notebook02-resolvendo_um_problema_de_regressao

November 17, 2020

1 Resolvendo um problema de regressão

1.1 Dados do Curso

Instituição: IFES

Curso: Mestrado Profissional Computação Aplicada

Professor: Francisco de Assis Boldt

Aluno: Arthur Chisté Lucas

1.2 Ambiente

IDE: MS Visual Studio Code

Versão Python: 3.8.3 64bits com anaconda 2020.07

1.3 Introdução

Nesta tarefa, será utilizado um dataset contendo preços de casas, obtido no site Kaggle:

https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data

Conforme abaixo, o dataset precisa ser baixado e armazenado no diretório **data/house_prices_dataset**

[45]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

url = 'https://github.com/arthurclucas/ReconhecimentoPadroes/blob/main/data/
 ↪house_prices_dataset/train.csv?raw=true'
dados = pd.read_csv(url)
dados.head()
```

[45]:
```
   Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
0   1          60       RL         65.0     8450   Pave   NaN      Reg
1   2          20       RL         80.0     9600   Pave   NaN      Reg
2   3          60       RL         68.0    11250   Pave   NaN      IR1
```

```
3   4          70       RL          60.0     9550    Pave    NaN       IR1
4   5          60       RL          84.0    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]
```

Removendo colunas com poucos dados preenchidos e preenchendo as demais com N/A

[46]:
```python
dados.fillna(dados.mean(), inplace=True)
dados.fillna('N/A', inplace=True)
#dados.drop('Alley', axis = 1, inplace=True)
#dados.drop('FireplaceQu', axis = 1, inplace=True)
#dados.drop('PoolQC', axis = 1, inplace=True)
#dados.drop('Fence', axis = 1, inplace=True)
#dados.drop('MiscFeature', axis = 1, inplace=True)
dados.columns[dados.isna().any()].tolist()
```

[46]:
```
[]
```

[47]:
```python
dados.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    1460 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   Alley          1460 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     1460 non-null   object
26  MasVnrArea     1460 non-null   float64
27  ExterQual      1460 non-null   object
28  ExterCond      1460 non-null   object
29  Foundation     1460 non-null   object
30  BsmtQual       1460 non-null   object
31  BsmtCond       1460 non-null   object
32  BsmtExposure   1460 non-null   object
33  BsmtFinType1   1460 non-null   object
34  BsmtFinSF1     1460 non-null   int64
35  BsmtFinType2   1460 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     1460 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    1460 non-null   object
 58  GarageType     1460 non-null   object
 59  GarageYrBlt    1460 non-null   float64
 60  GarageFinish   1460 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1460 non-null   object
 64  GarageCond     1460 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         1460 non-null   object
 73  Fence          1460 non-null   object
 74  MiscFeature    1460 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
```

Posteriormente, transformar as colunas não numéricas, gerando uma coluna para cada valor com 0 - não possui característica, 1 - possui característica

[48]:

```
#dados['MSZoning']
#pd.get_dummies(dados['MSZoning'], prefix='MSZoning')
```

Por hora, separando apenas os dados numéricos para análise

[49]:

```
dados = dados.select_dtypes(include=np.number)
dados
```

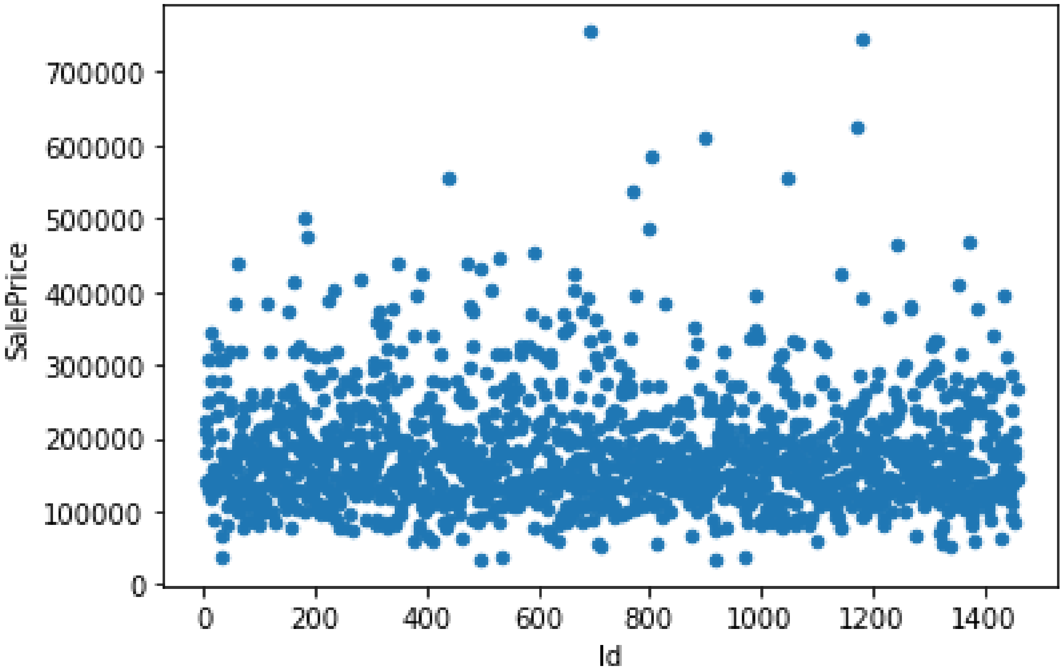
[49]:

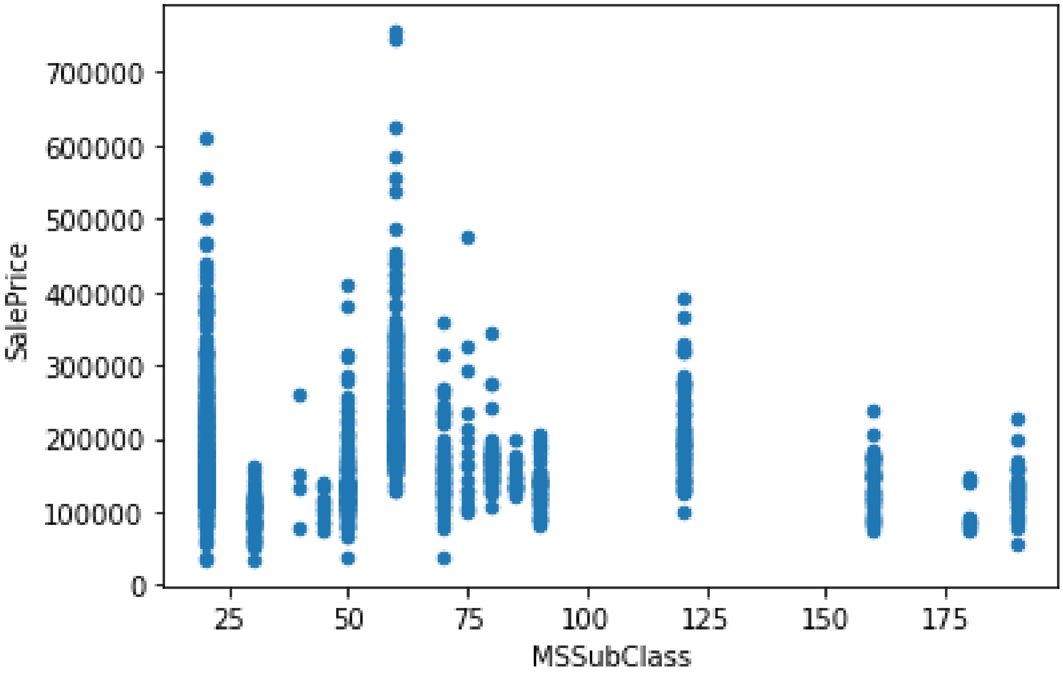
```
        Id  MSSubClass  LotFrontage  LotArea  OverallQual  OverallCond  \
0        1          60         65.0     8450            7            5
1        2          20         80.0     9600            6            8
2        3          60         68.0    11250            7            5
3        4          70         60.0     9550            7            5
4        5          60         84.0    14260            8            5
…      …          …            …        …          …            …
1455  1456          60         62.0     7917            6            5
1456  1457          20         85.0    13175            6            6
1457  1458          70         66.0     9042            7            9
```

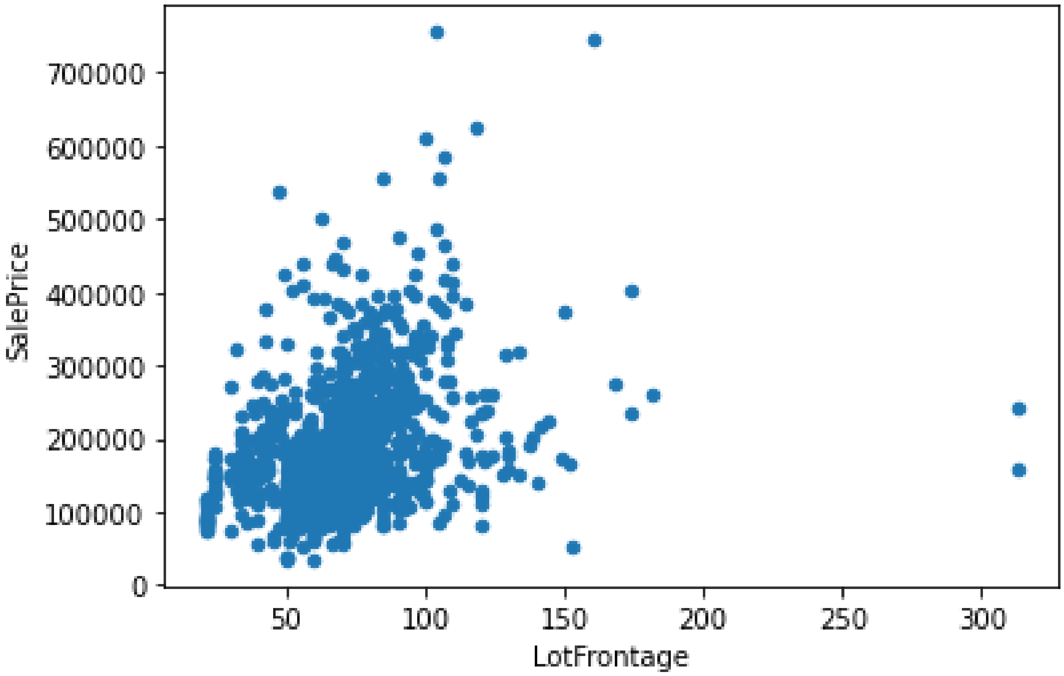
```
1458  1459          20          68.0     9717            5            6
1459  1460          20          75.0     9937            5            6

      YearBuilt  YearRemodAdd  MasVnrArea  BsmtFinSF1  …  WoodDeckSF  \
0          2003          2003       196.0         706  …           0
1          1976          1976         0.0         978  …         298
2          2001          2002       162.0         486  …           0
3          1915          1970         0.0         216  …           0
4          2000          2000       350.0         655  …         192
…         …           …         …         …  …         …
1455       1999          2000         0.0           0  …           0
1456       1978          1988       119.0         790  …         349
1457       1941          2006         0.0         275  …           0
1458       1950          1996         0.0          49  …         366
1459       1965          1965         0.0         830  …         736

      OpenPorchSF  EnclosedPorch  3SsnPorch  ScreenPorch  PoolArea  MiscVal  \
0              61              0          0            0         0        0
1               0              0          0            0         0        0
2              42              0          0            0         0        0
3              35            272          0            0         0        0
4              84              0          0            0         0        0
…           …            …        …          …       …      …
1455           40              0          0            0         0        0
1456            0              0          0            0         0        0
1457           60              0          0            0         0     2500
1458            0            112          0            0         0        0
1459           68              0          0            0         0        0

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

[1460 rows x 38 columns]
```

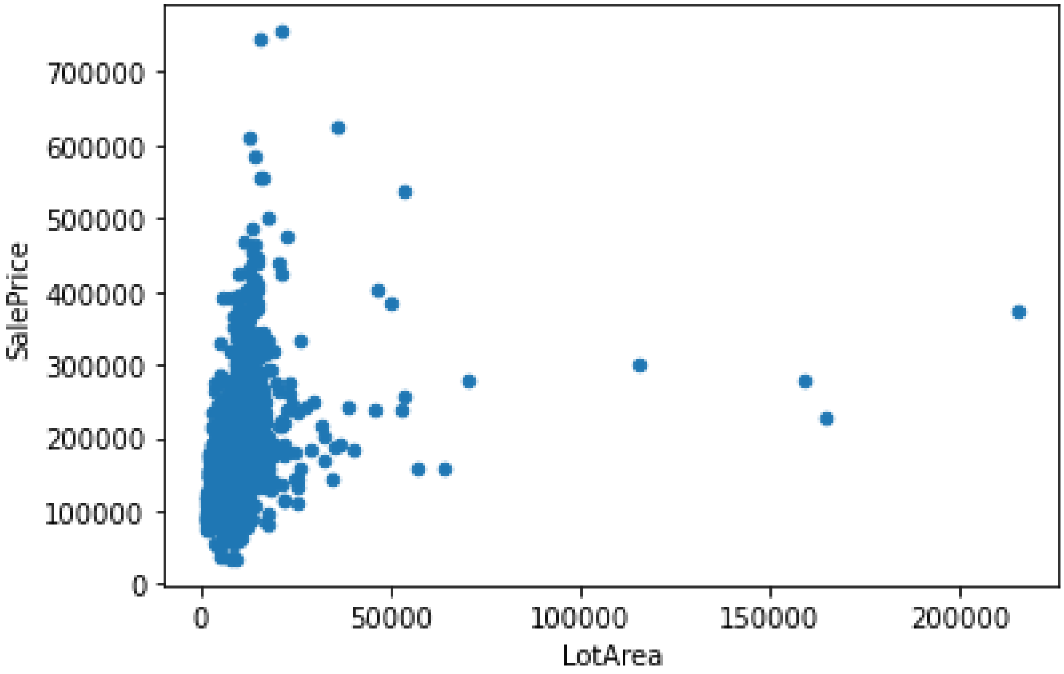
Imprime cada uma das características

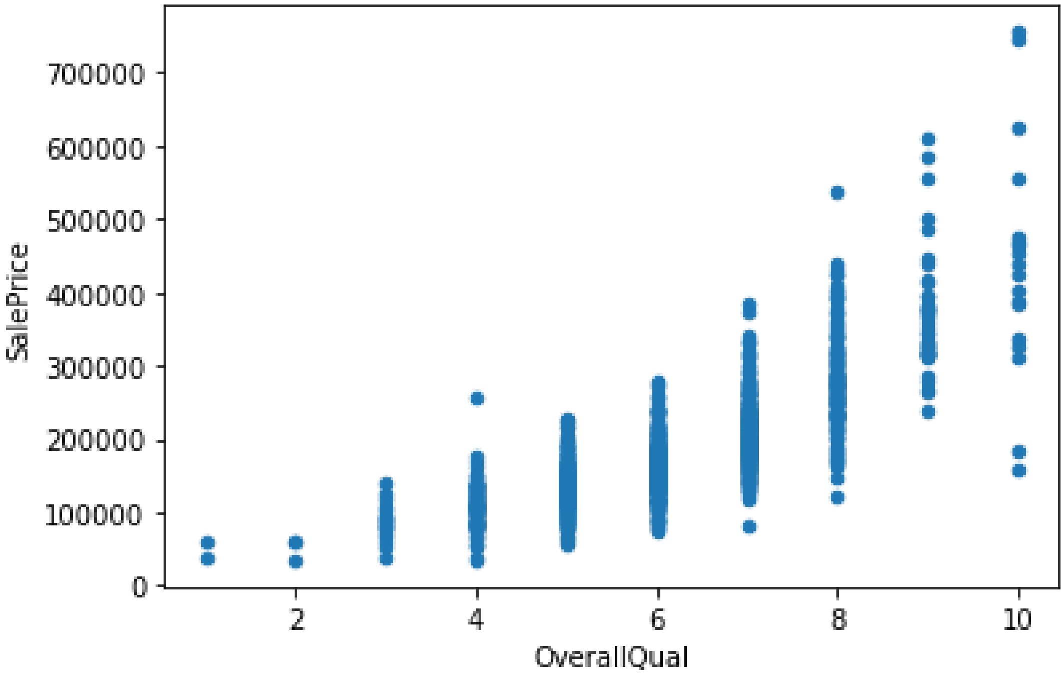
[50]:
```
for i in range(len(dados.columns) -1):
    dados.plot.scatter(x=i, y='SalePrice')
```

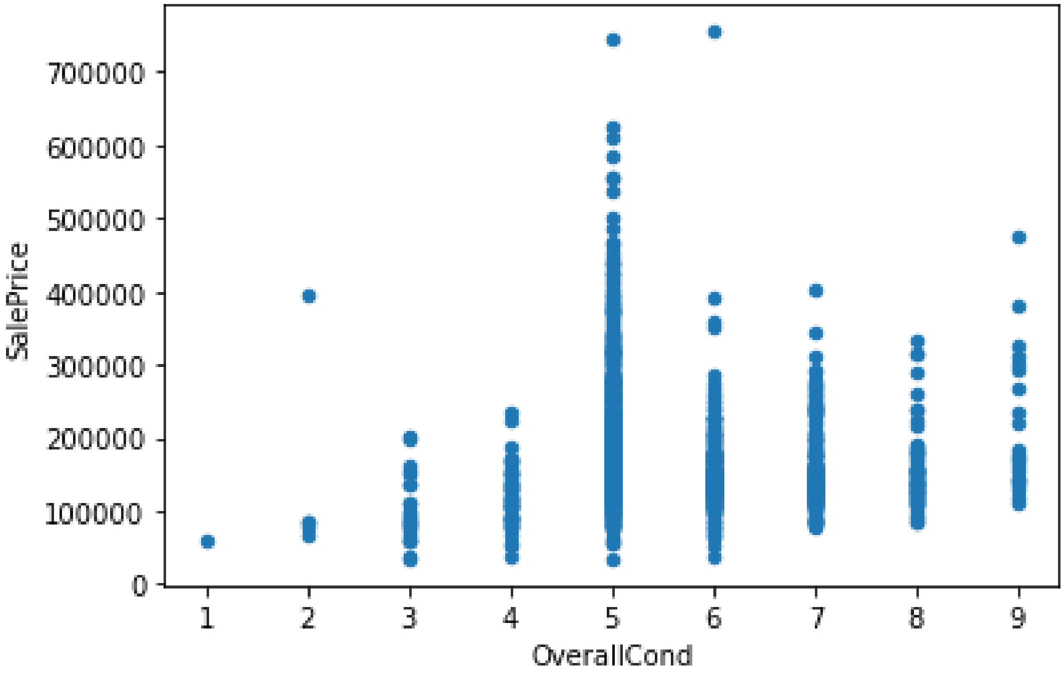


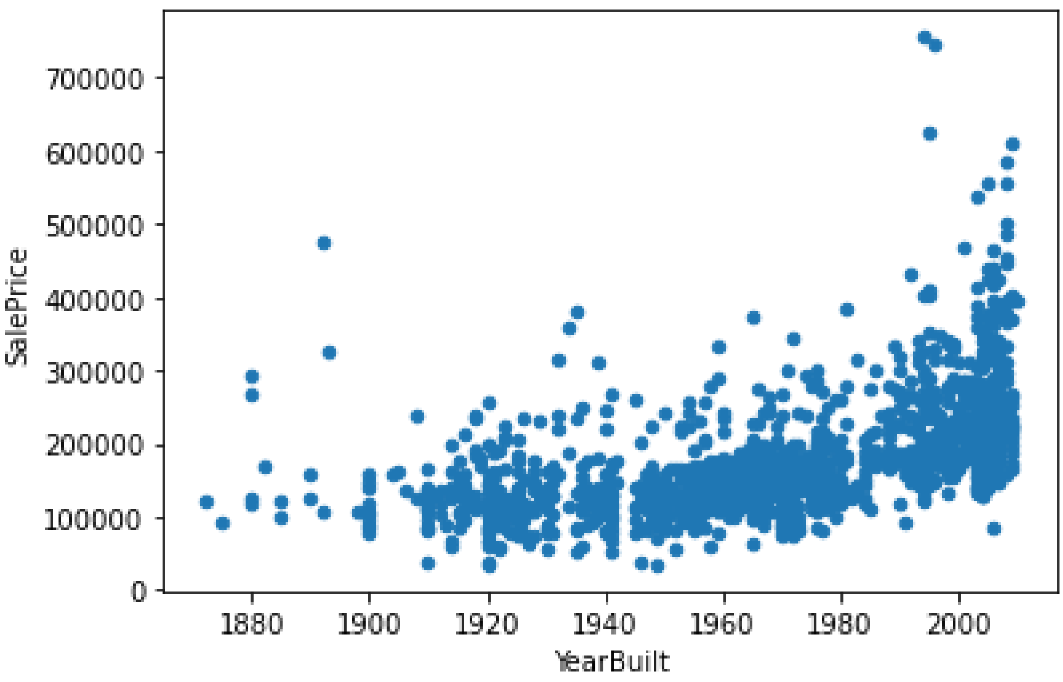


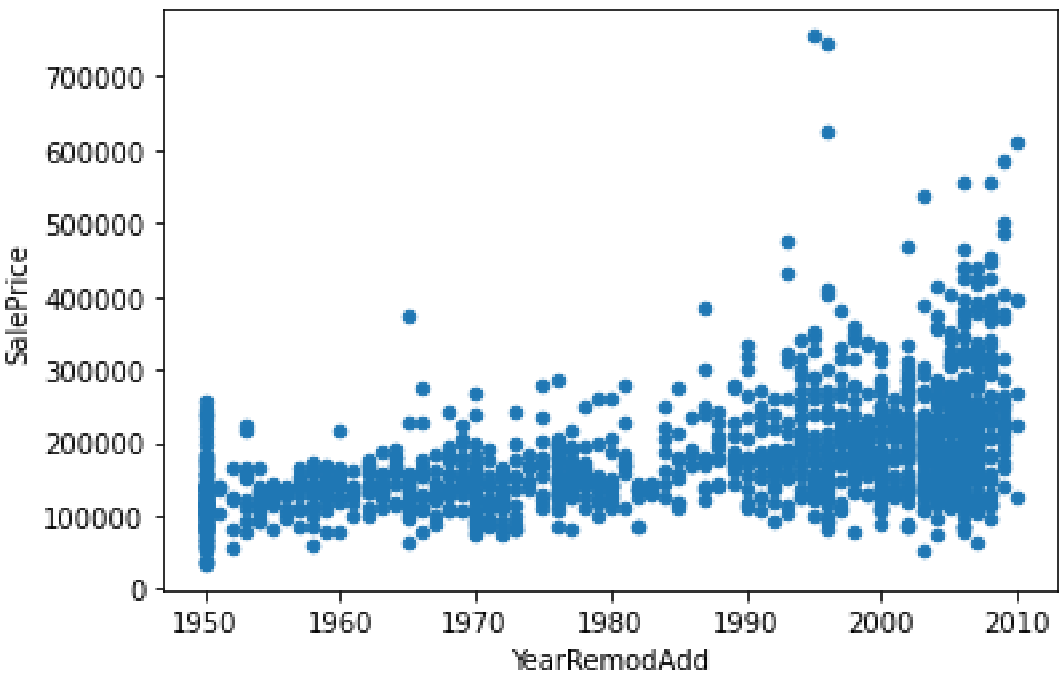

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
50
100
150
200
250
300
LotFrontage

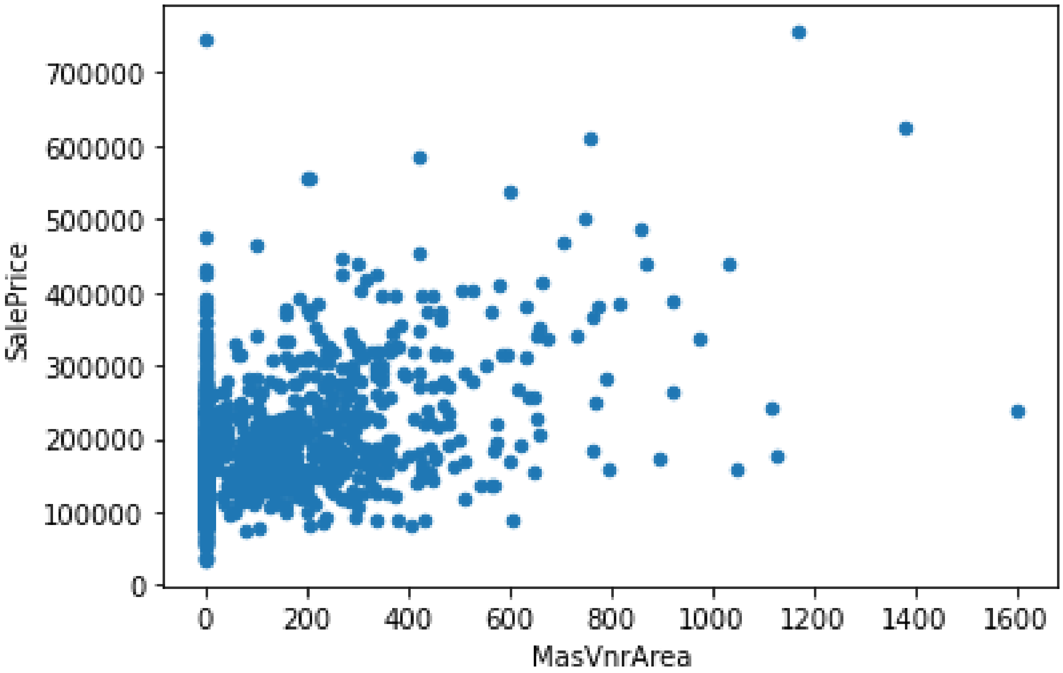

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
50000
100000
150000
200000
LotArea


700000
600000
500000
400000
300000
200000
100000
0
SalePrice
2
4
6
8
10
OverallQual

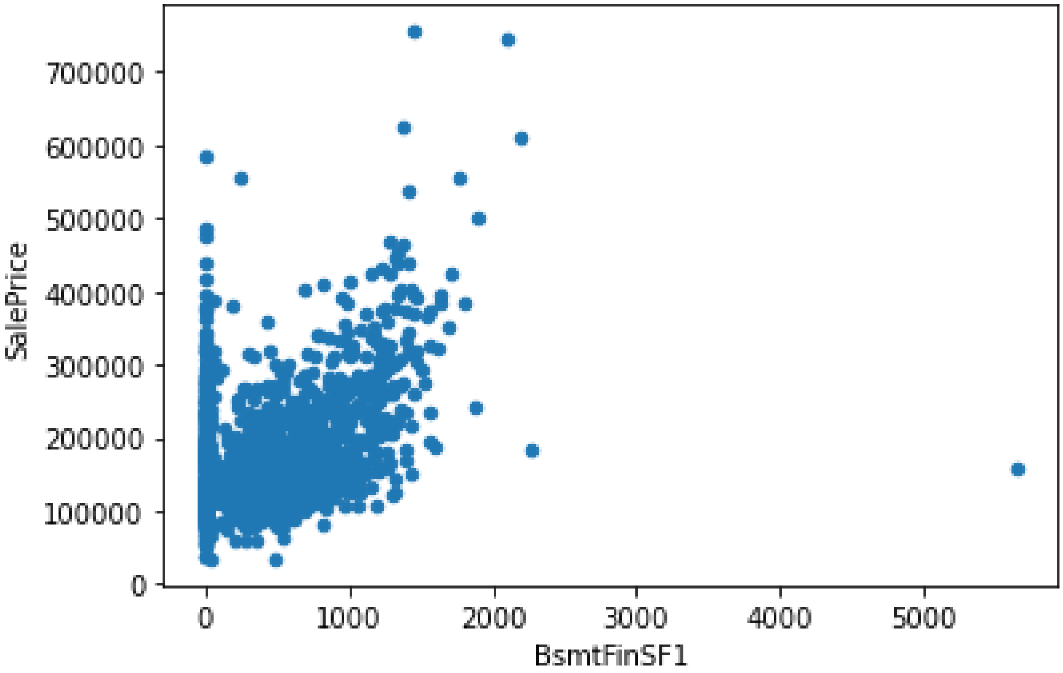

700000
600000
500000
400000
300000
200000
100000
0
SalePrice
1
2
3
4
5
6
7
8
9
OverallCond


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

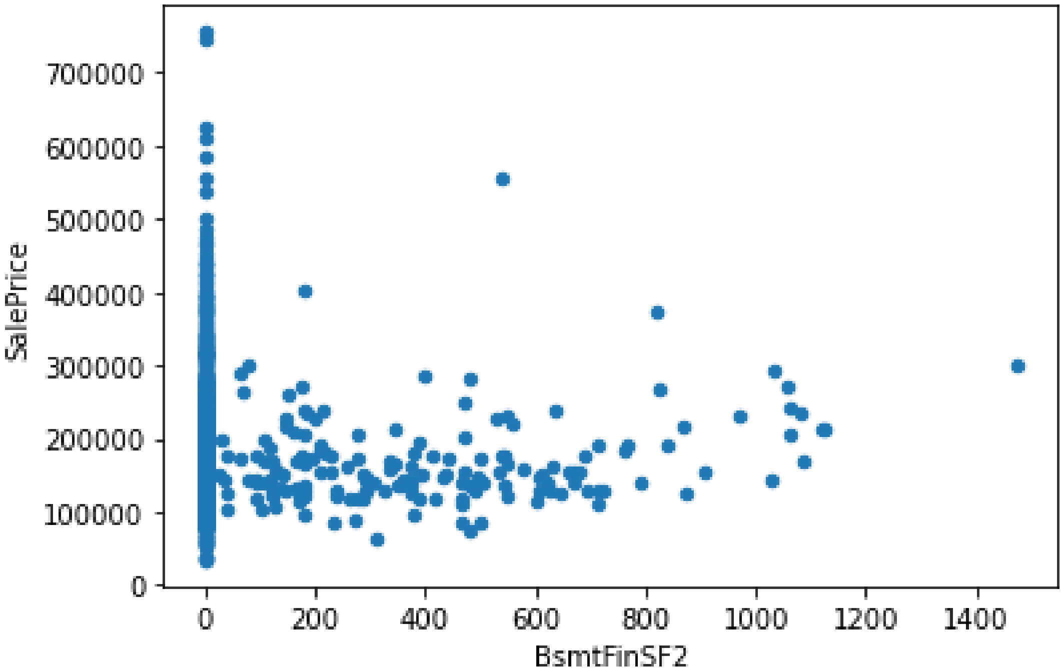

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
1950
1960
1970
1980
1990
2000
2010
YearRemodAdd



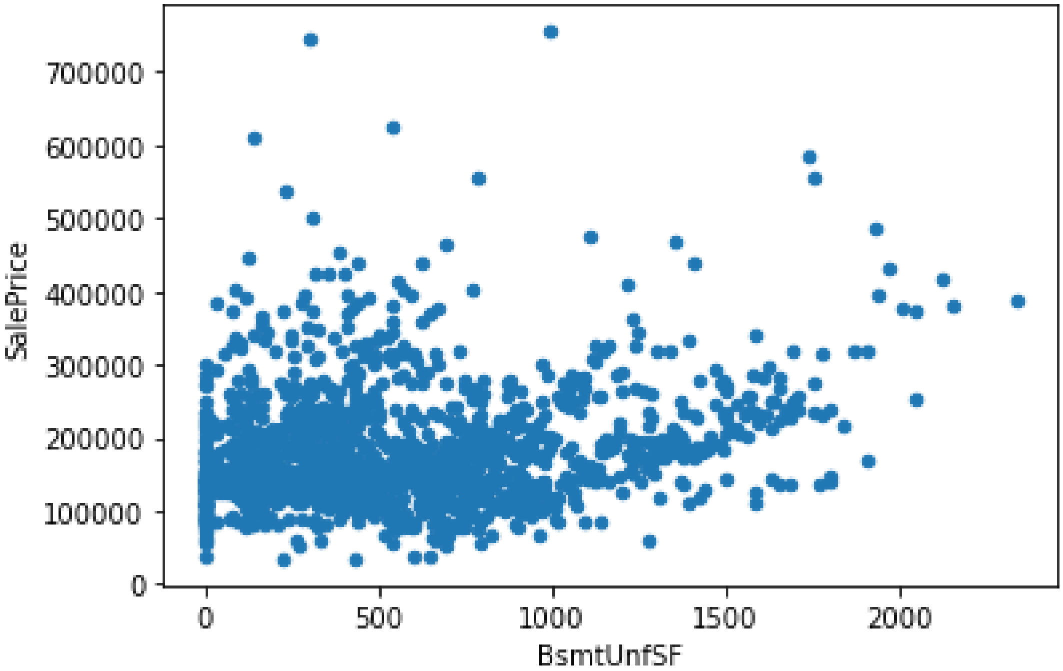
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
200
400
600
800
1000
1200
1400
1600
MasVnrArea



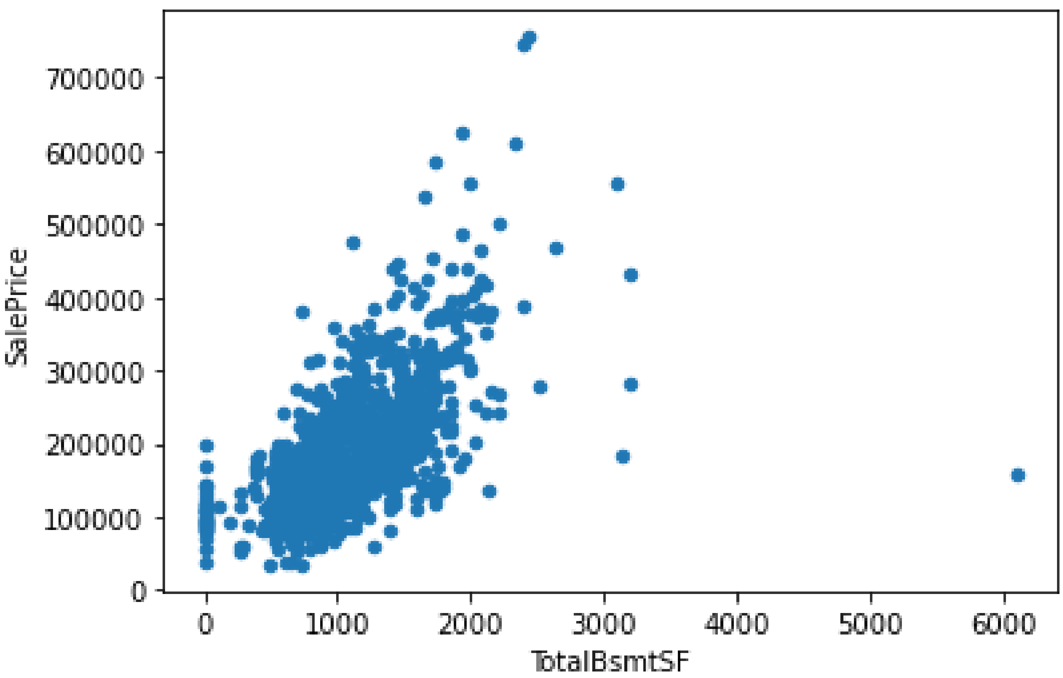
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
1000
2000
3000
4000
5000
BsmtFinSF1

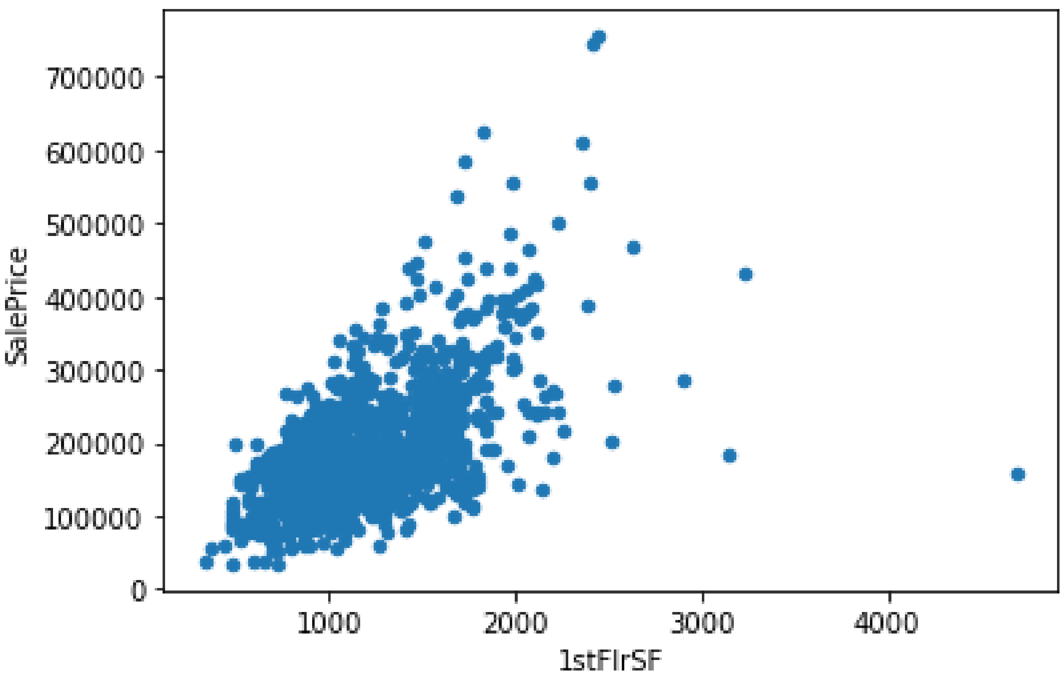


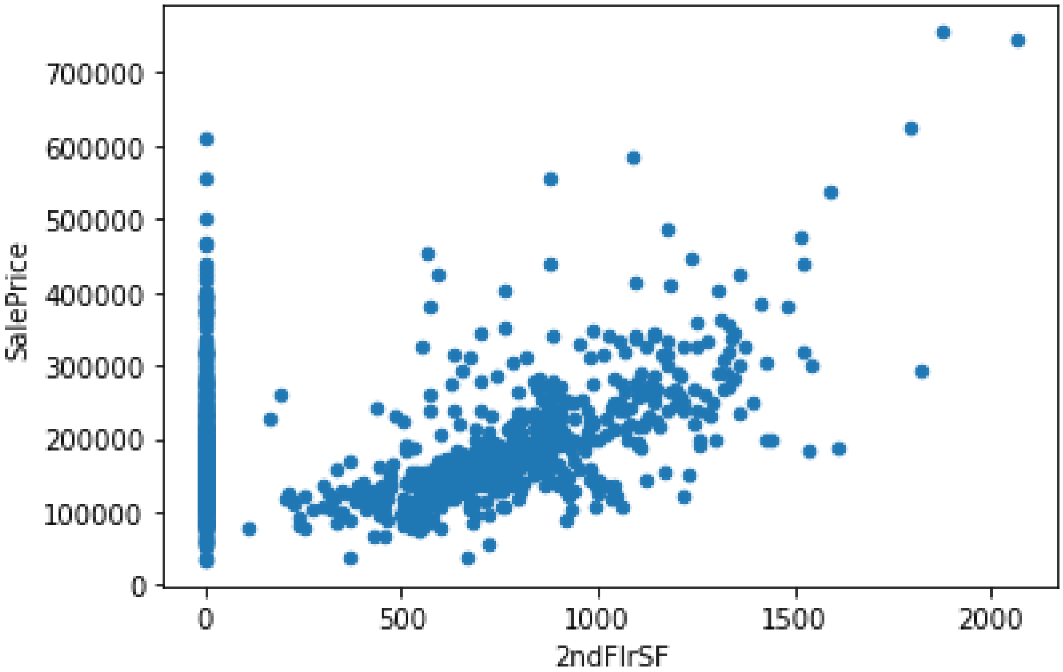
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
200
400
600
800
1000
1200
1400
BsmtFinSF2



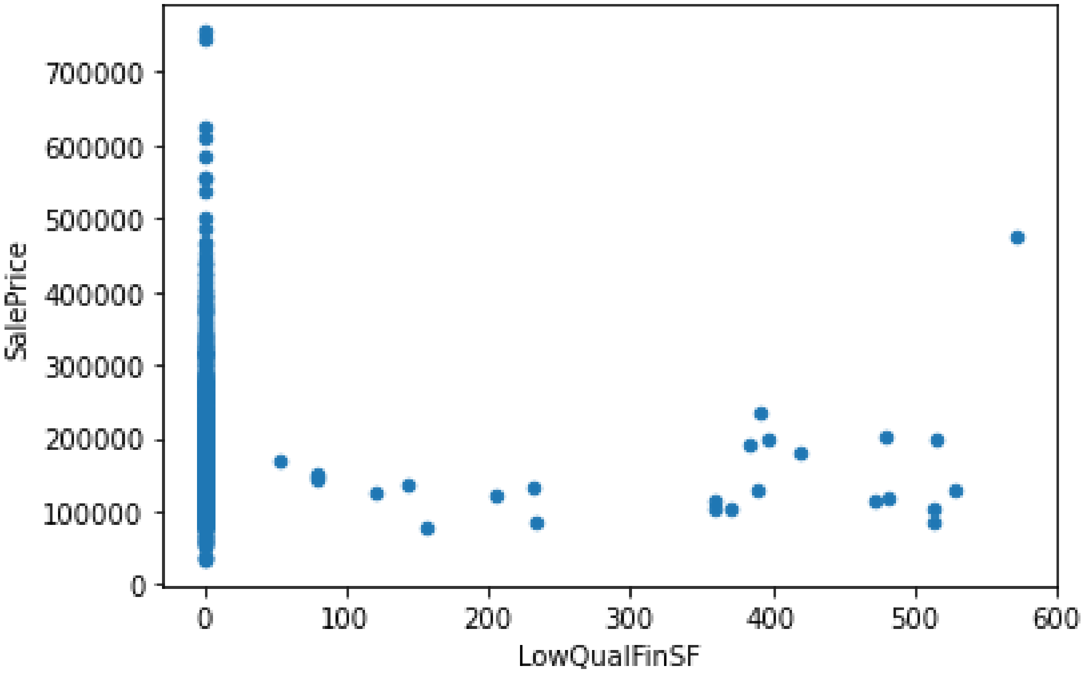
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
500
1000
1500
2000
BsmtUnfSF


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

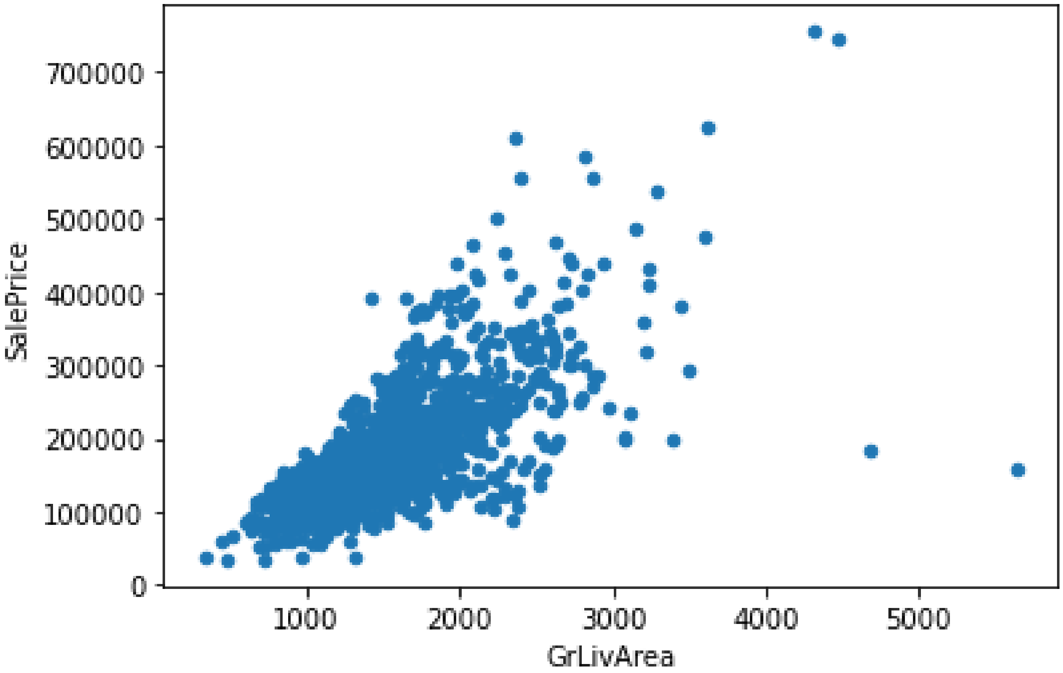

700000
600000
500000
400000
300000
200000
100000
0
SalePrice
1000
2000
3000
4000
1stFlrSF

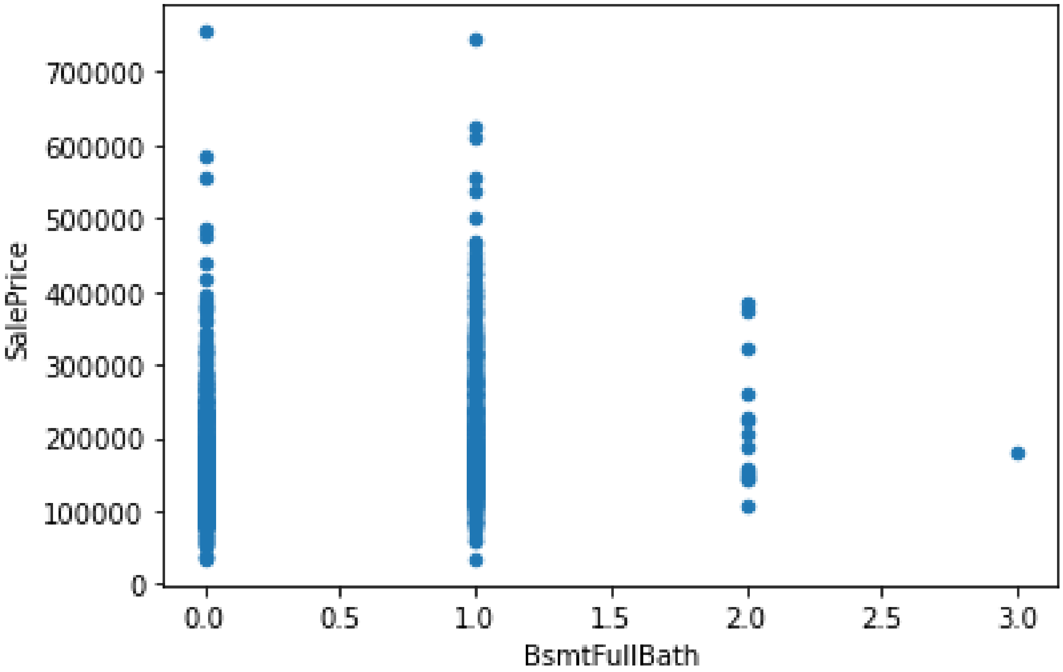


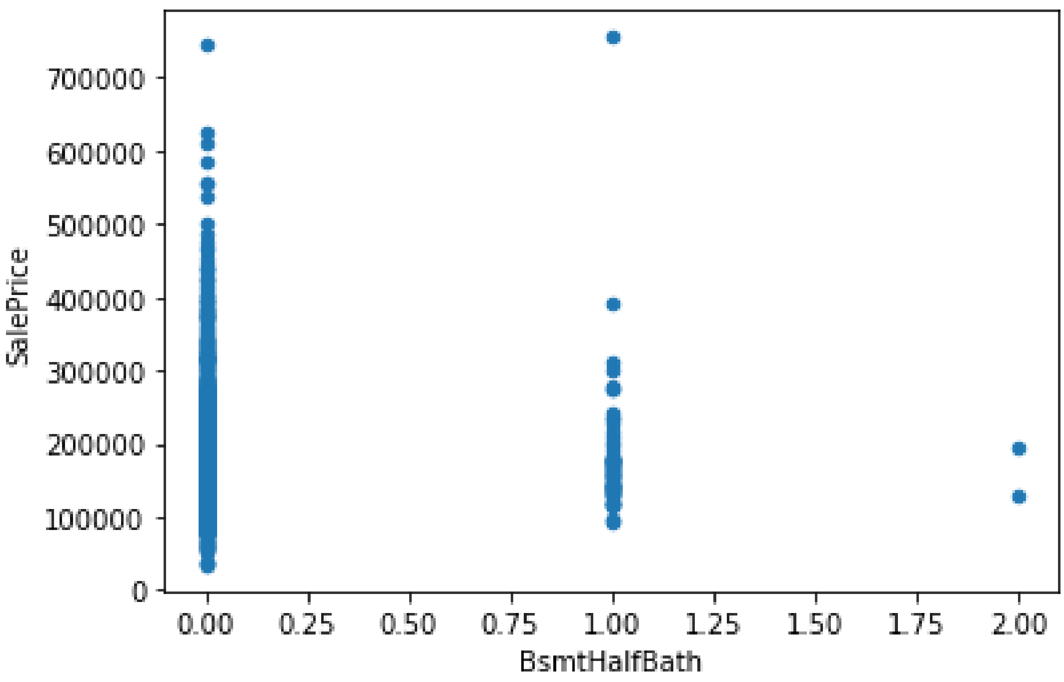
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
500
1000
1500
2000
2ndFlrSF

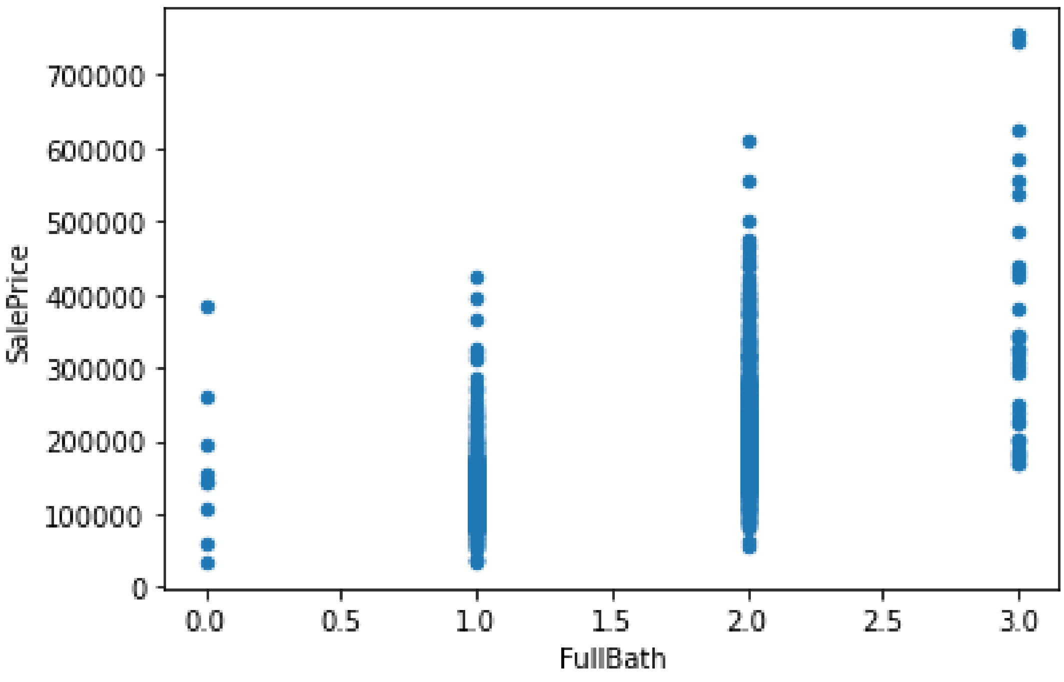


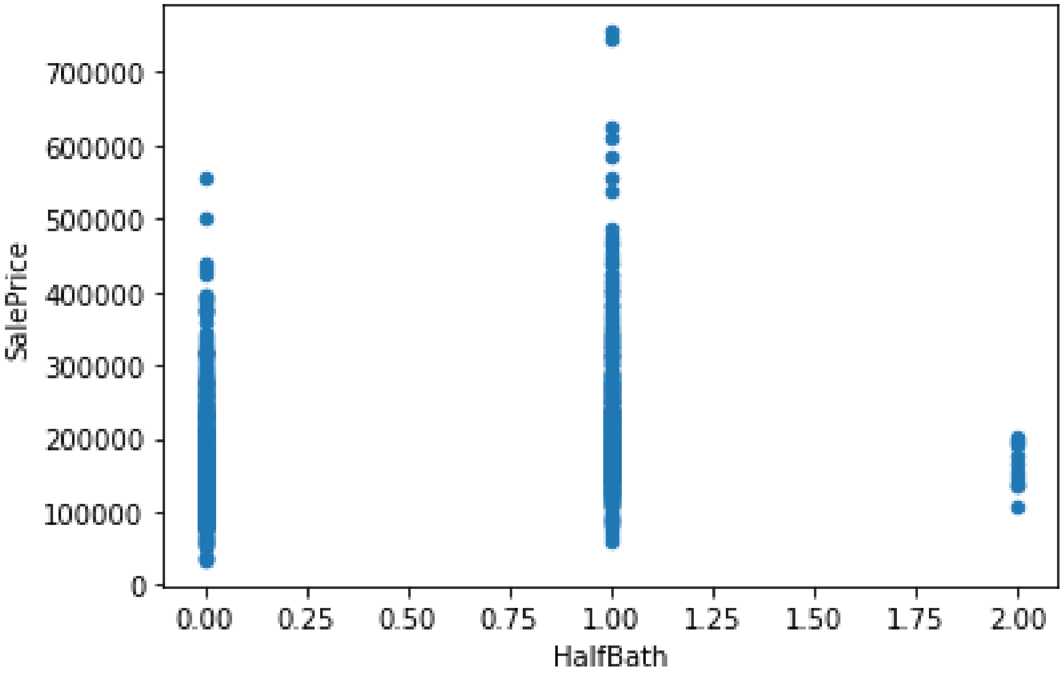
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
100
200
300
400
500
600
LowQualFinSF


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

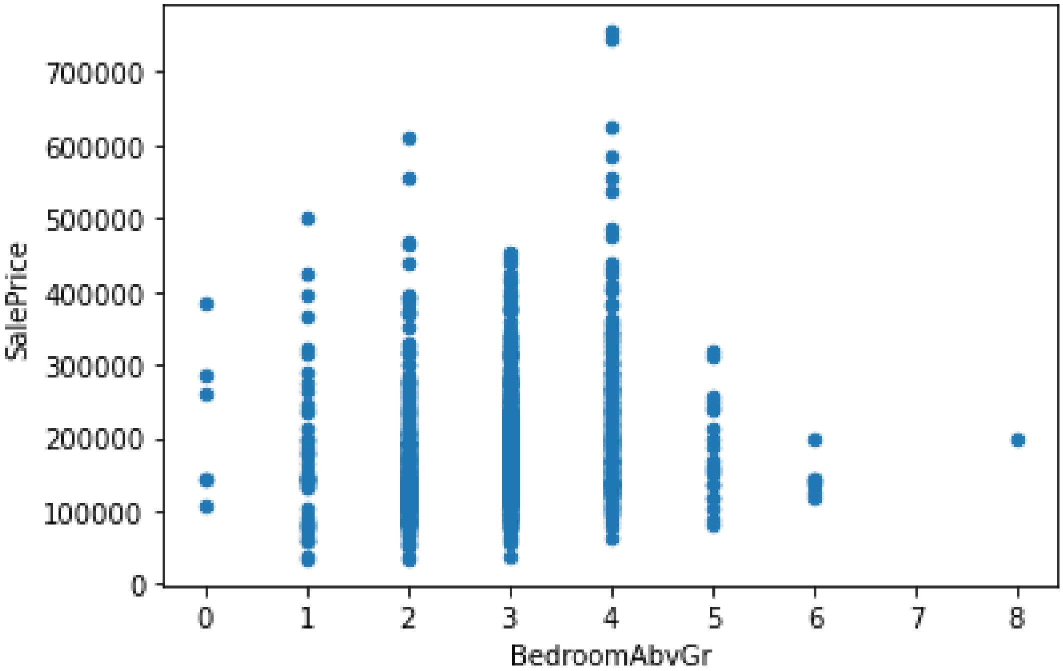

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
BsmtFullBath


700000
600000
500000
400000
300000
200000
100000
0
SalePrice
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
BsmtHalfBath

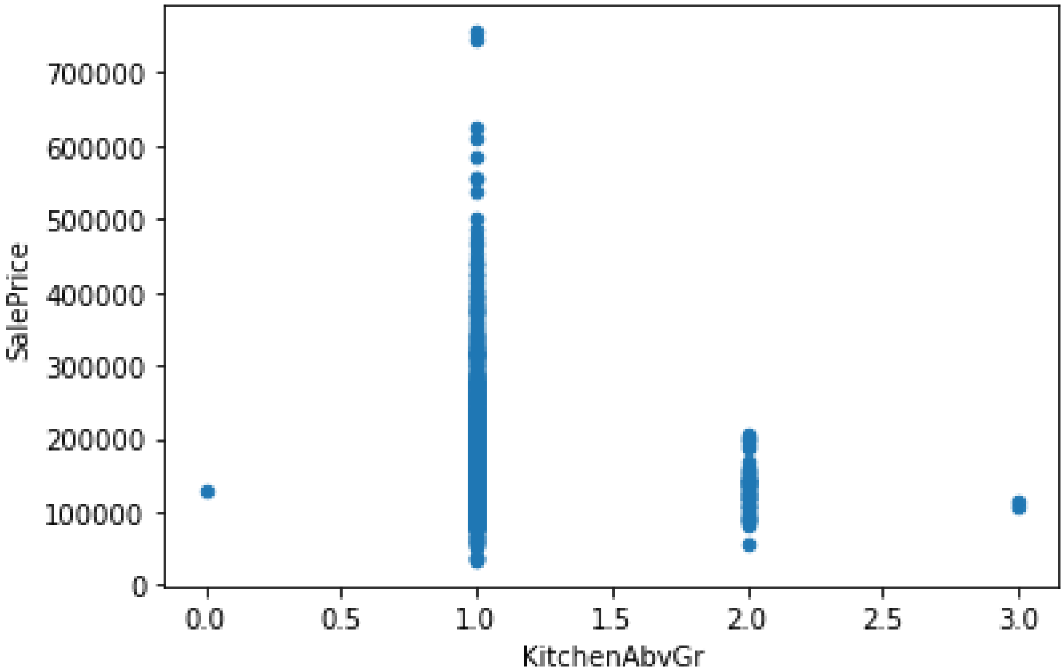

700000
600000
500000
400000
300000
200000
100000
0
SalePrice
0.0
0.5
1.0
1.5
2.0
2.5
3.0
FullBath

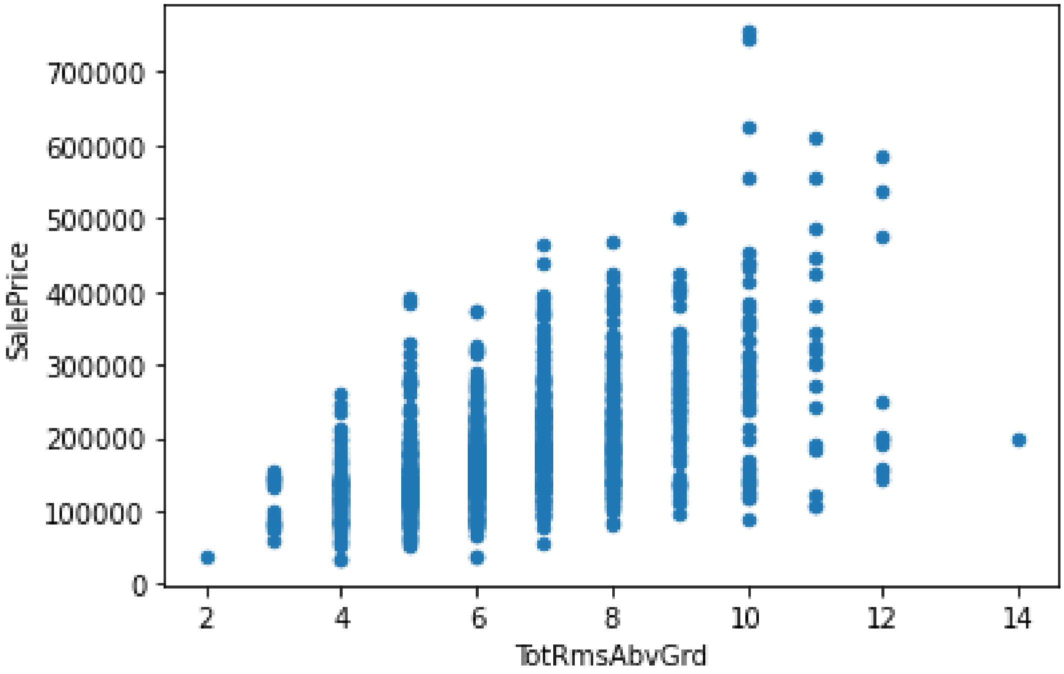


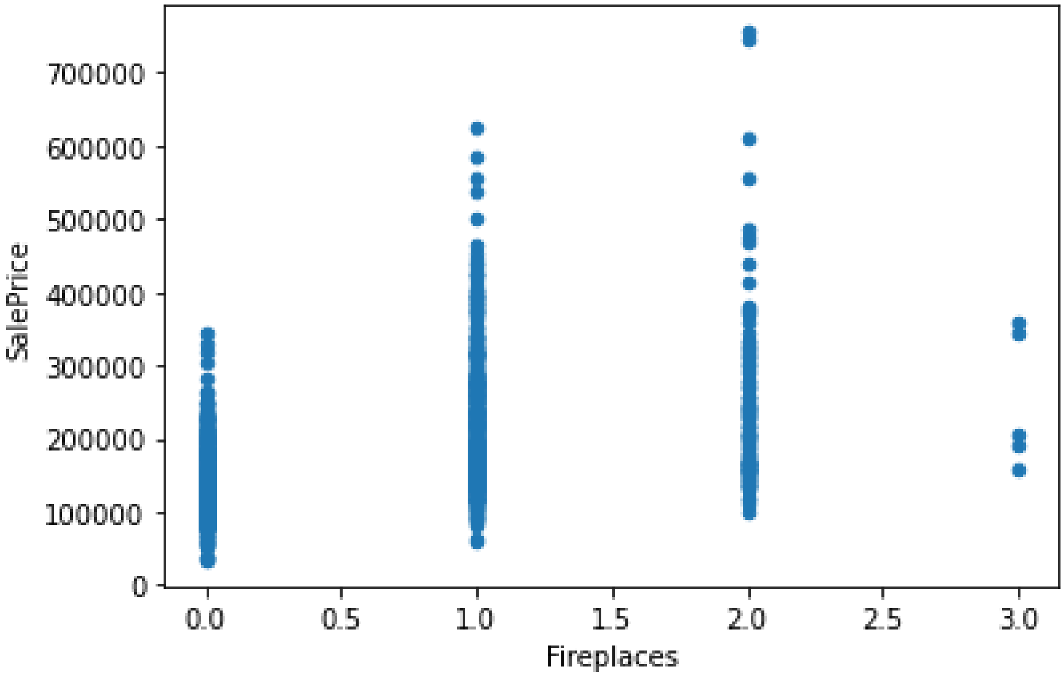
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
HalfBath

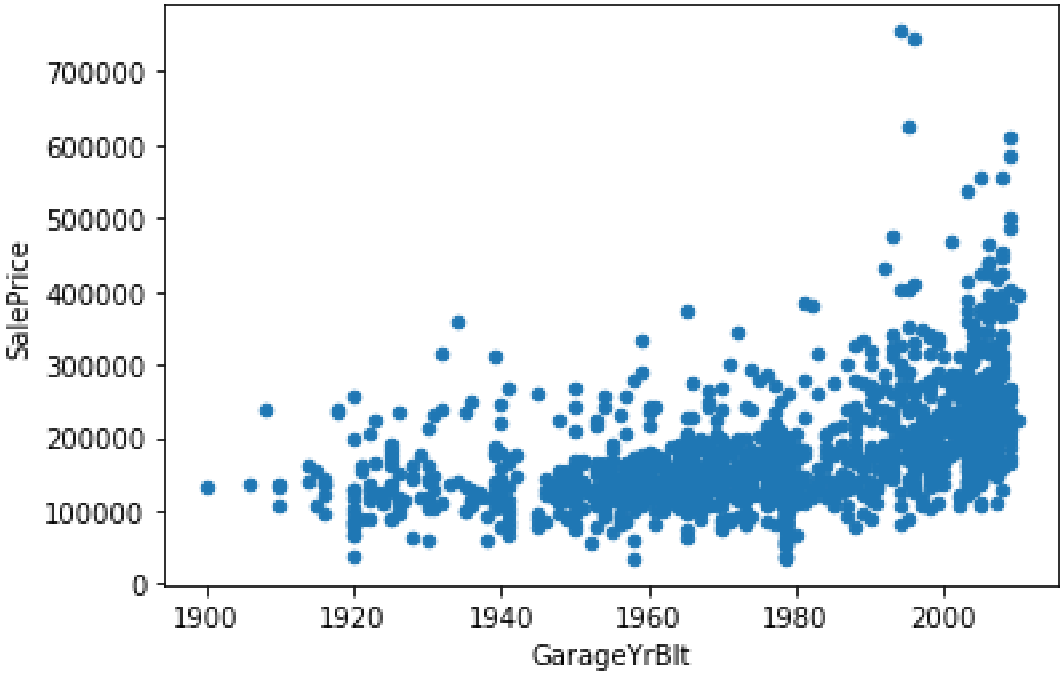


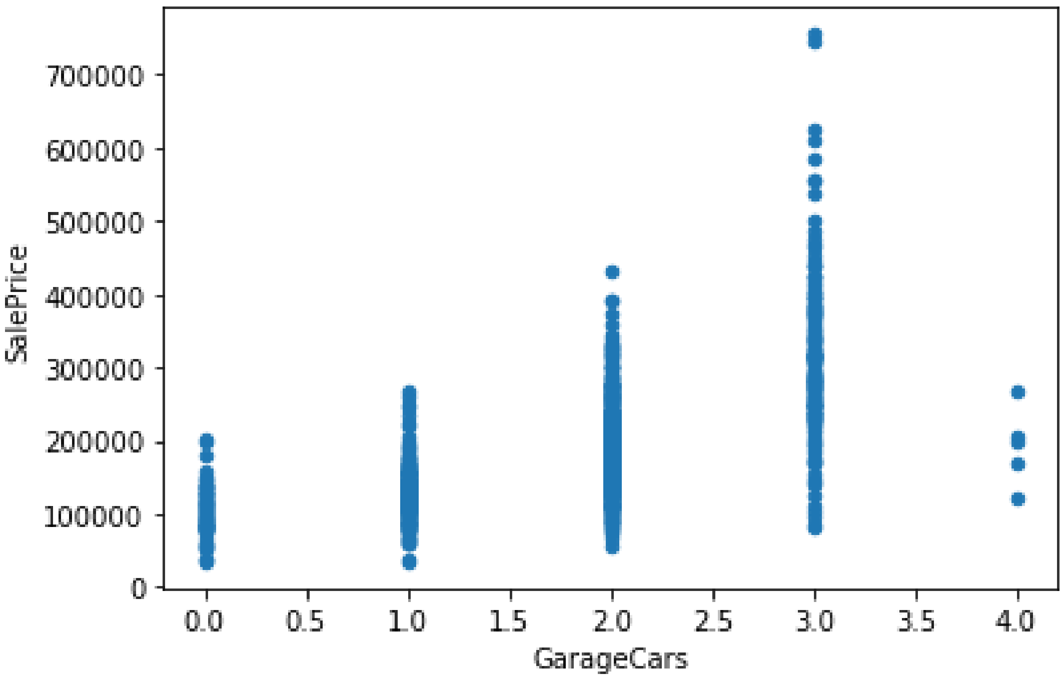
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
1
2
3
4
5
6
7
8
BedroomAbvGr

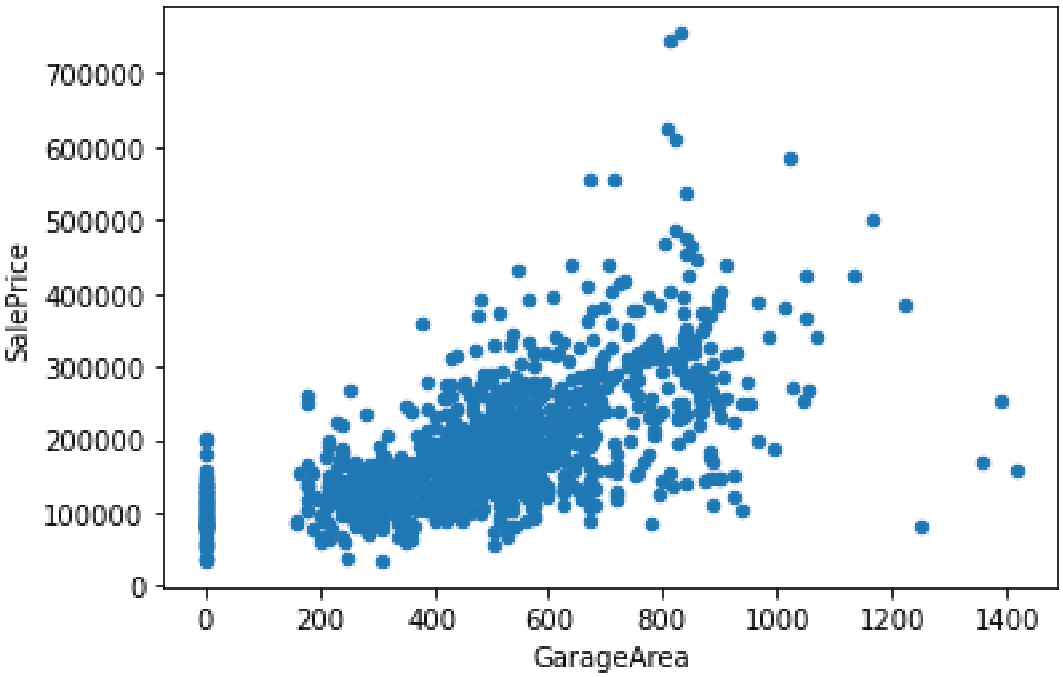

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
KitchenAbvGr

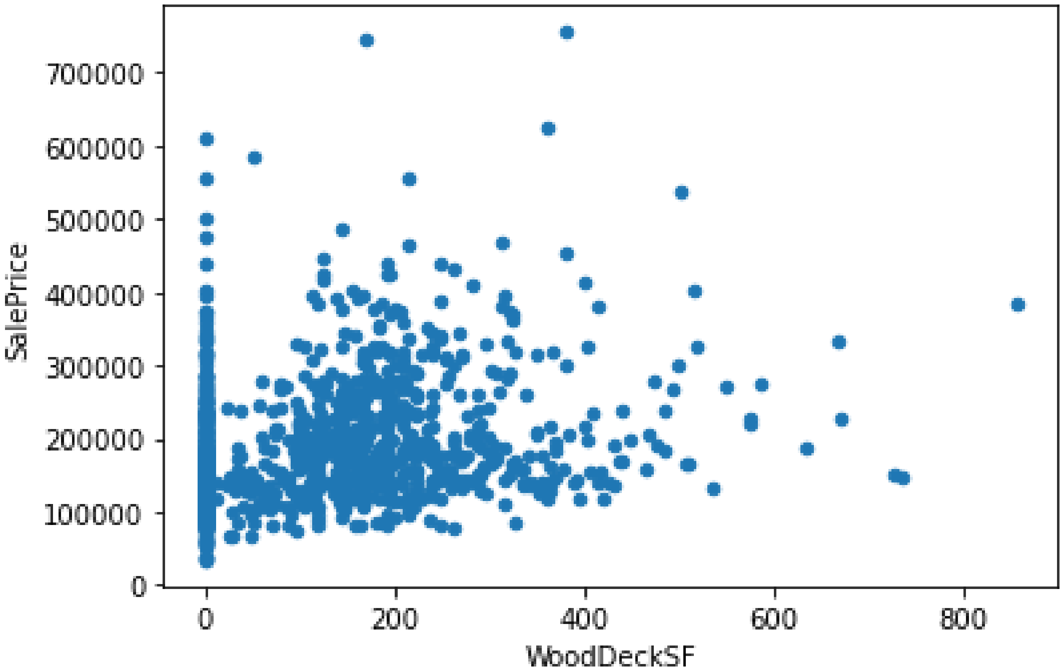

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
2
4
6
8
10
12
14
TotRmsAbvGrd

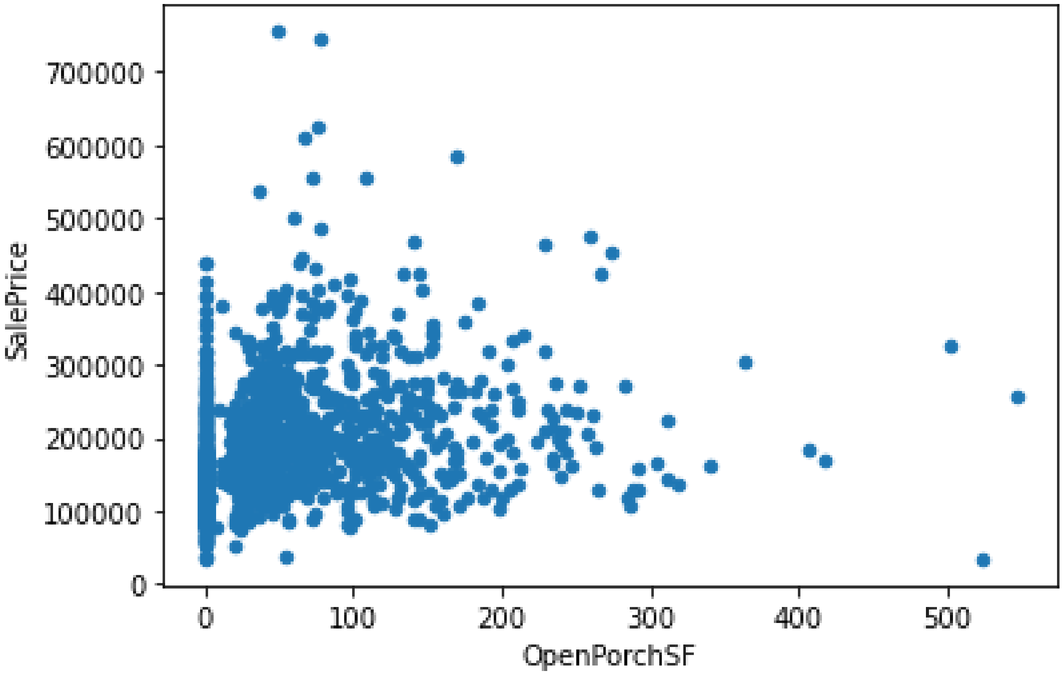

700000
600000
500000
400000
300000
200000
100000
0
SalePrice
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Fireplaces

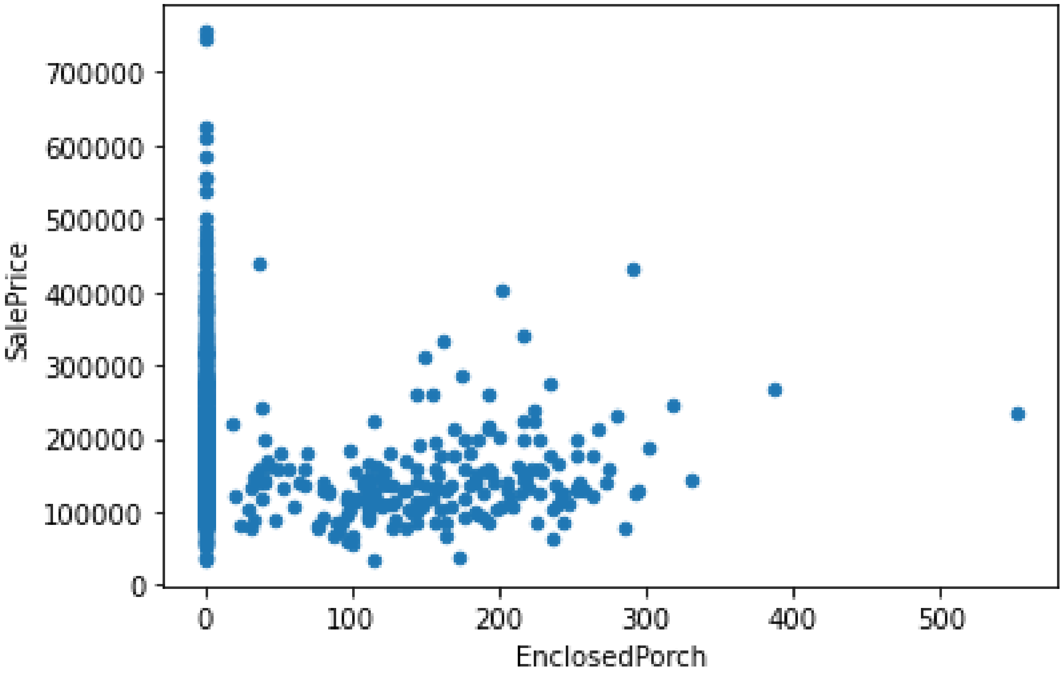

700000
600000
500000
400000
300000
200000
100000
0
SalePrice
1900
1920
1940
1960
1980
2000
GarageYrBlt

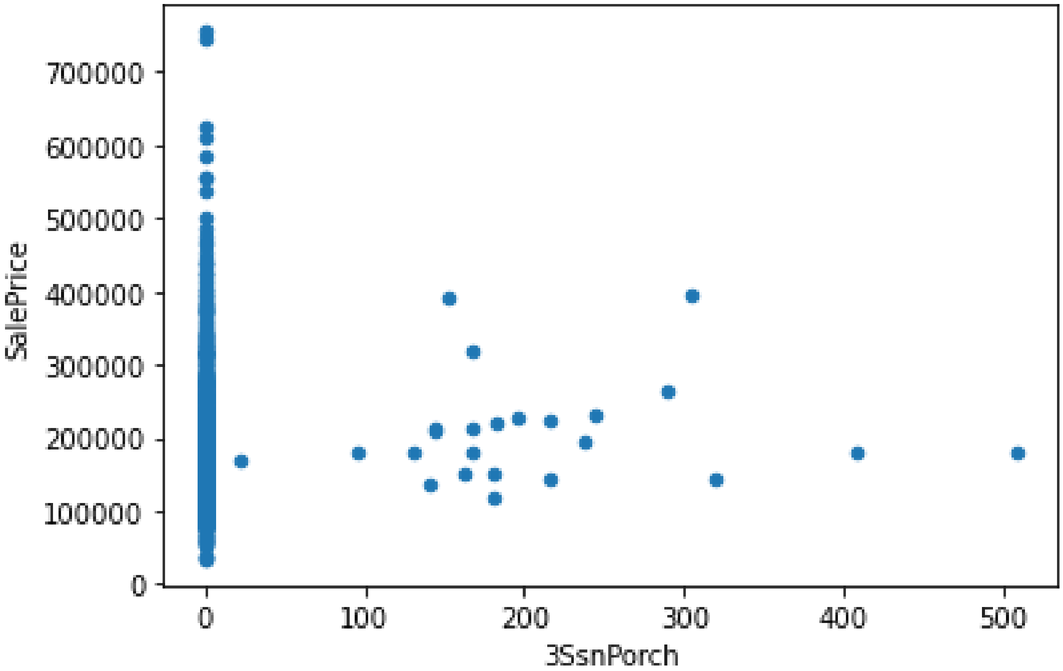

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
GarageCars

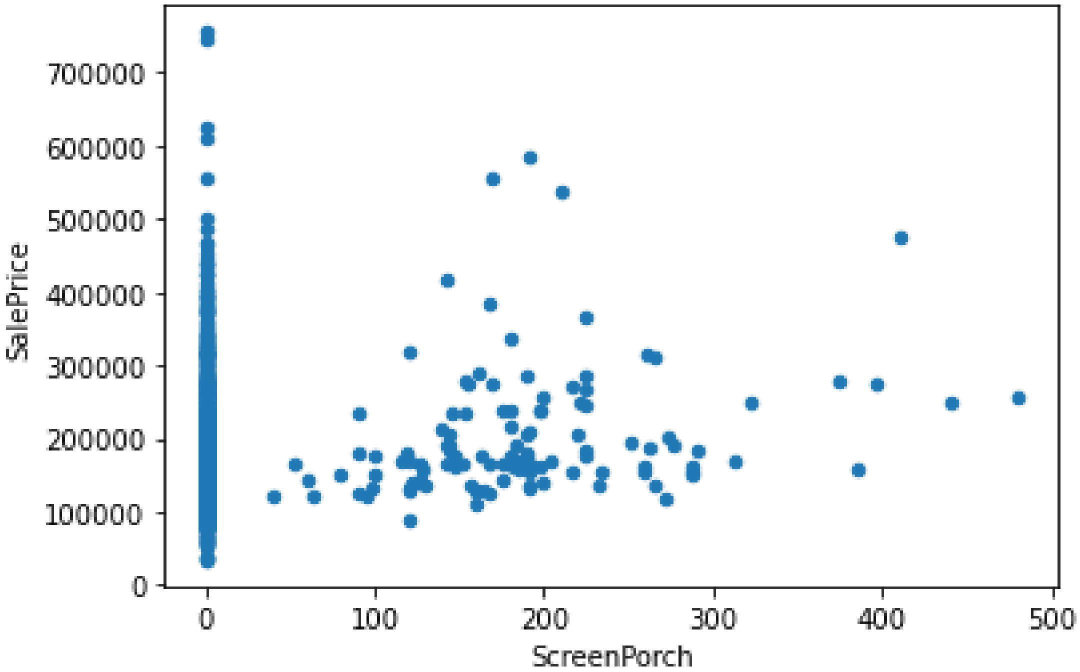

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
200
400
600
800
1000
1200
1400
GarageArea

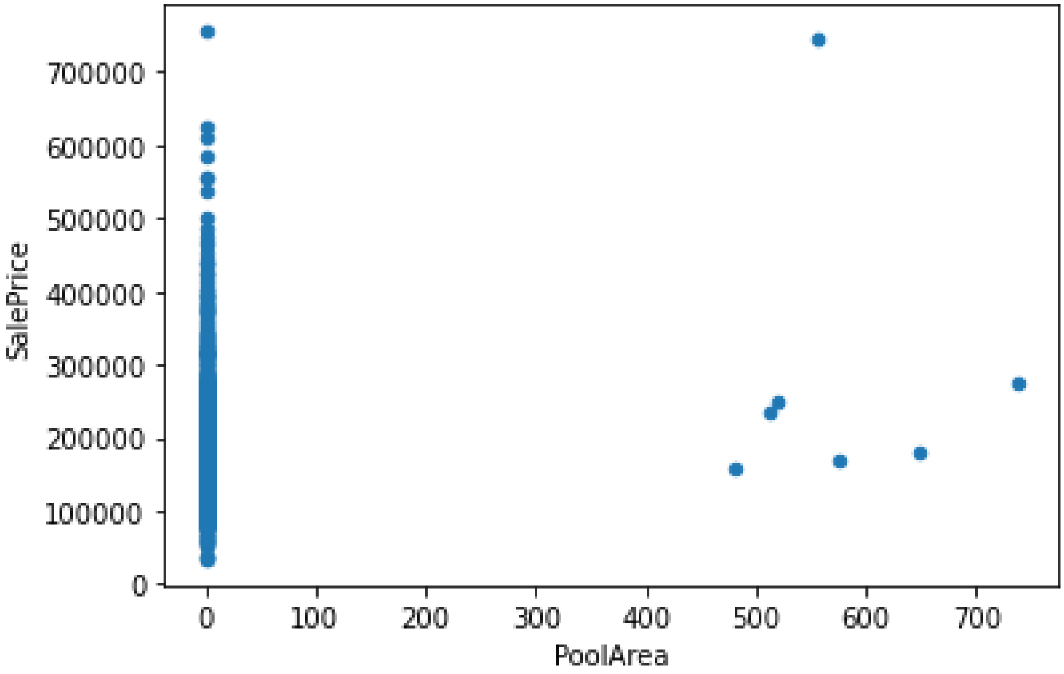

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
200
400
600
800
WoodDeckSF

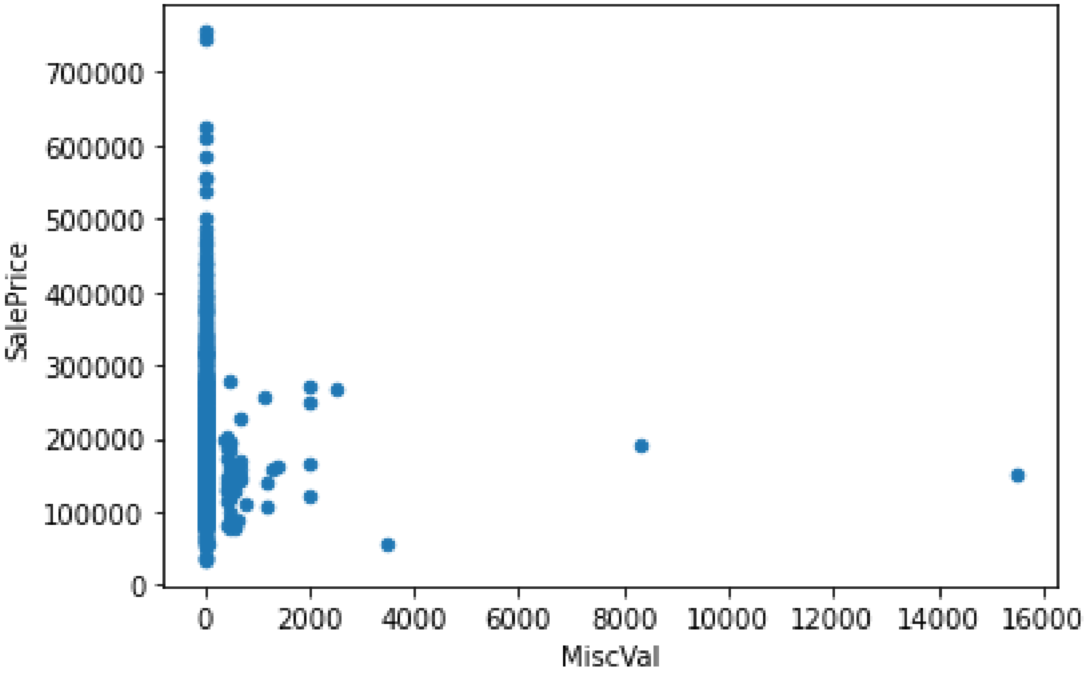

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
100
200
300
400
500
OpenPorchSF

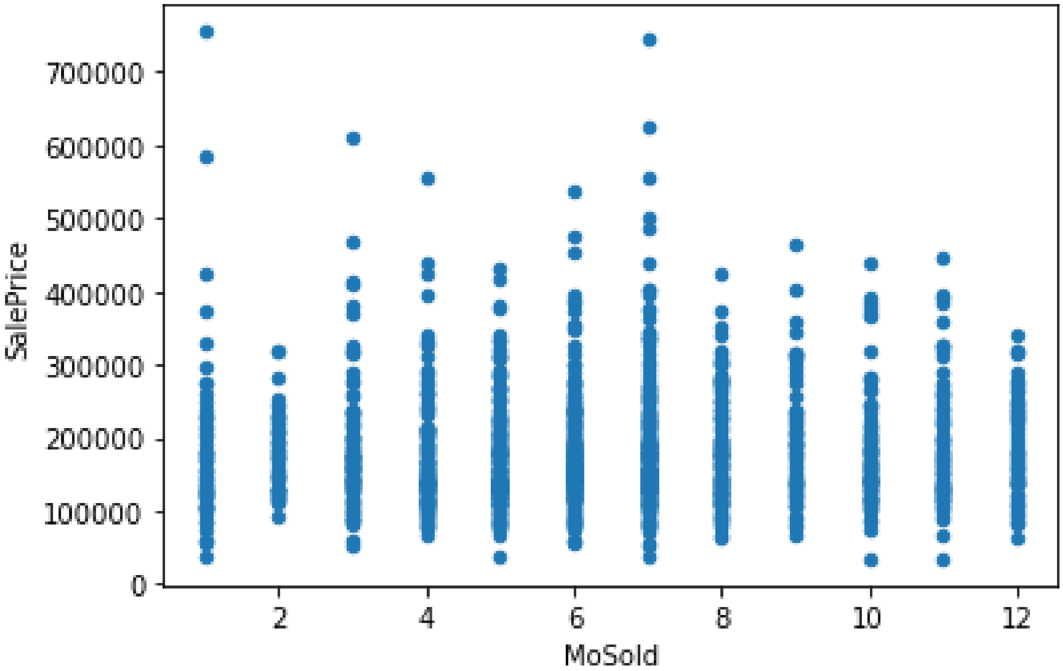

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
100
200
300
400
500
EnclosedPorch

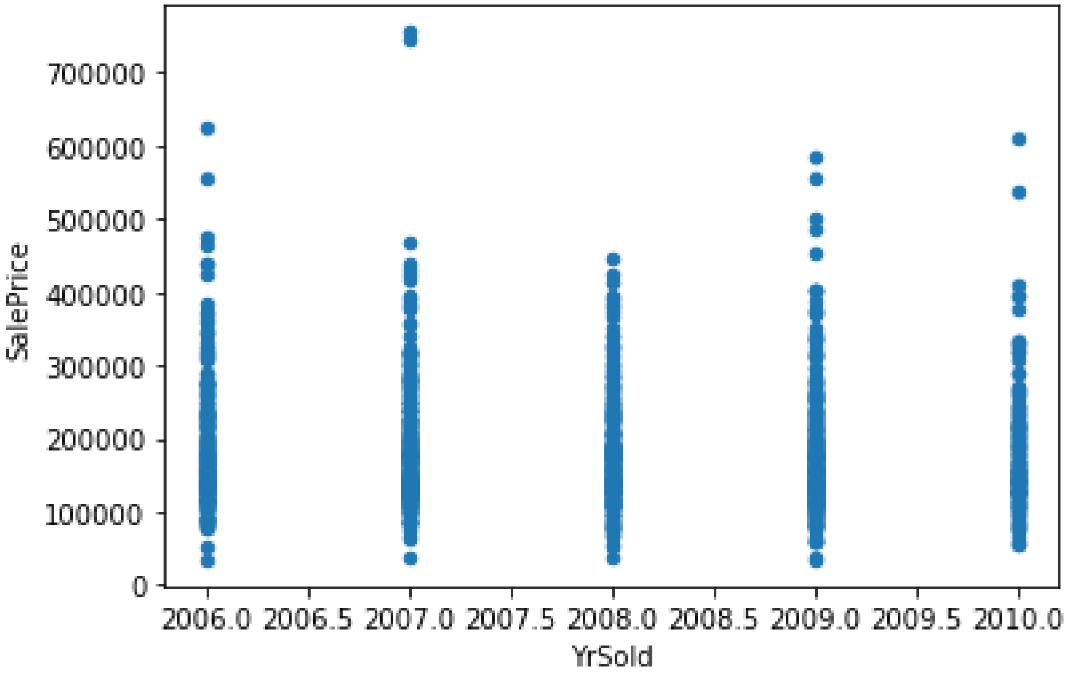

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
100
200
300
400
500
3SsnPorch


SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
100
200
300
400
500
ScreenPorch


SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
100
200
300
400
500
600
700
PoolArea

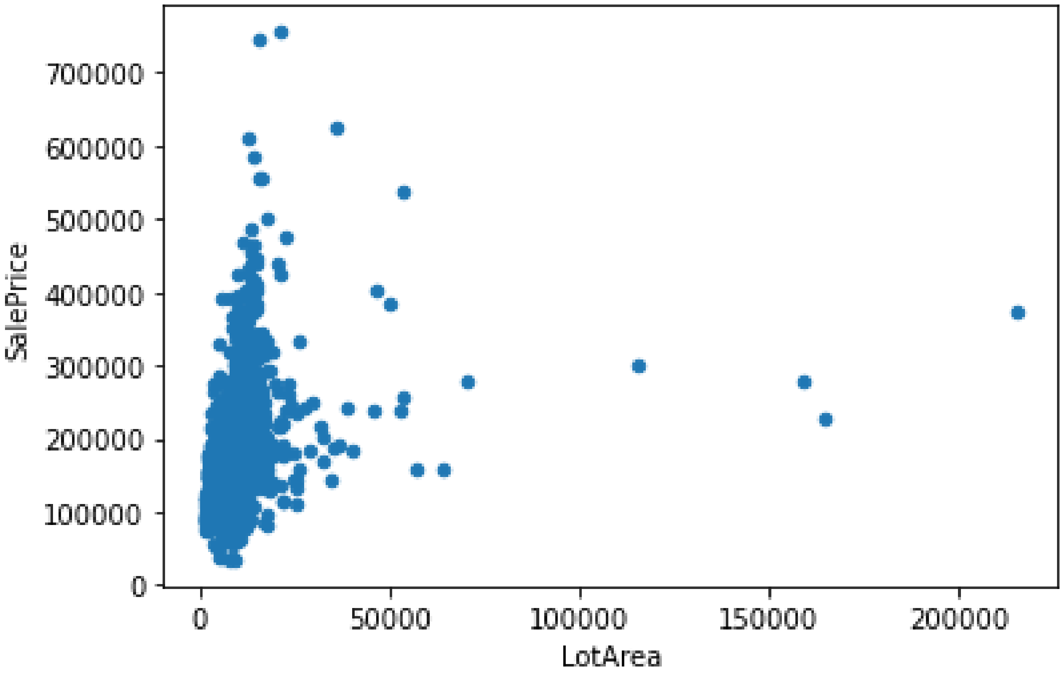

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
2000
4000
6000
8000
10000
12000
14000
16000
MiscVal


SalePrice
700000
600000
500000
400000
300000
200000
100000
0
2
4
6
8
10
12
MoSold

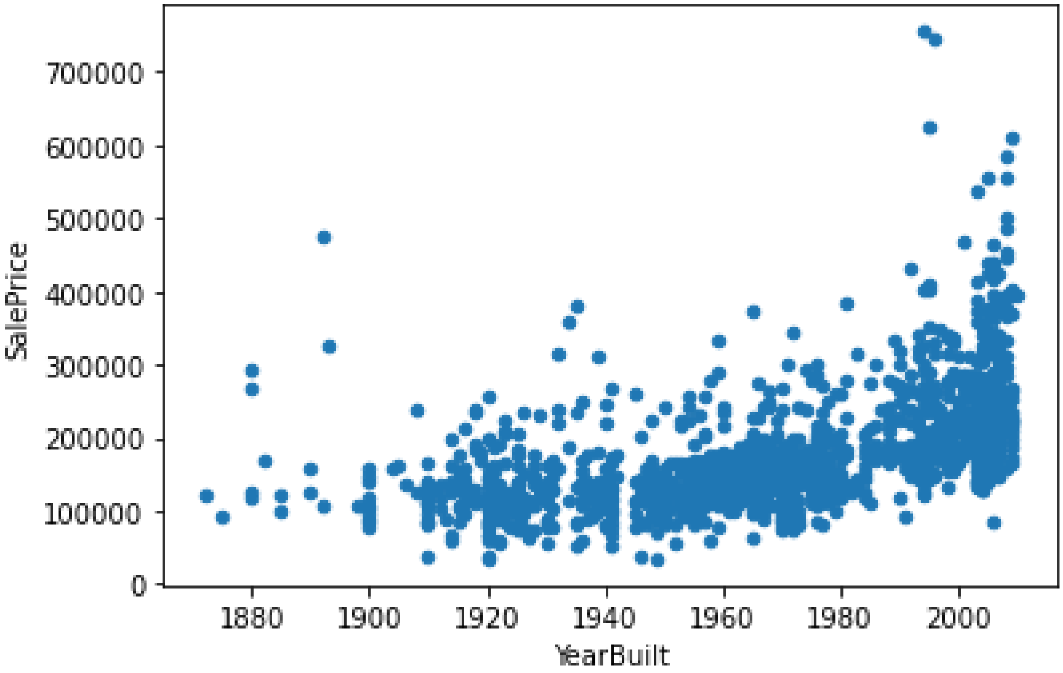


Selecionando apenas algumas características que aparentam realmente influenciar no modelo

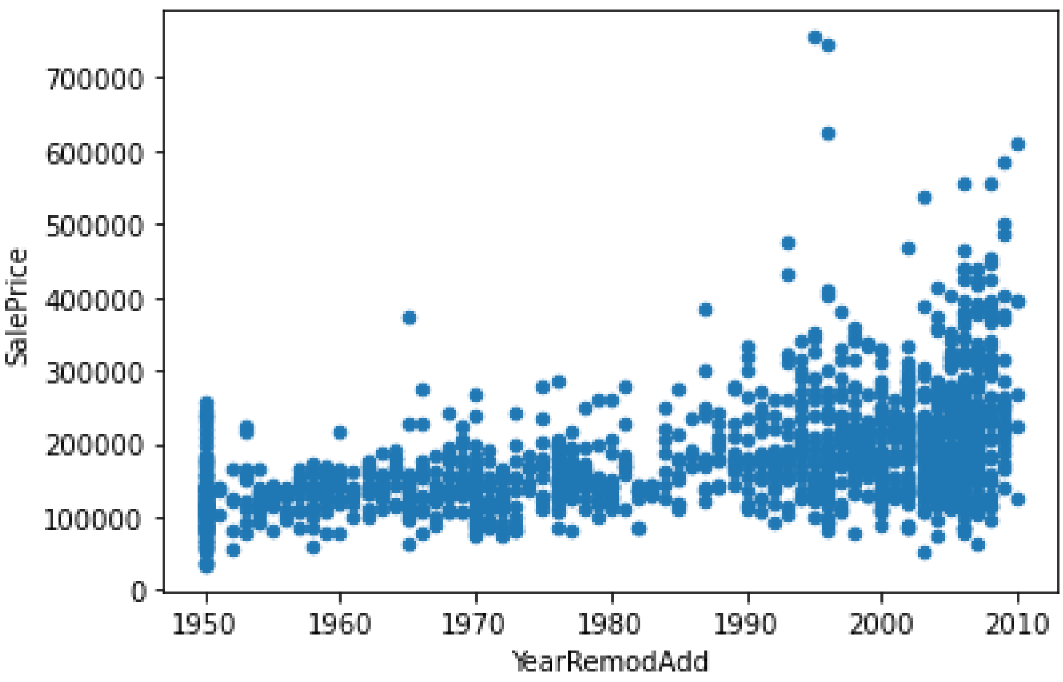
[51]:
```
dados = dados[['LotArea', 'YearBuilt', 'YearRemodAdd', 'BsmtFinSF1',␣
 ↪'BsmtFinSF2', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF',␣
 ↪'GrLivArea', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'SalePrice']]

#dados = dados[['GrLivArea', 'SalePrice']]

for i in range(len(dados.columns) -1):
    dados.plot.scatter(x=i, y='SalePrice')
```

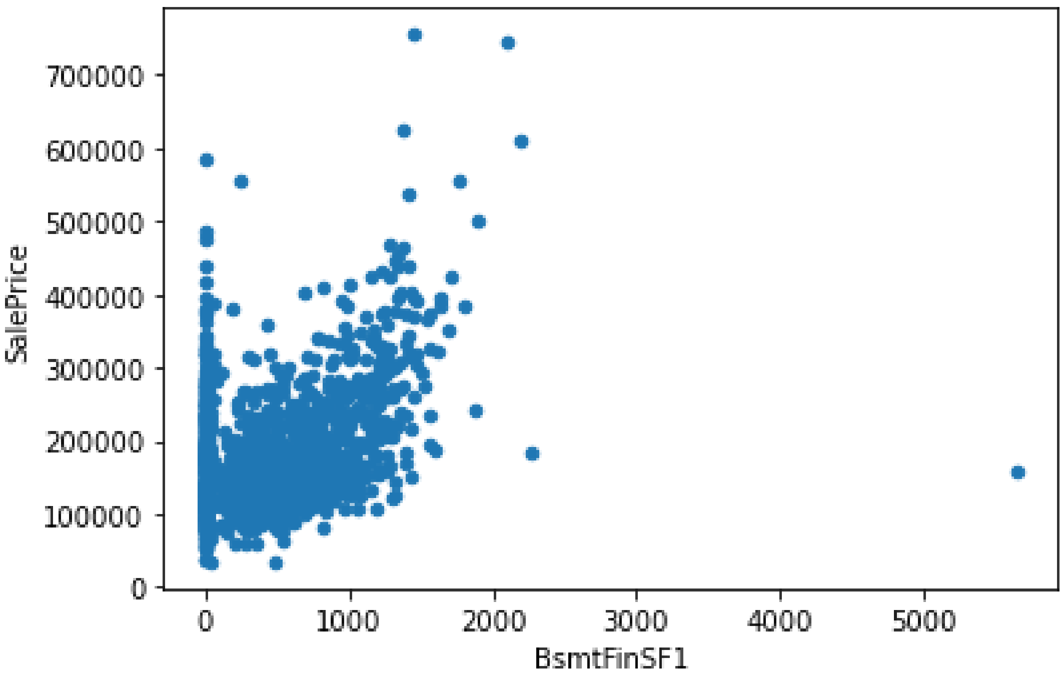


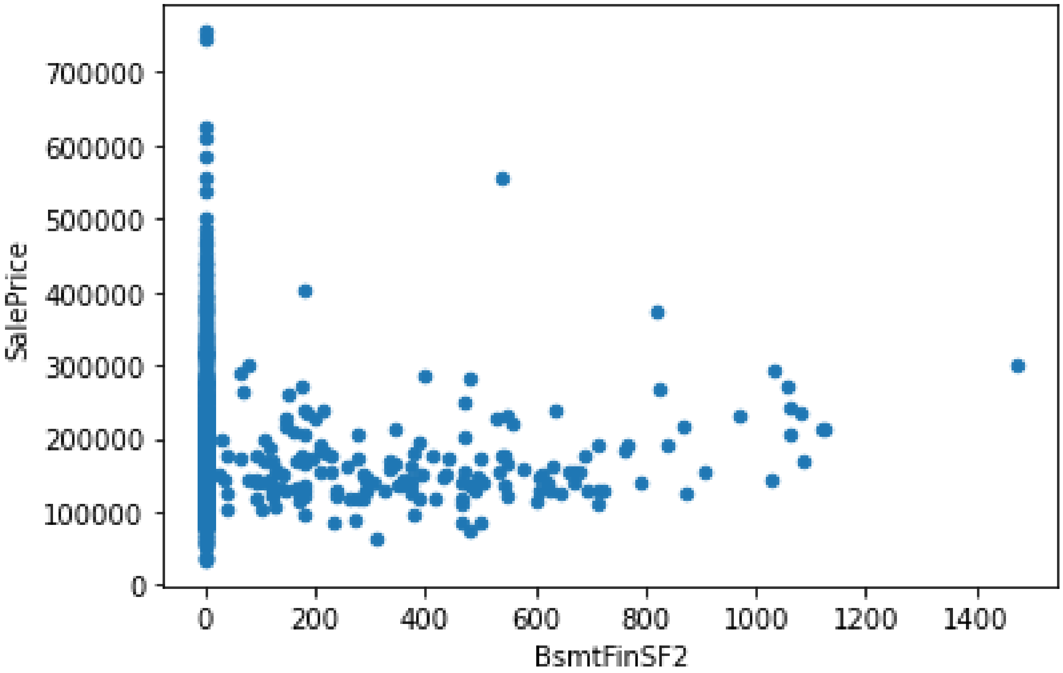
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
50000
100000
150000
200000
LotArea

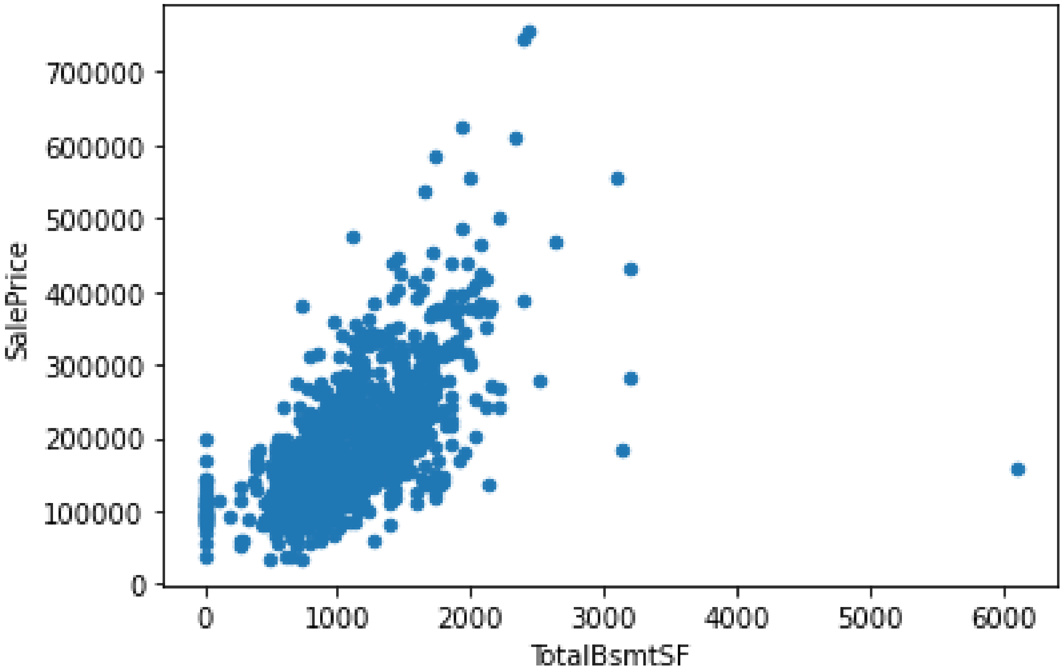


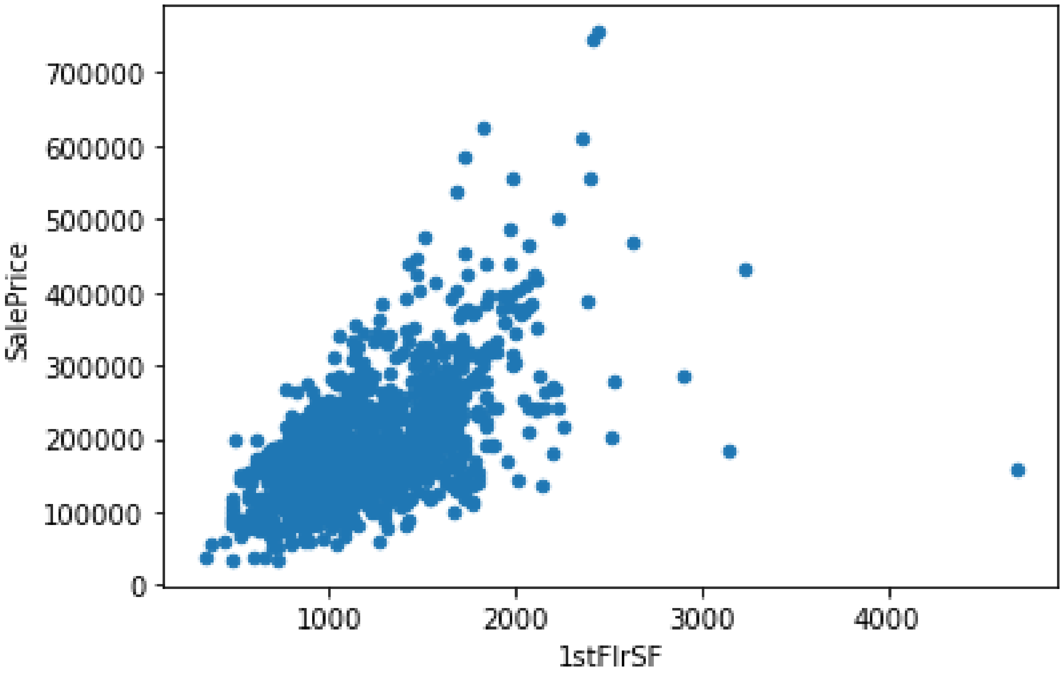
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
1880
1900
1920
1940
1960
1980
2000
YearBuilt

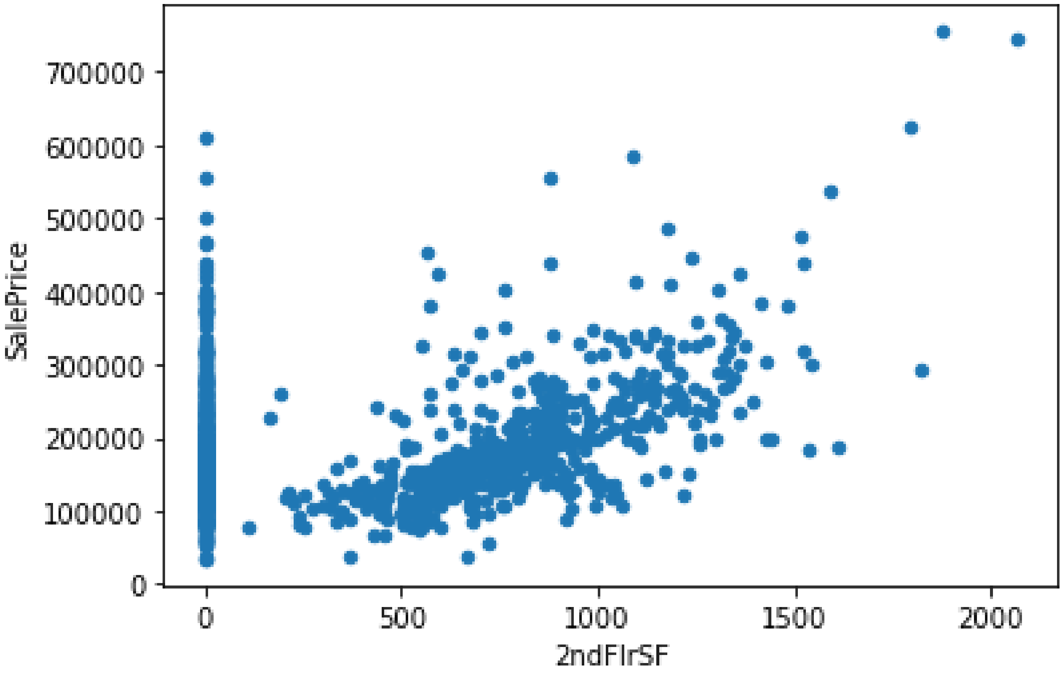

700000
600000
500000
400000
300000
200000
100000
0
SalePrice
1950
1960
1970
1980
1990
2000
2010
YearRemodAdd

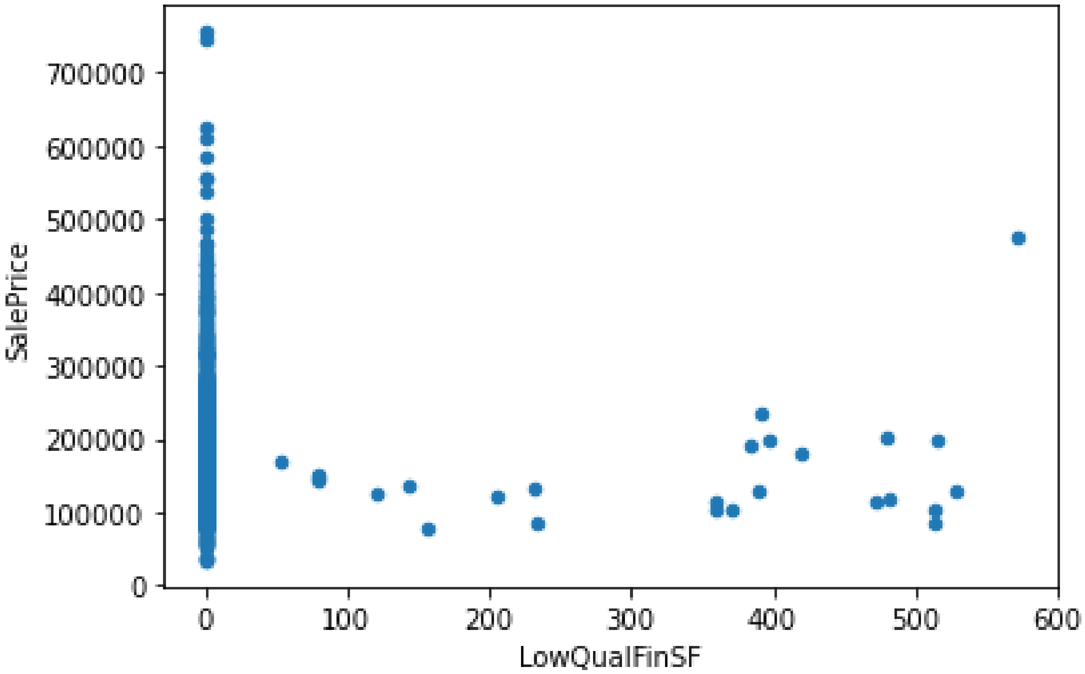

700000
600000
500000
400000
300000
200000
100000
0
SalePrice
0
1000
2000
3000
4000
5000
BsmtFinSF1

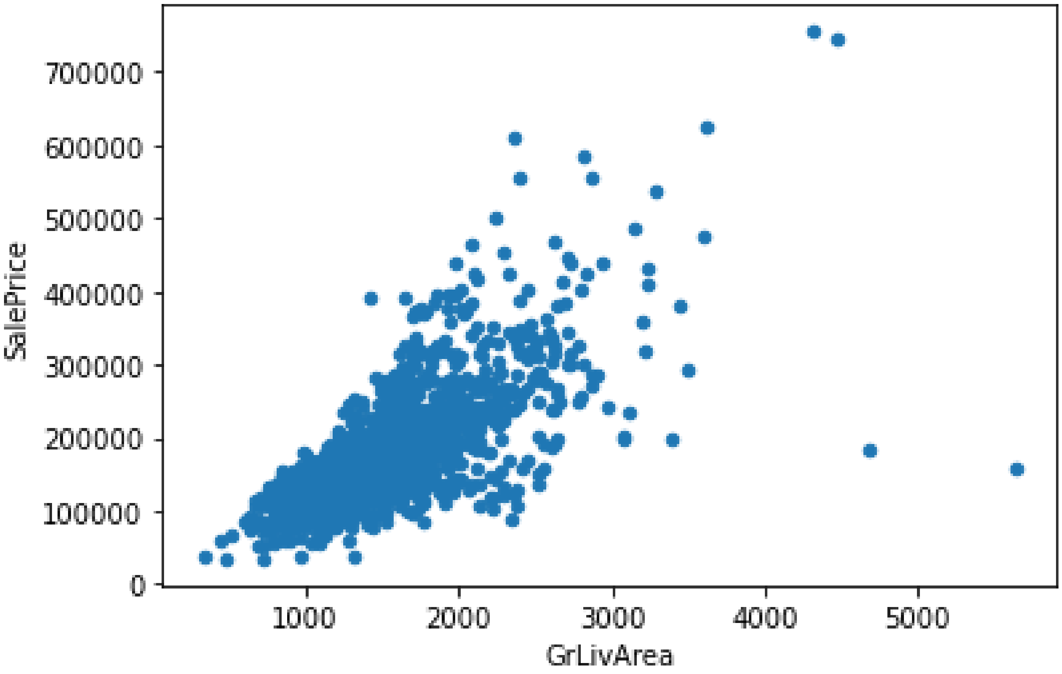

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
200
400
600
800
1000
1200
1400
BsmtFinSF2

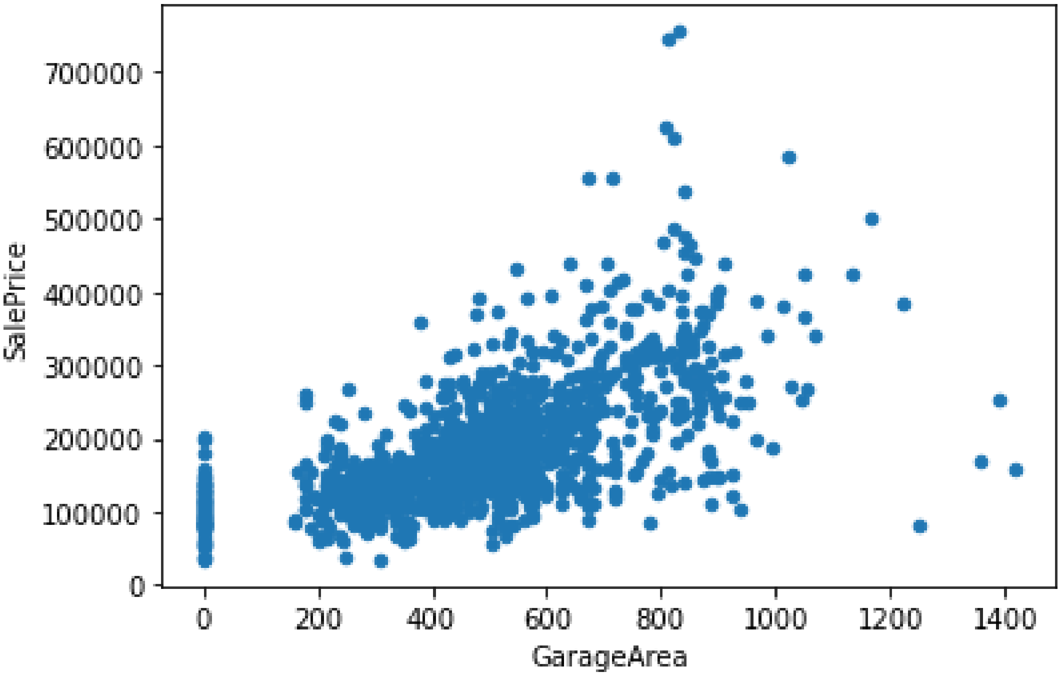

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
1000
2000
3000
4000
5000
6000
TotalBsmtSF

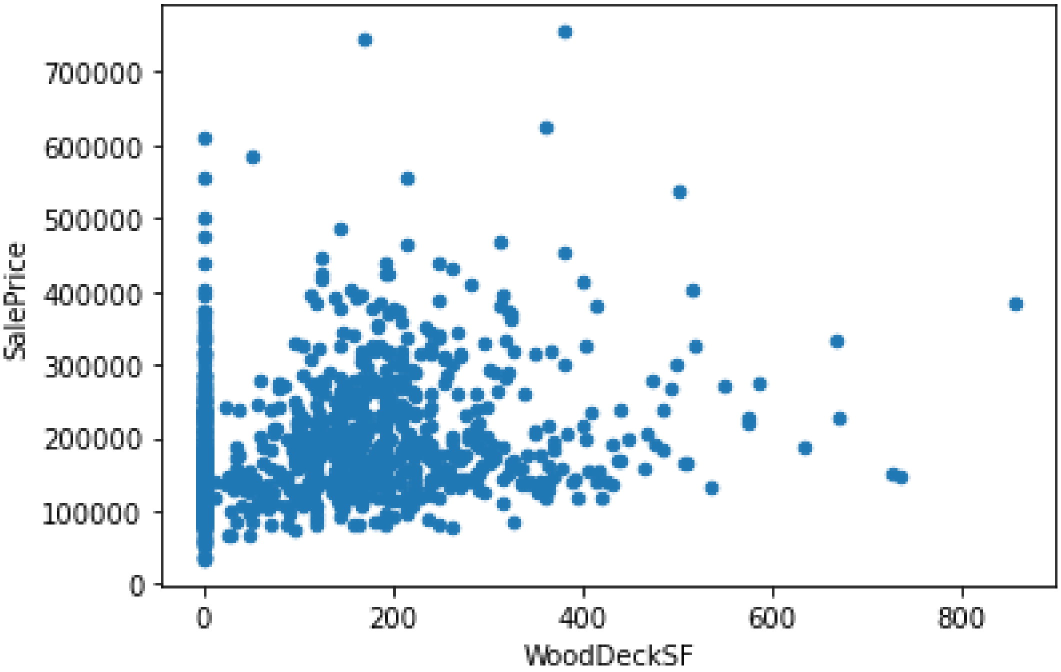

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
1000
2000
3000
4000
1stFlrSF

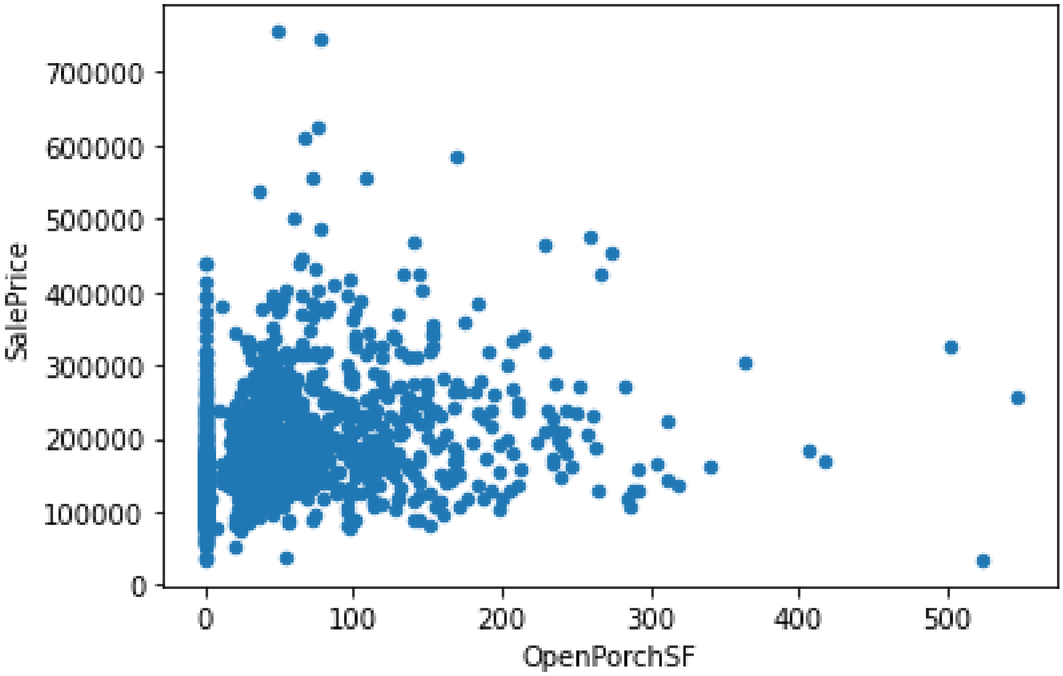

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
500
1000
1500
2000
2ndFlrSF


SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
100
200
300
400
500
600
LowQualFinSF


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


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


SalePrice
WoodDeckSF
0
100000
200000
300000
400000
500000
600000
700000
0
200
400
600
800



Treina a base e colhe os resultados

```
[52]: y = dados['SalePrice']
X = dados.drop('SalePrice', axis = 1)

modelo = LinearRegression()
modelo.fit(X, y)
ypred = modelo.predict(X)
print(mean_squared_error(y, ypred))
print(modelo.intercept_)
print(modelo.coef_)
```

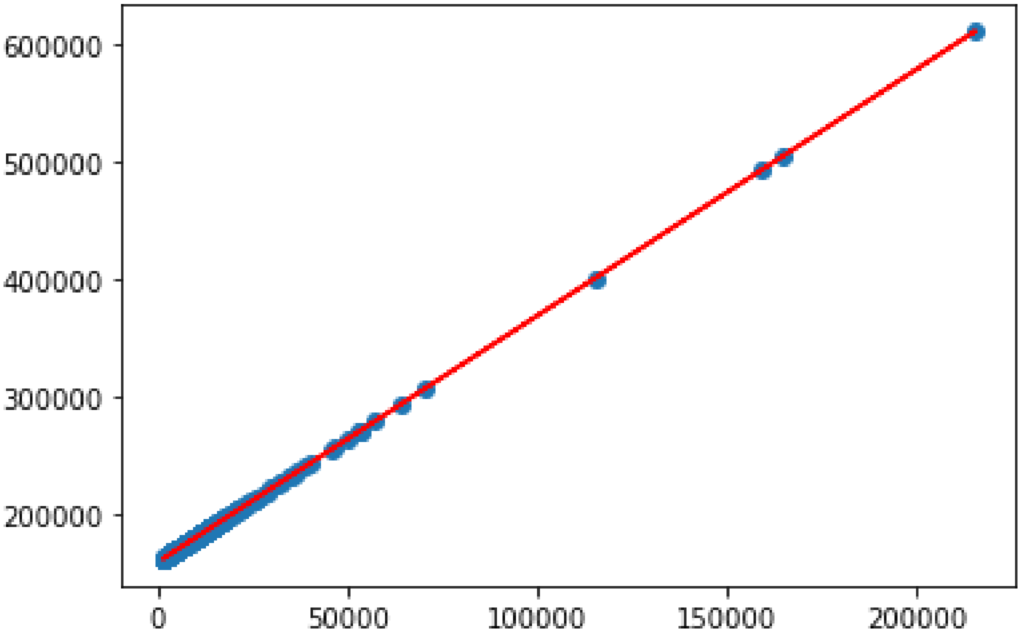
```
1657756395.789723
-2029319.208442636
[ 3.78041735e-01  4.58446085e+02  5.71251949e+02  1.40963221e+01
 -3.18799322e+00  2.74742801e+01  2.72462693e+01  2.58480117e+01
 -1.15609015e+01  4.15333795e+01  5.60067286e+01  3.05670480e+01
  1.36486018e+01]
```

```
[53]: # Xi = X.iloc[:, 0].to_frame()
# modelo = LinearRegression()
# modelo.fit(Xi, y)
# ypred = modelo.predict(Xi)
# print(mean_squared_error(y, ypred))
# print(modelo.intercept_)
# print(modelo.coef_)
# plt.scatter(Xi, ypred)
# plt.plot(Xi, ypred, 'r')
```

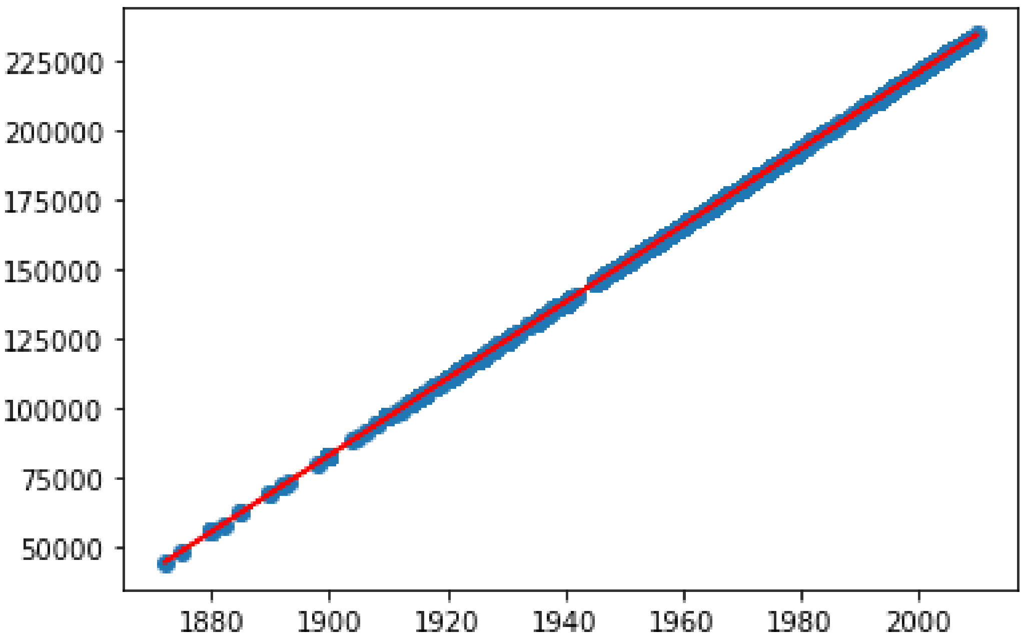
[54]:

```
for i in range(len(X.columns) -1):
    Xi = X.iloc[:, i].to_frame()
    modelo = LinearRegression()
    modelo.fit(Xi, y)
    ypred = modelo.predict(Xi)
    print(mean_squared_error(y, ypred))
    print(modelo.intercept_)
    print(modelo.coef_)
    plt.scatter(Xi, ypred)
    plt.plot(Xi, ypred, 'r')
    plt.show()
```

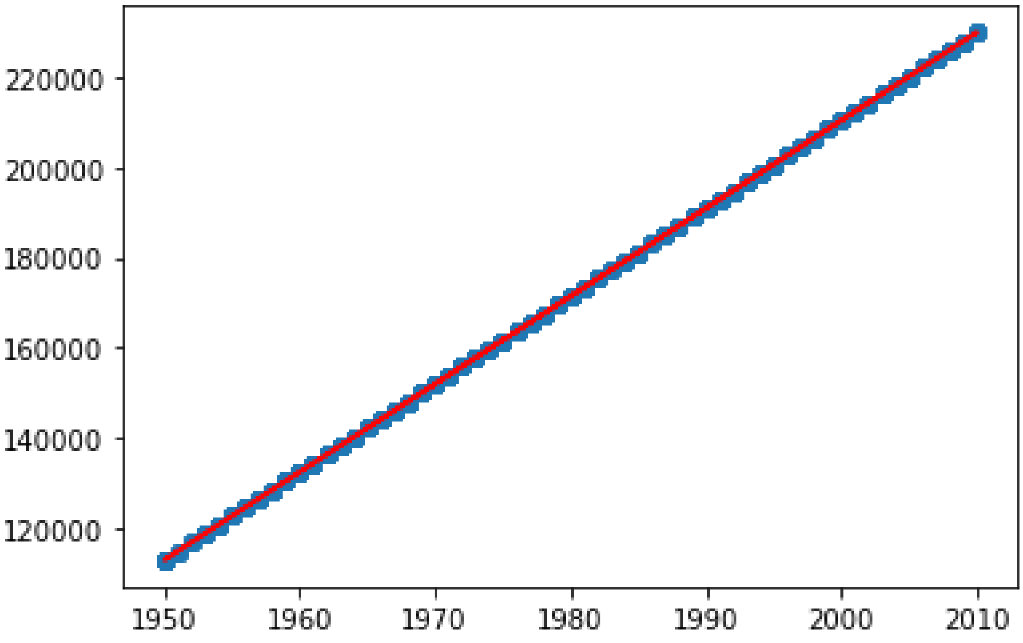
```
5867752122.509074
158836.1518968766
[2.09997195]
```



```
4582376228.725916
-2530308.2457323573
[1375.37346794]
```

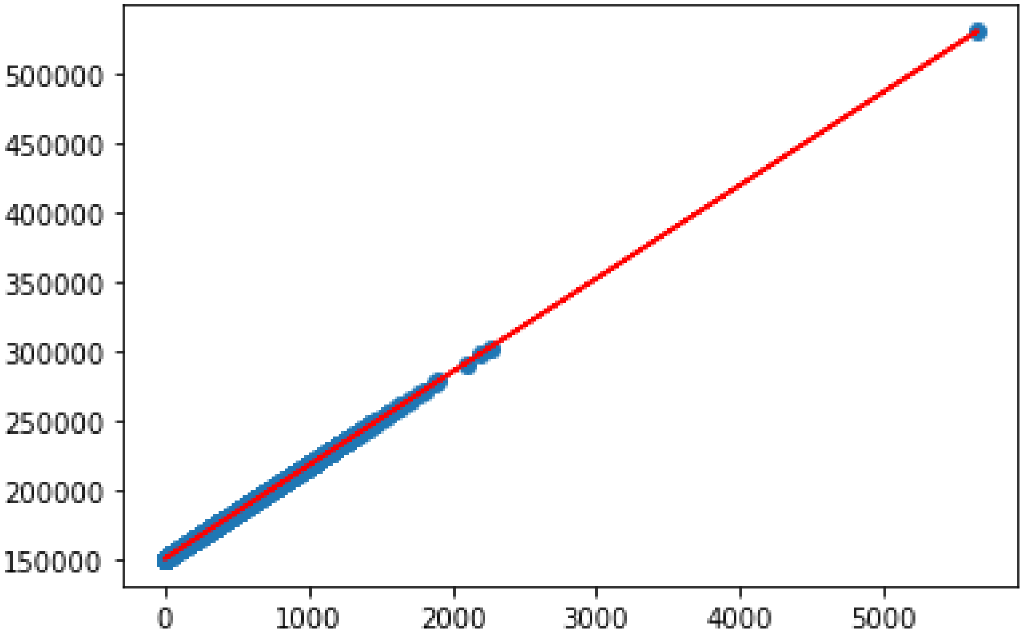


```
4684989128.859445
-3692146.1698673465
[1951.29940606]
```

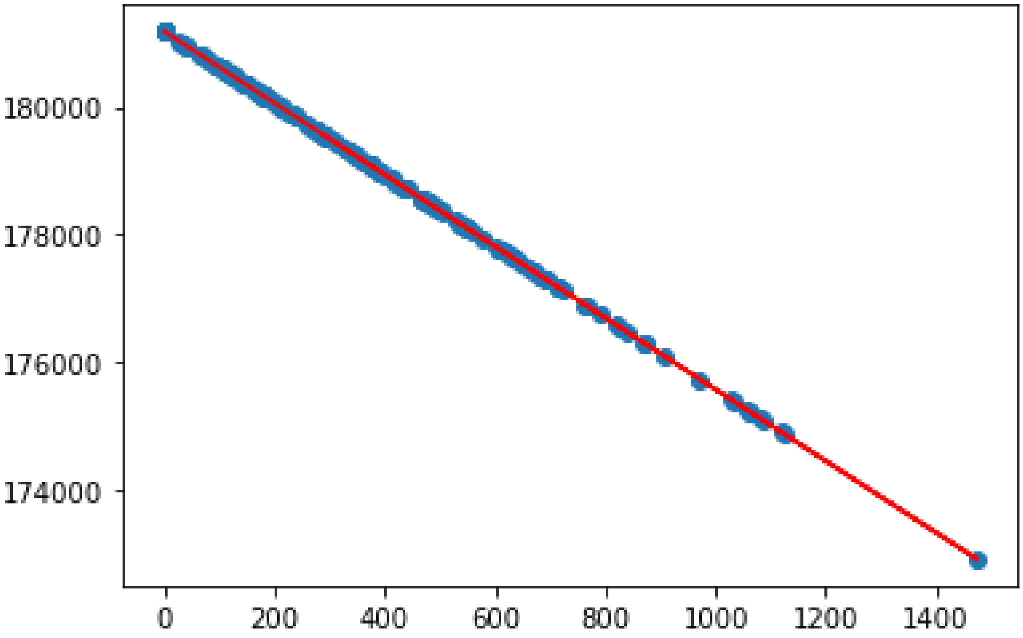


```
5365057232.040517
```

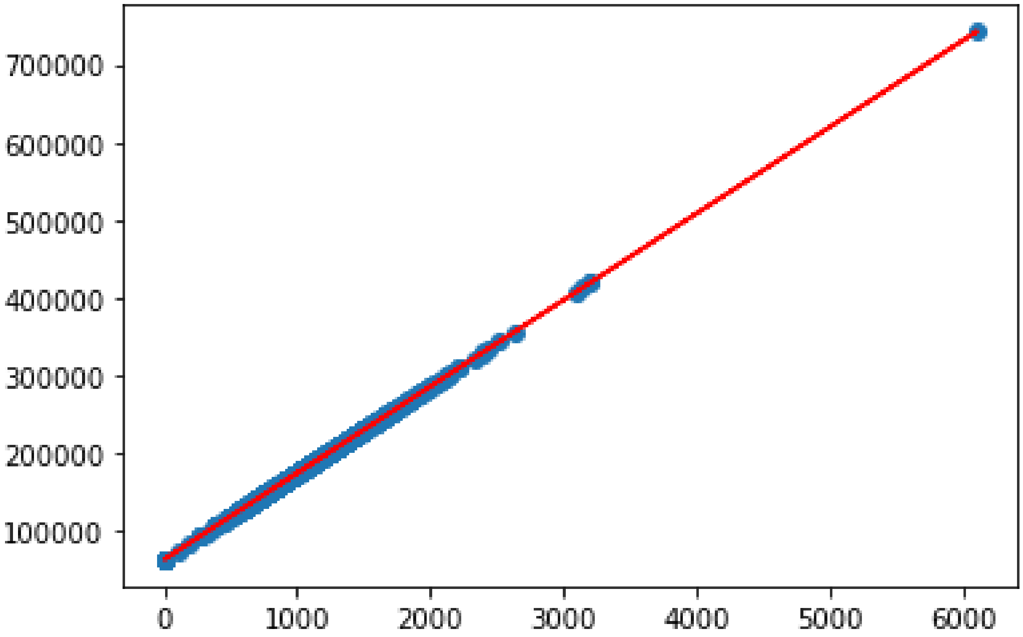
```
151061.5625256433
[67.30604049]
```



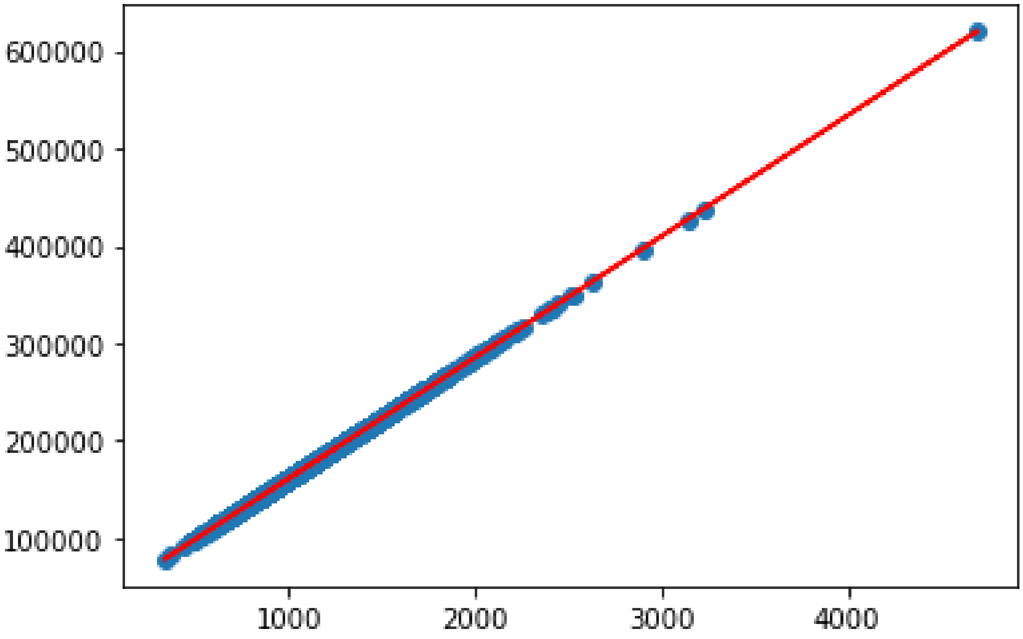
```
6305972098.107117
181182.02167533064
[-5.60321424]
```



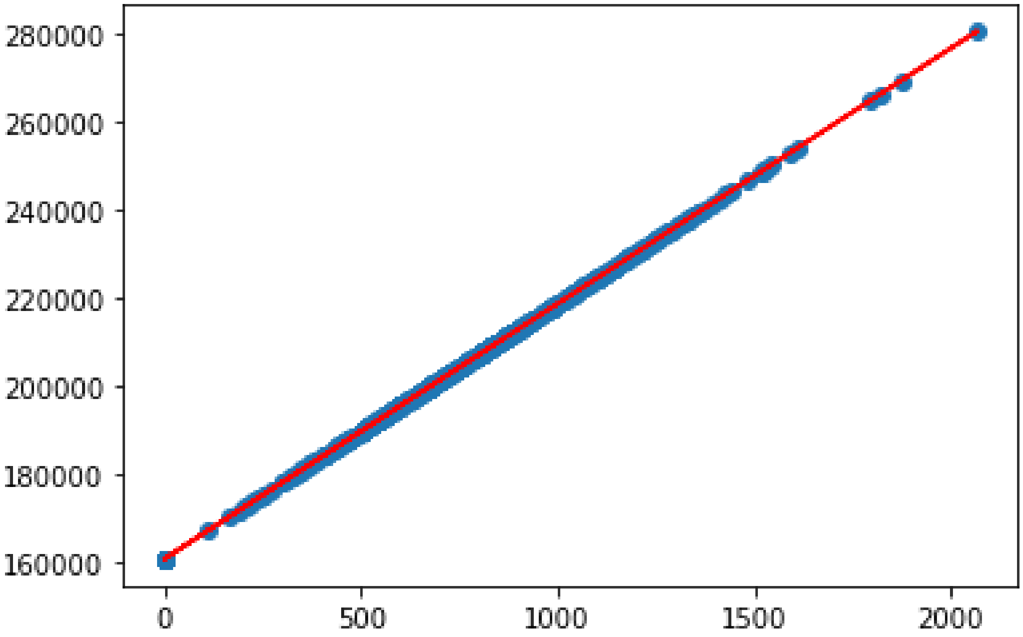
```
3932401923.829737
63430.62854550623
[111.10960369]
```



```
3991838509.3767366
36173.4467951213
[124.50062222]
```

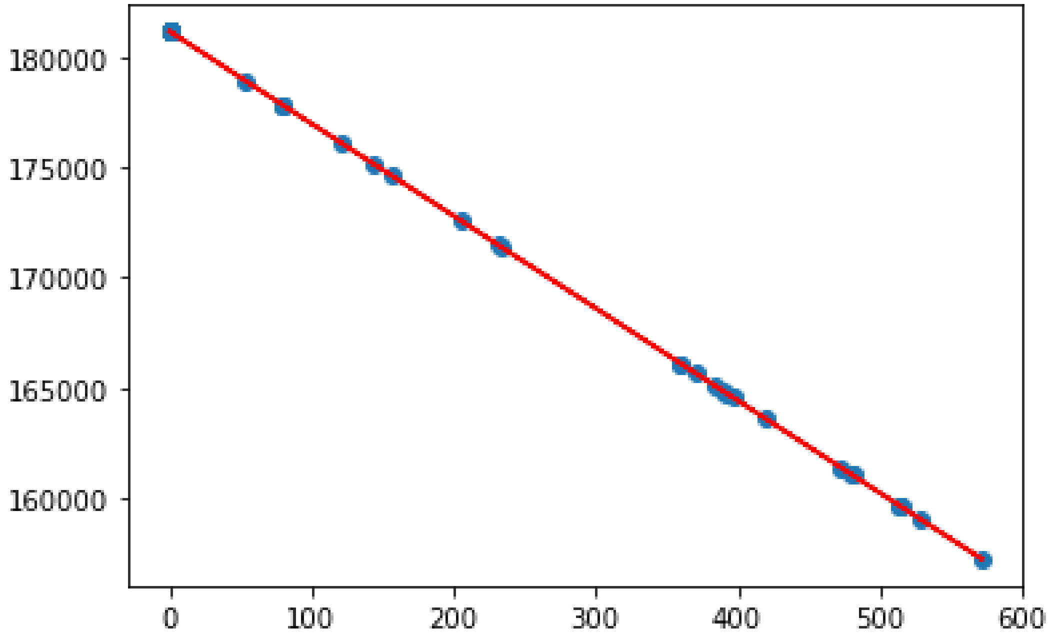


```
5663659636.543386
160755.8666562706
[58.11460255]
```

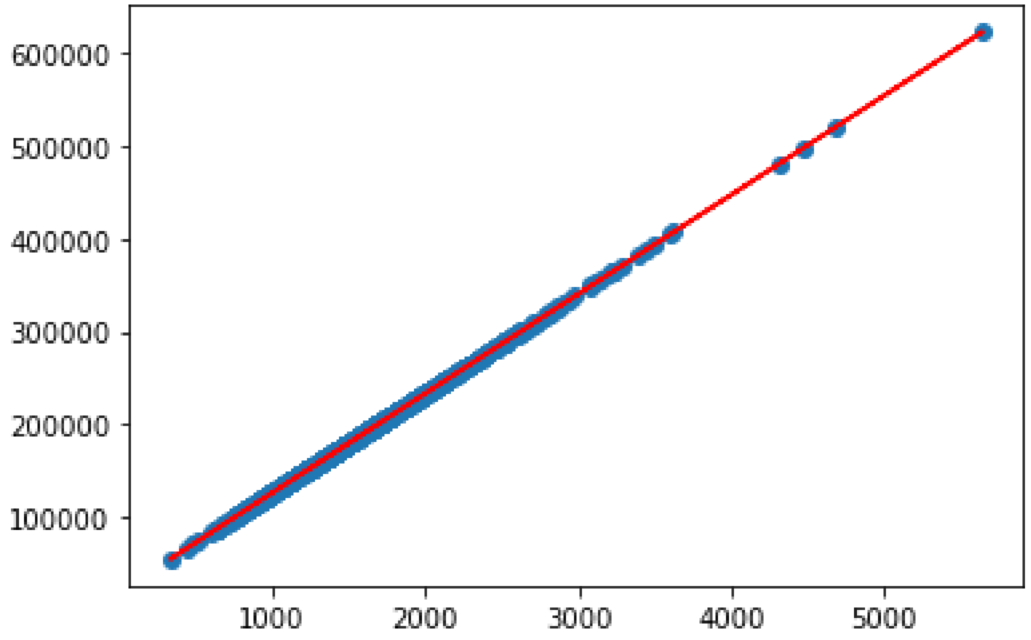


```
6302653388.721356
```

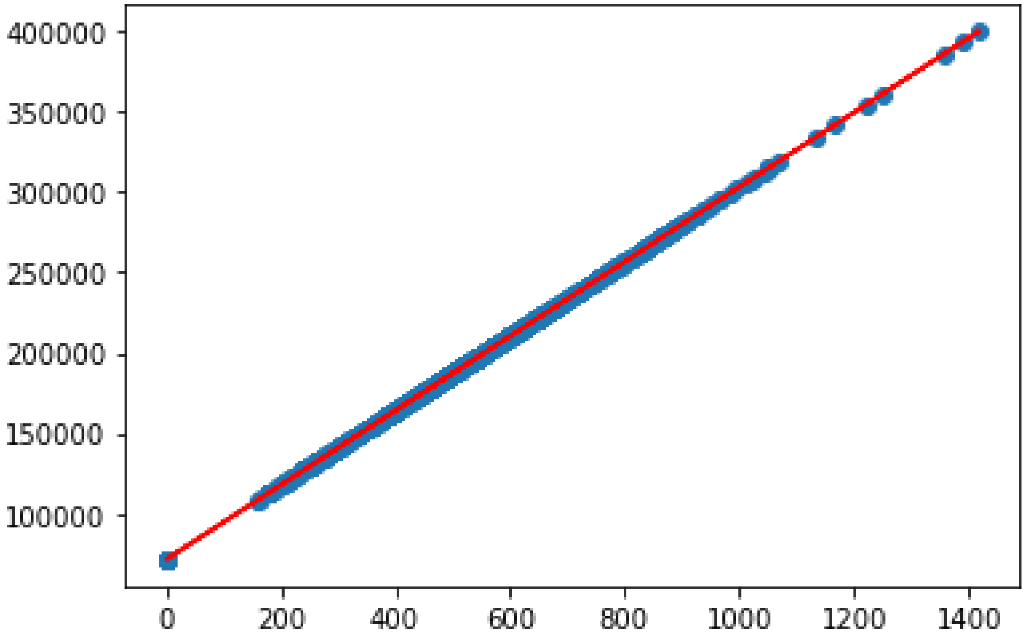
```
181165.70963461525
[-41.83640766]
```



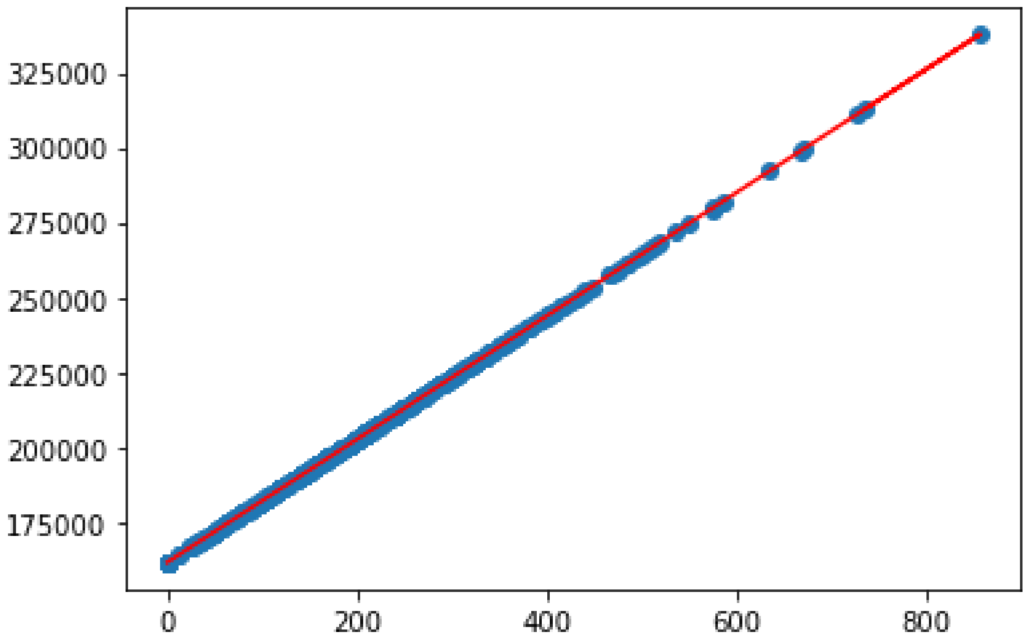
```
3139843209.6665273
18569.02585648728
[107.13035897]
```



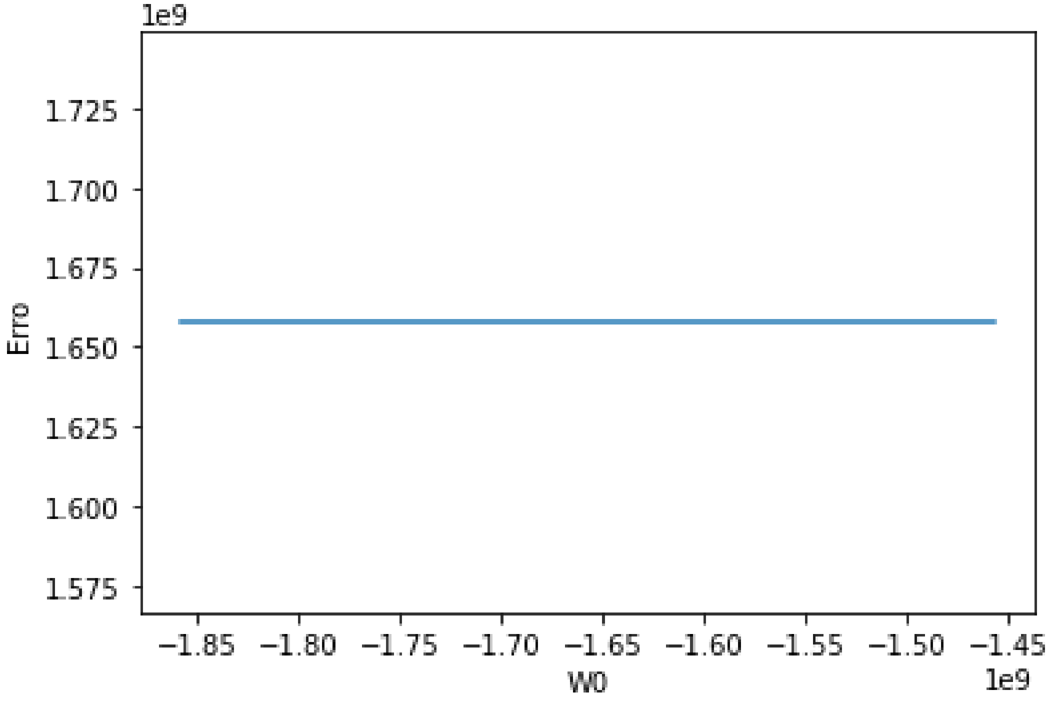
```
3855549505.982716
71357.42140747685
[231.64561451]
```

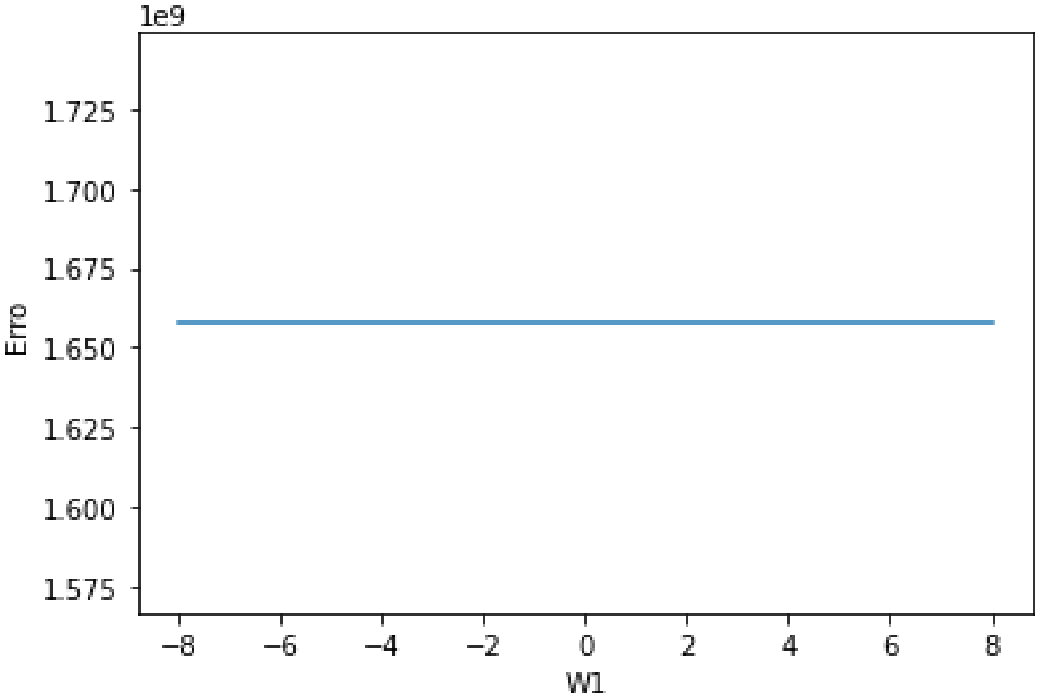


```
5643036403.904909
161542.59764040346
[205.62042374]
```



[55]:
```
def z_function(W0, W1):
    modelo = LinearRegression()
    modelo.fit(X, y)
    Erro = np.empty(W0.shape)
    for j in range(Erro.shape[0]):
        for k in range(Erro.shape[1]):
            ypred = modelo.predict(X)
            mse = mean_squared_error(y, ypred)
            Erro[j][k] = mse
    return Erro

w0 = np.linspace(-1857756395, -1457756395, 30)
w1 = np.linspace(-8, 8, 30)

W0, W1 = np.meshgrid(w0, w1)

Erro = z_function(W0, W1)

plt.ylabel("Erro")
plt.xlabel("W0")
i=8
plt.plot(W0[i,:],Erro[i,:])
plt.show()
plt.ylabel("Erro")
plt.xlabel("W1")
i=8
plt.plot(W1[:,i],Erro[:,i])
plt.show()
```




1e9
1.725
1.700
1.675
1.650
1.625
1.600
1.575
Erro
−8
−6
−4
−2
0
2
4
6
8
W1