Sberbank Russian Housing Market

- Price Prediction

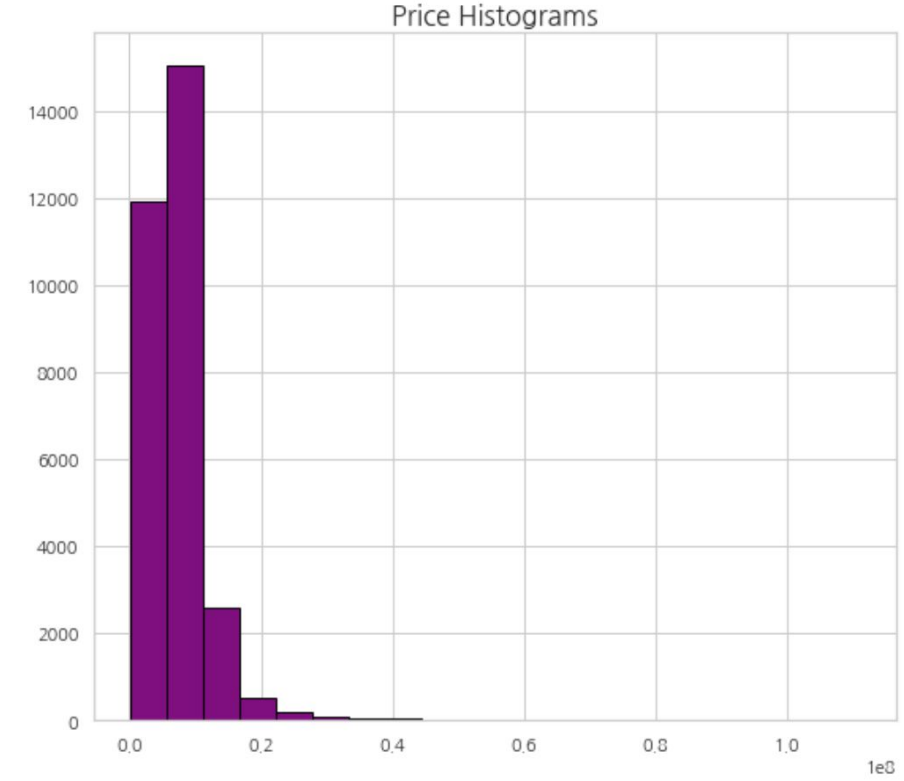
Contents

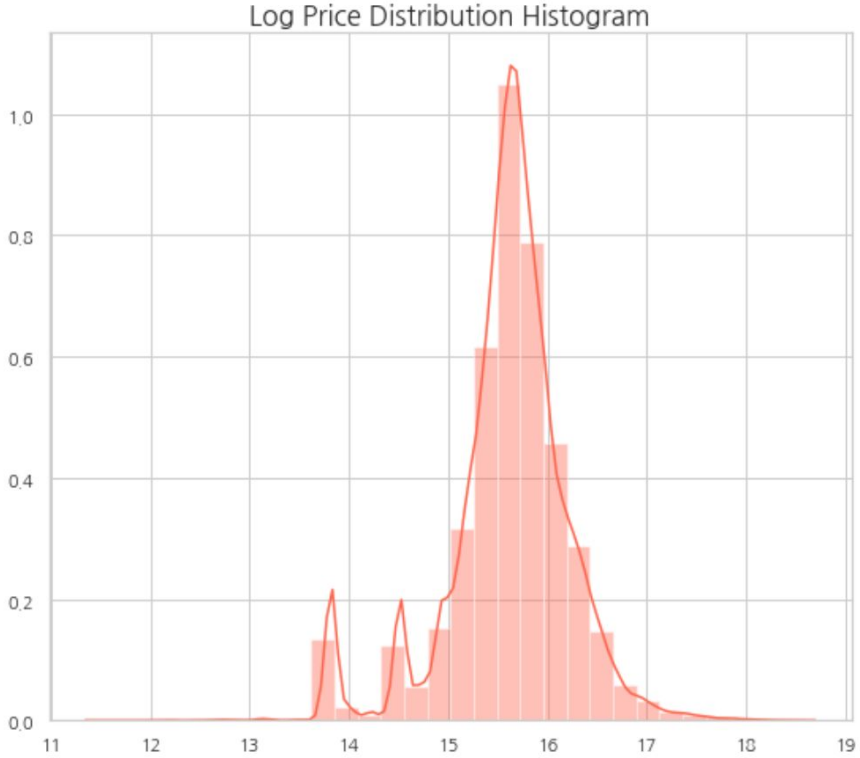
01. Exploratory Data Analysis

- Select Dataset : 캐글에서 제공되는 Sberbank의 러시아 모스코 주택 시작의 가격 변동 예측 데이터를 선택
  - https://www.kaggle.com/c/sberbank-russian-housing-market
- 에러평가
  - RMSLE (Root Mean Squared Logarithmic Error)
- 독립변수 : 평수, 층수, 주변 환경 및 1인당 소득, 외환 환율, GDP등을 포함한 시계열 거시경제 데이터 등 300가지 집 특성을 가진 3만건의 주택거래 내역
- 종속변수 : price

01.Exploratory Data Analysis

- Distribution price and log_price





Log price가 정규 분포와 유사함을 보였습니다.

02. Preprocessing & Feature Engineering

- 데이터 타입은 정수, 실수, 카테고리값으로 이루어져 있습니다.
- 카테고리 변수들은 Yes/No 아니면 순서를 가진 데이터로 Lable Encoding을 적용하였습니다.
- Missing value는 상관관계가 높은 변수들끼리 선형회귀를 하였습니다. 덜 채워진 수치 데이터는 평균값으로 카테고리 데이터는 모드값으로 대신하였습니다.
- 독립변수들 중 skewness가 1 이상인 것과 이분산성이 나타나는 변수들은 로그를 적용하여 정규분포에 가깝게 만들어 주었습니다.
- 선형 회귀 시 조건수가 크게 나와 스케일링이 필요 하다고 판단하였고 독립변수 간의 상관관계를 확인해 볼 필요가 있었습니다.

02. Preprocessing & Feature Engineering

- 차원축소
  - F 검정을 통해

    각 독립 변수가 종속 변수에

    가진 영향력을 살펴보고

    p-value가 0.05보다 크며

    중요도가 작다고 판단하여

    제거하였습니다.

```
anova = sm.stats.anova_lm(result, typ=2)
anova
```

	sum_sq	df	F	PR(>F)
C(_radiation_raion)	1.935487	1.0	9.422105	2.145765e-03
C(_thermal_power_plant_raion)	0.756073	1.0	3.680624	5.505771e-02
C(_culture_objects_top_25)	0.343303	1.0	1.671224	1.961046e-01
C(_incineration_raion)	1.916718	1.0	9.330737	2.255392e-03
C(_nuclear_reactor_raion)	0.034975	1.0	0.170263	6.798815e-01
C(_big_road1_1line)	0.674207	1.0	3.282092	7.004958e-02
C(_railroad_terminal_raion)	0.658321	1.0	3.204760	7.343436e-02
C(_product_type)	23.553646	1.0	114.661007	1.043400e-26
C(_railroad_1line)	2.322160	1.0	11.304457	7.741776e-04
C(_oil_chemistry_raion)	0.018414	1.0	0.089640	7.646374e-01
C(_detention_facility_raion)	0.128872	1.0	0.627361	4.283312e-01
C(_ecology)	1.729298	4.0	2.104589	7.742909e-02
C(_water_1line)	0.028591	1.0	0.139183	7.090971e-01
C(_big_market_raion)	0.654787	1.0	3.187553	7.421111e-02
scale(np.log(_full_sq))	0.008510	1.0	0.041427	8.387168e-01
scale(np.log(_life_sq))	0.327313	1.0	1.593384	2.068531e-01
scale(_floor)	5.551864	1.0	27.026913	2.019553e-07
scale(np.log(_num_room))	0.166561	1.0	0.810831	3.678819e-01
scale(np.log(_kitch_sq))	0.233084	1.0	1.134670	2.867900e-01
scale(_state)	5.614360	1.0	27.331146	1.725742e-07
scale(np.log(_area_m))	1.154087	1.0	5.618184	1.778133e-02
scale(_raion_popul)	0.125732	1.0	0.612075	4.340137e-01
scale(_preschool_education_centers_raion)	0.345667	1.0	1.682733	1.945717e-01
scale(_school_education_centers_raion)	0.975895	1.0	4.750733	2.929356e-02

02. Preprocessing & Feature Engineering

- 차원축소
  - 다중공선성을 없애기 위해 Variance Inflation Factor를 계산하여 10 이상인 것들은 가장 큰 순서대로 제거 하였습니다.

```
df_train[df_train==np.inf]=np.nan
df_train.fillna(df_train.median(), inplace=True)
categorial_ivs = set(df_train.columns.drop('timestamp')) - set(df_train._get_numeric_data().columns)
numeric_ivs = df_train._get_numeric_data().columns.drop('price_doc')
temp = vif.VarInflationFactor(impute=True, thresh=10.0).fit_transform(df_train[numeric_ivs])
df_train = pd.concat([df_train['timestamp'], temp, df_train[categorial_ivs], df_train['price_doc']], axis=1)
```

```
feature_to_removes = [
    "raion_popul"
    "cafe_count_3000"
    "cafe_count_5000"
    "cafe_avg_price_1500"
    "raion_build_count_with_builddate_info"
    "kremlin_km"
    "cafe_count_2000"
    "sadovoe_km"
    "cafe_count_1500"
    "0_17_all"
    "bulvar_ring_km"
    "cafe_sum_1500_max_price_avg"
    "cafe_count_5000_price_1000"
    "yearweek"
    "school_km"
    "cafe_count_5000_price_1500"
    "cafe_count_5000_price_2500"
    "cafe_count_3000_price_1500"
    "office_count_5000"
    "cafe_count_1000"
    "cafe_count_3000_price_500"
    "office_count_3000"
    "cafe_count_3000_price_2500"
    "ttk_km"
    "cafe_count_2000_price_1500"
    "cafe_count_2000_price_500"
    "cafe_count_5000_price_500"
    "avg_price_ID_railroad_terminal"
    "office_count_2000"
    "church_count_5000"
    "cafe_count_2000_price_2500"
    "cafe_count_3000_price_1000"
    "cafe_count_1500_price_1500"
    "cafe_count_5000_na_price"
    "cafe_count_2000_price_1000"
    "work_all"
    "zd_vokzaly_avto_km"
    "church_count_3000"
    "oil_chemistry_km"
    "cafe_count_1500_price_500"
    "cafe_count_5000_price_4000"
    "avg_price_ID_bus_terminal"
```

03. Modeling with Statsmodels

- 회귀 분석 성능 향상을 위해 다양한 시도를 진행하였습니다.
  - 아웃라이어
  - Lasso

03. Modeling with StatsModels

- Outlier
  - 회귀 성능 향상을 위해 가격예측에 영향을 주는 큰 레버리지를 가진 데이터를 Cook's Distance를 사용하여 회귀 분석 시 잔차와 레버리지가 큰 데이터들을

```
df_train_macro_with_outliers = df_train_macro.copy(deep=True)
df_train_macro, model, result = pp.remove_outliers(df_train_macro, formula, repeat=3)
result.summary()
```

OLS Regression Results

Dep. Variable:	np.log(_price_doc)	**R-squared:**	0.887
Model:	OLS	**Adj. R-squared:**	0.886
Method:	Least Squares	**F-statistic:**	3152.
Date:	Mon, 16 Dec 2019	**Prob (F-statistic):**	0.00
Time:	22:01:24	**Log-Likelihood:**	14330.
No. Observations:	26684	**AIC:**	-2.853e+04
Df Residuals:	26617	**BIC:**	-2.798e+04
Df Model:	66		
Covariance Type:	nonrobust		

03. Modeling with StatsModels

- Lasso

  - R-squared : 0.910

- Score(RMSLE) :

  - 0.33945

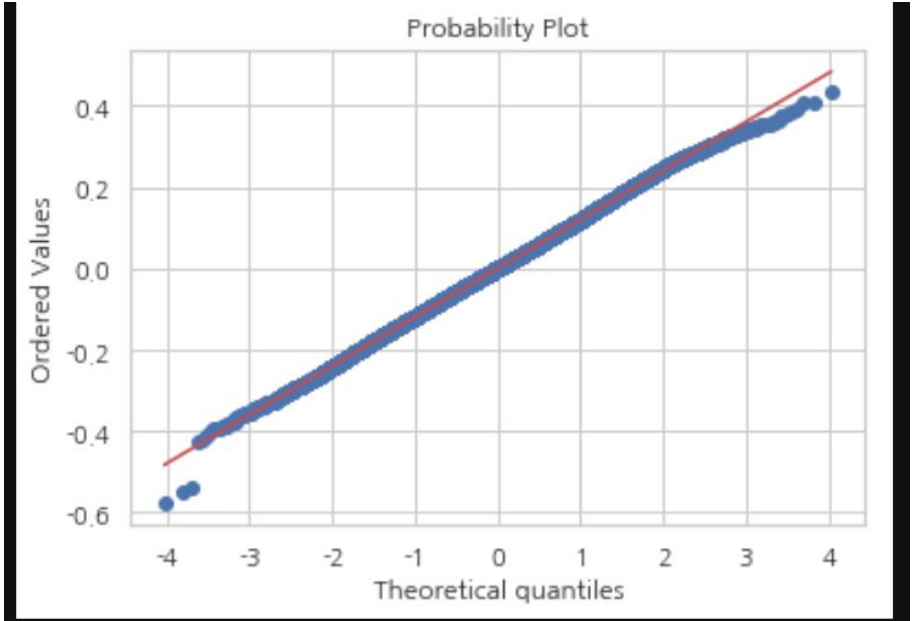
```
formula = 'np.log(_price_doc) ~ ' + " + ".join(sms_vars)
model = sm.OLS.from_formula(formula, data=df_train_macro)
result = model.fit()
result.summary()
```

OLS Regression Results

Dep. Variable:	np.log(_price_doc)	**R-squared:**	0.910
Model:	OLS	**Adj. R-squared:**	0.910
Method:	Least Squares	**F-statistic:**	6306.
Date:	Mon, 16 Dec 2019	**Prob (F-statistic):**	0.00
Time:	22:01:41	**Log-Likelihood:**	17503.
No. Observations:	25000	**AIC:**	-3.492e+04
Df Residuals:	24959	**BIC:**	-3.459e+04
Df Model:	40		
Covariance Type:	nonrobust		

03. Modeling with StatsModels

- 잔차 정규성 테스트
  - 회귀분석 모형 진단을 위해 잔차 정규성 테스트와 부분회귀 플롯을 그려 확인하였습니다. 잔차는 정규 분포를 따르는 것으로 보여졌습니다.

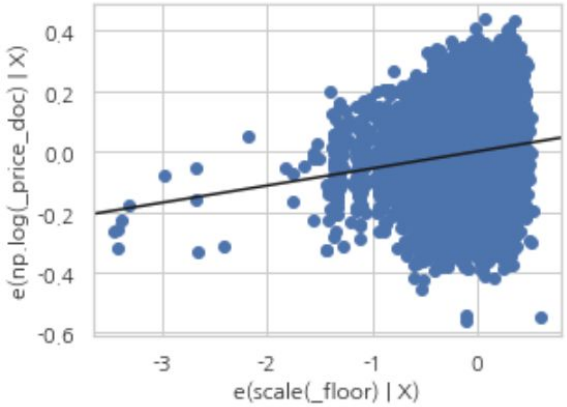


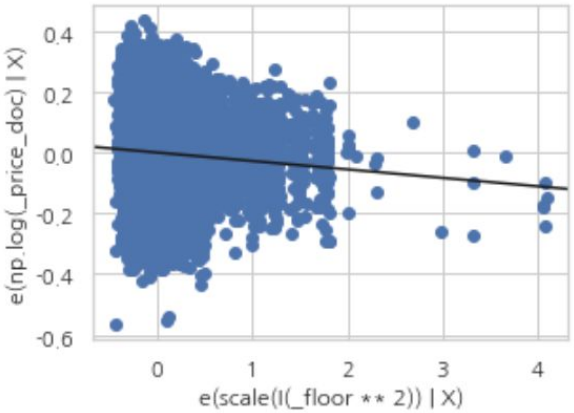

```
test = sms.omni_normtest(result.resid)
for xi in zip(['Chi^2', 'P-value'], test):
    print("%-12s: %6.3f" % xi)
```

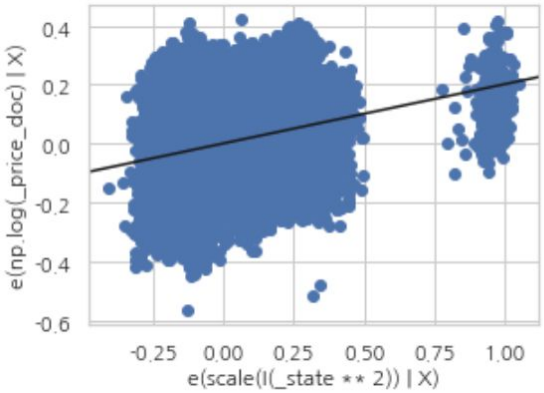
```
Chi^2        :  0.481
P-value      :  0.786
```

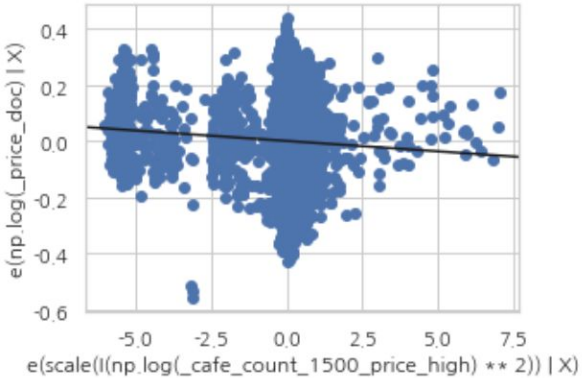
03. Modeling with StatsModels

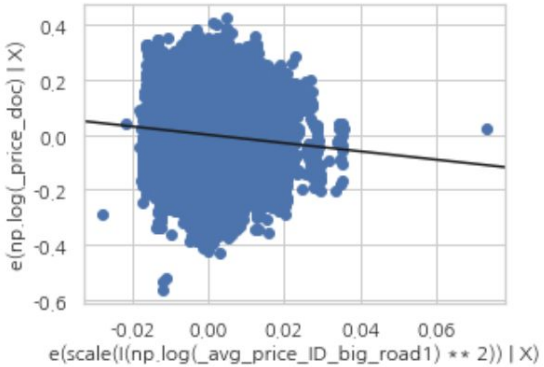
- Residual Plot

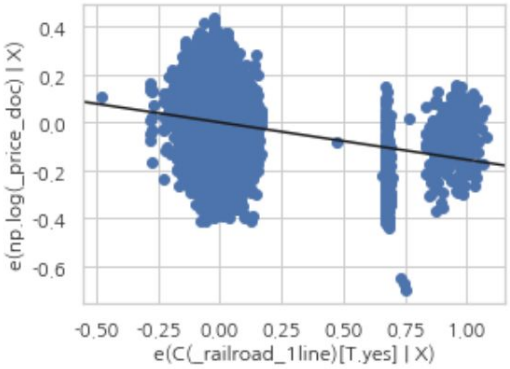












04. Modeling with XGBoost

- Preprocessing
  - Label Encoding, Log Normalization, Standard Scaling
- Regression
  - Cross-validation and Results
  - Feature Importance
- Submission Score

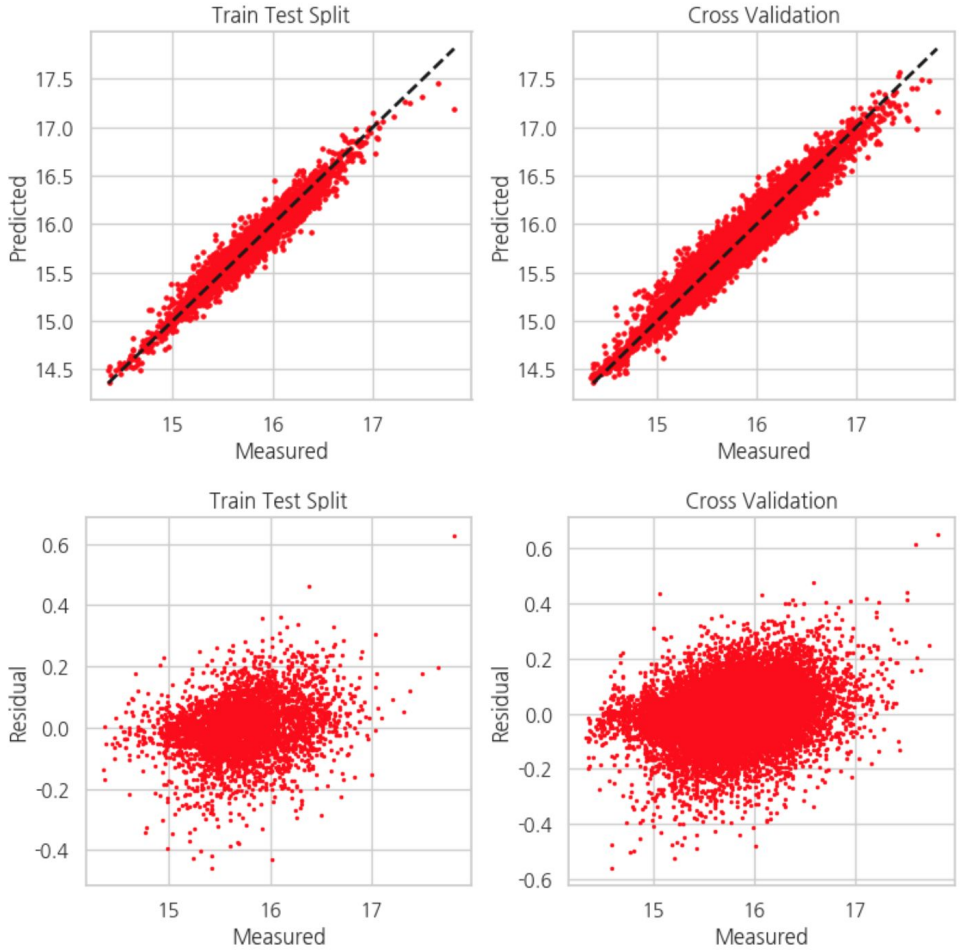
04. Modeling with XGBoost

- Train Test Split

  RMSE : 0.0865

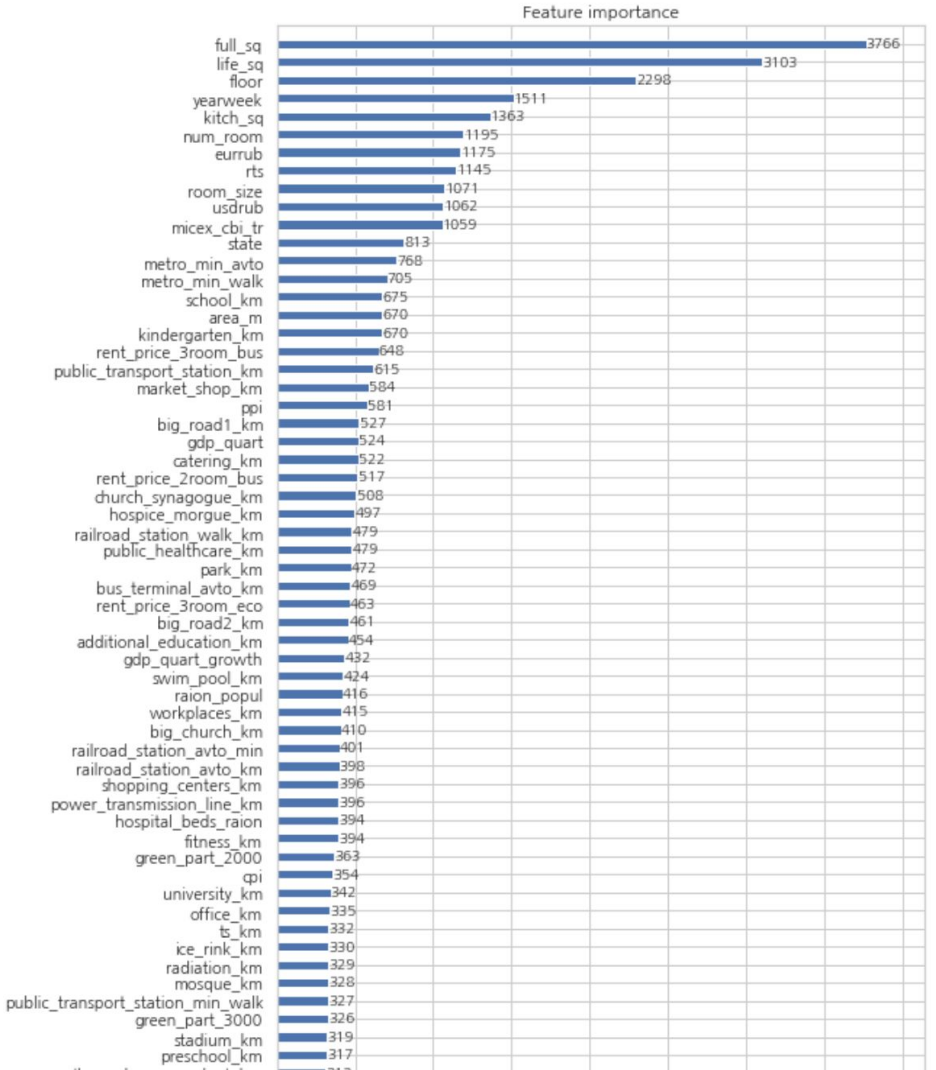
- Cross Validation

  RMSE : 0.0850



04. Modeling with XGBoost

- Feature Importance



향후 발전 방향

- OLS와 XGBoost 모델의 퍼포먼스의 차이가 근소하였습니다. 반복적인 Feature Engineering과 모델 튜닝이 필요합니다.

Github url : https://github.com/hojisu/sberbank-russian-housing-market