**프로젝트 보고서**

**머신러닝 알고리즘을 통한 서울지역 아파트 실거래가 예측**



**제출일** 2020. 08. 26 (수)

**과목** 머신러닝과 딥러닝

**교수**  손영두

**학번/학과**  2015313693 통계학과

**이름**  권남택

**1. 문제 설명**

서울 지역의 아파트 실거래가를 예측하는 머신러닝 알고리즘을 구축하려고 한다. 한국인의 절반 가까이는 아파트에 거주하고 있으며, 한국에서 아파트는 인기있는 투자 자산이다. 그렇기 때문에 아파트 가격에 대한 관심이 높지만, 최근의 아파트 거래에 대한 정보 업데이트는 다소 느린 편이다. 물론 ‘직방’ 혹은 ‘다방’ 같은 부동산 거래 관련 어플리케이션이나, 네이버 부동산을 통해 최근 매물 가격 정보를 알 수 있으나, 다소 부정확한 정보일 수 있다. 따라서 주어진 데이터로 아파트 거래가를 예측하는 알고리즘을 만들어, 아파트 구매자들의 비대칭성을 해소하려 한다.실외, 건물, 기차, 앉아있는이(가) 표시된 사진

자동 생성된 설명

**2. 데이터 설명**

우리의 데이터는 '직방' 에서 제공한 데이터로, 한국의 Kaggle인 ‘Dacon’ 4회 컴피티션 ‘아파트 실거래가 예측’의 데이터이다. 이 데이터에는 우리가 예측하고자 하는 ‘y : 아파트 실거래가’ 와 아파트 거래일, 지역, 전용면적, 공급면적 등의 정보가 포함된다.

1,000,000 여 개의 train data와 500,000 여 개의 test data가 주어진다. 하지만 test data에는 아파트 실거래가가 포함되어 있지 않기 때문에, 우리 모델의 평가가 불가능하다. 따라서 train data를 분리해서 평가하려 한다. 아파트 거래일 기준으로 2008년부터 2017년까지 데이터가 존재하기 때문에, 2017년의 거래 데이터를 test set으로 분리해 우리의 모델을 평가할 것이다.

또한 우리의 데이터가 1,000,000 개의 관측치를 갖고 있기 때문에, 가벼운 데이터라고 보기는 어렵다. 따라서 이중 일부만을 사용해서 모델링에 걸리는 시간을 줄이려 한다. 단순 샘플링을 고려하기보다, 2012년부터의 데이터들을 선택 모델링을 진행한다. 그리고 데이터에는 서울/부산의 아파트 실거래가가 포함되어 있는데, 서울 지역의 아파트 거래만 고려해, 부산의 데이터는 모델링 과정에서 제외한다.

기존의 Dacon 컴피티션에 사용된 데이터와 다르게, 연습용으로 만들어진 데이터 셋이기 때문에, 분석 프로세스와 관련되어 참고할 자료는 없는 상황이다. 따라서 본 데이터로 진행되는 분석 과정은 표절과는 무관하다. 데이터를 가져온 링크는 다음과 같다.

https://dacon.io/competitions/official/21265/data/

**3. 데이터 확인**

**3-1. 변수 확인 및 제거**

처음 train-test split 이전에, 전체 데이터을 확인하고 모델링에 불필요한 변수를 제거하려하고 한다. 우리의 데이터는 1,216,553 행 13 열의 데이터이다. 변수는 ‘거래 id’, ‘아파트 id’, ‘도시’, ‘행정동’, ‘지번’, ‘아파트명’, ‘아파트 주소’, ‘전용면적’, ‘건축연도, ‘거래년월’, ‘거래일자’, ‘층 수’, ‘거래가격’ 이다. 이 중에 ‘거래 id’, ‘아파트 id’는 인덱스에 불과하고 다른 정보를 지니고 있지 않으므로 삭제한다. 또한 주소 정보를 지니는 ‘지번’, ‘아파트 주소’ 또한 아파트 거래가격에 영향을 미치기 때문에 있는 변수가 아닌, 이 변수를 기준으로 외부 데이터를 조인하기 위한 변수이므로 삭제한다. 또한 ‘도시’ 중에 서울시만 추출한다.

스크린샷, 실내, 컴퓨터, 노트북이(가) 표시된 사진

자동 생성된 설명

head()를 통해 호출한 결과

스크린샷이(가) 표시된 사진

자동 생성된 설명

**3-2. 데이터 정제 및 파생 변수**

‘거래년월’ 변수에서 ‘거래연도’와 ‘거래월’을 분리한다. ‘거래연도’는 우리가 생성할 파생변수와 매우 밀접한 관련을 갖고 있다. 만들어낼 파생 변수는 1) 거래당시 아파트 연차, 2) 시간에 대한 삼각변환이다.

‘아파트 연차’라는 파생 변수를 만들어내는 이유는 단순히 ‘아파트 건축연도’ 만을 고려하는 것보다, 거래 당시에 아파트가 얼마나 오래되었는지를 고려하는 것이 더 영향력 있는 변수일 것이라고 생각했기 때문이다. 아파트가 거래일자 기준으로 오래될수록 거래가는 낮을 것이다.

‘시간에 대한 삼각변환’을 고려하는 이유는 1년 안에서 아파트 거래가격의 주기성을 고려하기 위함이다. 1년 사이에 주기성을 갖고 있을 때, 단순히 거래 월을 ‘라벨 인코딩’을 하는 것은 11월과 1월의 가까움을 표현하지 못한다. 또한 ‘원핫 인코딩’을 시행할 경우 11개의 범주가 생성되어 모델링 속도와 분석에 어려움을 준다. 1년 내의 주기성을 고려한 삼각변환을 시행할 경우, 2개의 숫자형 변수로 표현할 수 있으며, 월 간의 거리를 반영할 수 있다. 이 두 파생변수를 만들어내고, 사용된 변수는 삭제한다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

**3-3. 인코딩 방법**

우리가 다루는 변수들 중 범주형 변수들이 존재한다. ‘행정동’, ‘아파트명’, ‘연도’ 이 세가지 변수는 범주형 데이터로 다뤄야 하는데, 이를 어떤식으로 인코딩하느냐는 모델링 과정의 단순화 및 모델 성능과도 연관성이 깊다. 따라서 변수에 적절한 인코딩 방법을 고려해야 하고, 가능하면 범주 개수를 제한해야 한다.

먼저 연도를 고려한다. 매년 지날수록 부동산 가격이 상승하는 것은 누구나 알고 있는 사실이다. ‘라벨 인코딩’을 통해 2012년을 0, 2017년을 5로 순차적으로 인코딩할 경우, 연도에 따른 물가 상승을 반영할 수 있다. 물론 ‘원핫 인코딩’을 진행할 경우에도 이런 선형적 증가 관계를 반영할 수 있지만, ‘라벨 인코딩’으로 대체함으로써 얻는 이득이 더 크다고 판단해서 ‘라벨 인코딩’을 진행하려 한다.

다음은 ‘행정동’, ‘아파트명’에 대한 인코딩이다. 이를 ‘원핫 인코딩’을 진행할 경우, 범주 개수가 너무 많아 6000여개의 변수가 추가된다. 근데 이는 희박한 열벡터를 많이 만들어내기 때문에, 모델링 속도에도 치명적이면서, 모델 성능에도 안좋은 영향을 미친다. 실제로 노트북에서는 램을 할당하기에 부족해 모델링 진행이 불가능하다. ‘라벨 인코딩’은 각각 아파트가 순서를 가지고 있다고 할 수 없으므로 고려하지 않는다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

그래서 이를 해결하기 위한 인코딩 방법으로 ‘Ordered Target Encoding(OTE)’ 방법을 선택한다. 해당 인코딩 방법은 ‘Catboost Encoding’ 방법이라고도 불리는데, 두 가지 특징이 있다. 먼저 해당 범주에 따라 y값의 평균을 넣어주게 됨에 따라 불필요하게 차원을 늘리지 않아 모델링의 성능과 속도를 올린다. 또한 단순히 y값의 평균을 넣어주는 ‘Mean Encoding’의 경우에는 자신의 값까지 고려해서 평균을 내기 때문에, ‘Data Leakage’ 가 발생해 과적합의 위험성이 있다. 하지만 OTE 방법은 n번째 행을 인코딩할 때 n-1번째 행까지의 결과만 고려하기 때문에 과적합의 위험성이 비교적 적다. 이런 방식은 데이터 분석 컴피티션에서도 많이 쓰이는 방법으로 알려져 있고, 성능면에서 매우 우수하다고 알려져 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

스크린샷, 사진, 컴퓨터, 목재의이(가) 표시된 사진

자동 생성된 설명

**3-4. 데이터 분할**

현재 우리 데이터는 train data만 존재하기 때문에 train-test 분할을 통해 test data를 생성해줘야 한다. 그런데 이때 random sampling을 통해 분할할 경우, 미래의 데이터를 학습해 과거를 예측하는 문제가 발생할 수 있다. 따라서 2017년의 데이터 전체를 test data로 취하고, 나머지를 train data로 취한다.

그런데 모델링 과정에서 최적의 hyper parameter를 찾기 위해서 cross-validation을 하기도 하지만, 현재 우리 데이터의 특성을 고려했을 때, 단순한 CV 방법은 부적절하다. 이를 위해서는 ‘롤링 윈도우’라는 기법을 사용해야 한다고 알려져 있다. 하지만 현재 데이터 사이즈가 상당히 크기 때문에, ‘롤링 윈도우’를 하지 않더라도 어느정도 무작위 오차가 상쇄될 것이다. 그래서 validation data로 2016년 데이터를 선택해서 모델링을 진행하려 한다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

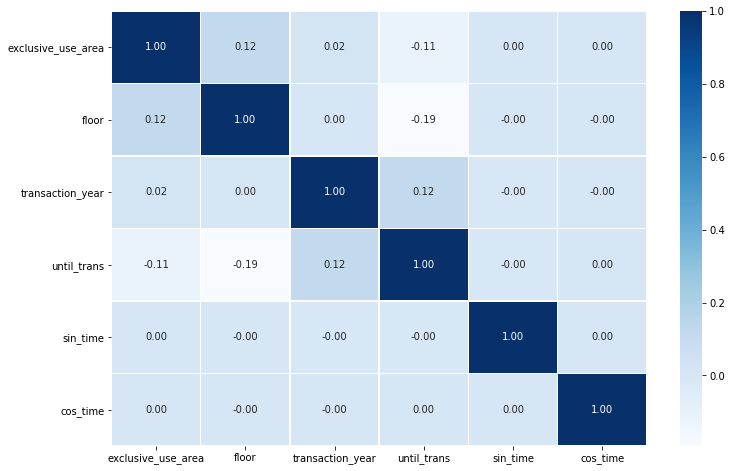
**4. 모델링**

우리의 모델링 회귀 모델링이다. 수업시간에 직접적으로 배운 회귀 모형은 선형회귀 모델뿐이지만, 분류 모델에서 다룬 Decision Tree, Bagging, Random Forest, KNN, Boosting 등등의 모델들은 모두 회귀모델링에도 적용 가능하다. 따라서 배운 모델들 안에서 모델링을 진행할 계획인데, 이중에 우리 데이터를 고려했을 때 선택하지 않을 모델들을 먼저 선정할 것이다.

우리의 데이터는 현재 500,000 여 개의 관측치를 지니고 있기 때문에, 적다고 말할 수는 없다. 만약에 Bagging이나 Random Forest 모델을 선택할 경우, Bootstrap sampling 하는 계산 비용이 너무 많고, 샘플링에 따른 데이터를 저장할 램(RAM) 용량을 로컬에서 확보하는 것은 매우 어렵다. 따라서 Bagging과 Random Forest는 선택하지 않는다. 또한 KNN 방법은 50만개의 데이터들 사이의 거리를 고려해야한다. 너무 계산량이 많을 뿐만 아니라, 6개의 변수에서 거리를 고려하는 것은 부적절한 방법일 수 있으므로 제외한다. 따라서 선형회귀 기반의 모델, Decision Tree, 그리고 AdaBoost 이 세가지 모델을 선택하고, Root Mean Square Error(RMSE)가 낮은 모델을 최종 모델로 선정하겠다.

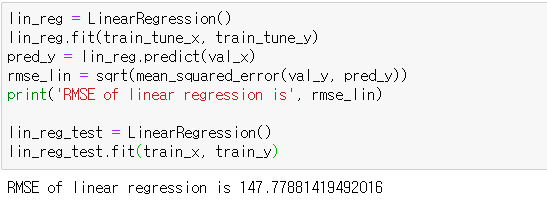
**4-1. 선형회귀 모형**

회귀문제를 푸는 데에 있어서 가장 간단하면서 해석력이 좋은 모델은 ‘선형회귀 모형’이다. 일반선형회귀(OLS)부터 변수선택을 고려한 선형회귀, penalty term을 고려한 선형회귀인 Ridge나 Lasso regression, 차원 축소를 고려하는 PCR 등이 존재한다. 일반선형회귀 이외에 언급한 모델들 모두 ‘다중공선성’을 해결하기 위한 방법들이라고 이해할 수 있다. ‘다중공선성’이란 X 변수들 간의 선형독립 가정이 심각하게 깨져, 변수들 간의 correlated 되어 있을 때 발생하는 문제이다. 다중공선성으로 인해 선형회귀식 계수의 분산이 증가해 추정과 예측 모두 불안정해지고, 결국 모델의 성능이 심각하게 하락한다. 이를 확인하는 방법은 분산팽창인자(VIF), 상태지수, 상관계수 행렬을 살펴보는 방법들이 있다. 데이터의 X값들이 correlated 되어 있는지 파악하는 빠른 방법은 상관계수 행렬을 히트맵(Heatmap)으로 시각화하는 것이다.



해당 상관계수 히트맵을 확인했을 때, 제일 높은 상관계수는 -0.19이다. 이는 X변수들이 uncorrelated 되었다고 보아도 괜찮은 수치이고, 따라서 다중공선성이 없다고 진단할 수 있다. 이런 경우 변수선택법, ridge나 lasso 추정, 차원축소 방법들은 모두 굳이 시행할 필요가 없다. 오히려 ridge나 lasso, 차원축소법은 모델의 성능 저하를 야기할 수 있다. 또한 현재 변수가 6개인데, 관측치는 400,000개를 넘기 때문에, p-value를 통해 변수를 선택하는 변수선택법의 경우 Full model이 제일 좋다고 할 가능성이 높다. 따라서 6개 변수를 모두 고려한 일반 선형회귀로 모델링을 먼저 진행한다.

일반 선형회귀 모델에서는 튜닝해야할 hyper parameter가 존재하지 않아서 validation data에 확인할 필요성이 적지만, 어느정도의 RMSE가 산출되는지 확인하고, 전체 train data에 적합한다.

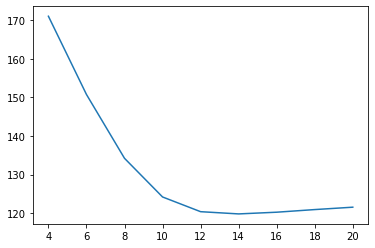


**4-2. Decision Tree**

일반 선형회귀를 시행했을 때, RMSE 값이 예상보다 되게 낮다. 해석력이 높은 선형모델의 성능이 괜찮으므로, 해석력이 높은 비선형 모델인 Decision Tree 모델을 선택한다. 모델의 복잡도를 많이 늘리지는 않지만, 비선형적 관계까지 모델링할 수 있는 트리 모델을 고려해, 점점 복잡도를 높일 것이다.

Decision Tree는 hyper parameter로 ‘max\_depth’가 있다. 깊이가 깊어질수록 복잡한 모델인 것이고, 모델이 복잡해질 수록 모델의 bias는 줄어들지만 variance는 높아진다. 하지만 우리의 데이터가 충분히 많기 때문에, 어느정도 깊은 트리가 적절할 것이다. for문을 통해 파라미터 튜닝을 진행하고, 이때 최저의 RMSE를 산출하는 max\_depth를 반환하는데, 이때 max\_depth는 14로 정해졌다. 이후 전체 train data에 적합함으로써, 모델을 만든다.

스크린샷이(가) 표시된 사진

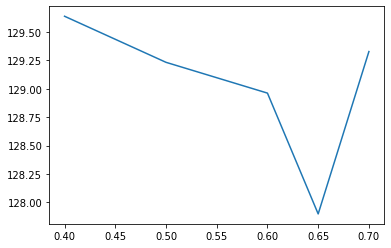
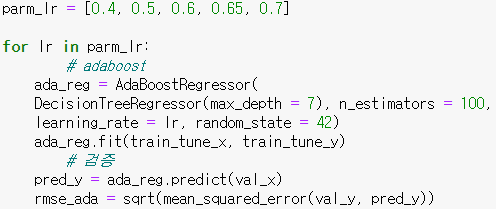
자동 생성된 설명

max\_depth에 따른 Validation RMSE

**4-3. AdaBoost**

부스팅은 weak learner의 집합을 이용해 strong learner를 만들어내는 알고리즘이다. Weak learner로 학습한 모델의 잔차를 통해 가중치를 수정해가며, 데이터를 제일 잘 설명하는 비선형 관계를 찾아내는 것이다. AdaBoost는 부스팅 중에서 가장 기본적인 부스팅 알고리즘이다. 부스팅 자체가 Decision Tree보다 복잡한 모델이기 때문에 모델의 bias는 낮고 variance는 높다. 만약 부스팅 파라미터를 적절하게 튜닝한다면, 관측치가 충분히 높기 때문에 제일 낮은 RMSE를 기대할 수 있다. 하지만 모델의 복잡도가 높은만큼, Decision Tree보다 파라미터 튜닝에 걸리는 시간이 더 많이 걸린다.

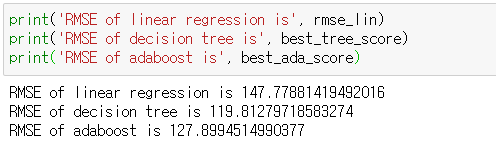
AdaBoost는 hyper parameter로 max\_depth, n\_estimator, learning\_rate가 있다. 이때 max\_depth는 weak learner를 만들어야하므로 낮은 수를 넣어준다. n\_estimator와 learning\_rate는 서로 상쇄되는 관계에 있다. 모델의 속도를 위해, 적당한 n\_estimator안에서 learning rate를 튜닝해주는 것이 적절하다.



learning\_rate에 따른 RMSE

**5. 결론**

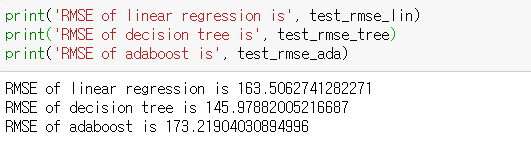
현재 일반선형회귀, Decision Tree, 그리고 AdaBoost의 파라미터 튜닝을 마치고, 전체 train data에 대해 적합을 했다. Test RMSE에 대한 추정치인 Validation RMSE는 다음과 같다. 트리모델의 RMSE가 제일 낮고, 다음으로 AdaBoost, 그리고 일반선형회귀 순이다. 이후 Test RMSE를 비교해서, Test RMSE가 제일 낮은 모델을 최종 모델로 선정하려 한다.



**5-1. 최종 모델링**

Train data에 모델들을 적합하고, Test data에 대해 평가한 결과는 다음과 같다. Test RMSE가 145.978로 제일 낮은 최종 모델은 Decision Tree 모델이다.

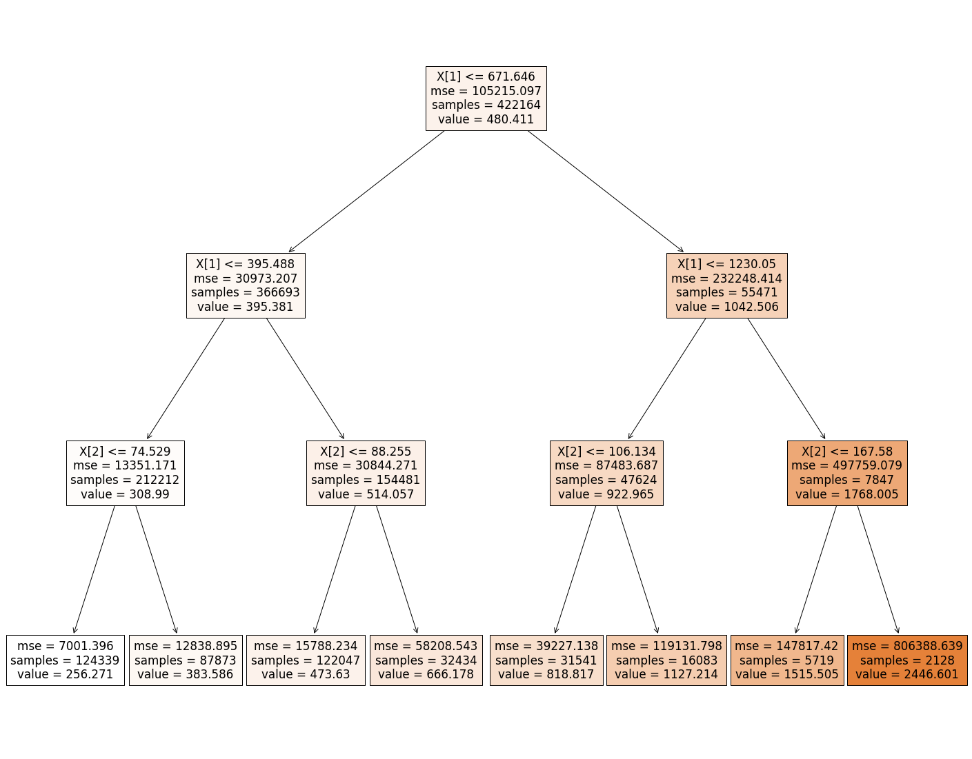
조류이(가) 표시된 사진

자동 생성된 설명

**5-2. 해석**

Validation data에 대해 평가하면서 과적합을 방지했지만, Test RMSE는 Validation RMSE에 비해 유의미하게 상승한 것으로 보인다. 특히 AdaBoost 알고리즘의 경우, Validation RMSE 기준으로는 일반 선형회귀 모델보다 RMSE가 낮았지만, Test RMSE는 더 높다. AdaBoost의 경우 모델 복잡도가 높기 때문에, 과적합의 위험성이 있는 모델이다. 충분한 관측치로 학습했고, 적절한 파라미터 튜닝 과정을 거쳤기 때문에 일정 수준의 성능을 기대했지만, 선형회귀보다 낮은 성능이 나왔다. 선형회귀의 경우 AdaBoost보다 훨씬 가볍고 개별 변수의 영향력에 대한 해석도 가능하다는 점을 생각했을 때, AdaBoost 모델은 제외해야 한다.

또한 트리모델의 경우에도 Test RMSE가 선형회귀보다 많이 증가했지만, Test RMSE 차이가 꽤나 나기 때문에 트리모델이 더 적합하다. 트리의 깊이(max\_depth)가 깊기 때문에, 일반 선형회귀보다 더 복잡한 모델이어서, 일부분 과적합이 발생했다고 볼 수 있다. 하지만 이를 상쇄하기에 충분한 데이터 양과, 더 나은 RMSE 기준의 성능, 그리고 해석 가능한 모델이라는 점에서 유용성이 매우 높다. 물론 트리의 깊이가 깊어짐에 따라 해석력이 감소하겠지만, 그래도 여전히 해석이 가능한 모델이다.



**5-3. 의의**

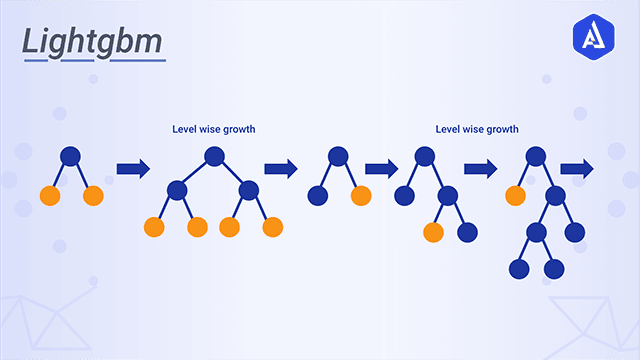
먼저 우리 모델은 해석가능한 모델이라는 점이다. 트리 모델의 깊이가 깊어 정확히 개별 결과값들에 대한 해석에는 한계가 있을지라도, 가장 뿌리노드부터 보았을 때 어떤 기준으로 트리가 분기하는지 알 수 있다. 따라서 가격을 결정하는데에 가장 중요한 영향을 미치는 변수를 알 수 있으면서, 좋은 성능을 지니는 알고리즘이다.

또한 다양한 방법들로 전처리를 시행해 좋은 성능을 이끌어 냈다. 시간에 대한 삼각변환을 취해줌으로써 주기성을 고려한 모델링을 만들었고, 범주의 레벨이 매우 높은 것을 고려해 ‘원핫 인코딩’이나 ‘라벨 인코딩’이 아닌 ‘Ordered Target Encoding’을 사용해 변수 개수를 6개로 유지했다. 이에 따라 변수들 간의 상관관계도 매우 낮은 변수들만이 남게 되었다. 이에 따라 일반 선형회귀도 부스팅 모델보다 더 나은 성능을 발휘할 정도로, 좋은 모델링을 진행했다.

**5-4. 한계**

Decision Tree 모델의 특성상 고도화가 어렵다. 이번 모델링을 시작으로 모델의 정확도를 더 높이는 과정들이 진행될 수 있는데, Decision Tree의 경우 트리 깊이만 튜닝하기 때문에 획기적인 성능 개선이 불가능하다. Decision Tree의 깊이가 깊어질수록 모델의 bias는 줄어들고 variance가 증가하는데, 관측치가 많아지면서 variance를 줄일 수 있다. 하지만 Variance를 더 통제할 수 있는 Random Forest나, Bias를 더 통제할 수 있는 Boosting모델과 비교했을 때 일반적인 성능 차이가 존재한다.

AdaBoost 자체 알고리즘이 느리고, 과적합에 대한 방지를 덜 고려한 알고리즘이기 때문에 모델의 한계가 존재한다. 최근 많이 쓰이는 XGBoost를 사용할 경우 더 좋은 성능을 기대할 수 있고, Light GBM과 같은 모델을 사용할 경우에는 많은 관측치에서도 가벼운 모델링과 좋은 성능을 기대할 수 있다. 또한 범주 레벨이 높은 경우 CatBoost와 같은 모델이 좋은 성능을 발휘한다고 알려져 있다. 만약 AdaBoost보다 발전된 알고리즘을 사용할 경우, 더 빠른 모델 학습속도에 더 좋은 성능을 발휘할 것이다.

그리기, 꽃이(가) 표시된 사진

자동 생성된 설명

**6. 부록 – 코드 첨부**

https://github.com/Namtaek/Namtaek/blob/master/project.ipynb