Kaggle Submission Relative to Peers

Submission and Description	Public Score
house_rf_xgb.csv a few seconds ago by Anaswar Jayakumar MSDS 422 - Predicting house prices using XGBoost Random Forests	0.16658
house_xgb.csv 2 hours ago by Anaswar Jayakumar MSDS 422 - Predicting house prices	0.13599
house_xgb.csv 8 hours ago by Anaswar Jayakumar MSDS 422 - Predicting house prices with XGBoost and just log columns	0.13552
house_xgb.csv 8 hours ago by Anaswar Jayakumar MSDS 422 - XGBoost with Log features	0.14208
house_xgb.csv 8 hours ago by Anaswar Jayakumar MSDS 422 - Predicting house prices	0.13798

MSDS 422 Assignment 4

April 25, 2021

1 Assignment 4: Random Forests and Gradient Boosting

Compete in the Kaggle.com House Prices: Advanced Regression Techniques competition located here.

Investigate many variables. Employ at least two regression modeling methods selected from those discussed in Chapter 4 of the Géron (2017) textbook: linear regression, stochastic gradient descent, ridge regression, lasso regression, and elastic net. Also employ random forests to the regression problem, following methods described in Géron (2017) Chapter 7. Evaluate these methods within a cross-validation design, using root mean-squared error (RMSE) as an index of prediction error. Submit at least two models to Kaggle.com for evaluation.

Try alternative versions of random forests and gradient boosting. Select a best modeling method . Employ that method on the full data set, obtaining results that you can report to management.

Regarding the management problem, imagine that you again are advising a real estate brokerage firm in its attempt to employ machine learning methods. The firm wants to use machine learning to complement conventional methods for assessing the market value of residential real estate. Of the modeling methods examined in your study, which would you recommend to management and why? Reviewing the results of the random forests and gradient boosting model you have selected to present to management, which explanatory variables are most important in predicting home prices?

For all Kaggle competitions, you must submit a screen snapshot that identifies you along with your scores on the submissions. Submit your work as a single .pdf file that is legible. Include your code as an appendix. Look at the rubric to see how you will be graded. Your work will be compared against your peers on the performance metric(s).

Descriptive Features - SalePrice : The property's sale price in dollars. This is the target variable that you're trying to predict. - MSSubClass: The building class - MSZoning: The general zoning classification - LotFrontage: Linear feet of street connected to property - LotArea: Lot size in square feet - Street: Type of road access - Alley: Type of alley access - LotShape: General shape of property - LandContour: Flatness of the property - Utilities: Type of utilities available - LotConfig: Lot configuration - LandSlope: Slope of property - Neighborhood: Physical locations within Ames city limits - Condition1: Proximity to main road or railroad - Condition2: Proximity to main road or railroad (if a second is present) - BldgType: Type of dwelling - HouseStyle: Style of dwelling - OverallQual: Overall material and finish quality - OverallCond: Overall condition rating - YearBuilt: Original construction date - YearRemodAdd: Remodel date - RoofStyle: Type of roof - RoofMatl: Roof material - Exterior1st: Exterior covering on house - Exterior2nd: Exterior covering on house (if more than one material) - MasVnrType: Masonry veneer type - MasVnrArea: Masonry veneer area in square feet - ExterQual: Exterior material quality - ExterCond: Present condition of the material on the exterior - Foundation: Type of foundation - BsmtQual: Height of the basement - BsmtCond: General condition of the basement - BsmtExposure: Walkout or garden level basement walls - BsmtFinType1: Quality of basement finished area - BsmtFinSF1: Type 1 finished square feet - BsmtFinType2: Quality of second finished area (if present) - BsmtFinSF2: Type 2 finished square feet - BsmtUnfSF: Unfinished square feet of basement area - TotalBsmtSF: Total square feet of basement area - Heating: Type of heating - HeatingQC: Heating quality and condition - CentralAir: Central air conditioning - Electrical: Electrical system - 1stFlrSF: First Floor square feet - 2ndFlrSF: Second floor square feet - LowQualFinSF: Low quality finished square feet (all floors) - GrLivArea: Above grade (ground) living area square feet - BsmtFullBath: Basement full bathrooms - BsmtHalfBath: Basement half bathrooms - FullBath: Full bathrooms above grade - HalfBath: Half baths above grade - Bedroom: Number of bedrooms above basement level - Kitchen: Number of kitchens - KitchenQual: Kitchen quality - TotRmsAbvGrd: Total rooms above grade (does not include bathrooms) - Functional: Home functionality rating - Fireplaces: Number of fireplaces - FireplaceQu: Fireplace quality - GarageType: Garage location - GarageYrBlt: Year garage was built - GarageFinish: Interior finish of the garage - GarageCars: Size of garage in car capacity - GarageArea: Size of garage in square feet - GarageQual: Garage quality - GarageCond: Garage condition - PavedDrive: Paved driveway - WoodDeckSF: Wood deck area in square feet - OpenPorchSF: Open porch area in square feet - EnclosedPorch: Enclosed porch area in square feet - 3SsnPorch: Three season porch area in square feet - ScreenPorch: Screen porch area in square feet - PoolArea: Pool area in square feet - PoolQC: Pool quality - Fence: Fence quality - MiscFeature: Miscellaneous feature not covered in other categories - MiscVal: Value of miscellaneous feature - MoSold: Month Sold - YrSold: Year Sold - SaleType: Type of sale - SaleCondition: Condition of sale

[230]:
```
# main libraries
import os
import re
import pickle
import numpy as np
import pandas as pd

# visualization libraries
import matplotlib.pyplot as plt
import seaborn as sns
import plotly
import plotly.graph_objs as go
import plotly.io as pio
from plotly.subplots import make_subplots
import plotly.express as px
from plotly.offline import iplot, init_notebook_mode
import cufflinks as cf

# machine learning libraries:
from sklearn.model_selection import StratifiedKFold, cross_validate,␣
↪cross_val_score, train_test_split
from sklearn.preprocessing  import StandardScaler
```

```
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from xgboost import XGBClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import KNNImputer, IterativeImputer
from sklearn.ensemble import BaggingClassifier,␣
↪AdaBoostClassifier,GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
import warnings
warnings.filterwarnings("ignore")

# set some display options:
plt.rcParams['figure.dpi'] = 100
colors = px.colors.qualitative.Prism
pio.templates.default = "plotly_white"
```

2 Data Preprocessing

[231]:
```
house_train = pd.read_csv('/Users/anaswarjayakumar/Downloads/train (2).csv')
house_test = pd.read_csv('/Users/anaswarjayakumar/Downloads/test (2).csv')
```

[232]:
```
house_train.head()
```

[232]:
```
   Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
0   1          60       RL      65.0000     8450   Pave   NaN      Reg
1   2          20       RL      80.0000     9600   Pave   NaN      Reg
2   3          60       RL      68.0000    11250   Pave   NaN      IR1
3   4          70       RL      60.0000     9550   Pave   NaN      IR1
4   5          60       RL      84.0000    14260   Pave   NaN      IR1

  LandContour Utilities  … PoolArea PoolQC Fence MiscFeature MiscVal MoSold  \
0         Lvl    AllPub  …        0    NaN   NaN         NaN       0      2
1         Lvl    AllPub  …        0    NaN   NaN         NaN       0      5
2         Lvl    AllPub  …        0    NaN   NaN         NaN       0      9
3         Lvl    AllPub  …        0    NaN   NaN         NaN       0      2
4         Lvl    AllPub  …        0    NaN   NaN         NaN       0     12

  YrSold  SaleType  SaleCondition  SalePrice
0   2008        WD         Normal     208500
1   2007        WD         Normal     181500
2   2008        WD         Normal     223500
```

```
3   2006         WD        Abnorml      140000
4   2008         WD         Normal      250000

[5 rows x 81 columns]
```

[233]:
```
house_test.head()
```

[233]:
```
     Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
0  1461          20       RH      80.0000    11622   Pave   NaN      Reg
1  1462          20       RL      81.0000    14267   Pave   NaN      IR1
2  1463          60       RL      74.0000    13830   Pave   NaN      IR1
3  1464          60       RL      78.0000     9978   Pave   NaN      IR1
4  1465         120       RL      43.0000     5005   Pave   NaN      IR1

  LandContour Utilities  … ScreenPorch PoolArea PoolQC  Fence MiscFeature  \
0         Lvl    AllPub  …         120        0    NaN  MnPrv         NaN
1         Lvl    AllPub  …           0        0    NaN    NaN        Gar2
2         Lvl    AllPub  …           0        0    NaN  MnPrv         NaN
3         Lvl    AllPub  …           0        0    NaN    NaN         NaN
4         HLS    AllPub  …         144        0    NaN    NaN         NaN

  MiscVal MoSold  YrSold  SaleType  SaleCondition
0       0      6    2010        WD         Normal
1   12500      6    2010        WD         Normal
2       0      3    2010        WD         Normal
3       0      6    2010        WD         Normal
4       0      1    2010        WD         Normal

[5 rows x 80 columns]
```

[234]:
```
ignore_cols = ['Id', 'LotFrontage', 'Alley', 'PoolQC', 'Fence', 'MiscFeature',␣
 ↪'SalePrice']
```

[235]:
```
house_train
```

[235]:
```
        Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
0        1          60       RL      65.0000     8450   Pave   NaN      Reg
1        2          20       RL      80.0000     9600   Pave   NaN      Reg
2        3          60       RL      68.0000    11250   Pave   NaN      IR1
3        4          70       RL      60.0000     9550   Pave   NaN      IR1
4        5          60       RL      84.0000    14260   Pave   NaN      IR1
…     …         …       …          …        …      …     …        …
1455  1456          60       RL      62.0000     7917   Pave   NaN      Reg
1456  1457          20       RL      85.0000    13175   Pave   NaN      Reg
1457  1458          70       RL      66.0000     9042   Pave   NaN      Reg
1458  1459          20       RL      68.0000     9717   Pave   NaN      Reg
1459  1460          20       RL      75.0000     9937   Pave   NaN      Reg
```

```
     LandContour Utilities  … PoolArea PoolQC  Fence MiscFeature MiscVal  \
0            Lvl    AllPub  …        0    NaN    NaN         NaN       0
1            Lvl    AllPub  …        0    NaN    NaN         NaN       0
2            Lvl    AllPub  …        0    NaN    NaN         NaN       0
3            Lvl    AllPub  …        0    NaN    NaN         NaN       0
4            Lvl    AllPub  …        0    NaN    NaN         NaN       0
…          …        …  …      …     …    …         …     …
1455         Lvl    AllPub  …        0    NaN    NaN         NaN       0
1456         Lvl    AllPub  …        0    NaN  MnPrv         NaN       0
1457         Lvl    AllPub  …        0    NaN  GdPrv        Shed    2500
1458         Lvl    AllPub  …        0    NaN    NaN         NaN       0
1459         Lvl    AllPub  …        0    NaN    NaN         NaN       0

     MoSold YrSold  SaleType  SaleCondition  SalePrice
0         2   2008        WD         Normal     208500
1         5   2007        WD         Normal     181500
2         9   2008        WD         Normal     223500
3         2   2006        WD        Abnorml     140000
4        12   2008        WD         Normal     250000
…      …    …         …             …      …
1455      8   2007        WD         Normal     175000
1456      2   2010        WD         Normal     210000
1457      5   2010        WD         Normal     266500
1458      4   2010        WD         Normal     142125
1459      6   2008        WD         Normal     147500

[1460 rows x 81 columns]
```

[236]:
```
house_test
```

[236]:
```
        Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
0     1461          20       RH      80.0000    11622   Pave   NaN      Reg
1     1462          20       RL      81.0000    14267   Pave   NaN      IR1
2     1463          60       RL      74.0000    13830   Pave   NaN      IR1
3     1464          60       RL      78.0000     9978   Pave   NaN      IR1
4     1465         120       RL      43.0000     5005   Pave   NaN      IR1
…      …           …       …          …      …   …    …       …
1454  2915         160       RM      21.0000     1936   Pave   NaN      Reg
1455  2916         160       RM      21.0000     1894   Pave   NaN      Reg
1456  2917          20       RL     160.0000    20000   Pave   NaN      Reg
1457  2918          85       RL      62.0000    10441   Pave   NaN      Reg
1458  2919          60       RL      74.0000     9627   Pave   NaN      Reg

     LandContour Utilities  … ScreenPorch PoolArea PoolQC  Fence  \
0            Lvl    AllPub  …         120        0    NaN  MnPrv
1            Lvl    AllPub  …           0        0    NaN    NaN
```

```
2             Lvl    AllPub  …          0         0    NaN  MnPrv
3             Lvl    AllPub  …          0         0    NaN    NaN
4             HLS    AllPub  …        144         0    NaN    NaN
…           …      … …        …       …    …    …
1454          Lvl    AllPub  …          0         0    NaN    NaN
1455          Lvl    AllPub  …          0         0    NaN    NaN
1456          Lvl    AllPub  …          0         0    NaN    NaN
1457          Lvl    AllPub  …          0         0    NaN  MnPrv
1458          Lvl    AllPub  …          0         0    NaN    NaN

     MiscFeature MiscVal  MoSold  YrSold  SaleType  SaleCondition
0            NaN       0       6    2010        WD         Normal
1           Gar2   12500       6    2010        WD         Normal
2            NaN       0       3    2010        WD         Normal
3            NaN       0       6    2010        WD         Normal
4            NaN       0       1    2010        WD         Normal
…          …     …    …     …      …         …
1454         NaN       0       6    2006        WD         Normal
1455         NaN       0       4    2006        WD        Abnorml
1456         NaN       0       9    2006        WD        Abnorml
1457        Shed     700       7    2006        WD         Normal
1458         NaN       0      11    2006        WD         Normal

[1459 rows x 80 columns]
```

[237]:
```
house_train_n = house_train[[c for c in house_train.columns if house_train[c].
 ↪dtypes!='O']].copy()
house_train_c = house_train[[c for c in house_train.columns if house_train[c].
 ↪dtypes=='O']].copy()

house_test_n = house_test[[c for c in house_test.columns if house_test[c].
 ↪dtypes!='O']].copy()
house_test_c = house_test[[c for c in house_test.columns if house_test[c].
 ↪dtypes=='O']].copy()
```

[238]:
```
house_train_n.columns
```

[238]:
```
Index(['Id', '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', 'SalePrice'],
      dtype='object')
```

[239]:
```
house_train_c.columns
```

[239]:
```
Index(['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities',
       'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
       'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st',
       'Exterior2nd', 'MasVnrType', 'ExterQual', 'ExterCond', 'Foundation',
       'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',
       'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual',
       'Functional', 'FireplaceQu', 'GarageType', 'GarageFinish', 'GarageQual',
       'GarageCond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature',
       'SaleType', 'SaleCondition'],
      dtype='object')
```

2.1 Data Exploration/Analysis

[240]:
```
# see information about the data
house_train.info()
print('_'*40)
house_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   Alley          91 non-null     object
 7   LotShape       1460 non-null   object
 8   LandContour    1460 non-null   object
 9   Utilities      1460 non-null   object
 10  LotConfig      1460 non-null   object
 11  LandSlope      1460 non-null   object
 12  Neighborhood   1460 non-null   object
 13  Condition1     1460 non-null   object
 14  Condition2     1460 non-null   object
 15  BldgType       1460 non-null   object
 16  HouseStyle     1460 non-null   object
 17  OverallQual    1460 non-null   int64
 18  OverallCond    1460 non-null   int64
 19  YearBuilt      1460 non-null   int64
 20  YearRemodAdd   1460 non-null   int64
 21  RoofStyle      1460 non-null   object
 22  RoofMatl       1460 non-null   object
```

23	Exterior1st	1460 non-null	object
24	Exterior2nd	1460 non-null	object
25	MasVnrType	1452 non-null	object
26	MasVnrArea	1452 non-null	float64
27	ExterQual	1460 non-null	object
28	ExterCond	1460 non-null	object
29	Foundation	1460 non-null	object
30	BsmtQual	1423 non-null	object
31	BsmtCond	1423 non-null	object
32	BsmtExposure	1422 non-null	object
33	BsmtFinType1	1423 non-null	object
34	BsmtFinSF1	1460 non-null	int64
35	BsmtFinType2	1422 non-null	object
36	BsmtFinSF2	1460 non-null	int64
37	BsmtUnfSF	1460 non-null	int64
38	TotalBsmtSF	1460 non-null	int64
39	Heating	1460 non-null	object
40	HeatingQC	1460 non-null	object
41	CentralAir	1460 non-null	object
42	Electrical	1459 non-null	object
43	1stFlrSF	1460 non-null	int64
44	2ndFlrSF	1460 non-null	int64
45	LowQualFinSF	1460 non-null	int64
46	GrLivArea	1460 non-null	int64
47	BsmtFullBath	1460 non-null	int64
48	BsmtHalfBath	1460 non-null	int64
49	FullBath	1460 non-null	int64
50	HalfBath	1460 non-null	int64
51	BedroomAbvGr	1460 non-null	int64
52	KitchenAbvGr	1460 non-null	int64
53	KitchenQual	1460 non-null	object
54	TotRmsAbvGrd	1460 non-null	int64
55	Functional	1460 non-null	object
56	Fireplaces	1460 non-null	int64
57	FireplaceQu	770 non-null	object
58	GarageType	1379 non-null	object
59	GarageYrBlt	1379 non-null	float64
60	GarageFinish	1379 non-null	object
61	GarageCars	1460 non-null	int64
62	GarageArea	1460 non-null	int64
63	GarageQual	1379 non-null	object
64	GarageCond	1379 non-null	object
65	PavedDrive	1460 non-null	object
66	WoodDeckSF	1460 non-null	int64
67	OpenPorchSF	1460 non-null	int64
68	EnclosedPorch	1460 non-null	int64
69	3SsnPorch	1460 non-null	int64
70	ScreenPorch	1460 non-null	int64

```
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
----------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1459 entries, 0 to 1458
Data columns (total 80 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1459 non-null   int64
 1   MSSubClass     1459 non-null   int64
 2   MSZoning       1455 non-null   object
 3   LotFrontage    1232 non-null   float64
 4   LotArea        1459 non-null   int64
 5   Street         1459 non-null   object
 6   Alley          107 non-null    object
 7   LotShape       1459 non-null   object
 8   LandContour    1459 non-null   object
 9   Utilities      1457 non-null   object
 10  LotConfig      1459 non-null   object
 11  LandSlope      1459 non-null   object
 12  Neighborhood   1459 non-null   object
 13  Condition1     1459 non-null   object
 14  Condition2     1459 non-null   object
 15  BldgType       1459 non-null   object
 16  HouseStyle     1459 non-null   object
 17  OverallQual    1459 non-null   int64
 18  OverallCond    1459 non-null   int64
 19  YearBuilt      1459 non-null   int64
 20  YearRemodAdd   1459 non-null   int64
 21  RoofStyle      1459 non-null   object
 22  RoofMatl       1459 non-null   object
 23  Exterior1st    1458 non-null   object
 24  Exterior2nd    1458 non-null   object
 25  MasVnrType     1443 non-null   object
 26  MasVnrArea     1444 non-null   float64
 27  ExterQual      1459 non-null   object
 28  ExterCond      1459 non-null   object
 29  Foundation     1459 non-null   object
```

```
30  BsmtQual       1415 non-null   object
31  BsmtCond       1414 non-null   object
32  BsmtExposure   1415 non-null   object
33  BsmtFinType1   1417 non-null   object
34  BsmtFinSF1     1458 non-null   float64
35  BsmtFinType2   1417 non-null   object
36  BsmtFinSF2     1458 non-null   float64
37  BsmtUnfSF      1458 non-null   float64
38  TotalBsmtSF    1458 non-null   float64
39  Heating        1459 non-null   object
40  HeatingQC      1459 non-null   object
41  CentralAir     1459 non-null   object
42  Electrical     1459 non-null   object
43  1stFlrSF       1459 non-null   int64
44  2ndFlrSF       1459 non-null   int64
45  LowQualFinSF   1459 non-null   int64
46  GrLivArea      1459 non-null   int64
47  BsmtFullBath   1457 non-null   float64
48  BsmtHalfBath   1457 non-null   float64
49  FullBath       1459 non-null   int64
50  HalfBath       1459 non-null   int64
51  BedroomAbvGr   1459 non-null   int64
52  KitchenAbvGr   1459 non-null   int64
53  KitchenQual    1458 non-null   object
54  TotRmsAbvGrd   1459 non-null   int64
55  Functional     1457 non-null   object
56  Fireplaces     1459 non-null   int64
57  FireplaceQu    729 non-null    object
58  GarageType     1383 non-null   object
59  GarageYrBlt    1381 non-null   float64
60  GarageFinish   1381 non-null   object
61  GarageCars     1458 non-null   float64
62  GarageArea     1458 non-null   float64
63  GarageQual     1381 non-null   object
64  GarageCond     1381 non-null   object
65  PavedDrive     1459 non-null   object
66  WoodDeckSF     1459 non-null   int64
67  OpenPorchSF    1459 non-null   int64
68  EnclosedPorch  1459 non-null   int64
69  3SsnPorch      1459 non-null   int64
70  ScreenPorch    1459 non-null   int64
71  PoolArea       1459 non-null   int64
72  PoolQC         3 non-null      object
73  Fence          290 non-null    object
74  MiscFeature    51 non-null     object
75  MiscVal        1459 non-null   int64
76  MoSold         1459 non-null   int64
77  YrSold         1459 non-null   int64
```

```
 78  SaleType       1458 non-null   object
 79  SaleCondition  1459 non-null   object
dtypes: float64(11), int64(26), object(43)
memory usage: 912.0+ KB
```

[241]:

```python
#join all the data together
full_df = pd.concat([house_train, house_test]).reset_index(drop=True)
del full_df['SalePrice']
```

[242]:

```python
#finding the unique values in each column (type object)
for col in house_train.select_dtypes('O').columns:
    print('We have {} unique values in {} column : {}'.format(len(full_df[col].
 ↪unique()),col,
                                                              full_df[col].
 ↪unique().tolist()))
    print('__'*30)
```

```
We have 6 unique values in MSZoning column : ['RL', 'RM', 'C (all)', 'FV', 'RH',
nan]
____________________________________________________________
We have 2 unique values in Street column : ['Pave', 'Grvl']
____________________________________________________________
We have 3 unique values in Alley column : [nan, 'Grvl', 'Pave']
____________________________________________________________
We have 4 unique values in LotShape column : ['Reg', 'IR1', 'IR2', 'IR3']
____________________________________________________________
We have 4 unique values in LandContour column : ['Lvl', 'Bnk', 'Low', 'HLS']
____________________________________________________________
We have 3 unique values in Utilities column : ['AllPub', 'NoSeWa', nan]
____________________________________________________________
We have 5 unique values in LotConfig column : ['Inside', 'FR2', 'Corner',
'CulDSac', 'FR3']
____________________________________________________________
We have 3 unique values in LandSlope column : ['Gtl', 'Mod', 'Sev']
____________________________________________________________
We have 25 unique values in Neighborhood column : ['CollgCr', 'Veenker',
'Crawfor', 'NoRidge', 'Mitchel', 'Somerst', 'NWAmes', 'OldTown', 'BrkSide',
'Sawyer', 'NridgHt', 'NAmes', 'SawyerW', 'IDOTRR', 'MeadowV', 'Edwards',
'Timber', 'Gilbert', 'StoneBr', 'ClearCr', 'NPkVill', 'Blmngtn', 'BrDale',
'SWISU', 'Blueste']
____________________________________________________________
We have 9 unique values in Condition1 column : ['Norm', 'Feedr', 'PosN',
'Artery', 'RRAe', 'RRNn', 'RRAn', 'PosA', 'RRNe']
____________________________________________________________
We have 8 unique values in Condition2 column : ['Norm', 'Artery', 'RRNn',
'Feedr', 'PosN', 'PosA', 'RRAn', 'RRAe']
____________________________________________________________
We have 5 unique values in BldgType column : ['1Fam', '2fmCon', 'Duplex',
```

```
'TwnhsE', 'Twnhs']
------------------------------------------------------------
We have 8 unique values in HouseStyle column : ['2Story', '1Story', '1.5Fin',
'1.5Unf', 'SFoyer', 'SLvl', '2.5Unf', '2.5Fin']
------------------------------------------------------------
We have 6 unique values in RoofStyle column : ['Gable', 'Hip', 'Gambrel',
'Mansard', 'Flat', 'Shed']
------------------------------------------------------------
We have 8 unique values in RoofMatl column : ['CompShg', 'WdShngl', 'Metal',
'WdShake', 'Membran', 'Tar&Grv', 'Roll', 'ClyTile']
------------------------------------------------------------
We have 16 unique values in Exterior1st column : ['VinylSd', 'MetalSd', 'Wd
Sdng', 'HdBoard', 'BrkFace', 'WdShing', 'CemntBd', 'Plywood', 'AsbShng',
'Stucco', 'BrkComm', 'AsphShn', 'Stone', 'ImStucc', 'CBlock', nan]
------------------------------------------------------------
We have 17 unique values in Exterior2nd column : ['VinylSd', 'MetalSd', 'Wd
Shng', 'HdBoard', 'Plywood', 'Wd Sdng', 'CmentBd', 'BrkFace', 'Stucco',
'AsbShng', 'Brk Cmn', 'ImStucc', 'AsphShn', 'Stone', 'Other', 'CBlock', nan]
------------------------------------------------------------
We have 5 unique values in MasVnrType column : ['BrkFace', 'None', 'Stone',
'BrkCmn', nan]
------------------------------------------------------------
We have 4 unique values in ExterQual column : ['Gd', 'TA', 'Ex', 'Fa']
------------------------------------------------------------
We have 5 unique values in ExterCond column : ['TA', 'Gd', 'Fa', 'Po', 'Ex']
------------------------------------------------------------
We have 6 unique values in Foundation column : ['PConc', 'CBlock', 'BrkTil',
'Wood', 'Slab', 'Stone']
------------------------------------------------------------
We have 5 unique values in BsmtQual column : ['Gd', 'TA', 'Ex', nan, 'Fa']
------------------------------------------------------------
We have 5 unique values in BsmtCond column : ['TA', 'Gd', nan, 'Fa', 'Po']
------------------------------------------------------------
We have 5 unique values in BsmtExposure column : ['No', 'Gd', 'Mn', 'Av', nan]
------------------------------------------------------------
We have 7 unique values in BsmtFinType1 column : ['GLQ', 'ALQ', 'Unf', 'Rec',
'BLQ', nan, 'LwQ']
------------------------------------------------------------
We have 7 unique values in BsmtFinType2 column : ['Unf', 'BLQ', nan, 'ALQ',
'Rec', 'LwQ', 'GLQ']
------------------------------------------------------------
We have 6 unique values in Heating column : ['GasA', 'GasW', 'Grav', 'Wall',
'OthW', 'Floor']
------------------------------------------------------------
We have 5 unique values in HeatingQC column : ['Ex', 'Gd', 'TA', 'Fa', 'Po']
------------------------------------------------------------
We have 2 unique values in CentralAir column : ['Y', 'N']
------------------------------------------------------------
```

```
We have 6 unique values in Electrical column : ['SBrkr', 'FuseF', 'FuseA',
'FuseP', 'Mix', nan]

------------------------------------------------------------
We have 5 unique values in KitchenQual column : ['Gd', 'TA', 'Ex', 'Fa', nan]

------------------------------------------------------------
We have 8 unique values in Functional column : ['Typ', 'Min1', 'Maj1', 'Min2',
'Mod', 'Maj2', 'Sev', nan]

------------------------------------------------------------
We have 6 unique values in FireplaceQu column : [nan, 'TA', 'Gd', 'Fa', 'Ex',
'Po']

------------------------------------------------------------
We have 7 unique values in GarageType column : ['Attchd', 'Detchd', 'BuiltIn',
'CarPort', nan, 'Basment', '2Types']

------------------------------------------------------------
We have 4 unique values in GarageFinish column : ['RFn', 'Unf', 'Fin', nan]

------------------------------------------------------------
We have 6 unique values in GarageQual column : ['TA', 'Fa', 'Gd', nan, 'Ex',
'Po']

------------------------------------------------------------
We have 6 unique values in GarageCond column : ['TA', 'Fa', nan, 'Gd', 'Po',
'Ex']

------------------------------------------------------------
We have 3 unique values in PavedDrive column : ['Y', 'N', 'P']

------------------------------------------------------------
We have 4 unique values in PoolQC column : [nan, 'Ex', 'Fa', 'Gd']

------------------------------------------------------------
We have 5 unique values in Fence column : [nan, 'MnPrv', 'GdWo', 'GdPrv',
'MnWw']

------------------------------------------------------------
We have 5 unique values in MiscFeature column : [nan, 'Shed', 'Gar2', 'Othr',
'TenC']

------------------------------------------------------------
We have 10 unique values in SaleType column : ['WD', 'New', 'COD', 'ConLD',
'ConLI', 'CWD', 'ConLw', 'Con', 'Oth', nan]

------------------------------------------------------------
We have 6 unique values in SaleCondition column : ['Normal', 'Abnorml',
'Partial', 'AdjLand', 'Alloca', 'Family']

------------------------------------------------------------
```

[243]:

```
# describe our data
house_train[house_train.select_dtypes(exclude='object').columns].describe().
 ↪style.\
background_gradient(axis=1,cmap=sns.light_palette('green', as_cmap=True))
```

[243]: <pandas.io.formats.style.Styler at 0x7feb487acf10>

[244]:
```
# lets see the correlation between columns and target column
corr = house_train.corr()
corr['SalePrice'].sort_values(ascending=False)[1:15].to_frame()\
.style.background_gradient(axis=1,cmap=sns.light_palette('green', as_cmap=True))
```

[244]:
```
<pandas.io.formats.style.Styler at 0x7feb5a4a2910>
```

[245]:
```
corrmat = house_train_n.corr()
# fig,ax=plt.subplots(figsize=(12,12))
# sns.heatmap(corrmat,vmax=.8, square=True,ax=ax,annot=True, fmt='.2f',␣
↪annot_kws={'size': 6})

corrmat
```

[245]:
```
                   Id  MSSubClass  LotFrontage  LotArea  OverallQual  \
Id             1.0000      0.0112      -0.0106  -0.0332      -0.0284
MSSubClass     0.0112      1.0000      -0.3863  -0.1398       0.0326
LotFrontage   -0.0106     -0.3863       1.0000   0.4261       0.2516
LotArea       -0.0332     -0.1398       0.4261   1.0000       0.1058
OverallQual   -0.0284      0.0326       0.2516   0.1058       1.0000
OverallCond    0.0126     -0.0593      -0.0592  -0.0056      -0.0919
YearBuilt     -0.0127      0.0279       0.1233   0.0142       0.5723
YearRemodAdd  -0.0220      0.0406       0.0889   0.0138       0.5507
MasVnrArea    -0.0503      0.0229       0.1935   0.1042       0.4119
BsmtFinSF1    -0.0050     -0.0698       0.2336   0.2141       0.2397
BsmtFinSF2    -0.0060     -0.0656       0.0499   0.1112      -0.0591
BsmtUnfSF     -0.0079     -0.1408       0.1326  -0.0026       0.3082
TotalBsmtSF   -0.0154     -0.2385       0.3921   0.2608       0.5378
1stFlrSF       0.0105     -0.2518       0.4572   0.2995       0.4762
2ndFlrSF       0.0056      0.3079       0.0802   0.0510       0.2955
LowQualFinSF  -0.0442      0.0465       0.0385   0.0048      -0.0304
GrLivArea      0.0083      0.0749       0.4028   0.2631       0.5930
BsmtFullBath   0.0023      0.0035       0.1009   0.1582       0.1111
BsmtHalfBath  -0.0202     -0.0023      -0.0072   0.0480      -0.0402
FullBath       0.0056      0.1316       0.1988   0.1260       0.5506
HalfBath       0.0068      0.1774       0.0535   0.0143       0.2735
BedroomAbvGr   0.0377     -0.0234       0.2632   0.1197       0.1017
KitchenAbvGr   0.0030      0.2817      -0.0061  -0.0178      -0.1839
TotRmsAbvGrd   0.0272      0.0404       0.3521   0.1900       0.4275
Fireplaces    -0.0198     -0.0456       0.2666   0.2714       0.3968
GarageYrBlt    0.0001      0.0851       0.0702  -0.0249       0.5478
GarageCars     0.0166     -0.0401       0.2857   0.1549       0.6007
GarageArea     0.0176     -0.0987       0.3450   0.1804       0.5620
WoodDeckSF    -0.0296     -0.0126       0.0885   0.1717       0.2389
OpenPorchSF   -0.0005     -0.0061       0.1520   0.0848       0.3088
EnclosedPorch  0.0029     -0.0120       0.0107  -0.0183      -0.1139
3SsnPorch     -0.0466     -0.0438       0.0700   0.0204       0.0304
```

ScreenPorch	0.0013	-0.0260	0.0414	0.0432	0.0649
PoolArea	0.0570	0.0083	0.2062	0.0777	0.0652
MiscVal	-0.0062	-0.0077	0.0034	0.0381	-0.0314
MoSold	0.0212	-0.0136	0.0112	0.0012	0.0708
YrSold	0.0007	-0.0214	0.0074	-0.0143	-0.0273
SalePrice	-0.0219	-0.0843	0.3518	0.2638	0.7910

	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1 \
Id	0.0126	-0.0127	-0.0220	-0.0503	-0.0050
MSSubClass	-0.0593	0.0279	0.0406	0.0229	-0.0698
LotFrontage	-0.0592	0.1233	0.0889	0.1935	0.2336
LotArea	-0.0056	0.0142	0.0138	0.1042	0.2141
OverallQual	-0.0919	0.5723	0.5507	0.4119	0.2397
OverallCond	1.0000	-0.3760	0.0737	-0.1281	-0.0462
YearBuilt	-0.3760	1.0000	0.5929	0.3157	0.2495
YearRemodAdd	0.0737	0.5929	1.0000	0.1796	0.1285
MasVnrArea	-0.1281	0.3157	0.1796	1.0000	0.2647
BsmtFinSF1	-0.0462	0.2495	0.1285	0.2647	1.0000
BsmtFinSF2	0.0402	-0.0491	-0.0678	-0.0723	-0.0501
BsmtUnfSF	-0.1368	0.1490	0.1811	0.1144	-0.4953
TotalBsmtSF	-0.1711	0.3915	0.2911	0.3639	0.5224
1stFlrSF	-0.1442	0.2820	0.2404	0.3445	0.4459
2ndFlrSF	0.0289	0.0103	0.1400	0.1746	-0.1371
LowQualFinSF	0.0255	-0.1838	-0.0624	-0.0691	-0.0645
GrLivArea	-0.0797	0.1990	0.2874	0.3909	0.2082
BsmtFullBath	-0.0549	0.1876	0.1195	0.0853	0.6492
BsmtHalfBath	0.1178	-0.0382	-0.0123	0.0267	0.0674
FullBath	-0.1941	0.4683	0.4390	0.2768	0.0585
HalfBath	-0.0608	0.2427	0.1833	0.2014	0.0043
BedroomAbvGr	0.0130	-0.0707	-0.0406	0.1028	-0.1074
KitchenAbvGr	-0.0870	-0.1748	-0.1496	-0.0376	-0.0810
TotRmsAbvGrd	-0.0576	0.0956	0.1917	0.2807	0.0443
Fireplaces	-0.0238	0.1477	0.1126	0.2491	0.2600
GarageYrBlt	-0.3243	0.8257	0.6423	0.2527	0.1535
GarageCars	-0.1858	0.5379	0.4206	0.3642	0.2241
GarageArea	-0.1515	0.4790	0.3716	0.3731	0.2970
WoodDeckSF	-0.0033	0.2249	0.2057	0.1597	0.2043
OpenPorchSF	-0.0326	0.1887	0.2263	0.1257	0.1118
EnclosedPorch	0.0704	-0.3873	-0.1939	-0.1102	-0.1023
3SsnPorch	0.0255	0.0314	0.0453	0.0188	0.0265
ScreenPorch	0.0548	-0.0504	-0.0387	0.0615	0.0620
PoolArea	-0.0020	0.0049	0.0058	0.0117	0.1405
MiscVal	0.0688	-0.0344	-0.0103	-0.0298	0.0036
MoSold	-0.0035	0.0124	0.0215	-0.0060	-0.0157
YrSold	0.0439	-0.0136	0.0357	-0.0082	0.0144
SalePrice	-0.0779	0.5229	0.5071	0.4775	0.3864

	…	WoodDeckSF	OpenPorchSF	EnclosedPorch	3SsnPorch	\
Id	…	-0.0296	-0.0005	0.0029	-0.0466	
MSSubClass	…	-0.0126	-0.0061	-0.0120	-0.0438	
LotFrontage	…	0.0885	0.1520	0.0107	0.0700	
LotArea	…	0.1717	0.0848	-0.0183	0.0204	
OverallQual	…	0.2389	0.3088	-0.1139	0.0304	
OverallCond	…	-0.0033	-0.0326	0.0704	0.0255	
YearBuilt	…	0.2249	0.1887	-0.3873	0.0314	
YearRemodAdd	…	0.2057	0.2263	-0.1939	0.0453	
MasVnrArea	…	0.1597	0.1257	-0.1102	0.0188	
BsmtFinSF1	…	0.2043	0.1118	-0.1023	0.0265	
BsmtFinSF2	…	0.0679	0.0031	0.0365	-0.0300	
BsmtUnfSF	…	-0.0053	0.1290	-0.0025	0.0208	
TotalBsmtSF	…	0.2320	0.2473	-0.0955	0.0374	
1stFlrSF	…	0.2355	0.2117	-0.0653	0.0561	
2ndFlrSF	…	0.0922	0.2080	0.0620	-0.0244	
LowQualFinSF	…	-0.0254	0.0183	0.0611	-0.0043	
GrLivArea	…	0.2474	0.3302	0.0091	0.0206	
BsmtFullBath	…	0.1753	0.0673	-0.0499	-0.0001	
BsmtHalfBath	…	0.0402	-0.0253	-0.0086	0.0351	
FullBath	…	0.1877	0.2600	-0.1151	0.0354	
HalfBath	…	0.1081	0.1997	-0.0953	-0.0050	
BedroomAbvGr	…	0.0469	0.0938	0.0416	-0.0245	
KitchenAbvGr	…	-0.0901	-0.0701	0.0373	-0.0246	
TotRmsAbvGrd	…	0.1660	0.2342	0.0042	-0.0067	
Fireplaces	…	0.2000	0.1694	-0.0248	0.0113	
GarageYrBlt	…	0.2246	0.2284	-0.2970	0.0235	
GarageCars	…	0.2263	0.2136	-0.1514	0.0358	
GarageArea	…	0.2247	0.2414	-0.1218	0.0351	
WoodDeckSF	…	1.0000	0.0587	-0.1260	-0.0328	
OpenPorchSF	…	0.0587	1.0000	-0.0931	-0.0058	
EnclosedPorch	…	-0.1260	-0.0931	1.0000	-0.0373	
3SsnPorch	…	-0.0328	-0.0058	-0.0373	1.0000	
ScreenPorch	…	-0.0742	0.0743	-0.0829	-0.0314	
PoolArea	…	0.0734	0.0608	0.0542	-0.0080	
MiscVal	…	-0.0096	-0.0186	0.0184	0.0004	
MoSold	…	0.0210	0.0713	-0.0289	0.0295	
YrSold	…	0.0223	-0.0576	-0.0099	0.0186	
SalePrice	…	0.3244	0.3159	-0.1286	0.0446	

	ScreenPorch	PoolArea	MiscVal	MoSold	YrSold	SalePrice
Id	0.0013	0.0570	-0.0062	0.0212	0.0007	-0.0219
MSSubClass	-0.0260	0.0083	-0.0077	-0.0136	-0.0214	-0.0843
LotFrontage	0.0414	0.2062	0.0034	0.0112	0.0074	0.3518
LotArea	0.0432	0.0777	0.0381	0.0012	-0.0143	0.2638
OverallQual	0.0649	0.0652	-0.0314	0.0708	-0.0273	0.7910
OverallCond	0.0548	-0.0020	0.0688	-0.0035	0.0439	-0.0779

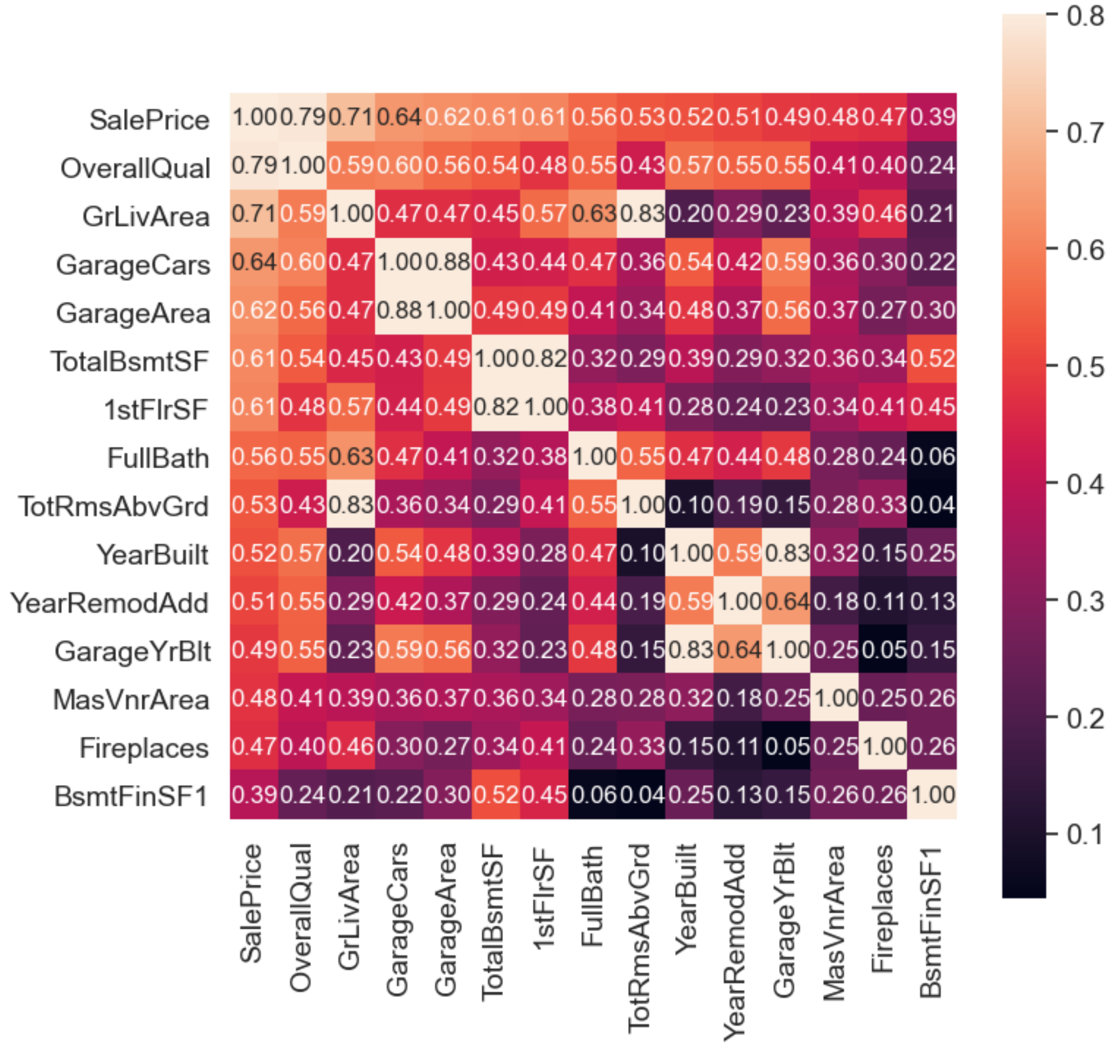
```
YearBuilt        -0.0504   0.0049  -0.0344  0.0124 -0.0136     0.5229
YearRemodAdd     -0.0387   0.0058  -0.0103  0.0215  0.0357     0.5071
MasVnrArea        0.0615   0.0117  -0.0298 -0.0060 -0.0082     0.4775
BsmtFinSF1        0.0620   0.1405   0.0036 -0.0157  0.0144     0.3864
BsmtFinSF2        0.0889   0.0417   0.0049 -0.0152  0.0317    -0.0114
BsmtUnfSF        -0.0126  -0.0351  -0.0238  0.0349 -0.0413     0.2145
TotalBsmtSF       0.0845   0.1261  -0.0185  0.0132 -0.0150     0.6136
1stFlrSF          0.0888   0.1315  -0.0211  0.0314 -0.0136     0.6059
2ndFlrSF          0.0406   0.0815   0.0162  0.0352 -0.0287     0.3193
LowQualFinSF      0.0268   0.0622  -0.0038 -0.0222 -0.0289    -0.0256
GrLivArea         0.1015   0.1702  -0.0024  0.0502 -0.0365     0.7086
BsmtFullBath      0.0231   0.0676  -0.0230 -0.0254  0.0670     0.2271
BsmtHalfBath      0.0321   0.0200  -0.0074  0.0329 -0.0465    -0.0168
FullBath         -0.0081   0.0496  -0.0143  0.0559 -0.0197     0.5607
HalfBath          0.0724   0.0224   0.0013 -0.0090 -0.0103     0.2841
BedroomAbvGr      0.0443   0.0707   0.0078  0.0465 -0.0360     0.1682
KitchenAbvGr     -0.0516  -0.0145   0.0623  0.0266  0.0317    -0.1359
TotRmsAbvGrd      0.0594   0.0838   0.0248  0.0369 -0.0345     0.5337
Fireplaces        0.1845   0.0951   0.0014  0.0464 -0.0241     0.4669
GarageYrBlt      -0.0754  -0.0145  -0.0324  0.0053 -0.0010     0.4864
GarageCars        0.0505   0.0209  -0.0431  0.0405 -0.0391     0.6404
GarageArea        0.0514   0.0610  -0.0274  0.0280 -0.0274     0.6234
WoodDeckSF       -0.0742   0.0734  -0.0096  0.0210  0.0223     0.3244
OpenPorchSF       0.0743   0.0608  -0.0186  0.0713 -0.0576     0.3159
EnclosedPorch    -0.0829   0.0542   0.0184 -0.0289 -0.0099    -0.1286
3SsnPorch        -0.0314  -0.0080   0.0004  0.0295  0.0186     0.0446
ScreenPorch       1.0000   0.0513   0.0319  0.0232  0.0107     0.1114
PoolArea          0.0513   1.0000   0.0297 -0.0337 -0.0597     0.0924
MiscVal           0.0319   0.0297   1.0000 -0.0065  0.0049    -0.0212
MoSold            0.0232  -0.0337  -0.0065  1.0000 -0.1457     0.0464
YrSold            0.0107  -0.0597   0.0049 -0.1457  1.0000    -0.0289
SalePrice         0.1114   0.0924  -0.0212  0.0464 -0.0289     1.0000

[38 rows x 38 columns]
```

[246]:
```python
n = 15
top15_cols = corrmat.nlargest(n,'SalePrice')['SalePrice'].index

top15_cols
```

[246]:
```
Index(['SalePrice', 'OverallQual', 'GrLivArea', 'GarageCars', 'GarageArea',
       'TotalBsmtSF', '1stFlrSF', 'FullBath', 'TotRmsAbvGrd', 'YearBuilt',
       'YearRemodAdd', 'GarageYrBlt', 'MasVnrArea', 'Fireplaces',
       'BsmtFinSF1'],
      dtype='object')
```

[247]:
```
corrmat_top15 = house_train_n[top15_cols].corr()
fig1,ax1 = plt.subplots(figsize=(8,8))
sns.heatmap(corrmat_top15,vmax=.8, square=True,ax=ax1,annot=True, fmt='.2f',␣
 ↪annot_kws={'size': 12})
```

[247]: <AxesSubplot:>



[248]:
```
def list_subtract(x, y):
    return [item for item in x if item not in y]
```

[249]:
```
cols = list_subtract(house_train.columns.tolist(), ignore_cols)
```

[250]:
```
# lets take a look to the shape of columns
pd.set_option("display.float", "{:.4f}".format)
```

```
skew_df = full_df[cols].skew().to_frame().rename(columns={0:'Skewness'}).
↪sort_values('Skewness')
skew_df
```

[250]:

	Skewness
YearBuilt	-0.6001
YearRemodAdd	-0.4513
GarageYrBlt	-0.3822
GarageCars	-0.2184
YrSold	0.1325
FullBath	0.1677
MoSold	0.1960
OverallQual	0.1972
GarageArea	0.2413
BedroomAbvGr	0.3265
OverallCond	0.5706
BsmtFullBath	0.6241
HalfBath	0.6949
Fireplaces	0.7339
TotRmsAbvGrd	0.7588
2ndFlrSF	0.8621
BsmtUnfSF	0.9198
TotalBsmtSF	1.1629
GrLivArea	1.2700
MSSubClass	1.3762
BsmtFinSF1	1.4257
1stFlrSF	1.4704
WoodDeckSF	1.8434
OpenPorchSF	2.5364
MasVnrArea	2.6026
BsmtHalfBath	3.9320
ScreenPorch	3.9487
EnclosedPorch	4.0060
BsmtFinSF2	4.1475
KitchenAbvGr	4.3045
3SsnPorch	11.3819
LowQualFinSF	12.0950
LotArea	12.8290
PoolArea	16.9070
MiscVal	21.9585

[251]:
```
def find_skew_rows(df):
    return df[(df['Skewness'] < -1) | (df['Skewness'] > 1)]
```

[252]:
```
skew_cols = find_skew_rows(skew_df).index.tolist()
skew_cols
```

[252]:
```
['TotalBsmtSF',
 'GrLivArea',
 'MSSubClass',
 'BsmtFinSF1',
 '1stFlrSF',
 'WoodDeckSF',
 'OpenPorchSF',
 'MasVnrArea',
 'BsmtHalfBath',
 'ScreenPorch',
 'EnclosedPorch',
 'BsmtFinSF2',
 'KitchenAbvGr',
 '3SsnPorch',
 'LowQualFinSF',
 'LotArea',
 'PoolArea',
 'MiscVal']
```

[253]:
```
# use all data in visualization
full_df
```

[253]:
```
        Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
0        1          60       RL      65.0000     8450   Pave   NaN      Reg
1        2          20       RL      80.0000     9600   Pave   NaN      Reg
2        3          60       RL      68.0000    11250   Pave   NaN      IR1
3        4          70       RL      60.0000     9550   Pave   NaN      IR1
4        5          60       RL      84.0000    14260   Pave   NaN      IR1
…     …          …     …           …       …    …   …      …
2914  2915         160       RM      21.0000     1936   Pave   NaN      Reg
2915  2916         160       RM      21.0000     1894   Pave   NaN      Reg
2916  2917          20       RL     160.0000    20000   Pave   NaN      Reg
2917  2918          85       RL      62.0000    10441   Pave   NaN      Reg
2918  2919          60       RL      74.0000     9627   Pave   NaN      Reg

     LandContour Utilities  … ScreenPorch PoolArea PoolQC  Fence  \
0            Lvl    AllPub  …           0        0    NaN    NaN
1            Lvl    AllPub  …           0        0    NaN    NaN
2            Lvl    AllPub  …           0        0    NaN    NaN
3            Lvl    AllPub  …           0        0    NaN    NaN
4            Lvl    AllPub  …           0        0    NaN    NaN
…           …       …  …         …      …    …
2914         Lvl    AllPub  …           0        0    NaN    NaN
2915         Lvl    AllPub  …           0        0    NaN    NaN
2916         Lvl    AllPub  …           0        0    NaN    NaN
2917         Lvl    AllPub  …           0        0    NaN  MnPrv
2918         Lvl    AllPub  …           0        0    NaN    NaN
```

```
     MiscFeature MiscVal MoSold  YrSold  SaleType  SaleCondition
0            NaN       0      2    2008        WD         Normal
1            NaN       0      5    2007        WD         Normal
2            NaN       0      9    2008        WD         Normal
3            NaN       0      2    2006        WD        Abnorml
4            NaN       0     12    2008        WD         Normal
…          …     …    …      …     …         …
2914         NaN       0      6    2006        WD         Normal
2915         NaN       0      4    2006        WD        Abnorml
2916         NaN       0      9    2006        WD        Abnorml
2917        Shed     700      7    2006        WD         Normal
2918         NaN       0     11    2006        WD         Normal

[2919 rows x 80 columns]
```

[254]:
```
house_train_n.columns.sort_values(ascending = True)
```

[254]:
```
Index(['1stFlrSF', '2ndFlrSF', '3SsnPorch', 'BedroomAbvGr', 'BsmtFinSF1',
       'BsmtFinSF2', 'BsmtFullBath', 'BsmtHalfBath', 'BsmtUnfSF',
       'EnclosedPorch', 'Fireplaces', 'FullBath', 'GarageArea', 'GarageCars',
       'GarageYrBlt', 'GrLivArea', 'HalfBath', 'Id', 'KitchenAbvGr', 'LotArea',
       'LotFrontage', 'LowQualFinSF', 'MSSubClass', 'MasVnrArea', 'MiscVal',
       'MoSold', 'OpenPorchSF', 'OverallCond', 'OverallQual', 'PoolArea',
       'SalePrice', 'ScreenPorch', 'TotRmsAbvGrd', 'TotalBsmtSF', 'WoodDeckSF',
       'YearBuilt', 'YearRemodAdd', 'YrSold'],
      dtype='object')
```

[255]:
```
house_train_c.columns.sort_values(ascending = True)
```

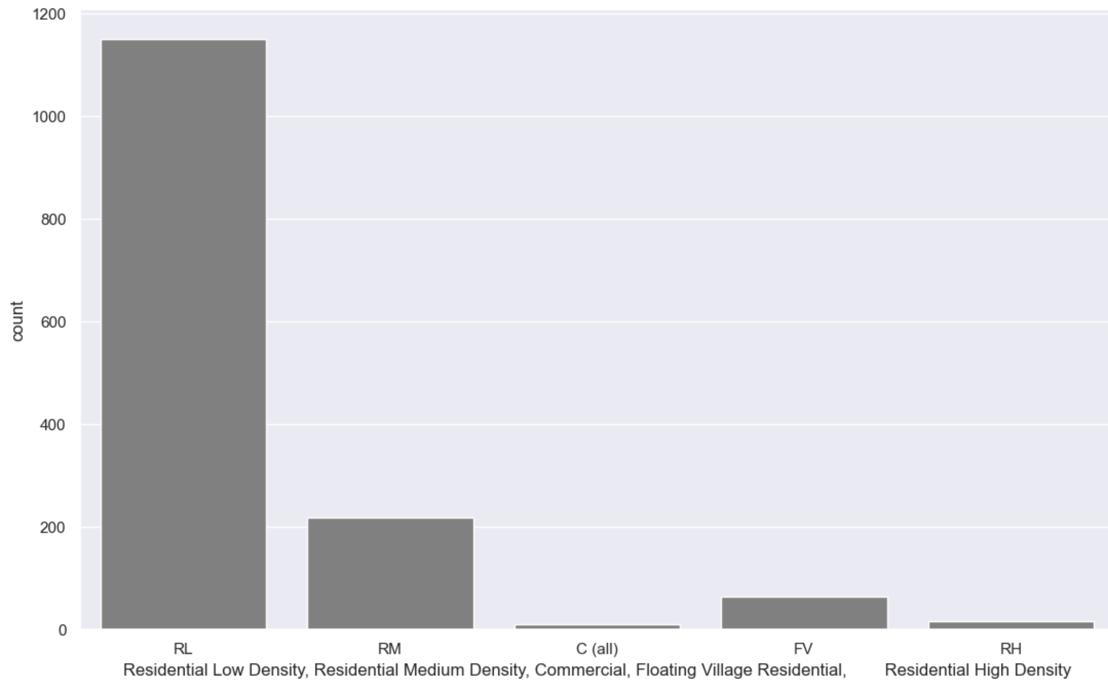
[255]:
```
Index(['Alley', 'BldgType', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1',
       'BsmtFinType2', 'BsmtQual', 'CentralAir', 'Condition1', 'Condition2',
       'Electrical', 'ExterCond', 'ExterQual', 'Exterior1st', 'Exterior2nd',
       'Fence', 'FireplaceQu', 'Foundation', 'Functional', 'GarageCond',
       'GarageFinish', 'GarageQual', 'GarageType', 'Heating', 'HeatingQC',
       'HouseStyle', 'KitchenQual', 'LandContour', 'LandSlope', 'LotConfig',
       'LotShape', 'MSZoning', 'MasVnrType', 'MiscFeature', 'Neighborhood',
       'PavedDrive', 'PoolQC', 'RoofMatl', 'RoofStyle', 'SaleCondition',
       'SaleType', 'Street', 'Utilities'],
      dtype='object')
```

3 Data Visualizations - Bar Plots

3.1 Column: MSZoning

[256]:
```
sns.set(rc={'figure.figsize':(13, 8)})
ax = sns.countplot(house_train['MSZoning'], color = 'gray')

ax.set(xlabel = "Residential Low Density, Residential Medium Density,␣
↪Commercial, Floating Village Residential, \
       Residential High Density")
```

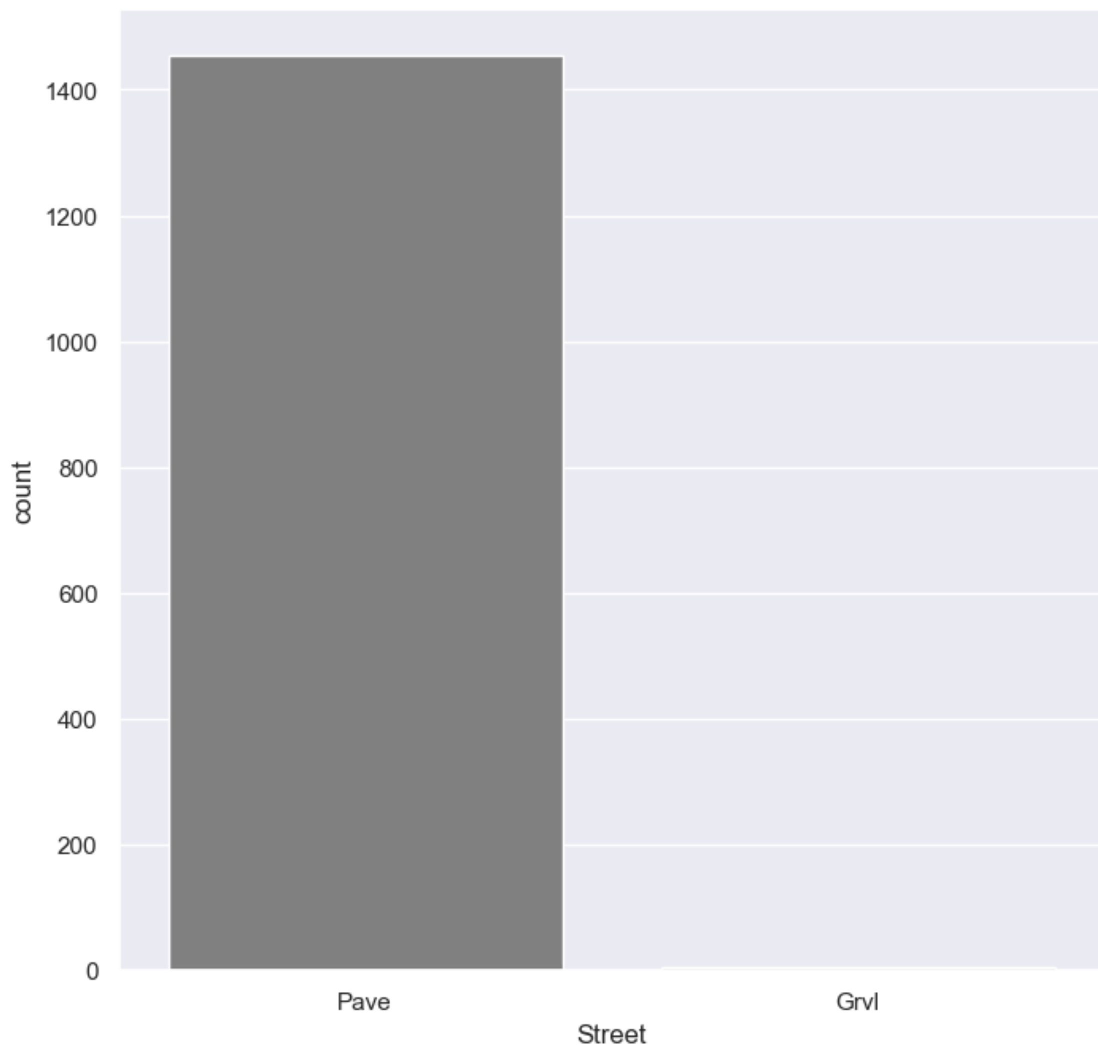
[256]:
```
[Text(0.5, 0, 'Residential Low Density, Residential Medium Density, Commercial,
Floating Village Residential,        Residential High Density')]
```



3.2 Column: Street

[257]:
```
sns.set(rc={'figure.figsize':(8, 8)})
sns.countplot(house_train['Street'], color = 'gray')
```

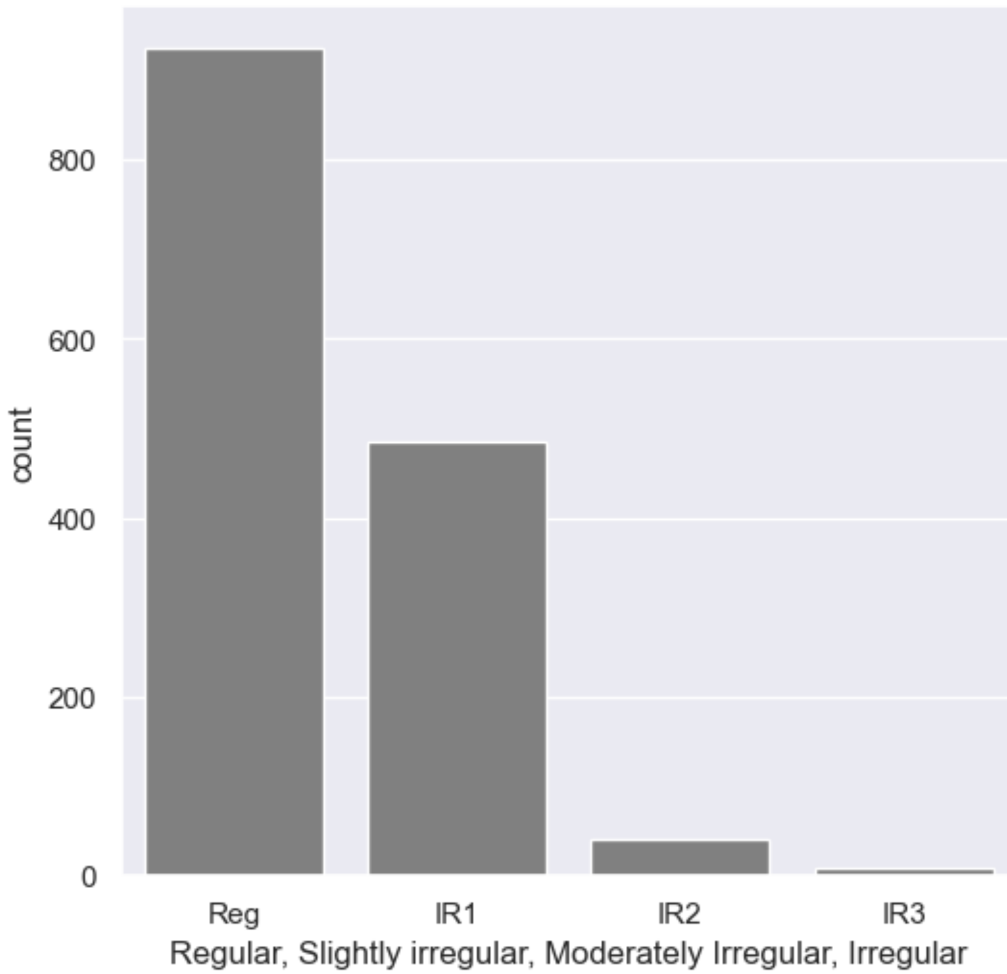
[257]:
```
<AxesSubplot:xlabel='Street', ylabel='count'>
```



3.3 Column: LotShape

[258]:
```
sns.set(rc={'figure.figsize':(6, 6)})
ax = sns.countplot(house_train['LotShape'], color = 'gray')
ax.set(xlabel = "Regular, Slightly irregular, Moderately Irregular, Irregular")
```

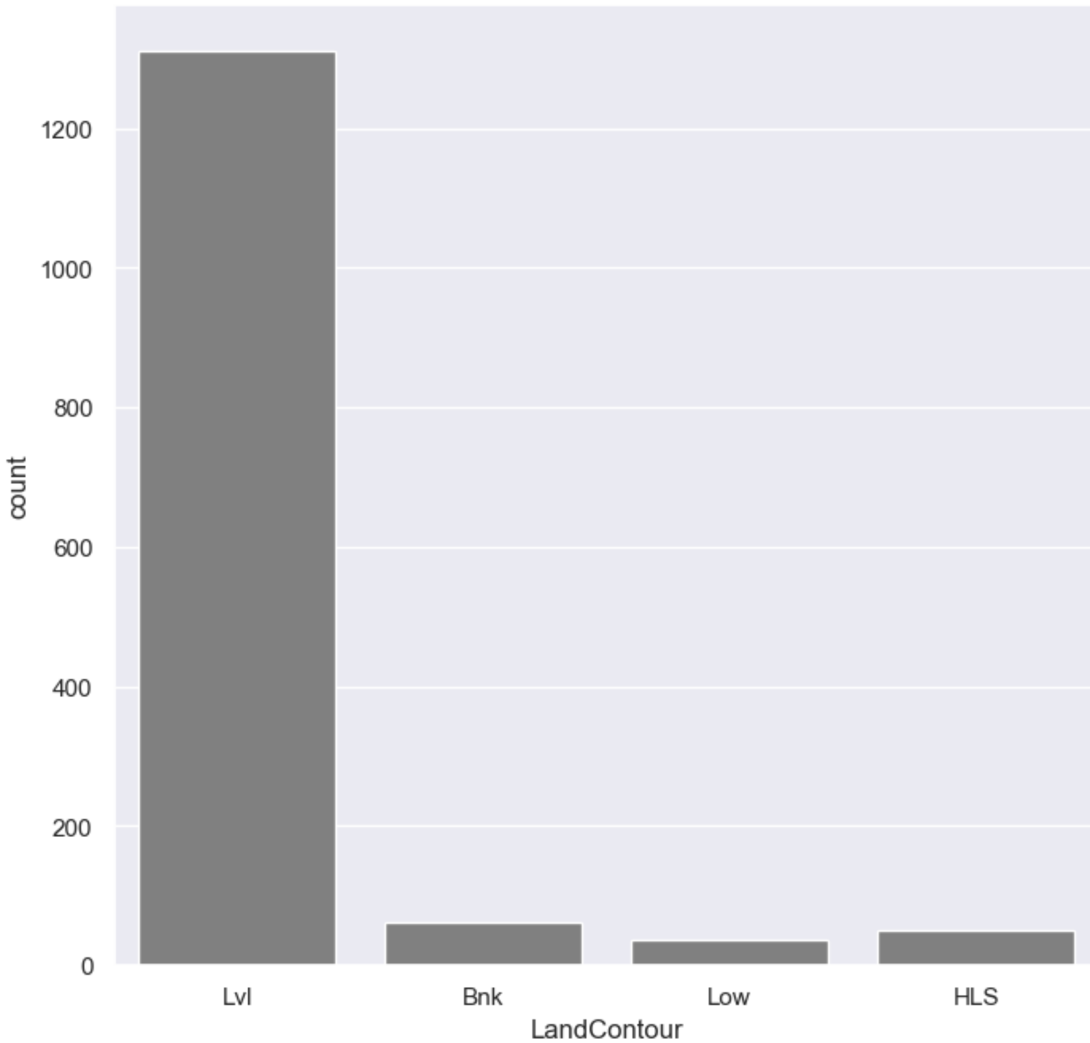
[258]:
```
[Text(0.5, 0, 'Regular, Slightly irregular, Moderately Irregular, Irregular')]
```



3.4 Column: LandContour

[259]:
```
sns.set(rc={'figure.figsize':(8, 8)})
sns.countplot(house_train['LandContour'], color = 'gray')
```

[259]: <AxesSubplot:xlabel='LandContour', ylabel='count'>



3.5 Column: LotConfig

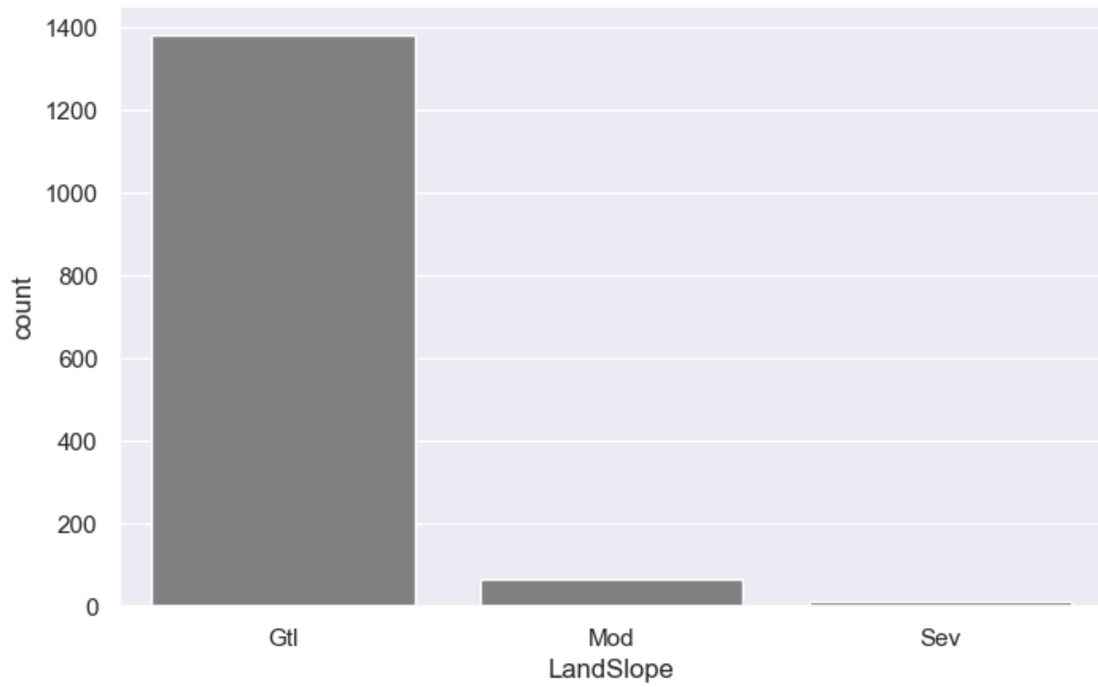
[260]:
```
sns.set(rc={'figure.figsize':(8, 5)})
sns.countplot(house_train['LotConfig'], color = 'gray')
```

[260]: <AxesSubplot:xlabel='LotConfig', ylabel='count'>

3.6 Column: LandSlope

[261]:
```
sns.countplot(house_train['LandSlope'], color = 'gray')
```

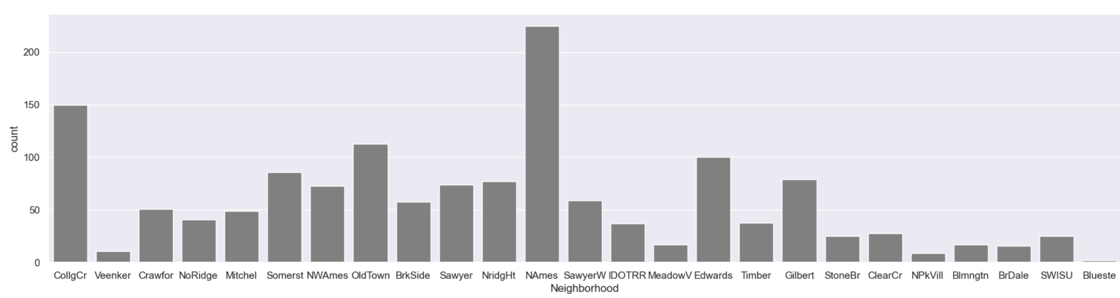
[261]: <AxesSubplot:xlabel='LandSlope', ylabel='count'>



3.7 Column: Neighborhood

[262]:
```
sns.set(rc={'figure.figsize':(21, 5)})
sns.countplot(house_train['Neighborhood'], color = 'gray')
```

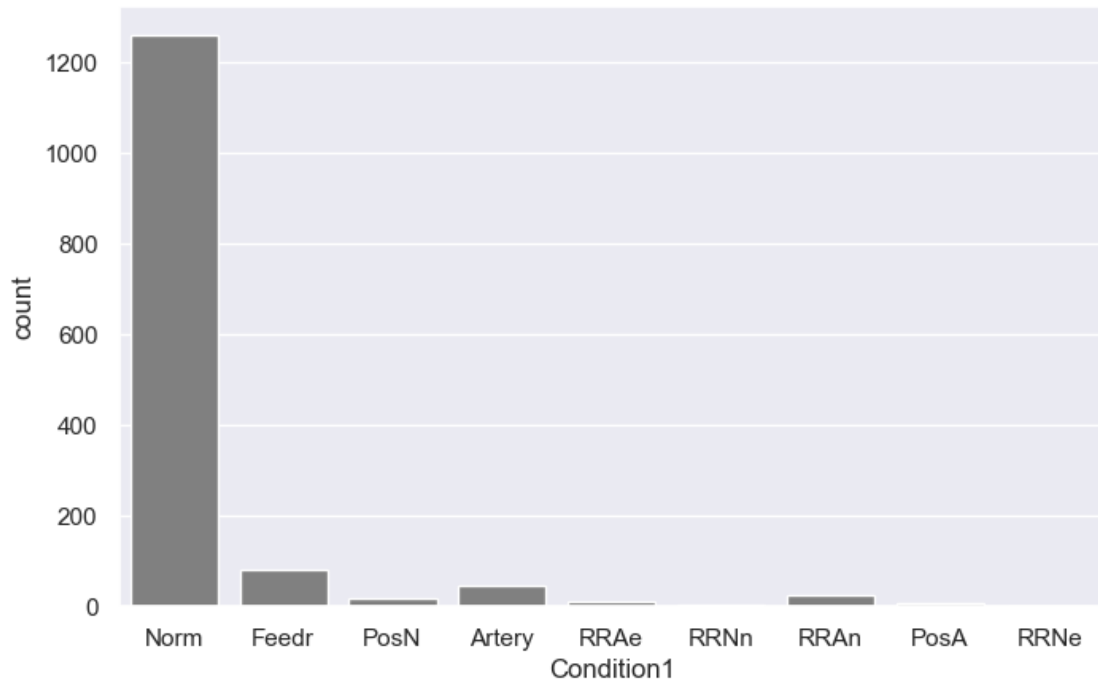
[262]: <AxesSubplot:xlabel='Neighborhood', ylabel='count'>



3.8 Column: Condition1

[263]:
```
sns.set(rc={'figure.figsize':(8, 5)})
sns.countplot(house_train['Condition1'], color = 'gray')
```

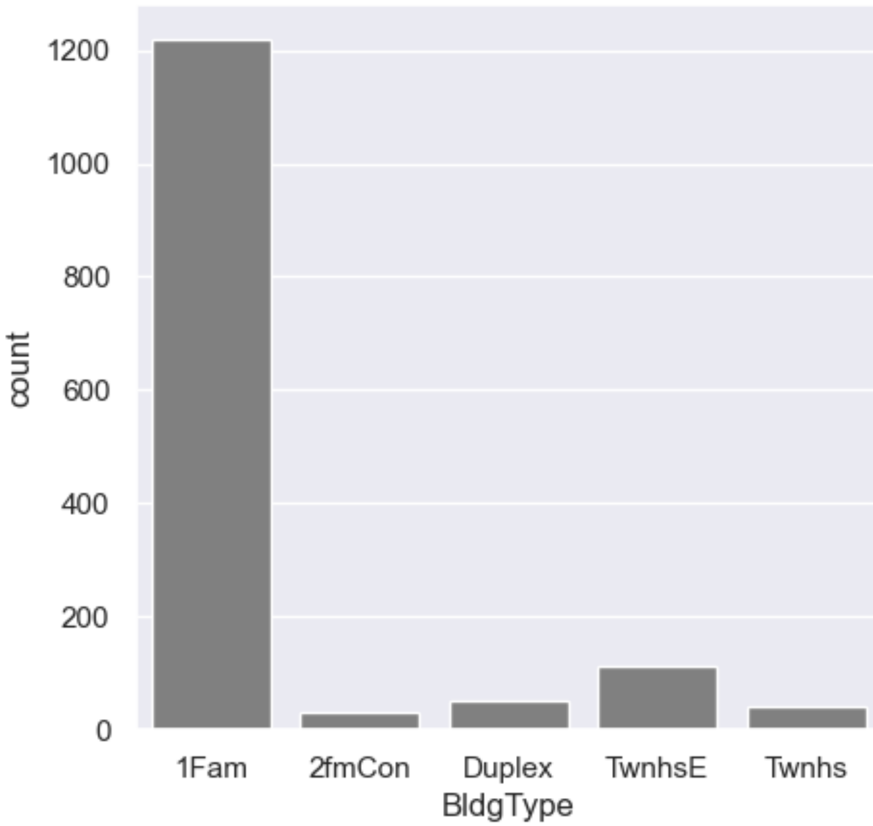
[263]: <AxesSubplot:xlabel='Condition1', ylabel='count'>



3.9 Column: BldgType

[264]:
```
sns.set(rc={'figure.figsize':(5, 5)})
sns.countplot(house_train['BldgType'], color = 'gray')
```

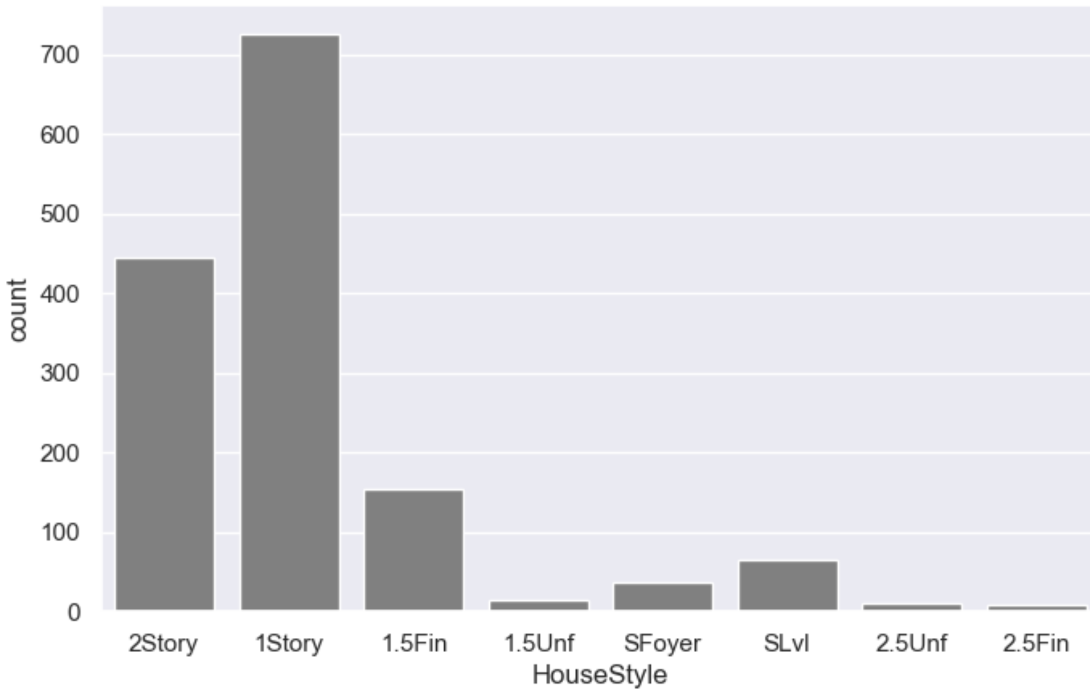
[264]: <AxesSubplot:xlabel='BldgType', ylabel='count'>



3.10 Column: HouseStyle

[265]:
```
sns.set(rc={'figure.figsize':(8, 5)})
sns.countplot(house_train['HouseStyle'], color = 'gray')
```

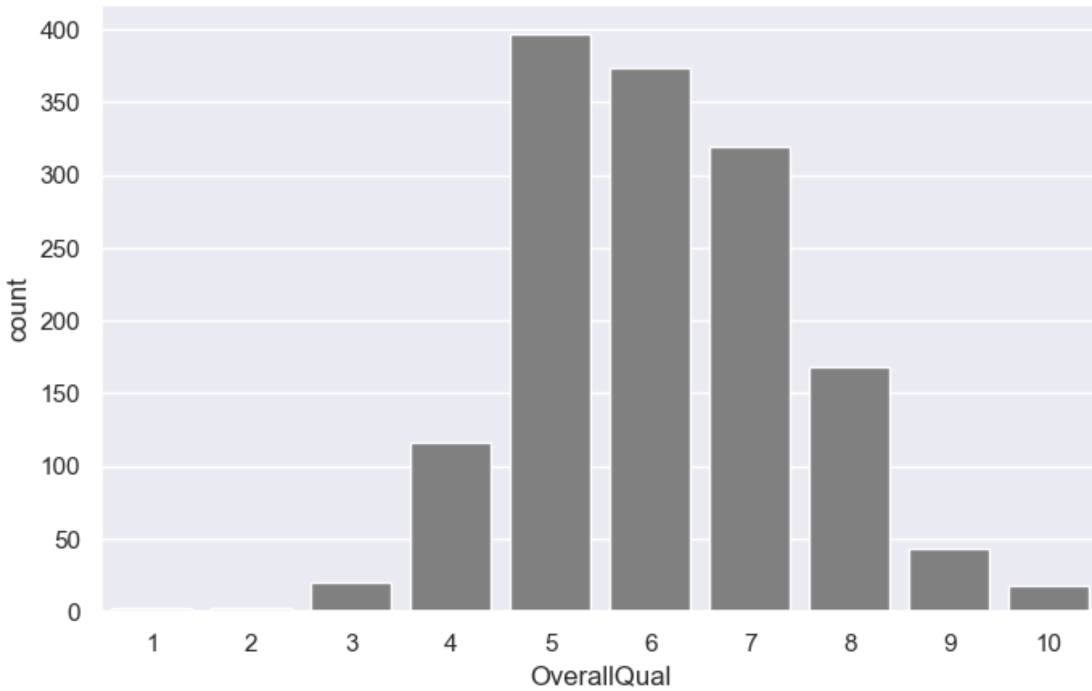
[265]: <AxesSubplot:xlabel='HouseStyle', ylabel='count'>



3.11 Column: OverallQual

[266]:
```
sns.countplot(house_train['OverallQual'], color = 'gray')
```

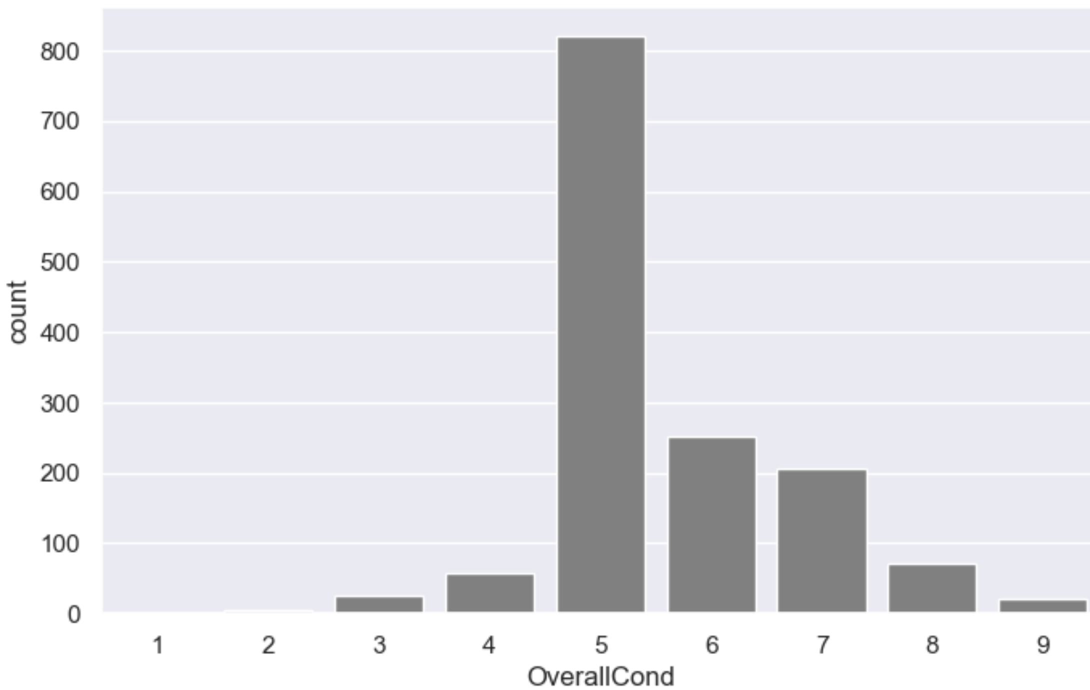
[266]: <AxesSubplot:xlabel='OverallQual', ylabel='count'>



3.12 Column: OverallCond

[267]:
```
sns.countplot(house_train['OverallCond'], color = 'gray')
```

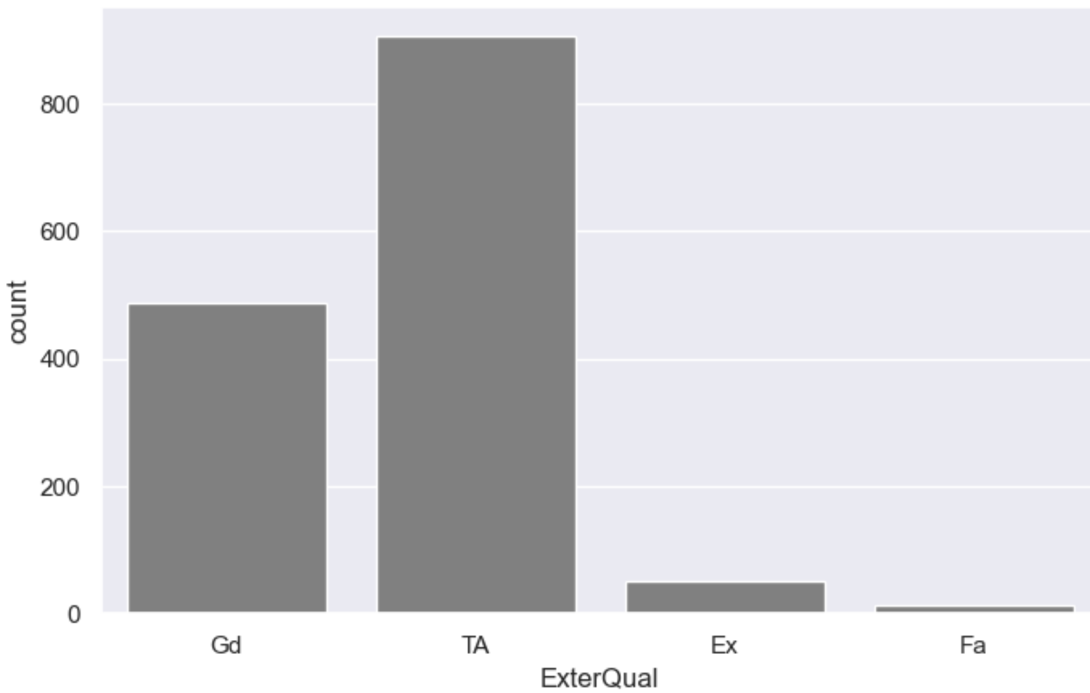
[267]: <AxesSubplot:xlabel='OverallCond', ylabel='count'>



3.13 Column: ExterQual

[268]:
```
sns.countplot(house_train['ExterQual'], color = 'gray')
```

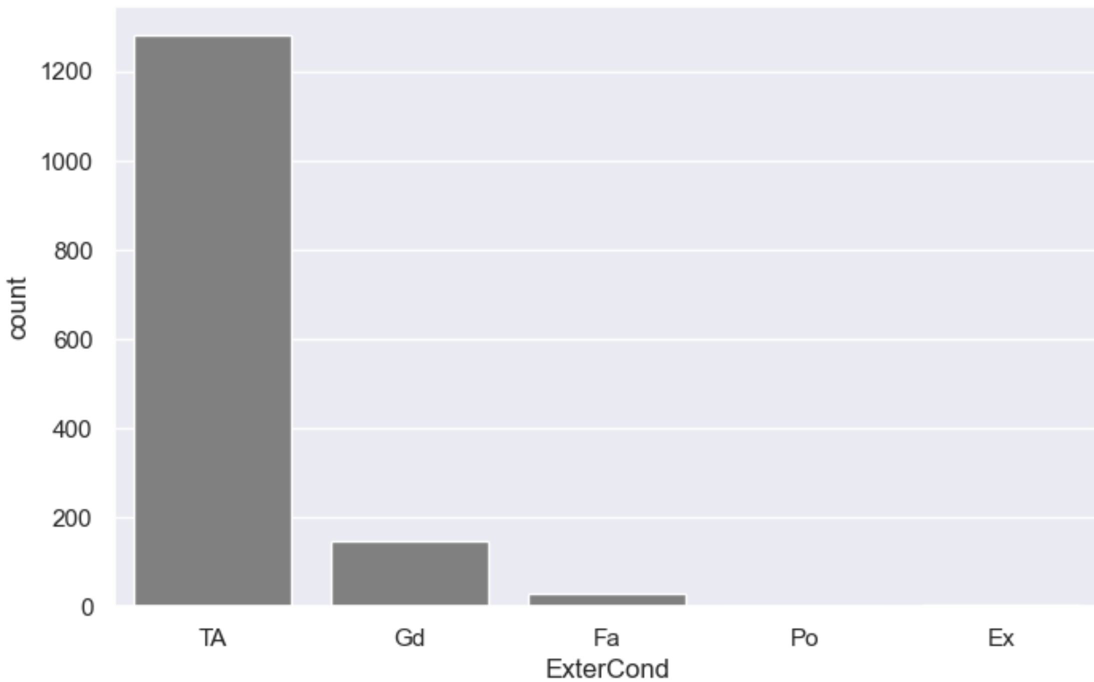
[268]: <AxesSubplot:xlabel='ExterQual', ylabel='count'>



3.14 Column: ExterCond

[269]:
```
sns.countplot(house_train['ExterCond'], color = 'gray')
```

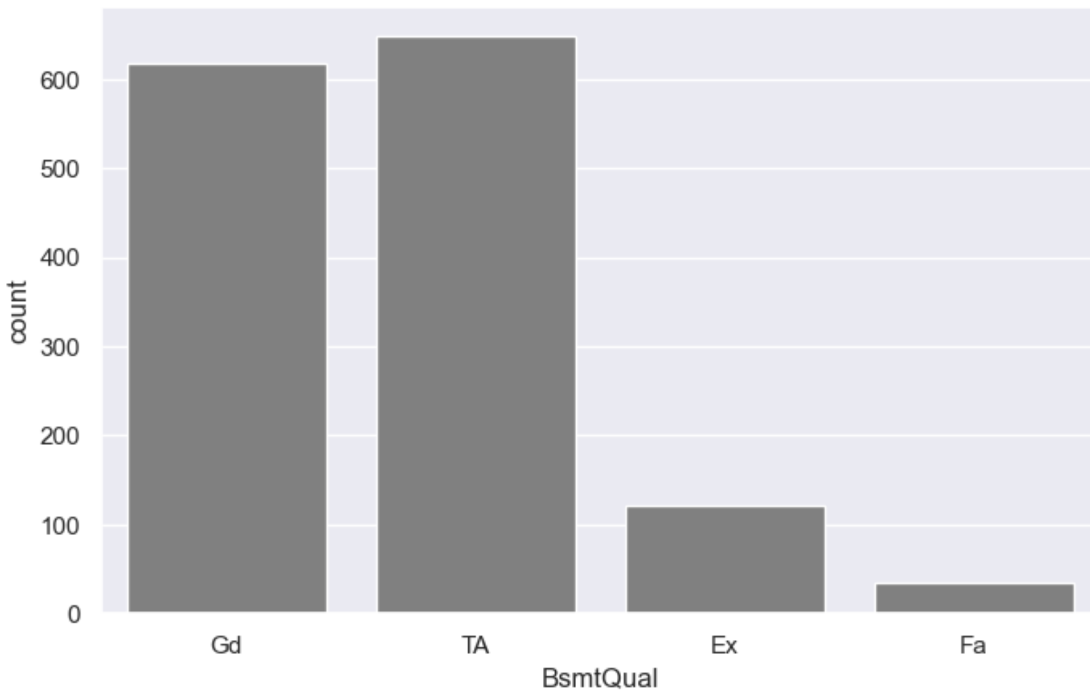
[269]: <AxesSubplot:xlabel='ExterCond', ylabel='count'>



3.15 Column: BsmtQual

[270]:
```
sns.countplot(house_train['BsmtQual'], color = 'gray')
```

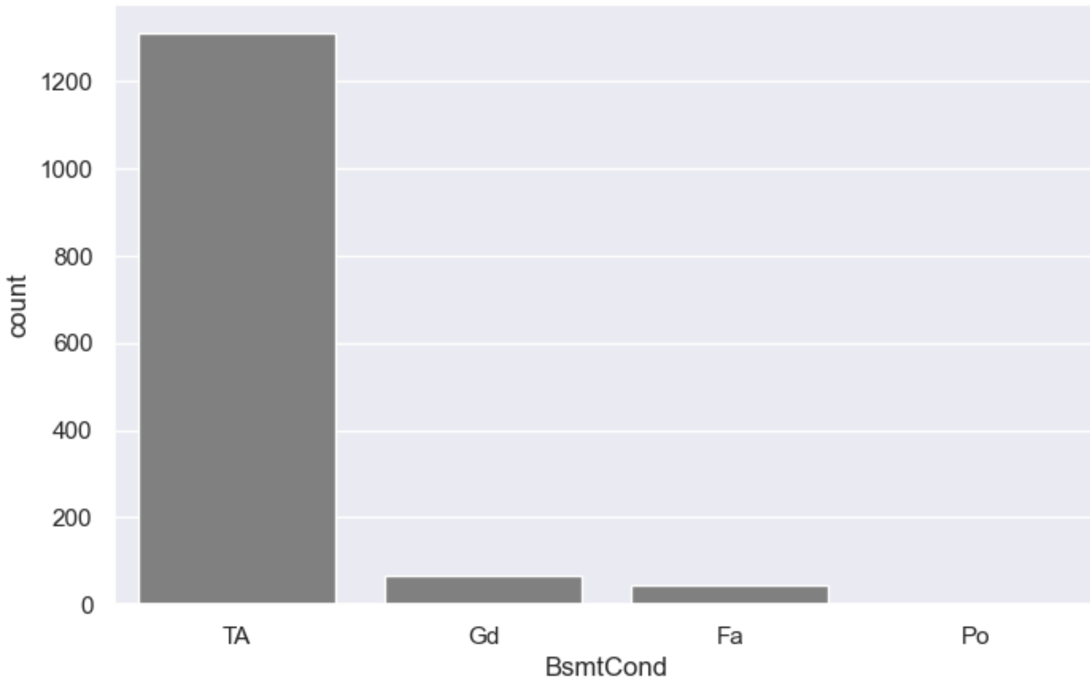
[270]: <AxesSubplot:xlabel='BsmtQual', ylabel='count'>



3.16 Column: BsmtCond

[271]:
```
sns.countplot(house_train['BsmtCond'], color = 'gray')
```

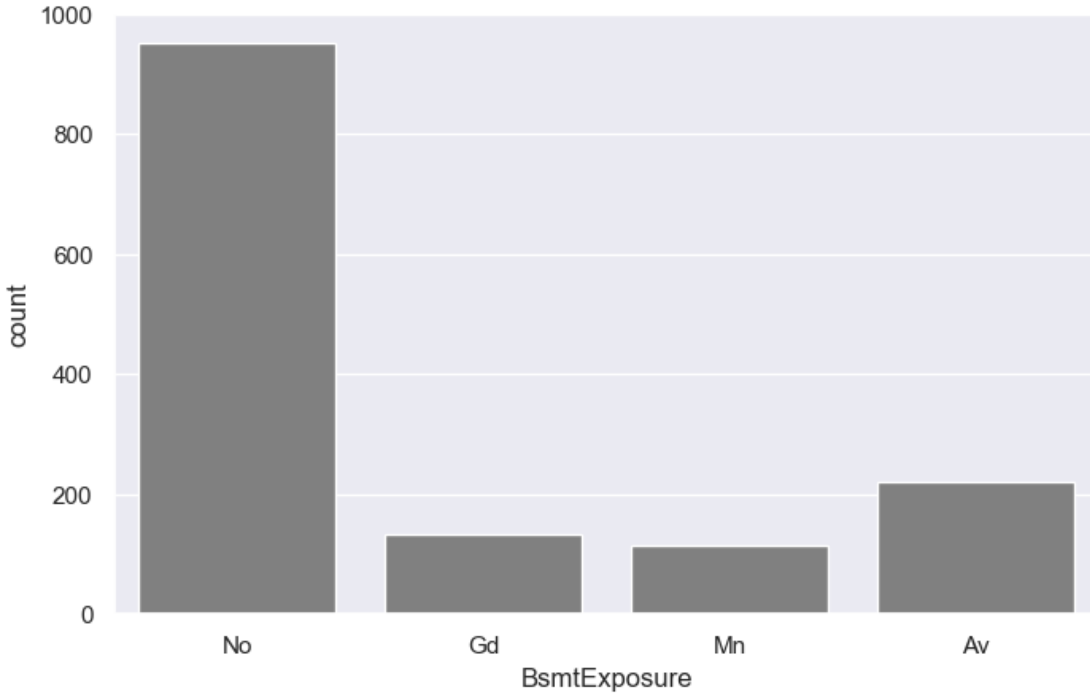
[271]: <AxesSubplot:xlabel='BsmtCond', ylabel='count'>



3.17 Column: BsmtExposure

[272]:
```
sns.countplot(house_train['BsmtExposure'], color = 'gray')
```

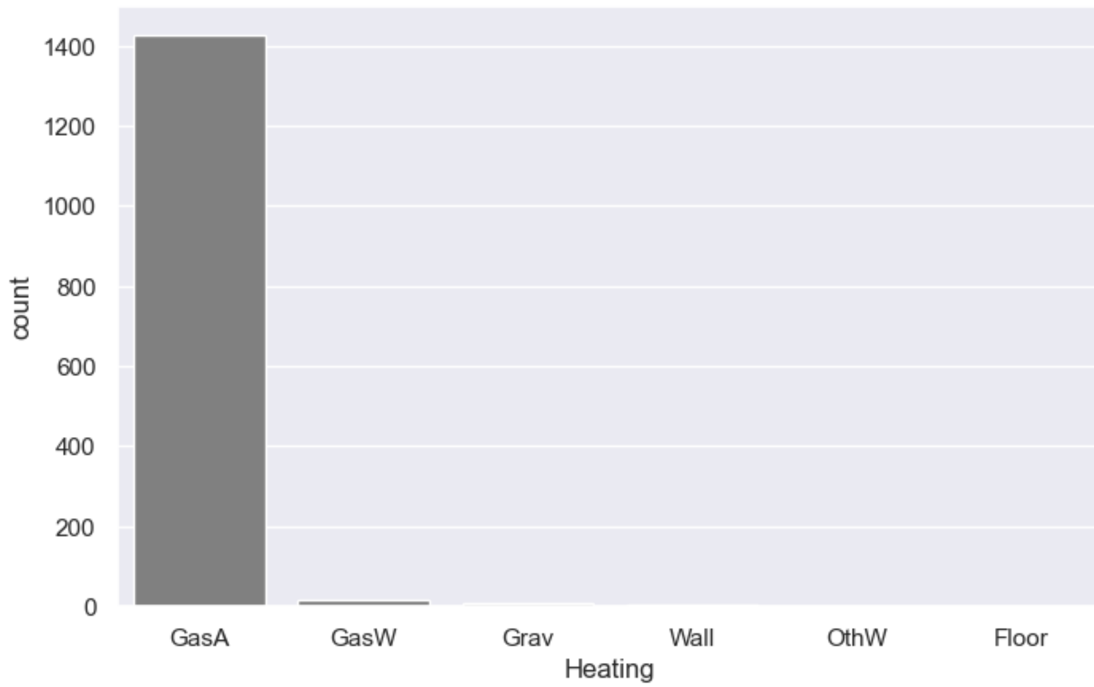
[272]: <AxesSubplot:xlabel='BsmtExposure', ylabel='count'>



3.18 Column: Heating

[273]:
```
sns.countplot(house_train['Heating'], color = 'gray')
```

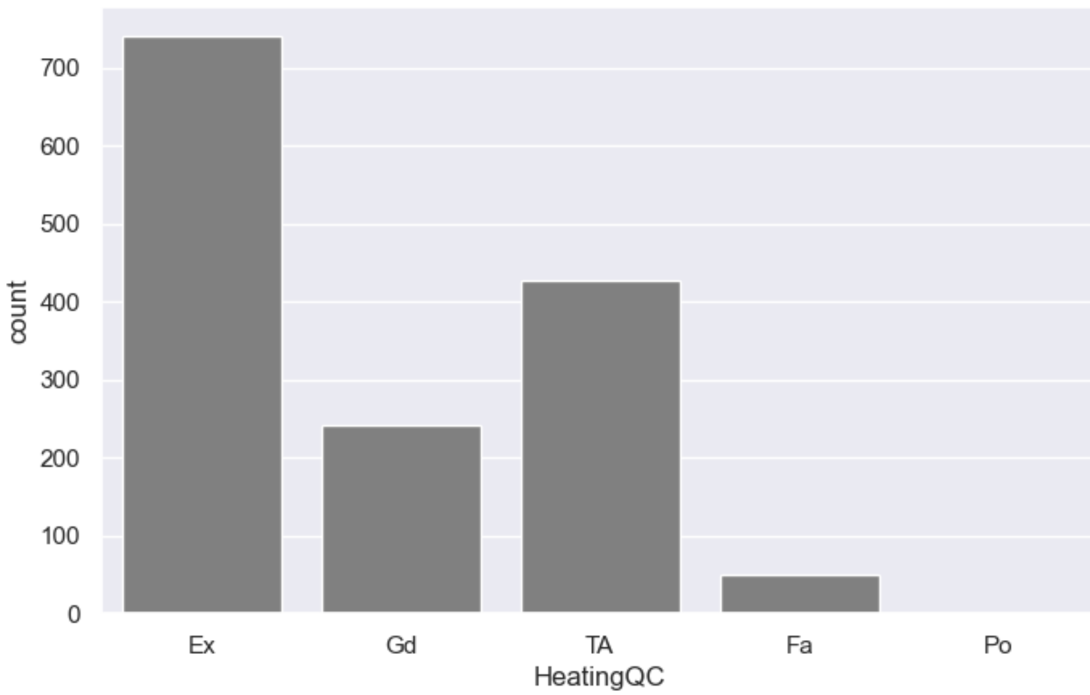
[273]: <AxesSubplot:xlabel='Heating', ylabel='count'>



3.19 Column: HeatingQC

[274]:
```
sns.countplot(house_train['HeatingQC'], color = 'gray')
```

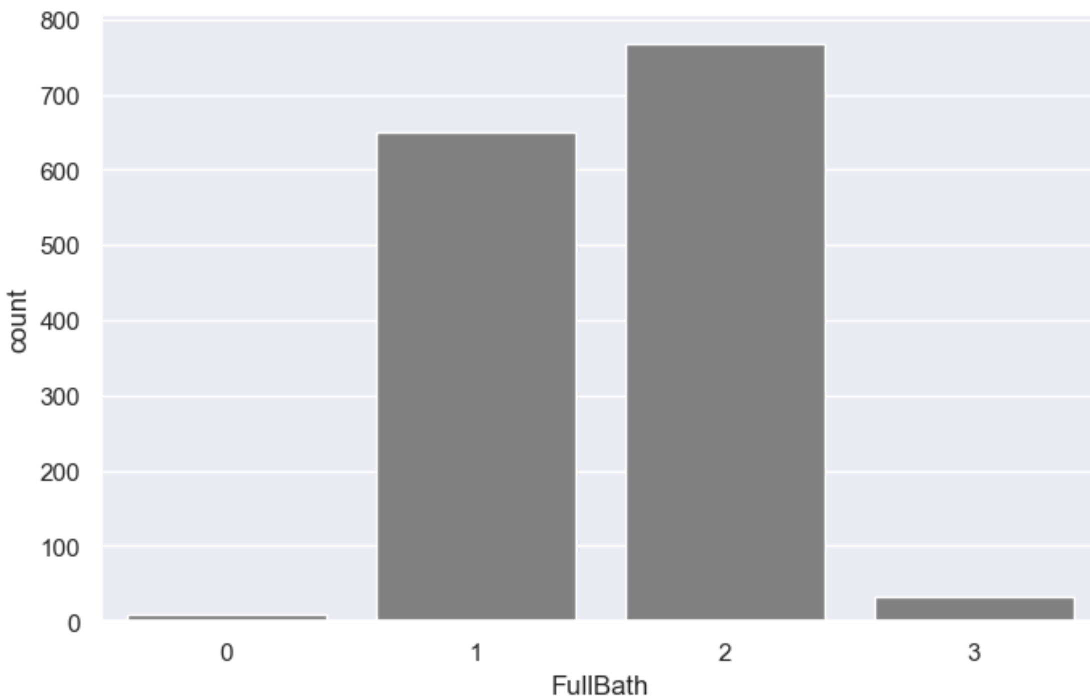
[274]: <AxesSubplot:xlabel='HeatingQC', ylabel='count'>



3.20 Column: FullBath

[275]:
```
sns.countplot(house_train['FullBath'], color = 'gray')
```

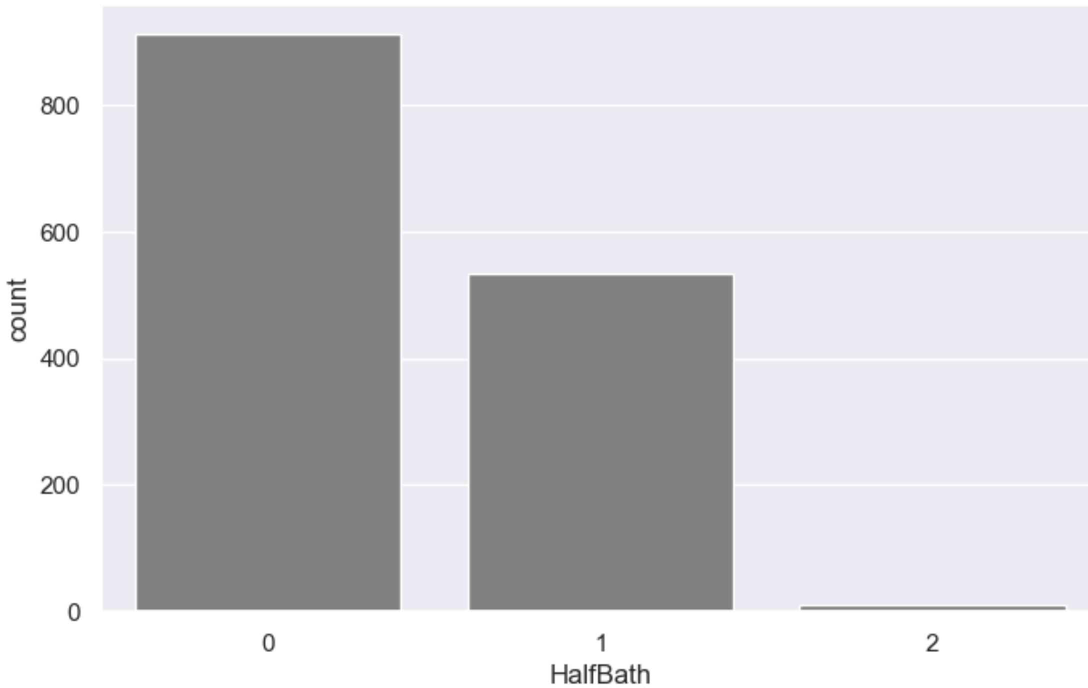
[275]: <AxesSubplot:xlabel='FullBath', ylabel='count'>



3.21 Column: HalfBath

[276]:
```
sns.countplot(house_train['HalfBath'], color = 'gray')
```

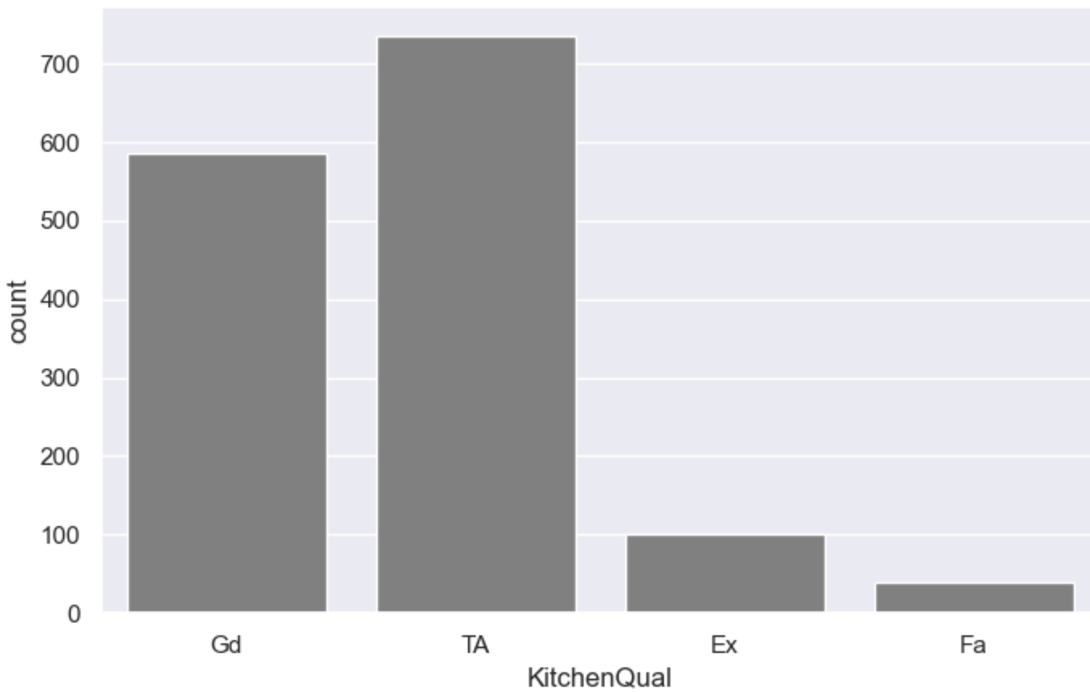
[276]: <AxesSubplot:xlabel='HalfBath', ylabel='count'>



3.22 Column: KitchenQual

[277]:
```
sns.countplot(house_train['KitchenQual'], color = 'gray')
```

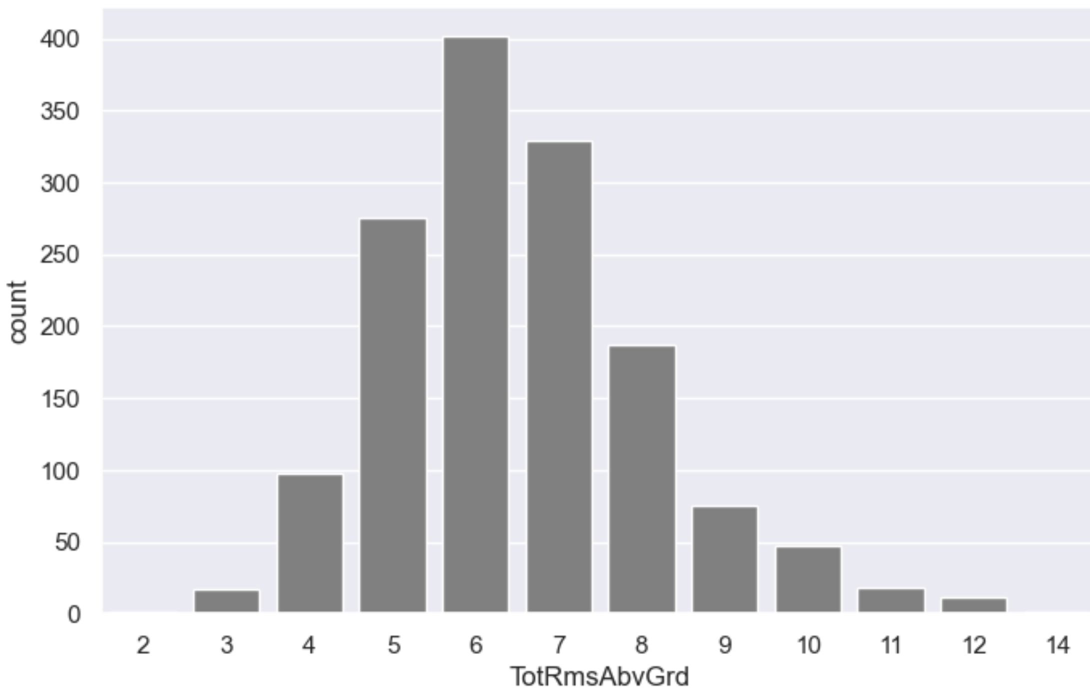
[277]:
```
<AxesSubplot:xlabel='KitchenQual', ylabel='count'>
```



3.23 Column: TotRmsAbvGrd

[278]:
```
sns.countplot(house_train['TotRmsAbvGrd'], color = 'gray')
```

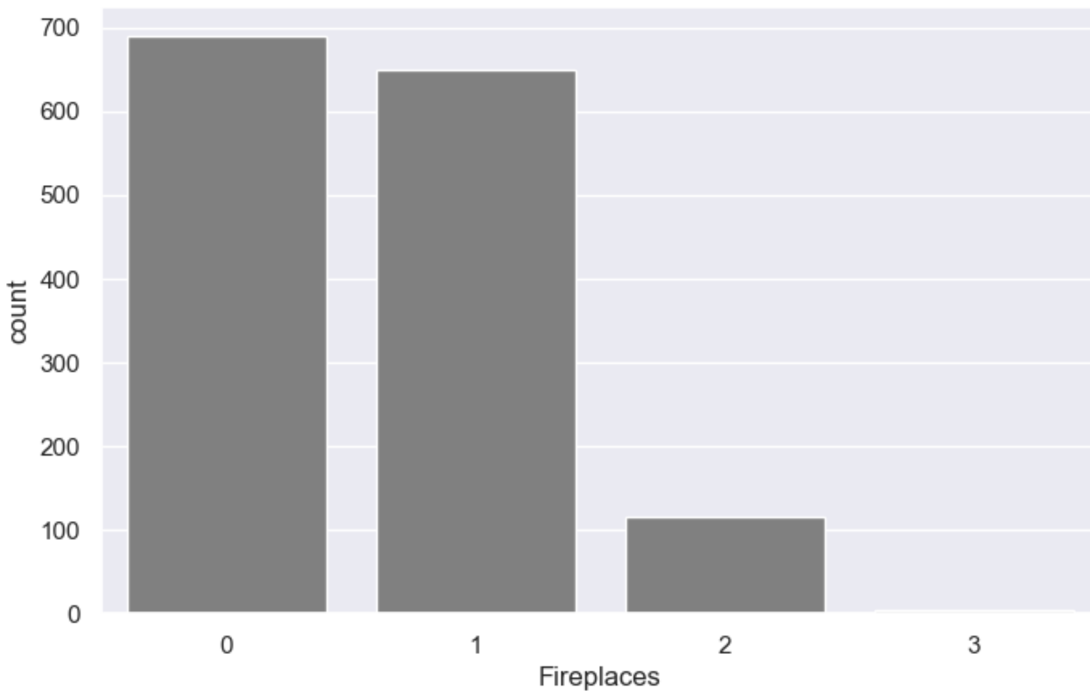
[278]: <AxesSubplot:xlabel='TotRmsAbvGrd', ylabel='count'>



3.24 Column: Fireplaces

[279]:
```
sns.countplot(house_train['Fireplaces'], color = 'gray')
```

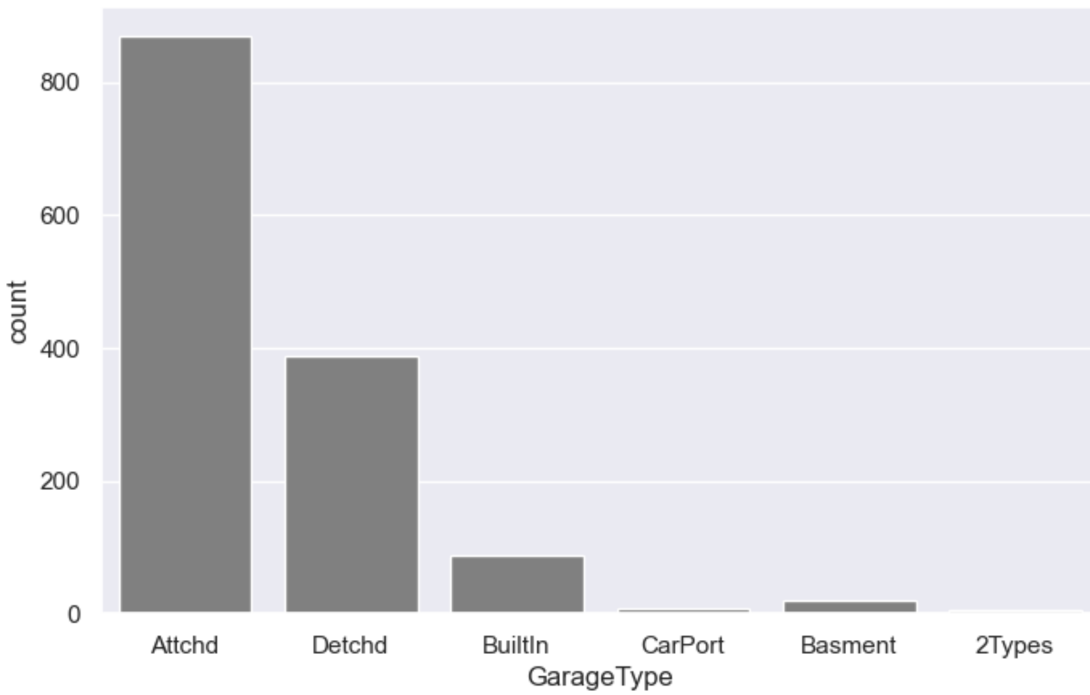
[279]: <AxesSubplot:xlabel='Fireplaces', ylabel='count'>



3.25 Column: GarageType

[280]:
```
sns.countplot(house_train['GarageType'], color = 'gray')
```

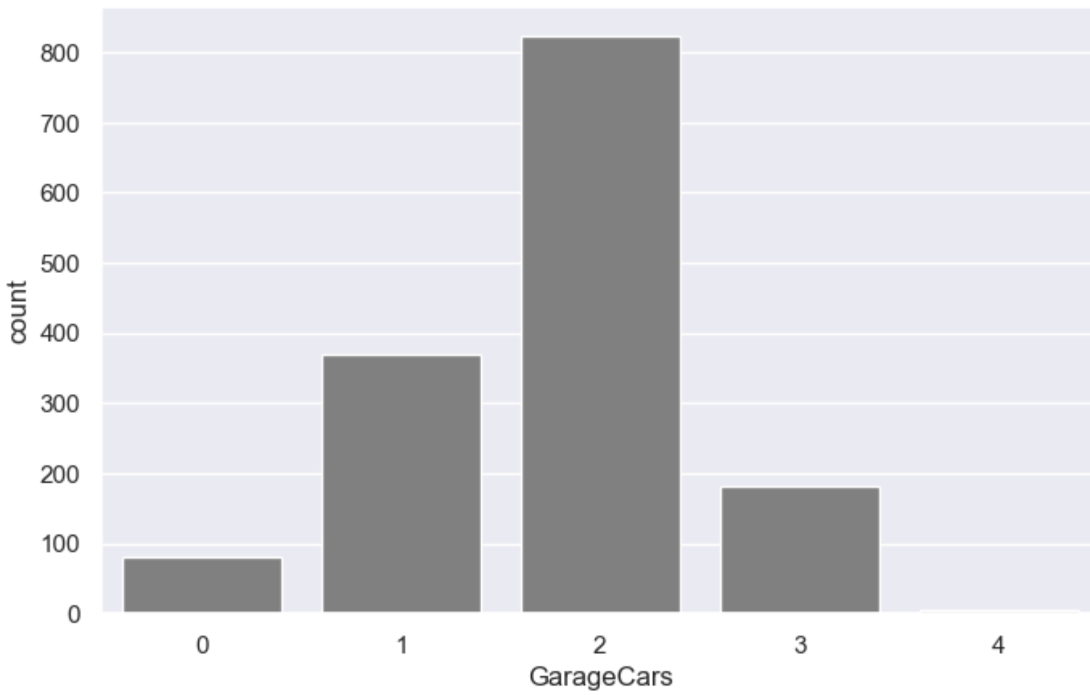
[280]: <AxesSubplot:xlabel='GarageType', ylabel='count'>



3.26 Column: GarageCars

[281]:
```
sns.countplot(house_train['GarageCars'], color = 'gray')
```

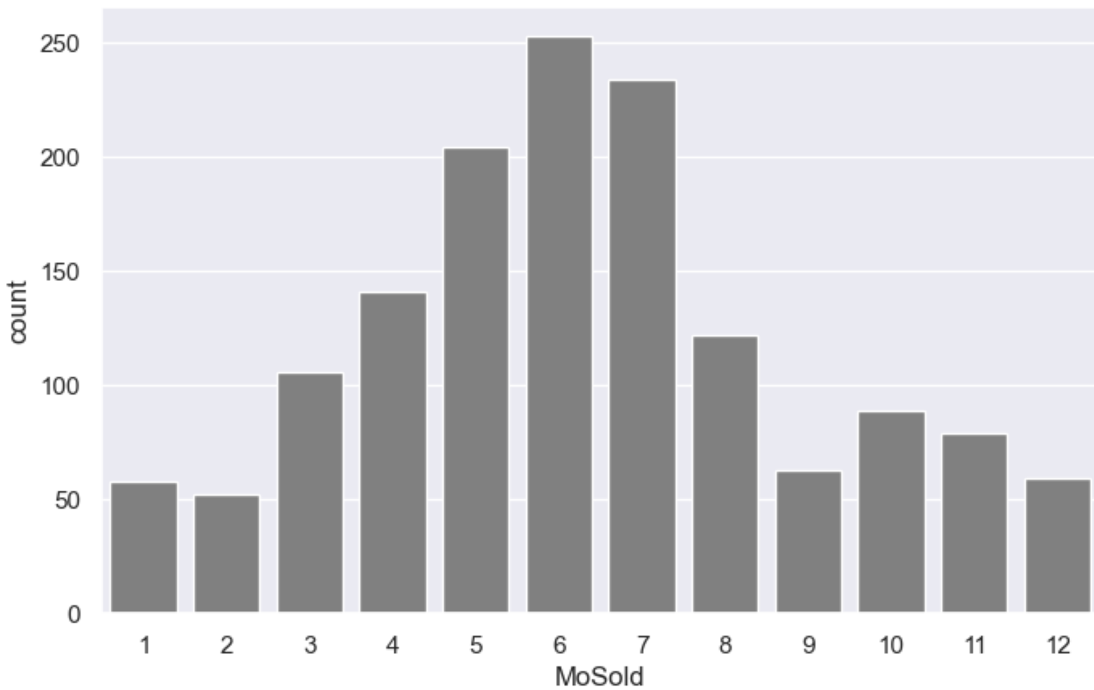
[281]: <AxesSubplot:xlabel='GarageCars', ylabel='count'>



3.27 Column: MoSold

[282]:
```
sns.countplot(house_train['MoSold'], color = 'gray')
```

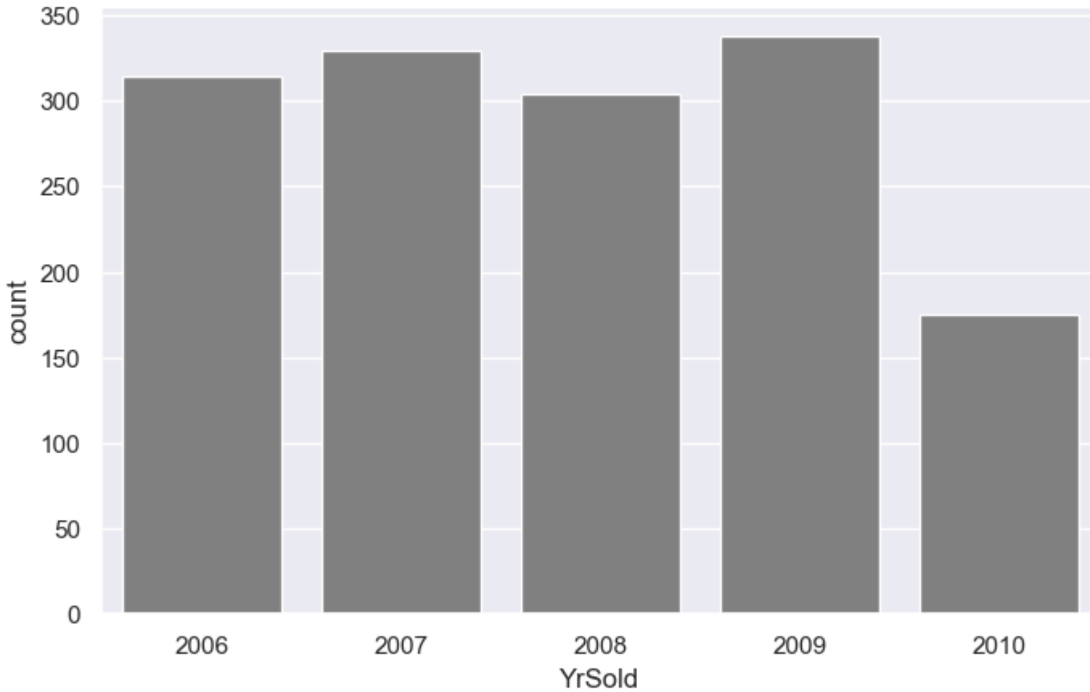
[282]: <AxesSubplot:xlabel='MoSold', ylabel='count'>



3.28 Column: YrSold

[283]:
```
sns.countplot(house_train['YrSold'], color = 'gray')
```

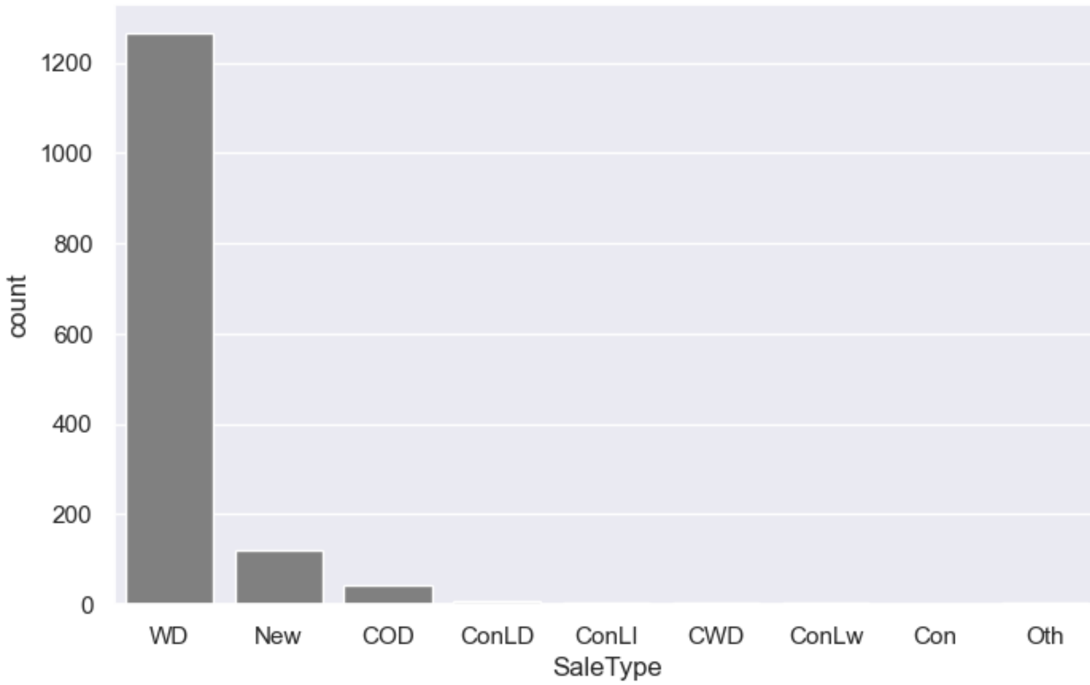
[283]: <AxesSubplot:xlabel='YrSold', ylabel='count'>



3.29 Column: SaleType

[284]:
```
sns.countplot(house_train['SaleType'], color = 'gray')
```

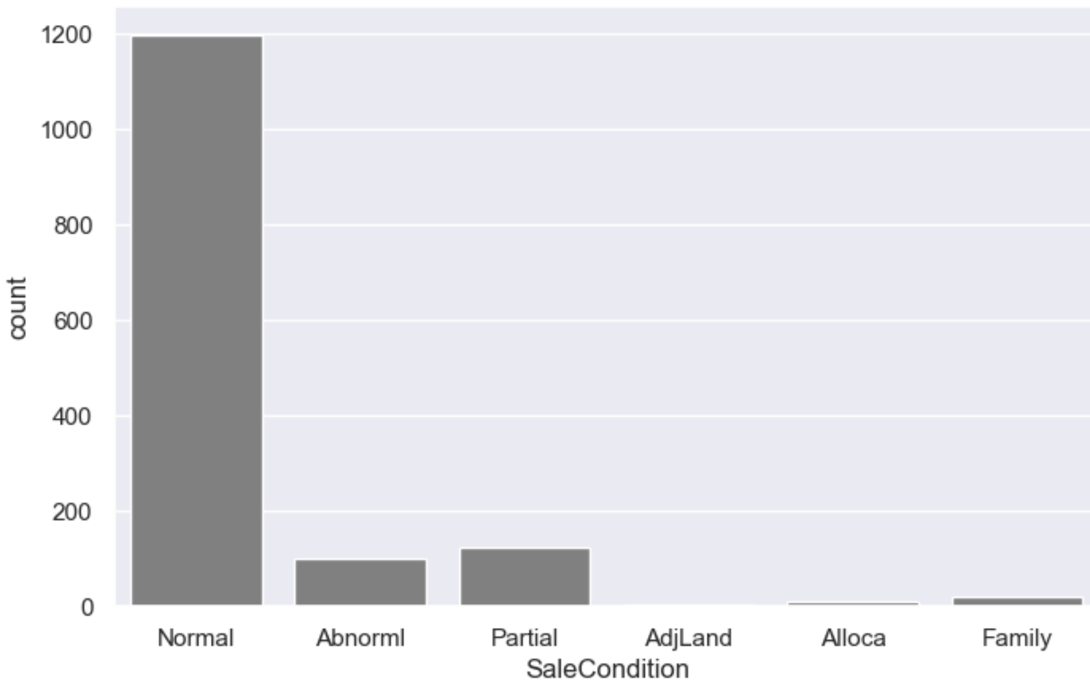
[284]: <AxesSubplot:xlabel='SaleType', ylabel='count'>



3.30 Column: SaleCondition

[285]:
```
sns.countplot(house_train['SaleCondition'], color = 'gray')
```

[285]:
```
<AxesSubplot:xlabel='SaleCondition', ylabel='count'>
```



3.31 Data Visualization - Box Plots and Scatter Plots

[286]:
```
#create a function to plot multi box plots easily

def multi_box(df,cat_col,dist_col,color):

    y = []
    x = []

    print(df[cat_col].unique().tolist())

    for c in set(df[cat_col].unique().tolist()):
        y.append(df[(df[cat_col]==c)][dist_col].values)
        x.append(str(c))

    colors = ['rgba(251, 43, 43, 0.5)', 'rgba(125, 251, 137, 0.5)',
              'rgba(251, 43, 43, 0.5)', 'rgba(125, 251, 137, 0.5)',
              'rgba(251, 43, 43, 0.5)', 'rgba(125, 251, 137, 0.5)',
              'rgba(251, 43, 43, 0.5)', 'rgba(125, 251, 137, 0.5)',
              'rgba(251, 43, 43, 0.5)', 'rgba(125, 251, 137, 0.5)',
              'rgba(251, 43, 43, 0.5)', 'rgba(125, 251, 137, 0.5)']

    traces = []
```

```
    for xd, yd, cls in zip(x, y, colors[:2*len(df[cat_col].unique())]):
            traces.append(go.Box(y=yd,
                                 name=xd,
                                 boxpoints='all',
                                 jitter=0.5,
                                 whiskerwidth=0.2,
                                 fillcolor=cls,
                                 marker=dict(size=2),
                                 line=dict(width=1)))

    layout = go.Layout(title='{} distribution grouped by {}'.format(dist_col.
 ↪title(), cat_col.title()),
        xaxis=dict(title=cat_col,
                   titlefont=dict(size=16)),

        yaxis=dict(title='Distribution',
                   autorange=True,
                   showgrid=True,
                   zeroline=True,
                   dtick=100000,
                   gridcolor='rgb(255, 255, 255)',
                   gridwidth=1,
                   zerolinecolor='rgb(255, 255, 255)',
                   zerolinewidth=2,
                   titlefont=dict(
                   size=16)),

        margin=dict(l=40,
                    r=30,
                    b=80,
                    t=100),

        paper_bgcolor='rgb(255, 255, 255)',
        plot_bgcolor='rgb(255, 243, 192)',
        showlegend=False)

    fig = go.Figure(data=traces, layout=layout)
    iplot(fig)
```

3.32 Saleprice distribution for Neighborhood

[287]:
```
print(house_train['Neighborhood'].unique().tolist())

multi_box(house_train[['Neighborhood','SalePrice']].
 ↪dropna(),'Neighborhood','SalePrice','coral')
```

```
['CollgCr', 'Veenker', 'Crawfor', 'NoRidge', 'Mitchel', 'Somerst', 'NWAmes',
```

```
'OldTown', 'BrkSide', 'Sawyer', 'NridgHt', 'NAmes', 'SawyerW', 'IDOTRR',
'MeadowV', 'Edwards', 'Timber', 'Gilbert', 'StoneBr', 'ClearCr', 'NPkVill',
'Blmngtn', 'BrDale', 'SWISU', 'Blueste']
['CollgCr', 'Veenker', 'Crawfor', 'NoRidge', 'Mitchel', 'Somerst', 'NWAmes',
'OldTown', 'BrkSide', 'Sawyer', 'NridgHt', 'NAmes', 'SawyerW', 'IDOTRR',
'MeadowV', 'Edwards', 'Timber', 'Gilbert', 'StoneBr', 'ClearCr', 'NPkVill',
'Blmngtn', 'BrDale', 'SWISU', 'Blueste']
```

3.33 Saleprice distribution for Basement Height

[288]:
```
print(house_train['BsmtQual'].unique().tolist())

multi_box(house_train[['BsmtQual','SalePrice']].
↪dropna(),'BsmtQual','SalePrice','coral')
```

```
['Gd', 'TA', 'Ex', nan, 'Fa']
['Gd', 'TA', 'Ex', 'Fa']
```

3.34 Saleprice distribution for Exterior Material quality

[289]:
```
print(house_train['ExterQual'].unique().tolist())

multi_box(house_train[['ExterQual','SalePrice']].
↪dropna(),'ExterQual','SalePrice','coral')
```

```
['Gd', 'TA', 'Ex', 'Fa']
['Gd', 'TA', 'Ex', 'Fa']
```

3.35 Saleprice distribution for Fireplace Quality

[290]:
```
print(house_train['FireplaceQu'].unique().tolist())

multi_box(house_train[['FireplaceQu','SalePrice']].
↪dropna(),'FireplaceQu','SalePrice','coral')
```

```
[nan, 'TA', 'Gd', 'Fa', 'Ex', 'Po']
['TA', 'Gd', 'Fa', 'Ex', 'Po']
```

3.36 Saleprice distribution for Present condition of the material of the exterior

[291]:
```
print(house_train['ExterCond'].unique().tolist())

multi_box(house_train[['ExterCond','SalePrice']].
↪dropna(),'ExterCond','SalePrice','coral')
```

```
['TA', 'Gd', 'Fa', 'Po', 'Ex']
['TA', 'Gd', 'Fa', 'Po', 'Ex']
```

3.37 Saleprice distribution for Kitchen Quality

[292]:
```
print(house_train['KitchenQual'].unique().tolist())

multi_box(house_train[['KitchenQual','SalePrice']].
 ↪dropna(),'KitchenQual','SalePrice','coral')
```

```
['Gd', 'TA', 'Ex', 'Fa']
['Gd', 'TA', 'Ex', 'Fa']
```

3.38 Saleprice distribution for General shape of the property

[293]:
```
print(house_train['LotShape'].unique().tolist())

multi_box(house_train[['LotShape','SalePrice']].
 ↪dropna(),'LotShape','SalePrice','coral')
```

```
['Reg', 'IR1', 'IR2', 'IR3']
['Reg', 'IR1', 'IR2', 'IR3']
```

3.39 Saleprice distribution for Overall material and finish quality

[294]:
```
print(house_train['OverallQual'].unique().tolist())

multi_box(house_train[['OverallQual','SalePrice']].
 ↪dropna(),'OverallQual','SalePrice','coral')
```

```
[7, 6, 8, 5, 9, 4, 10, 3, 1, 2]
[7, 6, 8, 5, 9, 4, 10, 3, 1, 2]
```

3.40 Saleprice distribution for Number of full bathrooms above grade

[295]:
```
print(house_train['FullBath'].unique().tolist())

multi_box(house_train[['FullBath','SalePrice']].
 ↪dropna(),'FullBath','SalePrice','coral')
```

```
[2, 1, 3, 0]
[2, 1, 3, 0]
```

3.41 Saleprice distribution for Number of half bathrooms above grade

[296]:
```
print(house_train['HalfBath'].unique().tolist())

multi_box(house_train[['HalfBath','SalePrice']].
 ↪dropna(),'HalfBath','SalePrice','coral')
```

```
[1, 0, 2]
[1, 0, 2]
```

3.42 Saleprice distribution for Total rooms above grade (does not include bathrooms)

[297]:
```
print(house_train['TotRmsAbvGrd'].unique().tolist())

multi_box(house_train[['TotRmsAbvGrd','SalePrice']].
 ↪dropna(),'TotRmsAbvGrd','SalePrice','coral')
```

```
[8, 6, 7, 9, 5, 11, 4, 10, 12, 3, 2, 14]
[8, 6, 7, 9, 5, 11, 4, 10, 12, 3, 2, 14]
```

3.43 Saleprice distribution for Number of fireplaces

[298]:
```
print(house_train['Fireplaces'].unique().tolist())

multi_box(house_train[['Fireplaces','SalePrice']].
 ↪dropna(),'Fireplaces','SalePrice','coral')
```

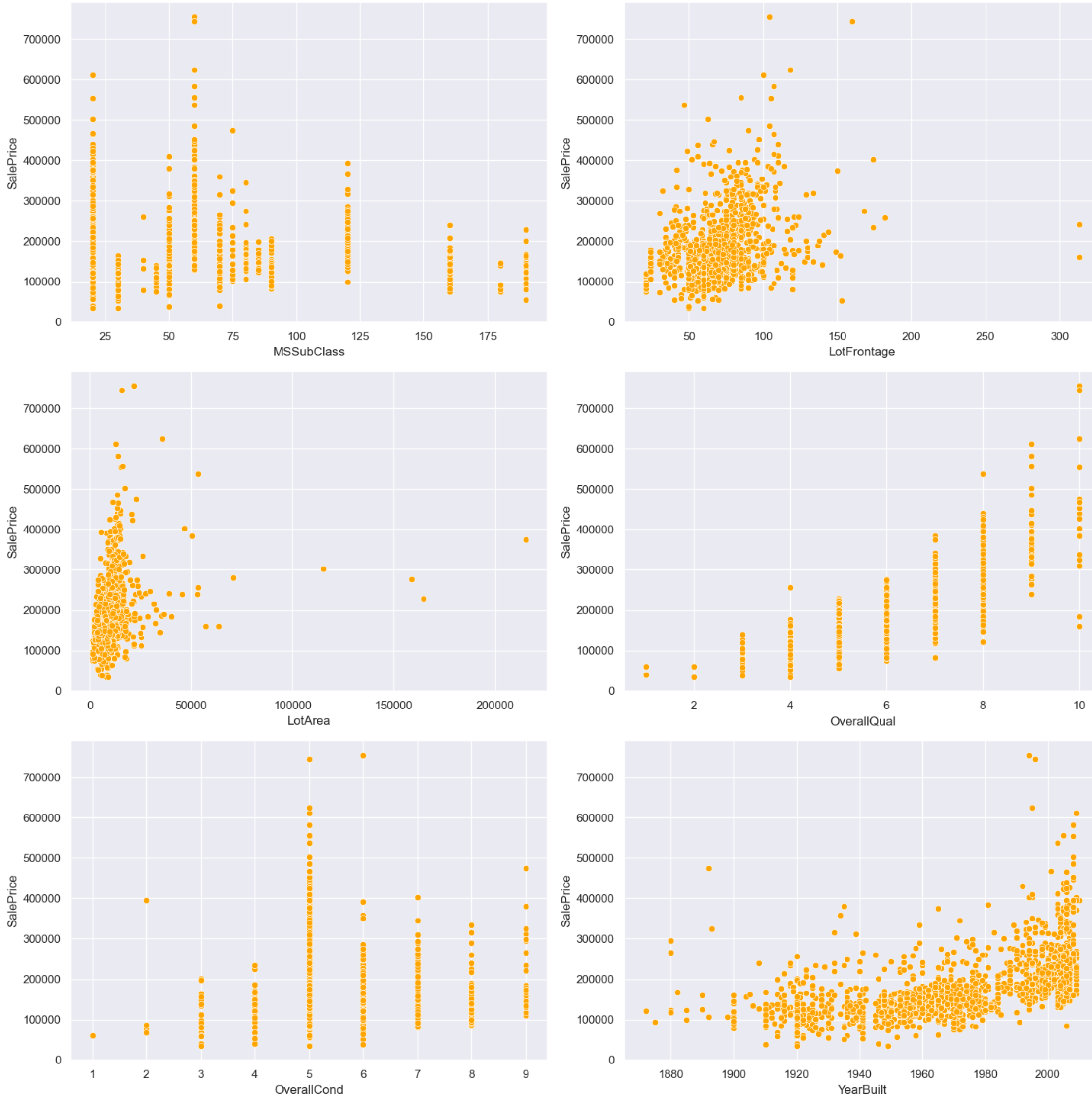
```
[0, 1, 2, 3]
[0, 1, 2, 3]
```

3.44 Saleprice distribution for Number of kitchens above grade

[299]:
```
print(house_train['KitchenAbvGr'].unique().tolist())

multi_box(house_train[['KitchenAbvGr','SalePrice']].
 ↪dropna(),'KitchenAbvGr','SalePrice','coral')
```

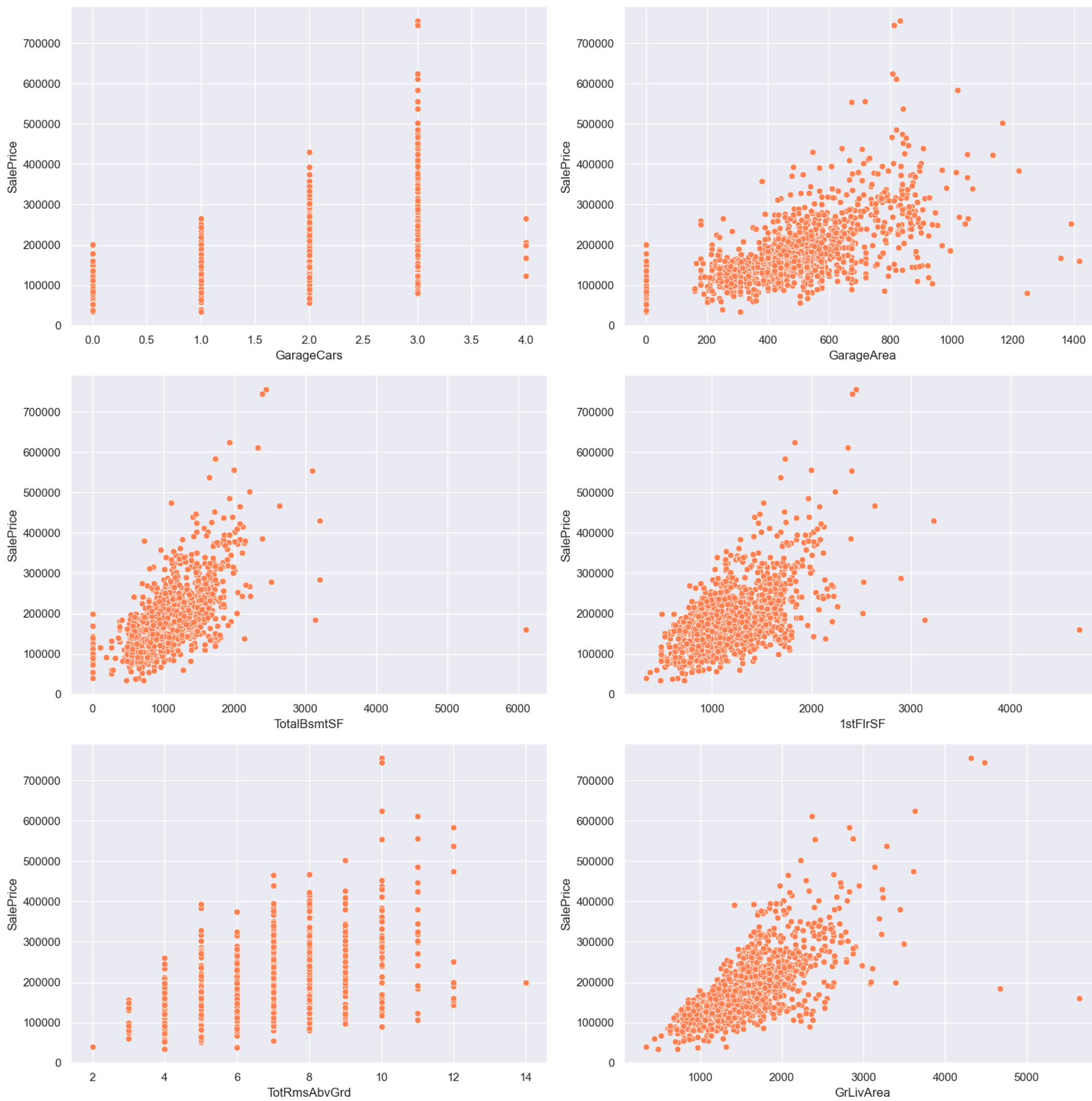
```
[1, 2, 3, 0]
[1, 2, 3, 0]
```

[300]:
```
sns.set(style="darkgrid")

fig,ax=plt.subplots(3,2,figsize=(15,15))
sns.scatterplot(house_train_n['MSSubClass'],house_train_n['SalePrice'],
                ax=ax[0][0],color='orange')
sns.scatterplot(house_train_n['LotFrontage'],house_train_n['SalePrice'],
                ax=ax[0][1],color='orange')
sns.scatterplot(house_train_n['LotArea'],house_train_n['SalePrice'],
                ax=ax[1][0],color='orange')
sns.scatterplot(house_train_n['OverallQual'],house_train_n['SalePrice'],
                ax=ax[1][1],color='orange')
sns.scatterplot(house_train_n['OverallCond'],house_train_n['SalePrice'],
                ax=ax[2][0],color='orange')
sns.scatterplot(house_train_n['YearBuilt'],house_train_n['SalePrice'],
                ax=ax[2][1],color='orange')

fig.tight_layout()
```



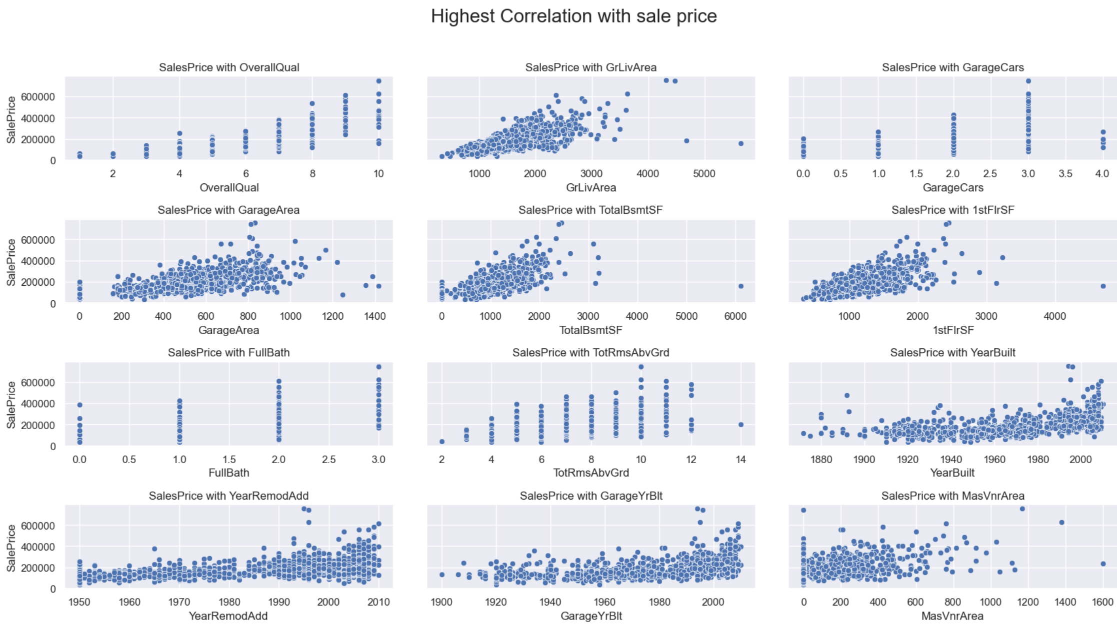
[301]:
```
sns.set(style="darkgrid")

fig,ax=plt.subplots(3,2,figsize=(15,15))
sns.scatterplot(house_train_n['GarageCars'],house_train_n['SalePrice'],
                ax=ax[0][0],color='coral')
sns.scatterplot(house_train_n['GarageArea'],house_train_n['SalePrice'],
                ax=ax[0][1],color='coral')
sns.scatterplot(house_train_n['TotalBsmtSF'],house_train_n['SalePrice'],
                ax=ax[1][0],color='coral')
sns.scatterplot(house_train_n['1stFlrSF'],house_train_n['SalePrice'],
                ax=ax[1][1],color='coral')
sns.scatterplot(house_train_n['TotRmsAbvGrd'],house_train_n['SalePrice'],
                ax=ax[2][0],color='coral')
```

```
sns.scatterplot(house_train_n['GrLivArea'],house_train_n['SalePrice'],
                ax=ax[2][1],color='coral')
fig.tight_layout()
```



[302]:
```
#Visualize columns have corr with SalePrice

high_corr = corr['SalePrice'].sort_values(ascending=False)[1:][:13].index.
 ↪tolist()

fig, axes = plt.subplots(4,3, figsize=(20, 10), sharey=True);
plt.subplots_adjust(hspace = 0.7, wspace=0.1)
fig.suptitle('Highest Correlation with sale price', fontsize=20);
```

```
for i,col in zip(range(12),high_corr):
    sns.scatterplot(y=house_train['SalePrice'], x=house_train[col], ax=axes[i//
 ↪3][i%3])
    axes[i//3][i%3].set_title('SalesPrice with '+col)
```



4 Setup and Basic EDA Part II

[303]:
```
cat_cols =␣
 ↪["Neighborhood","BsmtQual","ExterQual","FireplaceQu","ExterCond","KitchenQual",
            "LotShape",␣
 ↪"OverallQual","FullBath","HalfBath","TotRmsAbvGrd","Fireplaces",
            "KitchenAbvGr"]

encode_cols = ["Neighborhood","BsmtQual","ExterQual","FireplaceQu","ExterCond",
               "KitchenQual","LotShape"]
```

4.1 Label Encoding

[304]:
```
def count_nulls(df):
    nulls_df = df.isnull().sum().to_frame().rename(columns={0:'Null values'})
    nulls_df = nulls_df[~(nulls_df['Null values'] <= 0)]
    return nulls_df
```

[305]:
```
count_nulls(house_train)
```

[305]:

```
                Null values
LotFrontage            259
Alley                 1369
MasVnrType               8
MasVnrArea               8
BsmtQual                37
BsmtCond                37
BsmtExposure            38
BsmtFinType1            37
BsmtFinType2            38
Electrical               1
FireplaceQu            690
GarageType              81
GarageYrBlt             81
GarageFinish            81
GarageQual              81
GarageCond              81
PoolQC                1453
Fence                 1179
MiscFeature           1406
```

[306]:

```
house_train_c
```

[306]:

```
     MSZoning Street Alley LotShape LandContour Utilities LotConfig LandSlope  \
0          RL   Pave   NaN      Reg         Lvl    AllPub    Inside       Gtl
1          RL   Pave   NaN      Reg         Lvl    AllPub       FR2       Gtl
2          RL   Pave   NaN      IR1         Lvl    AllPub    Inside       Gtl
3          RL   Pave   NaN      IR1         Lvl    AllPub    Corner       Gtl
4          RL   Pave   NaN      IR1         Lvl    AllPub       FR2       Gtl
…         …    …   …      …         …    …       …       …
1455       RL   Pave   NaN      Reg         Lvl    AllPub    Inside       Gtl
1456       RL   Pave   NaN      Reg         Lvl    AllPub    Inside       Gtl
1457       RL   Pave   NaN      Reg         Lvl    AllPub    Inside       Gtl
1458       RL   Pave   NaN      Reg         Lvl    AllPub    Inside       Gtl
1459       RL   Pave   NaN      Reg         Lvl    AllPub    Inside       Gtl

     Neighborhood Condition1  … GarageType GarageFinish GarageQual  \
0         CollgCr       Norm  …     Attchd          RFn         TA
1         Veenker      Feedr  …     Attchd          RFn         TA
2         CollgCr       Norm  …     Attchd          RFn         TA
3         Crawfor       Norm  …     Detchd          Unf         TA
4         NoRidge       Norm  …     Attchd          RFn         TA
…             …    … …        …          …        …
1455      Gilbert       Norm  …     Attchd          RFn         TA
1456       NWAmes       Norm  …     Attchd          Unf         TA
1457      Crawfor       Norm  …     Attchd          RFn         TA
1458        NAmes       Norm  …     Attchd          Unf         TA
```

```
1459      Edwards       Norm  …     Attchd         Fin         TA

     GarageCond PavedDrive PoolQC  Fence MiscFeature SaleType SaleCondition
0            TA          Y    NaN    NaN         NaN       WD        Normal
1            TA          Y    NaN    NaN         NaN       WD        Normal
2            TA          Y    NaN    NaN         NaN       WD        Normal
3            TA          Y    NaN    NaN         NaN       WD       Abnorml
4            TA          Y    NaN    NaN         NaN       WD        Normal
…          …          …    …    …          …       …           …
1455         TA          Y    NaN    NaN         NaN       WD        Normal
1456         TA          Y    NaN  MnPrv         NaN       WD        Normal
1457         TA          Y    NaN  GdPrv        Shed       WD        Normal
1458         TA          Y    NaN    NaN         NaN       WD        Normal
1459         TA          Y    NaN    NaN         NaN       WD        Normal

[1460 rows x 43 columns]
```

[307]:
```
house_train[cat_cols]
```

[307]:
```
     Neighborhood BsmtQual ExterQual FireplaceQu ExterCond KitchenQual  \
0         CollgCr       Gd        Gd         NaN        TA          Gd
1         Veenker       Gd        TA          TA        TA          TA
2         CollgCr       Gd        Gd          TA        TA          Gd
3         Crawfor       TA        TA          Gd        TA          Gd
4         NoRidge       Gd        Gd          TA        TA          Gd
…             …      …       …         …        …          …
1455      Gilbert       Gd        TA          TA        TA          TA
1456       NWAmes       Gd        TA          TA        TA          TA
1457      Crawfor       TA        Ex          Gd        Gd          Gd
1458        NAmes       TA        TA         NaN        TA          Gd
1459      Edwards       TA        Gd         NaN        TA          TA

     LotShape  OverallQual  FullBath  HalfBath  TotRmsAbvGrd  Fireplaces  \
0         Reg            7         2         1             8           0
1         Reg            6         2         0             6           1
2         IR1            7         2         1             6           1
3         IR1            7         1         0             7           1
4         IR1            8         2         1             9           1
…         …          …       …       …          …           …
1455      Reg            6         2         1             7           1
1456      Reg            6         2         0             7           2
1457      Reg            7         2         0             9           2
1458      Reg            5         1         0             5           0
1459      Reg            5         1         1             6           0

      KitchenAbvGr
0                1
```

```
1                1
2                1
3                1
4                1
…            …
1455             1
1456             1
1457             1
1458             1
1459             1

[1460 rows x 13 columns]
```

[308]:
```
house_train['Neighborhood'].unique().tolist()
```

[308]:
```
['CollgCr',
 'Veenker',
 'Crawfor',
 'NoRidge',
 'Mitchel',
 'Somerst',
 'NWAmes',
 'OldTown',
 'BrkSide',
 'Sawyer',
 'NridgHt',
 'NAmes',
 'SawyerW',
 'IDOTRR',
 'MeadowV',
 'Edwards',
 'Timber',
 'Gilbert',
 'StoneBr',
 'ClearCr',
 'NPkVill',
 'Blmngtn',
 'BrDale',
 'SWISU',
 'Blueste']
```

[309]:
```
def unique_values(df, keys):
    for item in keys:
        values = df[item].unique().tolist()
        print(f'{item}: {values}')
```

[310]:
```
unique_values(house_train, cat_cols)
```

```
Neighborhood: ['CollgCr', 'Veenker', 'Crawfor', 'NoRidge', 'Mitchel', 'Somerst',
'NWAmes', 'OldTown', 'BrkSide', 'Sawyer', 'NridgHt', 'NAmes', 'SawyerW',
'IDOTRR', 'MeadowV', 'Edwards', 'Timber', 'Gilbert', 'StoneBr', 'ClearCr',
'NPkVill', 'Blmngtn', 'BrDale', 'SWISU', 'Blueste'])
BsmtQual: ['Gd', 'TA', 'Ex', nan, 'Fa'])
ExterQual: ['Gd', 'TA', 'Ex', 'Fa'])
FireplaceQu: [nan, 'TA', 'Gd', 'Fa', 'Ex', 'Po'])
ExterCond: ['TA', 'Gd', 'Fa', 'Po', 'Ex'])
KitchenQual: ['Gd', 'TA', 'Ex', 'Fa'])
LotShape: ['Reg', 'IR1', 'IR2', 'IR3'])
OverallQual: [7, 6, 8, 5, 9, 4, 10, 3, 1, 2])
FullBath: [2, 1, 3, 0])
HalfBath: [1, 0, 2])
TotRmsAbvGrd: [8, 6, 7, 9, 5, 11, 4, 10, 12, 3, 2, 14])
Fireplaces: [0, 1, 2, 3])
KitchenAbvGr: [1, 2, 3, 0])
```

[311]:
```python
unique_values(house_test, cat_cols)
```

```
Neighborhood: ['NAmes', 'Gilbert', 'StoneBr', 'BrDale', 'NPkVill', 'NridgHt',
'Blmngtn', 'NoRidge', 'Somerst', 'SawyerW', 'Sawyer', 'NWAmes', 'OldTown',
'BrkSide', 'ClearCr', 'SWISU', 'Edwards', 'CollgCr', 'Crawfor', 'Blueste',
'IDOTRR', 'Mitchel', 'Timber', 'MeadowV', 'Veenker'])
BsmtQual: ['TA', 'Gd', 'Ex', 'Fa', nan])
ExterQual: ['TA', 'Gd', 'Ex', 'Fa'])
FireplaceQu: [nan, 'TA', 'Gd', 'Po', 'Fa', 'Ex'])
ExterCond: ['TA', 'Gd', 'Fa', 'Po', 'Ex'])
KitchenQual: ['TA', 'Gd', 'Ex', 'Fa', nan])
LotShape: ['Reg', 'IR1', 'IR2', 'IR3'])
OverallQual: [5, 6, 8, 7, 4, 9, 2, 3, 10, 1])
FullBath: [1, 2, 3, 4, 0])
HalfBath: [0, 1, 2])
TotRmsAbvGrd: [5, 6, 7, 4, 10, 8, 9, 3, 12, 11, 13, 15])
Fireplaces: [0, 1, 2, 3, 4])
KitchenAbvGr: [1, 2, 0])
```

[312]:
```python
def fill_nulls(df, key, value):
    df[key].fillna(value, inplace = True)
    null_count = df[key].isnull().sum()
    print(f'Null count for {key} = {null_count}')
```

[313]:
```python
fill_nulls(house_train, 'BsmtQual', 'N')
```

```
Null count for BsmtQual = 0
```

[314]:
```python
fill_nulls(house_test, 'BsmtQual', 'N')
```

```
Null count for BsmtQual = 0
```

[315]:
```
fill_nulls(house_train, 'FireplaceQu', 'N')
```

```
Null count for FireplaceQu = 0
```

[316]:
```
fill_nulls(house_test, 'FireplaceQu', 'N')
```

```
Null count for FireplaceQu = 0
```

[317]:
```
fill_nulls(house_test, 'KitchenQual', 'N')
```

```
Null count for KitchenQual = 0
```

[318]:
```
# Import LabelEncoder from sklearn.preprocessing
from sklearn.preprocessing import LabelEncoder

# Iterate through each category column and convert to numeric using␣
↪LabelEncoder. Then transform the column
# and assign back to the original column
for key in encode_cols:
    print(f'Label Encoding column: {key}')
    le = LabelEncoder()
    labels = list(house_train[key].unique())
    labels += list(house_test[key].unique())

    # Create mapping from labels to integers
    le.fit(labels)
    # Transform the train and test consistently
    house_train[key] = le.transform(house_train[key])
    house_test[key] = le.transform(house_test[key])
```

```
Label Encoding column: Neighborhood
Label Encoding column: BsmtQual
Label Encoding column: ExterQual
Label Encoding column: FireplaceQu
Label Encoding column: ExterCond
Label Encoding column: KitchenQual
Label Encoding column: LotShape
```

[319]:
```
unique_values(house_train, cat_cols)
```

```
Neighborhood: [5, 24, 6, 15, 11, 21, 14, 17, 3, 19, 16, 12, 20, 9, 10, 7, 23, 8,
22, 4, 13, 0, 2, 18, 1])
BsmtQual: [2, 4, 0, 3, 1])
ExterQual: [2, 3, 0, 1])
FireplaceQu: [3, 5, 2, 1, 0, 4])
ExterCond: [4, 2, 1, 3, 0])
KitchenQual: [2, 4, 0, 1])
LotShape: [3, 0, 1, 2])
OverallQual: [7, 6, 8, 5, 9, 4, 10, 3, 1, 2])
FullBath: [2, 1, 3, 0])
```

```
HalfBath: [1, 0, 2])
TotRmsAbvGrd: [8, 6, 7, 9, 5, 11, 4, 10, 12, 3, 2, 14])
Fireplaces: [0, 1, 2, 3])
KitchenAbvGr: [1, 2, 3, 0])
```

[320]:
```
unique_values(house_test, cat_cols)
```

```
Neighborhood: [12, 8, 22, 2, 13, 16, 0, 15, 21, 20, 19, 14, 17, 3, 4, 18, 7, 5,
6, 1, 9, 11, 23, 10, 24])
BsmtQual: [4, 2, 0, 1, 3])
ExterQual: [3, 2, 0, 1])
FireplaceQu: [3, 5, 2, 4, 1, 0])
ExterCond: [4, 2, 1, 3, 0])
KitchenQual: [4, 2, 0, 1, 3])
LotShape: [3, 0, 1, 2])
OverallQual: [5, 6, 8, 7, 4, 9, 2, 3, 10, 1])
FullBath: [1, 2, 3, 4, 0])
HalfBath: [0, 1, 2])
TotRmsAbvGrd: [5, 6, 7, 4, 10, 8, 9, 3, 12, 11, 13, 15])
Fireplaces: [0, 1, 2, 3, 4])
KitchenAbvGr: [1, 2, 0])
```

[321]:
```
numeric_predictors = list_subtract(house_train_n.columns.tolist(), ignore_cols)
```

[322]:
```
count_nulls(house_train[numeric_predictors])
```

[322]:

	Null values
MasVnrArea	8
GarageYrBlt	81

[323]:
```
house_train[numeric_predictors] = house_train[numeric_predictors].
↪fillna(house_train[numeric_predictors].median())
```

[324]:
```
count_nulls(house_train[numeric_predictors])
```

[324]:
```
Empty DataFrame
Columns: [Null values]
Index: []
```

[325]:
```
count_nulls(house_test[numeric_predictors])
```

[325]:

	Null values
MasVnrArea	15
BsmtFinSF1	1
BsmtFinSF2	1
BsmtUnfSF	1
TotalBsmtSF	1
BsmtFullBath	2

```
BsmtHalfBath           2
GarageYrBlt           78
GarageCars             1
GarageArea             1
```

[327]:
```
house_test[numeric_predictors] = house_test[numeric_predictors].
 ↪fillna(house_test[numeric_predictors].median())
```

[328]:
```
count_nulls(house_test[numeric_predictors])
```

[328]:
```
Empty DataFrame
Columns: [Null values]
Index: []
```

[329]:
```
set(numeric_predictors).intersection(set(cat_cols))
```

[329]:
```
{'Fireplaces',
 'FullBath',
 'HalfBath',
 'KitchenAbvGr',
 'OverallQual',
 'TotRmsAbvGrd'}
```

5 Creating arrays for the features and the response variable.

[330]:
```
target = ['SalePrice']
predictors = list(set(numeric_predictors).union(set(cat_cols)))
predictors
```

[330]:
```
['2ndFlrSF',
 'BsmtHalfBath',
 'GarageCars',
 'WoodDeckSF',
 'BsmtFinSF1',
 'MiscVal',
 'ExterCond',
 'MoSold',
 'YrSold',
 'FullBath',
 'YearRemodAdd',
 'MSSubClass',
 'PoolArea',
 'LotArea',
 'YearBuilt',
 'BsmtFinSF2',
 'KitchenAbvGr',
 'OverallQual',
```

```
 'ScreenPorch',
 'KitchenQual',
 'OverallCond',
 'TotalBsmtSF',
 'ExterQual',
 'GarageYrBlt',
 'FireplaceQu',
 'TotRmsAbvGrd',
 'GarageArea',
 'Neighborhood',
 '1stFlrSF',
 'MasVnrArea',
 'LotShape',
 'BsmtUnfSF',
 '3SsnPorch',
 'HalfBath',
 'BsmtQual',
 'Fireplaces',
 'EnclosedPorch',
 'GrLivArea',
 'BsmtFullBath',
 'LowQualFinSF',
 'OpenPorchSF',
 'BedroomAbvGr']
```

[331]:
```
count_nulls(house_train[predictors])
```

[331]:
```
Empty DataFrame
Columns: [Null values]
Index: []
```

[332]:
```
count_nulls(house_test[predictors])
```

[332]:
```
Empty DataFrame
Columns: [Null values]
Index: []
```

[333]:
```
def log_transform(df, cols):
    for col in cols:
        log_col = 'log' + col
        df[log_col] = np.log(1 + df[col])
    return df
```

[334]:
```
log_transform(house_train, skew_cols)[predictors]
```

[334]:
```
      2ndFlrSF  BsmtHalfBath  GarageCars  WoodDeckSF  BsmtFinSF1  MiscVal  \
0          854             0           2           0         706        0
```

```
1          0          1         2      298      978        0
2        866          0         2        0      486        0
3        756          0         3        0      216        0
4       1053          0         3      192      655        0
…         …          …        …        …        …      …
1455     694          0         2        0        0        0
1456       0          0         2      349      790        0
1457    1152          0         1        0      275     2500
1458       0          0         1      366       49        0
1459       0          0         1      736      830        0

      ExterCond  MoSold  YrSold  FullBath  …  3SsnPorch  HalfBath  BsmtQual  \
0             4       2    2008         2  …          0         1         2
1             4       5    2007         2  …          0         0         2
2             4       9    2008         2  …          0         1         2
3             4       2    2006         1  …          0         0         4
4             4      12    2008         2  …          0         1         2
…           …     …     …        …  …        …        …       …
1455          4       8    2007         2  …          0         1         2
1456          4       2    2010         2  …          0         0         2
1457          2       5    2010         2  …          0         0         4
1458          4       4    2010         1  …          0         0         4
1459          4       6    2008         1  …          0         1         4

      Fireplaces  EnclosedPorch  GrLivArea  BsmtFullBath  LowQualFinSF  \
0              0              0       1710             1             0
1              1              0       1262             0             0
2              1              0       1786             1             0
3              1            272       1717             1             0
4              1              0       2198             1             0
…            …            …        …          …           …
1455           1              0       1647             0             0
1456           2              0       2073             1             0
1457           2              0       2340             0             0
1458           0            112       1078             1             0
1459           0              0       1256             1             0

      OpenPorchSF  BedroomAbvGr
0              61             3
1               0             3
2              42             3
3              35             3
4              84             4
…             …           …
1455           40             3
1456            0             3
1457           60             4
```

```
1458            0             2
1459           68             3

[1460 rows x 42 columns]
```

[335]:
```python
log_transform(house_test, skew_cols)[predictors]
```

[335]:
```
      2ndFlrSF  BsmtHalfBath  GarageCars  WoodDeckSF  BsmtFinSF1  MiscVal  \
0            0        0.0000      1.0000         140    468.0000        0
1            0        0.0000      1.0000         393    923.0000    12500
2          701        0.0000      2.0000         212    791.0000        0
3          678        0.0000      2.0000         360    602.0000        0
4            0        0.0000      2.0000           0    263.0000        0
…          …             …           …          …           …        …
1454       546        0.0000      0.0000           0      0.0000        0
1455       546        0.0000      1.0000           0    252.0000        0
1456         0        0.0000      2.0000         474   1224.0000        0
1457         0        1.0000      0.0000          80    337.0000      700
1458      1004        0.0000      3.0000         190    758.0000        0

      ExterCond  MoSold  YrSold  FullBath  …  3SsnPorch  HalfBath  BsmtQual  \
0             4       6    2010         1  …          0         0         4
1             4       6    2010         1  …          0         1         4
2             4       3    2010         2  …          0         1         2
3             4       6    2010         2  …          0         1         4
4             4       1    2010         2  …          0         0         2
…           …     …      …       …     …  …        …         …
1454          4       6    2006         1  …          0         1         4
1455          4       4    2006         1  …          0         1         4
1456          4       9    2006         1  …          0         0         4
1457          4       7    2006         1  …          0         0         2
1458          4      11    2006         2  …          0         1         2

      Fireplaces  EnclosedPorch  GrLivArea  BsmtFullBath  LowQualFinSF  \
0              0              0        896        0.0000             0
1              0              0       1329        0.0000             0
2              1              0       1629        0.0000             0
3              1              0       1604        0.0000             0
4              0              0       1280        0.0000             0
…            …             …        …           …           …
1454           0              0       1092        0.0000             0
1455           0              0       1092        0.0000             0
1456           1              0       1224        1.0000             0
1457           0              0        970        0.0000             0
1458           1              0       2000        0.0000             0

      OpenPorchSF  BedroomAbvGr
```

```
0               0            2
1              36            3
2              34            3
3              36            3
4              82            2
…            …            …
1454            0            3
1455           24            3
1456            0            4
1457           32            3
1458           48            3

[1459 rows x 42 columns]
```

[336]:
```
l = house_train.columns.tolist()
log_predictors = [x for x in l if x.startswith('log')]
log_predictors
```

[336]:
```
['logTotalBsmtSF',
 'logGrLivArea',
 'logMSSubClass',
 'logBsmtFinSF1',
 'log1stFlrSF',
 'logWoodDeckSF',
 'logOpenPorchSF',
 'logMasVnrArea',
 'logBsmtHalfBath',
 'logScreenPorch',
 'logEnclosedPorch',
 'logBsmtFinSF2',
 'logKitchenAbvGr',
 'log3SsnPorch',
 'logLowQualFinSF',
 'logLotArea',
 'logPoolArea',
 'logMiscVal']
```

[337]:
```
predictors += log_predictors
predictors = list_subtract(predictors, skew_cols)
predictors
```

[337]:
```
['2ndFlrSF',
 'GarageCars',
 'ExterCond',
 'MoSold',
 'YrSold',
 'FullBath',
```

```
 'YearRemodAdd',
 'YearBuilt',
 'OverallQual',
 'KitchenQual',
 'OverallCond',
 'ExterQual',
 'GarageYrBlt',
 'FireplaceQu',
 'TotRmsAbvGrd',
 'GarageArea',
 'Neighborhood',
 'LotShape',
 'BsmtUnfSF',
 'HalfBath',
 'BsmtQual',
 'Fireplaces',
 'BsmtFullBath',
 'BedroomAbvGr',
 'logTotalBsmtSF',
 'logGrLivArea',
 'logMSSubClass',
 'logBsmtFinSF1',
 'log1stFlrSF',
 'logWoodDeckSF',
 'logOpenPorchSF',
 'logMasVnrArea',
 'logBsmtHalfBath',
 'logScreenPorch',
 'logEnclosedPorch',
 'logBsmtFinSF2',
 'logKitchenAbvGr',
 'log3SsnPorch',
 'logLowQualFinSF',
 'logLotArea',
 'logPoolArea',
 'logMiscVal']
```

5.1 Normalizing predictor columns

[338]:
```
# Import MinMaxScaler from sklearn.preprocessing and ColumnTransformer from␣
↪sklearn.compose
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer

# Normalize the values in the columns of the categorical dataframe
```

```
minmax_transformer = Pipeline(steps=[('minmax', MinMaxScaler())])
standard_transformer = Pipeline(steps=[('standard', StandardScaler())])
preprocessor = ColumnTransformer(
        remainder='passthrough', # passthough features not listed
        transformers=[('ss', standard_transformer, predictors)])

preprocessor.fit(house_train[predictors])

# Create an array containing the normalized values for both the train and the␣
 ↪test
norm_train = preprocessor.transform(house_train[predictors])
norm_test = preprocessor.transform(house_test[predictors])
```

[339]:
```
print(norm_train.shape)
```

```
(1460, 42)
```

[340]:
```
norm_train = np.c_[norm_train, house_train['SalePrice'].values]
print(norm_train.shape)
```

```
(1460, 43)
```

[341]:
```
print(norm_test.shape)
```

```
(1459, 42)
```

[342]:
```
house_train['SalePrice'].values.shape
```

[342]:
```
(1460,)
```

[343]:
```
# Convert the array containing the normalized values to a dataframe
train_numeric_df = pd.DataFrame(data = norm_train, index = house_train.index,
                                columns = predictors + ['SalePrice'])
print(train_numeric_df)

print("\n")

test_numeric_df = pd.DataFrame(data = norm_test, index = house_test.index,␣
 ↪columns = predictors)
print(test_numeric_df)
```

```
      2ndFlrSF  GarageCars  ExterCond  MoSold  YrSold  FullBath  YearRemodAdd  \
0       1.1619      0.3117     0.3642 -1.5991  0.1388    0.7897        0.8787
1      -0.7952      0.3117     0.3642 -0.4891 -0.6144    0.7897       -0.4296
2       1.1894      0.3117     0.3642  0.9909  0.1388    0.7897        0.8302
3       0.9373      1.6503     0.3642 -1.5991 -1.3677   -1.0260       -0.7203
4       1.6179      1.6503     0.3642  2.1009  0.1388    0.7897        0.7333
…          …          …          …       …       …        …            …
1455    0.7952      0.3117     0.3642  0.6209 -0.6144    0.7897        0.7333
```

```
1456   -0.7952      0.3117     0.3642 -1.5991  1.6452    0.7897        0.1519
1457    1.8447     -1.0269    -2.3697 -0.4891  1.6452    0.7897        1.0240
1458   -0.7952     -1.0269     0.3642 -0.8591  1.6452   -1.0260        0.5395
1459   -0.7952     -1.0269     0.3642 -0.1191  0.1388   -1.0260       -0.9626

      YearBuilt  OverallQual  KitchenQual  …  logScreenPorch  \
0        1.0510       0.6515      -0.6658  …         -0.2928
1        0.1567      -0.0718       0.9136  …         -0.2928
2        0.9848       0.6515      -0.6658  …         -0.2928
3       -1.8636       0.6515      -0.6658  …         -0.2928
4        0.9516       1.3748      -0.6658  …         -0.2928
…           …            …            …  …             …
1455     0.9185      -0.0718       0.9136  …         -0.2928
1456     0.2230      -0.0718       0.9136  …         -0.2928
1457    -1.0025       0.6515      -0.6658  …         -0.2928
1458    -0.7044      -0.7952      -0.6658  …         -0.2928
1459    -0.2076      -0.7952       0.9136  …         -0.2928

      logEnclosedPorch  logBsmtFinSF2  logKitchenAbvGr  log3SsnPorch  \
0              -0.4042        -0.3553          -0.2076       -0.1285
1              -0.4042        -0.3553          -0.2076       -0.1285
2              -0.4042        -0.3553          -0.2076       -0.1285
3               2.8444        -0.3553          -0.2076       -0.1285
4              -0.4042        -0.3553          -0.2076       -0.1285
…                  …            …              …            …
1455           -0.4042        -0.3553          -0.2076       -0.1285
1456           -0.4042         2.4097          -0.2076       -0.1285
1457           -0.4042        -0.3553          -0.2076       -0.1285
1458            2.3335         3.4059          -0.2076       -0.1285
1459           -0.4042         2.7206          -0.2076       -0.1285

      logLowQualFinSF  logLotArea  logPoolArea  logMiscVal   SalePrice
0             -0.1336     -0.1333      -0.0694     -0.1905 208500.0000
1             -0.1336      0.1134      -0.0694     -0.1905 181500.0000
2             -0.1336      0.4200      -0.0694     -0.1905 223500.0000
3             -0.1336      0.1033      -0.0694     -0.1905 140000.0000
4             -0.1336      0.8784      -0.0694     -0.1905 250000.0000
…                 …          …          …          …          …
1455          -0.1336     -0.2592      -0.0694     -0.1905 175000.0000
1456          -0.1336      0.7254      -0.0694     -0.1905 210000.0000
1457          -0.1336     -0.0024      -0.0694      6.1936 266500.0000
1458          -0.1336      0.1368      -0.0694     -0.1905 142125.0000
1459          -0.1336      0.1801      -0.0694     -0.1905 147500.0000

[1460 rows x 43 columns]


      2ndFlrSF  GarageCars  ExterCond  MoSold  YrSold  FullBath  YearRemodAdd  \
```

```
0      -0.7952     -1.0269     0.3642 -0.1191  1.6452   -1.0260       -1.1564
1      -0.7952     -1.0269     0.3642 -0.1191  1.6452   -1.0260       -1.3017
2       0.8112      0.3117     0.3642 -1.2291  1.6452    0.7897        0.6364
3       0.7585      0.3117     0.3642 -0.1191  1.6452    0.7897        0.6364
4      -0.7952      0.3117     0.3642 -1.9691  1.6452    0.7897        0.3457
…         …          …        …      …      …       …           …
1454    0.4560     -2.3654     0.3642 -0.1191 -1.3677   -1.0260       -0.7203
1455    0.4560     -1.0269     0.3642 -0.8591 -1.3677   -1.0260       -0.7203
1456   -0.7952      0.3117     0.3642  0.9909 -1.3677   -1.0260        0.5395
1457   -0.7952     -2.3654     0.3642  0.2509 -1.3677   -1.0260        0.3457
1458    1.5056      1.6503     0.3642  1.7309 -1.3677    0.7897        0.4426

      YearBuilt  OverallQual  KitchenQual  …  logBsmtHalfBath  \
0       -0.3401      -0.7952       0.9136  …          -0.2429
1       -0.4394      -0.0718      -0.6658  …          -0.2429
2        0.8523      -0.7952       0.9136  …          -0.2429
3        0.8854      -0.0718      -0.6658  …          -0.2429
4        0.6867       1.3748      -0.6658  …          -0.2429
…          …            …          …   …              …
1454    -0.0420      -1.5185       0.9136  …          -0.2429
1455    -0.0420      -1.5185       0.9136  …          -0.2429
1456    -0.3732      -0.7952       0.9136  …          -0.2429
1457     0.6867      -0.7952       0.9136  …           4.0215
1458     0.7198       0.6515       0.9136  …          -0.2429

      logScreenPorch  logEnclosedPorch  logBsmtFinSF2  logKitchenAbvGr  \
0             3.1262           -0.4042         2.3429          -0.2076
1            -0.2928           -0.4042        -0.3553          -0.2076
2            -0.2928           -0.4042        -0.3553          -0.2076
3            -0.2928           -0.4042        -0.3553          -0.2076
4             3.2552           -0.4042        -0.3553          -0.2076
…              …                …              …              …
1454         -0.2928           -0.4042        -0.3553          -0.2076
1455         -0.2928           -0.4042        -0.3553          -0.2076
1456         -0.2928           -0.4042        -0.3553          -0.2076
1457         -0.2928           -0.4042        -0.3553          -0.2076
1458         -0.2928           -0.4042        -0.3553          -0.2076

      log3SsnPorch  logLowQualFinSF  logLotArea  logPoolArea  logMiscVal
0          -0.1285          -0.1336      0.4829      -0.0694     -0.1905
1          -0.1285          -0.1336      0.8794      -0.0694      7.5066
2          -0.1285          -0.1336      0.8192      -0.0694     -0.1905
3          -0.1285          -0.1336      0.1881      -0.0694     -0.1905
4          -0.1285          -0.1336     -1.1458      -0.0694     -0.1905
…            …               …          …           …           …
1454       -0.1285          -0.1336     -2.9816      -0.0694     -0.1905
1455       -0.1285          -0.1336     -3.0240      -0.0694     -0.1905
1456       -0.1285          -0.1336      1.5325      -0.0694     -0.1905
```

```
1457       -0.1285           -0.1336      0.2758      -0.0694      5.1558
1458       -0.1285           -0.1336      0.1188      -0.0694     -0.1905

[1459 rows x 42 columns]
```

6 Add Bias

[344]:
```python
train_numeric_df.insert(0, "bias", 1)
print(train_numeric_df)
print(train_numeric_df.dtypes)
```

```
      bias  2ndFlrSF  GarageCars  ExterCond  MoSold  YrSold  FullBath  \
0        1    1.1619      0.3117     0.3642 -1.5991  0.1388    0.7897
1        1   -0.7952      0.3117     0.3642 -0.4891 -0.6144    0.7897
2        1    1.1894      0.3117     0.3642  0.9909  0.1388    0.7897
3        1    0.9373      1.6503     0.3642 -1.5991 -1.3677   -1.0260
4        1    1.6179      1.6503     0.3642  2.1009  0.1388    0.7897
…     …        …          …          …        …       …        …
1455     1    0.7952      0.3117     0.3642  0.6209 -0.6144    0.7897
1456     1   -0.7952      0.3117     0.3642 -1.5991  1.6452    0.7897
1457     1    1.8447     -1.0269    -2.3697 -0.4891  1.6452    0.7897
1458     1   -0.7952     -1.0269     0.3642 -0.8591  1.6452   -1.0260
1459     1   -0.7952     -1.0269     0.3642 -0.1191  0.1388   -1.0260

      YearRemodAdd  YearBuilt  OverallQual  …  logScreenPorch  \
0           0.8787     1.0510       0.6515  …         -0.2928
1          -0.4296     0.1567      -0.0718  …         -0.2928
2           0.8302     0.9848       0.6515  …         -0.2928
3          -0.7203    -1.8636       0.6515  …         -0.2928
4           0.7333     0.9516       1.3748  …         -0.2928
…              …          …          …   …             …
1455        0.7333     0.9185      -0.0718  …         -0.2928
1456        0.1519     0.2230      -0.0718  …         -0.2928
1457        1.0240    -1.0025       0.6515  …         -0.2928
1458        0.5395    -0.7044      -0.7952  …         -0.2928
1459       -0.9626    -0.2076      -0.7952  …         -0.2928

      logEnclosedPorch  logBsmtFinSF2  logKitchenAbvGr  log3SsnPorch  \
0              -0.4042        -0.3553          -0.2076       -0.1285
1              -0.4042        -0.3553          -0.2076       -0.1285
2              -0.4042        -0.3553          -0.2076       -0.1285
3               2.8444        -0.3553          -0.2076       -0.1285
4              -0.4042        -0.3553          -0.2076       -0.1285
…                  …             …                …              …
1455           -0.4042        -0.3553          -0.2076       -0.1285
1456           -0.4042         2.4097          -0.2076       -0.1285
1457           -0.4042        -0.3553          -0.2076       -0.1285
```

```
1458           2.3335         3.4059         -0.2076       -0.1285
1459          -0.4042         2.7206         -0.2076       -0.1285

      logLowQualFinSF  logLotArea  logPoolArea  logMiscVal   SalePrice
0             -0.1336     -0.1333      -0.0694     -0.1905 208500.0000
1             -0.1336      0.1134      -0.0694     -0.1905 181500.0000
2             -0.1336      0.4200      -0.0694     -0.1905 223500.0000
3             -0.1336      0.1033      -0.0694     -0.1905 140000.0000
4             -0.1336      0.8784      -0.0694     -0.1905 250000.0000
…                  …           …            …           …           …
1455          -0.1336     -0.2592      -0.0694     -0.1905 175000.0000
1456          -0.1336      0.7254      -0.0694     -0.1905 210000.0000
1457          -0.1336     -0.0024      -0.0694      6.1936 266500.0000
1458          -0.1336      0.1368      -0.0694     -0.1905 142125.0000
1459          -0.1336      0.1801      -0.0694     -0.1905 147500.0000

[1460 rows x 44 columns]
bias                  int64
2ndFlrSF            float64
GarageCars          float64
ExterCond           float64
MoSold              float64
YrSold              float64
FullBath            float64
YearRemodAdd        float64
YearBuilt           float64
OverallQual         float64
KitchenQual         float64
OverallCond         float64
ExterQual           float64
GarageYrBlt         float64
FireplaceQu         float64
TotRmsAbvGrd        float64
GarageArea          float64
Neighborhood        float64
LotShape            float64
BsmtUnfSF           float64
HalfBath            float64
BsmtQual            float64
Fireplaces          float64
BsmtFullBath        float64
BedroomAbvGr        float64
logTotalBsmtSF      float64
logGrLivArea        float64
logMSSubClass       float64
logBsmtFinSF1       float64
log1stFlrSF         float64
logWoodDeckSF       float64
```

```
logOpenPorchSF      float64
logMasVnrArea       float64
logBsmtHalfBath     float64
logScreenPorch      float64
logEnclosedPorch    float64
logBsmtFinSF2       float64
logKitchenAbvGr     float64
log3SsnPorch        float64
logLowQualFinSF     float64
logLotArea          float64
logPoolArea         float64
logMiscVal          float64
SalePrice           float64
dtype: object
```

[345]:

```
test_numeric_df.insert(0, "bias", 1)
print(test_numeric_df)
print(test_numeric_df.dtypes)
```

```
      bias  2ndFlrSF  GarageCars  ExterCond  MoSold  YrSold  FullBath  \
0        1   -0.7952     -1.0269     0.3642 -0.1191  1.6452   -1.0260
1        1   -0.7952     -1.0269     0.3642 -0.1191  1.6452   -1.0260
2        1    0.8112      0.3117     0.3642 -1.2291  1.6452    0.7897
3        1    0.7585      0.3117     0.3642 -0.1191  1.6452    0.7897
4        1   -0.7952      0.3117     0.3642 -1.9691  1.6452    0.7897
…     …        …         …          …       …       …       …
1454     1    0.4560     -2.3654     0.3642 -0.1191 -1.3677   -1.0260
1455     1    0.4560     -1.0269     0.3642 -0.8591 -1.3677   -1.0260
1456     1   -0.7952      0.3117     0.3642  0.9909 -1.3677   -1.0260
1457     1   -0.7952     -2.3654     0.3642  0.2509 -1.3677   -1.0260
1458     1    1.5056      1.6503     0.3642  1.7309 -1.3677    0.7897

      YearRemodAdd  YearBuilt  OverallQual  …  logBsmtHalfBath  \
0          -1.1564    -0.3401      -0.7952  …          -0.2429
1          -1.3017    -0.4394      -0.0718  …          -0.2429
2           0.6364     0.8523      -0.7952  …          -0.2429
3           0.6364     0.8854      -0.0718  …          -0.2429
4           0.3457     0.6867       1.3748  …          -0.2429
…             …          …          …   …            …
1454       -0.7203    -0.0420      -1.5185  …          -0.2429
1455       -0.7203    -0.0420      -1.5185  …          -0.2429
1456        0.5395    -0.3732      -0.7952  …          -0.2429
1457        0.3457     0.6867      -0.7952  …           4.0215
1458        0.4426     0.7198       0.6515  …          -0.2429

      logScreenPorch  logEnclosedPorch  logBsmtFinSF2  logKitchenAbvGr  \
0             3.1262           -0.4042         2.3429          -0.2076
1            -0.2928           -0.4042        -0.3553          -0.2076
```

```
2            -0.2928            -0.4042       -0.3553           -0.2076
3            -0.2928            -0.4042       -0.3553           -0.2076
4             3.2552            -0.4042       -0.3553           -0.2076
…               …               …          …              …
1454         -0.2928            -0.4042       -0.3553           -0.2076
1455         -0.2928            -0.4042       -0.3553           -0.2076
1456         -0.2928            -0.4042       -0.3553           -0.2076
1457         -0.2928            -0.4042       -0.3553           -0.2076
1458         -0.2928            -0.4042       -0.3553           -0.2076

      log3SsnPorch  logLowQualFinSF  logLotArea  logPoolArea  logMiscVal
0          -0.1285          -0.1336      0.4829      -0.0694     -0.1905
1          -0.1285          -0.1336      0.8794      -0.0694      7.5066
2          -0.1285          -0.1336      0.8192      -0.0694     -0.1905
3          -0.1285          -0.1336      0.1881      -0.0694     -0.1905
4          -0.1285          -0.1336     -1.1458      -0.0694     -0.1905
…            …               …         …           …          …
1454       -0.1285          -0.1336     -2.9816      -0.0694     -0.1905
1455       -0.1285          -0.1336     -3.0240      -0.0694     -0.1905
1456       -0.1285          -0.1336      1.5325      -0.0694     -0.1905
1457       -0.1285          -0.1336      0.2758      -0.0694      5.1558
1458       -0.1285          -0.1336      0.1188      -0.0694     -0.1905

[1459 rows x 43 columns]
bias                  int64
2ndFlrSF            float64
GarageCars          float64
ExterCond           float64
MoSold              float64
YrSold              float64
FullBath            float64
YearRemodAdd        float64
YearBuilt           float64
OverallQual         float64
KitchenQual         float64
OverallCond         float64
ExterQual           float64
GarageYrBlt         float64
FireplaceQu         float64
TotRmsAbvGrd        float64
GarageArea          float64
Neighborhood        float64
LotShape            float64
BsmtUnfSF           float64
HalfBath            float64
BsmtQual            float64
Fireplaces          float64
BsmtFullBath        float64
```

```
BedroomAbvGr        float64
logTotalBsmtSF      float64
logGrLivArea        float64
logMSSubClass       float64
logBsmtFinSF1       float64
log1stFlrSF         float64
logWoodDeckSF       float64
logOpenPorchSF      float64
logMasVnrArea       float64
logBsmtHalfBath     float64
logScreenPorch      float64
logEnclosedPorch    float64
logBsmtFinSF2       float64
logKitchenAbvGr     float64
log3SsnPorch        float64
logLowQualFinSF     float64
logLotArea          float64
logPoolArea         float64
logMiscVal          float64
dtype: object
```

7 Build, Predict and Evaluate the Regression Models

7.1 Linear Regression

[346]:
```
from sklearn.model_selection import cross_val_score
# function to get cross validation scores
def get_cv_scores(model):
    scores = cross_val_score(model,
                             X_train,
                             y_train,
                             cv=5,
                             scoring='r2')

    print('CV Mean: ', np.mean(scores))
    print('STD: ', np.std(scores))
```

[347]:
```
from sklearn.linear_model import LinearRegression
# Train model
lr = LinearRegression().fit(X_train, y_train)
# get cross val scores
get_cv_scores(lr)
```

```
CV Mean:  0.7891931216501084
STD:  0.06805628998571318
```

[348]:
```python
# Function to evaluate different models for different values of alpha.
def run_model(model_class, alphas, **model_kargs):
    for alpha in alphas:
        model = model_class(alpha = alpha, **model_kargs) if alpha > 0 else␣
↪LinearRegression()
        model.fit(X_train, y_train)
        y_train_pred = model.predict(X_train)
        y_val_pred = model.predict(X_val)

        print(f'alpha = {alpha}')
        print("Evaluation metrics, MAE and R-squared, for the training data:")
        print(mean_squared_error(y_train, y_train_pred))
        print(r2_score(y_train, y_train_pred))

        print("Evaluation metrics, MAE and R-squared, for the validation set:")
        print(mean_squared_error(y_val, y_val_pred))
        print(r2_score(y_val, y_val_pred))
```

[349]:
```python
# instantiate the LinearRegression() algorithm
lin_reg = LinearRegression()

# fit the model on the training set.
lin_reg.fit(X_train, y_train)
```

[349]:
```
LinearRegression()
```

[350]:
```python
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

[351]:
```python
# predict on the training set.
pred_train_lin_reg = lin_reg.predict(X_train)

print("Evaluation metrics, RMSE and R-squared, for the training set:")
print(np.sqrt(mean_squared_error(y_train, pred_train_lin_reg)))
print(r2_score(y_train, pred_train_lin_reg))
```

```
Evaluation metrics, RMSE and R-squared, for the training set:
33272.53914559085
0.8307726199057027
```

[352]:
```python
# predict on the training set.
pred_val_lin_reg = lin_reg.predict(X_val)

print("Evaluation metrics, RMSE and R-squared, for the training set:")
print(np.sqrt(mean_squared_error(y_val, pred_val_lin_reg)))
print(r2_score(y_val, pred_val_lin_reg))
```

```
Evaluation metrics, RMSE and R-squared, for the training set:
```

28897.083801712197
0.8438696135066126

7.2 Ridge Regression

[353]:
```
from sklearn.linear_model import Ridge
# Train model with default alpha=1
ridge = Ridge(alpha=1).fit(X_train, y_train)
# get cross val scores
get_cv_scores(ridge)
```

```
CV Mean:  0.7894603479220247
STD:  0.06785643107007511
```

[354]:
```
# find optimal alpha with grid search
alpha = [0.001, 0.01, 0.1, 1, 10, 100, 1000]
param_grid = dict(alpha=alpha)
grid = GridSearchCV(estimator=ridge, param_grid=param_grid, scoring='r2',␣
 ↪verbose=1, n_jobs=-1)
grid_result = grid.fit(X_train, y_train)
print('Best Score: ', grid_result.best_score_)
print('Best Params: ', grid_result.best_params_)
```

```
Fitting 5 folds for each of 7 candidates, totalling 35 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.

Best Score:  0.7952094507565439
Best Params:  {'alpha': 100}

[Parallel(n_jobs=-1)]: Done  35 out of  35 | elapsed:    1.5s finished
```

[355]:
```
# instantiate the Ridge Regression model with an alpha value of 0.01
model_ridge = Ridge(alpha = 100, solver="cholesky", random_state=42)

# fit the model to the training data.
model_ridge.fit(X_train, y_train)
```

[355]:
```
Ridge(alpha=100, random_state=42, solver='cholesky')
```

[356]:
```
# Predict on the training data
pred_train_ridge = model_ridge.predict(X_train)

print("Evaluation metrics, RMSE and R-squared, for the training set:")
print(np.sqrt(mean_squared_error(y_train, pred_train_ridge)))
print(r2_score(y_train, pred_train_ridge))
```

```
Evaluation metrics, RMSE and R-squared, for the training set:
33787.79285962448
0.8254907758862396
```

[357]:
```
# Predict on the validation data
pred_test_ridge = model_ridge.predict(X_val)

print("Evaluation metrics, RMSE and R-squared, for the validation set:")
print(np.sqrt(mean_squared_error(y_val, pred_test_ridge)))
print(r2_score(y_val, pred_test_ridge))
```

```
Evaluation metrics, RMSE and R-squared, for the validation set:
29071.051785656975
0.8419840634640094
```

7.3 Lasso Regression

[358]:
```
from sklearn.linear_model import Lasso
# Train model with default alpha=1
lasso = Lasso(alpha=1).fit(X_train, y_train)
# get cross val scores
get_cv_scores(lasso)
```

```
CV Mean:  0.7892076222765457
STD:  0.06804731767133065
```

[359]:
```
# find optimal alpha with grid search
alpha = [0.001, 0.01, 0.1, 1, 10, 100, 1000]
param_grid = dict(alpha=alpha)
grid = GridSearchCV(estimator=lasso, param_grid=param_grid, scoring='r2',␣
↪verbose=1, n_jobs=-1)
grid_result = grid.fit(X_train, y_train)
print('Best Score: ', grid_result.best_score_)
print('Best Params: ', grid_result.best_params_)
```

```
Fitting 5 folds for each of 7 candidates, totalling 35 fits
Best Score:  0.7959423514019661
Best Params:  {'alpha': 1000}
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  20 out of  35 | elapsed:    0.1s remaining:    0.1s
[Parallel(n_jobs=-1)]: Done  35 out of  35 | elapsed:    0.1s finished
```

[360]:
```
model_lasso = Lasso(alpha=1000, max_iter = 5000)
%time model_lasso.fit(X_train, y_train)
```

```
CPU times: user 12.1 ms, sys: 6.84 ms, total: 18.9 ms
Wall time: 10.4 ms
```

[360]:
```
Lasso(alpha=1000, max_iter=5000)
```

[361]:
```
# Predict on the training data
pred_train_lasso = model_lasso.predict(X_train)
```

```
print("Evaluation metrics, RMSE and R-squared, for the training set:")
print(np.sqrt(mean_squared_error(y_train, pred_train_lasso)))
print(r2_score(y_train, pred_train_lasso))
```

```
Evaluation metrics, RMSE and R-squared, for the training set:
33836.84830233911
0.8249836790340636
```

[362]:
```
# Predict on the validation data
pred_test_lasso= model_lasso.predict(X_val)

print("Evaluation metrics, RMSE and R-squared, for the validation set:")
print(np.sqrt(mean_squared_error(y_val, pred_test_lasso)))
print(r2_score(y_val, pred_test_lasso))
```

```
Evaluation metrics, RMSE and R-squared, for the validation set:
29130.328968474085
0.8413390032694951
```

7.4 ElasticNet Regression

[363]:
```
from sklearn.linear_model import ElasticNet
# Train model with default alpha=1 and l1_ratio=0.5
elastic_net = ElasticNet(alpha=1, l1_ratio=0.5).fit(X_train, y_train)
# get cross val scores
get_cv_scores(elastic_net)
```

```
CV Mean:  0.7974568966183408
STD:  0.05868180723477032
```

[364]:
```
# find optimal alpha with grid search
alpha = [0.001, 0.01, 0.1, 1, 10, 100, 1000]
l1_ratio = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]
param_grid = dict(alpha=alpha, l1_ratio=l1_ratio)
grid = GridSearchCV(estimator=elastic_net, param_grid=param_grid, scoring='r2',␣
 ↪verbose=1, n_jobs=-1)
grid_result = grid.fit(X_train, y_train)
print('Best Score: ', grid_result.best_score_)
print('Best Params: ', grid_result.best_params_)
```

```
Fitting 5 folds for each of 77 candidates, totalling 385 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  56 tasks      | elapsed:    0.3s

Best Score:  0.798376790920388
Best Params:  {'alpha': 1, 'l1_ratio': 0.7}

[Parallel(n_jobs=-1)]: Done 385 out of 385 | elapsed:    0.7s finished
```

[365]:
```
model_elastic_net = ElasticNet(alpha = 1, max_iter = 2500)
model_elastic_net.fit(X_train, y_train)
```

[365]:
```
ElasticNet(alpha=1, max_iter=2500)
```

[366]:
```
# Predict on the training data
pred_train_elastic_net = model_elastic_net.predict(X_train)

print("Evaluation metrics, RMSE and R-squared, for the training set:")
print(np.sqrt(mean_squared_error(y_train, pred_train_elastic_net)))
print(r2_score(y_train, pred_train_elastic_net))
```

```
Evaluation metrics, RMSE and R-squared, for the training set:
35068.50711848182
0.8120106276395401
```

[367]:
```
# Predict on the validation data
pred_test_elastic_net = model_elastic_net.predict(X_val)

print("Evaluation metrics, RMSE and R-squared, for the validation set:")
print(np.sqrt(mean_squared_error(y_val, pred_test_elastic_net)))
print(r2_score(y_val, pred_test_elastic_net))
```

```
Evaluation metrics, RMSE and R-squared, for the validation set:
30043.53083419703
0.8312354045655611
```

7.5 XGBoost Regressor

[368]:
```
predictors_with_bias = ['bias'] + predictors
predictors_with_bias
```

[368]:
```
['bias',
 '2ndFlrSF',
 'GarageCars',
 'ExterCond',
 'MoSold',
 'YrSold',
 'FullBath',
 'YearRemodAdd',
 'YearBuilt',
 'OverallQual',
 'KitchenQual',
 'OverallCond',
 'ExterQual',
 'GarageYrBlt',
 'FireplaceQu',
 'TotRmsAbvGrd',
```

```
 'GarageArea',
 'Neighborhood',
 'LotShape',
 'BsmtUnfSF',
 'HalfBath',
 'BsmtQual',
 'Fireplaces',
 'BsmtFullBath',
 'BedroomAbvGr',
 'logTotalBsmtSF',
 'logGrLivArea',
 'logMSSubClass',
 'logBsmtFinSF1',
 'log1stFlrSF',
 'logWoodDeckSF',
 'logOpenPorchSF',
 'logMasVnrArea',
 'logBsmtHalfBath',
 'logScreenPorch',
 'logEnclosedPorch',
 'logBsmtFinSF2',
 'logKitchenAbvGr',
 'log3SsnPorch',
 'logLowQualFinSF',
 'logLotArea',
 'logPoolArea',
 'logMiscVal']
```

[369]:

```
X = train_numeric_df[predictors_with_bias].values

y = train_numeric_df[target].values

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size = 0.20)
print(X_train.shape)
print(X_val.shape)
```

```
(1168, 43)
(292, 43)
```

[370]:

```
import sklearn
sorted(sklearn.metrics.SCORERS.keys())
```

[370]:

```
['accuracy',
 'adjusted_mutual_info_score',
 'adjusted_rand_score',
 'average_precision',
 'balanced_accuracy',
 'completeness_score',
```

```
'explained_variance',
'f1',
'f1_macro',
'f1_micro',
'f1_samples',
'f1_weighted',
'fowlkes_mallows_score',
'homogeneity_score',
'jaccard',
'jaccard_macro',
'jaccard_micro',
'jaccard_samples',
'jaccard_weighted',
'max_error',
'mutual_info_score',
'neg_brier_score',
'neg_log_loss',
'neg_mean_absolute_error',
'neg_mean_gamma_deviance',
'neg_mean_poisson_deviance',
'neg_mean_squared_error',
'neg_mean_squared_log_error',
'neg_median_absolute_error',
'neg_root_mean_squared_error',
'normalized_mutual_info_score',
'precision',
'precision_macro',
'precision_micro',
'precision_samples',
'precision_weighted',
'r2',
'recall',
'recall_macro',
'recall_micro',
'recall_samples',
'recall_weighted',
'roc_auc',
'roc_auc_ovo',
'roc_auc_ovo_weighted',
'roc_auc_ovr',
'roc_auc_ovr_weighted',
'v_measure_score']
```

[371]:
```
def algorithm_pipeline(X_train_data, X_test_data, y_train_data, y_test_data,
                       model, param_grid, cv=10,␣
 ↪scoring_fit='neg_mean_squared_error'):
    gs = GridSearchCV(
```

```
        estimator=model,
        param_grid=param_grid,
        cv=cv,
        n_jobs=-1,
        scoring=scoring_fit,
        verbose=2
    )

    fitted_model = gs.fit(X_train_data, y_train_data.ravel())
    pred = fitted_model.predict(X_test_data)

    return fitted_model, pred
```

[372]:
```
import xgboost as xgb

xgb_model = xgb.XGBRegressor()

param_grid = {
    'eta': [0.05],
    'n_estimators': [500, 1000],
    'colsample_bytree': [0.7, 0.8],
    'max_depth': [6, 10],
    'reg_alpha': [0, 1],
    'reg_lambda': [1, 2],
    'subsample': [0.5, 0.9]
}

%time xgb_model, pred = algorithm_pipeline(X_train, X_val, y_train, y_val,␣
↪xgb_model, param_grid, cv=5)
```

```
Fitting 5 folds for each of 64 candidates, totalling 320 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    8.6s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done 320 out of 320 | elapsed:  3.0min finished

CPU times: user 3.53 s, sys: 1.49 s, total: 5.02 s
Wall time: 3min 5s
```

[373]:
```
# Root Mean Squared Error
print(np.sqrt(-xgb_model.best_score_))
print(xgb_model.best_params_)
```

```
28450.02033010322
{'colsample_bytree': 0.7, 'eta': 0.05, 'max_depth': 6, 'n_estimators': 1000,
'reg_alpha': 0, 'reg_lambda': 2, 'subsample': 0.5}
```

[377]:
```
import xgboost
from xgboost import XGBRegressor

# 'colsample_bytree': 0.7, 'eta': 0.05, 'max_depth': 6, 'n_estimators': 500,␣
↪'reg_alpha': 0, 'reg_lambda': 2,
# 'subsample': 0.5

xgb_reg = XGBRegressor(n_estimators=500, learning_rate=0.05, subsample=0.5,␣
↪colsample_bytree=0.7,
                       max_depth=6, reg_alpha=0, reg_lambda=2)
%time xgb_reg.fit(X_train, y_train.ravel())
```

```
CPU times: user 1.29 s, sys: 13.7 ms, total: 1.3 s
Wall time: 1.33 s
```

[377]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=0.7, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.05, max_delta_step=0, max_depth=6,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=500, n_jobs=1, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=2, scale_pos_weight=1, subsample=0.5,
             tree_method='exact', validate_parameters=1, verbosity=None)
```

[378]:
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

[379]:
```
%time pred_train_xgb = xgb_reg.predict(X_train)

print("Evaluation metrics, RMSE and R-squared, for the training set:")
print(np.sqrt(mean_squared_error(y_train, pred_train_xgb)))
print(r2_score(y_train, pred_train_xgb))
```

```
CPU times: user 23 ms, sys: 1.39 ms, total: 24.4 ms
Wall time: 23.3 ms
Evaluation metrics, RMSE and R-squared, for the training set:
3244.4419013760307
0.9984082290927453
```

[380]:
```
%time pred_test_xgb = xgb_reg.predict(X_val)

print("Evaluation metrics, RMSE and R-squared, for the test set:")
print(np.sqrt(mean_squared_error(y_val, pred_test_xgb)))
print(r2_score(y_val, pred_test_xgb))
```

```
CPU times: user 11.4 ms, sys: 1.39 ms, total: 12.8 ms
Wall time: 11 ms
Evaluation metrics, RMSE and R-squared, for the test set:
21029.25611735775
```

0.9129364211290131

[381]:
```
X_submission = test_numeric_df[predictors_with_bias].values
X_submission.shape
```

[381]: (1459, 43)

[382]:
```
%time submission_xgb = xgb_reg.predict(X_submission)
```

```
CPU times: user 32.8 ms, sys: 1.38 ms, total: 34.2 ms
Wall time: 32.9 ms
```

[383]:
```
submission_xgb.shape
```

[383]: (1459,)

[384]:
```
house_test["Id"].values.shape
```

[384]: (1459,)

[385]:
```
submission_data = np.c_[house_test["Id"].values, submission_xgb]
submission_df = pd.DataFrame(data = submission_data, columns = ["Id",␣
↪"SalePrice"])
submission_df['Id'] = submission_df['Id'].astype('int64')
submission_df
submission_df.to_csv("/Users/anaswarjayakumar/Downloads/house_xgb.csv", index =␣
↪False)
```

7.6 Random Forests

[386]:
```
from xgboost import XGBRFRegressor
xgb_rf_model = XGBRFRegressor()

param_grid = {
    'n_estimators': [50, 100, 150],
    'colsample_bytree': [0.5, 0.8],
    'max_depth': [6, 10, 20],
    'reg_lambda': [1, 2, 5],
    'subsample': [0.5, 0.8]
}

%time xgb_rf_model, pred = algorithm_pipeline(X_train, X_val, y_train, y_val,␣
↪xgb_rf_model, param_grid, cv=5)
```

```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    0.8s
[Parallel(n_jobs=-1)]: Done 276 tasks      | elapsed:    6.0s
[Parallel(n_jobs=-1)]: Done 525 out of 540 | elapsed:   13.4s remaining:    0.4s
```

```
CPU times: user 914 ms, sys: 138 ms, total: 1.05 s
Wall time: 13.9 s
```

```
[Parallel(n_jobs=-1)]: Done 540 out of 540 | elapsed:   13.7s finished
```

[387]:
```
# Root Mean Squared Error
print(np.sqrt(-xgb_rf_model.best_score_))
print(xgb_rf_model.best_params_)
```

```
38446.166373727625
{'colsample_bytree': 0.5, 'max_depth': 10, 'n_estimators': 100, 'reg_lambda': 1,
'subsample': 0.8}
```

[388]:
```
xgb_rf_reg = XGBRFRegressor(n_estimators=100, subsample=0.8, colsample_bytree=0.
↪5, max_depth=10, reg_lambda=1)
%time xgb_rf_reg.fit(X_train, y_train.ravel())
```

```
CPU times: user 189 ms, sys: 3.48 ms, total: 192 ms
Wall time: 192 ms
```

[388]:
```
XGBRFRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
               colsample_bytree=0.5, gamma=0, gpu_id=-1, importance_type='gain',
               interaction_constraints='', max_delta_step=0, max_depth=10,
               min_child_weight=1, missing=nan, monotone_constraints='()',
               n_estimators=100, n_jobs=1, num_parallel_tree=100,
               objective='reg:squarederror', random_state=0, reg_alpha=0,
               reg_lambda=1, scale_pos_weight=1, tree_method='exact',
               validate_parameters=1, verbosity=None)
```

[389]:
```
%time pred_train_xgb = xgb_rf_reg.predict(X_train)

print("Evaluation metrics, RMSE and R-squared, for the training set:")
print(np.sqrt(mean_squared_error(y_train, pred_train_xgb)))
print(r2_score(y_train, pred_train_xgb))
```

```
CPU times: user 12.9 ms, sys: 1.7 ms, total: 14.6 ms
Wall time: 12.7 ms
Evaluation metrics, RMSE and R-squared, for the training set:
31064.584177126308
0.8540743458208875
```

[390]:
```
%time pred_val_xgb = xgb_rf_reg.predict(X_val)

print("Evaluation metrics, RMSE and R-squared, for the training set:")
print(np.sqrt(mean_squared_error(y_val, pred_val_xgb)))
print(r2_score(y_val, pred_val_xgb))
```

```
CPU times: user 5.68 ms, sys: 1.33 ms, total: 7.01 ms
Wall time: 5.48 ms
Evaluation metrics, RMSE and R-squared, for the training set:
```

```
28426.048571451738
0.8409177655803284
```

[391]:
```
%time submission_rf_xgb = xgb_rf_reg.predict(X_submission)
```

```
CPU times: user 16.8 ms, sys: 2.71 ms, total: 19.6 ms
Wall time: 16.9 ms
```

[394]:
```
submission_rf_xgb.shape
```

[394]:
```
(1459,)
```

[395]:
```
house_test["Id"].values.shape
```

[395]:
```
(1459,)
```

[396]:
```
submission_data = np.c_[house_test["Id"].values, submission_rf_xgb]
submission_df = pd.DataFrame(data = submission_data, columns = ["Id",␣
 ↪"SalePrice"])
submission_df['Id'] = submission_df['Id'].astype('int64')
submission_df
submission_df.to_csv("/Users/anaswarjayakumar/Downloads/house_rf_xgb.csv",␣
 ↪index = False)
```

8 Conclusion

8.1 Data preparation, exploration, visualization

Some of the data preparation techniques that were used in Assignments 2 and 3 such as Label Encoding were carried over to Assignment 4. Label Encoding was performed for both the training and test data. In addition, arrays for the features and the response variable were created as well. For the predictor variables, I used a combination of categorical and numerical features as well as the log of the numerical features that were skewed. The SalePrice variable was set aside as the response variable since the goal is to predict the sale price of a given house. I arrived at the predictor variables variables based on the visualizations that I did such as bar plots, scatter plots, and box plots. In addition, I also looked at which variables were highly correlated with the sale price and this was especially useful in deciding which variables should be set aside as the predictor variables.

The plots and graphs allowed me to gain further insight into the factors that would affect the sale price of a home. From the box plots, I noticed that the overall quality of the house certainly affects the sale price and that any features that are related to the house quality will definitely affect the sale price as well. In addition, I also noticed that the shape of the lot seems to affect the sale price as well, probably because a house that is custom made or built on a plot that is irregular in shape will most likely increase the sale price. Lastly, I noticed that features such as GrLivArea, YrBuilt, and Lot Area seem to have a linear relationship with the sale price. From the bar plots, it is easy to see how certain attributes affect the sale price. Lastly the correlation matrix was also useful in depicting the correlation between the different features and this is quite useful especially when selecting which features to use in the regression methods

8.2 Review research design and modeling methods

In this assignment, six methods were used, Linear Regression, Ridge Regression, Lasso Regression, ElasticNet Regression, XGBoost for Regression (XGBRegressor), and XGBoost for Random Forests. The Linear, Ridge, Lasso and ElasticNet Regression all seemed to perform well when tested on the validation set. However, the RMSE and the R-squared were the highest for the ElasticNet regression. The XGBRegressor generated the highest R-squared score. In addition, the XGBRegressor generated the best score when submitted to Kaggle, however, the XGBRegressor performed poorly when both the log and original features were included. This was because of overfitting. While both the training and validation errors improved, the test error as measured by Kaggle decreased. Random Forests didnt perform as well as I expected compared to XGBRegressor and my score on Kaggle worsened. For the Ridge, Lasso, and ElasticNet regressions, I performed GridSearch to find the optimal alpha via cross validation. For the XGBRegressor and Random Forests, I again used GridSearch, however I used other parameters besides alpha such as colsample_bytree, eta, max_depth, n_estimators, reg_alpha, reg_lambda, and subsample.

8.3 Implementation and programming

Regression methods that were implemented in Assignment 2 such as Linear, Ridge, Lasso, ElasticNet and XGBRegression were implemented in this assignment as well. In addition, new methods such as Random Forests were implemented in this assignment as well. For the Linear Regression, I first obtained the cross-validation scores and then evaluated different models for different values of alpha. Finally, I obtained the RMSE and R-Squared values for both the training and validation sets. For the Ridge, Lasso, and ElasticNet regressions, I again obtained the cross validation score, however I performed the GridSearch to obtain the optimal alpha via cross validation. Once the optimal alpha was obtained, I then obtained the RMSE and R-Squared for both the test and validation sets. In addition, the Linear, Ridge, Lasso, and ElasticNet regressions were all performed on the training and validation sets that originated from the original training data i.e. the training data that was provided by Kaggle. For the XGBoost Regression, I first added the bias column that was previously created to the list of predictors that was created before. Using the newly created list of predictors, I then created new training and validation sets. I then used a function to run the GridSearch via cross-validation and created a dictionary of the parameters that should be included when performing the regression. Finally, I generated the RMSE and R-Squared for both the training and validation sets. Performing Random Forests was similar to the XGBoost Regression as XGBoost is used to train decsion trees such as Random Forests

8.4 Exposition, problem description and management recommendations

The results in Kaggle could definitely be improved which means there is still more work to be done in terms of feature generation. Some of the Kagglers were able to extract additional information out of certain columns. Out of six methods that were used, I would recommend performing the XGBoost for Regression method. I would also recommend performing a log transformation on the columns that are of interest. The XGBoost method is the one I would recommend because it performs much better compared to other linear regression methods such as the Linear, Ridge, Lasso and ElasticNet methods. From a machine learning standpoint, there are many advantages that XGBoost has over other regression methods such as built in regularization which prevents overfitting, parallel processing which aids in faster execution time and efficiency, the ability to handle missing values and perform cross validation, and tree pruning which significantly improves computational performance.

I believe that the explanatory variables which are the most important in predicting home prices include variables that are related to the overall quality and condition of the house such as OverallQual, ExterQual, KitchenQual, GarageQual, HeatingQC, etc. In addition, other variables that have to do with the important features of a house such as the number of bedrooms, number of kitchens, number of fireplaces, garage size, and garage capacity would definitely be important as well as variables that describe the sale history of a house such as the type and condition of the sale and how long the house was on the market. Lastly, I do believe that the interaction variables i.e. new variables that are formed from existing variables are definitely factors in predicting the sale price of a home. For example, the home age or the difference between when a home is sold and when a home is built, could definitely be used to predict the sale price.

[ ]:

```

```