```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# 데이터 불러오기
train_df = pd.read_csv('/content/drive/MyDrive/data/open_train.csv')
test_df = pd.read_csv('/content/drive/MyDrive/data/open_test.csv')

train_df.head() #확인
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...

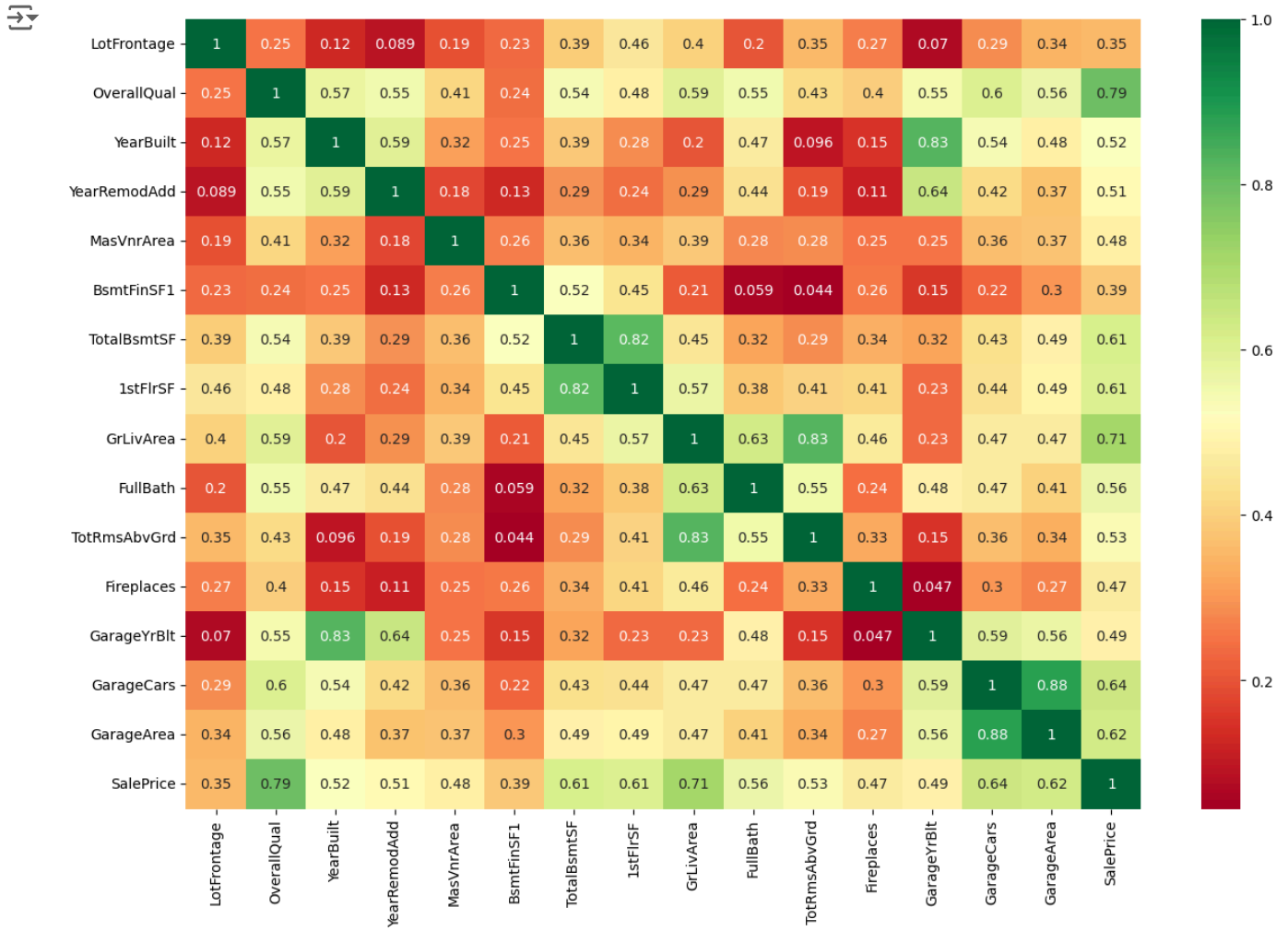
5 rows × 81 columns

```
#index 설
train_df.set_index('Id', inplace=True)
test_df.set_index('Id', inplace=True)
len_train_df = len(train_df)
len_test_df = len(test_df)
```

```
numeric_features = train_df.select_dtypes(include=['number'])
corrmat = numeric_features.corr()
top_corr_features = corrmat.index[abs(corrmat["SalePrice"]) >= 0.35]
top_corr_features
```

```
Index(['LotFrontage', 'OverallQual', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea',
       'BsmtFinSF1', 'TotalBsmtSF', '1stFlrSF', 'GrLivArea', 'FullBath',
       'TotRmsAbvGrd', 'Fireplaces', 'GarageYrBlt', 'GarageCars', 'GarageArea',
       'SalePrice'],
      dtype='object')
```

```
# heatmap 생성
plt.figure(figsize=(15,10))
g = sns.heatmap(train_df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```




```
#y_label 분리하기
train_y_label = train_df['SalePrice']
train_df.drop(['SalePrice'], axis=1, inplace=True)
```

```
#train, test 병합하기
boston_df = pd.concat((train_df, test_df), axis=0)
boston_df_index = boston_df.index
print('Length of Boston Dataset : ',len(boston_df))
boston_df.head()
```

```
Length of Boston Dataset :  2919
```

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConf
1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Insi
2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	F
3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Insi
4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corr
5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	F

5 rows × 79 columns

```
# 결측치 확인
check_null = boston_df.isna().sum() / len(boston_df)
```

```
# 결측 비율이 0.5 이상인 열
check_null[check_null >= 0.5]
```

	0
Alley	0.932169
MasVnrType	0.605002
PoolQC	0.996574
Fence	0.804385
MiscFeature	0.964029

dtype: float64

```
# 결측 비율이 0.5 이상인 열 제거
remove_cols = check_null[check_null >= 0.5].keys()
boston_df = boston_df.drop(remove_cols, axis=1)

boston_df.head()
```

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	Lan
1	60	RL	65.0	8450	Pave	Reg	Lvl	AllPub	Inside	
2	20	RL	80.0	9600	Pave	Reg	Lvl	AllPub	FR2	
3	60	RL	68.0	11250	Pave	IR1	Lvl	AllPub	Inside	
4	70	RL	60.0	9550	Pave	IR1	Lvl	AllPub	Corner	
5	60	RL	84.0	14260	Pave	IR1	Lvl	AllPub	FR2	

5 rows × 74 columns

```
# 카테고리형 분리
boston_obj_df = boston_df.select_dtypes(include='object')   # 카테고리형
print('Object type columns:₩n',boston_obj_df.columns)
```

```
Object type columns:
 Index(['MSZoning', 'Street', 'LotShape', 'LandContour', 'Utilities',
       'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
       'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st',
       'Exterior2nd', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional',
       'FireplaceQu', 'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond',
       'PavedDrive', 'SaleType', 'SaleCondition'],
      dtype='object')
```

```
# 수치형 분리
boston_num_df = boston_df.select_dtypes(exclude='object')   # 수치형
print('Numeric type columns:₩n',boston_num_df.columns)
```

```
Numeric type columns:
 Index(['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond',
       'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2',
       'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF',
       'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath',
       'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces',
       'GarageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal',
       'MoSold', 'YrSold'],
      dtype='object')
```

```
boston_dummy_df = pd.get_dummies(boston_obj_df, drop_first=True)
boston_dummy_df.index = boston_df_index
boston_dummy_df.head()
```

Id	MSZoning_FV	MSZoning_RH	MSZoning_RL	MSZoning_RM	Street_Pave	LotShape_IR2	LotShape_IR3	LotShap
1	False	False	True	False	True	False	False	
2	False	False	True	False	True	False	False	
3	False	False	True	False	True	False	False	
4	False	False	True	False	True	False	False	
5	False	False	True	False	True	False	False	

5 rows × 197 columns

```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='mean')
imputer.fit(boston_num_df)
boston_num_df_ = imputer.transform(boston_num_df)
boston_num_df = pd.DataFrame(boston_num_df_, columns=boston_num_df.columns, index=boston_df_index)
boston_num_df.head()
```

Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	Bsm
1	60.0	65.0	8450.0	7.0	5.0	2003.0	2003.0	196.0	
2	20.0	80.0	9600.0	6.0	8.0	1976.0	1976.0	0.0	
3	60.0	68.0	11250.0	7.0	5.0	2001.0	2002.0	162.0	
4	70.0	60.0	9550.0	7.0	5.0	1915.0	1970.0	0.0	
5	60.0	84.0	14260.0	8.0	5.0	2000.0	2000.0	350.0	

5 rows × 36 columns

```
boston_df = pd.merge(boston_dummy_df, boston_num_df, left_index=True, right_index=True)
boston_df.head()
```

Id	MSZoning_FV	MSZoning_RH	MSZoning_RL	MSZoning_RM	Street_Pave	LotShape_IR2	LotShape_IR3	LotShap
1	False	False	True	False	True	False	False	
2	False	False	True	False	True	False	False	
3	False	False	True	False	True	False	False	
4	False	False	True	False	True	False	False	
5	False	False	True	False	True	False	False	

5 rows × 233 columns

```
train_df = boston_df[:len_train_df]
test_df = boston_df[len_train_df:]

train_df['SalePrice'] = train_y_label
```

```
print('train set length: ',len(train_df))
print('test set length: ',len(test_df))
```

```
train set length:  1460
test set length:  1459
<ipython-input-16-5b940b3cbdc6>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
  train_df['SalePrice'] = train_y_label
```

```
from sklearn.model_selection import train_test_split

X_train = train_df.drop(['SalePrice'], axis=1)
y_train = train_df['SalePrice']

X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, shuffle=True)

X_test = test_df
test_id_idx = test_df.index



print('X_train : ',len(X_train))
print('X_val : ',len(X_val))
print('X_test :',len(X_test))
```

```
X_train :  1168
X_val :  292
X_test : 1459
```

```
from sklearn.model_selection import GridSearchCV
import xgboost as xgb


param = {
    'max_depth':[2,3,4],
    'n_estimators':range(550,700,50),
    'colsample_bytree':[0.5,0.7,1],
    'colsample_bylevel':[0.5,0.7,1],
}


model = xgb.XGBRegressor()
grid_search = GridSearchCV(estimator=model, param_grid=param, cv=5,
                           scoring='neg_mean_squared_error',
                           n_jobs=-1)


grid_search.fit(X_train, y_train)
```

```
▸ GridSearchCV
▸ best_estimator_: XGBRegressor
▸ XGBRegressor
```

```
print(grid_search.best_params_)
print(grid_search.best_estimator_)
```

```
{'colsample_bylevel': 0.5, 'colsample_bytree': 0.5, 'max_depth': 3, 'n_estimators': 550}
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=0.5, colsample_bynode=None, colsample_bytree=0.5,
             device=None, early_stopping_rounds=None, enable_categorical=False,
```

```
                       eval_metric=None, feature_types=None, gamma=None, grow_policy=None,
                       importance_type=None, interaction_constraints=None,
                       learning_rate=None, max_bin=None, max_cat_threshold=None,
                       max_cat_to_onehot=None, max_delta_step=None, max_depth=3,
                       max_leaves=None, min_child_weight=None, missing=nan,
                       monotone_constraints=None, multi_strategy=None, n_estimators=550,
                       n_jobs=None, num_parallel_tree=None, random_state=None, ...)
```

```
from sklearn.metrics import mean_squared_error, mean_absolute_error
pred_train = grid_search.predict(X_train)
pred_val = grid_search.predict(X_val)
```

```
print('train mae score: ', mean_absolute_error(y_train, pred_train))
print('val mae score:', mean_absolute_error(y_val, pred_val))
```

```
train mae score:  1624.049858197774
val mae score: 16875.79366438356
```

```
test_y_pred = grid_search.predict(X_test)
id_pred_df = pd.DataFrame()
id_pred_df['Id'] = test_id_idx
id_pred_df['SalePrice'] = test_y_pred
id_pred_df
```

	Id	SalePrice
0	1461	130890.179688
1	1462	173872.281250
2	1463	189052.437500
3	1464	197206.156250
4	1465	165175.906250
...	...	...
1454	2915	77812.890625
1455	2916	62406.050781
1456	2917	159629.171875
1457	2918	114051.226562
1458	2919	205811.656250

1459 rows × 2 columns

다음 단계: 추천 차트 보기 New interactive sheet

```
plt.figure(figsize=(15,15))
plt.plot(range(0, len(y_val)), y_val,'o-', label='Validation Actual')
plt.plot(range(0, len(pred_val)), pred_val, '-', label='Validation Predict')
plt.title('Prediction of House Prices')
plt.ylabel('Prices')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7e7eaf34d2a0>
```