데이터과학 HW2(주택 가격 예측하기)

1. Dataset 로드

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
#dataset을 load합니다.
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
#load한 dataset을 확인합니다
print(train.head())
print(test.head())
```

처음에 dataset을 불러오고 제대로 데이터가 로드되었는지 확인했습니다.

```
train.columns
```

위의 코드로 주택가격에 영향을 줄 수 있는 요소들을 출력하였습니다. 우선 학습에 사용할 상관관계가 높은 요소를 파악하기 위해 correlation을 구하는 과정인 EDA를 진행하였습니다.

2. Preprocessing

데이터 전처리를 통해서 일부 범주형 데이터를 수치화하고 결측치를 제거해보겠습니다.

범주형 데이터 수치화

```
num_feats = ['LotFrontage', 'LotArea', 'OverallQual', 'OverallCond',
             ...#일부를 생략하였습니다.
             'YrSold','SalePrice']
```

num_feats는 수치 데이터를 가진 요소들의 리스트 입니다. 수치데이터로 변환 가능한 범주형 데이터들도 포함이 되어있습니다. 해당 데이터들은 이후에 수치형 데이터로 변환합니다.

```
grade_feat = ['OverallQual', 'OverallCond', 'ExterQual', 'ExterCond', 'BsmtQual', 'BsmtCond',
          'HeatingQC', 'KitchenQual', 'FireplaceQu', 'GarageQual', 'GarageCond', 'PoolQC']
grade = train[grade_feat]
grade.head()
```

수치형 데이터로 변환이 가능한 등급을 나타내는 범주형 데이터를 grade_feat 리스트로 저장합니다.

```
level = ['Ex', 'Gd', 'TA', 'Fa', 'Po']
num_level = [10, 8, 6, 4, 2]
changed = dict(zip(level, num_level))
grade = grade_data.replace(changed)
grade.head()
```

퀄리티를 나타내는 등급에는 'Ex', 'Gd', 'TA', 'Fa', 'Po' 가 있습니다. 등급 간의 차이를 나타내기 위해 2의 간격을 두고 수치 데이터와 1대1로 대응시켰습니다. 수치화된 데이터를 기존의 범주형 데이터와 교환해주면 완료입니다.

```
train[grade_feat] = train[grade_feat].replace(changed)
```

수치화할 수 있는데 범주형 데이터는 변환이 완료되었으므로 이제 범주형 데이터는 필요가 없습니다. 범주형 데이터를 train dataset에서 제거하겠습니다. 앞서 수치형데이터를 num_feats로 선언했으므로 여집합의 개념을 활용하면 됩니다.

```
#범주형 데이터를 제거해줍니다.
#num_feats를 활용해서 범주형 데이터 선언.
cat_feats = train.drop(num_feats, axis=1).columns
#범주형 데이터 제거
train.drop(cat_feats,axis=1,inplace=True)
```

결측치 제거

```
#preprocessing 2 : 결측치 제거
train.isnull().sum()[train.isnull().sum() > 0]
```

우선 결측치를 파악하기 위해 결측치가 있는 요소들을 모두 출력하였습니다.

```
train.drop(['LotFrontage','FireplaceQu','PoolQC'],axis=1,inplace=True)
```

결측치의 개수가 상대적으로 많은 세가지 요소 LotFrontage, FireplaceQu, PoolQC를 제거합니다.

```
print(train.shape)
train.dropna(axis=0,how='any',inplace=True)
print(train.shape)
```

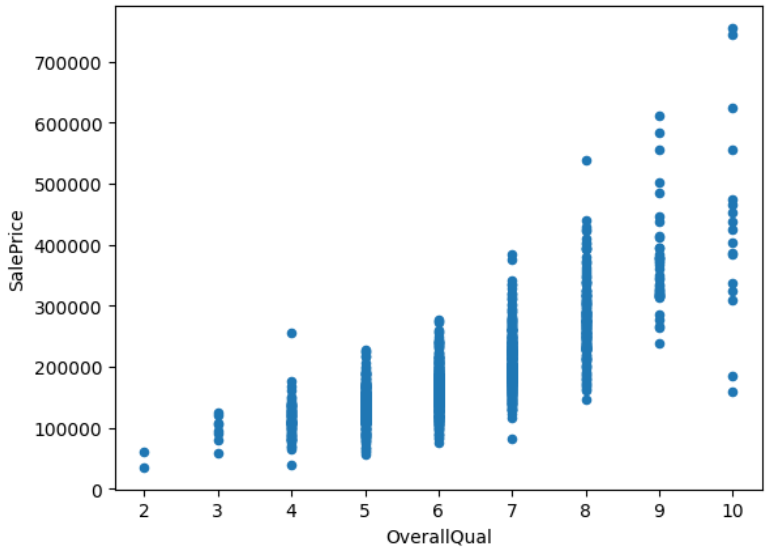
결측치를 포함한 행을 제거해주고 그 전후 데이터프레임을 비교해봅니다. 결측치 제거 전에는 (1460, 42) 즉, 행이 1460개, 열이 42개였는데 결측치 제거 후에는 (1341, 42) 즉, 행이 1341개로 감소하였습니다.

3. EDA

이제 모델의 학습을 위해서 변수를 구하는 과정을 진행하겠습니다. 처음에 상식적으로 주택가격에 상관관계가 높아보이는 요소들을 체크해보았습니다. 해당 요소들은 다음과 같습니다. OverallQual : 전반적인 사양(질), YearBuilt : 건축일, Electrical : 전기 시스템, Neighborhood : 이웃, Fireplaces : 벽난로 개수, Bedroom : 방 개수, MSZoning : 지구/지역, GrLivArea : 지상층의 크기.

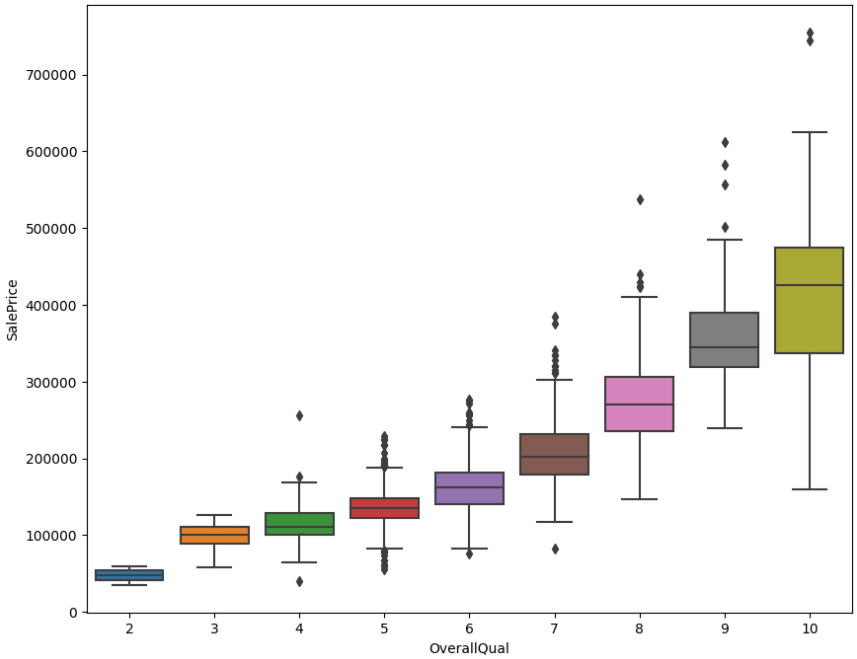
OverallQual

```
data = pd.concat([train['SalePrice'],train['OverallQual']],axis = 1)
data.plot.scatter(x = 'OverallQual', y = 'SalePrice')
```



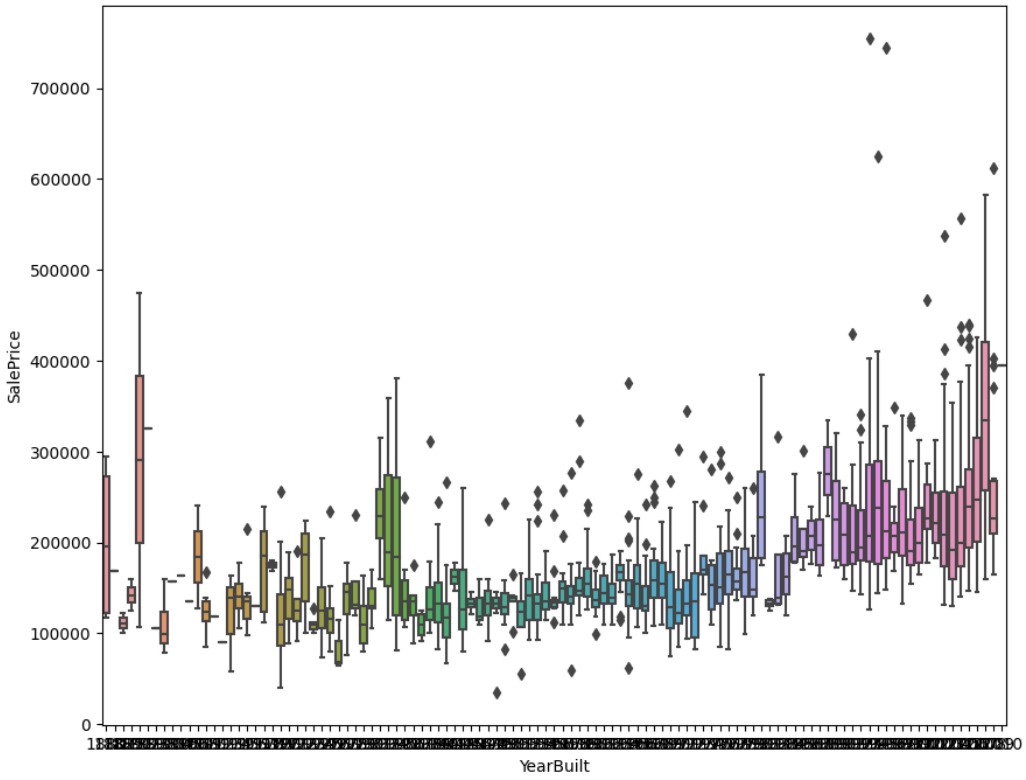
OverallQual의 경우 우상향하는 관계가 있음을 살펴볼 수 있습니다. 이는 boxplot을 사용하였을때 더 직관적으로 확인할 수 있었습니다.

```
data = pd.concat([train['SalePrice'],train['OverallQual']],axis = 1)
f, ax = plt.subplots(figsize=(10, 8))
fig = sns.boxplot(x = 'OverallQual', y = 'SalePrice', data = data)
```



YearBuilt

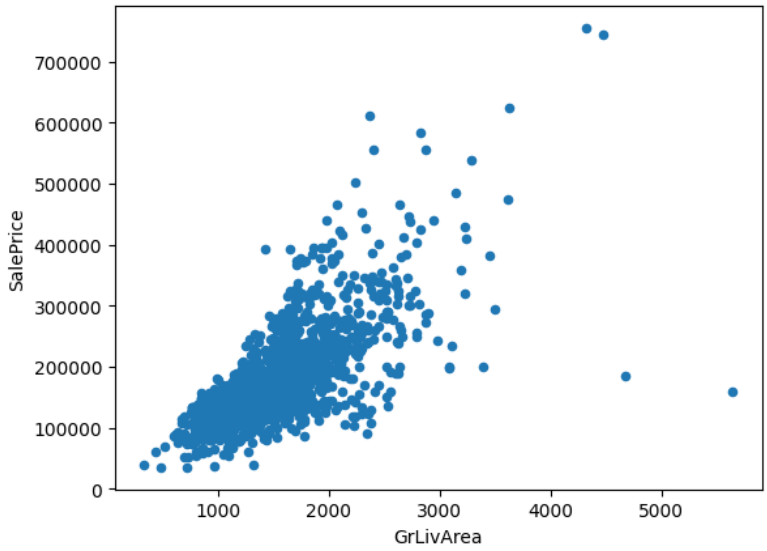
```
data = pd.concat([train['SalePrice'],train['YearBuilt']],axis = 1)
f, ax = plt.subplots(figsize=(10, 8))
fig = sns.boxplot(x = 'YearBuilt', y = 'SalePrice', data = data)
```



YearBuilt의 경우 우상향하는 경향이 있긴 하였지만 뭔가 애매하다는 느낌이 들었습니다.

GrLivArea

```
data = pd.concat([train['SalePrice'],train['GrLivArea']],axis = 1)
data.plot.scatter(x = 'GrLivArea', y = 'SalePrice')
```



GrLivArea의 경우 scatter plot으로 확인하여도 큰 상관관계가 있다고 판단되었습니다. 하지만 이렇게 하나하나 비교하는 것보다 heatmap을 활용해서 이미지로 보는게 더 직관적고 효율적이라고 판단하여서 heatmap을 활용하였습니다.

Heatmap을 활용한 correlation 구하기

Heatmap의 중요한 parameter

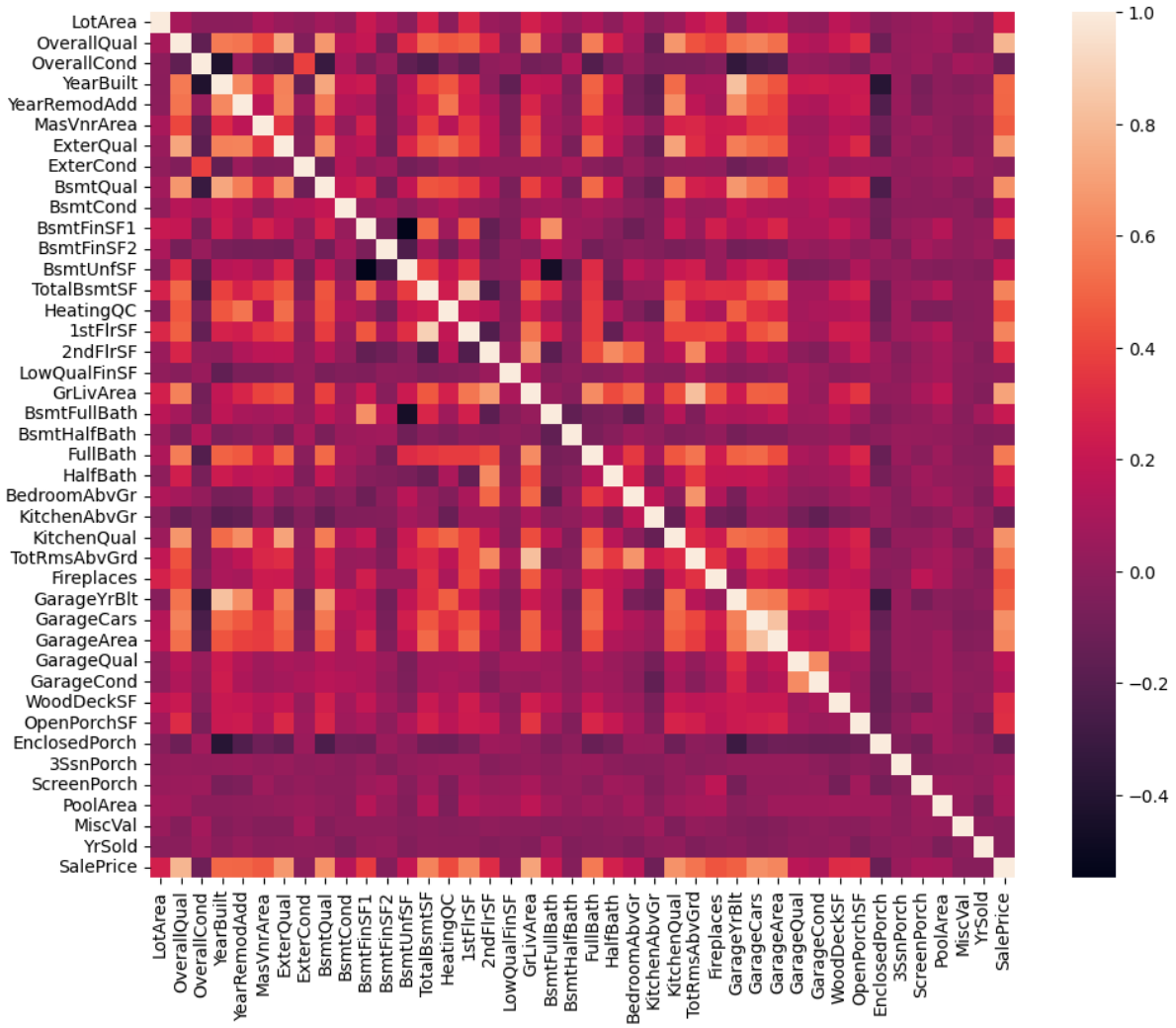
Seaborn heatmap에서 `annot` 매개변수를 `True`로 설정하면 각 셀에 해당하는 상관계수 값을 표시할 수 있습니다. 이 값을 통해 두 변수 간의 상관관계의 정도를 직접 확인할 수 있습니다. 또한 Seaborn heatmap에서 `mask` 매개변수를 사용하여 특정 데이터 값을 무시하거나 표시할 수 있습니다. 예를 들어, 특정 상관관계가 특별히 중요하지 않거나 불필요한 경우 해당 값의 셀을 표시하지 않도록 할 수 있습니다.

Heatmap으로 상관관계 알아보기

Seaborn heatmap에서 상관관계는 색상으로 표시됩니다. 일반적으로 높은 상관관계를 가진 쌍은 비슷한 색상을 가지며, 낮은 상관관계를 가진 쌍은 서로 다른 색상을 가집니다. 상관관계는 주어진 데이터에 대한 통계적 관계를 나타내며, -1에서 1까지의 값을 가집니다. 양의 상관관계는 값이 증가하면 함께 증가하는 변수들을 의미하며, 음의 상관관계는 값이 증가하면 함께 감소하는 변수들을 의미합니다. 상관계수는 변수들 간의 관계를 알아볼 때 유용한 개념입니다. 특정 데이터에 대한 상관관계를 볼 수 있습니다. 상관계수는 -1부터 1사이로 구성되고, 0.3부터 0.7 사이면 뚜렷한 양적 선형관계, 0.7 이상이면 강한 양적 상관관계라 할 수 있습니다.

Correalation 구하기

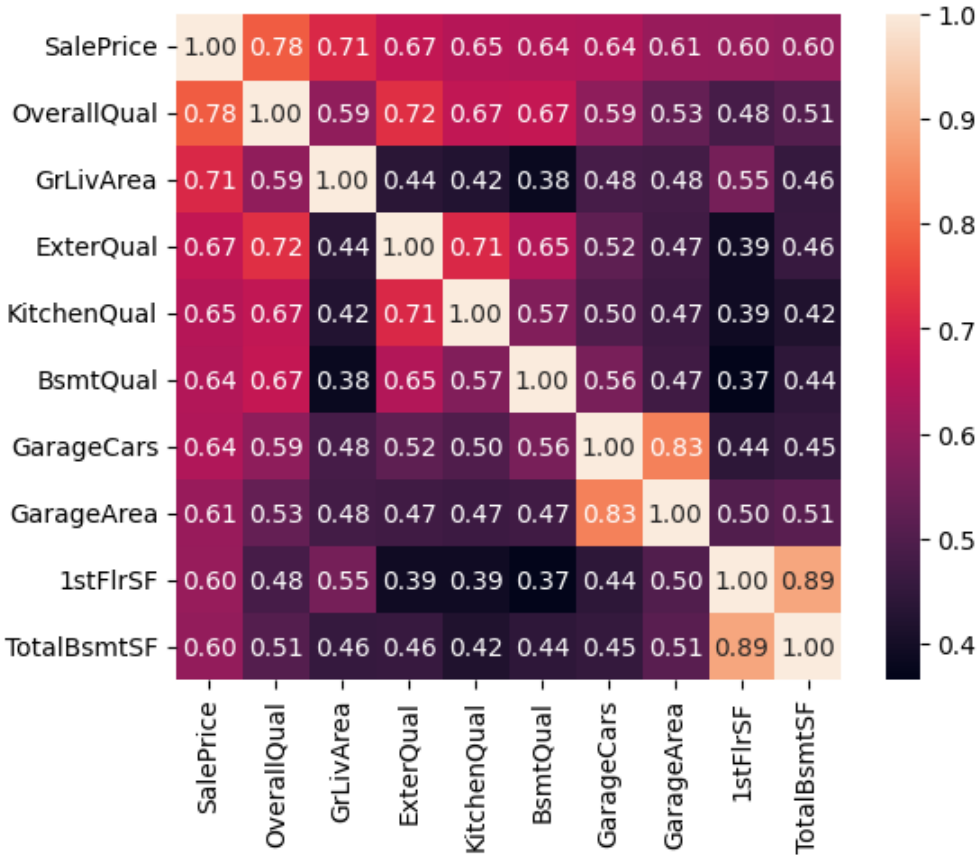
```
corr = train.corr()
f, ax = plt.subplots(figsize = (12,9))
sns.heatmap(corr, square = True)
```



처음에는 위의 그림과 같이 모든 요소들을 heatmap이미지로 출력하였으나 요소들이 너무 방대하여서 직관적으로 해석하기가 힘들었습니다. 따라서 주택 가격과 상관관계가 높은 9가지 요소들로 heatmap을 구성하였습니다.

Heatmap 규모 줄이기

```
#변수가 너무 많아서 확인하기 곤란합니다. 따라서, 변수의 개수를 9개로 제한했습니다.
corr_ten = corr.nlargest(10,'SalePrice')['SalePrice']
corr_ten_idx = corr_ten.index
corr_mat = np.corrcoef(train[corr_ten_idx].values.T)
heatmap = sns.heatmap(corr_mat, cbar=True, annot=True, square=True, fmt='.2f'
                      ,yticklabels = corr_ten_idx.values,xticklabels = corr_ten_idx.values )
plt.show()
```



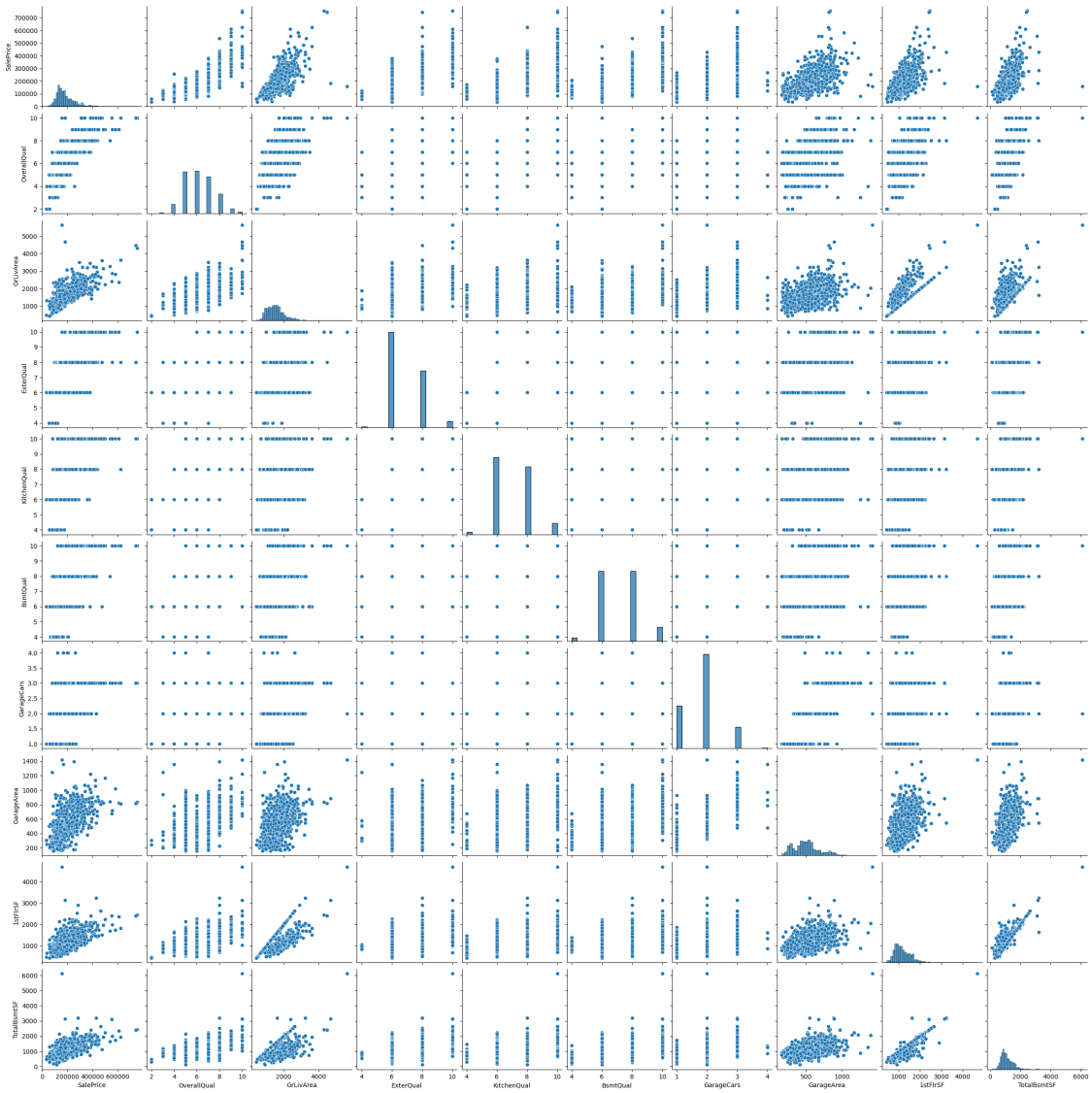

```
corr_df = pd.DataFrame([corr_ten_idx,corr_ten]).T
print(corr_df)
```

출력값

```
0 SalePrice 1.0
1 OverallQual 0.783641
2 GrLivArea 0.711438
3 ExterQual 0.66958
4 KitchenQual 0.650042
5 BsmtQual  0.643288
6 GarageCars  0.640611
7 GarageArea  0.608128
8 1stFlrSF  0.604934
9 TotalBsmtSF 0.600015
```

OverallQual과 GrLivArea는 강한 양적 선형관계, 나머지 7개의 요소는 뚜렷한 양적 선형관계에 있다고 할 수 있습니다.

```
sns.pairplot(train[corr_ten_idx])
```



상위 9개의 데이터에 대한 pairplot을 그려봤습니다.

4. 회귀 모델 훈련

우선 학습 데이터를 정답과 독립변수로 나눠야합니다.

```
X1 = train[corr_ten_idx[1:10]]
X2 = train[corr_ten_idx[1:9]]
X3 = train[corr_ten_idx[1:8]]
X4 = train[corr_ten_idx[1:7]]
X5 = train[corr_ten_idx[1:6]]
X6 = train[corr_ten_idx[1:5]]

#정답 데이터이다(SalePrice).
y = train[corr_ten_idx[0]]
```

회귀 모델을 훈련하기 위해서 독립변수를 설정해야합니다. 독립변수는 train dataset에서 앞서 선택한 feature들 중에서 선정하였는데 feature selection 방법 중에서 backward feature elimination을 통해

서 주요하지 않은 변수들을 하나씩 제거하는 방식을 택하였습니다. X1부터 X6까지 6가지의 서로 다른 독립변수를 설정하였다. 그리고 y에는 SalePrice 데이터가 저장되었습니다.

```
#선형 회귀모델을 import하고 훈련 데이터와 테스트 데이터를 구분하였다.
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

X1_train, X1_test, y1_train, y1_test = train_test_split(X1,y,test_size=0.2, random_state = 42)
X2_train, X2_test, y2_train, y2_test = train_test_split(X2,y,test_size=0.2, random_state = 42)
X3_train, X3_test, y3_train, y3_test = train_test_split(X3,y,test_size=0.2, random_state = 42)
X4_train, X4_test, y4_train, y4_test = train_test_split(X4,y,test_size=0.2, random_state = 42)
X5_train, X5_test, y5_train, y5_test = train_test_split(X5,y,test_size=0.2, random_state = 42)
X6_train, X6_test, y6_train, y6_test = train_test_split(X6,y,test_size=0.2, random_state = 42)

model1 = LinearRegression()
model2 = LinearRegression()
model3 = LinearRegression()
model4 = LinearRegression()
model5 = LinearRegression()
model6 = LinearRegression()

model1.fit(X1_train,y1_train)
model2.fit(X2_train,y2_train)
model3.fit(X3_train,y3_train)
model4.fit(X4_train,y4_train)
model5.fit(X5_train,y5_train)
model6.fit(X6_train,y6_train)
```

학습을 위해서 각각의 6개의 X에 대해서 따로 훈련데이터와 테스트데이터를 분리했고, 선형 회귀 모델을 6개를 정의해서 각각 따로 훈련시켰습니다.

6. 모델 평가

R-squared(Coefficient of determination, 결정 계수)

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST} = 1 - \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2)}$$

SSE(Sum of Squares Error) : 관측치와 예측치의 차이

SSR(Sum of Squares Regression) : 예측치와 평균 차이

SST(Sum of Squares Total) : 관측치와 평균 차이(SSE + SSR)

R-squared는 현재 사용하고 있는 x 변수가 y 변수의 분산을 얼마나 줄였는가이고 회귀모델에서 종속변수가 독립변수를 얼마나 설명해주는지 가리키는 지표입니다. 즉, R-squared 값이 1에 가까우면 데이터를 잘 설명하는 모델이고 0에 가까울수록 설명을 못하는 모델이라고 생각할 수 있다.

```
print("X1의 경우 모델 평가 결과 : ",model1.score(X1_test,y1_test).round(3))
print("X2의 경우 모델 평가 결과 : ",model2.score(X2_test,y2_test).round(3))
print("X3의 경우 모델 평가 결과 : ",model3.score(X3_test,y3_test).round(3))
print("X4의 경우 모델 평가 결과 : ",model4.score(X4_test,y4_test).round(3))
```

```
print("X5의 경우 모델 평가 결과 : ",model5.score(X5_test,y5_test).round(3))
print("X6의 경우 모델 평가 결과 : ",model6.score(X6_test,y6_test).round(3))
```

모델 평가를 하기 위해서 결정계수를 구해주는 score 메서드를 활용해서 결과를 출력했습니다. 결과는 아래와 같습니다.

```
X1의 경우 모델 평가 결과 :  0.738
X2의 경우 모델 평가 결과 :  0.735
X3의 경우 모델 평가 결과 :  0.74
X4의 경우 모델 평가 결과 :  0.732
X5의 경우 모델 평가 결과 :  0.719
X6의 경우 모델 평가 결과 :  0.705
```

Random state 제거 후 평가 결과

```
X1의 경우 모델 평가 결과 :  0.852
X2의 경우 모델 평가 결과 :  0.74
X3의 경우 모델 평가 결과 :  0.807
X4의 경우 모델 평가 결과 :  0.772
X5의 경우 모델 평가 결과 :  0.706
X6의 경우 모델 평가 결과 :  0.527
```

```
X1의 경우 모델 평가 결과 :  0.831
X2의 경우 모델 평가 결과 :  0.804
X3의 경우 모델 평가 결과 :  0.725
X4의 경우 모델 평가 결과 :  0.782
X5의 경우 모델 평가 결과 :  0.679
X6의 경우 모델 평가 결과 :  0.739
```

```
X1의 경우 모델 평가 결과 :  0.829
X2의 경우 모델 평가 결과 :  0.729
X3의 경우 모델 평가 결과 :  0.782
X4의 경우 모델 평가 결과 :  0.631
X5의 경우 모델 평가 결과 :  0.761
X6의 경우 모델 평가 결과 :  0.755
```

random state 값을 42으로 설정 했을때는 X3인 경우가 가장 결과가 좋았다. 하지만 random state를 제거후 총 3번의 추가적인 모델 학습 및 평가 결과 X1의 경우가 일반적으로 좋았습니다. 즉, corr_ten_idx 리스트의 해당하는 상관관계가 높은 9가지 feature를 모두 포함했을때 결과가 가장 좋았습니다.