CodeStates Project 2

Ames Housing Prices Prediction

AI_05_최중훈

1. 데이터 선정이유 및 문제 정의

데이터 설명, 선정이유, 문제정의

Start here if...

You have some experience with R or Python and machine learning basics. This is a perfect competition for data science students who have completed an online course in machine learning and are looking to expand their skill set before trying a featured competition.

Competition Description

Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But this playground competition's dataset proves that much more influences price negotiations than the number of bedrooms or a white-picket fence.

With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, this competition challenges you to predict the final price of each home.

Practice Skills

- Creative feature engineering
- Advanced regression techniques like random forest and gradient boosting

Acknowledgments

The Ames Housing dataset was compiled by Dean De Cock for use in data science education. It's an incredible alternative for data scientists looking for a modernized and expanded version of the often cited Boston Housing dataset.

- Iowa주 Ames의 집값 데이터.
- Kaggle 입문자용 대회 사용.
- 다양한 Feature Engineering 시도.
- Ames의 집값을 예측하는 문제이므로 회귀문제로 정의.

2. 데이터를 이용한 가설 및 평가지표, 베이스라인 선택

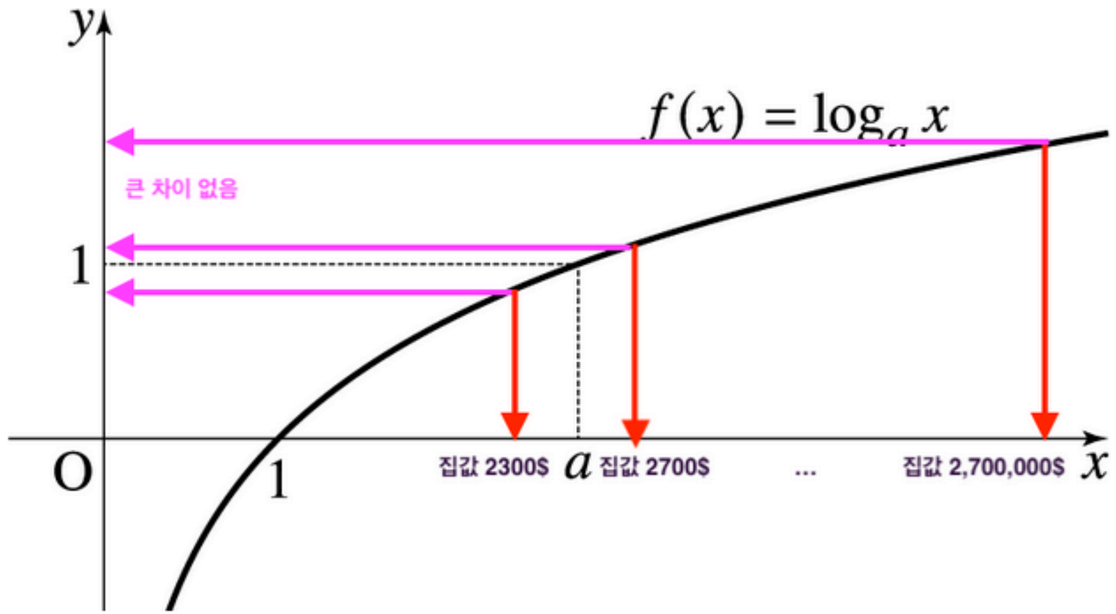
가설

- 건물이 얼마나 오래되었는지에 따라 집의 가격이 달라질 것이다.
- 리모델링한지 얼마나 오래되었는지에 따라 집의 가격이 달라질 것이다.
- 건물의 총 면적에 따라 집의 가격이 달라질 것이다.

2. 데이터를 이용한 가설 및 평가지표, 베이스라인 선택

평가지표

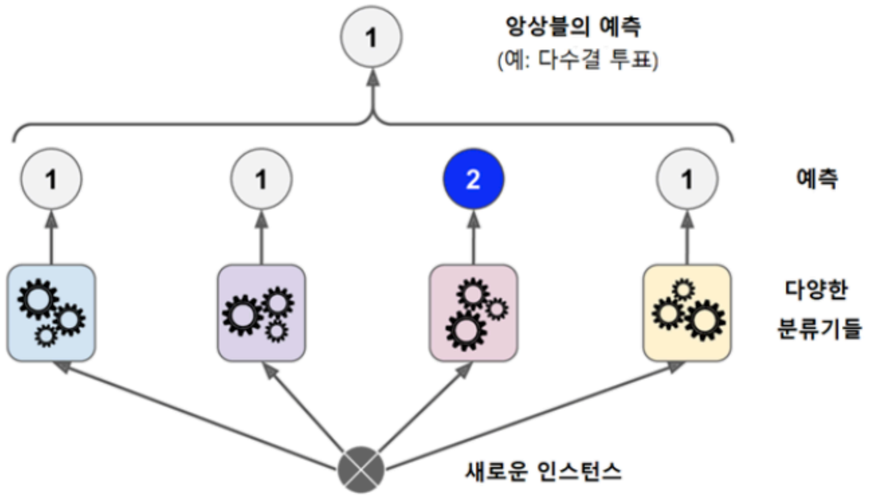
- 큰 오차값에 대해 영향을 덜 받는 RMSLE(Root-Mean-Squared-Log-Error)를 사용.
- RMSLE값이 작을수록 성능이 우수한 모델.



2. 데이터를 이용한 가설 및 평가지표, 베이스라인 선택

베이스라인 모델

- 기본적인 앙상블 모델 랜덤포레스트(Random Forest) 사용.



핸즈온 머신러닝

```
[42] # Baseline prediction

     pipe_base = make_pipeline(
         OrdinalEncoder(),
         RandomForestRegressor(random_state = 2)

     )

     pipe_base.fit(train_x, train_y)
     result = pipe_base.predict(valid_x)

     print('Baseline :',sklearn.metrics.mean_squared_log_error(result,valid_y)**0.5)

     Baseline : 0.1480684467825101
```

3. EDA / 전처리

불필요한 변수 제거

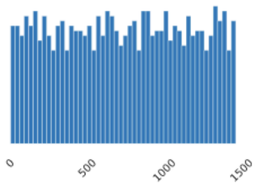
Id

Real number ($\mathbb{R}_{\geq 0}$)

`UNIQUE`

Distinct	1168
Distinct (%)	100.0%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	730.484589

Minimum	1
Maximum	1460
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	9.2 KiB



Toggle details

Utilities

Categorical

Distinct	2
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	9.2 KiB

Toggle details

Street

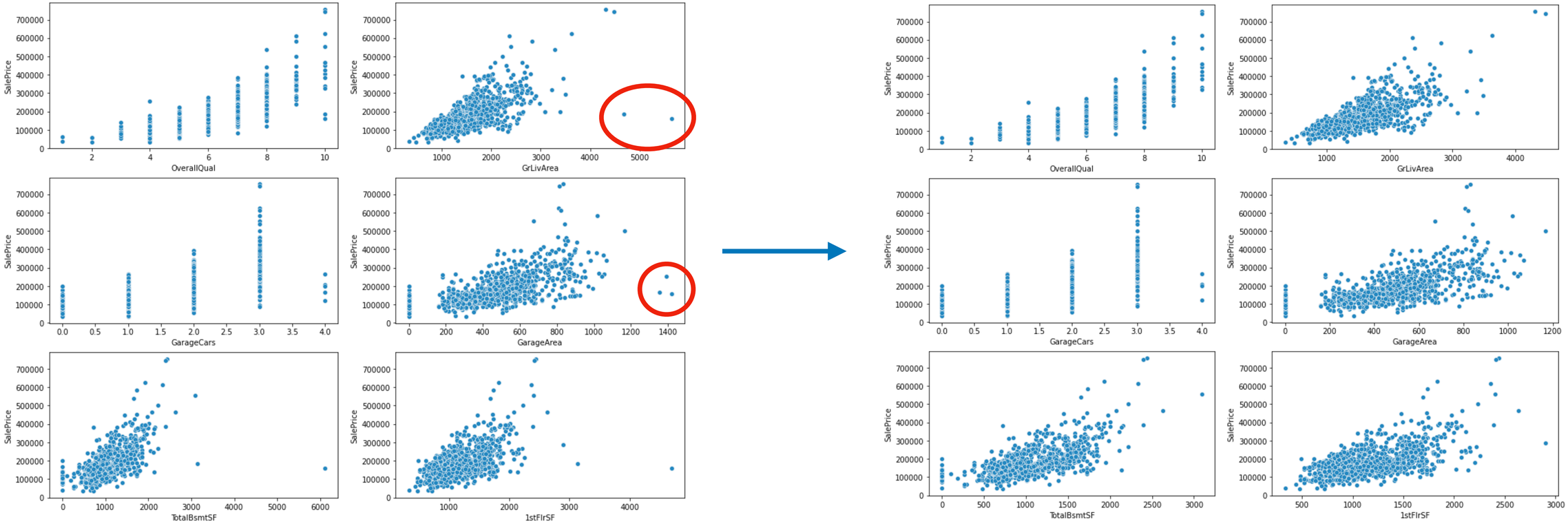
Categorical

Distinct	2
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	9.2 KiB

Toggle details

3. EDA / 전처리

이상치 제거



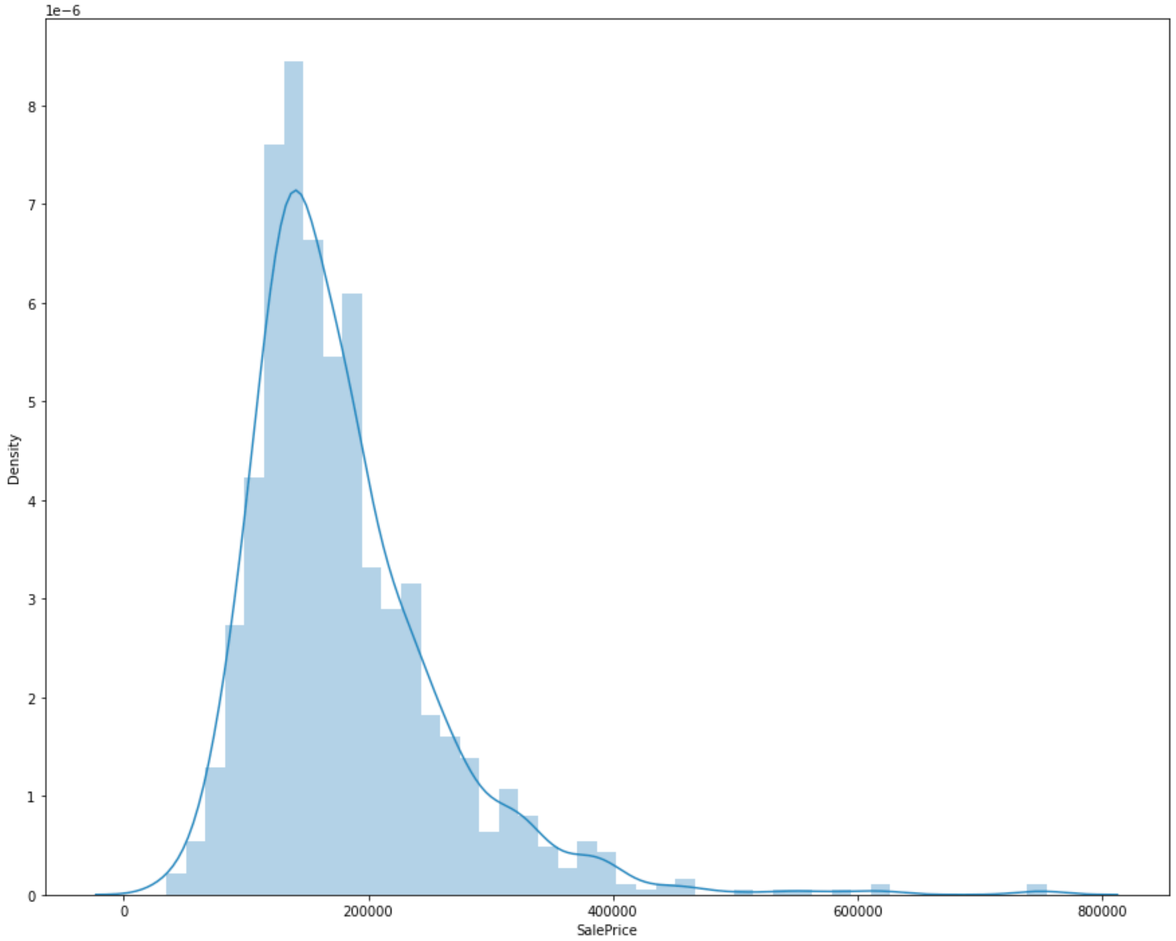
3. EDA / 전처리

결측값 처리

- 수치형 변수 : 집값의 편차가 큰 편이므로 극단치의 영향을 덜 받는 중앙값으로 대체
- 범주형 변수 : 범주형 변수의 분포에 영향을 덜 미치는 최빈값으로 대체

3. EDA / 전처리

로그변환



- 선형 회귀 모델을 사용할 때 예측하고자 하는 변수의 분포가 왼쪽으로 치우친 경우 로그 변환이 필요.
- 선형회귀 모델을 사용하지 않는 경우 생략

4. 머신러닝 방식 적용

모델 비교

```
[42] # Baseline prediction

     pipe_base = make_pipeline(
         OrdinalEncoder(),
         RandomForestRegressor(random_state = 2)

     )

     pipe_base.fit(train_x, train_y)
     result = pipe_base.predict(valid_x)

     print('Baseline :',sklearn.metrics.mean_squared_log_error(result,valid_y)**0.5)
```

```
     Baseline : 0.1480684467825101
```

```
[41] # XGBoost prediction

     pipe_xgb = make_pipeline(
         OrdinalEncoder(),
         XGBRegressor(verbosity = 0)

     )

     pipe_xgb.fit(train_x, train_y)
     result = pipe_xgb.predict(valid_x)

     print('XGBoost :',sklearn.metrics.mean_squared_log_error(result,valid_y)**0.5)
```

```
     XGBoost : 0.13462537294870838
```

```
[33] # LightGBM prediction

     pipe_lgbm = make_pipeline(
         OrdinalEncoder(),
         LGBMRegressor()

     )

     pipe_lgbm.fit(train_x, train_y)
     result = pipe_lgbm.predict(valid_x)

     print('LGBM :',sklearn.metrics.mean_squared_log_error(result,valid_y)**0.5)
```

```
     LGBM : 0.13963557307553534
```

```
[44] # Catboost prediction

     pipe_cat = make_pipeline(
         OrdinalEncoder(),
         CatBoostRegressor(verbose=0)

     )

     pipe_cat.fit(train_x, train_y)
     result = pipe_cat.predict(valid_x)

     print('catboost :',sklearn.metrics.mean_squared_log_error(result,valid_y)**0.5)
```

```
     catboost : 0.12839836933429205
```

4. 머신러닝 방식 적용

Hyper Parameter Tuning

```
[ ] randomized_search_result

dict_values([{'depth': 6, 'l2_leaf_reg': 3, 'learning_rate': 0.03}, defaultdict(<class 'list'>,
```

```
# Catboost prediction - RandomizedSearchCV

pipe_cat = make_pipeline(
    # OrdinalEncoder(),
    CatBoostRegressor(verbose=0, learning_rate = 0.03, depth=6, l2_leaf_reg = 3)
)

pipe_cat.fit(train_x, train_y)
result = pipe_cat.predict(valid_x)

print('catboost :',sklearn.metrics.mean_squared_log_error(result,valid_y)**0.5)
```

```
catboost : 0.4110031076860524
```

- 모델의 세부적인 설정값을 조정하기 위해 Hyper Parameter Tuning 진행
- 설정값의 범위를 잘못 지정하여 성능이 하락.

5. 머신러닝 모델 해석

변수 중요도 확인

Weight	Feature
0.0268 ± 0.0025	totalarea
0.0152 ± 0.0040	OverallQual
0.0052 ± 0.0010	GrLivArea
0.0030 ± 0.0008	OverallCond
0.0023 ± 0.0006	LotArea
0.0022 ± 0.0011	YearBuilt
0.0020 ± 0.0005	old
0.0020 ± 0.0009	GarageCars
0.0019 ± 0.0007	Neighborhood
0.0018 ± 0.0007	Fireplaces
0.0015 ± 0.0008	BsmtFinSF1
0.0014 ± 0.0004	2ndFlrSF
0.0012 ± 0.0003	GarageArea
0.0011 ± 0.0006	GarageFinish
0.0010 ± 0.0004	Functional

- 변수의 중요도를 알아보기 위해서 permutation Importance사용.
- 중요도 상위 15개변수중 가설을 세워 만든 변수가 2개 포함되어있음.