데이터 전처리

1- 1데이터 import

```
df = pd.read_csv("1st_train_mdf.csv")
```

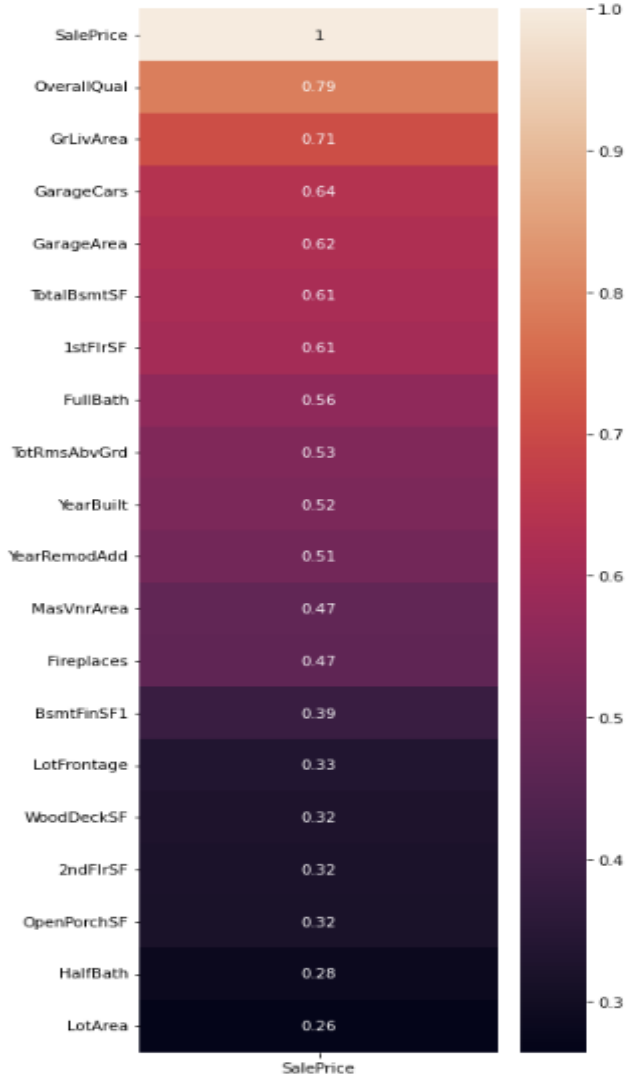
1- 2결측치 처리

결측치 이외에도 Id 변수 또한 필요치 않다고 판단, 삭제

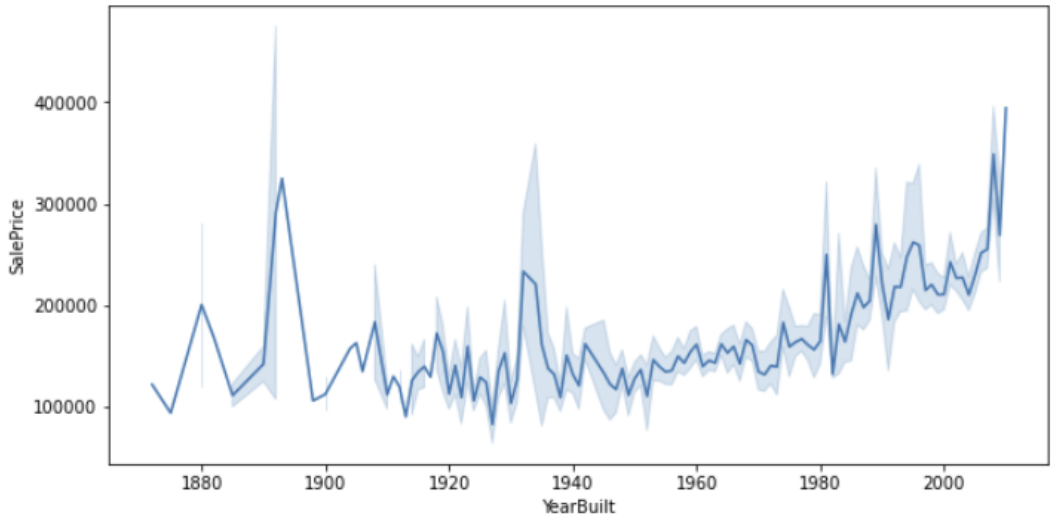
변수명	타입	결측 개수	원인	처리방식
Alley	범주/명목	1369(93.7%)	No alley access	결측치가 높은 비중을 차지, 삭제
PoolQC	범주/순위	1453(99.5%)	No Pool	결측치가 높은 비중을 차지, 삭제
Fence	범주/순위	1179(80.7%)	No Pence	결측치가 높은 비중을 차지, 삭제
MiscFeature	범주/명목	1406(99.5%)	None	결측치가 높은 비중을 차지, 삭제
LotFrontage	수치/연속	259(17.7%)	알 수 없음	LotFrontage의 중위값으로 설정
MasVnrArea	수치/연속	8(0.5%)	벽돌 외장 면적 0	0으로 처리
MasVnrType	범주/명목	8(0.5%)	None	NaN 값을 None로 처리
GarageType	범주/명목	81(5.5%)	No Garage	None로 처리
GarageYrBlt	범주/명목	81(5.5%)	No Garage	None로 처리
GarageFinish	범주/명목	81(5.5%)	No Garage	None로 처리
GarageQual	범주/순위	81(5.5%)	No Garage	None로 처리
GarageCond	범주/순위	81(5.5%)	No Garage	None로 처리
BsmtQual	범주/명목	37(2.5%)	No Basement	None로 처리
BsmtCond	범주/순위	37(2.5%)	No Basement	None로 처리
BsmtExposure	범주/순위	38(2.6%)	No Basement	None로 처리
BsmtFinType1	범주/순위	37(2.5%)	No Basement	None로 처리
BsmtFinType2	범주/순위	38(2.6%)	No Basement	None로 처리
FireplaceQu	범주/순위	690(47.2%)	No Fireplace	None로 처리
Electrical	범주/명목	1(0.06%)	알 수 없음	Top 값인 SBrkr 값으로 채워줌

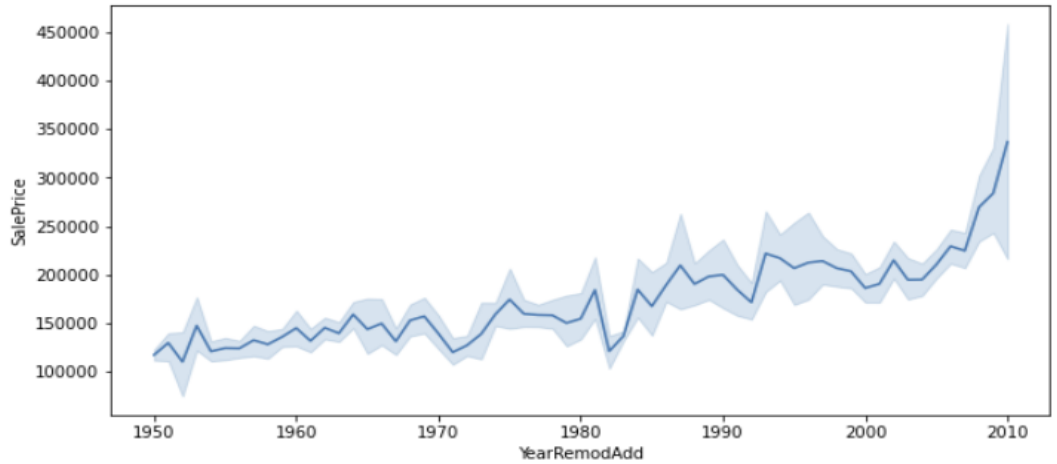
1- 3Visualization

중요 변수 20개



YearBuilt, YearRemod가 0.52, 0.51의 상관관계로 높은 수준의 상관관계를 보여주고 있음





리모델링 여부 관계를 확인하면서 '언제 팔렸는지'의 변수또한 중요하다고 생각했는데 년도들이 그리 차이나지 않아서(1년 4개월정도) 고려대상에서 제외하였음

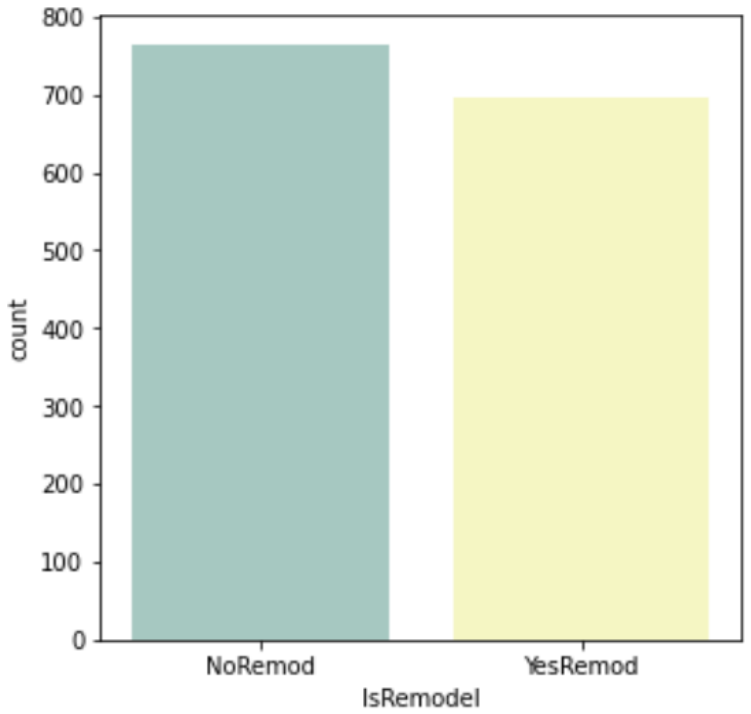
```
1  df['YrSold'].describe()
```

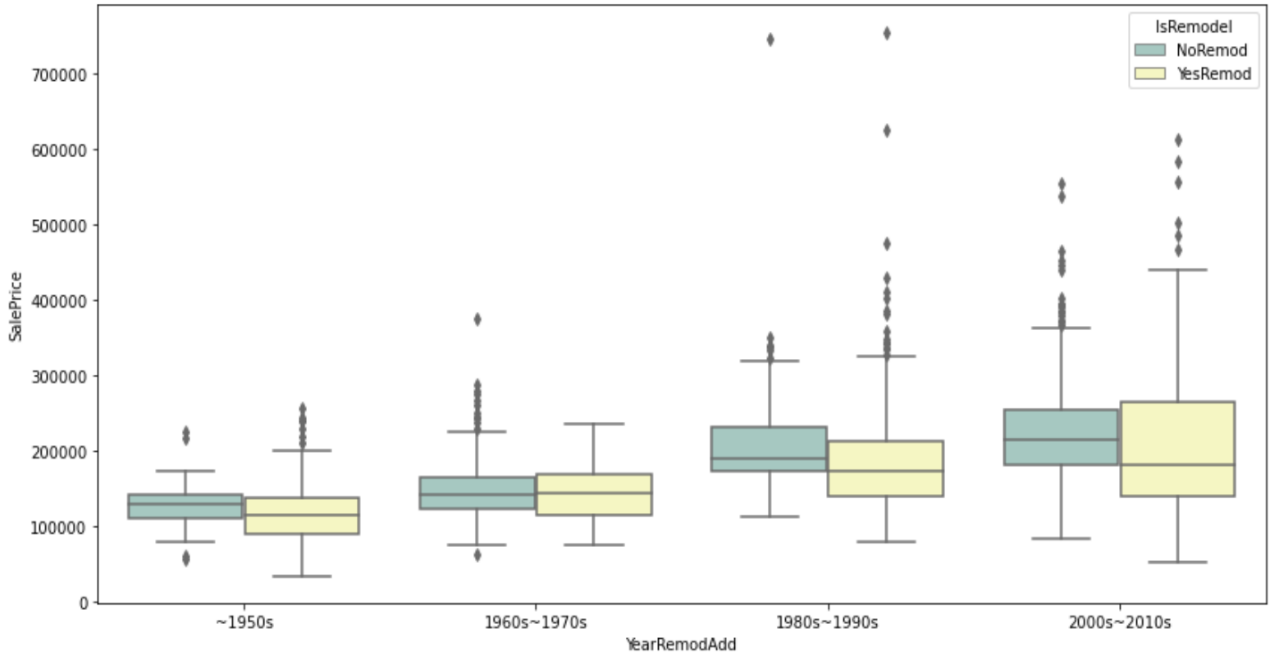
```
count     1460.000000
mean      2007.815753
std          1.328095
min       2006.000000
25%       2007.000000
50%       2008.000000
75%       2009.000000
max       2010.000000
Name: YrSold, dtype: float64
```

YearBuilt, YearRemodAdd의 년도가 동일한 경우: NoRemod
년도가 동일하지 않은 경우: YesRemod

```
1  df['IsRemodel'] = df[['YearBuilt', 'YearRemodAdd']].apply(lambda x: "NoRemod" if x[0] == x[1]
2                                                             else "YesRemod", axis = 1)
```

<IsRemodel; NoRemod, YesRemod의 개수>





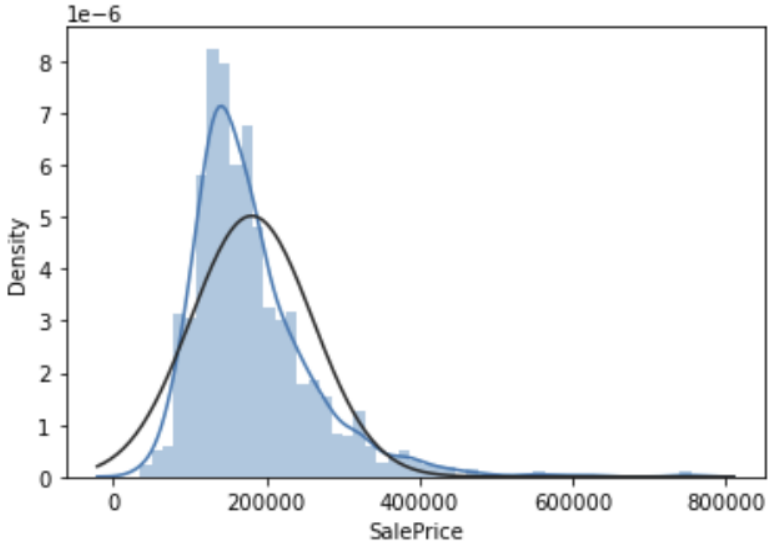
박스 플롯을 통해 확인해본 결과,
대다수의 데이터에서 생각보다
리모델링 여부가 가격에 영향을 미치지 않았음

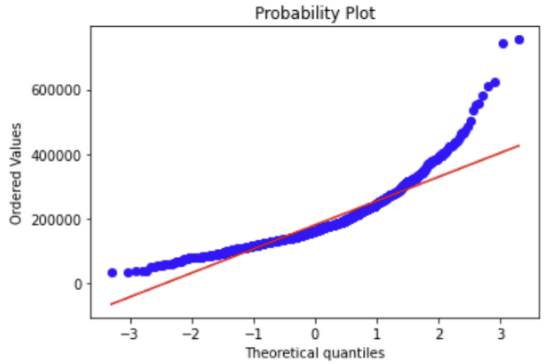
1- 4원핫인코딩

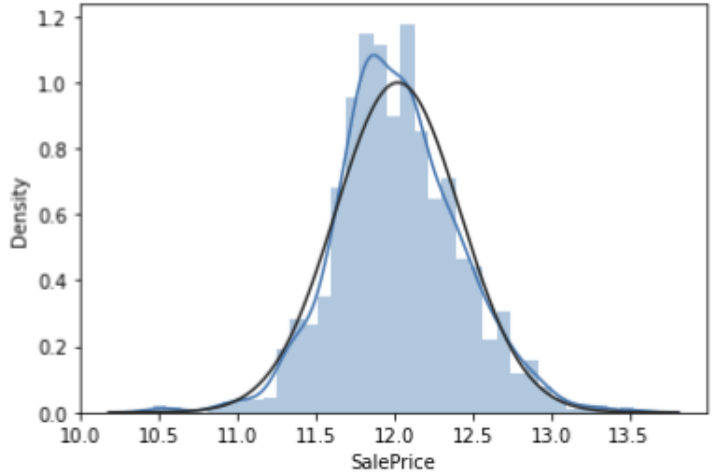
```
df = pd.get_dummies(df)
```

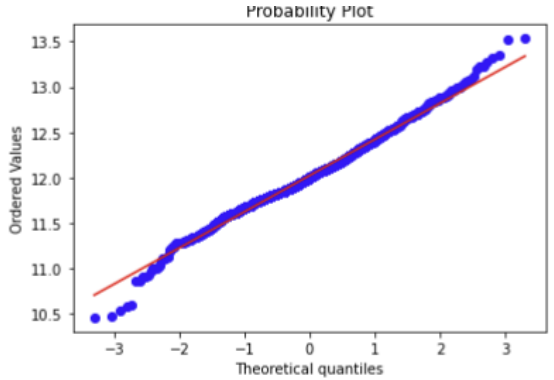
1- 5로그 변환

SalePrice 데이터를 보면 왼쪽으로 치우쳐 있고, 정규분포에서 많이 벗어난 모습을 볼 수 있음









로그 변환 후, 정규분포와 빨간 사선에 모두 부합한 모습

1- 6데이터세트를 훈련용, 테스트용으로 분할

```
# trainset, testset 분리
from sklearn.model_selection import train_test_split
X = df.drop('SalePrice', axis = 1)
y = df['SalePrice']
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2, random_state=42)
```

예측모델 구현 | 2- 1선형 회귀

```
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
lin_scores = cross_val_score(lin_reg, X_train, y_train, scoring="neg_mean_squared_error", cv=10, n_jobs=-1)
lin_rmse = np.sqrt((-lin_scores).mean())
lin_rmse
```

13712.537839819337

2-2. 규제 모델

릿지

```
ridge_reg = Ridge()
ridge_reg.fit(X_train, y_train)
ridge_scores = cross_val_score(ridge_reg, X_train, y_train, scoring = "neg_mean_squared_error", cv = 10, n_jobs=-1 )
ridge_reg_rmse = np.sqrt((-ridge_scores).mean())
ridge_reg_rmse
```

0.15696568480021297

라쏘

```
lasso_reg = Lasso()
lasso_reg.fit(X_train, y_train)
lasso_scores = cross_val_score(lasso_reg, X_train, y_train, scoring = "neg_mean_squared_error", cv=10, n_jobs=-1)
lasso_rmse = np.sqrt(-lasso_scores).mean()
lasso_rmse
```

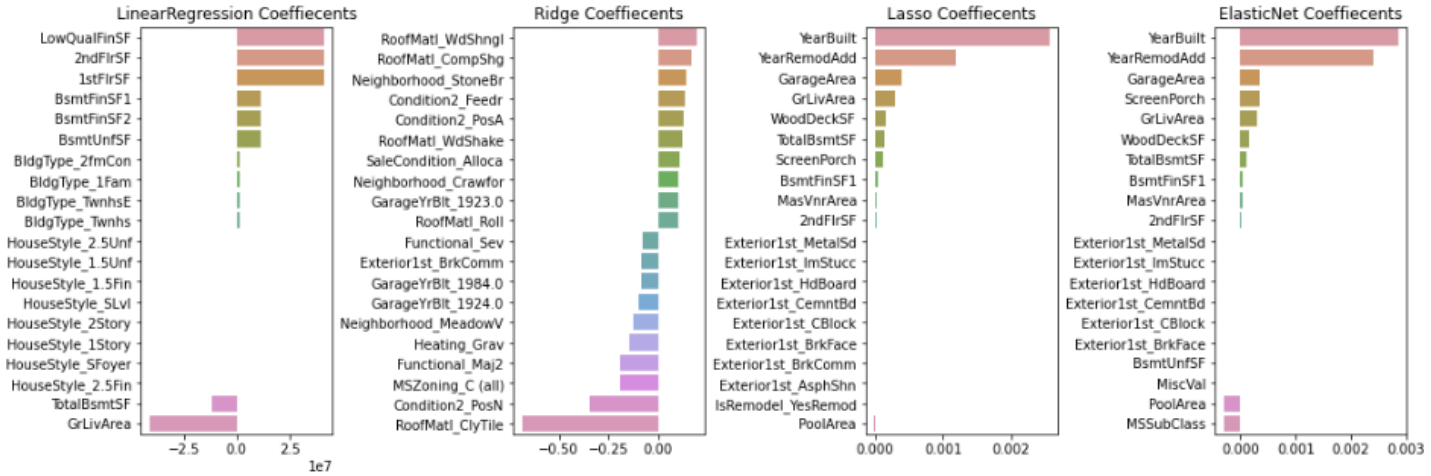
0.1965057828911776

엘라스틱넷

```
elastic_reg = ElasticNet()
elastic_reg.fit(X_train, y_train)
elastic_scores = cross_val_score(elastic_reg, X_train, y_train, scoring = "neg_mean_squared_error", cv=10, n_jobs=-1)
elastic_rmse = np.sqrt(-elastic_scores).mean()
elastic_rmse
```

0.19069598691213752

2-3. 선형 회귀계수 시각화



Lasso, ElasticNet에서 YearBuilt, YearRemodAdd가 주택가격에 영향을 미치는 변수 top2로 나타났음

2-4. 랜덤포레스트(RandomForest)

```
from sklearn.ensemble import RandomForestRegressor
rf_reg = RandomForestRegressor()
rf_reg.fit(X_train, y_train)
rf_scores = cross_val_score(rf_reg, X_train, y_train, scoring="neg_mean_squared_error", cv=10, n_jobs=-1)
rf_rmse = np.sqrt((-rf_scores).mean())
rf_rmse
```

0.14655598653186508

2-5. 의사결정트리(DecisionTree)

```
from sklearn.tree import DecisionTreeRegressor
tree_reg = DecisionTreeRegressor()
tree_reg.fit(X_train, y_train)
tree_scores = cross_val_score(tree_reg, X_train, y_train, scoring="neg_mean_squared_error", cv=10, n_jobs=-1)
tree_rmse = np.sqrt((-tree_scores).mean())
tree_rmse
```

0.21410793403446413

2-6. XGB

```
import xgboost as xgb
xgb_reg = xgb.XGBRegressor()
xgb_reg.fit(X_train, y_train)
xgb_scores = cross_val_score(xgb_reg, X_train, y_train, scoring="neg_mean_squared_error", cv=10, n_jobs=-1)
xgb_rmse = np.sqrt((-xgb_scores).mean())
xgb_rmse
```

0.1403799882028546

파라미터 | 3-1. 그리드탐색

Ridge 10 CV 시 최적 평균 RMSE 값: 0.14138500155873734, 최적 alpha:{ ' alpha': 15} : **0.156 -> 0.141**

Lasso 10 CV 시 최적 평균 RMSE 값: 0.14258860975561335, 최적 alpha:{ ' alpha': 0.001} **0.196 -> 0.142**

ElasticNet 10 CV 시 최적 평균 RMSE 값: 0.14026914972336943, 최적 alpha:{ ' alpha': 0.001} **0.19 -> 0.14**

랜덤포레스트 10CV 최적 평균 RMSE 값: 0.143067931, **0.146 -> 0.143**

DecisionTree10 CV시 최적 평균 RMSE 값: 0.204803587 **0.214 -> 0.204**

XGB 10 CV시 최적 평균 RMSE 값: 0.138070487 **0.140-> 0.138** **XGB가 가장 좋은 성능이 나왔음!**

검증 | 4-1. 유의성 검정

Dep. Variable:	SalePrice	R-squared:	0.950
Model:	OLS	Adj. R-squared:	0.930
Method:	Least Squares	F-statistic:	48.54
Date:	Sat, 04 Feb 2023	Prob (F-statistic):	0.00
Time:	03:11:23	Log-Likelihood:	1187.7
No. Observations:	1168	AIC:	-1719.
Df Residuals:	840	BIC:	-58.64
Df Model:	327		
Covariance Type:	nonrobust		

	coef	std err	t	P> | t |	[0.025	0.975]
YearBuilt	0.0019	0.000	4.442	0.000	0.001	0.003
YearRemodAdd	0.0011	0.000	3.174	0.002	0.000	0.002

- 일반적으로 P- value가 0.05보다 작으면 모델이 통계적으로 유의하다고 판단
- 원래 가설 변수인 'YearBuilt'와 'YearRemodAdd'는 0.001보다 작은, 0.002의 값을 가지므로
- 본 변수 자체로는 통계적으로 유의미

IsRemodel_No-Remod	0.6386	0.306	2.085	0.037	0.037	1.240
IsRemodel_Yes-Remod	0.6386	0.307	2.083	0.038	0.037	1.240

- IsRemodel.NoRemod, IsRemodel_ YesRemod또한 0.037, 0.038을 값을 가지므로(<0.05)
- 신뢰할 수 있는 결과값이라 할 수 있음

4-2. 설명력 검정

R-squared:	0.950
Adj. R-squared:	0.930

- 1에 가까울수록 회귀 모델이 추정한 가격과 실제 주택 가격의 차이가 작다는 것을 의미

4-3. 변수 영향력 분석

Coef

- YearBuilt는 0.0019, YearRemodAdd는 0.0011로 변수의 영향력이 작음
- 오히려, IsRemodel의 여부가 둘 다 0.6386으로 높은 영향력을 보여줌

Std err

- YearBuilt와 YearRemodAdd는 0의 값을 가지므로 해당 회귀 계수 추정치를 신뢰할 수 있음
- IsRemodel의 변수들은 각각 0.306, 0.307으로 회귀 계수의 약 40%의 높은 값을 가지므로 본 변수의 회수 계수 추정치를 신뢰할 수 없음

결론 | 리모델링 여부가 주택 가격에 영향을 미치지 않는다.

- YearBuilt, YearRemodAdd 변수들과 SalePrice의 상관관계가 0.52, 0.51로 높은 수준인 점,
- lineplot에서 우상향의 그래프를 보이는 점을 통해 리모델링의 여부가 집 가격에 영향이 있을 것이라 판단

- 하지만 해당 변수들을 통해 '리모델링 여부(IsRemodel) 을 만들어 확인해본결과,
- boxplot - 리모델링 한 데이터들이 보다 높은 가격이 형성되어 있다 말하기 어려웠으며
- IsRemodel- YesRemod, NoRemod가 Coef에 비하여 높은 Std err 값을 가졌기(40%)에 해당 추정치를 신뢰할 수 없었음