-128-32	0.024537	0.034465

Table 17. Additional Experiment 2 - Many Layer

Total hidden layer	Hidden layer node	mae	rmse
Many layers	32-32-32-8-8-8	0.025485	0.037570
	128-128-8-8-8-8	0.025029	0.034871
	128-128-128-8-8-8	0.021716	0.032492
	128-128-32-32-32	0.021059	0.032425
	128-128-128-32-32-32	0.027442	0.038417
	128-128-128-128-8-8	0.020669	0.031445
	128-128-128-128-32-32	0.021959	0.034212
	128-128-128-32-32-32-8-8-8	0.021300	0.032735
	128-128-128-32-32-32-8-8-8-2-2-2	0.023660	0.036862

Table 18. Additional Experiment 3 – Variable Number of Node

Total hidden layer	Hidden layer node	mae	rmse
2 layers	32-8	0.025062	0.037725
	128-8	0.021491	0.032510
	128-32	0.021076	0.033142
	256-8	0.021397	0.032252
	256-32	0.020904	0.031108
	256-128	0.019824	0.029743
3 layers	32-32-8	0.023586	0.034460
	128-32-8	0.022575	0.034400
	128-32-32	0.017902	0.029445
	128-128-8	0.022827	0.035499
	128-128-32	0.018385	0.030592
	256-8-8	0.020958	0.032229
	256-32-8	0.019933	0.029693
	256-32-32	0.021210	0.030324
	256-128-8	0.020503	0.030810
	256-128-32	0.020473	0.030973
	256-128-128	0.020547	0.030925
	256-256-8	0.021564	0.033102
	256-256-32	0.019766	0.031342
	256-256-128	0.021253	0.031285

본 논문은 총 86개의 DNN 모델을 가지고 모델을 학습했으며 3 layer의 128-32-32일 때 mae 0.017902, rmse 0.029445로 가장 좋은 성능을 보였다.

6. 모델 테스트

모델 학습에서 가장 예측력이 좋았던 모델은 CatBoost(트리 최대 깊이: 7, L2 Regularization: 5, 학습률: 0.1)이다. 모델 테스트에서는 다음 모델을 가지고 최신 데이터인 2021년 아파트 매매실거래가격을 예측해보고 실제 데이터와 비교해보았다.

Fig. 18은 2021년에 거래된 아파트 단지의 위도, 경도와 평당 가격을 나타낸 것이다. 평당 가격의 크기는 색깔로 확인 가능하며 파란색부터 빨간색까지 점점 커짐을 나타낸다.

모델의 예측 가격과 실제 가격의 비교는 평당 가격에 라벨(label)을 달고 진행하였다. 평당 가격이 5000만원 이하인 경우 '0~5000', 5000만원에서 1억 사이인 경우 '5000~10000', 1억에서 1억 5000만원 사이인 경우 '10000~15000', 1억 5000만원 이상인 경우 '15000~'로 하였다. Fig. 19는 실제 2021년도 아파트 매매실거래가격에 대한 도표이며, Fig. 20는 예측된 매매 실거래가격에 대한 도표이다. 두 도표를 비교해보면 모델이 평당 가격을 전체적으로 낮게 예측해서 5000만원 이하인 매물은 실제 데이터보다 많고 5000만원 이상은 실제 데이터보다 적다.

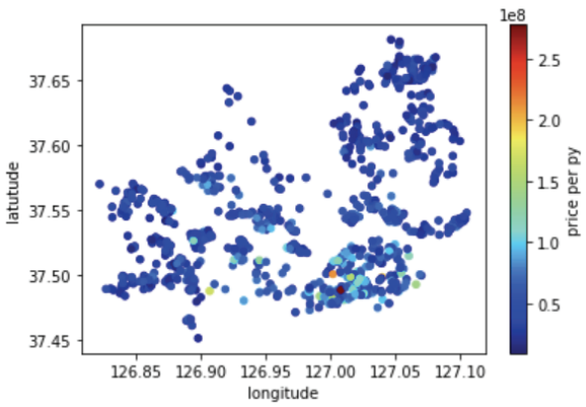

Fig. 18. Apartment Complexes Traded in 2021

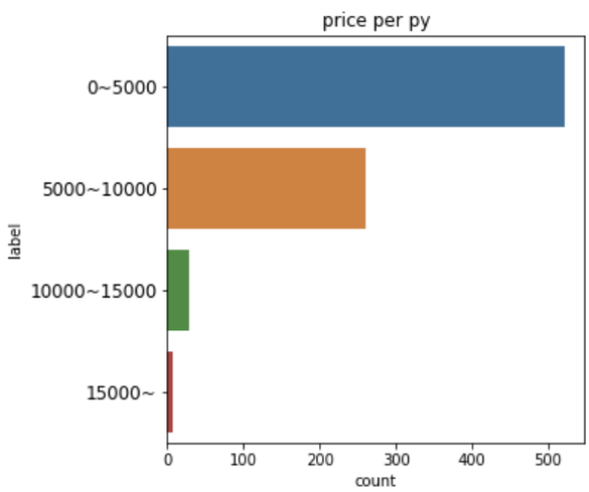

Fig. 19. Actual Price Per sqm of Apartments Traded in 2021

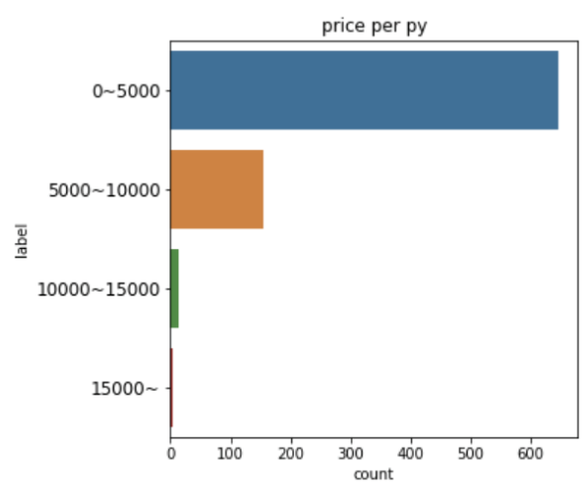

Fig. 20. Predict Price per sqm of Apartments Traded in 2021

Fig. 21은 실제 아파트 매매실거래가격이 5000만원 이하인 아파트 단지를 위도 경도를 토대로 지도에 나타낸 것이고 Fig. 22은 아파트 가격이 5000만원 이하로 예측된 아파트

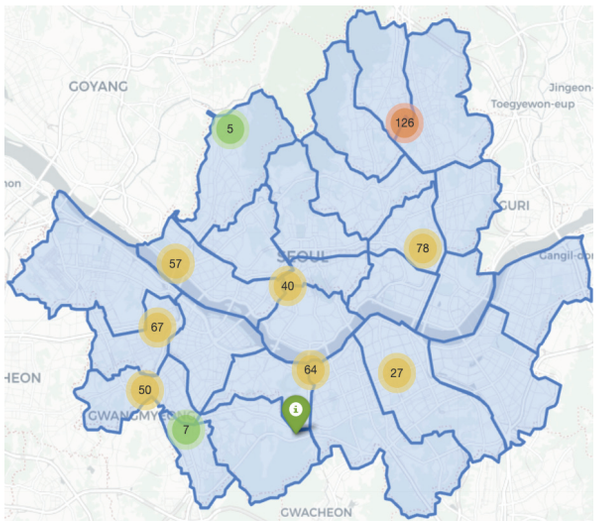

Fig. 21. Complexes with an Actual Price of 0 to 50 Million Won Per Pyeong

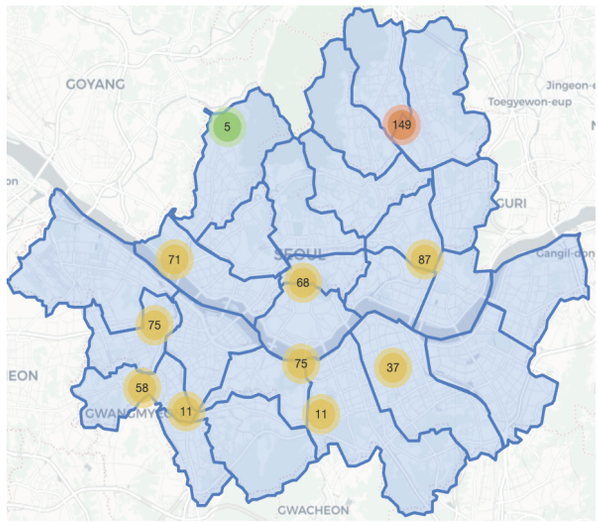

Fig. 22. Complexes with a Predicted Price of 0 to 50 Million Won Per Pyeong

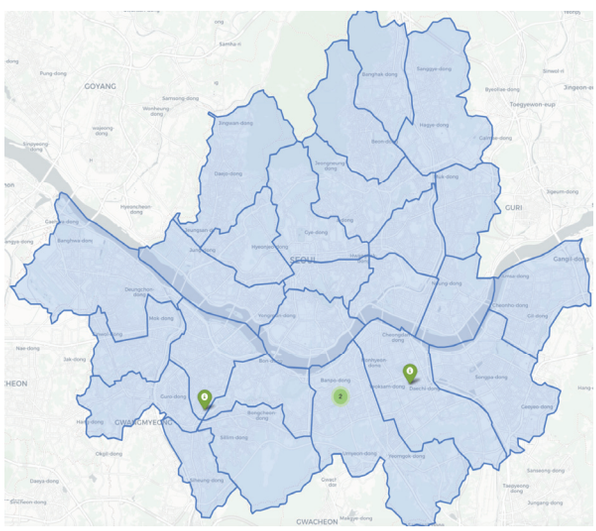

Fig. 23. Complexes with an Actual Price More Than 50 Million Won Per Pyeong

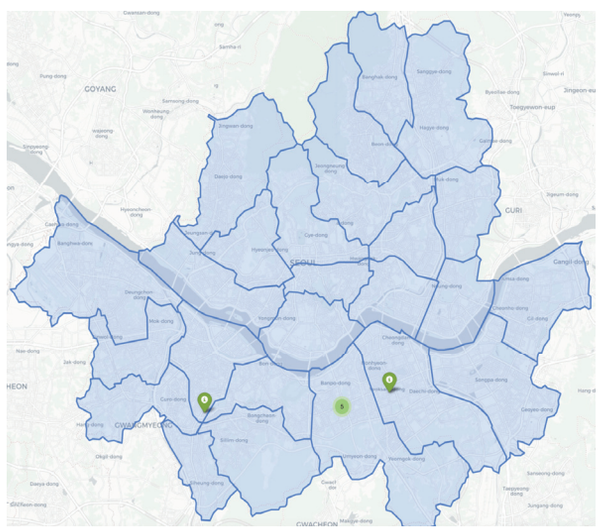

Fig. 24. Complexes with an Predicted Price More Than 50 Million Won Per Pyeong

단지를 지도에 나타낸 것이다. 앞서 언급했듯이 모델이 전체적으로 가격을 낮게 예측해 예측된 5000만원 이하의 아파트 단지가 실제보다 더 많음을 확인할 수 있다. 하지만 실제와 예측된 위도 경도 위치는 거의 같았다.

Fig. 23과 24은 아파트 매매실거래가격이 1억 5000만원 이상인 경우의 실제와 예측 아파트 단지를 나타낸 것이다. 결과를 보면 서초구에서 단지가 3개 적게 예측된 것을 제외하면 나머지는 동일하다.

7. 결 론

본 논문은 기존의 논문들과 달리 방대한 데이터셋을 구축하였다. 이후 다중공선성 문제를 해결 및 데이터 전처리를 하고 Forward Selection, Backward Selection, Stepwise Selection, L1 Regularization, PCA를 사용하여 합리적으로 독립변수들을 추출한 뒤 최신 머신러닝 기술인 XGBoost, CatBoost 모델을 만들어 성능을 평가했다. 또한 DNN(딥러닝) 모델을 다양한 layer와 node의 조합으로 학습하여 높은 성능의 모델을 만들었다. 본 논문은 높은 예측력을 가진 모델을 만들어 서울시 아파트 매매실거래가를 예측해보고자 했다는 점에서 의의가 있다.

본 연구를 요약하면 다음과 같다. 첫째, 우리는 나라의 부동산 데이터(국토교통부)에서 2015년부터 2020년까지 아파트 매매실거래 내역을 수집하고 '부동산 114', '네이트 부동산', '알동산'에서 서울시 아파트 단지 정보를 수집하였다. 또한 '통계청', '한국은행', 'e-나라지표'에서 제공되는 2015년부터

2020년까지의 경제지표 데이터를 수집하고 모두 합한 후 전처리를 진행하여 총 878,928개의 완성된 데이터셋을 구축하였다. 데이터셋의 독립변수는 총 42개로 '가구 수', '주차대수', '단지 기본시설 수', '주변 교육시설 유무', '교통정보(가장 가까운 역까지의 거리)', '주변 생활편의시설 수', '전용률', '방 수', '욕실 수', '층', '아파트 경과일', '국내총생산', '회사채수익률', '금리', '환율', '통화량', '경기선행지수', '소비자물가지수', '실업률', '고용률', '종합주가지수', '위도', '경도', '지역더미 CBD', '지역더미 KBD', '지역더미 YBD', '지역더미 그 외', '건설회사 DL이앤씨(주)', '건설회사 GS건설(주)', '건설회사 삼성물산(주)', '건설회사 현대건설(주)', '건설회사 그 외', '난방방식 개별난방', '난방방식 중앙난방', '난방방식 지역난방', '난방방식 그 외', '정남향', '정북향', '방향 그 외', '계단식 현관구조', '복도식 현관구조', '복합식 현관구조'이고 종속변수는 '거래금액'을 '평수'로 나눈 '평당 가격'이다.

둘째, 앞선 42개의 변수를 모두 이용하면 왜곡된 회귀 결과가 나오므로 VIF 값을 토대로 다중공선성 문제를 해결하였다. 다중공선성 문제를 해결한 후 남은 독립변수의 개수는 20개로 '가구 수', '주차대수', '단지 기본시설 수', '주변 교육시설 유무', '교통정보(가장 가까운 역까지의 거리)', '주변 생활편의시설 수', '전용률', '방 수', '층', '아파트 경과일', '통화량', '금리', '고용률', '위도', '경도', '건설회사 top4', '난방방식 개별난방', '정남향', '정북향', '복도식 현관구조'이다.

셋째, 수집한 독립변수들을 Forward Selection, Backward Selection, Stepwise Selection, L1 Regularization, PCA를 사용해서 의미있는 독립변수들을 뽑아냈다. 그 결과 Forward Selection, Backward Selection, Stepwise Selection에서는 모두 '통화량', '위도', '경도', '가구 수', '방 수', '난방방식 개별난방', '아파트 경과일', '교통정보', '건설회사 top4', '단지 기본시설 수', '층', '정북향', '교육시설 유무', '주변 편의시설 수' 총 14개의 같은 변수가 선택되었다. L1 Regularization에서는 '통화량', '경도', '건설회사 top4', '아파트 경과일', '전용률', '교통정보', '난방방식 개별난방', '방 수', '위도'로 총 9개에 변수가 선택되었고 PCA에서는 15개의 변수가 선택되었다.

넷째, 본 연구에서는 XGBoost, CatBoost, DNN, Linear Regression 총 4가지 알고리즘을 활용하였다. 또한 최적의 성능을 내기 위해 다양한 하이퍼파라미터와 변수 선정 방법들을 시도하였다. 평가모델은 mae와 rmse를 사용했으며 Table 19는 각 방식에서 나온 최상의 결과를 표로 정리한 것이다. XGBoost는 '학습률'은 0.0025부터 0.8까지, 'gamma'는 0부터 1까지, '샘플링 비율'은 0.1부터 1까지, '각 트리의 깊이마다 사용할 컬럼 비율'은 0.25부터 1까지의 조합을 GridsearchCV를 사용하여 최적의 하이퍼파라미터를 구했다. 그 결과 Feature Selection에서 추출한 변수들을 사용하고 하이퍼파라미터가 '트리 최대 깊이' 20, '학습률' 0.4, 'gamma' 0, '샘플링 비율' 1.0, '각 트리의 깊이마다 사용할 컬럼 비율' 1.0일 때 mae 값이 0.0182로 가장 좋은 결과를 보였다. CatBoost는 '트리 최대 깊이'가 2에서 8까지, '학습률'이 0.025에서 0.3까지, 'L2 Regularization'가 5에서 40까지 조합을 GridsearchCV를 통해 최적의 하이퍼파라미터를 구했다. 그 결과 Forward Selection에서 선택된 변수들을 사용하고 '트리 최대 깊이' 7, 'L2 Regularization' 5, '학습률' 0.1로 하이퍼파라미터를 결정할 때 mae 값이 0.0155로 가장 좋은 성능을 보였다. XGBoost와 CatBoost는 모두 좋은 성능을 보였지만 Linear Regression의 성능은 좋지 못했으며 성능을 그래프로 시각화해본 결과 다른 모델에 비해 급격한 변화에 반응하지 못하였다. DNN 기법도 매우 높은 성능을 보여준다. 본 논문에서는 성능을 높이기 위하여 hidden layer 수를 조절해 보았고 각 layer의 node 수를 다양하게 바꿔보았다. 그

Table 19. Summary of Learning Results Recording Only the Best Results for Each Learning

model	pick variable	hyperparameter	mae	rmse
DNN	Variable selection	hidden layer: 128-32-32	0.0179	0.0294
	L1 Regression	hidden layer: 128-32-32	0.0198	0.0361
	PCA	hidden layer: 128-32-32	0.0180	0.0285
XGBoost	Variable selection	learning rate: 0.4 max depth: 20 gamma: 0 sampling rate: 1 branch's minimum weight: 1	0.0182	0.0291
	L1 Regression	learning rate: 0.4 max depth: 20 gamma: 0 sampling rate: 1 branch's minimum weight: 1	0.0186	0.0296
	PCA	learning rate: 0.4 max depth: 10 gamma: 0 sampling rate: 1 branch's minimum weight: 1	0.0240	0.0375
CatBoost	Variable selection	max depth: 7 L2 regularization: 5 learning rate: 0.2	0.0155	0.0255
	L1 Regression	max depth: 5 L2 regularization: 5 learning rate: 0.2	0.0157	0.0268
	PCA	max depth: 7 L2 regularization: 5 learning rate: 0.2	0.0184	0.0298
Linear Regression	Variable selection	default	0.0332	0.0515
	L1 Regression		0.0334	0.0530
	PCA		0.0338	0.0528

결과 layer 수가 3개이고 hidden layer의 node 조합이 128-32-32일 때 mae값이 0.0179로 가장 좋은 성능을 보여줬으며 layer 수를 너무 늘렸을 경우, node 수가 너무 많을 경우, node 수가 너무 적을 경우 성능이 감소하는 현상이 나타났다. 학습한 모든 모델 중에 CatBoost 모델이 가장 높은 예측력을 보였으며 이는 데이터셋에 범주형 데이터가 많았기 때문으로 판단된다. 그러나 차후 연구가 발전도면 성능이 더 좋아질 여지가 보인다. 현재 layer의 수를 늘리기 위한 다양한 DNN 기법들이 연구되고 있으며 조만간 XGBoost나 CatBoost보다 더 좋은 성능을 가진 머신러닝 기법이 나타날 것으로 기대된다.

다섯째, 제일 성능이 좋았던 Feature Selection & CatBoost 모델로 2021년 아파트 매매실거래가격 예측을 진행하였다. 결과 분석을 위해 모든 데이터에 평당 가격이 5000만원 이하인 경우, 5000만원에서 1억 사이인 경우, 1억에서 1억 5000만원 사이인 경우, 1억 5000만원 이상인 경우 총 4가지 경우에 해당되는 label를 붙였다. 이후 예측 결과와 실제 결과를 도표와 지도를 통해 비교하였다. 분석 결과 모델이 전체적으로 가격을 낮게 측정하여 평당 가격이 5000만원 이하인 단지 수가 비교적 많았고 나머지는 실제보다 적었다. 하지만 예측된 결과를 위도, 경도를 통해 지도에 도시해보니 예측 단지 수에는 차이가 있었지만 실제 위치와 예측된 위치는 매우 유사하였다.

현재 부동산에 대한 국민들의 관심이 높아지고 있다. 이에 따라 부동산 투자에 관한 연구도 활발하게 진행되고 있다. 향후 부동산 업계에 아파트 가격예측은 매우 중요한 사안이 될 것이며 이에 따라 다양한 머신러닝 및 DNN 기법들이 그들에게 주요한 도구가 될 것이다.

References

[1] S.Nam, T. Han, L. Kim, and E. Lee, "Prediction of real estate price fluctuations in Korea using machine learning techniques," *The Journal of The Institute of Internet, Broadcasting and Communication*, Vol.20, No.6, pp.15-20, 2020. DOI: 10.7236/JIIBC.2020.20.6.15

[2] Y. Hwang, "A study on the calculation of the apartment price index: Focusing on machine learning algorithms," *Financial Research*, Vol.33, No.3, pp.51-83, 2019. DOI: 10.21023/JMF.33.3.3

[3] L. Whieldon and H. Ashqar, "Predicting residential property value in catonsville, maryland: A comparison of multiple regression techniques," *General Economics (econ.GN)*, 2020. DOI: 10.48550/arixiv.2101.01531

[4] S. B. Jha, R. F. Babiceanu, V. Pandey, and R. K. Jha, "Housing market prediction problem using different machine learning algorithms: A case study," Machine Learning (cs.LG), 2020.

[5] S. Bae and J. Yu, "Real estate price index prediction using deep learning," *Real Estate Research*, Vol.27, No.3, pp.71-86, 2017.

[6] S. Bae and J. Yu, "Real estate price index prediction using machine learning method and time series analysis model," *Housing Research*, Vol.26, No.1, pp.107-133, 2018. DOI: 10.24957/hsr.2018.26.107

[7] S. Bae and J. Yu, "Estimation of apartment house price using machine learning: Gangnam-gu, Seoul as an example," *Real Estate Research*, Vol.24, No.1, pp.69-85, 2018.

[8] T. Lee and M. Jeon, "A study on the prediction of the seoul housing price index using a deep learning model," *Housing and Urban Research*, Vol.8, No.2, pp.39-56, 2018. DOI: 10.26700/shuri.2018.08.2.39

[9] H. Chun, H. Yang, "A study on house price prediction using deep learning," *Housing and Urban Research*, Vol.8, No.2, pp.37-49, 2018. DOI :10.22313/reik.2019.17.2.37

[10] S. Lim, "Comparative study on the housing price index prediction model," *Journal of the Korean Data and Information Science Society*, Vol.25, No.1, pp.65-76, 2014. DOI: 10.7465/jk야.2014.25.1.65

[11] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: Unbiased boosting with categorical features," Machine Learning (cs.LG), 2017. DOI: 10.48550/arxiv.1706.09516

[12] G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree," *Advances in Neural Information Processing Systems*, 30, 2017. DOI: 10.5555/3294996.3295074

[13] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," Machine Learning (cs.LG), 2016, DOI: 10.48550/arxiv.1603.02754

[14] S. Kapoor and V. Perrone, "A simple and fast baseline for tuning large XGBoost models," Machine Learning (cs.LG), 2021. DOI: 10.4855-/arxiv.2111.06924

[15] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: A big comparison for NAS," Machine Learning (cs.LG), 2019, DOI : 10.48550/arxiv.1912.0659

[16] J. Shlens, "A tutorial on principal component analysis," Machine Learning (cs.LG), 2014, DOI: 10.48550/arxiv.1404.1100

[17] B. H. Park and J. K. Bae, "Using machine learning algorithms for housing price prediction: The case of Fairfax County, Virginia housing data," *Expert Systems with Applications*, Vol.42, No.6, pp.2928-2934, 2015. DOI: 10.1016/j.eswa.2014.11.040

[18] A. B. Khamis, and N. K. Kamarudin, "Comparative study on estimate house price using statistical and neural network model," *International Journal of Scientific & Technology Research*, Vol.3, No.12, pp.126-131, 2014.
[19] A. Gulli and S. Pal, "Deep learning with keras," Packt Publishing, Birmingham – Mumbai, 2017.
[20] A. S. Fotheringham, R. Crespo, and J. Yao, "Exploring, modelling and predicting spatiotemporal variations in house prices," *The Annals of Regional Science*, Vol.54, pp.417-436, 2015. DOI: 10.1007/s00168-015-0660-6

김 학 현
https://orcid.org/0000-0001-6846-3472
e-mail : hakhyun0615@naver.com
2019년 ~ 현 재 성균관대학교 전자전기공학부 학사과정
관심분야 : Data Science & Artificial Intelligence

유 환 규
https://orcid.org/0000-0001-8085-9453
e-mail : hkyoo52@naver.com
2019년 ~ 현 재 한양대학교 기계공학과 학사과정
관심분야 : Data Science & Artificial Intelligence

오 하 영
https://orcid.org/0000-0002-7362-5138
e-mail : hyoh79@skku.edu
2020년 ~ 현 재 성균관대학교 글로벌융합학부 부교수
관심분야 : Machine Learning & Deep Learning