house-prices-advanced-regression

December 9, 2021

1 Imports / Read Data

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

from pprint import pprint

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Lasso
from sklearn.feature_selection import SelectFromModel
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error,␣
↪explained_variance_score

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.optimizers import Adam

pd.pandas.set_option('display.max_columns', None)
```

[2]:
```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

2 EDA

[3]:
```
numerical_cont_features = [col for col in train.drop(['Id', 'SalePrice'],␣
↪axis=1) if
                       train[col].dtype in ['int64', 'float64'] and train[col].
↪nunique() > 15]
```

```
numerical_cat_features = [col for col in train.drop(['Id', 'SalePrice'],␣
↪axis=1) if
                          train[col].dtype in ['int64', 'float64'] and train[col].
↪nunique() <= 15]
categorical_features = [col for col in train.drop('SalePrice', axis=1) if
                        train[col].dtype in ['object']]
numerical_cols = numerical_cont_features + numerical_cat_features
```

[4]:
```
null_cols_train = [[col, train[col].isnull().sum(), \
                    round(train[col].isnull().sum() / len(train[col]), 2)] for␣
↪col in \
                   train if train[col].isnull().sum() > 0]
pprint(null_cols_train)
print(f'null_cols_train length: {len(null_cols_train)}')
```

```
[['LotFrontage', 259, 0.18],
 ['Alley', 1369, 0.94],
 ['MasVnrType', 8, 0.01],
 ['MasVnrArea', 8, 0.01],
 ['BsmtQual', 37, 0.03],
 ['BsmtCond', 37, 0.03],
 ['BsmtExposure', 38, 0.03],
 ['BsmtFinType1', 37, 0.03],
 ['BsmtFinType2', 38, 0.03],
 ['Electrical', 1, 0.0],
 ['FireplaceQu', 690, 0.47],
 ['GarageType', 81, 0.06],
 ['GarageYrBlt', 81, 0.06],
 ['GarageFinish', 81, 0.06],
 ['GarageQual', 81, 0.06],
 ['GarageCond', 81, 0.06],
 ['PoolQC', 1453, 1.0],
 ['Fence', 1179, 0.81],
 ['MiscFeature', 1406, 0.96]]
null_cols_train length: 19
```

[5]:
```
null_cols_test = [[col, test[col].isnull().sum(), \
                   round(test[col].isnull().sum() / len(test[col]), 2)] for␣
↪col in \
                  test if test[col].isnull().sum() > 0]
pprint(null_cols_test)
print(f'null_cols_test length: {len(null_cols_test)}')
```

```
[['MSZoning', 4, 0.0],
 ['LotFrontage', 227, 0.16],
 ['Alley', 1352, 0.93],
 ['Utilities', 2, 0.0],
```

```
 ['Exterior1st', 1, 0.0],
 ['Exterior2nd', 1, 0.0],
 ['MasVnrType', 16, 0.01],
 ['MasVnrArea', 15, 0.01],
 ['BsmtQual', 44, 0.03],
 ['BsmtCond', 45, 0.03],
 ['BsmtExposure', 44, 0.03],
 ['BsmtFinType1', 42, 0.03],
 ['BsmtFinSF1', 1, 0.0],
 ['BsmtFinType2', 42, 0.03],
 ['BsmtFinSF2', 1, 0.0],
 ['BsmtUnfSF', 1, 0.0],
 ['TotalBsmtSF', 1, 0.0],
 ['BsmtFullBath', 2, 0.0],
 ['BsmtHalfBath', 2, 0.0],
 ['KitchenQual', 1, 0.0],
 ['Functional', 2, 0.0],
 ['FireplaceQu', 730, 0.5],
 ['GarageType', 76, 0.05],
 ['GarageYrBlt', 78, 0.05],
 ['GarageFinish', 78, 0.05],
 ['GarageCars', 1, 0.0],
 ['GarageArea', 1, 0.0],
 ['GarageQual', 78, 0.05],
 ['GarageCond', 78, 0.05],
 ['PoolQC', 1456, 1.0],
 ['Fence', 1169, 0.8],
 ['MiscFeature', 1408, 0.97],
 ['SaleType', 1, 0.0]]
null_cols_test length: 33
```

[6]:
```python
null_cat_cols_train = [[col, train[col].isnull().sum(), \
                        round(train[col].isnull().sum() / len(train[col]), 2)]␣
 ↪for \
                       col in train[categorical_features] if train[col].
 ↪isnull().sum() > 0]
pprint(null_cat_cols_train)
print(f'null_cat_cols_train length: {len(null_cat_cols_train)}')
```

```
[['Alley', 1369, 0.94],
 ['MasVnrType', 8, 0.01],
 ['BsmtQual', 37, 0.03],
 ['BsmtCond', 37, 0.03],
 ['BsmtExposure', 38, 0.03],
 ['BsmtFinType1', 37, 0.03],
 ['BsmtFinType2', 38, 0.03],
 ['Electrical', 1, 0.0],
 ['FireplaceQu', 690, 0.47],
```

```
 ['GarageType', 81, 0.06],
 ['GarageFinish', 81, 0.06],
 ['GarageQual', 81, 0.06],
 ['GarageCond', 81, 0.06],
 ['PoolQC', 1453, 1.0],
 ['Fence', 1179, 0.81],
 ['MiscFeature', 1406, 0.96]]
null_cat_cols_train length: 16
```

[7]:
```
null_cat_cols_test = [[col, test[col].isnull().sum(), \
                        round(test[col].isnull().sum() / len(test[col]), 2)]␣
 ↪for \
                       col in test[categorical_features] if test[col].isnull().
 ↪sum() > 0]
pprint(null_cat_cols_test)
print(f'null_cat_cols_test length: {len(null_cat_cols_test)}')
```

```
[['MSZoning', 4, 0.0],
 ['Alley', 1352, 0.93],
 ['Utilities', 2, 0.0],
 ['Exterior1st', 1, 0.0],
 ['Exterior2nd', 1, 0.0],
 ['MasVnrType', 16, 0.01],
 ['BsmtQual', 44, 0.03],
 ['BsmtCond', 45, 0.03],
 ['BsmtExposure', 44, 0.03],
 ['BsmtFinType1', 42, 0.03],
 ['BsmtFinType2', 42, 0.03],
 ['KitchenQual', 1, 0.0],
 ['Functional', 2, 0.0],
 ['FireplaceQu', 730, 0.5],
 ['GarageType', 76, 0.05],
 ['GarageFinish', 78, 0.05],
 ['GarageQual', 78, 0.05],
 ['GarageCond', 78, 0.05],
 ['PoolQC', 1456, 1.0],
 ['Fence', 1169, 0.8],
 ['MiscFeature', 1408, 0.97],
 ['SaleType', 1, 0.0]]
null_cat_cols_test length: 22
```

[8]:
```
null_num_cols_train = [[col, train[col].isnull().sum(), \
                        round(train[col].isnull().sum() / len(train[col]), 2)]␣
 ↪for \
                       col in train[numerical_cols] if train[col].isnull().
 ↪sum() > 0]
pprint(null_num_cols_train)
```

```
print(f'null_num_cols_train length: {len(null_num_cols_train)}')
```

```
[['LotFrontage', 259, 0.18], ['MasVnrArea', 8, 0.01], ['GarageYrBlt', 81, 0.06]]
null_num_cols_train length: 3
```

[9]:

```
null_num_cols_test = [[col, test[col].isnull().sum(), \
                        round(test[col].isnull().sum() / len(test[col]), 2)]␣
 ↪for \
                       col in test[numerical_cols] if test[col].isnull().sum()␣
 ↪> 0]
pprint(null_num_cols_test)
print(f'null_num_cols_test length: {len(null_num_cols_test)}')
```

```
[['LotFrontage', 227, 0.16],
 ['MasVnrArea', 15, 0.01],
 ['BsmtFinSF1', 1, 0.0],
 ['BsmtFinSF2', 1, 0.0],
 ['BsmtUnfSF', 1, 0.0],
 ['TotalBsmtSF', 1, 0.0],
 ['GarageYrBlt', 78, 0.05],
 ['GarageArea', 1, 0.0],
 ['BsmtFullBath', 2, 0.0],
 ['BsmtHalfBath', 2, 0.0],
 ['GarageCars', 1, 0.0]]
null_num_cols_test length: 11
```

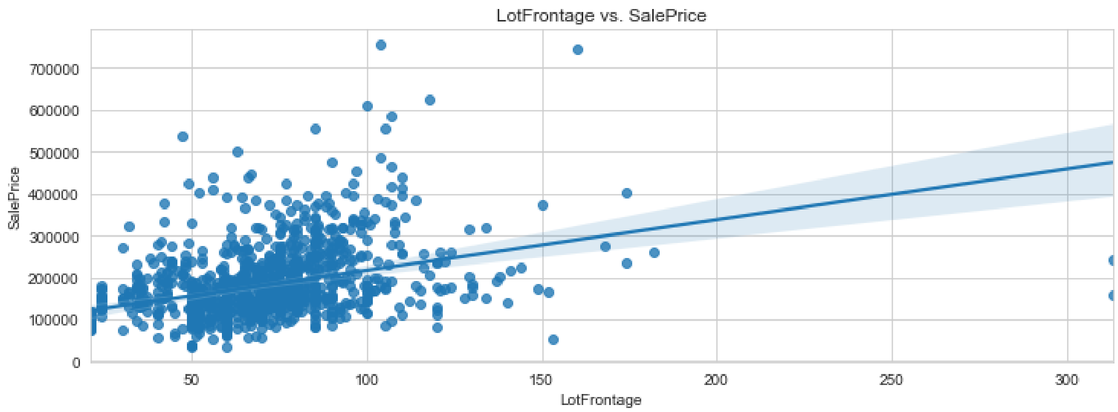
[10]:

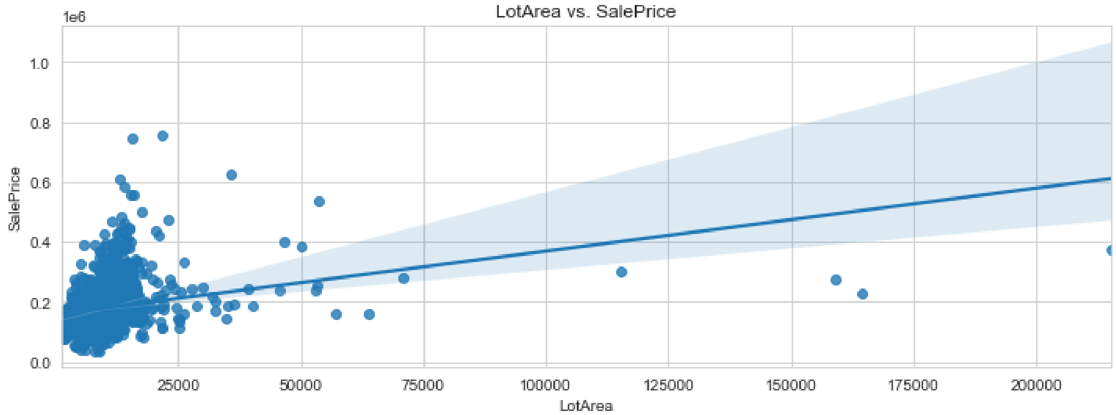
```
sns.set_style('whitegrid')
```

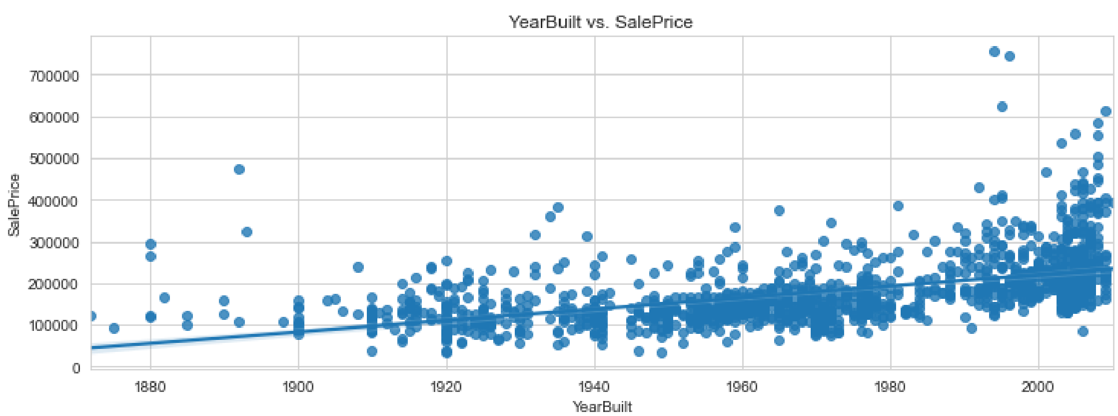
[11]:

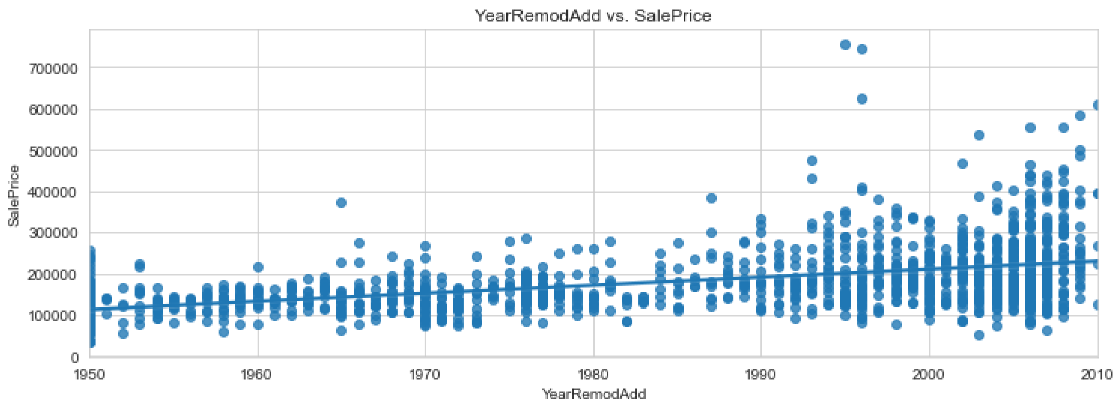
```
for ax in train[numerical_cont_features]:
    plt.figure(figsize=(12, 4))
    sns.regplot(x=train[ax], y=train['SalePrice'])
    plt.title(f'{ax} vs. SalePrice')
    plt.xlabel(ax)
    plt.ylabel('SalePrice')
```

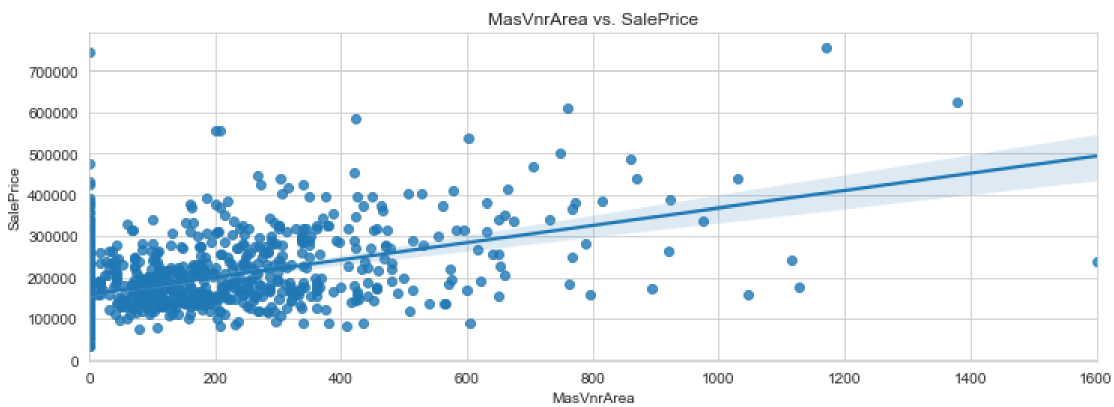
```
<ipython-input-11-ed337a0eae73>:2: RuntimeWarning: More than 20 figures have
been opened. Figures created through the pyplot interface
(`matplotlib.pyplot.figure`) are retained until explicitly closed and may
consume too much memory. (To control this warning, see the rcParam
`figure.max_open_warning`).
  plt.figure(figsize=(12, 4))
```

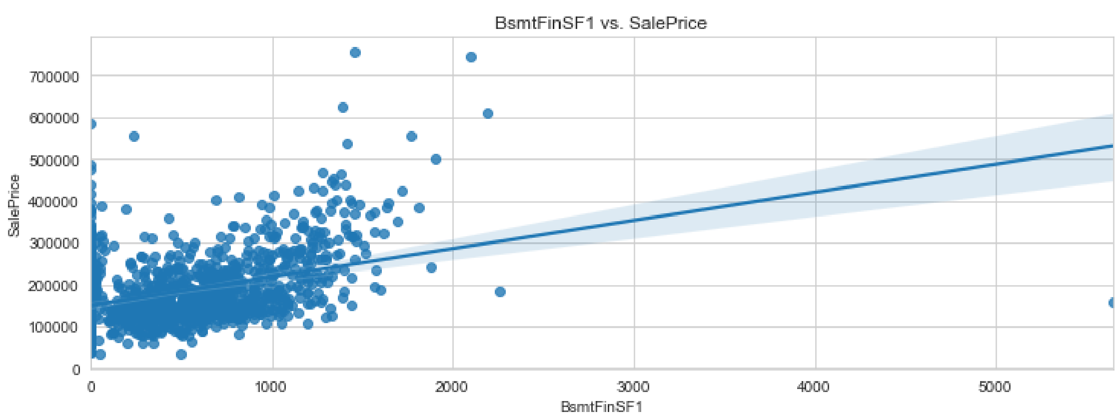

LotFrontage vs. SalePrice
SalePrice
LotFrontage
0
100000
200000
300000
400000
500000
600000
700000
50
100
150
200
250
300

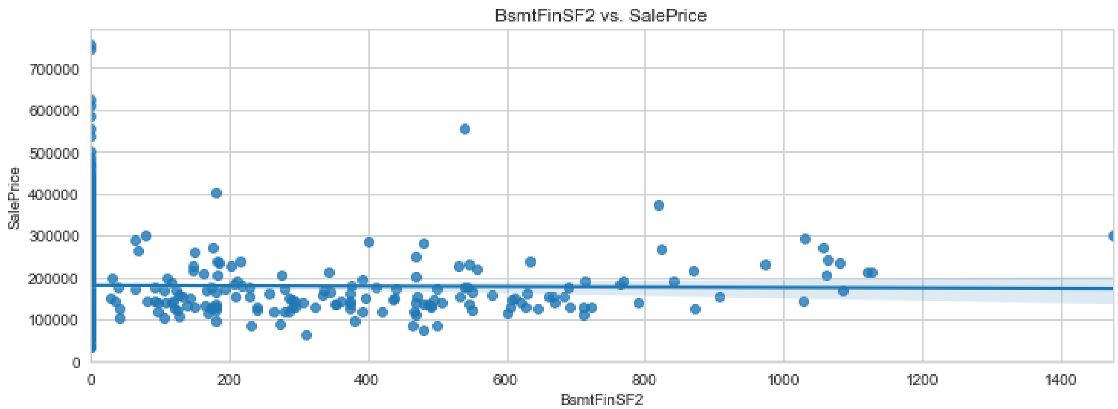

LotArea vs. SalePrice
1e6
SalePrice
LotArea
0.0
0.2
0.4
0.6
0.8
1.0
25000
50000
75000
100000
125000
150000
175000
200000

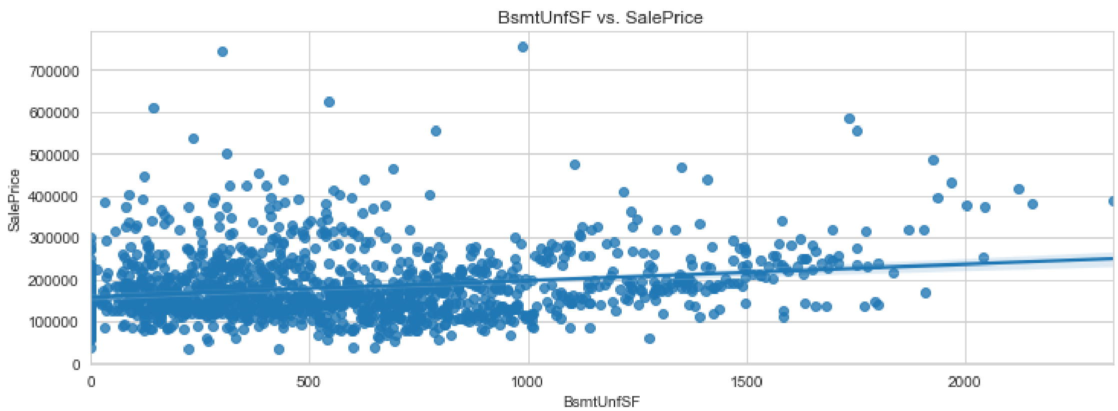

YearBuilt vs. SalePrice
SalePrice
YearBuilt
0
100000
200000
300000
400000
500000
600000
700000
1880
1900
1920
1940
1960
1980
2000

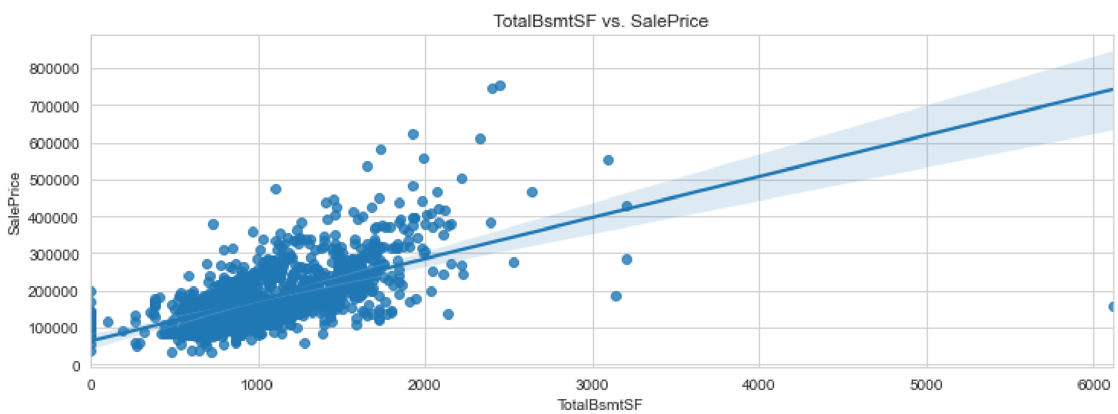

YearRemodAdd vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
1950
1960
1970
1980
1990
2000
2010
YearRemodAdd


MasVnrArea vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
200
400
600
800
1000
1200
1400
1600
MasVnrArea

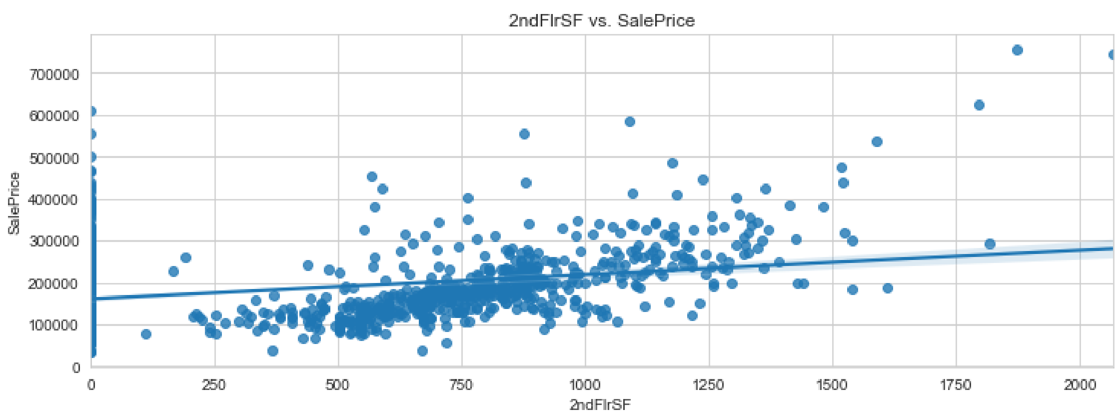

BsmtFinSF1 vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
1000
2000
3000
4000
5000
BsmtFinSF1

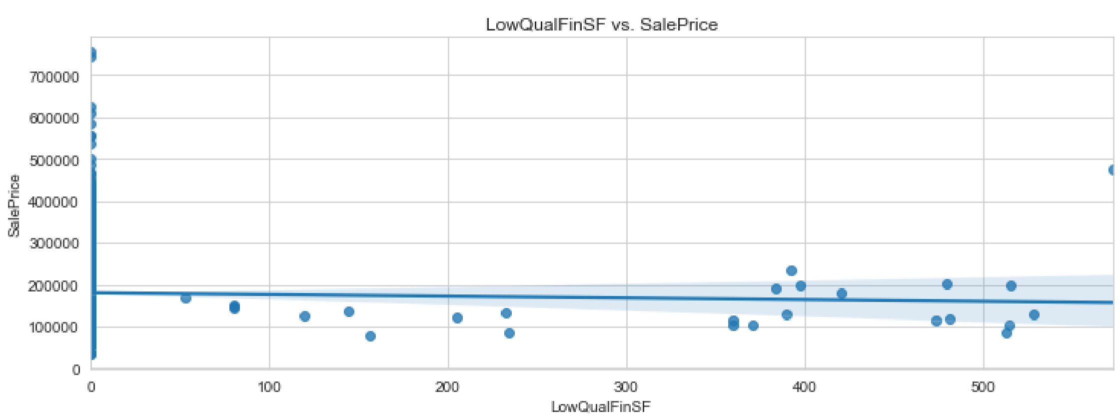

BsmtFinSF2 vs. SalePrice
SalePrice
BsmtFinSF2
0
100000
200000
300000
400000
500000
600000
700000
0
200
400
600
800
1000
1200
1400

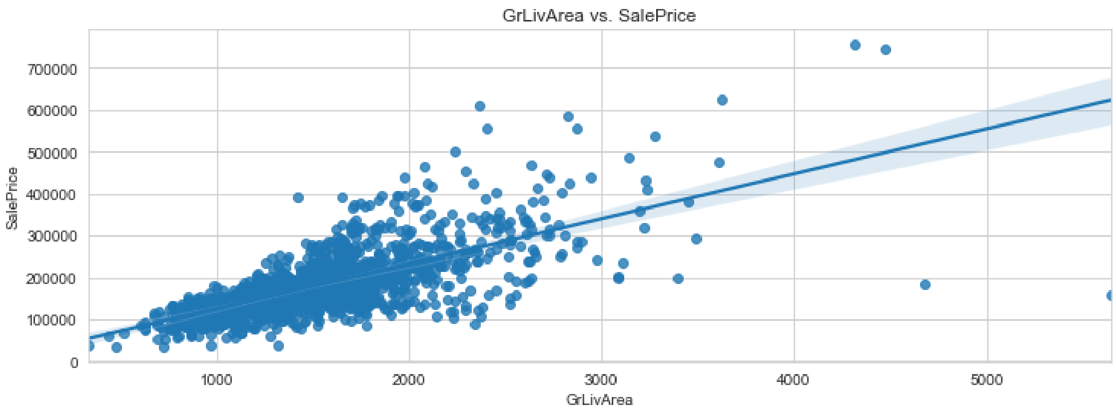

BsmtUnfSF vs. SalePrice
SalePrice
BsmtUnfSF
0
100000
200000
300000
400000
500000
600000
700000
0
500
1000
1500
2000

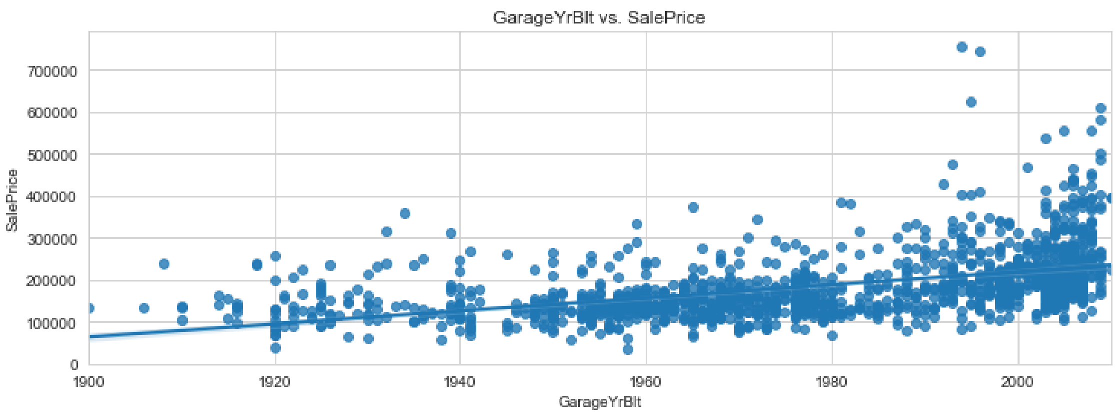

TotalBsmtSF vs. SalePrice
SalePrice
TotalBsmtSF
0
100000
200000
300000
400000
500000
600000
700000
800000
0
1000
2000
3000
4000
5000
6000

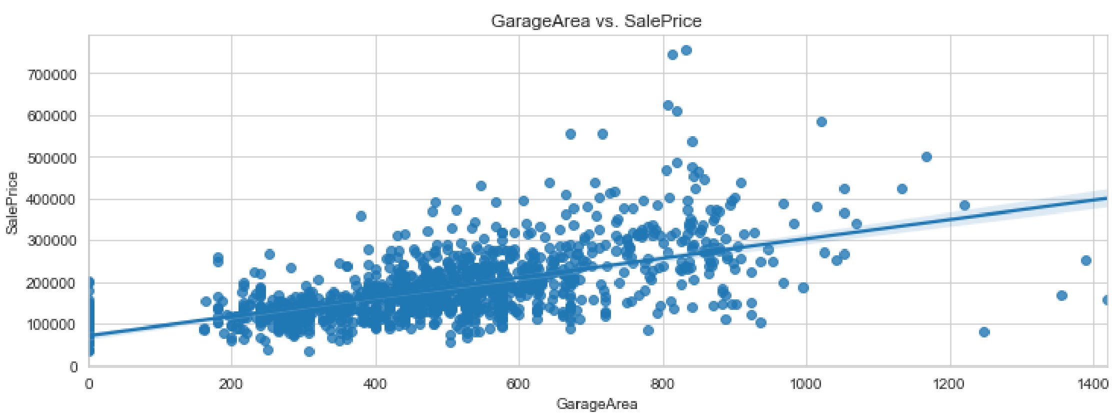

1stFlrSF vs. SalePrice
SalePrice
1stFlrSF
0
100000
200000
300000
400000
500000
600000
700000
500
1000
1500
2000
2500
3000
3500
4000
4500

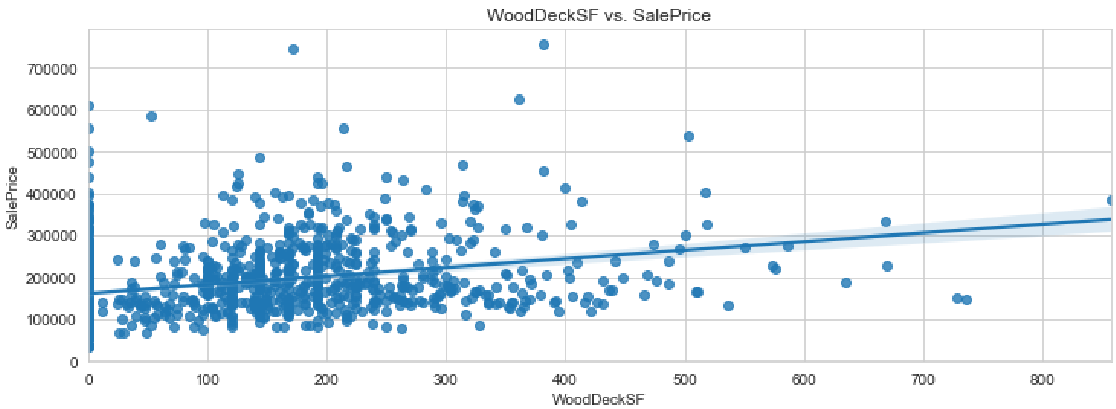

2ndFlrSF vs. SalePrice
SalePrice
2ndFlrSF
0
100000
200000
300000
400000
500000
600000
700000
0
250
500
750
1000
1250
1500
1750
2000

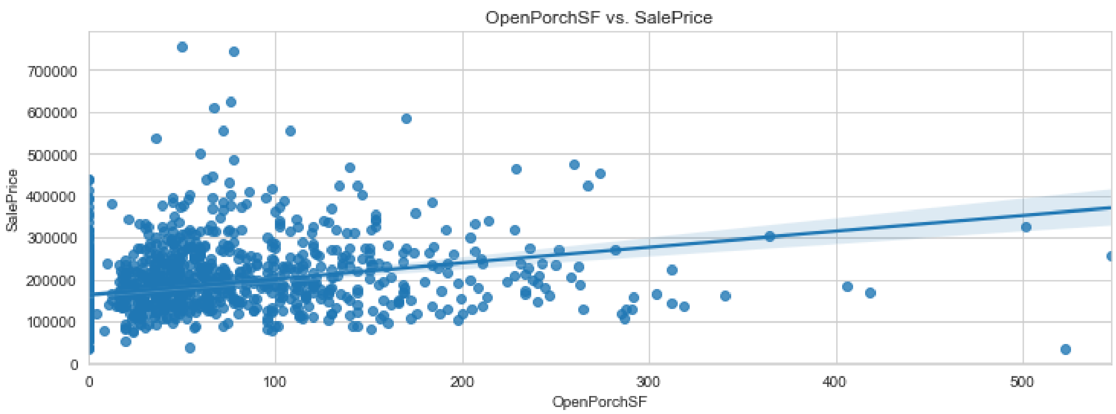

LowQualFinSF vs. SalePrice
SalePrice
LowQualFinSF
0
100000
200000
300000
400000
500000
600000
700000
0
100
200
300
400
500

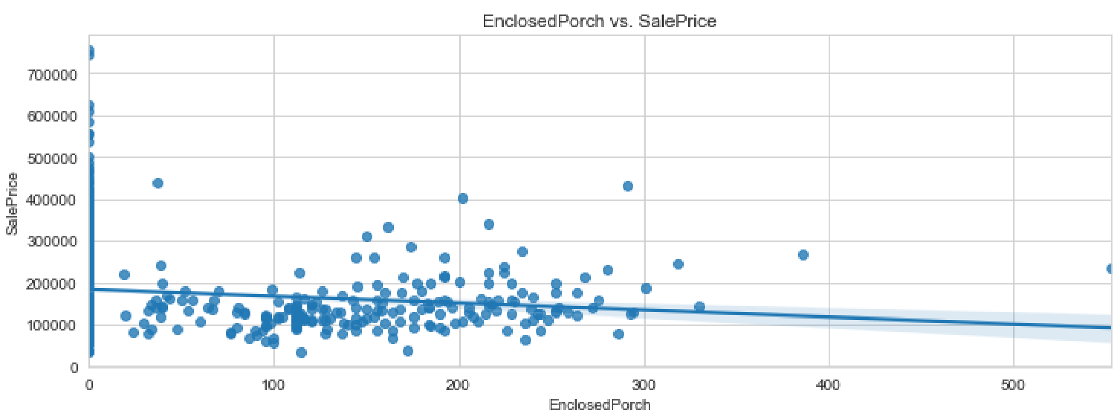

GrLivArea vs. SalePrice
SalePrice
GrLivArea
0
100000
200000
300000
400000
500000
600000
700000
1000
2000
3000
4000
5000

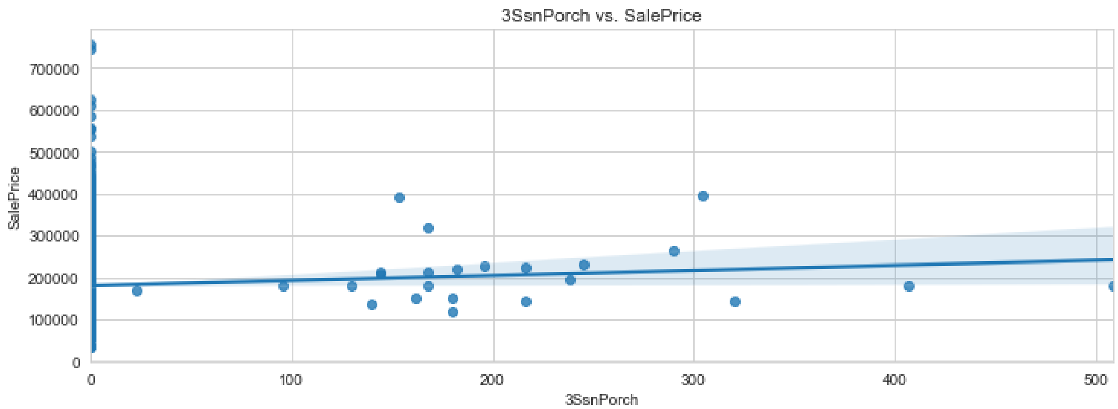

GarageYrBlt vs. SalePrice
SalePrice
GarageYrBlt
0
100000
200000
300000
400000
500000
600000
700000
1900
1920
1940
1960
1980
2000

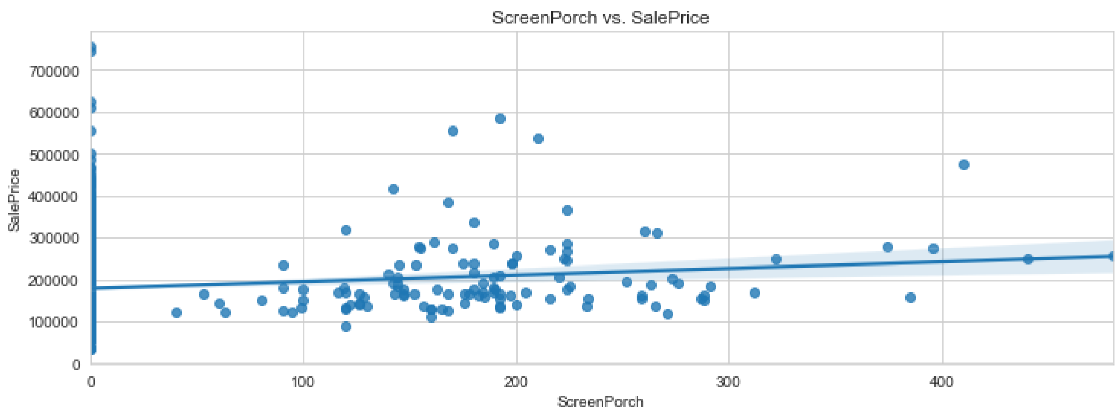

GarageArea vs. SalePrice
SalePrice
GarageArea
0
100000
200000
300000
400000
500000
600000
700000
0
200
400
600
800
1000
1200
1400

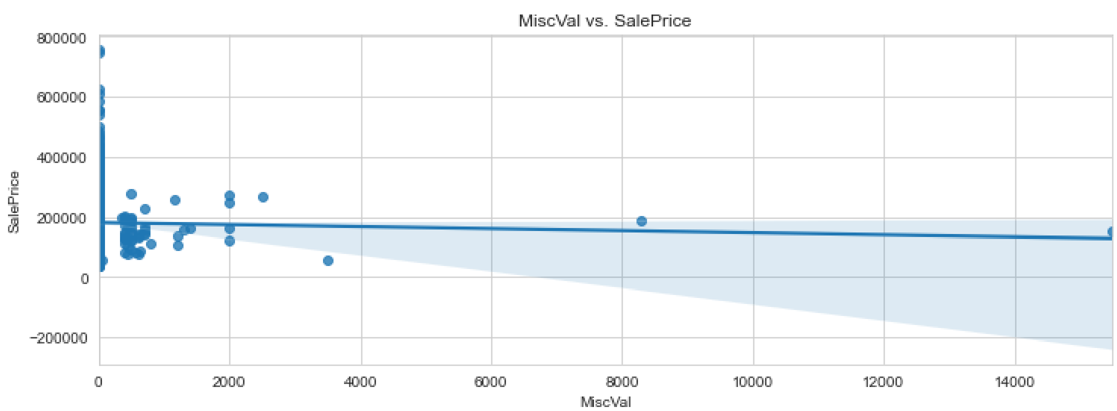

WoodDeckSF vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
100
200
300
400
500
600
700
800
WoodDeckSF


OpenPorchSF vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
100
200
300
400
500
OpenPorchSF


EnclosedPorch vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
100
200
300
400
500
EnclosedPorch

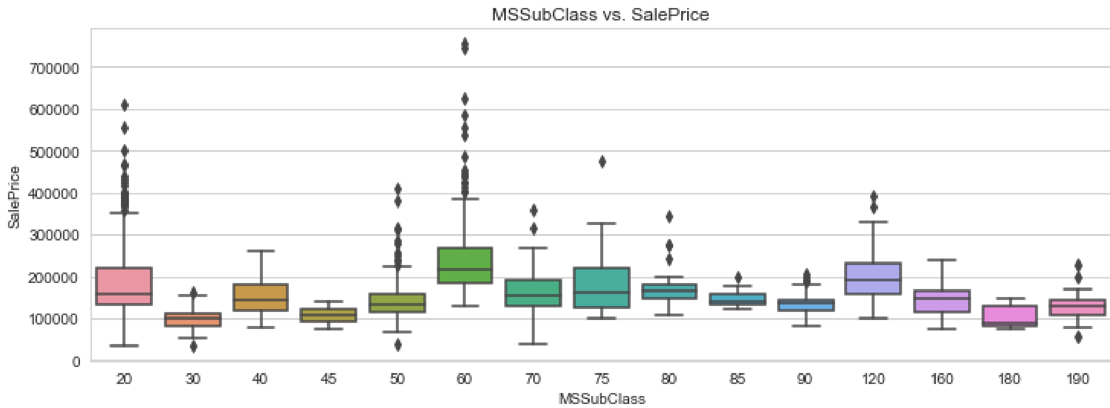


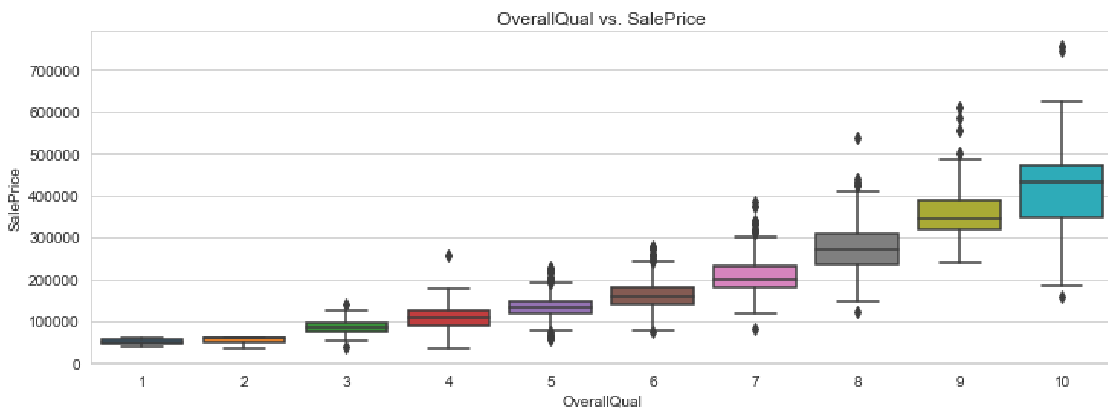


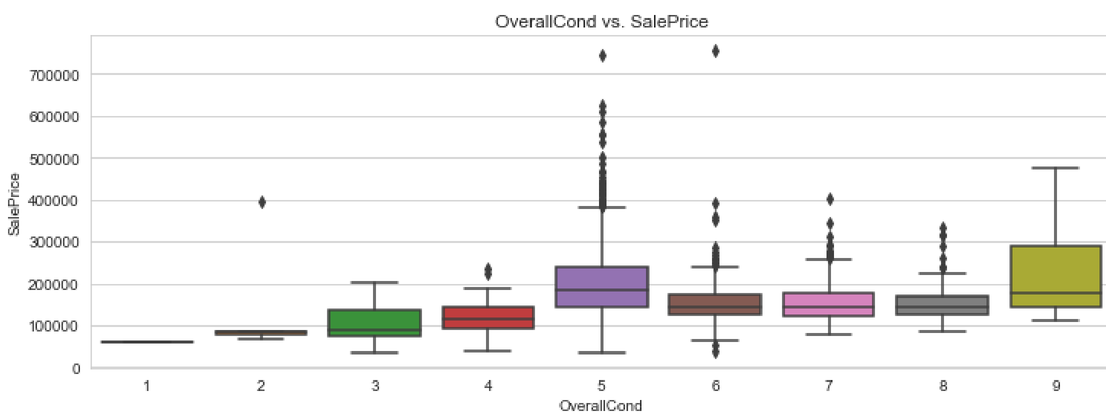


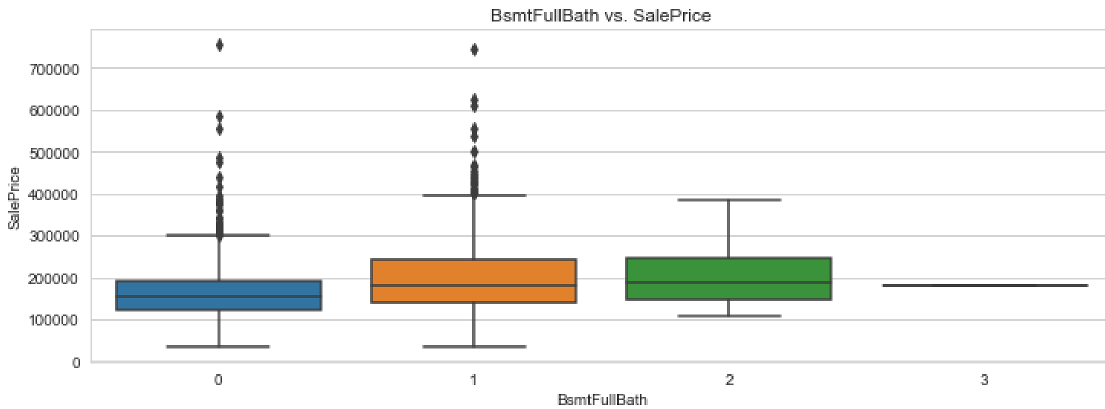
[12]:
```
for ax in train[numerical_cat_features]:
    plt.figure(figsize=(12, 4))
    sns.boxplot(x=train[ax], y=train['SalePrice'])
```

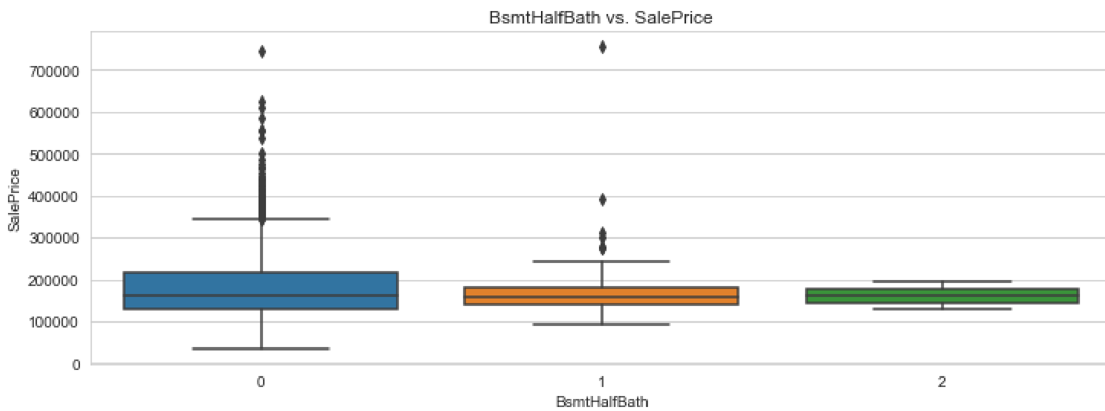
```
    plt.title(f'{ax} vs. SalePrice')
    plt.xlabel(ax)
    plt.ylabel('SalePrice')
```

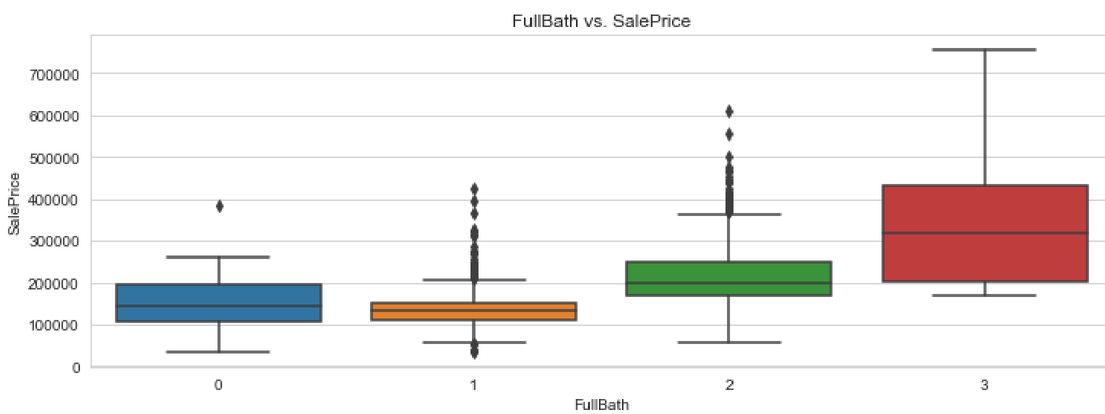


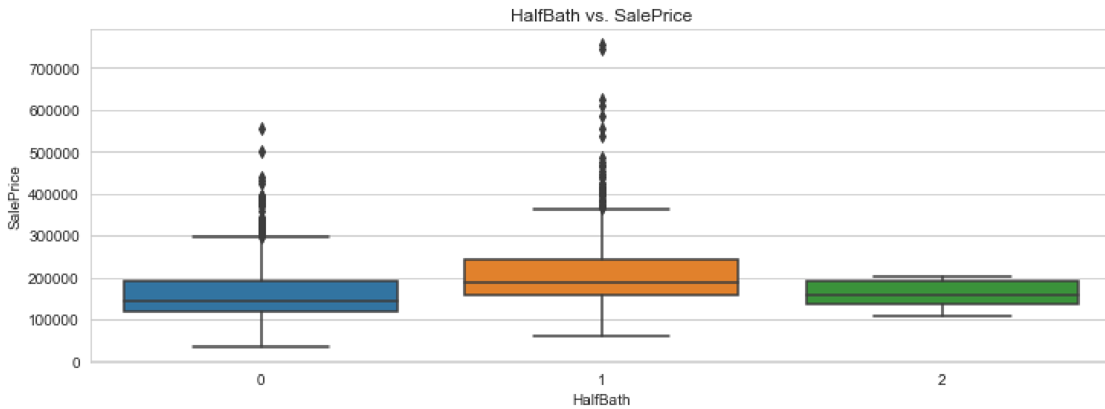


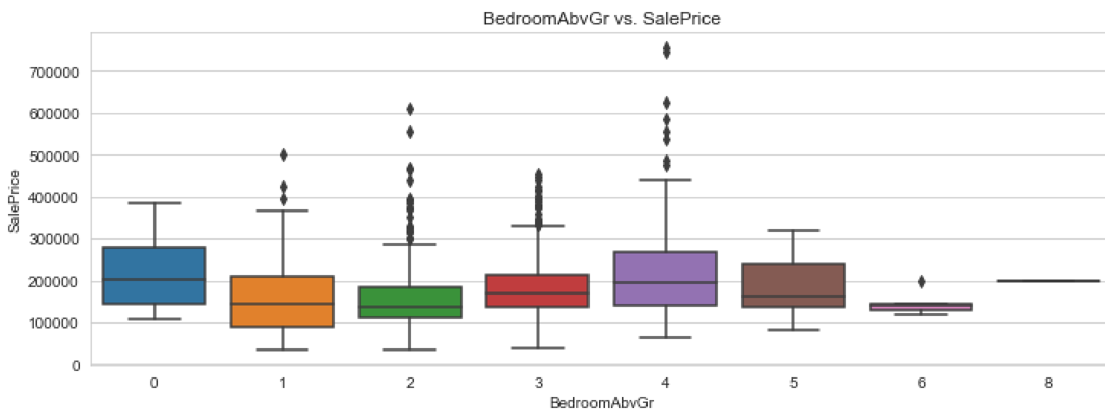


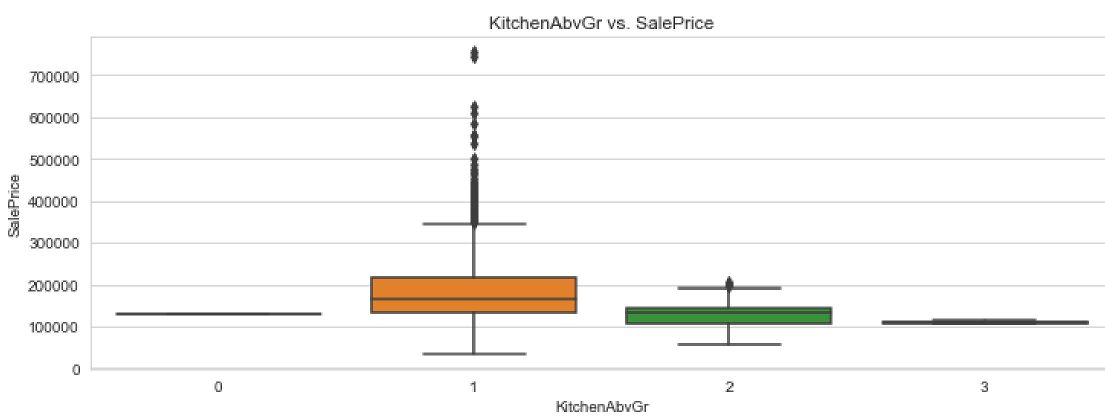

BsmtFullBath vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
1
2
3
BsmtFullBath

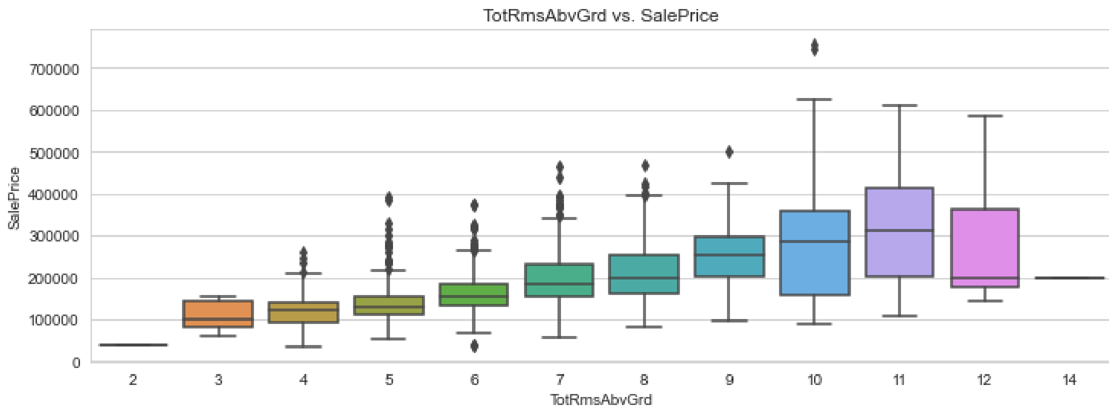

BsmtHalfBath vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
1
2
BsmtHalfBath

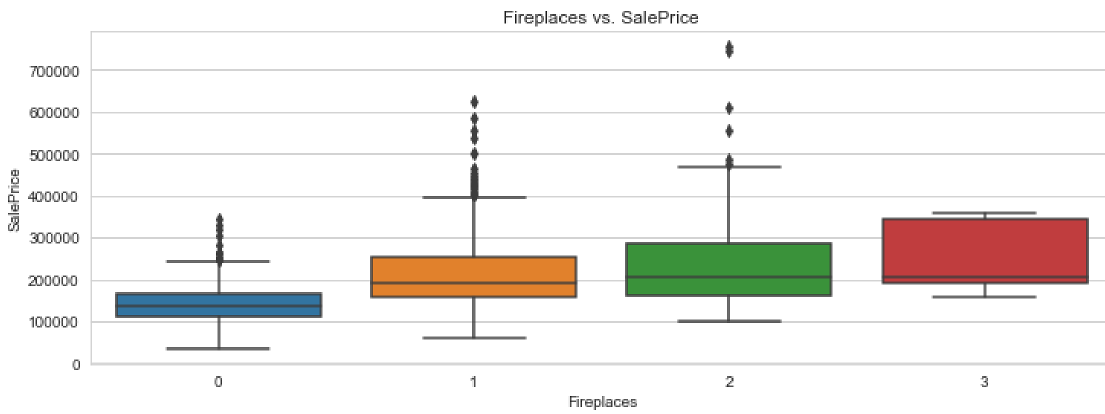

FullBath vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
1
2
3
FullBath

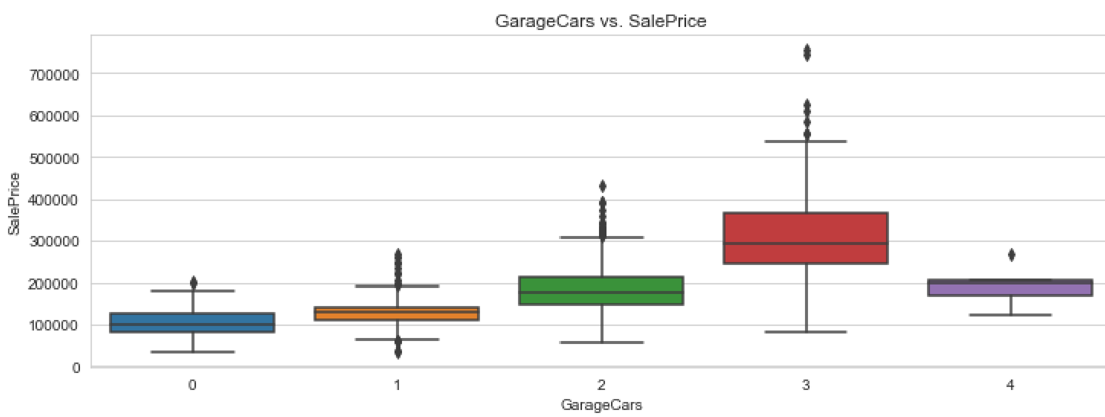

HalfBath vs. SalePrice
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
1
2
HalfBath

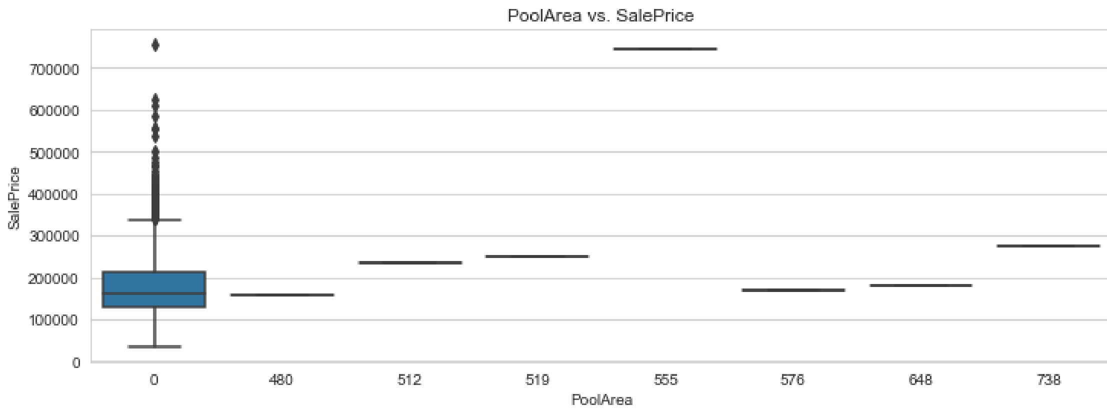

BedroomAbvGr vs. SalePrice
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
1
2
3
4
5
6
8
BedroomAbvGr

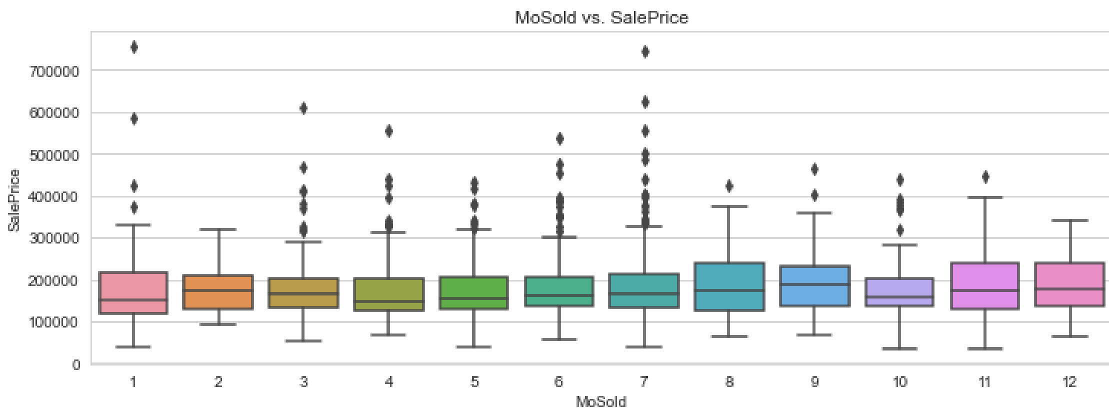

KitchenAbvGr vs. SalePrice
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
1
2
3
KitchenAbvGr


TotRmsAbvGrd vs. SalePrice
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
2
3
4
5
6
7
8
9
10
11
12
14
TotRmsAbvGrd


Fireplaces vs. SalePrice
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
1
2
3
Fireplaces


GarageCars vs. SalePrice
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
1
2
3
4
GarageCars

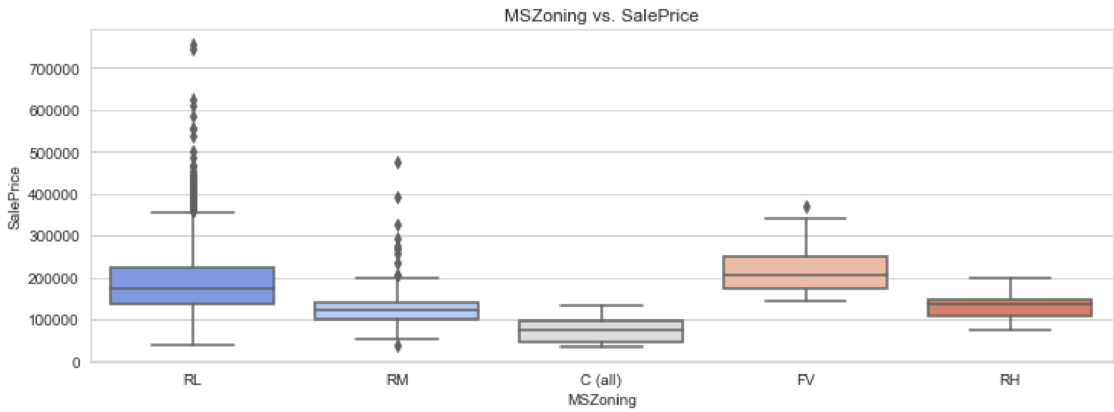


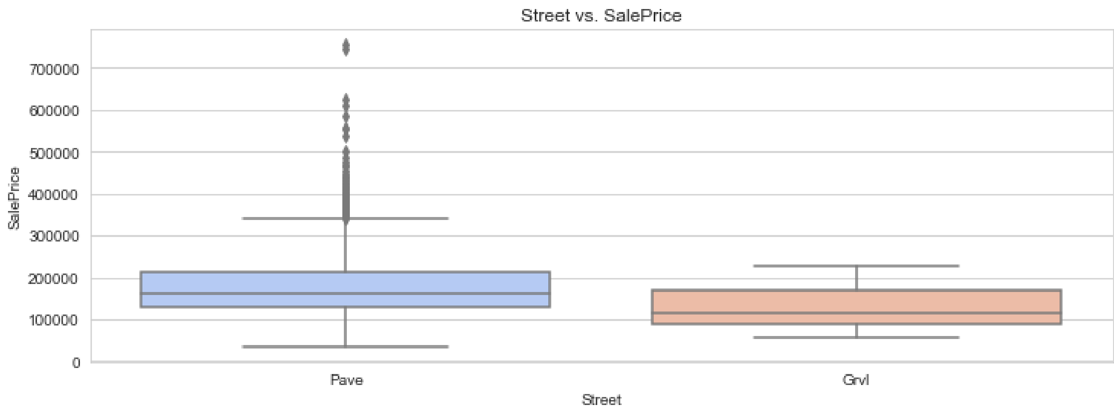


[13]:
```
for ax in train[categorical_features]:
    plt.figure(figsize=(12, 4))
    sns.boxplot(x=train[ax], y=train['SalePrice'], palette='coolwarm')
```

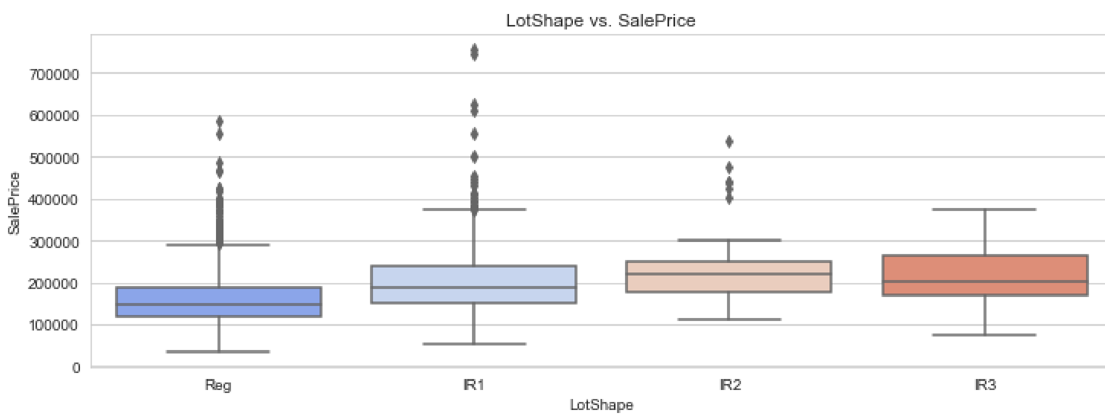
```
    plt.title(f'{ax} vs. SalePrice')
    plt.xlabel(ax)
    plt.ylabel('SalePrice')
```

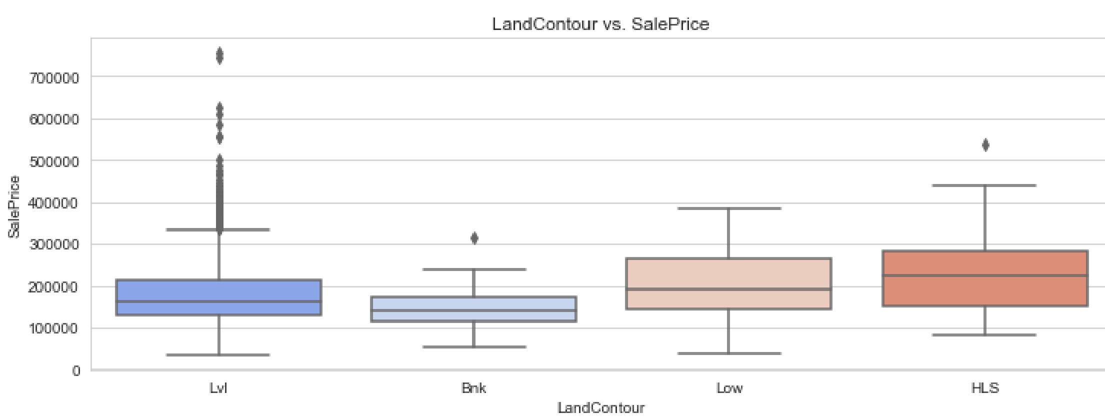
```
<ipython-input-13-6794b9677fd5>:2: RuntimeWarning: More than 20 figures have
been opened. Figures created through the pyplot interface
(`matplotlib.pyplot.figure`) are retained until explicitly closed and may
consume too much memory. (To control this warning, see the rcParam
`figure.max_open_warning`).
  plt.figure(figsize=(12, 4))
```

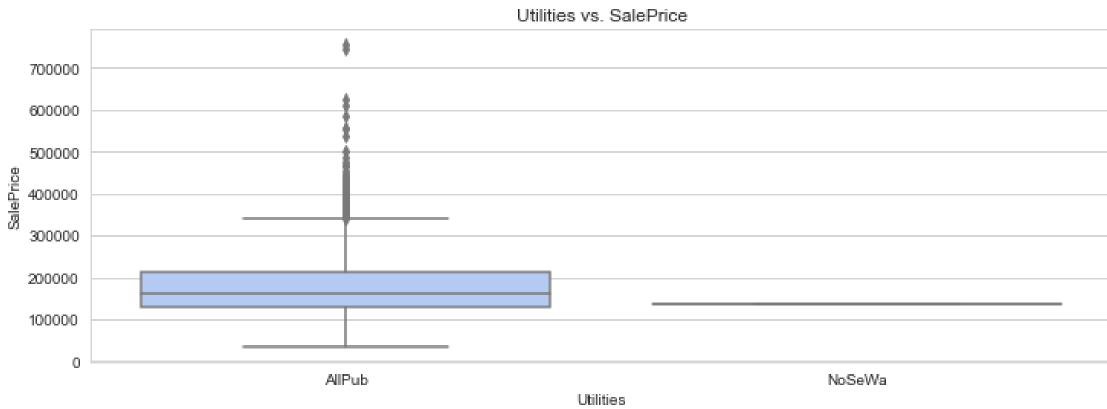


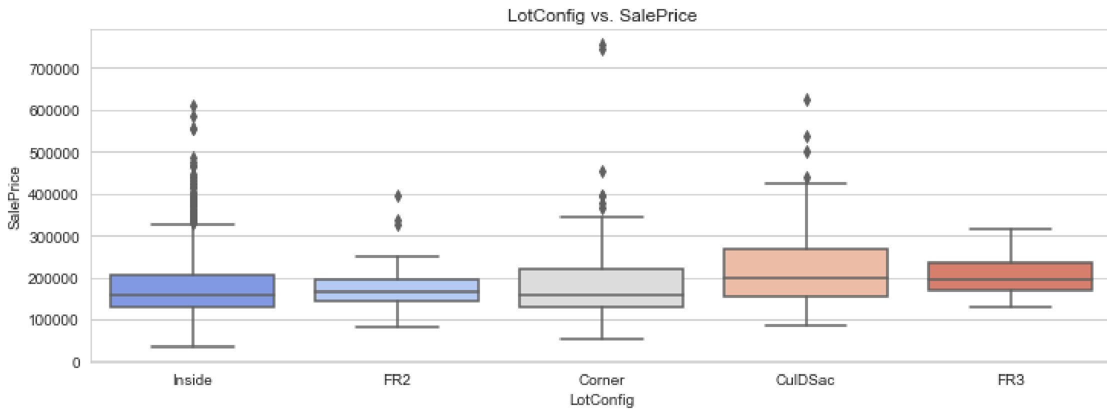


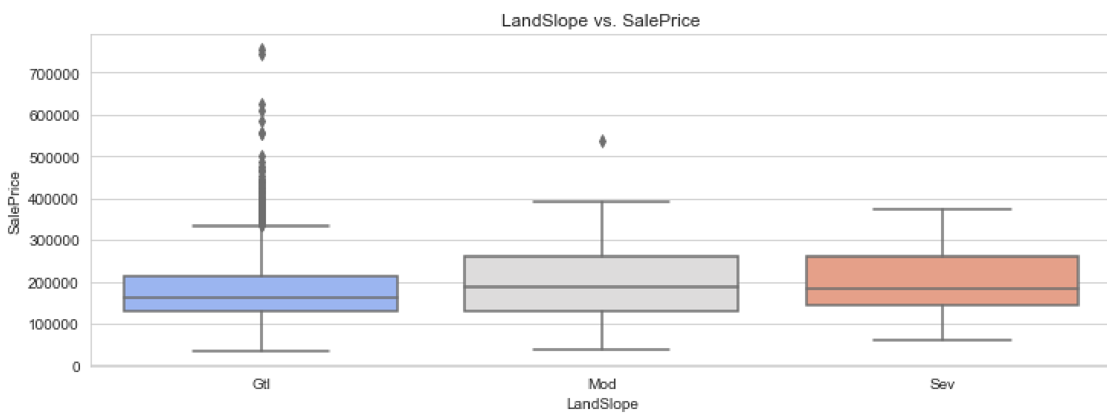

Alley vs. SalePrice
SalePrice
250000
200000
150000
100000
50000
Grvl
Pave
Alley

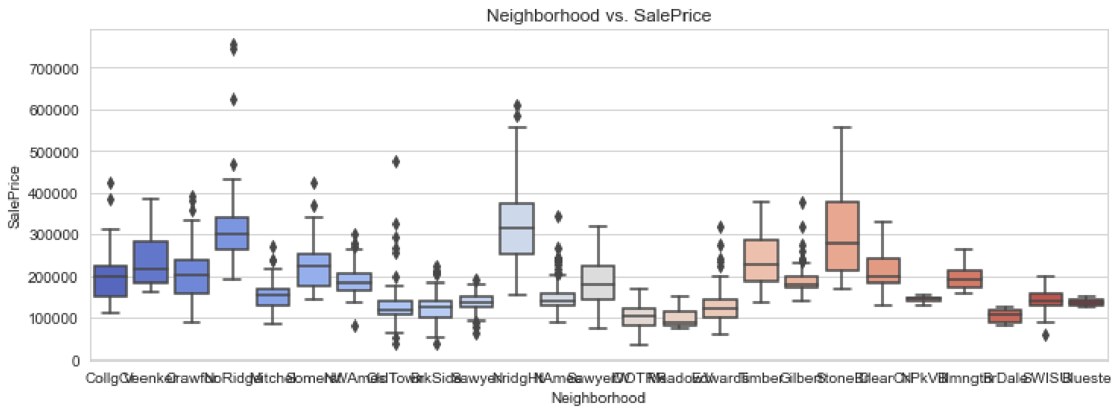

LotShape vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Reg
IR1
IR2
IR3
LotShape

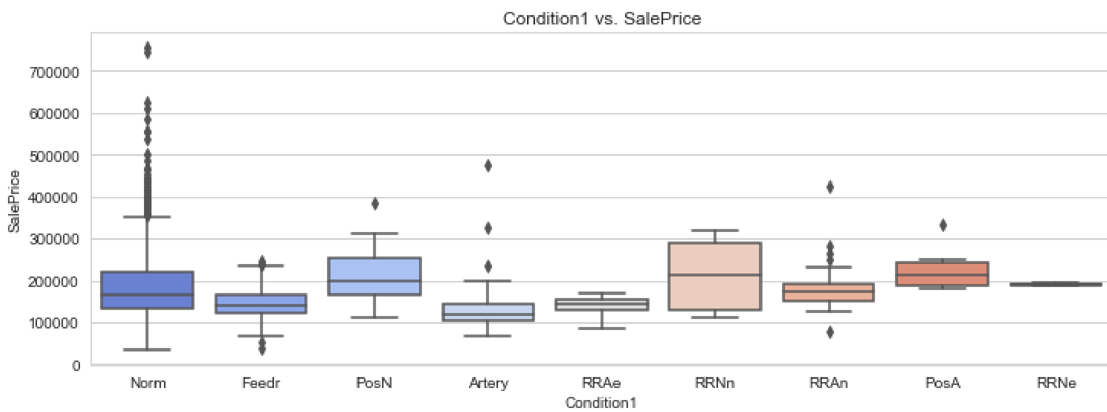

LandContour vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Lvl
Bnk
Low
HLS
LandContour

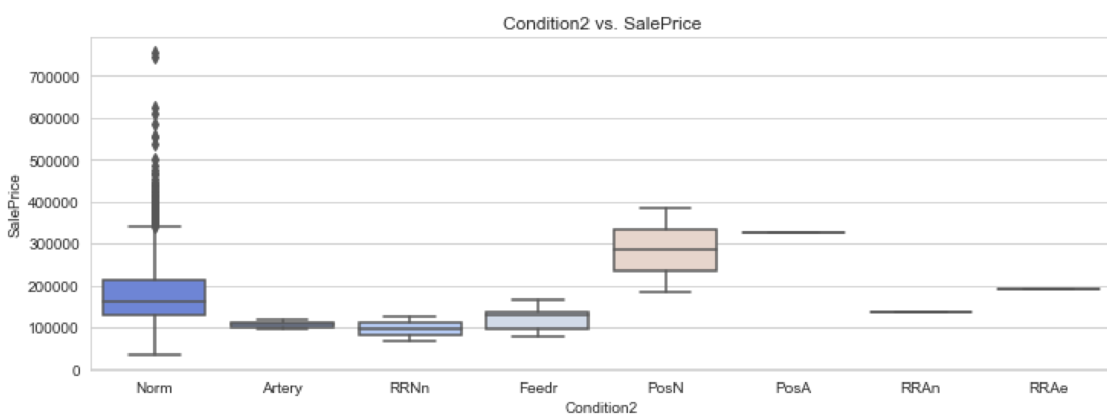

Utilities vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
AllPub
NoSeWa
Utilities

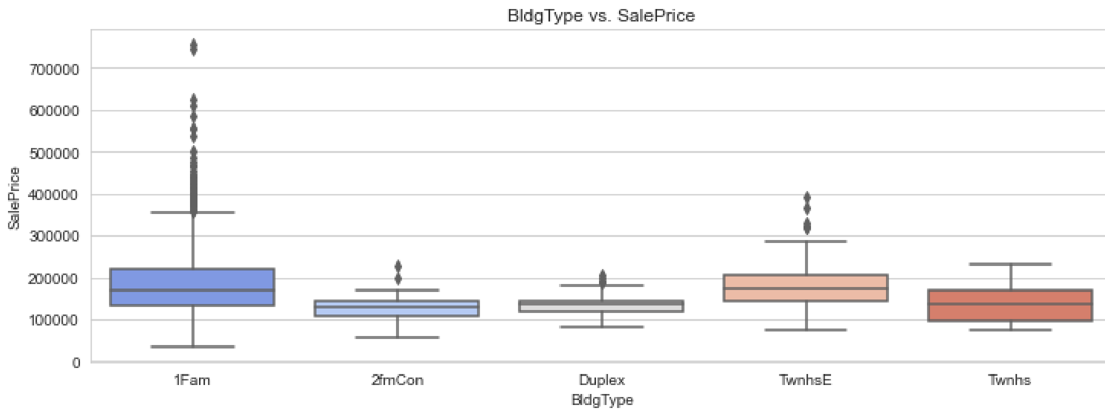

LotConfig vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Inside
FR2
Corner
CulDSac
FR3
LotConfig

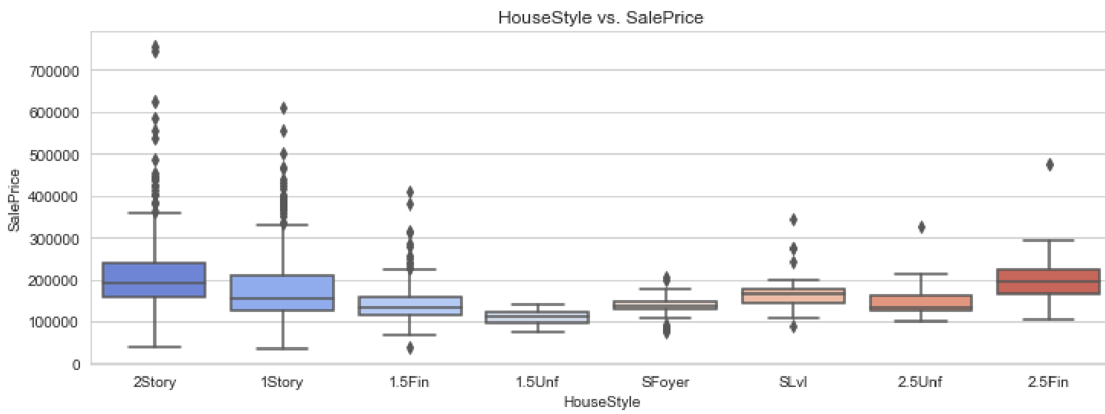

LandSlope vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Gtl
Mod
Sev
LandSlope

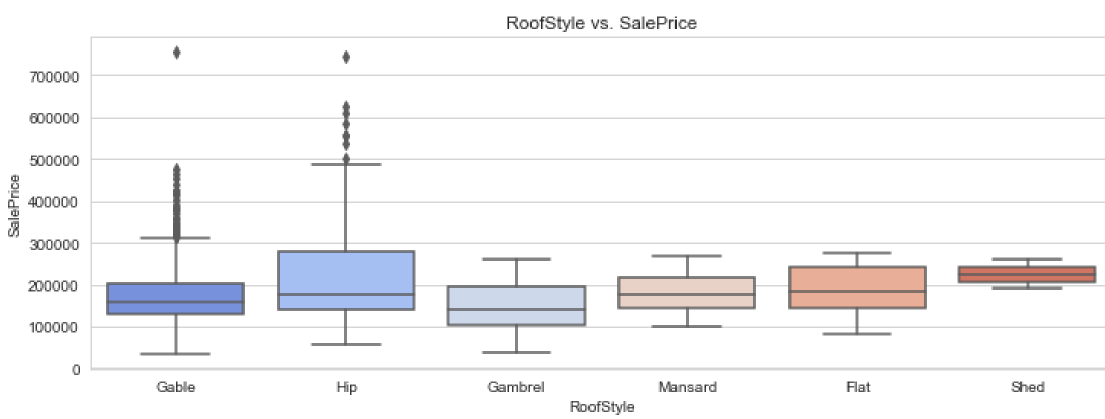

Neighborhood vs. SalePrice
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Neighborhood

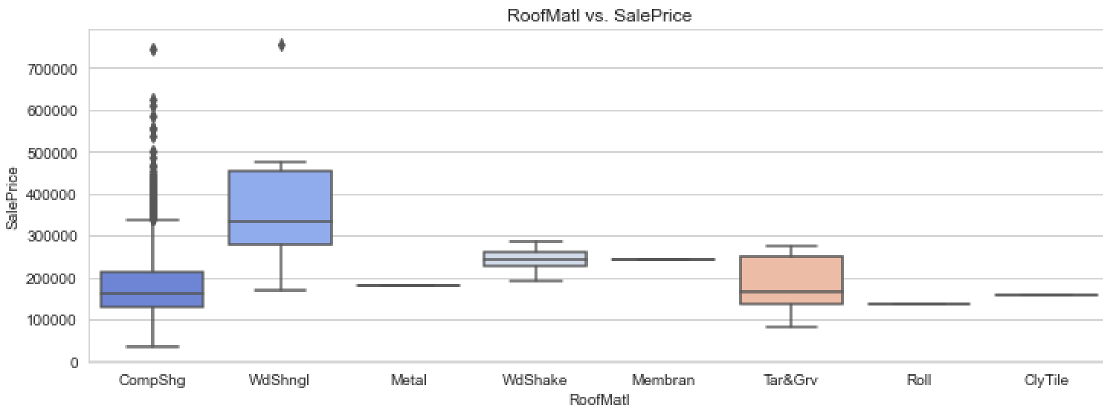

Condition1 vs. SalePrice
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Norm
Feedr
PosN
Artery
RRAe
RRNn
RRAn
PosA
RRNe
Condition1

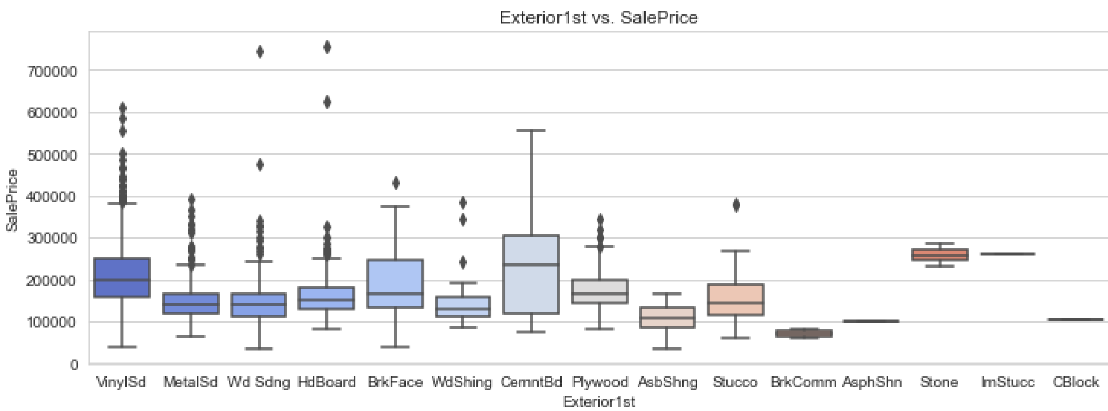

Condition2 vs. SalePrice
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Norm
Artery
RRNn
Feedr
PosN
PosA
RRAn
RRAe
Condition2

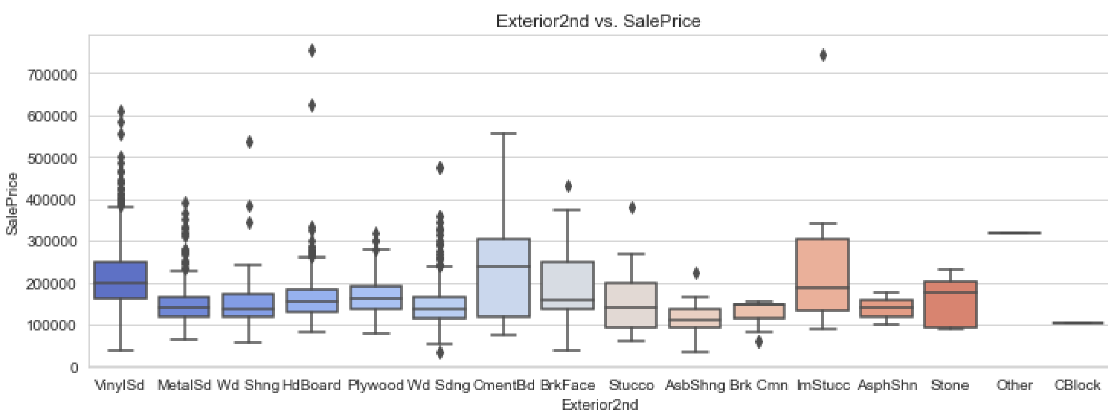

BldgType vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
1Fam
2fmCon
Duplex
TwnhsE
Twnhs
BldgType

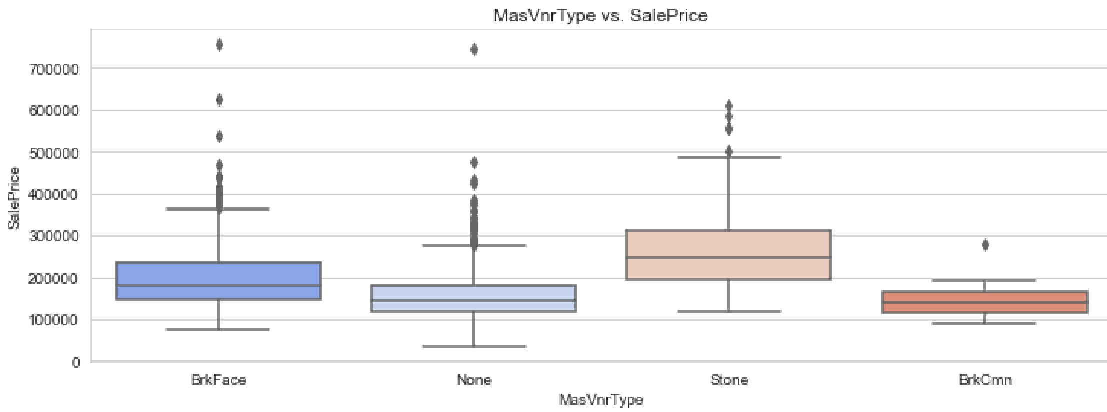

HouseStyle vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
2Story
1Story
1.5Fin
1.5Unf
SFoyer
SLvl
2.5Unf
2.5Fin
HouseStyle

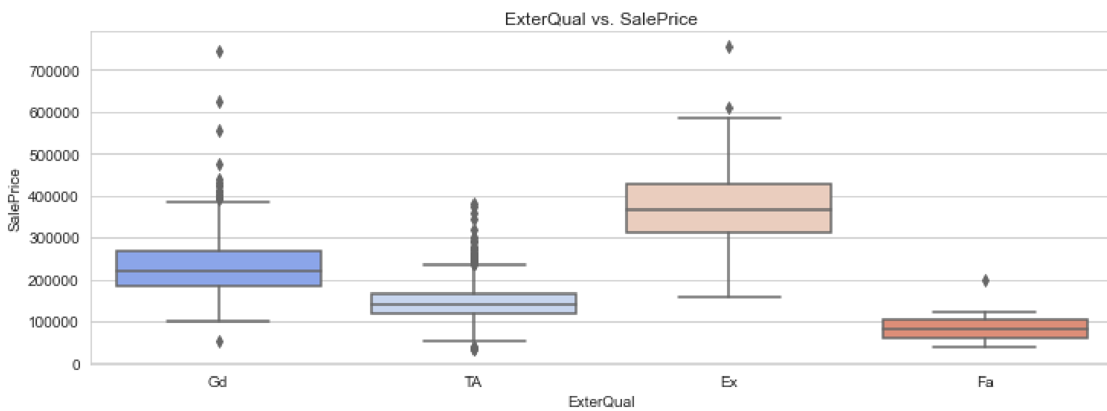

RoofStyle vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Gable
Hip
Gambrel
Mansard
Flat
Shed
RoofStyle

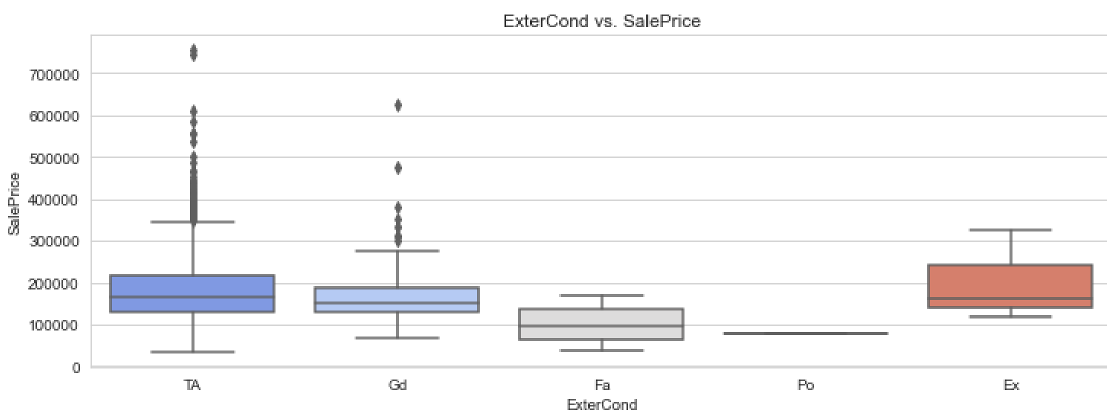

RoofMatl vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
CompShg
WdShngl
Metal
WdShake
Membran
Tar&Grv
Roll
ClyTile
RoofMatl

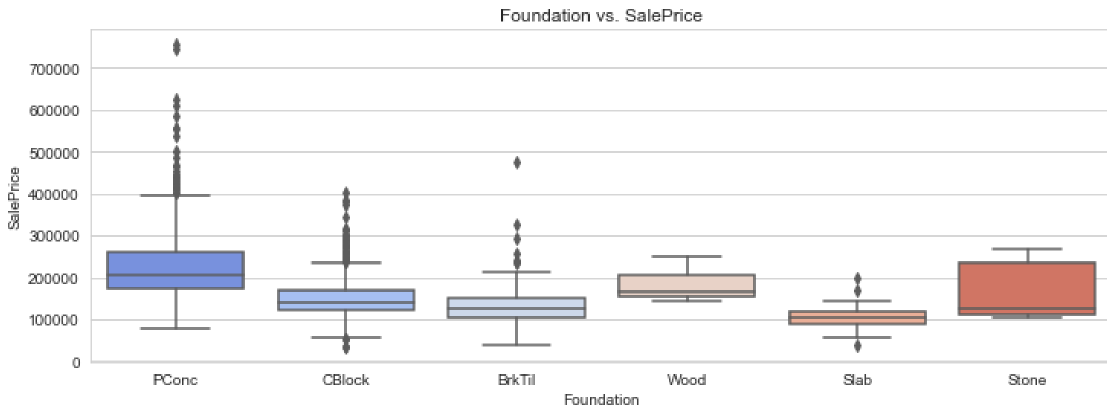

Exterior1st vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
VinylSd
MetalSd
Wd Sdng
HdBoard
BrkFace
WdShing
CemntBd
Plywood
AsbShng
Stucco
BrkComm
AsphShn
Stone
ImStucc
CBlock
Exterior1st


Exterior2nd vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
VinylSd
MetalSd
Wd Shng
HdBoard
Plywood
Wd Sdng
CmentBd
BrkFace
Stucco
AsbShng
Brk Cmn
ImStucc
AsphShn
Stone
Other
CBlock
Exterior2nd

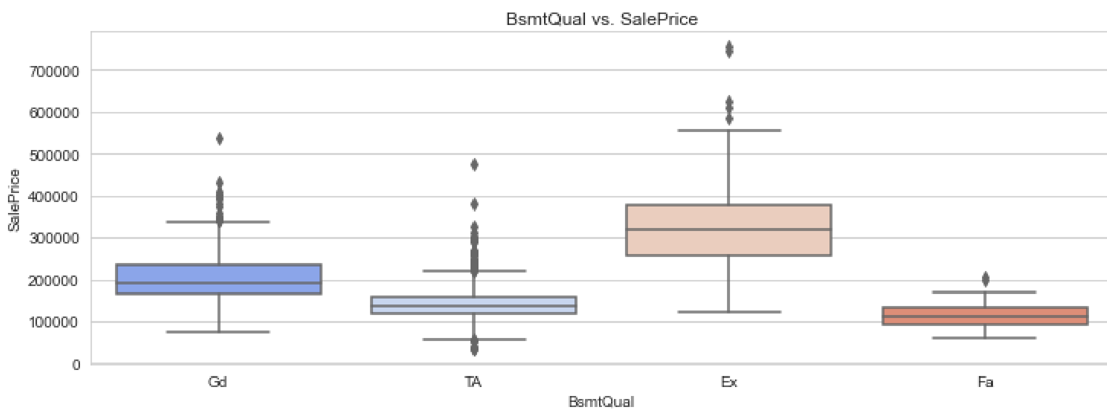

MasVnrType vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
BrkFace
None
Stone
BrkCmn
MasVnrType


ExterQual vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Gd
TA
Ex
Fa
ExterQual

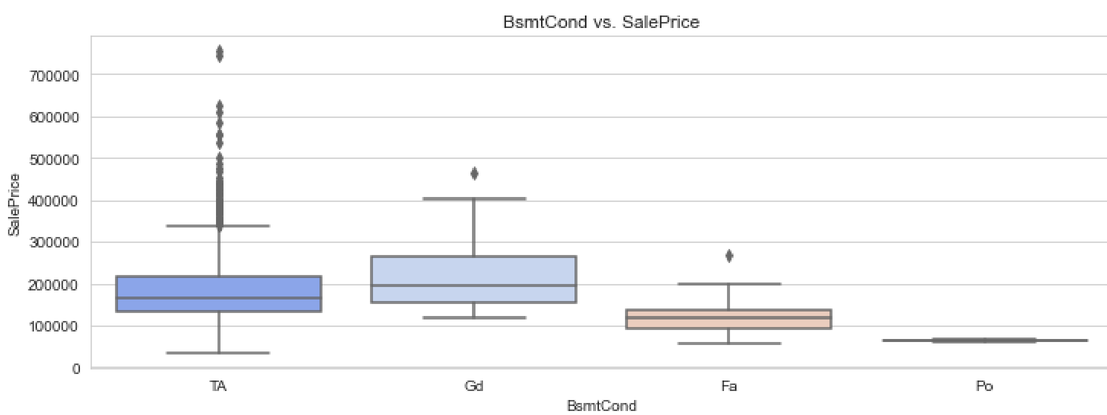

ExterCond vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
TA
Gd
Fa
Po
Ex
ExterCond



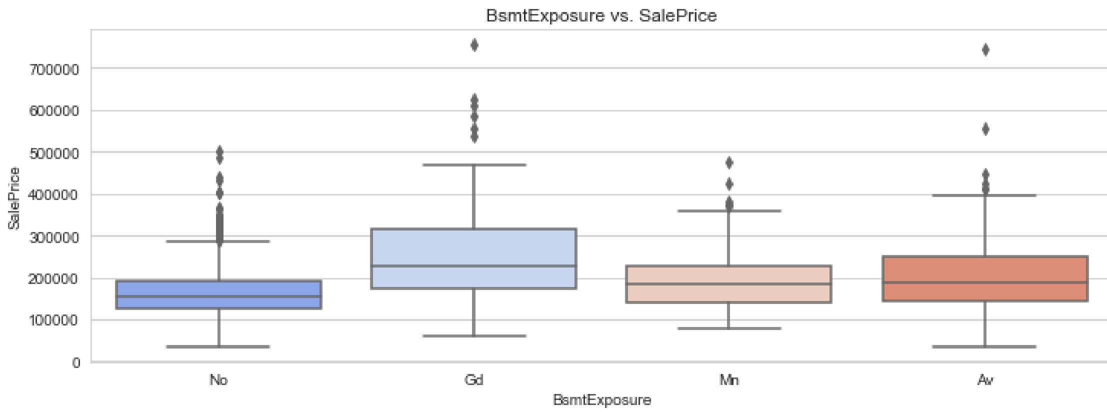
Foundation vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
PConc
CBlock
BrkTil
Wood
Slab
Stone
Foundation

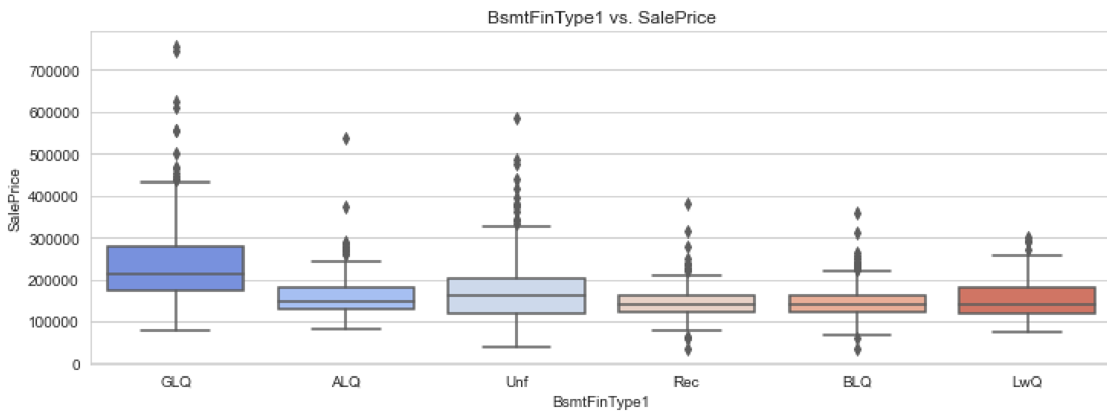


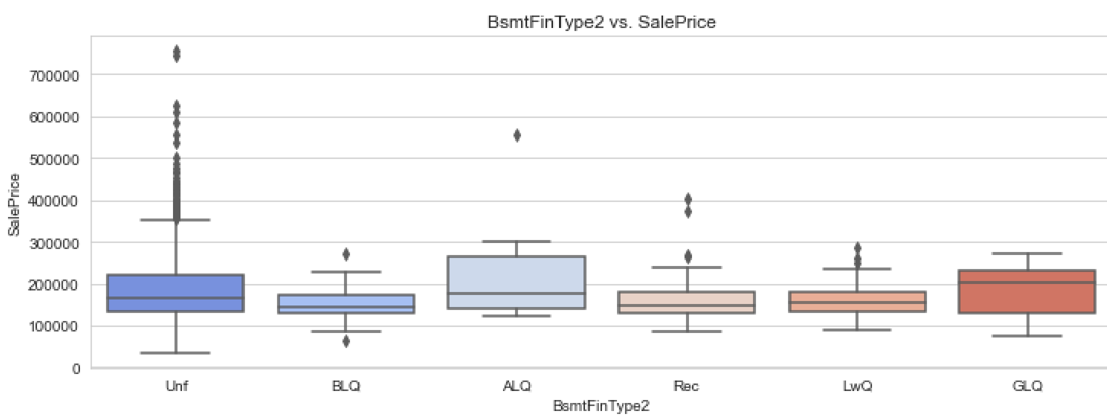
BsmtQual vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Gd
TA
Ex
Fa
BsmtQual

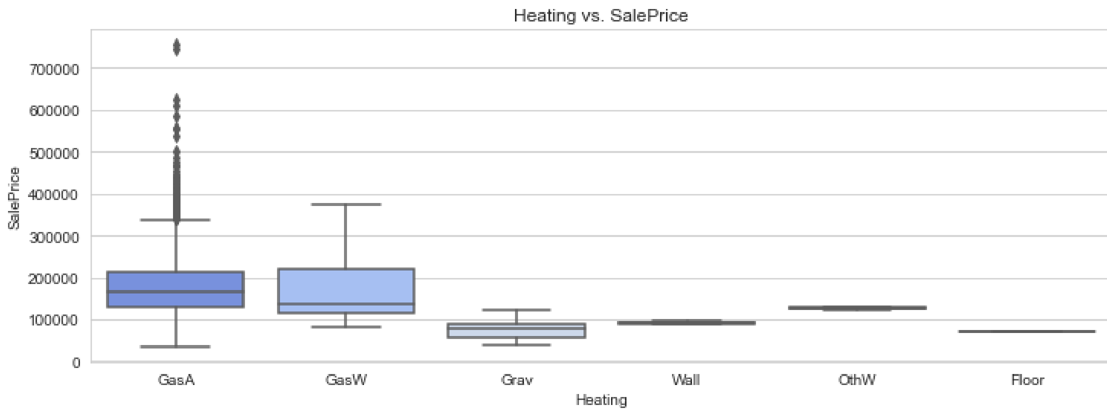


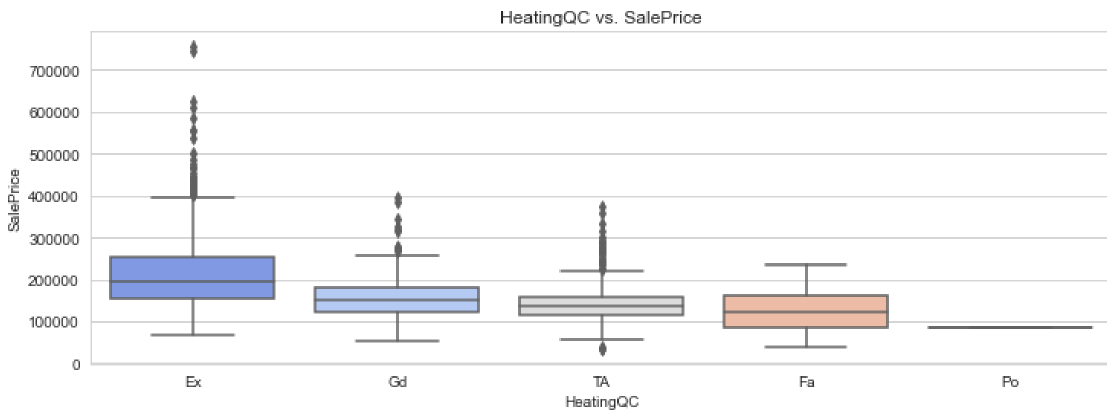
BsmtCond vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
TA
Gd
Fa
Po
BsmtCond

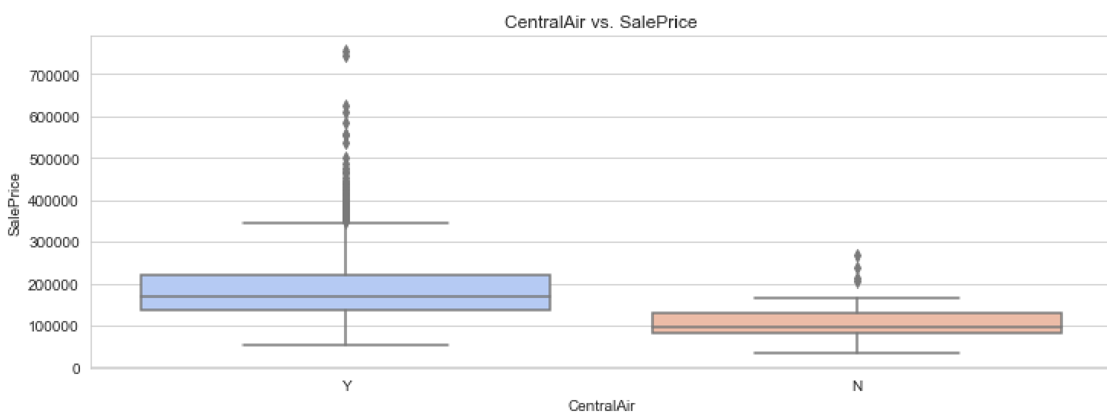

BsmtExposure vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
No
Gd
Mn
Av
BsmtExposure

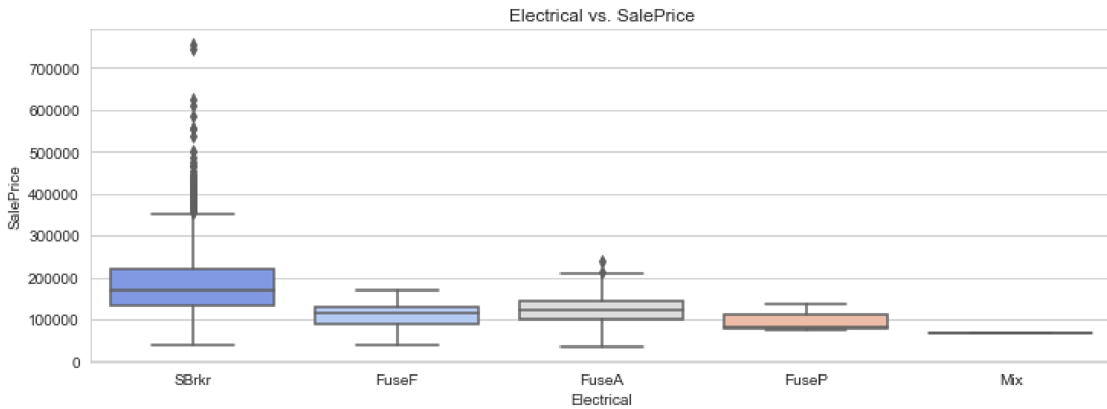

BsmtFinType1 vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
GLQ
ALQ
Unf
Rec
BLQ
LwQ
BsmtFinType1

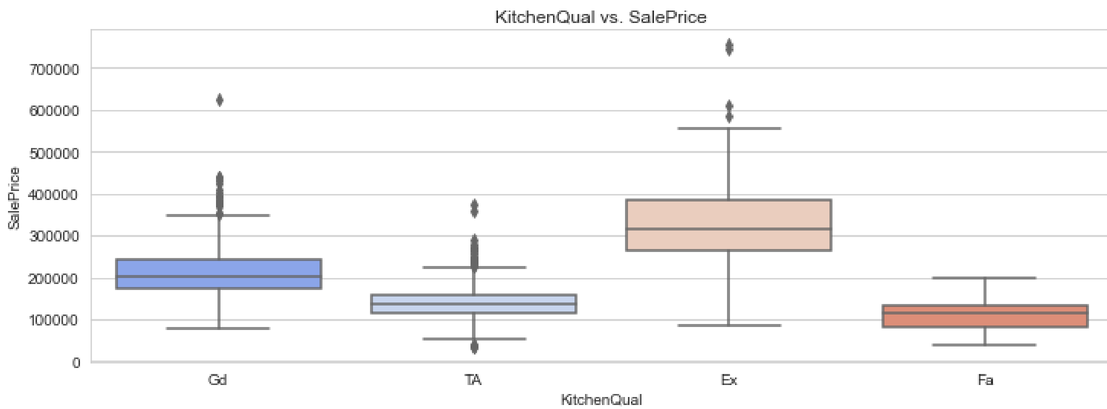

BsmtFinType2 vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Unf
BLQ
ALQ
Rec
LwQ
GLQ
BsmtFinType2

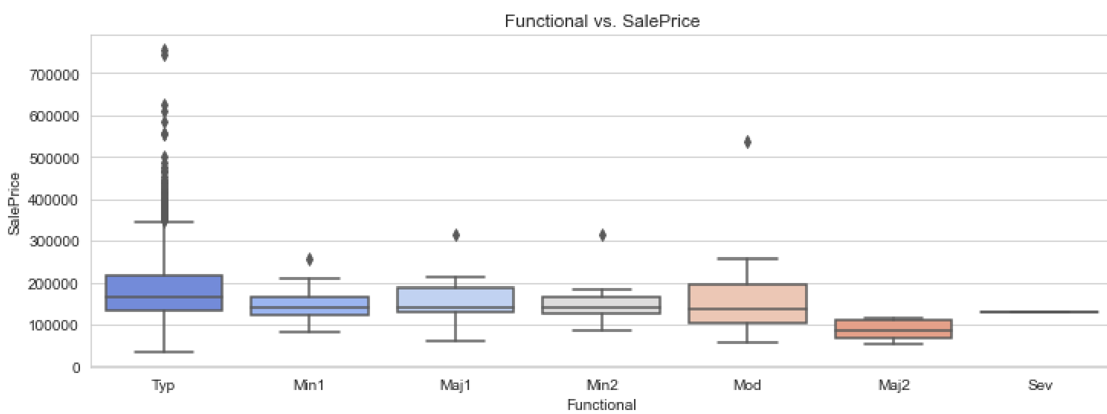

Heating vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
GasA
GasW
Grav
Wall
OthW
Floor
Heating

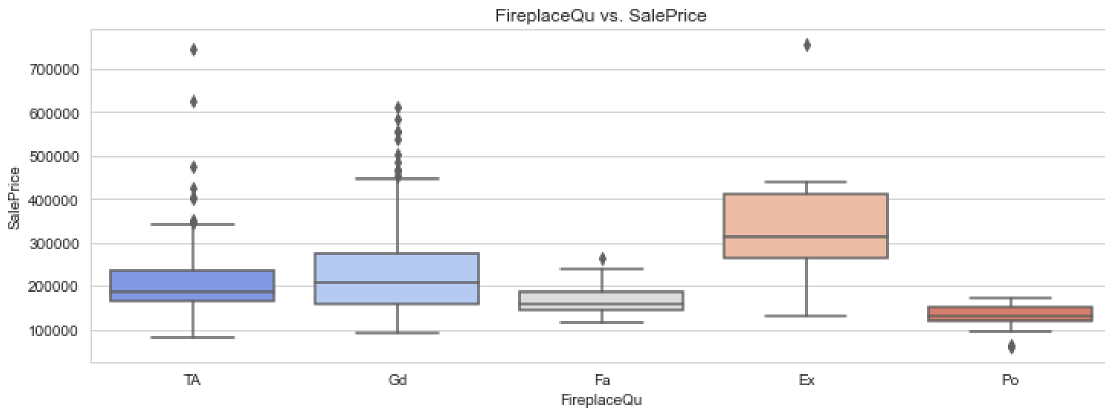

HeatingQC vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Ex
Gd
TA
Fa
Po
HeatingQC

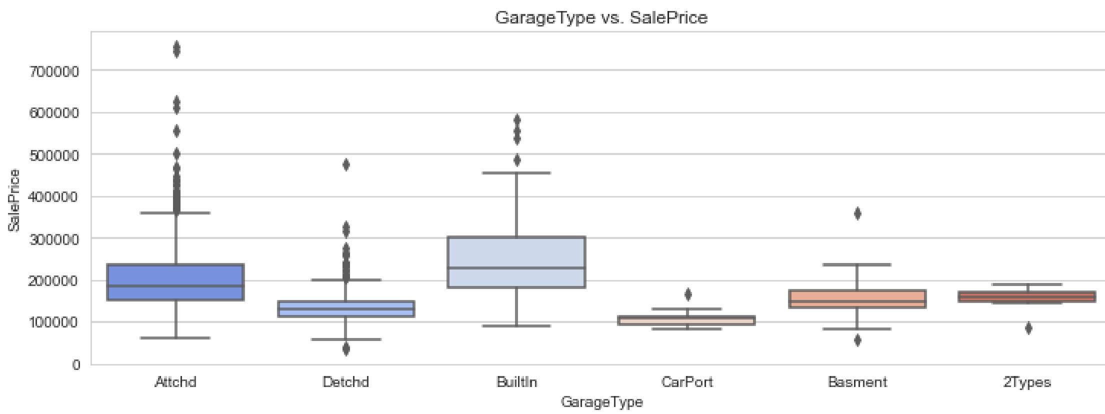

CentralAir vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Y
N
CentralAir

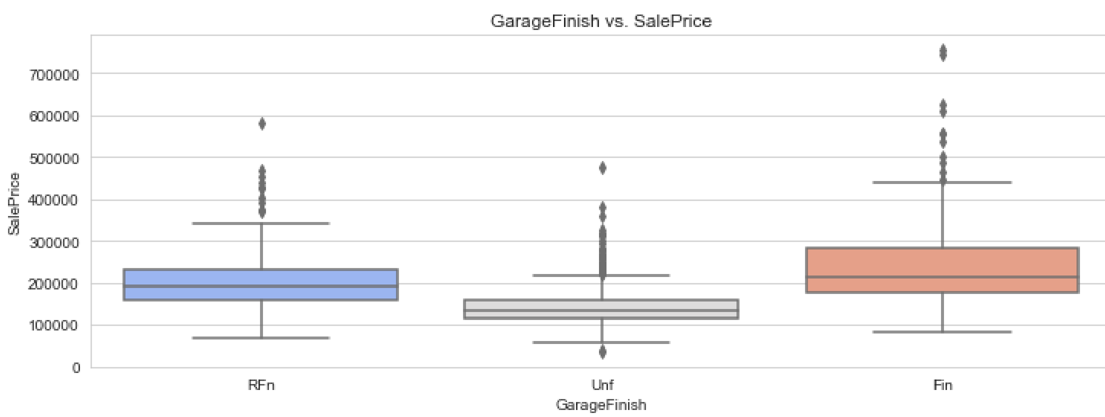

Electrical vs. SalePrice
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
SBrkr
FuseF
FuseA
FuseP
Mix
Electrical

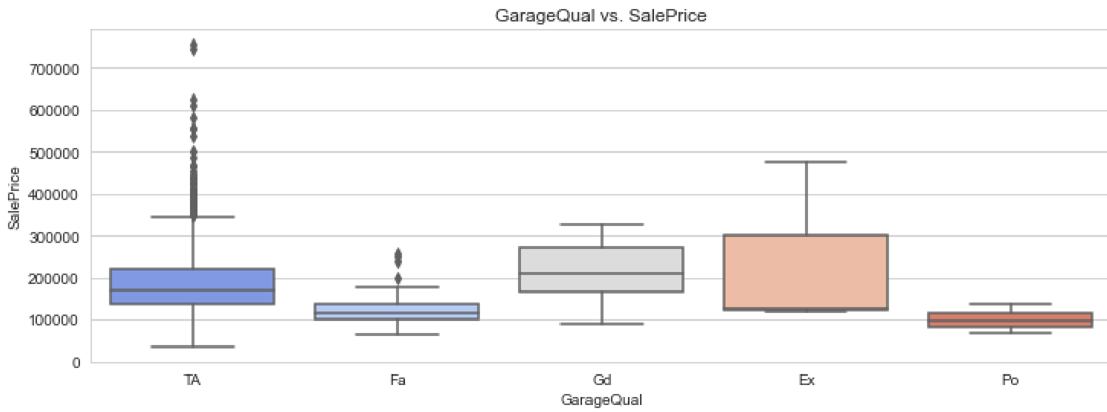

KitchenQual vs. SalePrice
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Gd
TA
Ex
Fa
KitchenQual

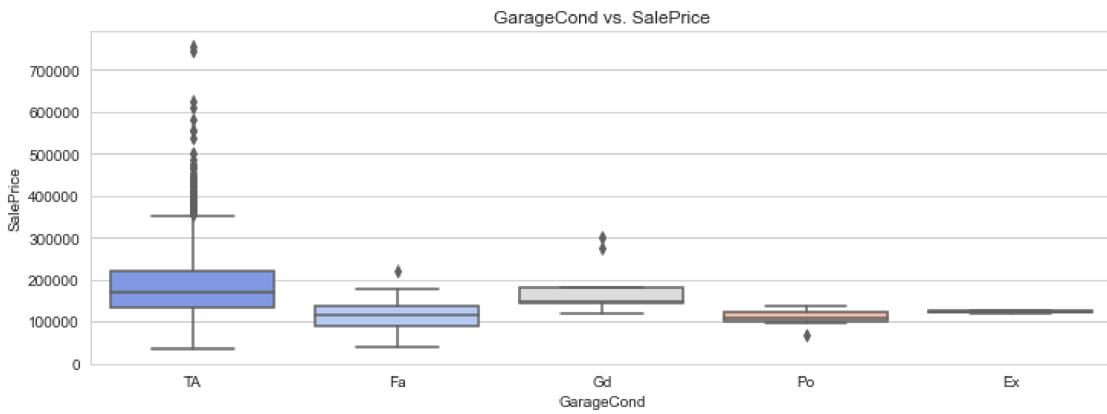

Functional vs. SalePrice
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Typ
Min1
Maj1
Min2
Mod
Maj2
Sev
Functional

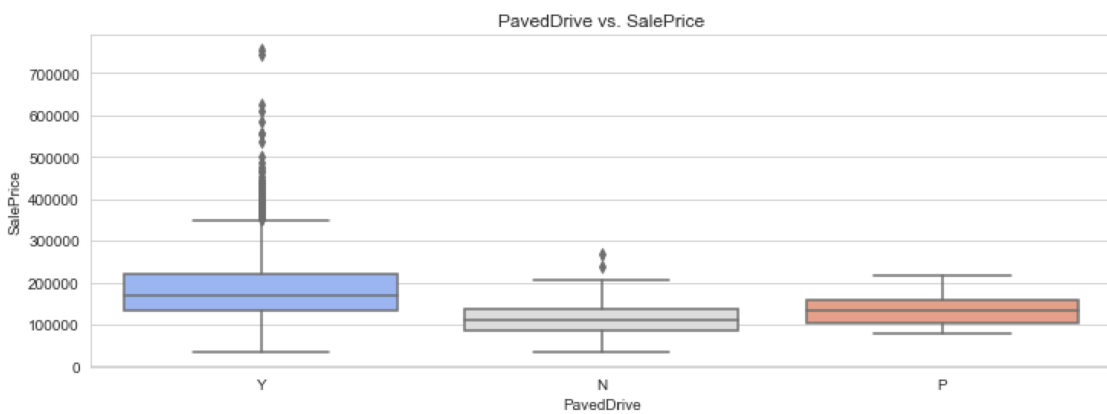

FireplaceQu vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
TA
Gd
Fa
Ex
Po
FireplaceQu

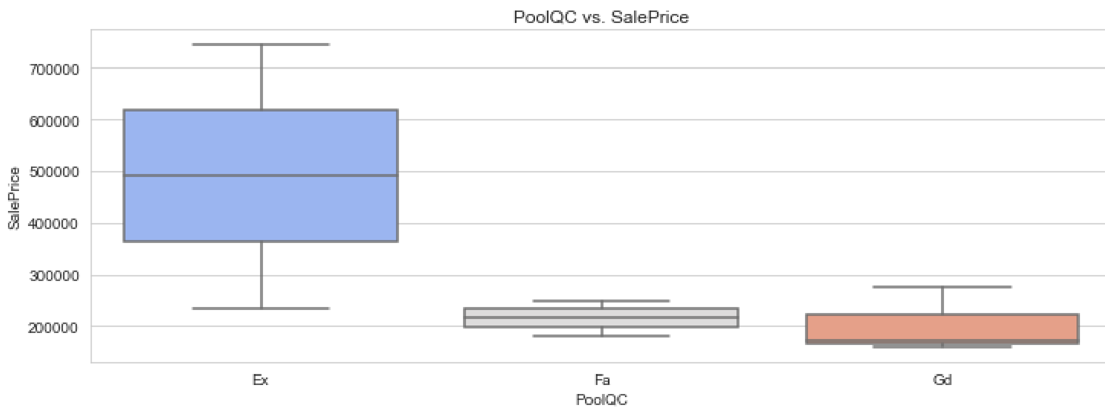

GarageType vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Attchd
Detchd
BuiltIn
CarPort
Basment
2Types
GarageType

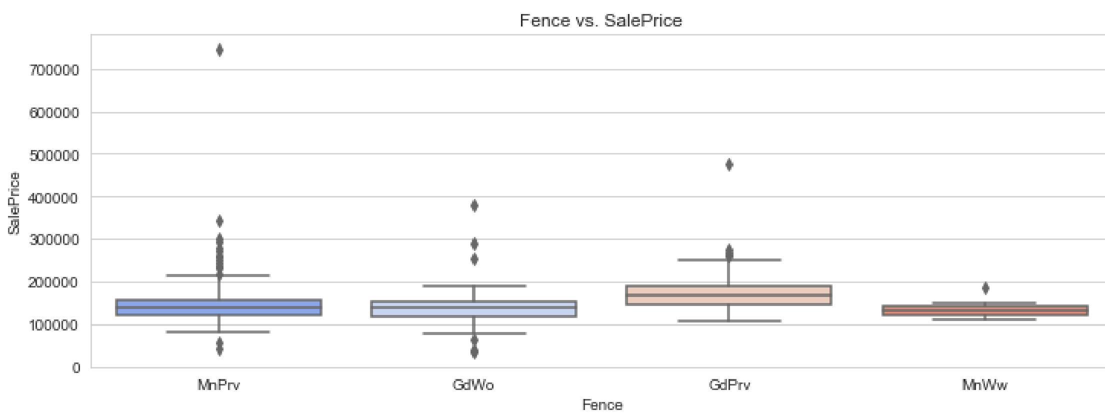

GarageFinish vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
RFn
Unf
Fin
GarageFinish

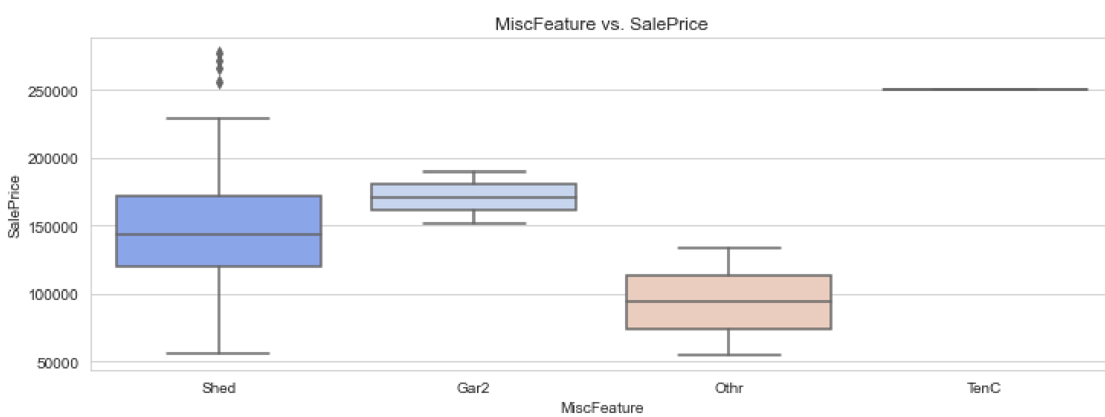

GarageQual vs. SalePrice
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
Fa
Gd
Ex
Po
GarageQual

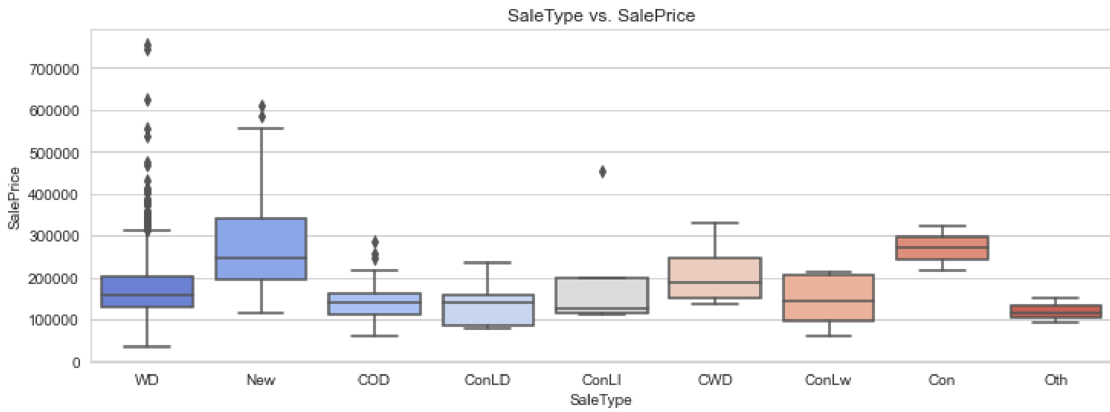

GarageCond vs. SalePrice
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
Fa
Gd
Po
Ex
GarageCond

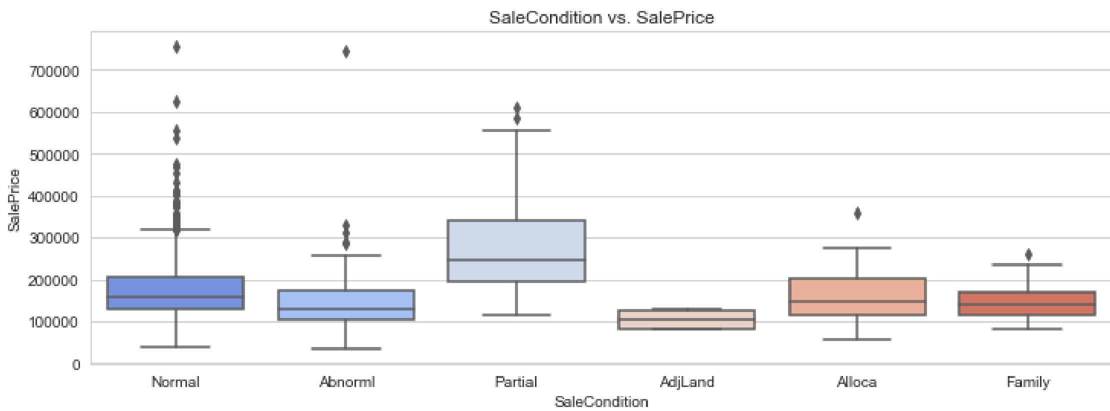

PavedDrive vs. SalePrice
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Y
N
P
PavedDrive


PoolQC vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
Ex
Fa
Gd
PoolQC


Fence vs. SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
MnPrv
GdWo
GdPrv
MnWw
Fence

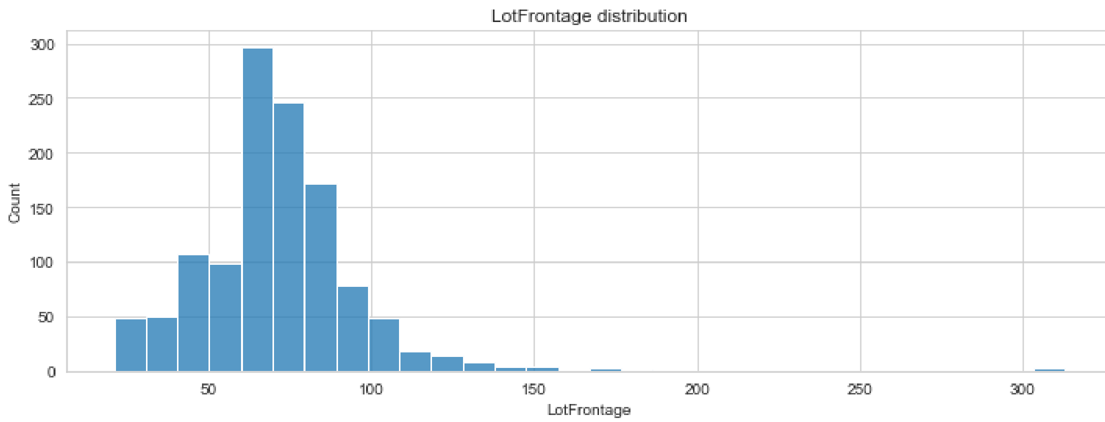

MiscFeature vs. SalePrice
SalePrice
250000
200000
150000
100000
50000
Shed
Gar2
Othr
TenC
MiscFeature

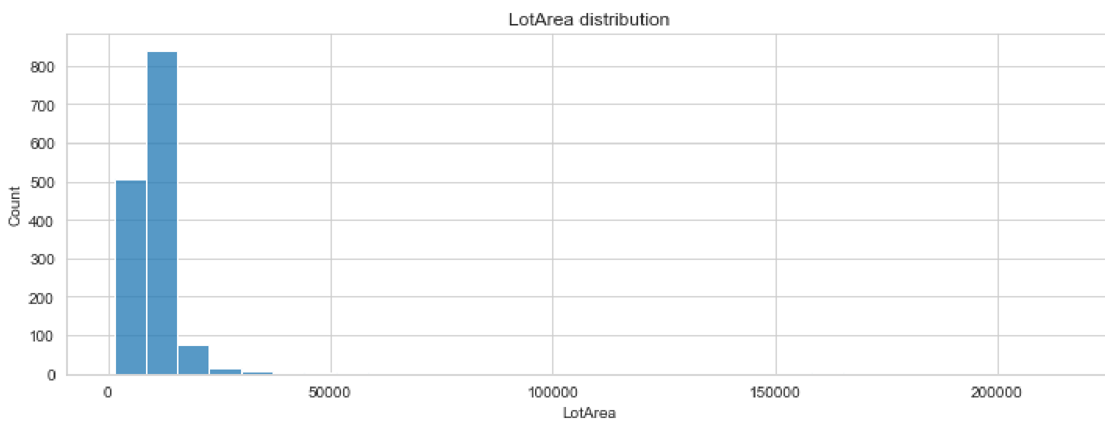


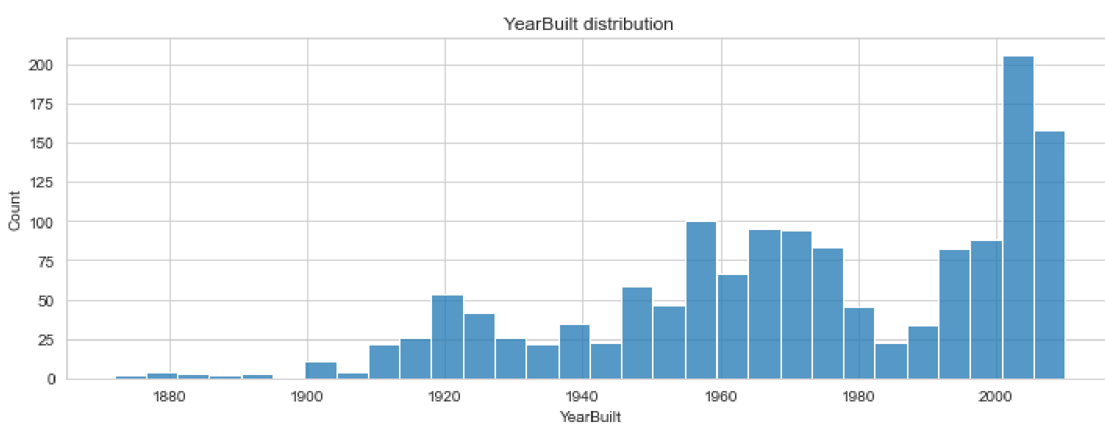


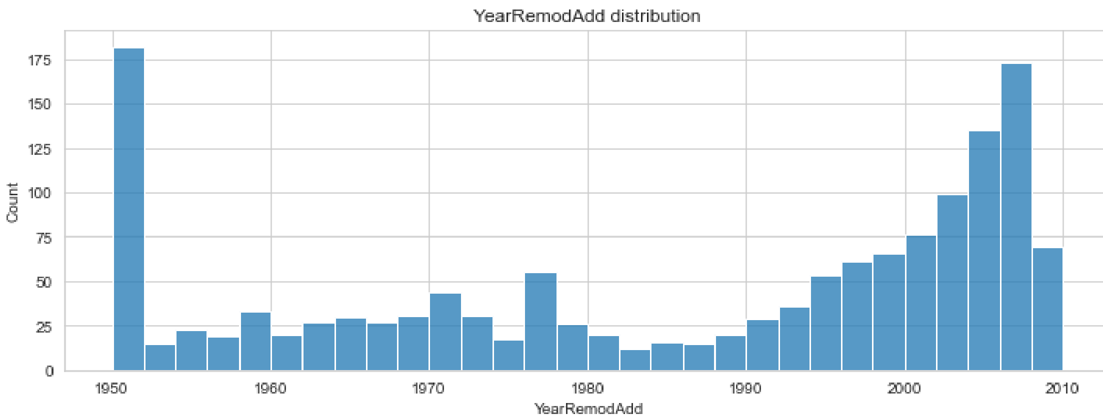
[14]:
```
for ax in train[numerical_cont_features]:
    plt.figure(figsize=(12, 4))
    sns.histplot(x=train[ax], bins=30)
    plt.title(f'{ax} distribution')
```

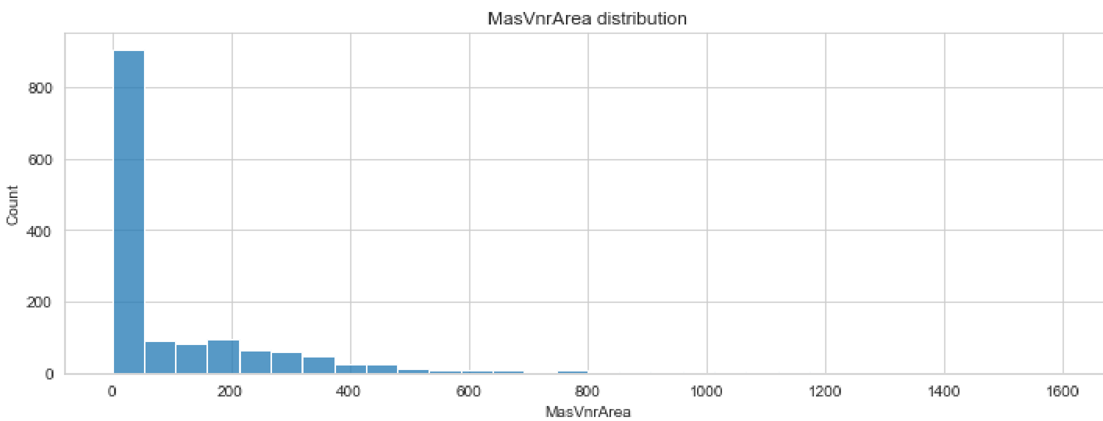
```
<ipython-input-14-a54a41e995e7>:2: RuntimeWarning: More than 20 figures have
been opened. Figures created through the pyplot interface
(`matplotlib.pyplot.figure`) are retained until explicitly closed and may
consume too much memory. (To control this warning, see the rcParam
`figure.max_open_warning`).
  plt.figure(figsize=(12, 4))
```

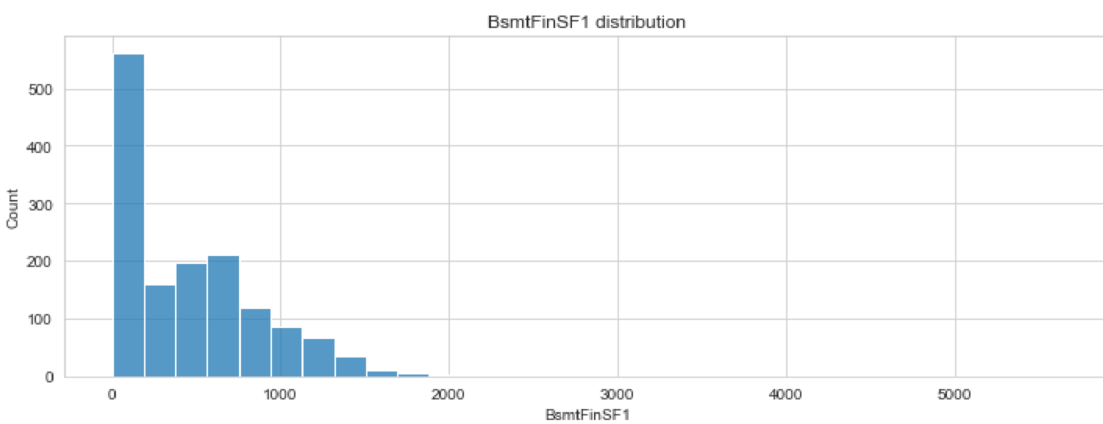

LotFrontage distribution
Count
LotFrontage
0
50
100
150
200
250
300

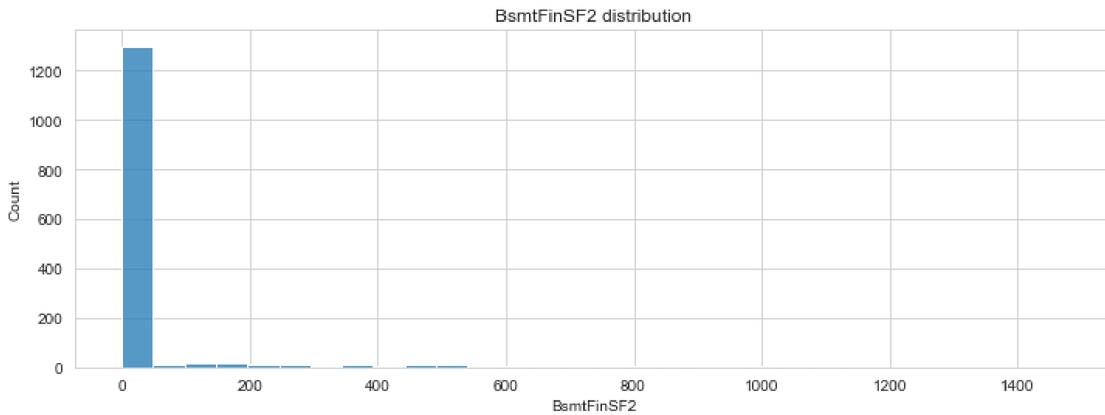

LotArea distribution
Count
LotArea
0
50000
100000
150000
200000
100
200
300
400
500
600
700
800

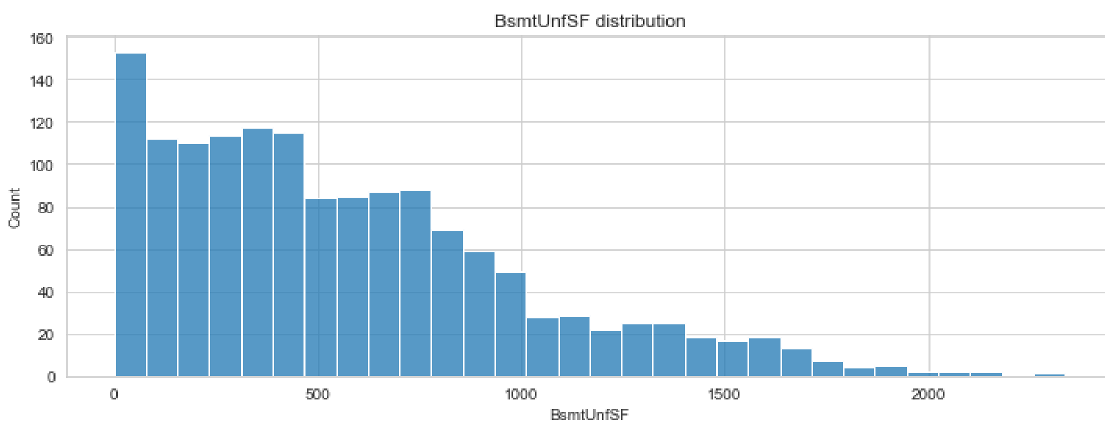

YearBuilt distribution
Count
YearBuilt
1880
1900
1920
1940
1960
1980
2000
0
25
50
75
100
125
150
175
200

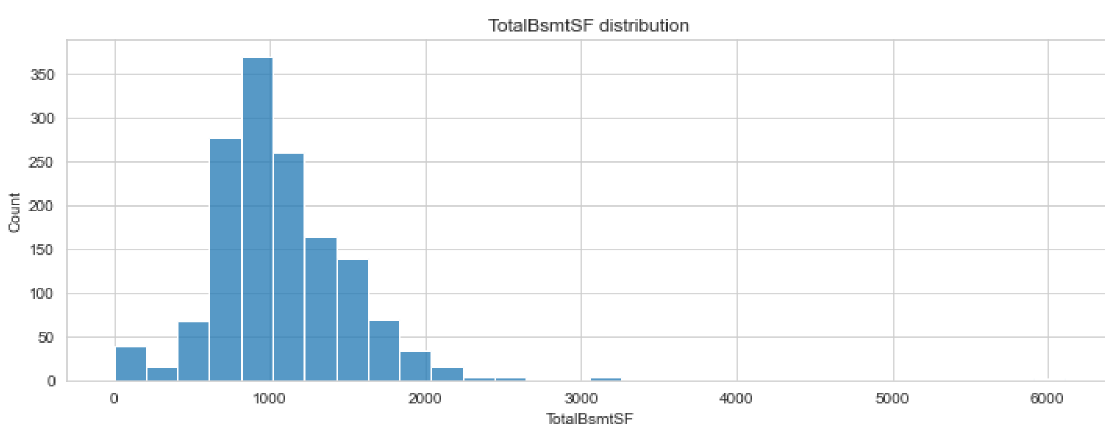

YearRemodAdd distribution
Count
175
150
125
100
75
50
25
0
1950
1960
1970
1980
1990
2000
2010
YearRemodAdd

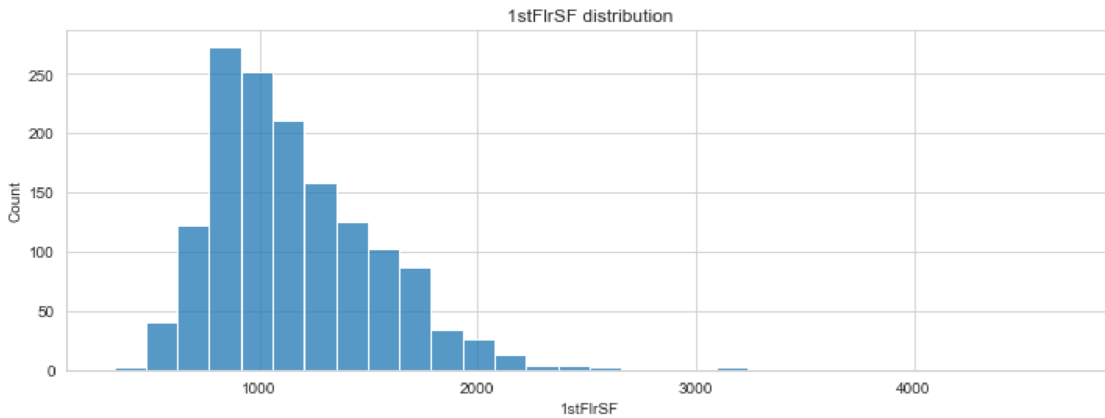

MasVnrArea distribution
Count
800
600
400
200
0
0
200
400
600
800
1000
1200
1400
1600
MasVnrArea

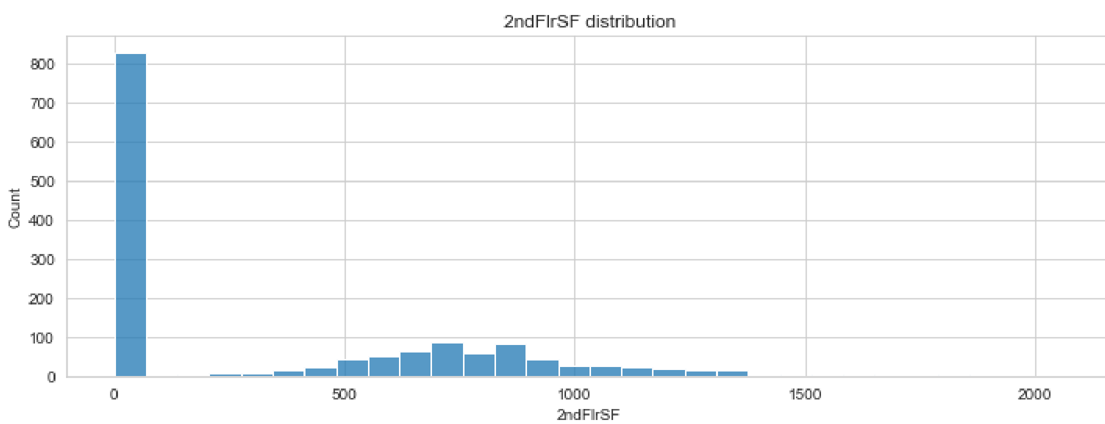

BsmtFinSF1 distribution
Count
500
400
300
200
100
0
0
1000
2000
3000
4000
5000
BsmtFinSF1

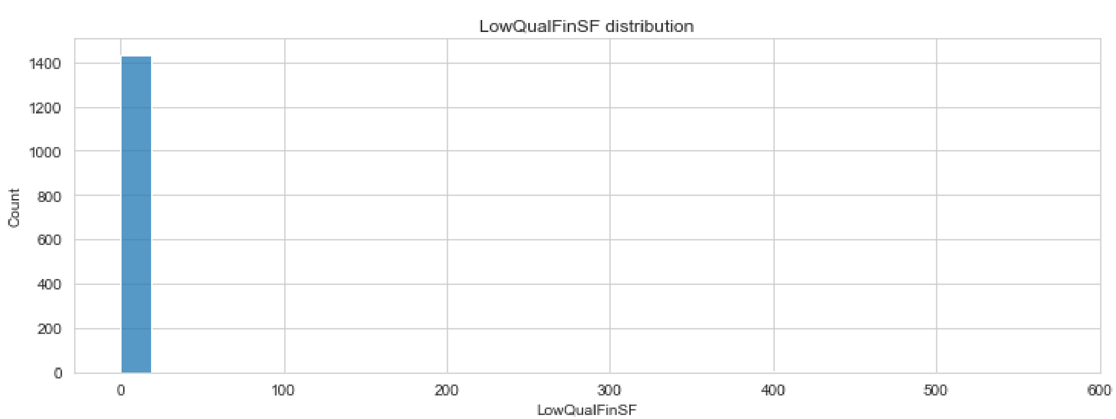

BsmtFinSF2 distribution
Count
0
200
400
600
800
1000
1200
0
200
400
600
800
1000
1200
1400
BsmtFinSF2

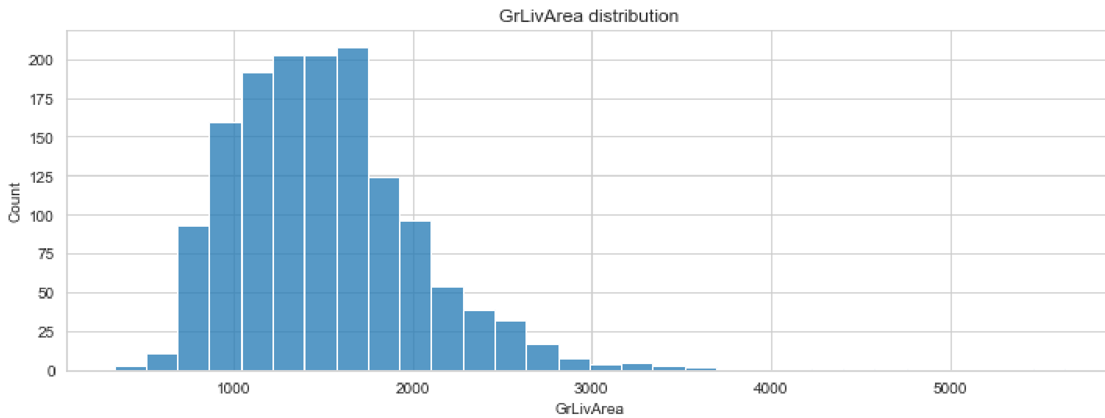

BsmtUnfSF distribution
Count
0
20
40
60
80
100
120
140
160
0
500
1000
1500
2000
BsmtUnfSF

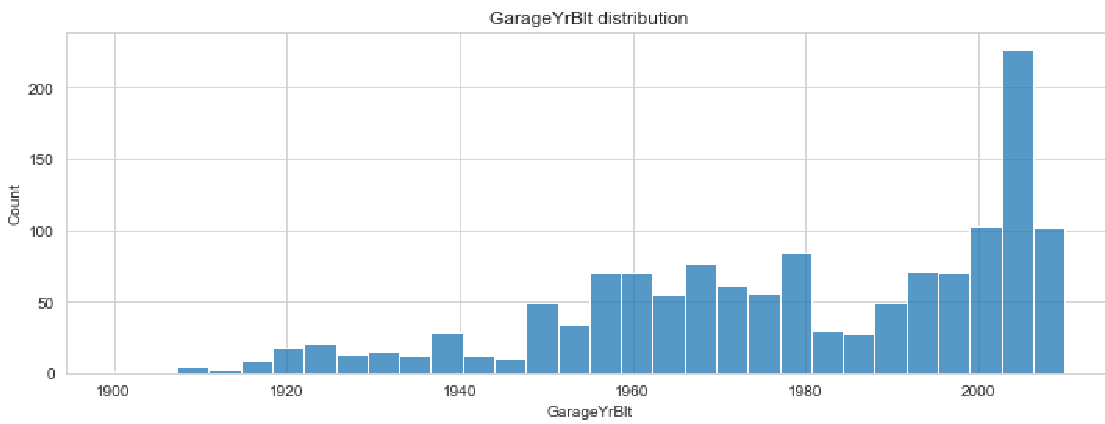

TotalBsmtSF distribution
Count
0
50
100
150
200
250
300
350
0
1000
2000
3000
4000
5000
6000
TotalBsmtSF

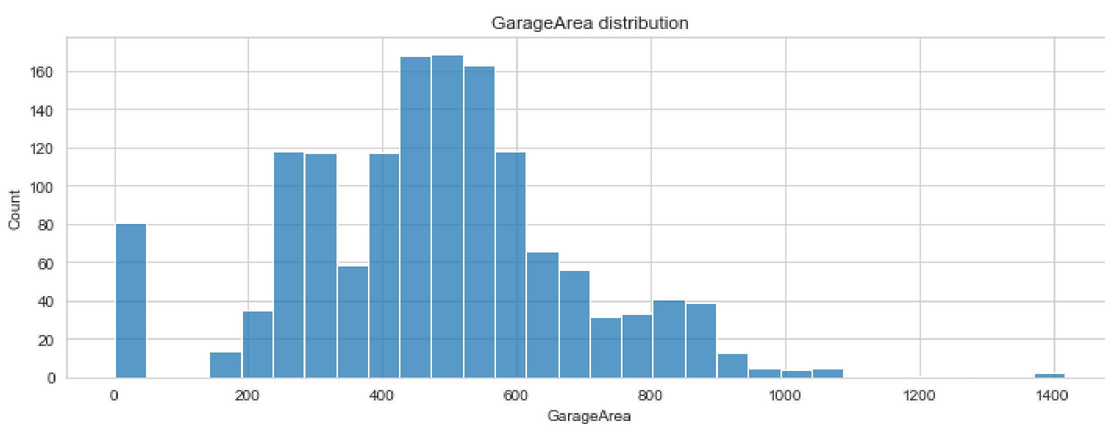

1stFlrSF distribution
Count
250
200
150
100
50
0
1000
2000
3000
4000
1stFlrSF

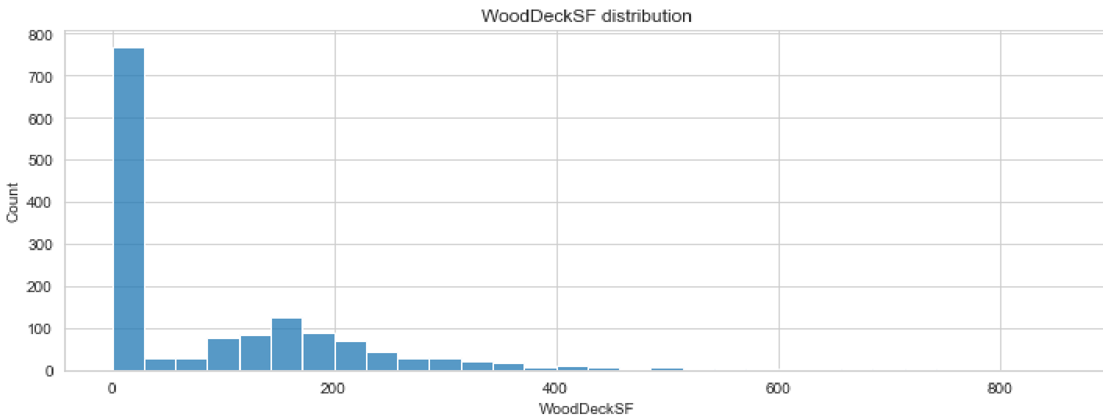

2ndFlrSF distribution
Count
800
700
600
500
400
300
200
100
0
0
500
1000
1500
2000
2ndFlrSF

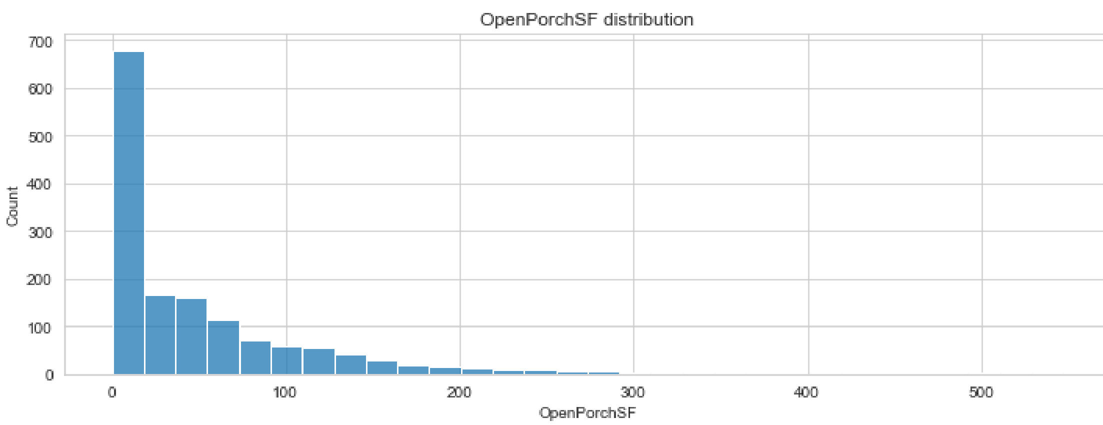

LowQualFinSF distribution
Count
1400
1200
1000
800
600
400
200
0
0
100
200
300
400
500
600
LowQualFinSF

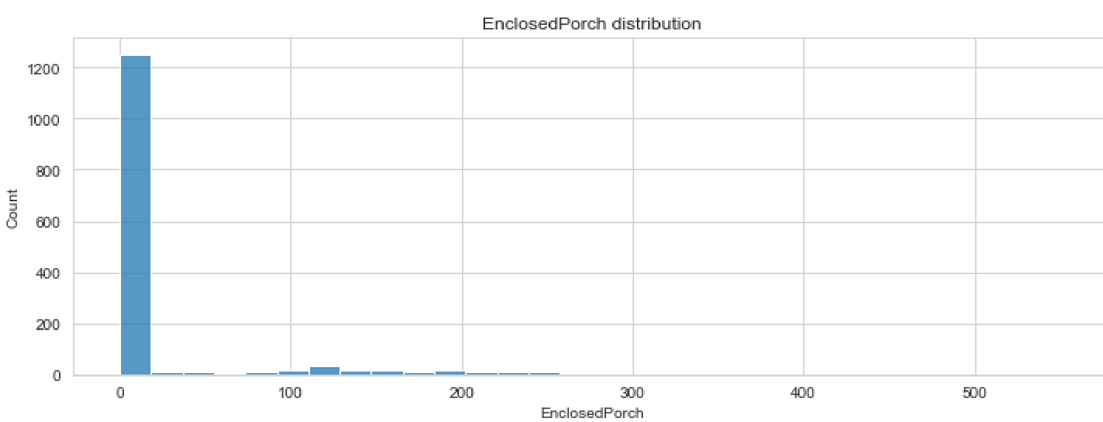

GrLivArea distribution
Count
GrLivArea
0
25
50
75
100
125
150
175
200
1000
2000
3000
4000
5000

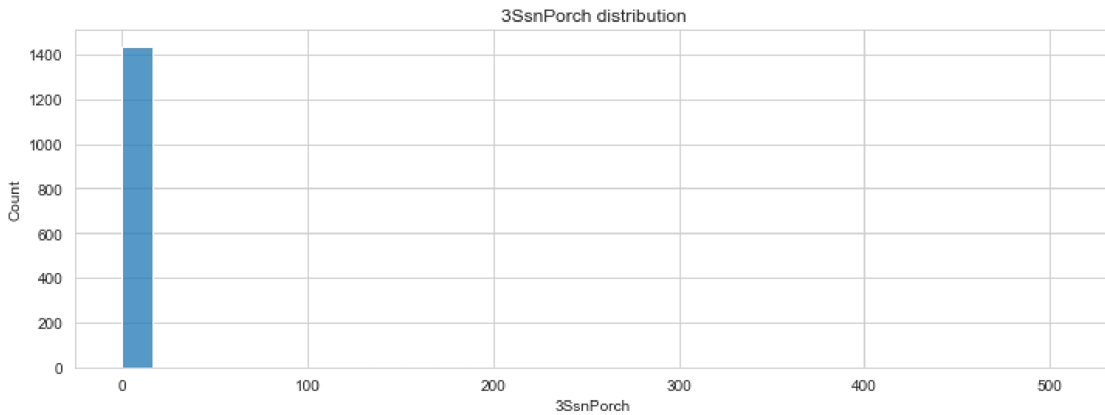

GarageYrBlt distribution
Count
GarageYrBlt
0
50
100
150
200
1900
1920
1940
1960
1980
2000

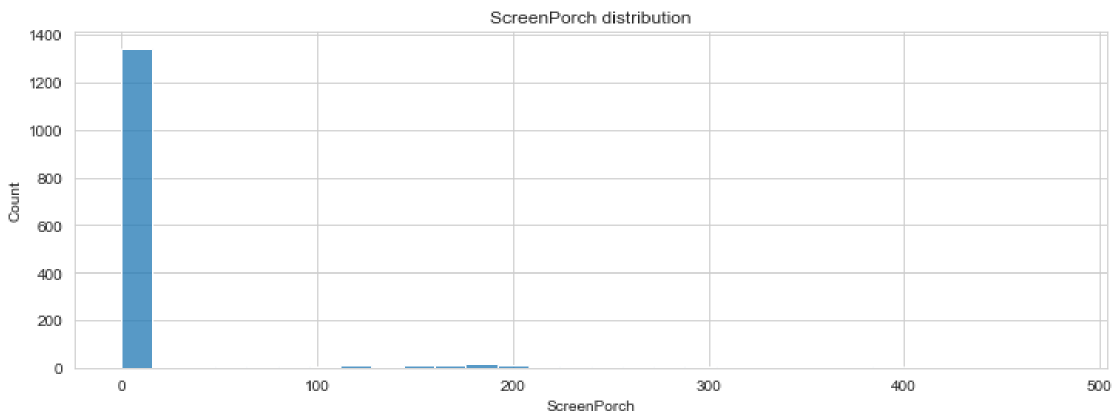

GarageArea distribution
Count
GarageArea
0
20
40
60
80
100
120
140
160
0
200
400
600
800
1000
1200
1400

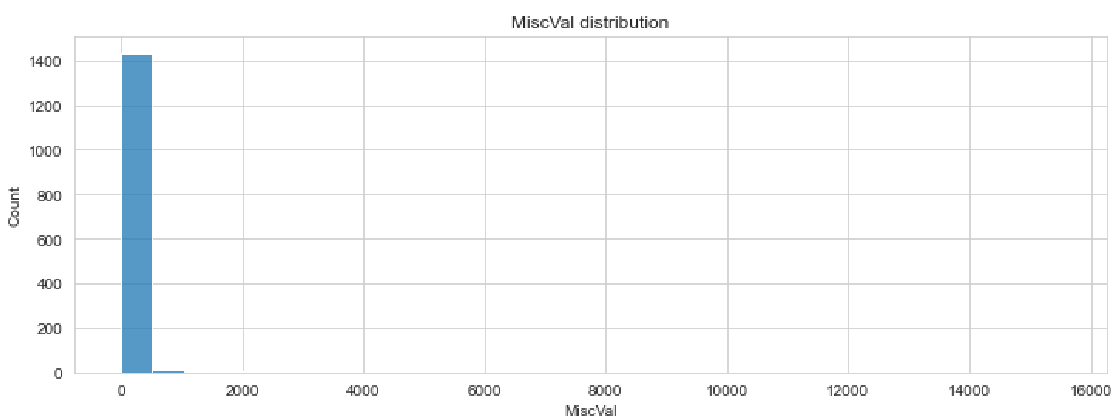

WoodDeckSF distribution
Count
0
100
200
300
400
500
600
700
800
0
200
400
600
800
WoodDeckSF


OpenPorchSF distribution
Count
0
100
200
300
400
500
600
700
0
100
200
300
400
500
OpenPorchSF


EnclosedPorch distribution
Count
0
200
400
600
800
1000
1200
0
100
200
300
400
500
EnclosedPorch

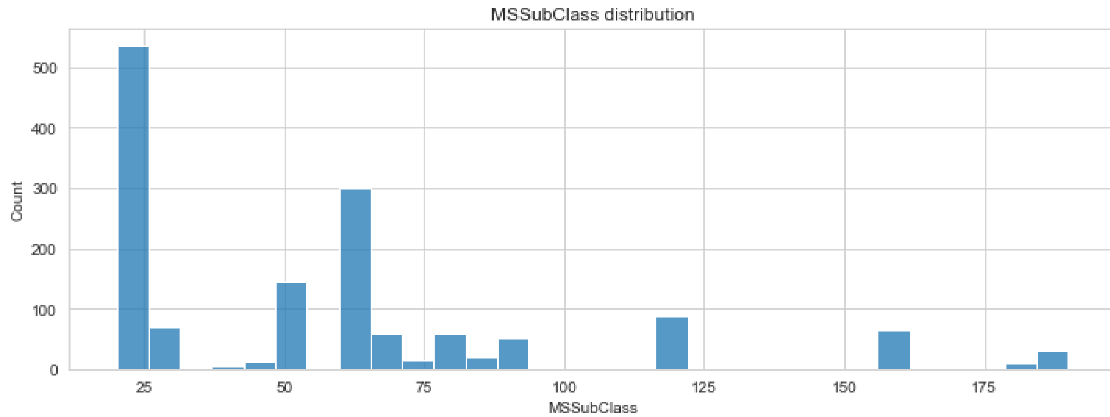


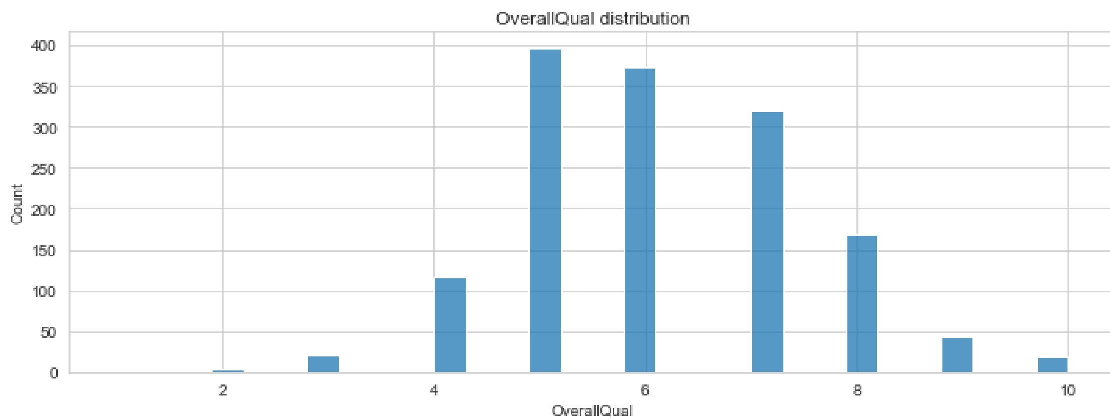


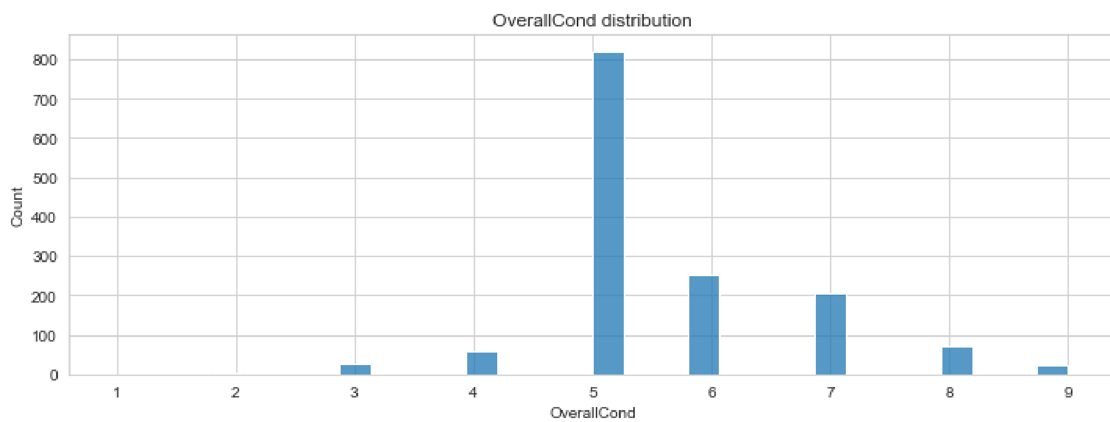


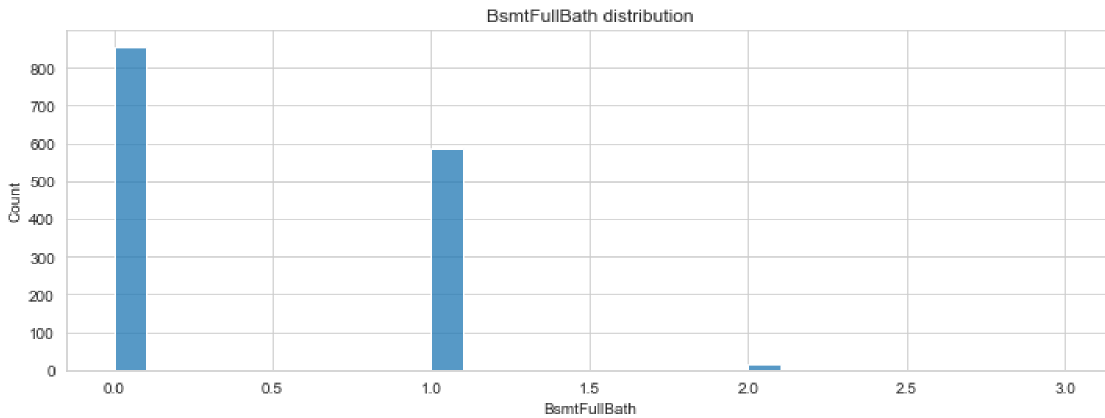
[15]:
```
for ax in train[numerical_cat_features]:
    plt.figure(figsize=(12, 4))
    sns.histplot(x=train[ax], bins=30)
```

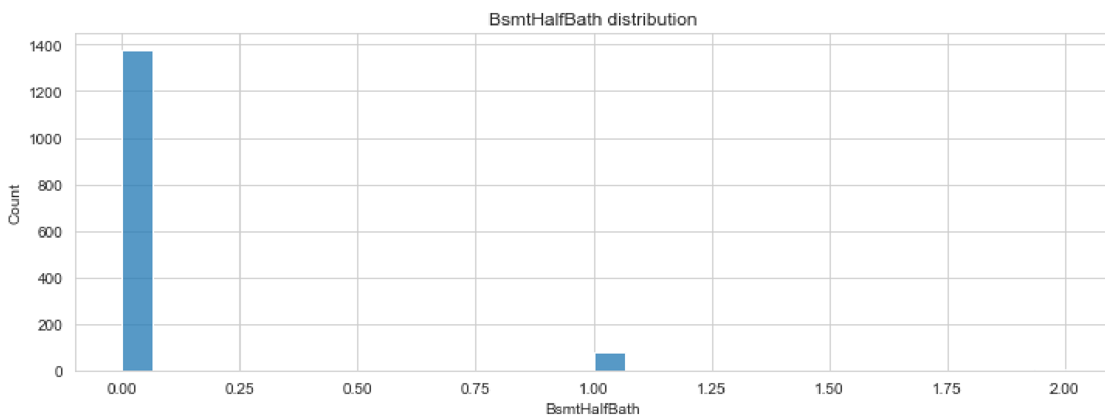
```
plt.title(f'{ax} distribution')
```

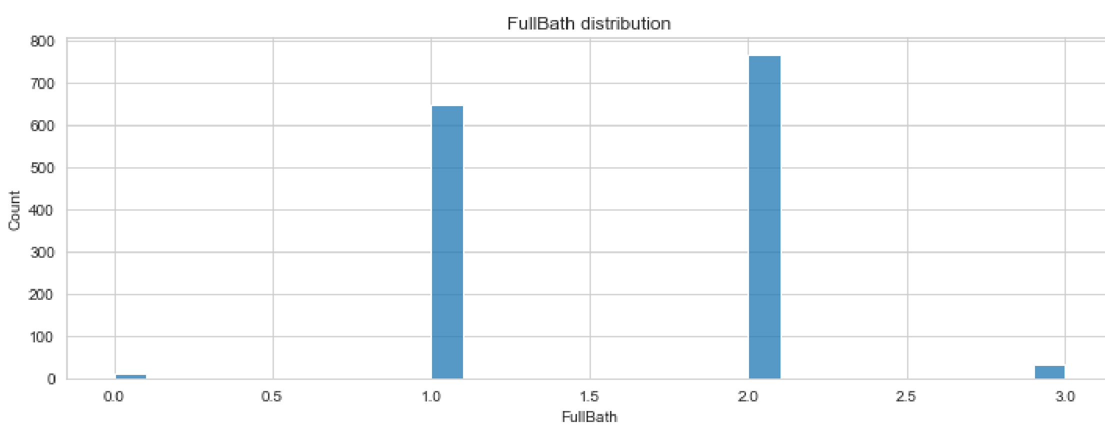


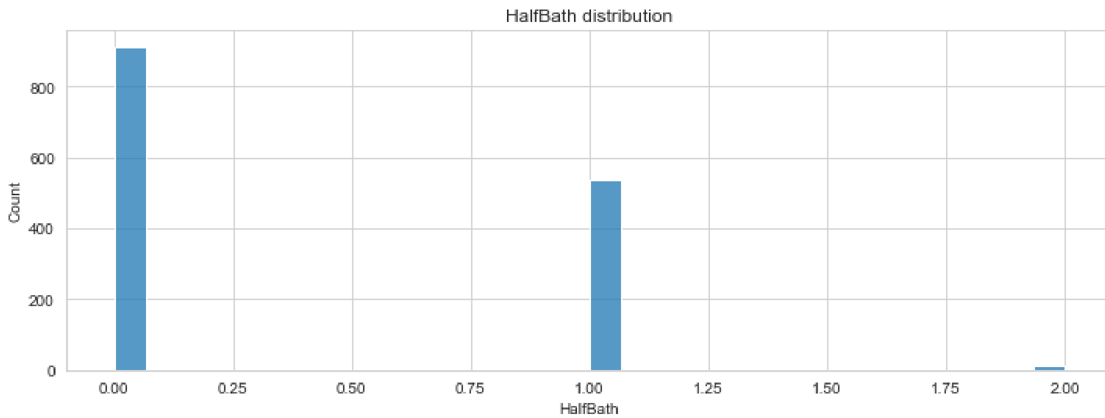


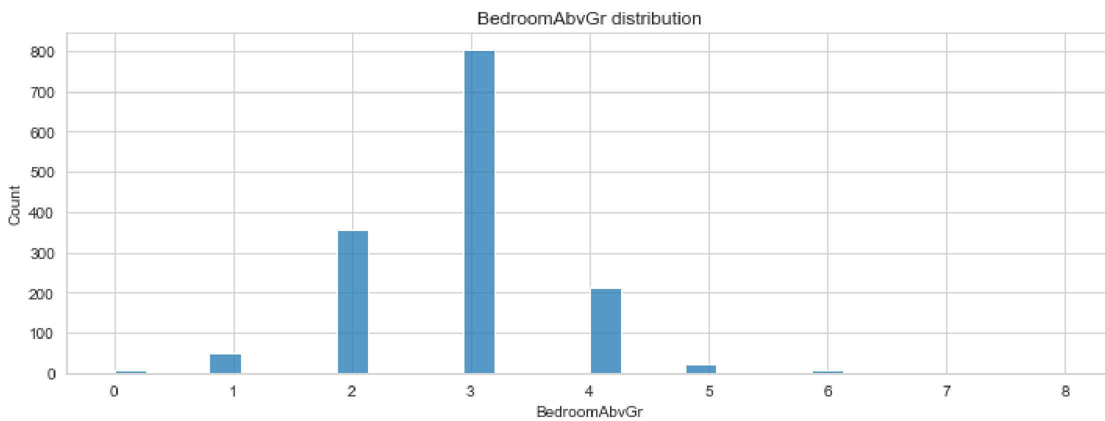


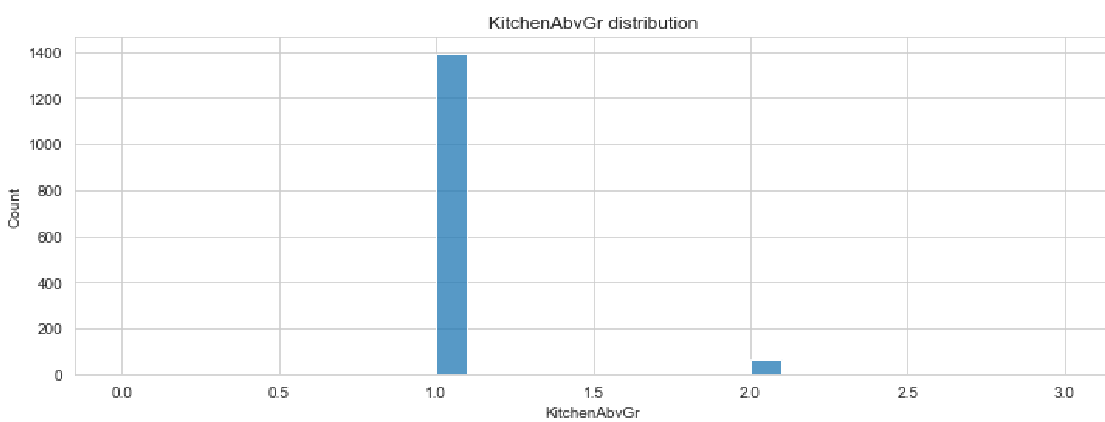

BsmtFullBath distribution
Count
800
700
600
500
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
BsmtFullBath

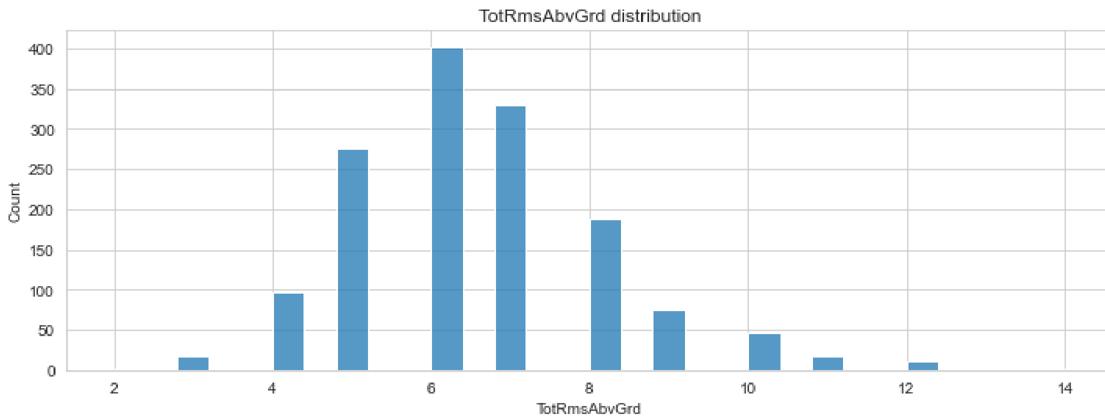

BsmtHalfBath distribution
Count
1400
1200
1000
800
600
400
200
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
BsmtHalfBath

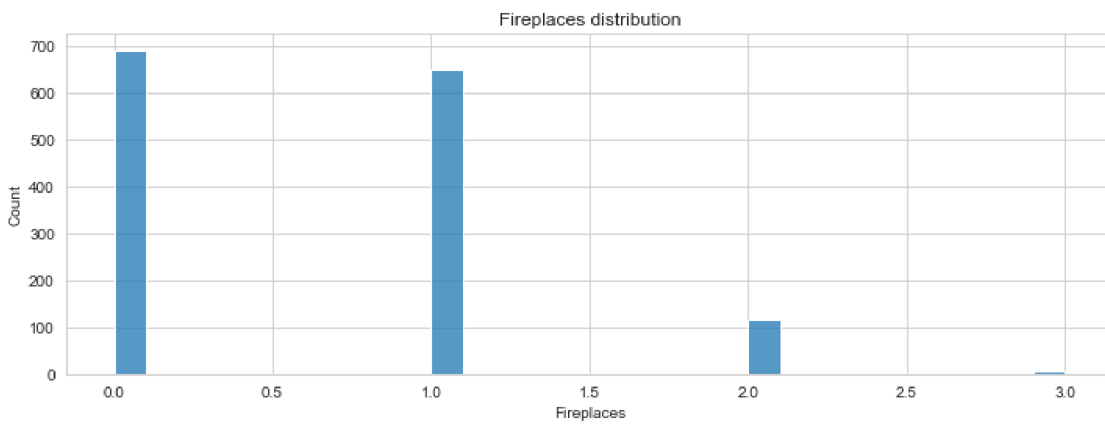

FullBath distribution
Count
800
700
600
500
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
FullBath

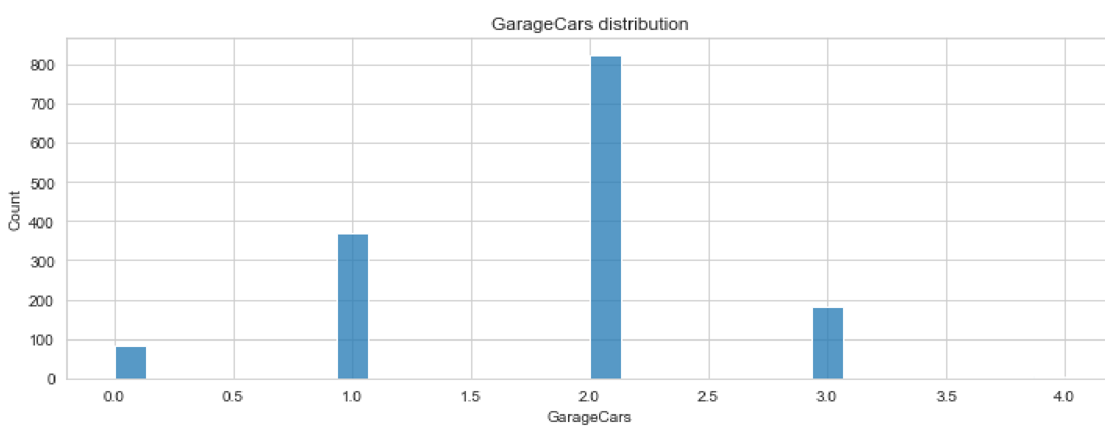

HalfBath distribution
Count
0
200
400
600
800
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
HalfBath

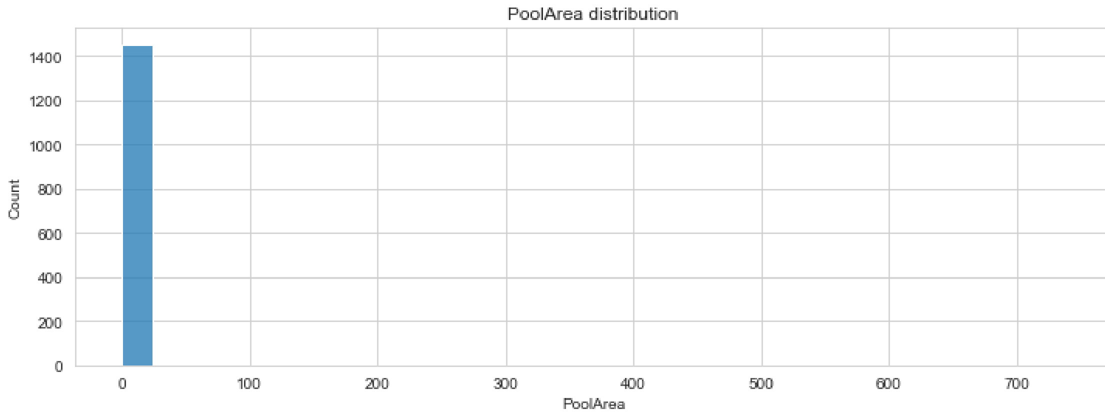

BedroomAbvGr distribution
Count
0
100
200
300
400
500
600
700
800
0
1
2
3
4
5
6
7
8
BedroomAbvGr

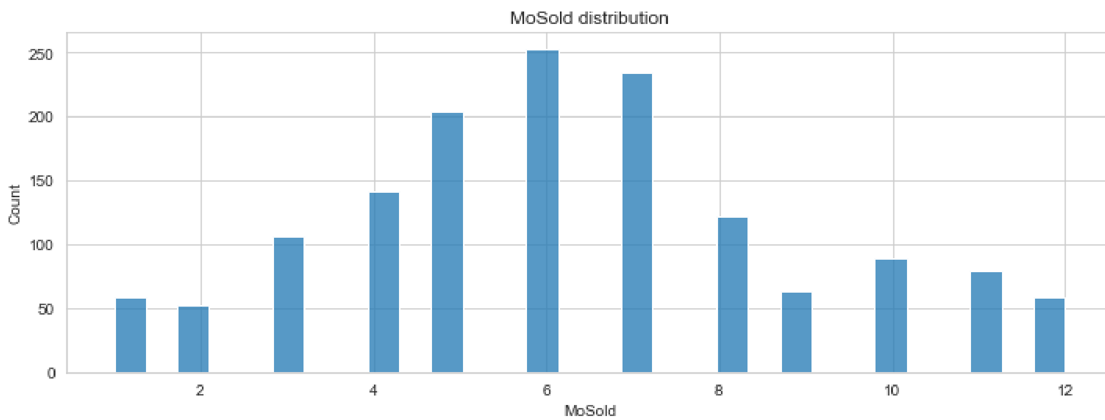

KitchenAbvGr distribution
Count
0
200
400
600
800
1000
1200
1400
0.0
0.5
1.0
1.5
2.0
2.5
3.0
KitchenAbvGr

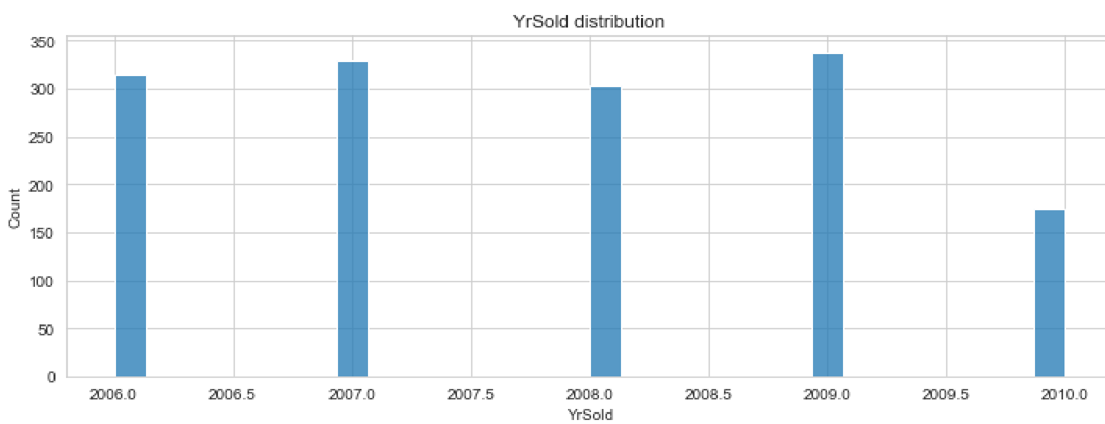

TotRmsAbvGrd distribution
Count
TotRmsAbvGrd
0
50
100
150
200
250
300
350
400
2
4
6
8
10
12
14


Fireplaces distribution
Count
Fireplaces
0
100
200
300
400
500
600
700
0.0
0.5
1.0
1.5
2.0
2.5
3.0


GarageCars distribution
Count
GarageCars
0
100
200
300
400
500
600
700
800
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0





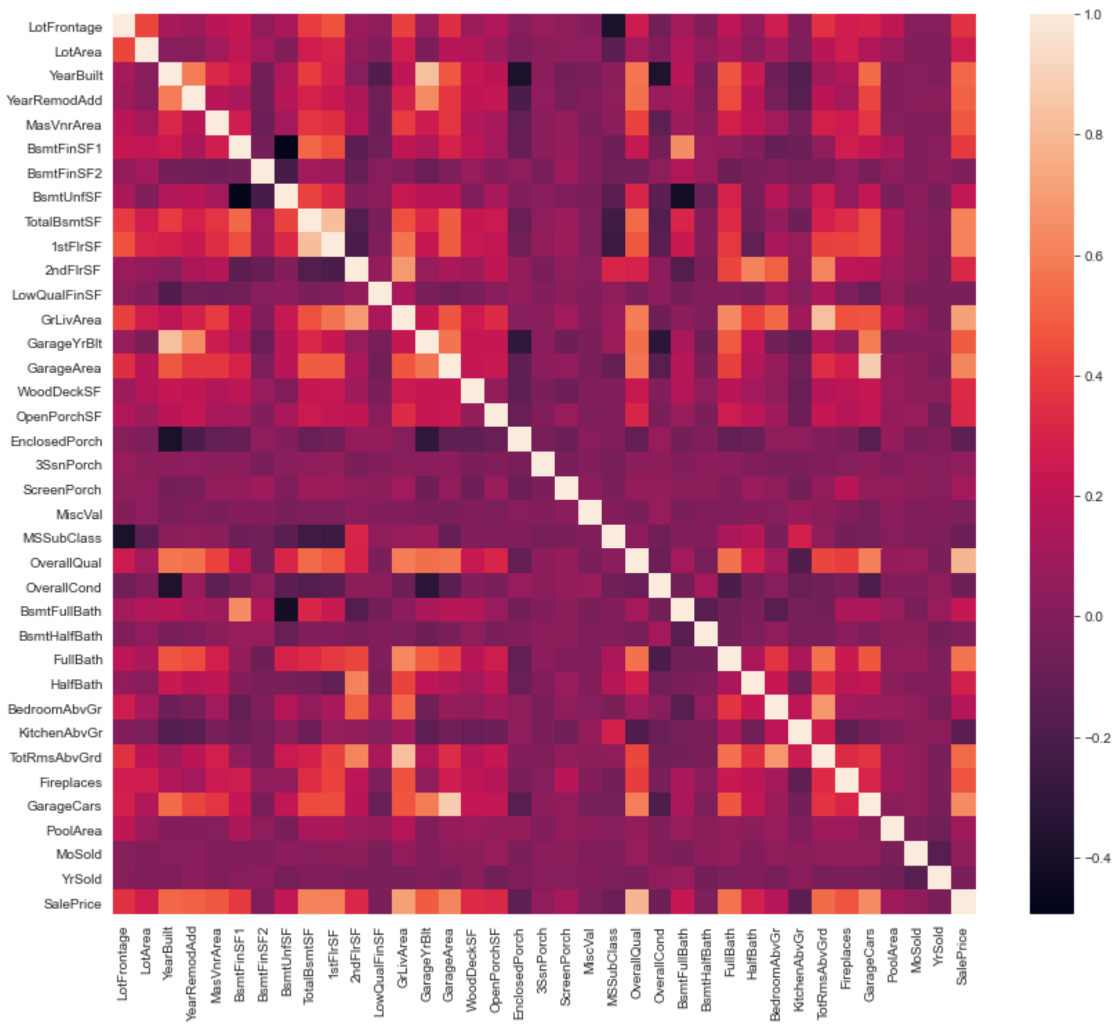


[16]:
```
# categorical feature distribution represented in boxplots
```

[17]:
```python
numerical_cols_w_price = numerical_cols + ['SalePrice']

plt.figure(figsize=(14, 12))
sns.heatmap(train[numerical_cols_w_price].corr())
```

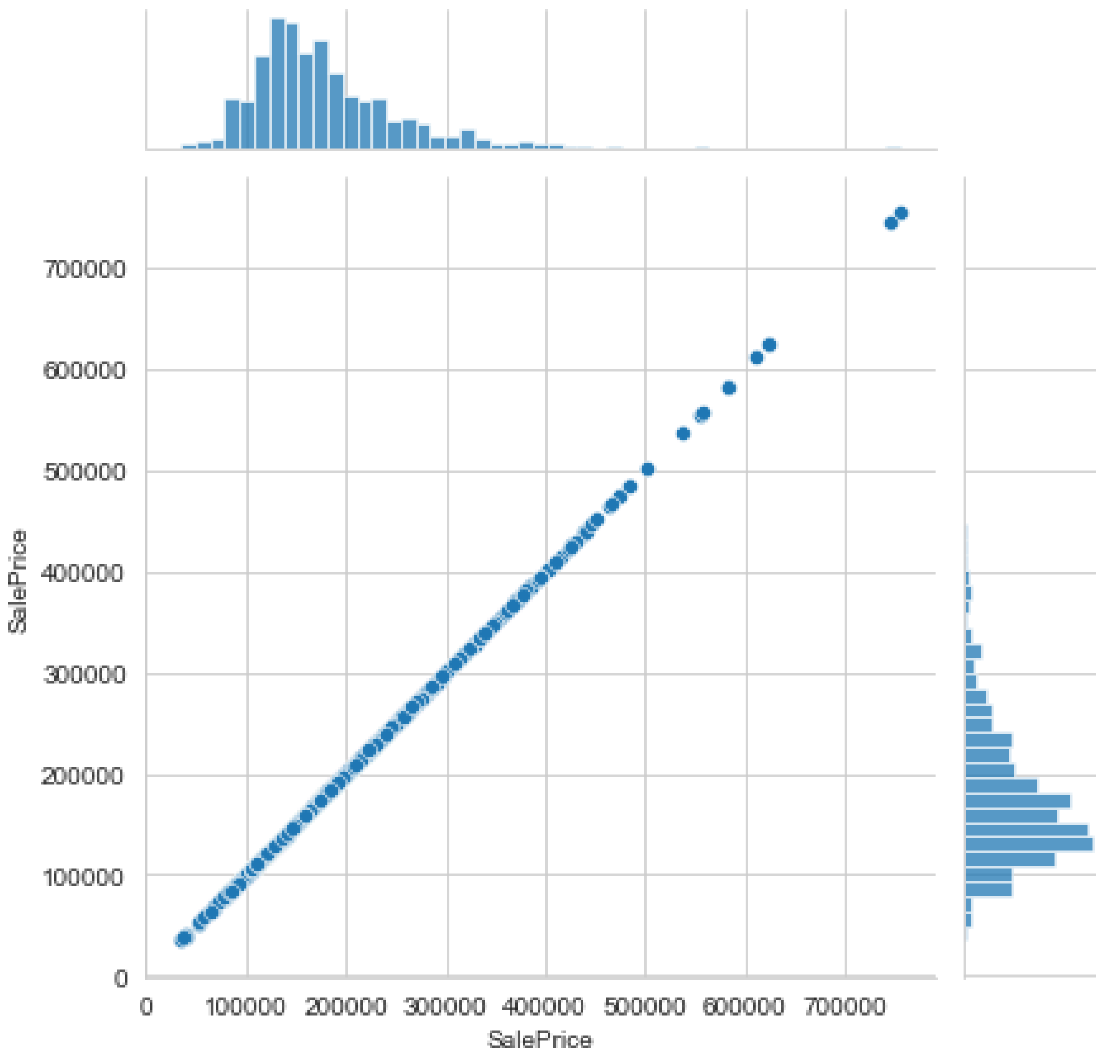
[17]: <AxesSubplot:>



3 Data Cleaning

[18]:
```python
sns.jointplot(x=train['SalePrice'], y=train['SalePrice'])
```

[18]: <seaborn.axisgrid.JointGrid at 0x1e1895ee9d0>



[19]:
```
# method to drop sale price outlier indexes (z score >= 3)

def drop_price_outliers(df):
    i = 0
    drop_indexes = []
    for value in df['SalePrice']:
        if abs((value - df['SalePrice'].mean()) / df['SalePrice'].std()) >= 3:
            drop_indexes.append(i)
        i += 1
    return df.drop(index=drop_indexes).reset_index()
```

[20]:
```
train = drop_price_outliers(train)
```

[21]:
```
# method to fill numerical cols with sparse data

def fill_sparse_num_cols(df):
    for col in df[numerical_cols]:
        if (df[col].isnull().sum() / len(df[col]) * 100) > 0:
            df[col] = df[col].fillna(value=round(df[col].mean(), 0))
```

[22]:
```
fill_sparse_num_cols(train)
fill_sparse_num_cols(test)
```

[23]:
```
# method to impute values to the features mean with a z score >= 3

pd.options.mode.chained_assignment = None

def impute_num_outliers(df):
    z_score_dic = {}
    i = 1
    for col in df[numerical_cont_features]:
        for value in df[col]:
            if abs((value - df[col].mean()) / df[col].std()) >= 3 and \
            col not in z_score_dic.keys():
                z_score_dic[col] = [i - 1]
            elif abs((value - df[col].mean()) / df[col].std()) >= 3 and \
            col in z_score_dic.keys():
                z_score_dic[col].append(i - 1)
            i += 1
            if i > len(df[col]):
                i = 1
    for key in z_score_dic.keys():
        for value in z_score_dic[key]:
            df[key][value] = round(df[key].mean(), 2)
    return df
```

[24]:
```
train = impute_num_outliers(train)
test = impute_num_outliers(test)
```

[25]:
```
# method to drop cols with > 45% missing data

def drop_sparse_cat_cols(df):
    drop_cols = []
    for col in df[categorical_features]:
        if ((df[col].isnull().sum() / len(df[col])) * 100) > 45:
            drop_cols.append(col)
    return df.drop(labels=drop_cols, axis=1)
```

[26]:
```
train = drop_sparse_cat_cols(train)
test = drop_sparse_cat_cols(test)
```

[27]:
```python
# method to fill missing categorical features with missing object

updated_cat_features = [col for col in train.drop('SalePrice', axis=1) if \
                        train[col].dtype in ['object']]

def fill_missing_cat_cols(df):
    for col in df[updated_cat_features]:
        df[col].fillna(value='Missing', inplace=True)
    return df
```

[28]:
```python
train = fill_missing_cat_cols(train)
test = fill_missing_cat_cols(test)
```

4 Feature Engineering / Scaling

[29]:
```python
# method to transform categorical data to discrete vars via label encoder

def labeling(df):
    for col in df[updated_cat_features]:
        le = LabelEncoder()
        temp = le.fit_transform(df[col])
        df[f'{col}_labels'] = temp
        df = df.drop(labels=col, axis=1)
    if 'index' in df.columns:
        df = df.drop(labels='index', axis=1)
    return df
```

[30]:
```python
train = labeling(train)
test = labeling(test)
```

[31]:
```python
sns.lineplot(x=train['YrSold'], y=train['SalePrice'])
```

[31]: <AxesSubplot:xlabel='YrSold', ylabel='SalePrice'>

[32]:
```
temporal_features = [col for col in train if 'Year' in col or 'Yr' in col]
```

[33]:
```
# method to reassign temporal attributes with respect to year sold (dropping␣
↪year sold)

def reassign_temporals(df):
    for col in df[temporal_features]:
        if col != 'YrSold':
            df[col] = df['YrSold'] - df[col]
    return df.drop(labels='YrSold', axis=1)
```

[34]:
```
train = reassign_temporals(train)
test = reassign_temporals(test)
```

[35]:
```
# method to normalize data via min max scaler w/ target var

train_cols = [col for col in train if col != 'Id']
test_cols = [col for col in test if col != 'Id']

def normalize(df):
    if len(df.columns) == 75:
        cols = train_cols
    else:
        cols = test_cols
    scaler = MinMaxScaler()
```

```
    return pd.DataFrame(data=scaler.fit_transform(df[cols]), columns=cols)
```

[36]:
```
# train_cols = [col for col in train if col not in ['Id', 'SalePrice']]
# test_cols = [col for col in test if col != 'Id']

# def normalize(df):
#     scaler = MinMaxScaler()
#     if len(df.columns) == 75:
#         df1 = pd.DataFrame(data=scaler.fit_transform(df[train_cols]),␣
↪columns=train_cols)
#         df1['SalePrice'] = df['SalePrice']
#         return df1
#     else:
#         return pd.DataFrame(data=scaler.fit_transform(df[test_cols]),␣
↪columns=test_cols)
```

[37]:
```
train_normal = normalize(train)
test_normal = normalize(test)
```

[38]:
```
# method to standardize data via standard scaler w/ target var

def standardize(df):
    if len(df.columns) == 75:
        cols = train_cols
    else:
        cols = test_cols
    scaler = StandardScaler()
    return pd.DataFrame(data=scaler.fit_transform(df[cols]), columns=cols)
```

[39]:
```
# def standardize(df):
#     scaler = StandardScaler()
#     if len(df.columns) == 75:
#         df1 = pd.DataFrame(data=scaler.fit_transform(df[train_cols]),␣
↪columns=train_cols)
#         df1['SalePrice'] = df['SalePrice']
#         return df1
#     else:
#         return pd.DataFrame(data=scaler.fit_transform(df[test_cols]),␣
↪columns=test_cols)
```

[40]:
```
train_standard = standardize(train)
test_standard = standardize(test)
```

5 Feature Selection

[41]:
```
# feature selection on normalized data

X_train_normal = train_normal.drop('SalePrice', axis=1)
y_train_normal = train_normal['SalePrice']

feature_selection_normal = SelectFromModel(Lasso(alpha=0.005, random_state=0))
feature_selection_normal.fit(X_train_normal, y_train_normal)
```

[41]:
```
SelectFromModel(estimator=Lasso(alpha=0.005, random_state=0))
```

[42]:
```
selected_feat_normal = X_train_normal.columns[(feature_selection_normal.
 ↪get_support())]
X_train_normal_selected = X_train_normal[selected_feat_normal]
```

[43]:
```
# feature selection on standardized data

X_train_standard = train_standard.drop('SalePrice', axis=1)
y_train_standard = train_standard['SalePrice']

feature_selection_standard = SelectFromModel(Lasso(alpha=0.005, random_state=0))
feature_selection_standard.fit(X_train_standard, y_train_standard)
```

[43]:
```
SelectFromModel(estimator=Lasso(alpha=0.005, random_state=0))
```

[44]:
```
selected_feat_standard = X_train_standard.columns[(feature_selection_standard.
 ↪get_support())]
X_train_standard_selected = X_train_standard[selected_feat_standard]
```

[45]:
```
count = len(X_train_normal_selected.columns)
for col in X_train_normal_selected.columns:
    if col in X_train_standard_selected.columns:
        count -= 1
print(f'{count} -> same selected cols')
```

```
0 -> same selected cols
```

6 Modelling Neural Nets

[46]:
```
# train test split overriding normal and standard scaled vars

X_train_normal, X_test_normal, y_train_normal, y_test_normal = \
train_test_split(X_train_normal_selected, y_train_normal, test_size=0.2,␣
 ↪random_state=0)
```

```
X_train_standard, X_test_standard, y_train_standard, y_test_standard = \
train_test_split(X_train_standard_selected, y_train_standard, test_size=0.2,␣
↪random_state=0)
```

[47]:
```
# configuring neural net w/ adam optimizer for normalized data

# 33 33 1 best

model_normal = Sequential()

model_normal.add(Dense(33, activation='relu'))
model_normal.add(Dropout(0.2))

model_normal.add(Dense(33, activation='relu'))
model_normal.add(Dropout(0.2))

model_normal.add(Dense(1))

model_normal.compile(optimizer='adam', loss='mse')
```

[48]:
```
model_normal.fit(x=X_train_normal, y=y_train_normal,␣
↪validation_data=(X_test_normal, \
                                                                      ␣
↪y_test_normal), \
          batch_size=128, epochs=100)
```

```
Epoch 1/100
9/9 [==============================] - 1s 19ms/step - loss: 0.1353 - val_loss:
0.0523
Epoch 2/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0740 - val_loss:
0.0342
Epoch 3/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0547 - val_loss:
0.0251
Epoch 4/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0456 - val_loss:
0.0213
Epoch 5/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0374 - val_loss:
0.0160
Epoch 6/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0339 - val_loss:
0.0136
Epoch 7/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0292 - val_loss:
0.0123
```

```
Epoch 8/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0283 - val_loss:
0.0100
Epoch 9/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0243 - val_loss:
0.0106
Epoch 10/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0222 - val_loss:
0.0104
Epoch 11/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0228 - val_loss:
0.0107
Epoch 12/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0201 - val_loss:
0.0102
Epoch 13/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0205 - val_loss:
0.0094
Epoch 14/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0189 - val_loss:
0.0097
Epoch 15/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0179 - val_loss:
0.0098
Epoch 16/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0181 - val_loss:
0.0091
Epoch 17/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0186 - val_loss:
0.0093
Epoch 18/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0168 - val_loss:
0.0095
Epoch 19/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0160 - val_loss:
0.0083
Epoch 20/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0159 - val_loss:
0.0100
Epoch 21/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0151 - val_loss:
0.0095
Epoch 22/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0159 - val_loss:
0.0079
Epoch 23/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0156 - val_loss:
0.0092
```

```
Epoch 24/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0149 - val_loss:
0.0093
Epoch 25/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0135 - val_loss:
0.0081
Epoch 26/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0138 - val_loss:
0.0086
Epoch 27/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0139 - val_loss:
0.0082
Epoch 28/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0141 - val_loss:
0.0085
Epoch 29/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0125 - val_loss:
0.0079
Epoch 30/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0129 - val_loss:
0.0093
Epoch 31/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0127 - val_loss:
0.0081
Epoch 32/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0122 - val_loss:
0.0085
Epoch 33/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0130 - val_loss:
0.0073
Epoch 34/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0115 - val_loss:
0.0092
Epoch 35/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0128 - val_loss:
0.0078
Epoch 36/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0121 - val_loss:
0.0077
Epoch 37/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0115 - val_loss:
0.0080
Epoch 38/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0117 - val_loss:
0.0083
Epoch 39/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0110 - val_loss:
0.0070
```

```
Epoch 40/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0109 - val_loss:
0.0074
Epoch 41/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0110 - val_loss:
0.0079
Epoch 42/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0115 - val_loss:
0.0076
Epoch 43/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0114 - val_loss:
0.0076
Epoch 44/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0104 - val_loss:
0.0074
Epoch 45/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0100 - val_loss:
0.0076
Epoch 46/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0103 - val_loss:
0.0077
Epoch 47/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0107 - val_loss:
0.0066
Epoch 48/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0103 - val_loss:
0.0073
Epoch 49/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0101 - val_loss:
0.0075
Epoch 50/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0100 - val_loss:
0.0066
Epoch 51/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0105 - val_loss:
0.0073
Epoch 52/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0097 - val_loss:
0.0067
Epoch 53/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0089 - val_loss:
0.0071
Epoch 54/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0100 - val_loss:
0.0072
Epoch 55/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0099 - val_loss:
0.0065
```

```
Epoch 56/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0099 - val_loss:
0.0073
Epoch 57/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0085 - val_loss:
0.0064
Epoch 58/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0091 - val_loss:
0.0068
Epoch 59/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0087 - val_loss:
0.0059
Epoch 60/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0087 - val_loss:
0.0074
Epoch 61/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0093 - val_loss:
0.0062
Epoch 62/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0095 - val_loss:
0.0069
Epoch 63/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0091 - val_loss:
0.0069
Epoch 64/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0092 - val_loss:
0.0065
Epoch 65/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0088 - val_loss:
0.0070
Epoch 66/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0083 - val_loss:
0.0066
Epoch 67/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0089 - val_loss:
0.0063
Epoch 68/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0085 - val_loss:
0.0064
Epoch 69/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0085 - val_loss:
0.0066
Epoch 70/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0085 - val_loss:
0.0064
Epoch 71/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0083 - val_loss:
0.0067
```

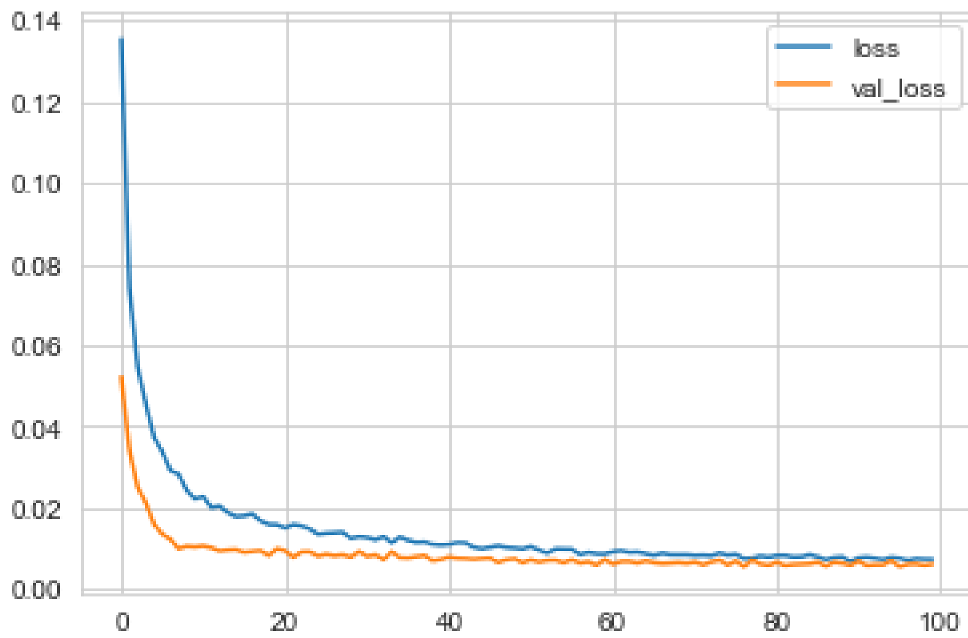
```
Epoch 72/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0084 - val_loss:
0.0060
Epoch 73/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0082 - val_loss:
0.0067
Epoch 74/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0089 - val_loss:
0.0072
Epoch 75/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0084 - val_loss:
0.0060
Epoch 76/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0086 - val_loss:
0.0069
Epoch 77/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0076 - val_loss:
0.0056
Epoch 78/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0077 - val_loss:
0.0072
Epoch 79/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0082 - val_loss:
0.0062
Epoch 80/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0079 - val_loss:
0.0059
Epoch 81/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0083 - val_loss:
0.0067
Epoch 82/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0083 - val_loss:
0.0058
Epoch 83/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0079 - val_loss:
0.0061
Epoch 84/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0080 - val_loss:
0.0062
Epoch 85/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0085 - val_loss:
0.0063
Epoch 86/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0080 - val_loss:
0.0067
Epoch 87/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0073 - val_loss:
0.0057
```

```
Epoch 88/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0078 - val_loss:
0.0067
Epoch 89/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0080 - val_loss:
0.0064
Epoch 90/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0069 - val_loss:
0.0059
Epoch 91/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0078 - val_loss:
0.0070
Epoch 92/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0079 - val_loss:
0.0058
Epoch 93/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0076 - val_loss:
0.0061
Epoch 94/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0073 - val_loss:
0.0061
Epoch 95/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0079 - val_loss:
0.0071
Epoch 96/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0076 - val_loss:
0.0056
Epoch 97/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0071 - val_loss:
0.0062
Epoch 98/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0076 - val_loss:
0.0063
Epoch 99/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0074 - val_loss:
0.0059
Epoch 100/100
9/9 [==============================] - 0s 4ms/step - loss: 0.0074 - val_loss:
0.0062
```

[48]: <keras.callbacks.History at 0x1e18b23c6a0>

[49]:
```
normal_losses = pd.DataFrame(model_normal.history.history)
normal_losses.plot()
```

[49]: <AxesSubplot:>



[50]:
```
normal_predictions = model_normal.predict(X_test_normal)
```

[51]:
```
mean_absolute_error(y_test_normal, normal_predictions)
```

[51]: 0.054884360017891126

[52]:
```
np.sqrt(mean_squared_error(y_test_normal, normal_predictions))
```

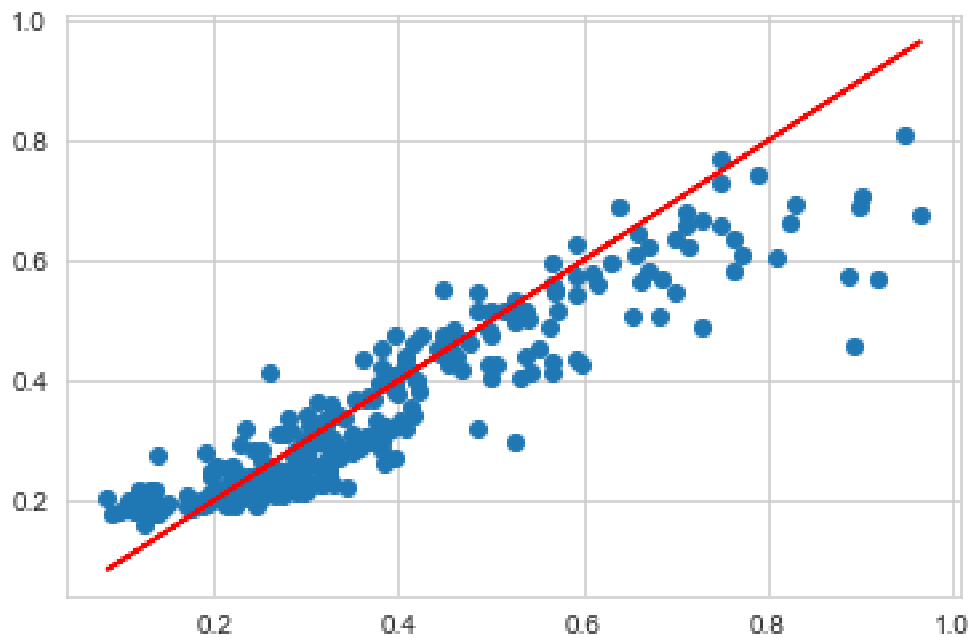
[52]: 0.07872164979893682

[53]:
```
explained_variance_score(y_test_normal, normal_predictions)
```

[53]: 0.8412406475412091

[54]:
```
plt.scatter(y_test_normal, normal_predictions)
plt.plot(y_test_normal, y_test_normal, 'r')
```

[54]: [<matplotlib.lines.Line2D at 0x1e18eb476a0>]



[55]:
```
# configuring neural net w/ adam optimizer for standardized data

# 33 33 1 best

model_standard = Sequential()

model_standard.add(Dense(33, activation='relu'))
model_standard.add(Dropout(0.2))

model_standard.add(Dense(33, activation='relu'))
model_standard.add(Dropout(0.2))

model_standard.add(Dense(1))

model_standard.compile(optimizer='adam', loss='mse')
```

[56]:
```
model_standard.fit(x=X_train_standard, y=y_train_standard,␣
↪validation_data=(X_test_standard, \
                                                                        ␣
↪y_test_standard), \
          batch_size=128, epochs=100)
```

```
Epoch 1/100
9/9 [==============================] - 0s 16ms/step - loss: 1.2218 - val_loss:
0.6603
Epoch 2/100
```

```
9/9 [==============================] - 0s 4ms/step - loss: 0.8315 - val_loss:
0.3434
Epoch 3/100
9/9 [==============================] - 0s 4ms/step - loss: 0.5968 - val_loss:
0.2691
Epoch 4/100
9/9 [==============================] - 0s 4ms/step - loss: 0.5301 - val_loss:
0.2488
Epoch 5/100
9/9 [==============================] - 0s 4ms/step - loss: 0.4791 - val_loss:
0.2309
Epoch 6/100
9/9 [==============================] - 0s 4ms/step - loss: 0.4046 - val_loss:
0.2138
Epoch 7/100
9/9 [==============================] - 0s 4ms/step - loss: 0.3954 - val_loss:
0.2011
Epoch 8/100
9/9 [==============================] - 0s 4ms/step - loss: 0.4167 - val_loss:
0.2000
Epoch 9/100
9/9 [==============================] - 0s 4ms/step - loss: 0.3740 - val_loss:
0.1949
Epoch 10/100
9/9 [==============================] - 0s 4ms/step - loss: 0.3390 - val_loss:
0.1867
Epoch 11/100
9/9 [==============================] - 0s 4ms/step - loss: 0.3009 - val_loss:
0.1866
Epoch 12/100
9/9 [==============================] - 0s 4ms/step - loss: 0.3248 - val_loss:
0.1791
Epoch 13/100
9/9 [==============================] - 0s 4ms/step - loss: 0.2865 - val_loss:
0.1649
Epoch 14/100
9/9 [==============================] - 0s 4ms/step - loss: 0.2967 - val_loss:
0.1591
Epoch 15/100
9/9 [==============================] - 0s 4ms/step - loss: 0.3163 - val_loss:
0.1573
Epoch 16/100
9/9 [==============================] - 0s 4ms/step - loss: 0.2708 - val_loss:
0.1584
Epoch 17/100
9/9 [==============================] - 0s 5ms/step - loss: 0.3058 - val_loss:
0.1572
Epoch 18/100
```

```
9/9 [==============================] - 0s 5ms/step - loss: 0.2642 - val_loss:
0.1554
Epoch 19/100
9/9 [==============================] - 0s 4ms/step - loss: 0.2757 - val_loss:
0.1544
Epoch 20/100
9/9 [==============================] - 0s 4ms/step - loss: 0.2556 - val_loss:
0.1546
Epoch 21/100
9/9 [==============================] - 0s 4ms/step - loss: 0.2115 - val_loss:
0.1484
Epoch 22/100
9/9 [==============================] - 0s 4ms/step - loss: 0.2239 - val_loss:
0.1425
Epoch 23/100
9/9 [==============================] - 0s 4ms/step - loss: 0.2365 - val_loss:
0.1406
Epoch 24/100
9/9 [==============================] - 0s 4ms/step - loss: 0.2252 - val_loss:
0.1408
Epoch 25/100
9/9 [==============================] - 0s 4ms/step - loss: 0.2205 - val_loss:
0.1408
Epoch 26/100
9/9 [==============================] - 0s 4ms/step - loss: 0.2258 - val_loss:
0.1491
Epoch 27/100
9/9 [==============================] - 0s 4ms/step - loss: 0.2308 - val_loss:
0.1570
Epoch 28/100
9/9 [==============================] - 0s 4ms/step - loss: 0.2156 - val_loss:
0.1456
Epoch 29/100
9/9 [==============================] - 0s 4ms/step - loss: 0.2009 - val_loss:
0.1379
Epoch 30/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1909 - val_loss:
0.1440
Epoch 31/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1900 - val_loss:
0.1501
Epoch 32/100
9/9 [==============================] - 0s 4ms/step - loss: 0.2193 - val_loss:
0.1405
Epoch 33/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1898 - val_loss:
0.1304
Epoch 34/100
```

```
9/9 [==============================] - 0s 4ms/step - loss: 0.1910 - val_loss:
0.1362
Epoch 35/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1826 - val_loss:
0.1418
Epoch 36/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1910 - val_loss:
0.1354
Epoch 37/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1794 - val_loss:
0.1436
Epoch 38/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1756 - val_loss:
0.1460
Epoch 39/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1787 - val_loss:
0.1375
Epoch 40/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1653 - val_loss:
0.1299
Epoch 41/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1886 - val_loss:
0.1339
Epoch 42/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1622 - val_loss:
0.1416
Epoch 43/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1595 - val_loss:
0.1317
Epoch 44/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1623 - val_loss:
0.1288
Epoch 45/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1570 - val_loss:
0.1328
Epoch 46/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1595 - val_loss:
0.1375
Epoch 47/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1692 - val_loss:
0.1398
Epoch 48/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1633 - val_loss:
0.1362
Epoch 49/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1509 - val_loss:
0.1299
Epoch 50/100
```

```
9/9 [==============================] - 0s 4ms/step - loss: 0.1530 - val_loss:
0.1320
Epoch 51/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1390 - val_loss:
0.1331
Epoch 52/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1502 - val_loss:
0.1330
Epoch 53/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1483 - val_loss:
0.1350
Epoch 54/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1345 - val_loss:
0.1361
Epoch 55/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1428 - val_loss:
0.1367
Epoch 56/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1540 - val_loss:
0.1320
Epoch 57/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1430 - val_loss:
0.1286
Epoch 58/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1442 - val_loss:
0.1404
Epoch 59/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1416 - val_loss:
0.1439
Epoch 60/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1493 - val_loss:
0.1284
Epoch 61/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1461 - val_loss:
0.1224
Epoch 62/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1331 - val_loss:
0.1351
Epoch 63/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1399 - val_loss:
0.1425
Epoch 64/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1423 - val_loss:
0.1305
Epoch 65/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1332 - val_loss:
0.1262
Epoch 66/100
```

```
9/9 [==============================] - 0s 4ms/step - loss: 0.1450 - val_loss:
0.1369
Epoch 67/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1271 - val_loss:
0.1402
Epoch 68/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1313 - val_loss:
0.1337
Epoch 69/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1265 - val_loss:
0.1255
Epoch 70/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1275 - val_loss:
0.1284
Epoch 71/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1330 - val_loss:
0.1421
Epoch 72/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1292 - val_loss:
0.1374
Epoch 73/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1296 - val_loss:
0.1280
Epoch 74/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1316 - val_loss:
0.1319
Epoch 75/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1232 - val_loss:
0.1367
Epoch 76/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1273 - val_loss:
0.1302
Epoch 77/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1204 - val_loss:
0.1366
Epoch 78/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1269 - val_loss:
0.1355
Epoch 79/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1163 - val_loss:
0.1296
Epoch 80/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1170 - val_loss:
0.1257
Epoch 81/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1331 - val_loss:
0.1346
Epoch 82/100
```

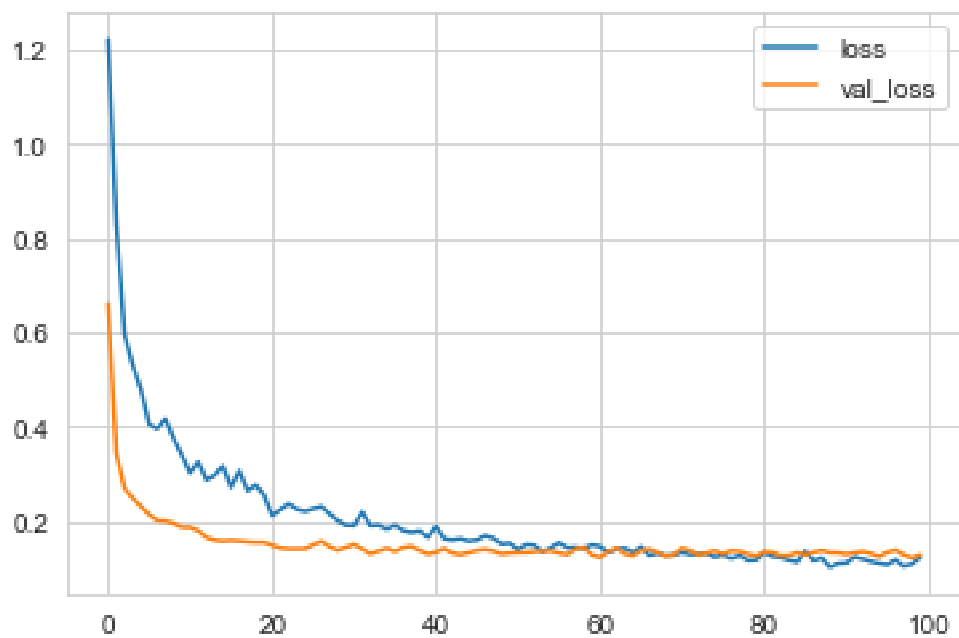
```
9/9 [==============================] - 0s 4ms/step - loss: 0.1249 - val_loss:
0.1334
Epoch 83/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1223 - val_loss:
0.1264
Epoch 84/100
9/9 [==============================] - 0s 5ms/step - loss: 0.1169 - val_loss:
0.1267
Epoch 85/100
9/9 [==============================] - 0s 5ms/step - loss: 0.1132 - val_loss:
0.1327
Epoch 86/100
9/9 [==============================] - 0s 5ms/step - loss: 0.1361 - val_loss:
0.1282
Epoch 87/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1159 - val_loss:
0.1336
Epoch 88/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1217 - val_loss:
0.1375
Epoch 89/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1019 - val_loss:
0.1327
Epoch 90/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1097 - val_loss:
0.1326
Epoch 91/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1107 - val_loss:
0.1295
Epoch 92/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1238 - val_loss:
0.1328
Epoch 93/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1193 - val_loss:
0.1349
Epoch 94/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1139 - val_loss:
0.1308
Epoch 95/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1102 - val_loss:
0.1247
Epoch 96/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1070 - val_loss:
0.1335
Epoch 97/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1176 - val_loss:
0.1383
Epoch 98/100
```

```
9/9 [==============================] - 0s 4ms/step - loss: 0.1037 - val_loss:
0.1282
Epoch 99/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1083 - val_loss:
0.1217
Epoch 100/100
9/9 [==============================] - 0s 4ms/step - loss: 0.1229 - val_loss:
0.1278
```

[56]: <keras.callbacks.History at 0x1e18ac423a0>

[57]:
```
standard_losses = pd.DataFrame(model_standard.history.history)
standard_losses.plot()
```

[57]: <AxesSubplot:>



[58]:
```
standard_predictions = model_standard.predict(X_test_standard)
```

[59]:
```
mean_absolute_error(y_test_standard, standard_predictions)
```

[59]: 0.24173186557614557

[60]:
```
np.sqrt(mean_squared_error(y_test_standard, standard_predictions))
```

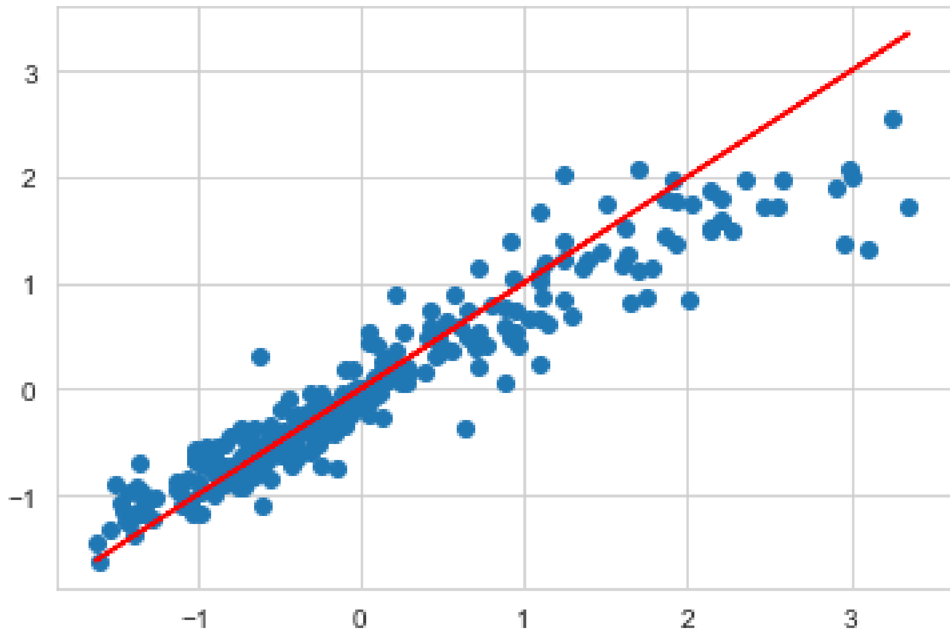
[60]: 0.3575448254640191

[61]:
```
explained_variance_score(y_test_standard, standard_predictions)
```

[61]:
```
0.8870123453564851
```

[62]:
```
plt.scatter(y_test_standard, standard_predictions)
plt.plot(y_test_standard, y_test_standard, 'r')
```

[62]:
```
[<matplotlib.lines.Line2D at 0x1e18b1b5e80>]
```



[63]:
```
# TODO: neural net on full data set (maybe)
# TODO: neural net on pca of standardized data (non-lasso-selected?) (maybe)
```

[64]:
```
# modelling neural net for all cols

X_train_standard = train_standard.drop('SalePrice', axis=1)
y_train_standard = train_standard['SalePrice']

X_train_stand, X_test_stand, y_train_stand, y_test_stand =␣
 ↪train_test_split(X_train_standard, \
                                                                                  ␣
 ↪y_train_standard, \
                                                                                  ␣
 ↪test_size=0.2, \
                                                                                  ␣
 ↪random_state=0)
```

[65]:
```
# configuring neural net w/ adam optimizer for standardized non-selected data

# rmse:
# 147 77 39 .0642
# 147 147 77 39 1 .0648
# 147 77 77 1 .0706
# 147 147 77 .0731
# 147 77 77 .0741
# 147 147 1 .0777

model_stand = Sequential()

model_stand.add(Dense(147, activation='relu'))
model_stand.add(Dropout(0.2))

# model_stand.add(Dense(147, activation='relu'))
# model_stand.add(Dropout(0.2))

model_stand.add(Dense(77, activation='relu'))
model_stand.add(Dropout(0.2))

model_stand.add(Dense(39, activation='relu'))
model_stand.add(Dropout(0.2))

model_stand.add(Dense(1))

model_stand.compile(optimizer='adam', loss='mse')
```

[66]:
```
model_stand.fit(x=X_train_stand, y=y_train_stand,␣
 ↪validation_data=(X_test_stand, \
                                                                                ␣
 ↪y_test_stand), \
          batch_size=128, epochs=100)
```

```
Epoch 1/100
9/9 [==============================] - 1s 18ms/step - loss: 0.8462 - val_loss:
0.3582
Epoch 2/100
9/9 [==============================] - 0s 6ms/step - loss: 0.4516 - val_loss:
0.2216
Epoch 3/100
9/9 [==============================] - 0s 5ms/step - loss: 0.3519 - val_loss:
0.2334
Epoch 4/100
9/9 [==============================] - 0s 5ms/step - loss: 0.3193 - val_loss:
0.2155
Epoch 5/100
```

```
9/9 [==============================] - 0s 6ms/step - loss: 0.2602 - val_loss:
0.1810
Epoch 6/100
9/9 [==============================] - 0s 5ms/step - loss: 0.2498 - val_loss:
0.1775
Epoch 7/100
9/9 [==============================] - 0s 6ms/step - loss: 0.2220 - val_loss:
0.1788
Epoch 8/100
9/9 [==============================] - 0s 5ms/step - loss: 0.2100 - val_loss:
0.1741
Epoch 9/100
9/9 [==============================] - 0s 5ms/step - loss: 0.2102 - val_loss:
0.1702
Epoch 10/100
9/9 [==============================] - 0s 5ms/step - loss: 0.2067 - val_loss:
0.1711
Epoch 11/100
9/9 [==============================] - 0s 6ms/step - loss: 0.2148 - val_loss:
0.1664
Epoch 12/100
9/9 [==============================] - 0s 5ms/step - loss: 0.1970 - val_loss:
0.1555
Epoch 13/100
9/9 [==============================] - 0s 6ms/step - loss: 0.1972 - val_loss:
0.1476
Epoch 14/100
9/9 [==============================] - 0s 5ms/step - loss: 0.1661 - val_loss:
0.1438
Epoch 15/100
9/9 [==============================] - 0s 5ms/step - loss: 0.1661 - val_loss:
0.1487
Epoch 16/100
9/9 [==============================] - 0s 6ms/step - loss: 0.1622 - val_loss:
0.1444
Epoch 17/100
9/9 [==============================] - 0s 6ms/step - loss: 0.1573 - val_loss:
0.1407
Epoch 18/100
9/9 [==============================] - 0s 6ms/step - loss: 0.1531 - val_loss:
0.1468
Epoch 19/100
9/9 [==============================] - 0s 6ms/step - loss: 0.1393 - val_loss:
0.1474
Epoch 20/100
9/9 [==============================] - 0s 6ms/step - loss: 0.1440 - val_loss:
0.1513
Epoch 21/100
```

```
9/9 [==============================] - 0s 5ms/step - loss: 0.1326 - val_loss:
0.1279
Epoch 22/100
9/9 [==============================] - 0s 6ms/step - loss: 0.1433 - val_loss:
0.1633
Epoch 23/100
9/9 [==============================] - 0s 5ms/step - loss: 0.1425 - val_loss:
0.1403
Epoch 24/100
9/9 [==============================] - 0s 5ms/step - loss: 0.1268 - val_loss:
0.1389
Epoch 25/100
9/9 [==============================] - 0s 5ms/step - loss: 0.1367 - val_loss:
0.1464
Epoch 26/100
9/9 [==============================] - 0s 6ms/step - loss: 0.1164 - val_loss:
0.1342
Epoch 27/100
9/9 [==============================] - 0s 5ms/step - loss: 0.1280 - val_loss:
0.1475
Epoch 28/100
9/9 [==============================] - 0s 6ms/step - loss: 0.1277 - val_loss:
0.1337
Epoch 29/100
9/9 [==============================] - 0s 5ms/step - loss: 0.1170 - val_loss:
0.1343
Epoch 30/100
9/9 [==============================] - 0s 6ms/step - loss: 0.1100 - val_loss:
0.1374
Epoch 31/100
9/9 [==============================] - 0s 5ms/step - loss: 0.1021 - val_loss:
0.1284
Epoch 32/100
9/9 [==============================] - 0s 5ms/step - loss: 0.1289 - val_loss:
0.1519
Epoch 33/100
9/9 [==============================] - 0s 5ms/step - loss: 0.1123 - val_loss:
0.1290
Epoch 34/100
9/9 [==============================] - 0s 5ms/step - loss: 0.1130 - val_loss:
0.1419
Epoch 35/100
9/9 [==============================] - 0s 6ms/step - loss: 0.1042 - val_loss:
0.1542
Epoch 36/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0990 - val_loss:
0.1332
Epoch 37/100
```

```
9/9 [==============================] - 0s 5ms/step - loss: 0.1056 - val_loss:
0.1356
Epoch 38/100
9/9 [==============================] - 0s 5ms/step - loss: 0.1021 - val_loss:
0.1333
Epoch 39/100
9/9 [==============================] - 0s 6ms/step - loss: 0.1019 - val_loss:
0.1346
Epoch 40/100
9/9 [==============================] - 0s 5ms/step - loss: 0.1040 - val_loss:
0.1288
Epoch 41/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0876 - val_loss:
0.1365
Epoch 42/100
9/9 [==============================] - 0s 6ms/step - loss: 0.0945 - val_loss:
0.1454
Epoch 43/100
9/9 [==============================] - 0s 6ms/step - loss: 0.0994 - val_loss:
0.1292
Epoch 44/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0968 - val_loss:
0.1346
Epoch 45/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0943 - val_loss:
0.1313
Epoch 46/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0967 - val_loss:
0.1423
Epoch 47/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0876 - val_loss:
0.1397
Epoch 48/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0932 - val_loss:
0.1299
Epoch 49/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0851 - val_loss:
0.1317
Epoch 50/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0805 - val_loss:
0.1286
Epoch 51/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0829 - val_loss:
0.1346
Epoch 52/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0847 - val_loss:
0.1411
Epoch 53/100
```

```
9/9 [==============================] - 0s 5ms/step - loss: 0.0986 - val_loss:
0.1297
Epoch 54/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0898 - val_loss:
0.1396
Epoch 55/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0908 - val_loss:
0.1393
Epoch 56/100
9/9 [==============================] - 0s 6ms/step - loss: 0.0788 - val_loss:
0.1382
Epoch 57/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0894 - val_loss:
0.1441
Epoch 58/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0851 - val_loss:
0.1298
Epoch 59/100
9/9 [==============================] - 0s 6ms/step - loss: 0.0798 - val_loss:
0.1414
Epoch 60/100
9/9 [==============================] - 0s 6ms/step - loss: 0.0816 - val_loss:
0.1331
Epoch 61/100
9/9 [==============================] - 0s 6ms/step - loss: 0.0864 - val_loss:
0.1366
Epoch 62/100
9/9 [==============================] - 0s 6ms/step - loss: 0.0788 - val_loss:
0.1344
Epoch 63/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0743 - val_loss:
0.1407
Epoch 64/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0793 - val_loss:
0.1341
Epoch 65/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0773 - val_loss:
0.1359
Epoch 66/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0779 - val_loss:
0.1177
Epoch 67/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0757 - val_loss:
0.1357
Epoch 68/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0695 - val_loss:
0.1333
Epoch 69/100
```

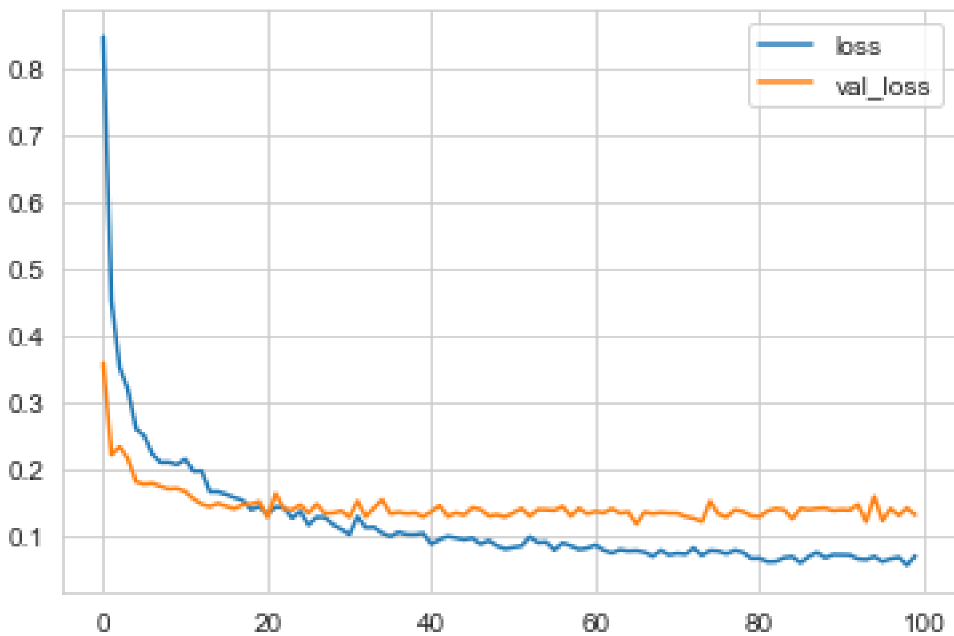
```
9/9 [==============================] - 0s 5ms/step - loss: 0.0783 - val_loss:
0.1351
Epoch 70/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0715 - val_loss:
0.1340
Epoch 71/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0739 - val_loss:
0.1339
Epoch 72/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0726 - val_loss:
0.1291
Epoch 73/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0819 - val_loss:
0.1262
Epoch 74/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0707 - val_loss:
0.1214
Epoch 75/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0782 - val_loss:
0.1518
Epoch 76/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0772 - val_loss:
0.1329
Epoch 77/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0732 - val_loss:
0.1285
Epoch 78/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0787 - val_loss:
0.1391
Epoch 79/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0753 - val_loss:
0.1365
Epoch 80/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0663 - val_loss:
0.1301
Epoch 81/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0667 - val_loss:
0.1288
Epoch 82/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0612 - val_loss:
0.1373
Epoch 83/100
9/9 [==============================] - 0s 6ms/step - loss: 0.0617 - val_loss:
0.1409
Epoch 84/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0673 - val_loss:
0.1392
Epoch 85/100
```

```
9/9 [==============================] - 0s 5ms/step - loss: 0.0690 - val_loss:
0.1255
Epoch 86/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0597 - val_loss:
0.1412
Epoch 87/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0688 - val_loss:
0.1393
Epoch 88/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0754 - val_loss:
0.1408
Epoch 89/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0673 - val_loss:
0.1413
Epoch 90/100
9/9 [==============================] - 0s 6ms/step - loss: 0.0719 - val_loss:
0.1380
Epoch 91/100
9/9 [==============================] - 0s 6ms/step - loss: 0.0717 - val_loss:
0.1393
Epoch 92/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0714 - val_loss:
0.1385
Epoch 93/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0658 - val_loss:
0.1469
Epoch 94/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0645 - val_loss:
0.1211
Epoch 95/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0691 - val_loss:
0.1591
Epoch 96/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0615 - val_loss:
0.1227
Epoch 97/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0656 - val_loss:
0.1408
Epoch 98/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0680 - val_loss:
0.1309
Epoch 99/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0564 - val_loss:
0.1417
Epoch 100/100
9/9 [==============================] - 0s 5ms/step - loss: 0.0699 - val_loss:
0.1315
```

[66]: <keras.callbacks.History at 0x1e1897a5700>

[67]:
```
stand_losses = pd.DataFrame(model_stand.history.history)
stand_losses.plot()
```

[67]: <AxesSubplot:>



[68]:
```
stand_predictions = model_stand.predict(X_test_stand)
```

[69]:
```
mean_absolute_error(y_test_stand, stand_predictions)
```

[69]: 0.2528513299851795

[70]:
```
np.sqrt(mean_squared_error(y_test_stand, stand_predictions))
```

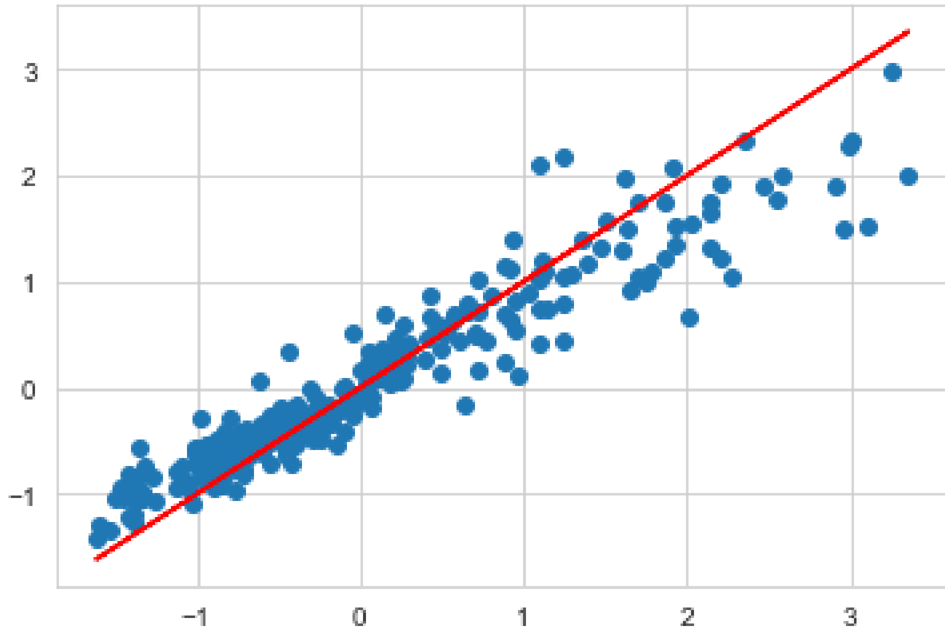
[70]: 0.36256629216467884

[71]:
```
explained_variance_score(y_test_stand, stand_predictions)
```

[71]: 0.8807082761768282

[72]:
```
plt.scatter(y_test_stand, stand_predictions)
plt.plot(y_test_stand, y_test_stand, 'r')
```

[72]: [<matplotlib.lines.Line2D at 0x1e1881c4760>]



[73]:
```
# TODO: inverse scale predictions

temp_train = train.drop('SalePrice', axis=1)
temp_train['SalePrice'] = train['SalePrice']

scaler = StandardScaler()
scaler.fit(temp_train.drop(labels='Id', axis=1))
```

[73]: StandardScaler()

[74]:
```
temp = X_test_stand.copy()
temp['Predictions'] = stand_predictions
```

[75]:
```
temp_trans = pd.DataFrame(scaler.inverse_transform(temp), columns=temp.columns)
```

[76]:
```
corrected_stand_pred = temp_trans['Predictions'].to_numpy()
```

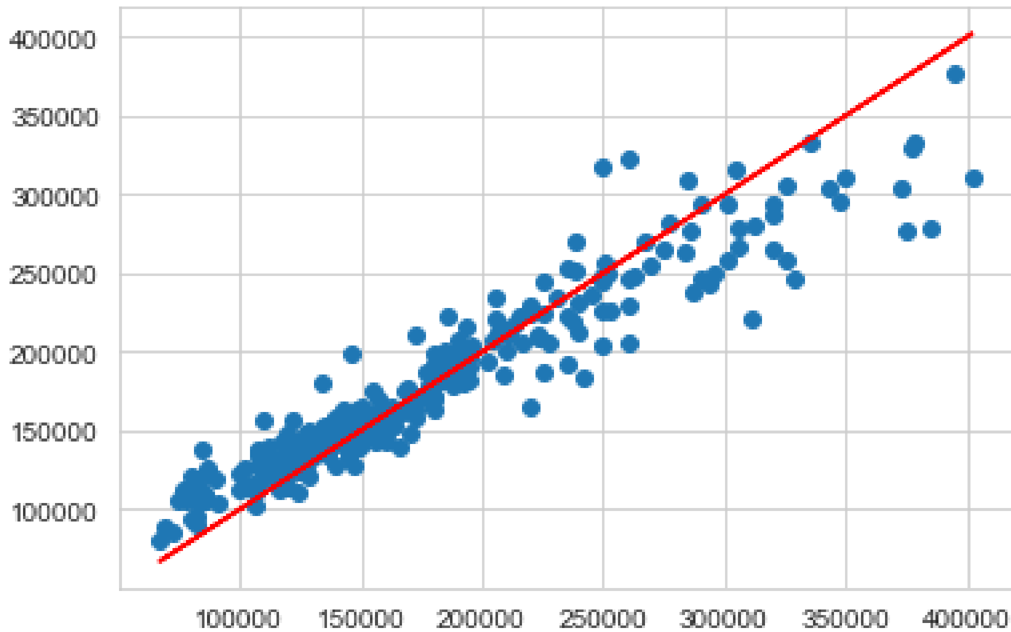
[77]:
```
temp_corrected_test = X_test_stand.copy()
temp_corrected_test['y_test'] = y_test_stand.to_numpy()
```

[78]:
```
temp_trans_test = pd.DataFrame(scaler.inverse_transform(temp_corrected_test),
                               columns=temp_corrected_test.columns)
```

[79]:
```
corrected_stand_test = temp_trans_test['y_test'].to_numpy()
```

[80]:
```
plt.scatter(corrected_stand_test, corrected_stand_pred)
plt.plot(corrected_stand_test, corrected_stand_test, 'r')
```

[80]: [<matplotlib.lines.Line2D at 0x1e18b0770a0>]



[81]:
```
# inverse scaling working properly, make test predictions and apply inverse␣
↪transformation
```

[82]:
```
scaled_predictions = model_stand.predict(test_standard)
```

[83]:
```
temp = test_standard.copy()
temp['Predictions'] = scaled_predictions
```

[84]:
```
temp_predictions = pd.DataFrame(scaler.inverse_transform(temp), columns=temp.
↪columns)
```

[85]:
```
predictions_array = temp_predictions['Predictions'].to_numpy()
```

[86]:
```
predictions = pd.DataFrame({'Id': test.Id, 'SalePrice': predictions_array})
```

[87]:
```
predictions.tail()
```

[87]:

	Id	SalePrice
1454	2915	109484.807776
1455	2916	99962.820805
1456	2917	158730.260232

```
1457  2918  139104.945553
1458  2919  241892.581487
```

[88]:
```
sample_sumbission = pd.read_csv('sample_submission.csv')
sample_sumbission.shape
```

[88]:
```
(1459, 2)
```

[89]:
```
predictions.shape
```

[89]:
```
(1459, 2)
```

[90]:
```
predictions.to_csv('submission.csv', index=False)
```