lasso_regression_housing price

May 9, 2019

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
#from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV
from sklearn import metrics

import os

# hide warnings
import warnings
```

In [10]:
```
path = '../input/'
#path = 'dataset/'
house = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
print('Number of rows and columns in train dataset:', train.shape)
print('Number of rows and columns in test dataset:', test.shape)
```

```
('Number of rows and columns in train dataset:', (1460, 81))
('Number of rows and columns in test dataset:', (1459, 80))
```

In [13]:
```
# head
house.head()
```

Out[13]:
```
   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]
```

In [14]:
```
# train Data shape
print('train Data Shape',train.shape)
# Test data shape
print('test data shape',test.shape)
```

```
('train Data Shape', (1460, 81))
('test data shape', (1459, 80))
```

In [15]:
```
house.describe()
```

Out[15]:
```
                Id    MSSubClass  LotFrontage        LotArea  OverallQual  \
count  1460.000000  1460.000000  1201.000000    1460.000000  1460.000000
mean    730.500000    56.897260    70.049958   10516.828082     6.099315
std     421.610009    42.300571    24.284752    9981.264932     1.382997
min       1.000000    20.000000    21.000000    1300.000000     1.000000
25%     365.750000    20.000000    59.000000    7553.500000     5.000000
50%     730.500000    50.000000    69.000000    9478.500000     6.000000
75%    1095.250000    70.000000    80.000000   11601.500000     7.000000
max    1460.000000   190.000000   313.000000  215245.000000    10.000000

       OverallCond    YearBuilt  YearRemodAdd   MasVnrArea   BsmtFinSF1  ...  \
count  1460.000000  1460.000000   1460.000000  1452.000000  1460.000000  ...
mean      5.575342  1971.267808   1984.865753   103.685262   443.639726  ...
std       1.112799    30.202904     20.645407   181.066207   456.098091  ...
min       1.000000  1872.000000   1950.000000     0.000000     0.000000  ...
25%       5.000000  1954.000000   1967.000000     0.000000     0.000000  ...
50%       5.000000  1973.000000   1994.000000     0.000000   383.500000  ...
75%       6.000000  2000.000000   2004.000000   166.000000   712.250000  ...
max       9.000000  2010.000000   2010.000000  1600.000000  5644.000000  ...

        WoodDeckSF  OpenPorchSF  EnclosedPorch    3SsnPorch  ScreenPorch  \
count  1460.000000  1460.000000    1460.000000  1460.000000  1460.000000
mean     94.244521    46.660274      21.954110     3.409589    15.060959
```

```
std     125.338794    66.256028     61.119149    29.317331    55.757415
min       0.000000     0.000000      0.000000     0.000000     0.000000
25%       0.000000     0.000000      0.000000     0.000000     0.000000
50%       0.000000    25.000000      0.000000     0.000000     0.000000
75%     168.000000    68.000000      0.000000     0.000000     0.000000
max     857.000000   547.000000    552.000000   508.000000   480.000000

          PoolArea       MiscVal       MoSold       YrSold      SalePrice
count  1460.000000   1460.000000  1460.000000  1460.000000    1460.000000
mean      2.758904     43.489041     6.321918  2007.815753  180921.195890
std      40.177307    496.123024     2.703626     1.328095   79442.502883
min       0.000000      0.000000     1.000000  2006.000000   34900.000000
25%       0.000000      0.000000     5.000000  2007.000000  129975.000000
50%       0.000000      0.000000     6.000000  2008.000000  163000.000000
75%       0.000000      0.000000     8.000000  2009.000000  214000.000000
max     738.000000  15500.000000    12.000000  2010.000000  755000.000000

[8 rows x 38 columns]
```

In [16]:
```python
# all numeric (float and int) variables in the dataset
house_numeric = house.select_dtypes(include=['float64', 'int64'])
house_numeric.head()
```

Out[16]:
```
   Id  MSSubClass  LotFrontage  LotArea  OverallQual  OverallCond  YearBuilt  \
0   1          60         65.0     8450            7            5       2003
1   2          20         80.0     9600            6            8       1976
2   3          60         68.0    11250            7            5       2001
3   4          70         60.0     9550            7            5       1915
4   5          60         84.0    14260            8            5       2000

   YearRemodAdd  MasVnrArea  BsmtFinSF1  ...  WoodDeckSF  OpenPorchSF  \
0          2003       196.0         706  ...           0           61
1          1976         0.0         978  ...         298            0
2          2002       162.0         486  ...           0           42
3          1970         0.0         216  ...           0           35
4          2000       350.0         655  ...         192           84

   EnclosedPorch  3SsnPorch  ScreenPorch  PoolArea  MiscVal  MoSold  YrSold  \
0              0          0            0         0        0       2    2008
1              0          0            0         0        0       5    2007
2              0          0            0         0        0       9    2008
3            272          0            0         0        0       2    2006
4              0          0            0         0        0      12    2008

   SalePrice
0     208500
1     181500
2     223500
```

```
3     140000
4     250000

[5 rows x 38 columns]
```

```
In [18]: #Check for missing data & list them in train and test sets
         datasetHasNan = False
         if train.count().min() == train.shape[0] and test.count().min() == test.shape[0] :
             print('We do not need to worry about missing values.')
         else:
             datasetHasNan = True
             print('yes, we have missing values')

         # now list items
         print('--'*40)
         if datasetHasNan == True:
             nas = pd.concat([train.isnull().sum(), test.isnull().sum()], axis=1, keys=['Train
             print('Nan in the data sets')
             print(nas[nas.sum(axis=1) > 0])
```

```
yes, we have missing values
--------------------------------------------------------------------------------
Nan in the data sets
              Train Dataset  Test Dataset
Alley                  1369        1352.0
BsmtCond                 37          45.0
BsmtExposure             38          44.0
BsmtFinSF1                0           1.0
BsmtFinSF2                0           1.0
BsmtFinType1             37          42.0
BsmtFinType2             38          42.0
BsmtFullBath              0           2.0
BsmtHalfBath              0           2.0
BsmtQual                 37          44.0
BsmtUnfSF                 0           1.0
Electrical                1           0.0
Exterior1st               0           1.0
Exterior2nd               0           1.0
Fence                  1179        1169.0
FireplaceQu             690         730.0
Functional                0           2.0
GarageArea                0           1.0
GarageCars                0           1.0
GarageCond               81          78.0
GarageFinish             81          78.0
GarageQual               81          78.0
GarageType               81          76.0
GarageYrBlt              81          78.0
```

```
KitchenQual              0          1.0
LotFrontage            259        227.0
MSZoning                 0          4.0
MasVnrArea               8         15.0
MasVnrType               8         16.0
MiscFeature           1406       1408.0
PoolQC                1453       1456.0
SaleType                 0          1.0
TotalBsmtSF              0          1.0
Utilities                0          2.0
```

```
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/ipykernel_launcher.py:12: FutureWarning
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.

  if sys.path[0] == '':
```

In [19]:
```
# Explore features
def feat_explore(column):
    return train[column].value_counts()

# Function to impute missing values
def feat_impute(column, value):
    train.loc[train[column].isnull(),column] = value
    test.loc[test[column].isnull(),column] = value
```

In [20]:
```
#PoolQC, MiscFeature, Alley, Fence will all be removed as they are missing over half

features_drop = ['PoolQC','MiscFeature','Alley','Fence']
train = train.drop(features_drop, axis=1)
test = test.drop(features_drop, axis=1)
```

In [21]:
```
# dropping the columns we want to treat as categorical variables
house_numeric = house_numeric.drop(['MSSubClass', 'OverallQual', 'OverallCond', 'Year
                                    'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'Half
                                   'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'Gar
                                   'MoSold', 'YrSold'], axis=1)
house_numeric.head()
```

Out[21]:
```
   Id  LotFrontage  LotArea  MasVnrArea  BsmtFinSF1  BsmtFinSF2  BsmtUnfSF  \
0   1         65.0     8450       196.0         706           0        150
1   2         80.0     9600         0.0         978           0        284
2   3         68.0    11250       162.0         486           0        434
3   4         60.0     9550         0.0         216           0        540
```

```
4   5          84.0     14260       350.0          655             0         490

   TotalBsmtSF  1stFlrSF  2ndFlrSF  ...  GrLivArea  GarageArea  WoodDeckSF  \
0          856       856       854  ...       1710         548           0
1         1262      1262         0  ...       1262         460         298
2          920       920       866  ...       1786         608           0
3          756       961       756  ...       1717         642           0
4         1145      1145      1053  ...       2198         836         192

   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

   SalePrice
0     208500
1     181500
2     223500
3     140000
4     250000

[5 rows x 21 columns]
```

In [22]:
```python
# correlation matrix
cor = house_numeric.corr()
cor
```

Out[22]:
```
                     Id  LotFrontage   LotArea  MasVnrArea  BsmtFinSF1  \
Id             1.000000    -0.010601 -0.033226   -0.050298   -0.005024
LotFrontage   -0.010601     1.000000  0.426095    0.193458    0.233633
LotArea       -0.033226     0.426095  1.000000    0.104160    0.214103
MasVnrArea    -0.050298     0.193458  0.104160    1.000000    0.264736
BsmtFinSF1    -0.005024     0.233633  0.214103    0.264736    1.000000
BsmtFinSF2    -0.005968     0.049900  0.111170   -0.072319   -0.050117
BsmtUnfSF     -0.007940     0.132644 -0.002618    0.114442   -0.495251
TotalBsmtSF   -0.015415     0.392075  0.260833    0.363936    0.522396
1stFlrSF       0.010496     0.457181  0.299475    0.344501    0.445863
2ndFlrSF       0.005590     0.080177  0.050986    0.174561   -0.137079
LowQualFinSF  -0.044230     0.038469  0.004779   -0.069071   -0.064503
GrLivArea      0.008273     0.402797  0.263116    0.390857    0.208171
GarageArea     0.017634     0.344997  0.180403    0.373066    0.296970
WoodDeckSF    -0.029643     0.088521  0.171698    0.159718    0.204306
OpenPorchSF   -0.000477     0.151972  0.084774    0.125703    0.111761
EnclosedPorch  0.002889     0.010700 -0.018340   -0.110204   -0.102303
3SsnPorch     -0.046635     0.070029  0.020423    0.018796    0.026451
ScreenPorch    0.001330     0.041383  0.043160    0.061466    0.062021
```

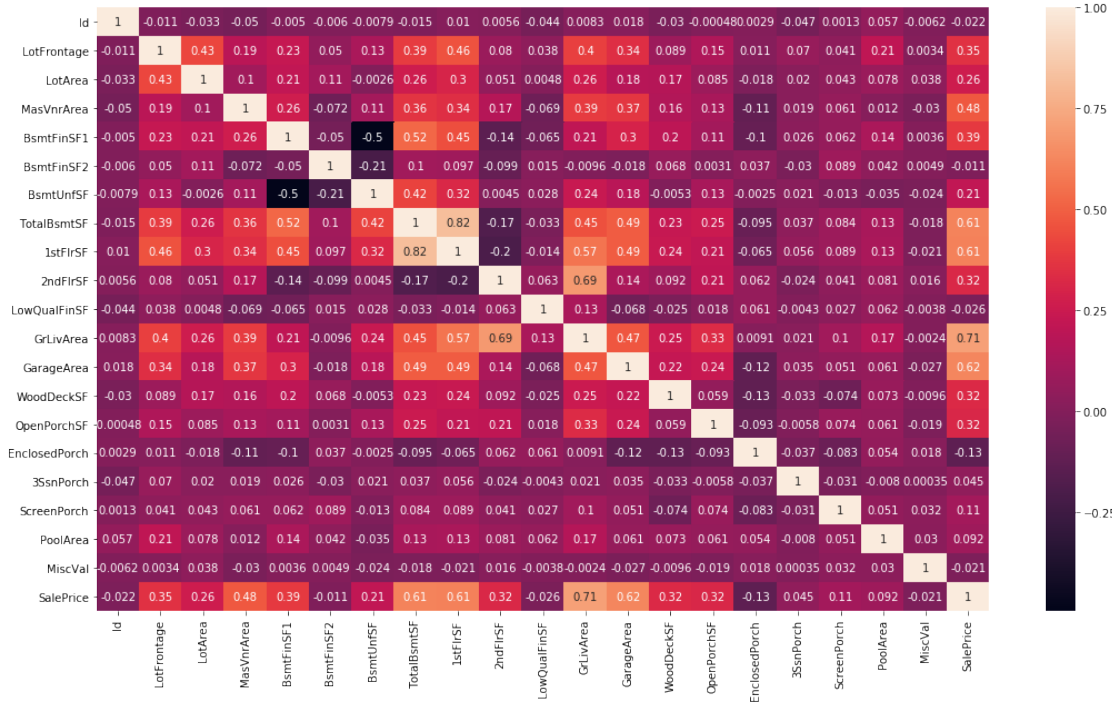
PoolArea	0.057044	0.206167	0.077672	0.011723	0.140491
MiscVal	-0.006242	0.003368	0.038068	-0.029815	0.003571
SalePrice	-0.021917	0.351799	0.263843	0.477493	0.386420

	BsmtFinSF2	BsmtUnfSF	TotalBsmtSF	1stFlrSF	2ndFlrSF	... \
Id	-0.005968	-0.007940	-0.015415	0.010496	0.005590	...
LotFrontage	0.049900	0.132644	0.392075	0.457181	0.080177	...
LotArea	0.111170	-0.002618	0.260833	0.299475	0.050986	...
MasVnrArea	-0.072319	0.114442	0.363936	0.344501	0.174561	...
BsmtFinSF1	-0.050117	-0.495251	0.522396	0.445863	-0.137079	...
BsmtFinSF2	1.000000	-0.209294	0.104810	0.097117	-0.099260	...
BsmtUnfSF	-0.209294	1.000000	0.415360	0.317987	0.004469	...
TotalBsmtSF	0.104810	0.415360	1.000000	0.819530	-0.174512	...
1stFlrSF	0.097117	0.317987	0.819530	1.000000	-0.202646	...
2ndFlrSF	-0.099260	0.004469	-0.174512	-0.202646	1.000000	...
LowQualFinSF	0.014807	0.028167	-0.033245	-0.014241	0.063353	...
GrLivArea	-0.009640	0.240257	0.454868	0.566024	0.687501	...
GarageArea	-0.018227	0.183303	0.486665	0.489782	0.138347	...
WoodDeckSF	0.067898	-0.005316	0.232019	0.235459	0.092165	...
OpenPorchSF	0.003093	0.129005	0.247264	0.211671	0.208026	...
EnclosedPorch	0.036543	-0.002538	-0.095478	-0.065292	0.061989	...
3SsnPorch	-0.029993	0.020764	0.037384	0.056104	-0.024358	...
ScreenPorch	0.088871	-0.012579	0.084489	0.088758	0.040606	...
PoolArea	0.041709	-0.035092	0.126053	0.131525	0.081487	...
MiscVal	0.004940	-0.023837	-0.018479	-0.021096	0.016197	...
SalePrice	-0.011378	0.214479	0.613581	0.605852	0.319334	...

	GrLivArea	GarageArea	WoodDeckSF	OpenPorchSF	EnclosedPorch \
Id	0.008273	0.017634	-0.029643	-0.000477	0.002889
LotFrontage	0.402797	0.344997	0.088521	0.151972	0.010700
LotArea	0.263116	0.180403	0.171698	0.084774	-0.018340
MasVnrArea	0.390857	0.373066	0.159718	0.125703	-0.110204
BsmtFinSF1	0.208171	0.296970	0.204306	0.111761	-0.102303
BsmtFinSF2	-0.009640	-0.018227	0.067898	0.003093	0.036543
BsmtUnfSF	0.240257	0.183303	-0.005316	0.129005	-0.002538
TotalBsmtSF	0.454868	0.486665	0.232019	0.247264	-0.095478
1stFlrSF	0.566024	0.489782	0.235459	0.211671	-0.065292
2ndFlrSF	0.687501	0.138347	0.092165	0.208026	0.061989
LowQualFinSF	0.134683	-0.067601	-0.025444	0.018251	0.061081
GrLivArea	1.000000	0.468997	0.247433	0.330224	0.009113
GarageArea	0.468997	1.000000	0.224666	0.241435	-0.121777
WoodDeckSF	0.247433	0.224666	1.000000	0.058661	-0.125989
OpenPorchSF	0.330224	0.241435	0.058661	1.000000	-0.093079
EnclosedPorch	0.009113	-0.121777	-0.125989	-0.093079	1.000000
3SsnPorch	0.020643	0.035087	-0.032771	-0.005842	-0.037305
ScreenPorch	0.101510	0.051412	-0.074181	0.074304	-0.082864
PoolArea	0.170205	0.061047	0.073378	0.060762	0.054203
MiscVal	-0.002416	-0.027400	-0.009551	-0.018584	0.018361

```
SalePrice       0.708624    0.623431    0.324413     0.315856      -0.128578

               3SsnPorch  ScreenPorch  PoolArea   MiscVal  SalePrice
Id             -0.046635     0.001330  0.057044 -0.006242  -0.021917
LotFrontage     0.070029     0.041383  0.206167  0.003368   0.351799
LotArea         0.020423     0.043160  0.077672  0.038068   0.263843
MasVnrArea      0.018796     0.061466  0.011723 -0.029815   0.477493
BsmtFinSF1      0.026451     0.062021  0.140491  0.003571   0.386420
BsmtFinSF2     -0.029993     0.088871  0.041709  0.004940  -0.011378
BsmtUnfSF       0.020764    -0.012579 -0.035092 -0.023837   0.214479
TotalBsmtSF     0.037384     0.084489  0.126053 -0.018479   0.613581
1stFlrSF        0.056104     0.088758  0.131525 -0.021096   0.605852
2ndFlrSF       -0.024358     0.040606  0.081487  0.016197   0.319334
LowQualFinSF   -0.004296     0.026799  0.062157 -0.003793  -0.025606
GrLivArea       0.020643     0.101510  0.170205 -0.002416   0.708624
GarageArea      0.035087     0.051412  0.061047 -0.027400   0.623431
WoodDeckSF     -0.032771    -0.074181  0.073378 -0.009551   0.324413
OpenPorchSF    -0.005842     0.074304  0.060762 -0.018584   0.315856
EnclosedPorch  -0.037305    -0.082864  0.054203  0.018361  -0.128578
3SsnPorch       1.000000    -0.031436 -0.007992  0.000354   0.044584
ScreenPorch    -0.031436     1.000000  0.051307  0.031946   0.111447
PoolArea       -0.007992     0.051307  1.000000  0.029669   0.092404
MiscVal         0.000354     0.031946  0.029669  1.000000  -0.021190
SalePrice       0.044584     0.111447  0.092404 -0.021190   1.000000

[21 rows x 21 columns]
```

In [23]:
```python
# plotting correlations on a heatmap

# figure size
plt.figure(figsize=(18,10))

# heatmap
sns.heatmap(cor, annot=True)
plt.show()
```




```
In [24]: #DATE CLEARING
         # variable formats
         house.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
Id               1460 non-null int64
MSSubClass       1460 non-null int64
MSZoning         1460 non-null object
LotFrontage      1201 non-null float64
LotArea          1460 non-null int64
Street           1460 non-null object
Alley            91 non-null object
LotShape         1460 non-null object
LandContour      1460 non-null object
Utilities        1460 non-null object
LotConfig        1460 non-null object
LandSlope        1460 non-null object
Neighborhood     1460 non-null object
Condition1       1460 non-null object
Condition2       1460 non-null object
BldgType         1460 non-null object
HouseStyle       1460 non-null object
OverallQual      1460 non-null int64
```

```
OverallCond      1460 non-null int64
YearBuilt        1460 non-null int64
YearRemodAdd     1460 non-null int64
RoofStyle        1460 non-null object
RoofMatl         1460 non-null object
Exterior1st      1460 non-null object
Exterior2nd      1460 non-null object
MasVnrType       1452 non-null object
MasVnrArea       1452 non-null float64
ExterQual        1460 non-null object
ExterCond        1460 non-null object
Foundation       1460 non-null object
BsmtQual         1423 non-null object
BsmtCond         1423 non-null object
BsmtExposure     1422 non-null object
BsmtFinType1     1423 non-null object
BsmtFinSF1       1460 non-null int64
BsmtFinType2     1422 non-null object
BsmtFinSF2       1460 non-null int64
BsmtUnfSF        1460 non-null int64
TotalBsmtSF      1460 non-null int64
Heating          1460 non-null object
HeatingQC        1460 non-null object
CentralAir       1460 non-null object
Electrical       1459 non-null object
1stFlrSF         1460 non-null int64
2ndFlrSF         1460 non-null int64
LowQualFinSF     1460 non-null int64
GrLivArea        1460 non-null int64
BsmtFullBath     1460 non-null int64
BsmtHalfBath     1460 non-null int64
FullBath         1460 non-null int64
HalfBath         1460 non-null int64
BedroomAbvGr     1460 non-null int64
KitchenAbvGr     1460 non-null int64
KitchenQual      1460 non-null object
TotRmsAbvGrd     1460 non-null int64
Functional       1460 non-null object
Fireplaces       1460 non-null int64
FireplaceQu      770 non-null object
GarageType       1379 non-null object
GarageYrBlt      1379 non-null float64
GarageFinish     1379 non-null object
GarageCars       1460 non-null int64
GarageArea       1460 non-null int64
GarageQual       1379 non-null object
GarageCond       1379 non-null object
PavedDrive       1460 non-null object
```

```
WoodDeckSF        1460 non-null int64
OpenPorchSF       1460 non-null int64
EnclosedPorch     1460 non-null int64
3SsnPorch         1460 non-null int64
ScreenPorch       1460 non-null int64
PoolArea          1460 non-null int64
PoolQC            7 non-null object
Fence             281 non-null object
MiscFeature       54 non-null object
MiscVal           1460 non-null int64
MoSold            1460 non-null int64
YrSold            1460 non-null int64
SaleType          1460 non-null object
SaleCondition     1460 non-null object
SalePrice         1460 non-null int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

In [25]:
```
house.isnull().sum()  #checking the number of null values in the dataset
```

Out[25]:
```
Id                  0
MSSubClass          0
MSZoning            0
LotFrontage       259
LotArea             0
Street              0
Alley            1369
LotShape            0
LandContour         0
Utilities           0
LotConfig           0
LandSlope           0
Neighborhood        0
Condition1          0
Condition2          0
BldgType            0
HouseStyle          0
OverallQual         0
OverallCond         0
YearBuilt           0
YearRemodAdd        0
RoofStyle           0
RoofMatl            0
Exterior1st         0
Exterior2nd         0
MasVnrType          8
MasVnrArea          8
```

```
ExterQual           0
ExterCond           0
Foundation          0
                  ...
BedroomAbvGr        0
KitchenAbvGr        0
KitchenQual         0
TotRmsAbvGrd        0
Functional          0
Fireplaces          0
FireplaceQu       690
GarageType         81
GarageYrBlt        81
GarageFinish       81
GarageCars          0
GarageArea          0
GarageQual         81
GarageCond         81
PavedDrive          0
WoodDeckSF          0
OpenPorchSF         0
EnclosedPorch       0
3SsnPorch           0
ScreenPorch         0
PoolArea            0
PoolQC           1453
Fence            1179
MiscFeature      1406
MiscVal             0
MoSold              0
YrSold              0
SaleType            0
SaleCondition       0
SalePrice           0
Length: 81, dtype: int64
```

In [27]:
```
#NULL VALUE TREATMENT
house.shape
```

Out[27]:
```
(1460, 81)
```

In [28]:
```
house = pd.concat((house,test))
```

```
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/ipykernel_launcher.py:1: FutureWarning
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.
```

```
  """Entry point for launching an IPython kernel.
```

```
In [29]: #NA in Alley column means No Alley, so we will replace NA by it.
         house['Alley'].fillna('No Alley', inplace=True)
         house['MasVnrType'].fillna('None', inplace=True)

In [30]: #NA in FireplaceQu column means No Fireplace, so we will replace NA by it.
         house['FireplaceQu'].fillna('No Fireplace', inplace=True)

In [31]: #NA in PoolQC column means No Pool, so we will replace NA by it.
         house['PoolQC'].fillna('No Pool', inplace=True)

In [32]: #NA in Fence column means No Fence, so we will replace NA by it.
         house['Fence'].fillna('No Fence', inplace=True)

In [33]: house['MasVnrArea'].fillna(0, inplace=True)

In [34]: house['LotFrontage'].fillna(0, inplace=True)

In [35]: #NA in GarageType, GarageFinish, GarageQual, GarageCond columns mean No Garage, so we

         house['GarageType'].fillna('No Garage', inplace=True)
         house['GarageFinish'].fillna('No Garage', inplace=True)
         house['GarageQual'].fillna('No Garage', inplace=True)
         house['GarageCond'].fillna('No Garage', inplace=True)

In [36]: # MiscFeature column has almost 99% null values so we will drop it
         house= house.drop('MiscFeature', axis=1)

In [37]: house.isnull().sum()

Out[37]: 1stFlrSF          0
         2ndFlrSF          0
         3SsnPorch         0
         Alley             0
         BedroomAbvGr      0
         BldgType          0
         BsmtCond         82
         BsmtExposure     82
         BsmtFinSF1        1
         BsmtFinSF2        1
         BsmtFinType1     79
         BsmtFinType2     80
         BsmtFullBath      2
         BsmtHalfBath      2
         BsmtQual         81
         BsmtUnfSF         1
```

```
CentralAir          0
Condition1          0
Condition2          0
Electrical          1
EnclosedPorch       0
ExterCond           0
ExterQual           0
Exterior1st         1
Exterior2nd         1
Fence               0
FireplaceQu         0
Fireplaces          0
Foundation          0
FullBath            0
                 ...
LotFrontage         0
LotShape            0
LowQualFinSF        0
MSSubClass          0
MSZoning            4
MasVnrArea          0
MasVnrType          0
MiscVal             0
MoSold              0
Neighborhood        0
OpenPorchSF         0
OverallCond         0
OverallQual         0
PavedDrive          0
PoolArea            0
PoolQC              0
RoofMatl            0
RoofStyle           0
SaleCondition       0
SalePrice        1459
SaleType            1
ScreenPorch         0
Street              0
TotRmsAbvGrd        0
TotalBsmtSF         1
Utilities           2
WoodDeckSF          0
YearBuilt           0
YearRemodAdd        0
YrSold              0
Length: 80, dtype: int64
```

In [38]:
```
#converting year to number of years
```

```
house['YearBuilt'] = 2019 - house['YearBuilt']
house['YearRemodAdd'] = 2019 - house['YearRemodAdd']
house['GarageYrBlt'] = 2019 - house['GarageYrBlt']
house['YrSold'] = 2019 - house['YrSold']
```

In [39]:
```
#converting from int type to object to treat the variables as categorical variables
house['MSSubClass'] = house['MSSubClass'].astype('object')
house['OverallQual'] = house['OverallQual'].astype('object')
house['OverallCond'] = house['OverallCond'].astype('object')
house['BsmtFullBath'] = house['BsmtFullBath'].astype('object')
house['BsmtHalfBath'] = house['BsmtHalfBath'].astype('object')
house['FullBath'] = house['FullBath'].astype('object')
house['HalfBath'] = house['HalfBath'].astype('object')
house['BedroomAbvGr'] = house['BedroomAbvGr'].astype('object')
house['KitchenAbvGr'] = house['KitchenAbvGr'].astype('object')
house['TotRmsAbvGrd'] = house['TotRmsAbvGrd'].astype('object')
house['Fireplaces'] = house['Fireplaces'].astype('object')
house['GarageCars'] = house['GarageCars'].astype('object')
```

In [40]:
```
house.shape
```

Out[40]:
```
(2919, 80)
```

In [41]:
```
final = house
```

In [42]:
```
#DUMMY VARIABLE
# List of variables to map

varlist1 =  ['Street']

# Defining the map function
def binary_map(x):
    return x.map({'Pave': 1, "Grvl": 0})

# Applying the function to the Lead list
final[varlist1] = final[varlist1].apply(binary_map)
```

In [43]:
```
# List of variables to map

varlist2 =  ['Utilities']

# Defining the map function
def binary_map(x):
    return x.map({'AllPub': 1, "NoSeWa": 0})

# Applying the function to the Lead list
final[varlist2] = final[varlist2].apply(binary_map)
```

In [44]:
```
# List of variables to map
```

```python
varlist3 =  ['CentralAir']

# Defining the map function
def binary_map(x):
    return x.map({'Y': 1, "N": 0})

# Applying the function to the Lead list
final[varlist3] = final[varlist3].apply(binary_map)
```

In [45]:
```python
#DATE PREPRATION
# split into X and y
X = final.drop([ 'Id'], axis=1)
```

In [46]:
```python
# creating dummy variables for categorical variables

# subset all categorical variables
house_categorical = X.select_dtypes(include=['object'])
house_categorical.head()
```

Out[46]:
```
      Alley BedroomAbvGr BldgType BsmtCond BsmtExposure BsmtFinType1  \
0  No Alley            3     1Fam       TA           No          GLQ
1  No Alley            3     1Fam       TA           Gd          ALQ
2  No Alley            3     1Fam       TA           Mn          GLQ
3  No Alley            3     1Fam       Gd           No          ALQ
4  No Alley            4     1Fam       TA           Av          GLQ

  BsmtFinType2 BsmtFullBath BsmtHalfBath BsmtQual  ... Neighborhood  \
0          Unf            1            0       Gd  ...      CollgCr
1          Unf            0            1       Gd  ...      Veenker
2          Unf            1            0       Gd  ...      CollgCr
3          Unf            1            0       TA  ...      Crawfor
4          Unf            1            0       Gd  ...      NoRidge

  OverallCond OverallQual PavedDrive   PoolQC RoofMatl RoofStyle  \
0           5           7          Y  No Pool  CompShg     Gable
1           8           6          Y  No Pool  CompShg     Gable
2           5           7          Y  No Pool  CompShg     Gable
3           5           7          Y  No Pool  CompShg     Gable
4           5           8          Y  No Pool  CompShg     Gable

  SaleCondition SaleType TotRmsAbvGrd
0        Normal       WD            8
1        Normal       WD            6
2        Normal       WD            6
3       Abnorml       WD            7
4        Normal       WD            9

[5 rows x 51 columns]
```

In [47]:
```
# convert into dummies
house_dummies = pd.get_dummies(house_categorical, drop_first=True)
house_dummies.head()
```

Out[47]:
```
   Alley_No Alley  Alley_Pave  BedroomAbvGr_1  BedroomAbvGr_2  BedroomAbvGr_3  \
0               1           0               0               0               1
1               1           0               0               0               1
2               1           0               0               0               1
3               1           0               0               0               1
4               1           0               0               0               0

   BedroomAbvGr_4  BedroomAbvGr_5  BedroomAbvGr_6  BedroomAbvGr_8  \
0               0               0               0               0
1               0               0               0               0
2               0               0               0               0
3               0               0               0               0
4               1               0               0               0

   BldgType_2fmCon  ...  TotRmsAbvGrd_6  TotRmsAbvGrd_7  TotRmsAbvGrd_8  \
0                0  ...               0               0               1
1                0  ...               1               0               0
2                0  ...               1               0               0
3                0  ...               0               1               0
4                0  ...               0               0               0

   TotRmsAbvGrd_9  TotRmsAbvGrd_10  TotRmsAbvGrd_11  TotRmsAbvGrd_12  \
0               0                0                0                0
1               0                0                0                0
2               0                0                0                0
3               0                0                0                0
4               1                0                0                0

   TotRmsAbvGrd_13  TotRmsAbvGrd_14  TotRmsAbvGrd_15
0                0                0                0
1                0                0                0
2                0                0                0
3                0                0                0
4                0                0                0

[5 rows x 286 columns]
```

In [48]:
```
# drop categorical variables
final = final.drop(list(house_categorical.columns), axis=1)
```

In [49]:
```
# concat dummy variables with X
final = pd.concat([final, house_dummies], axis=1)
```

In [51]:
```
final.shape
```

Out[51]: (2919, 315)

```
In [52]: test = final.tail(1459)
         test.shape
```

Out[52]: (1459, 315)

```
In [53]: X = final.head(1253)
         y = np.log(X.SalePrice)
         X = X.drop("SalePrice",1) # take out the target variable
```

```
In [54]: test = test.fillna(test.interpolate())
```

```
In [55]: X = X.fillna(X.interpolate())
```

```
In [56]: test = test.drop("SalePrice",1) # take out the target variable
```

```
In [57]: # scaling the features

         from sklearn.preprocessing import StandardScaler
         scaler = StandardScaler()
         scaler.fit(X)
```

```
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/preprocessing/data.py:645: Data
  return self.partial_fit(X, y)
```

Out[57]: StandardScaler(copy=True, with_mean=True, with_std=True)

```
In [58]: # scaling the features

         from sklearn.preprocessing import StandardScaler
         scaler = StandardScaler()
         scaler.fit(test)
```

```
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/preprocessing/data.py:645: Data
  return self.partial_fit(X, y)
```

Out[58]: StandardScaler(copy=True, with_mean=True, with_std=True)

```
In [59]: # split into train and test
         from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                             train_size=0.7,
                                                             test_size = 0.3, random_state=100)
```

```
In [62]: # list of alphas to tune
         params = {'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1,
          0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
```

```
 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50, 100, 500, 1000 ]}



lasso = Lasso()

# cross validation
folds = 5
model_cv = GridSearchCV(estimator = lasso,
                        param_grid = params,
                        scoring= 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True,
                        verbose = 1)

model_cv.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 28 candidates, totalling 140 fits
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
```

```
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
[Parallel(n_jobs=1)]: Done 140 out of 140 | elapsed:    7.3s finished
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
```

```
Out[62]: GridSearchCV(cv=5, error_score='raise-deprecating',
                estimator=Lasso(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=1000,
            normalize=False, positive=False, precompute=False, random_state=None,
            selection='cyclic', tol=0.0001, warm_start=False),
                fit_params=None, iid='warn', n_jobs=None,
                param_grid={'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6,
                pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
                scoring='neg_mean_absolute_error', verbose=1)
```

```
In [63]: cv_results = pd.DataFrame(model_cv.cv_results_)
         cv_results
```

Out[63]:

	mean_fit_time	mean_score_time	mean_test_score	mean_train_score \
0	0.274212	0.004110	-0.091523	-0.056075
1	0.244861	0.002870	-0.091161	-0.072613
2	0.173516	0.002770	-0.116587	-0.109403
3	0.082357	0.002938	-0.118583	-0.112752
4	0.033549	0.002182	-0.118156	-0.112917
5	0.024452	0.002076	-0.117573	-0.113066
6	0.025984	0.003211	-0.117488	-0.113369
7	0.028631	0.002370	-0.117897	-0.113877
8	0.030108	0.002851	-0.118635	-0.114587
9	0.035297	0.002943	-0.119515	-0.115495
10	0.038051	0.003373	-0.120513	-0.116623
11	0.032366	0.002769	-0.121746	-0.117907
12	0.028407	0.002819	-0.123133	-0.119297
13	0.025129	0.002542	-0.124659	-0.120834
14	0.014724	0.002274	-0.139562	-0.136367
15	0.014366	0.002575	-0.152810	-0.149749
16	0.011741	0.002108	-0.153929	-0.151368
17	0.013865	0.002367	-0.155187	-0.152794
18	0.012107	0.002085	-0.156775	-0.154286
19	0.011731	0.002189	-0.158286	-0.155826
20	0.011086	0.002568	-0.159610	-0.157266
21	0.008549	0.002020	-0.160773	-0.158409
22	0.009219	0.002247	-0.161983	-0.159568
23	0.009728	0.002636	-0.177749	-0.174876
24	0.007513	0.002218	-0.210756	-0.207993
25	0.008968	0.002932	-0.266682	-0.263665
26	0.008236	0.002518	-0.304928	-0.303828

27	0.006962	0.002076	-0.311764	-0.310564

	param_alpha	params	rank_test_score	split0_test_score \
0	0.0001	{u'alpha': 0.0001}	2	-0.098976
1	0.001	{u'alpha': 0.001}	1	-0.101993
2	0.01	{u'alpha': 0.01}	3	-0.127575
3	0.05	{u'alpha': 0.05}	8	-0.129530
4	0.1	{u'alpha': 0.1}	7	-0.128250
5	0.2	{u'alpha': 0.2}	5	-0.127181
6	0.3	{u'alpha': 0.3}	4	-0.127288
7	0.4	{u'alpha': 0.4}	6	-0.127765
8	0.5	{u'alpha': 0.5}	9	-0.128265
9	0.6	{u'alpha': 0.6}	10	-0.128895
10	0.7	{u'alpha': 0.7}	11	-0.129769
11	0.8	{u'alpha': 0.8}	12	-0.130917
12	0.9	{u'alpha': 0.9}	13	-0.132269
13	1	{u'alpha': 1.0}	14	-0.133954
14	2	{u'alpha': 2.0}	15	-0.153099
15	3	{u'alpha': 3.0}	16	-0.168463
16	4	{u'alpha': 4.0}	17	-0.169628
17	5	{u'alpha': 5.0}	18	-0.170999
18	6	{u'alpha': 6.0}	19	-0.172201
19	7	{u'alpha': 7.0}	20	-0.173399
20	8	{u'alpha': 8.0}	21	-0.174353
21	9	{u'alpha': 9.0}	22	-0.175099
22	10	{u'alpha': 10.0}	23	-0.175884
23	20	{u'alpha': 20}	24	-0.187780
24	50	{u'alpha': 50}	25	-0.224415
25	100	{u'alpha': 100}	26	-0.281424
26	500	{u'alpha': 500}	27	-0.320906
27	1000	{u'alpha': 1000}	28	-0.326423

	split0_train_score	split1_test_score	...	split2_test_score \
0	-0.052832	-0.087747	...	-0.094678
1	-0.070897	-0.082141	...	-0.100868
2	-0.107663	-0.109348	...	-0.132904
3	-0.111656	-0.110469	...	-0.135012
4	-0.111901	-0.109704	...	-0.134569
5	-0.111720	-0.109033	...	-0.133512
6	-0.111811	-0.108362	...	-0.133718
7	-0.112074	-0.108307	...	-0.134392
8	-0.112556	-0.108922	...	-0.135174
9	-0.113244	-0.109662	...	-0.136032
10	-0.114253	-0.110466	...	-0.136974
11	-0.115526	-0.111303	...	-0.138182
12	-0.116961	-0.112419	...	-0.139213
13	-0.118515	-0.113866	...	-0.140071
14	-0.134866	-0.126508	...	-0.146993

15	-0.146971	-0.140368	...	-0.157595
16	-0.148344	-0.142352	...	-0.158110
17	-0.149836	-0.143783	...	-0.159027
18	-0.151288	-0.145432	...	-0.160466
19	-0.153000	-0.147124	...	-0.161736
20	-0.154430	-0.148667	...	-0.162994
21	-0.155491	-0.150071	...	-0.164252
22	-0.156601	-0.151651	...	-0.165378
23	-0.171337	-0.171805	...	-0.178717
24	-0.208423	-0.212157	...	-0.201977
25	-0.267331	-0.278180	...	-0.243918
26	-0.299555	-0.319317	...	-0.277228
27	-0.307031	-0.324939	...	-0.281743

	split2_train_score	split3_test_score	split3_train_score	\
0	-0.056104	-0.089407	-0.058059	
1	-0.071895	-0.088140	-0.073606	
2	-0.104958	-0.103975	-0.113211	
3	-0.107658	-0.106698	-0.115745	
4	-0.107881	-0.106876	-0.115871	
5	-0.108303	-0.107082	-0.115969	
6	-0.108900	-0.107146	-0.116075	
7	-0.109626	-0.107870	-0.116545	
8	-0.110516	-0.108870	-0.117305	
9	-0.111579	-0.110060	-0.118256	
10	-0.112849	-0.111285	-0.119386	
11	-0.114342	-0.112677	-0.120580	
12	-0.115859	-0.114364	-0.121895	
13	-0.117439	-0.116216	-0.123426	
14	-0.131245	-0.133601	-0.140268	
15	-0.147499	-0.145184	-0.150404	
16	-0.148656	-0.146027	-0.151704	
17	-0.150016	-0.147276	-0.152839	
18	-0.151584	-0.148837	-0.154158	
19	-0.153073	-0.150588	-0.155703	
20	-0.154709	-0.152257	-0.157201	
21	-0.155967	-0.153173	-0.158208	
22	-0.157056	-0.154158	-0.159277	
23	-0.172292	-0.166666	-0.174116	
24	-0.203281	-0.192565	-0.206979	
25	-0.256158	-0.243030	-0.263175	
26	-0.309777	-0.286035	-0.309377	
27	-0.315136	-0.293805	-0.315462	

	split4_test_score	split4_train_score	std_fit_time	std_score_time	\
0	-0.086785	-0.058051	0.045233	0.001836	
1	-0.082653	-0.073090	0.078582	0.000422	
2	-0.109111	-0.109922	0.017550	0.000474	

3	-0.111191	-0.113811	0.014737	0.001787
4	-0.111372	-0.113972	0.004502	0.000253
5	-0.111051	-0.114105	0.004740	0.000125
6	-0.110921	-0.114391	0.003742	0.001572
7	-0.111149	-0.114938	0.005585	0.000308
8	-0.111943	-0.115725	0.006069	0.000672
9	-0.112929	-0.116731	0.005201	0.000299
10	-0.114075	-0.117882	0.006551	0.000485
11	-0.115655	-0.119124	0.004641	0.000914
12	-0.117408	-0.120500	0.004283	0.000708
13	-0.119198	-0.122048	0.003572	0.000748
14	-0.137607	-0.138591	0.001467	0.000430
15	-0.152419	-0.151701	0.002376	0.000667
16	-0.153506	-0.153354	0.000692	0.000098
17	-0.154826	-0.155223	0.002933	0.000413
18	-0.156918	-0.156923	0.001441	0.000226
19	-0.158559	-0.158290	0.001470	0.000197
20	-0.159755	-0.159402	0.001857	0.000452
21	-0.161251	-0.160605	0.000496	0.000134
22	-0.162826	-0.161877	0.000679	0.000397
23	-0.183752	-0.177547	0.001529	0.000338
24	-0.222577	-0.209993	0.000362	0.000214
25	-0.286707	-0.264955	0.000934	0.000830
26	-0.320984	-0.299886	0.001167	0.000687
27	-0.331749	-0.308472	0.000443	0.000179

	std_test_score	std_train_score
0	0.004623	0.001945
1	0.008657	0.001058
2	0.011435	0.002864
3	0.011411	0.002891
4	0.011095	0.002843
5	0.010691	0.002782
6	0.010886	0.002685
7	0.011020	0.002645
8	0.010959	0.002623
9	0.010866	0.002618
10	0.010806	0.002591
11	0.010793	0.002499
12	0.010642	0.002424
13	0.010406	0.002397
14	0.009479	0.003127
15	0.009820	0.002139
16	0.009610	0.002537
17	0.009578	0.002569
18	0.009427	0.002586
19	0.009222	0.002535
20	0.008985	0.002457

```
21       0.008841         0.002477
22       0.008652         0.002549
23       0.007692         0.002988
24       0.012130         0.002766
25       0.019110         0.004023
26       0.019198         0.004706
27       0.020046         0.003913

[28 rows x 21 columns]
```

In [64]:
```python
#lets find out the R-squared value of the lasso model
model_cv1 = GridSearchCV(estimator = lasso,
                         param_grid = params,
                         scoring= 'r2',
                         cv = folds,
                         verbose = 1,
                         return_train_score=True)

# fit the model
model_cv1.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 28 candidates, totalling 140 fits


[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
```

```
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
[Parallel(n_jobs=1)]: Done 140 out of 140 | elapsed:    8.2s finished
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
```

Out[64]:
```
GridSearchCV(cv=5, error_score='raise-deprecating',
       estimator=Lasso(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=1000,
   normalize=False, positive=False, precompute=False, random_state=None,
   selection='cyclic', tol=0.0001, warm_start=False),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid={'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6,
       pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
       scoring='r2', verbose=1)
```

In [65]:
```
# cv results
cv_results1 = pd.DataFrame(model_cv1.cv_results_)
cv_results
```

Out[65]:

	mean_fit_time	mean_score_time	mean_test_score	mean_train_score	\
0	0.274212	0.004110	-0.091523	-0.056075	
1	0.244861	0.002870	-0.091161	-0.072613	
2	0.173516	0.002770	-0.116587	-0.109403	
3	0.082357	0.002938	-0.118583	-0.112752	
4	0.033549	0.002182	-0.118156	-0.112917	
5	0.024452	0.002076	-0.117573	-0.113066	
6	0.025984	0.003211	-0.117488	-0.113369	
7	0.028631	0.002370	-0.117897	-0.113877	
8	0.030108	0.002851	-0.118635	-0.114587	
9	0.035297	0.002943	-0.119515	-0.115495	
10	0.038051	0.003373	-0.120513	-0.116623	
11	0.032366	0.002769	-0.121746	-0.117907	
12	0.028407	0.002819	-0.123133	-0.119297	
13	0.025129	0.002542	-0.124659	-0.120834	
14	0.014724	0.002274	-0.139562	-0.136367	
15	0.014366	0.002575	-0.152810	-0.149749	
16	0.011741	0.002108	-0.153929	-0.151368	
17	0.013865	0.002367	-0.155187	-0.152794	
18	0.012107	0.002085	-0.156775	-0.154286	
19	0.011731	0.002189	-0.158286	-0.155826	
20	0.011086	0.002568	-0.159610	-0.157266	
21	0.008549	0.002020	-0.160773	-0.158409	

22	0.009219	0.002247	-0.161983	-0.159568
23	0.009728	0.002636	-0.177749	-0.174876
24	0.007513	0.002218	-0.210756	-0.207993
25	0.008968	0.002932	-0.266682	-0.263665
26	0.008236	0.002518	-0.304928	-0.303828
27	0.006962	0.002076	-0.311764	-0.310564

	param_alpha	params	rank_test_score	split0_test_score	\
0	0.0001	{u'alpha': 0.0001}	2	-0.098976	
1	0.001	{u'alpha': 0.001}	1	-0.101993	
2	0.01	{u'alpha': 0.01}	3	-0.127575	
3	0.05	{u'alpha': 0.05}	8	-0.129530	
4	0.1	{u'alpha': 0.1}	7	-0.128250	
5	0.2	{u'alpha': 0.2}	5	-0.127181	
6	0.3	{u'alpha': 0.3}	4	-0.127288	
7	0.4	{u'alpha': 0.4}	6	-0.127765	
8	0.5	{u'alpha': 0.5}	9	-0.128265	
9	0.6	{u'alpha': 0.6}	10	-0.128895	
10	0.7	{u'alpha': 0.7}	11	-0.129769	
11	0.8	{u'alpha': 0.8}	12	-0.130917	
12	0.9	{u'alpha': 0.9}	13	-0.132269	
13	1	{u'alpha': 1.0}	14	-0.133954	
14	2	{u'alpha': 2.0}	15	-0.153099	
15	3	{u'alpha': 3.0}	16	-0.168463	
16	4	{u'alpha': 4.0}	17	-0.169628	
17	5	{u'alpha': 5.0}	18	-0.170999	
18	6	{u'alpha': 6.0}	19	-0.172201	
19	7	{u'alpha': 7.0}	20	-0.173399	
20	8	{u'alpha': 8.0}	21	-0.174353	
21	9	{u'alpha': 9.0}	22	-0.175099	
22	10	{u'alpha': 10.0}	23	-0.175884	
23	20	{u'alpha': 20}	24	-0.187780	
24	50	{u'alpha': 50}	25	-0.224415	
25	100	{u'alpha': 100}	26	-0.281424	
26	500	{u'alpha': 500}	27	-0.320906	
27	1000	{u'alpha': 1000}	28	-0.326423	

	split0_train_score	split1_test_score	...	split2_test_score	\
0	-0.052832	-0.087747	...	-0.094678	
1	-0.070897	-0.082141	...	-0.100868	
2	-0.107663	-0.109348	...	-0.132904	
3	-0.111656	-0.110469	...	-0.135012	
4	-0.111901	-0.109704	...	-0.134569	
5	-0.111720	-0.109033	...	-0.133512	
6	-0.111811	-0.108362	...	-0.133718	
7	-0.112074	-0.108307	...	-0.134392	
8	-0.112556	-0.108922	...	-0.135174	
9	-0.113244	-0.109662	...	-0.136032	

```
10         -0.114253         -0.110466  ...          -0.136974
11         -0.115526         -0.111303  ...          -0.138182
12         -0.116961         -0.112419  ...          -0.139213
13         -0.118515         -0.113866  ...          -0.140071
14         -0.134866         -0.126508  ...          -0.146993
15         -0.146971         -0.140368  ...          -0.157595
16         -0.148344         -0.142352  ...          -0.158110
17         -0.149836         -0.143783  ...          -0.159027
18         -0.151288         -0.145432  ...          -0.160466
19         -0.153000         -0.147124  ...          -0.161736
20         -0.154430         -0.148667  ...          -0.162994
21         -0.155491         -0.150071  ...          -0.164252
22         -0.156601         -0.151651  ...          -0.165378
23         -0.171337         -0.171805  ...          -0.178717
24         -0.208423         -0.212157  ...          -0.201977
25         -0.267331         -0.278180  ...          -0.243918
26         -0.299555         -0.319317  ...          -0.277228
27         -0.307031         -0.324939  ...          -0.281743

    split2_train_score  split3_test_score  split3_train_score  \
0            -0.056104          -0.089407           -0.058059
1            -0.071895          -0.088140           -0.073606
2            -0.104958          -0.103975           -0.113211
3            -0.107658          -0.106698           -0.115745
4            -0.107881          -0.106876           -0.115871
5            -0.108303          -0.107082           -0.115969
6            -0.108900          -0.107146           -0.116075
7            -0.109626          -0.107870           -0.116545
8            -0.110516          -0.108870           -0.117305
9            -0.111579          -0.110060           -0.118256
10           -0.112849          -0.111285           -0.119386
11           -0.114342          -0.112677           -0.120580
12           -0.115859          -0.114364           -0.121895
13           -0.117439          -0.116216           -0.123426
14           -0.131245          -0.133601           -0.140268
15           -0.147499          -0.145184           -0.150404
16           -0.148656          -0.146027           -0.151704
17           -0.150016          -0.147276           -0.152839
18           -0.151584          -0.148837           -0.154158
19           -0.153073          -0.150588           -0.155703
20           -0.154709          -0.152257           -0.157201
21           -0.155967          -0.153173           -0.158208
22           -0.157056          -0.154158           -0.159277
23           -0.172292          -0.166666           -0.174116
24           -0.203281          -0.192565           -0.206979
25           -0.256158          -0.243030           -0.263175
26           -0.309777          -0.286035           -0.309377
27           -0.315136          -0.293805           -0.315462
```

	split4_test_score	split4_train_score	std_fit_time	std_score_time	\
0	-0.086785	-0.058051	0.045233	0.001836	
1	-0.082653	-0.073090	0.078582	0.000422	
2	-0.109111	-0.109922	0.017550	0.000474	
3	-0.111191	-0.113811	0.014737	0.001787	
4	-0.111372	-0.113972	0.004502	0.000253	
5	-0.111051	-0.114105	0.004740	0.000125	
6	-0.110921	-0.114391	0.003742	0.001572	
7	-0.111149	-0.114938	0.005585	0.000308	
8	-0.111943	-0.115725	0.006069	0.000672	
9	-0.112929	-0.116731	0.005201	0.000299	
10	-0.114075	-0.117882	0.006551	0.000485	
11	-0.115655	-0.119124	0.004641	0.000914	
12	-0.117408	-0.120500	0.004283	0.000708	
13	-0.119198	-0.122048	0.003572	0.000748	
14	-0.137607	-0.138591	0.001467	0.000430	
15	-0.152419	-0.151701	0.002376	0.000667	
16	-0.153506	-0.153354	0.000692	0.000098	
17	-0.154826	-0.155223	0.002933	0.000413	
18	-0.156918	-0.156923	0.001441	0.000226	
19	-0.158559	-0.158290	0.001470	0.000197	
20	-0.159755	-0.159402	0.001857	0.000452	
21	-0.161251	-0.160605	0.000496	0.000134	
22	-0.162826	-0.161877	0.000679	0.000397	
23	-0.183752	-0.177547	0.001529	0.000338	
24	-0.222577	-0.209993	0.000362	0.000214	
25	-0.286707	-0.264955	0.000934	0.000830	
26	-0.320984	-0.299886	0.001167	0.000687	
27	-0.331749	-0.308472	0.000443	0.000179	

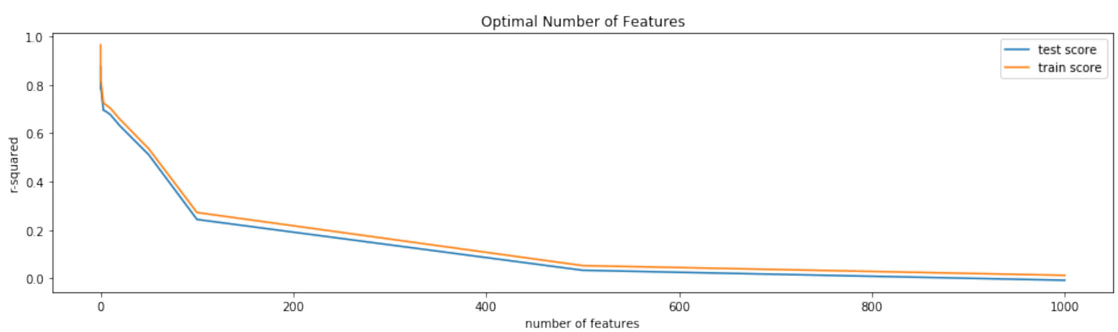
	std_test_score	std_train_score
0	0.004623	0.001945
1	0.008657	0.001058
2	0.011435	0.002864
3	0.011411	0.002891
4	0.011095	0.002843
5	0.010691	0.002782
6	0.010886	0.002685
7	0.011020	0.002645
8	0.010959	0.002623
9	0.010866	0.002618
10	0.010806	0.002591
11	0.010793	0.002499
12	0.010642	0.002424
13	0.010406	0.002397
14	0.009479	0.003127
15	0.009820	0.002139

16	0.009610	0.002537
17	0.009578	0.002569
18	0.009427	0.002586
19	0.009222	0.002535
20	0.008985	0.002457
21	0.008841	0.002477
22	0.008652	0.002549
23	0.007692	0.002988
24	0.012130	0.002766
25	0.019110	0.004023
26	0.019198	0.004706
27	0.020046	0.003913

[28 rows x 21 columns]

In [66]:
```
# plotting cv results
plt.figure(figsize=(16,4))

plt.plot(cv_results1["param_alpha"], cv_results1["mean_test_score"])
plt.plot(cv_results1["param_alpha"], cv_results1["mean_train_score"])
plt.xlabel('number of features')
plt.ylabel('r-squared')
plt.title("Optimal Number of Features")
plt.legend(['test score', 'train score'], loc='upper right')
```

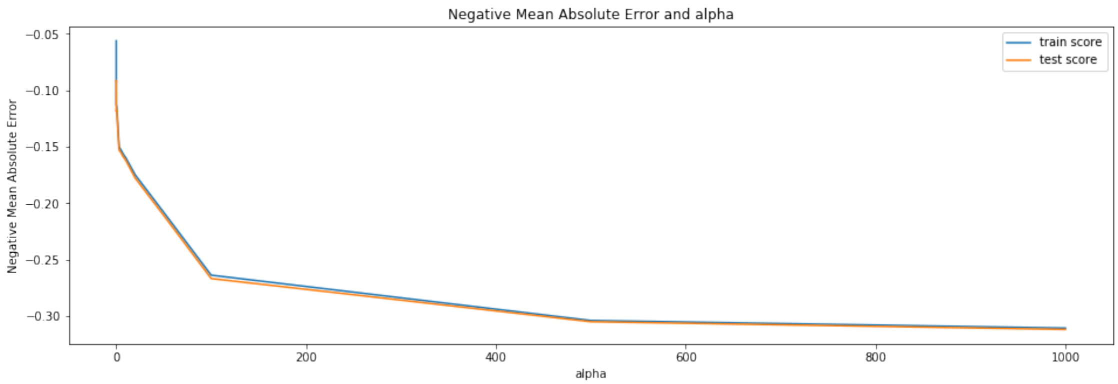
Out[66]: <matplotlib.legend.Legend at 0x1a1990cb50>



In [67]:
```
#checking the value of optimum number of parameters
print(model_cv.best_params_)
print(model_cv.best_score_)
```

```
{'alpha': 0.001}
-0.0911609631185767
```

In [68]:
```
# plotting mean test and train scoes with alpha
cv_results['param_alpha'] = cv_results['param_alpha'].astype('float32')

# plotting
plt.figure(figsize=(16,5))
plt.plot(cv_results['param_alpha'], cv_results['mean_train_score'])
plt.plot(cv_results['param_alpha'], cv_results['mean_test_score'])
plt.xlabel('alpha')
plt.ylabel('Negative Mean Absolute Error')

plt.title("Negative Mean Absolute Error and alpha")
plt.legend(['train score', 'test score'], loc='upper right')
plt.show()
```



In [69]:
```
alpha = 0.0001

lasso = Lasso(alpha=alpha)

lasso.fit(X_train, y_train)
```

```
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
```

Out[69]:
```
Lasso(alpha=0.0001, copy_X=True, fit_intercept=True, max_iter=1000,
   normalize=False, positive=False, precompute=False, random_state=None,
   selection='cyclic', tol=0.0001, warm_start=False)
```

In [70]:
```
#lets predict the R-squared value of test and train data
y_train_pred = lasso.predict(X_train)
print(metrics.r2_score(y_true=y_train, y_pred=y_train_pred))
```

```
0.9605681614162449
```

```
In [71]: alpha = 0.0001

         lasso = Lasso(alpha=alpha)

         lasso.fit(X_train, y_train)
```

```
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
```

```
Out[71]: Lasso(alpha=0.0001, copy_X=True, fit_intercept=True, max_iter=1000,
            normalize=False, positive=False, precompute=False, random_state=None,
            selection='cyclic', tol=0.0001, warm_start=False)
```

```
In [72]: #lets predict the R-squared value of test and train data
         y_test_pred = lasso.predict(X_test)
         print(metrics.r2_score(y_true=y_test, y_pred=y_test_pred))
```

```
0.8696490258097069
```

```
In [73]: from sklearn.metrics import mean_squared_error
         print ('RMSE is: \n', mean_squared_error(y_test, y_test_pred))
```

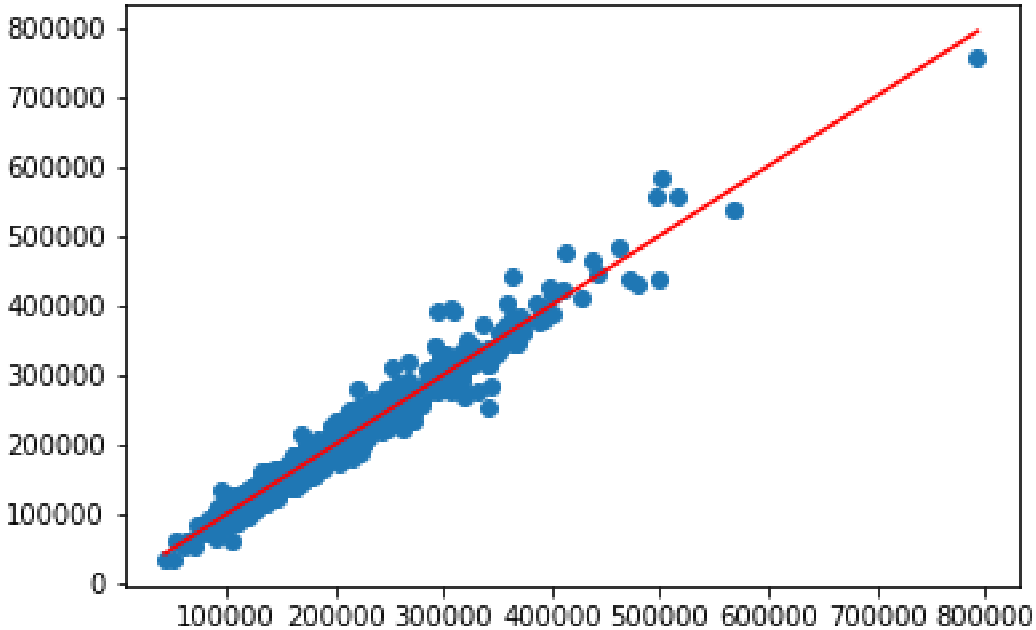
```
('RMSE is: \n', 0.02104436592084776)
```

```
In [74]: alpha = 0.0001

         lasso = Lasso(alpha=alpha)

         lasso.fit(X_train,y_train)
         preds = lasso.predict(test)
         final_predictions = np.exp(preds)
```

```
/Users/rohityadav/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/coordinate_descen
  ConvergenceWarning)
```

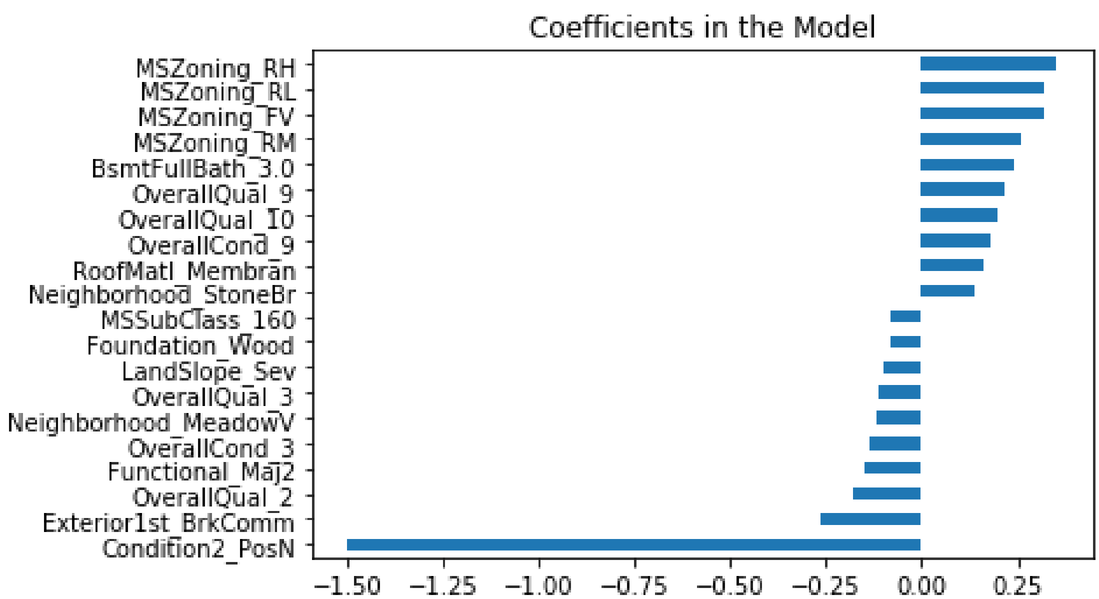
```
In [75]: # This is a good way to see how model predict data
         p_pred = np.expm1(lasso.predict(X_train))
         plt.scatter(p_pred, np.expm1(y_train))
         plt.plot([min(p_pred),max(p_pred)], [min(p_pred),max(p_pred)], c="red")
```

```
Out[75]: [<matplotlib.lines.Line2D at 0x1a18fad810>]
```



In [76]:
```
coef = pd.Series(lasso.coef_, index = X_train.columns).sort_values()
imp_coef = pd.concat([coef.head(10), coef.tail(10)])
imp_coef.plot(kind = "barh")
plt.title("Coefficients in the Model")
```

Out[76]:
```
Text(0.5,1,'Coefficients in the Model')
```



In [77]:
```
test.index = test.index + 1461
```

In [78]:
```
submission = pd.DataFrame({'Id': test.index ,'SalePrice': final_predictions })
```

In [79]:
```
submission.to_csv("submission.csv",index=False)
```