HousePrediction

April 23, 2019

```
In [1]: import numpy as np
        import pandas as pd

In [2]: d=pd.read_csv(r'D:\HousePredictions\train.csv',encoding='unicode_escape')
        d.head()

Out[2]:    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  \
        0         Lvl    AllPub    ...            0    NaN   NaN         NaN       0
        1         Lvl    AllPub    ...            0    NaN   NaN         NaN       0
        2         Lvl    AllPub    ...            0    NaN   NaN         NaN       0
        3         Lvl    AllPub    ...            0    NaN   NaN         NaN       0
        4         Lvl    AllPub    ...            0    NaN   NaN         NaN       0

          MoSold YrSold  SaleType  SaleCondition  SalePrice
        0      2   2008        WD         Normal     208500
        1      5   2007        WD         Normal     181500
        2      9   2008        WD         Normal     223500
        3      2   2006        WD        Abnorml     140000
        4     12   2008        WD         Normal     250000

        [5 rows x 81 columns]

In [3]: d.describe()

Out[3]:                 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  \
count      ...      1460.000000  1460.000000    1460.000000  1460.000000
mean       ...        94.244521    46.660274      21.954110     3.409589
std        ...       125.338794    66.256028      61.119149    29.317331
min        ...         0.000000     0.000000       0.000000     0.000000
25%        ...         0.000000     0.000000       0.000000     0.000000
50%        ...         0.000000    25.000000       0.000000     0.000000
75%        ...       168.000000    68.000000       0.000000     0.000000
max        ...       857.000000   547.000000     552.000000   508.000000

       ScreenPorch     PoolArea       MiscVal       MoSold       YrSold  \
count  1460.000000  1460.000000   1460.000000  1460.000000  1460.000000
mean     15.060959     2.758904     43.489041     6.321918  2007.815753
std      55.757415    40.177307    496.123024     2.703626     1.328095
min       0.000000     0.000000      0.000000     1.000000  2006.000000
25%       0.000000     0.000000      0.000000     5.000000  2007.000000
50%       0.000000     0.000000      0.000000     6.000000  2008.000000
75%       0.000000     0.000000      0.000000     8.000000  2009.000000
max     480.000000   738.000000  15500.000000    12.000000  2010.000000

           SalePrice
count    1460.000000
mean   180921.195890
std     79442.502883
min     34900.000000
25%    129975.000000
50%    163000.000000
75%    214000.000000
max    755000.000000

[8 rows x 38 columns]
```

In [4]:
```
d.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 [5]: d_missing=d.isna().sum()
```

missing[d_missing>0].sort_values(ascending=False)

```
In [6]: d_missing[d_missing>0].sort_values(ascending=False)
```

```
Out[6]: PoolQC          1453
        MiscFeature     1406
        Alley           1369
        Fence           1179
        FireplaceQu      690
```

```
LotFrontage      259
GarageYrBlt       81
GarageType        81
GarageFinish      81
GarageQual        81
GarageCond        81
BsmtFinType2      38
BsmtExposure      38
BsmtFinType1      37
BsmtCond          37
BsmtQual          37
MasVnrArea         8
MasVnrType         8
Electrical         1
dtype: int64
```

In [7]:
```
#keeping only columns which dont have na
```

In [8]:
```
d=d.dropna(axis=1, how='any')
d.shape
```

Out[8]:
```
(1460, 62)
```

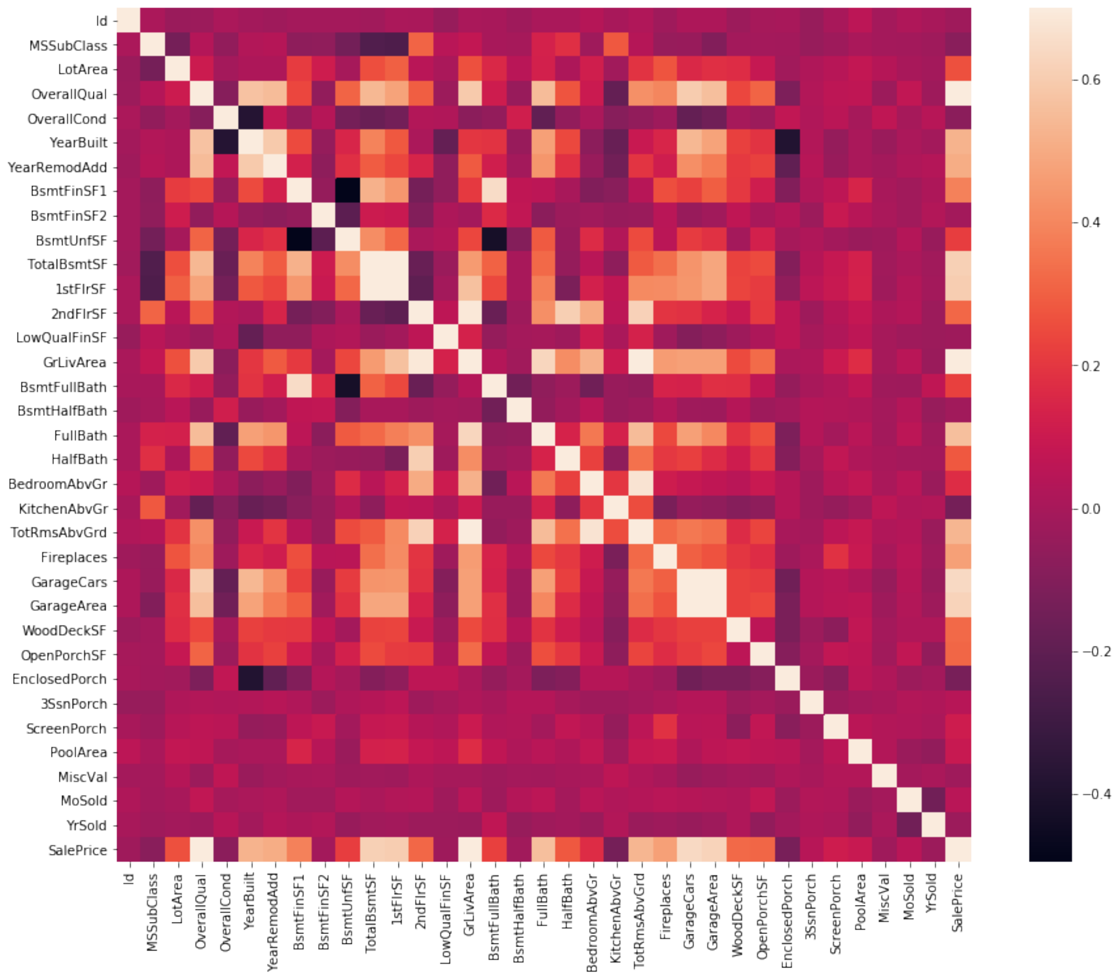
In [9]:
```
#removing Id field which doesnt have impact on house price.
#del d['Id']
d.head()
```

Out[9]:
```
   Id  MSSubClass MSZoning  LotArea Street LotShape LandContour Utilities  \
0   1          60       RL     8450   Pave      Reg         Lvl    AllPub
1   2          20       RL     9600   Pave      Reg         Lvl    AllPub
2   3          60       RL    11250   Pave      IR1         Lvl    AllPub
3   4          70       RL     9550   Pave      IR1         Lvl    AllPub
4   5          60       RL    14260   Pave      IR1         Lvl    AllPub

  LotConfig LandSlope    ...     EnclosedPorch 3SsnPorch ScreenPorch PoolArea  \
0    Inside       Gtl    ...                 0         0           0        0
1       FR2       Gtl    ...                 0         0           0        0
2    Inside       Gtl    ...                 0         0           0        0
3    Corner       Gtl    ...               272         0           0        0
4       FR2       Gtl    ...                 0         0           0        0

  MiscVal  MoSold  YrSold  SaleType  SaleCondition SalePrice
0       0       2    2008        WD         Normal    208500
1       0       5    2007        WD         Normal    181500
2       0       9    2008        WD         Normal    223500
3       0       2    2006        WD        Abnorml    140000
4       0      12    2008        WD         Normal    250000

[5 rows x 62 columns]
```

In [10]:
```python
#corelation matrix
```

In [86]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
matrix = d.corr()
f, ax = plt.subplots(figsize=(16, 12))
sns.heatmap(matrix, vmax=0.7, square=True)
```

Out[86]: <matplotlib.axes._subplots.AxesSubplot at 0x1ad3cb39e80>



In [87]:
```python
#selcting only features which are highly correlated
tcf=matrix['SalePrice'].sort_values(ascending=False)
```

In [88]:
```python
# Filter out the target variables (SalePrice) and variables with a low correlation sc
tcf = tcf[abs(tcf) >= 0.6]
```

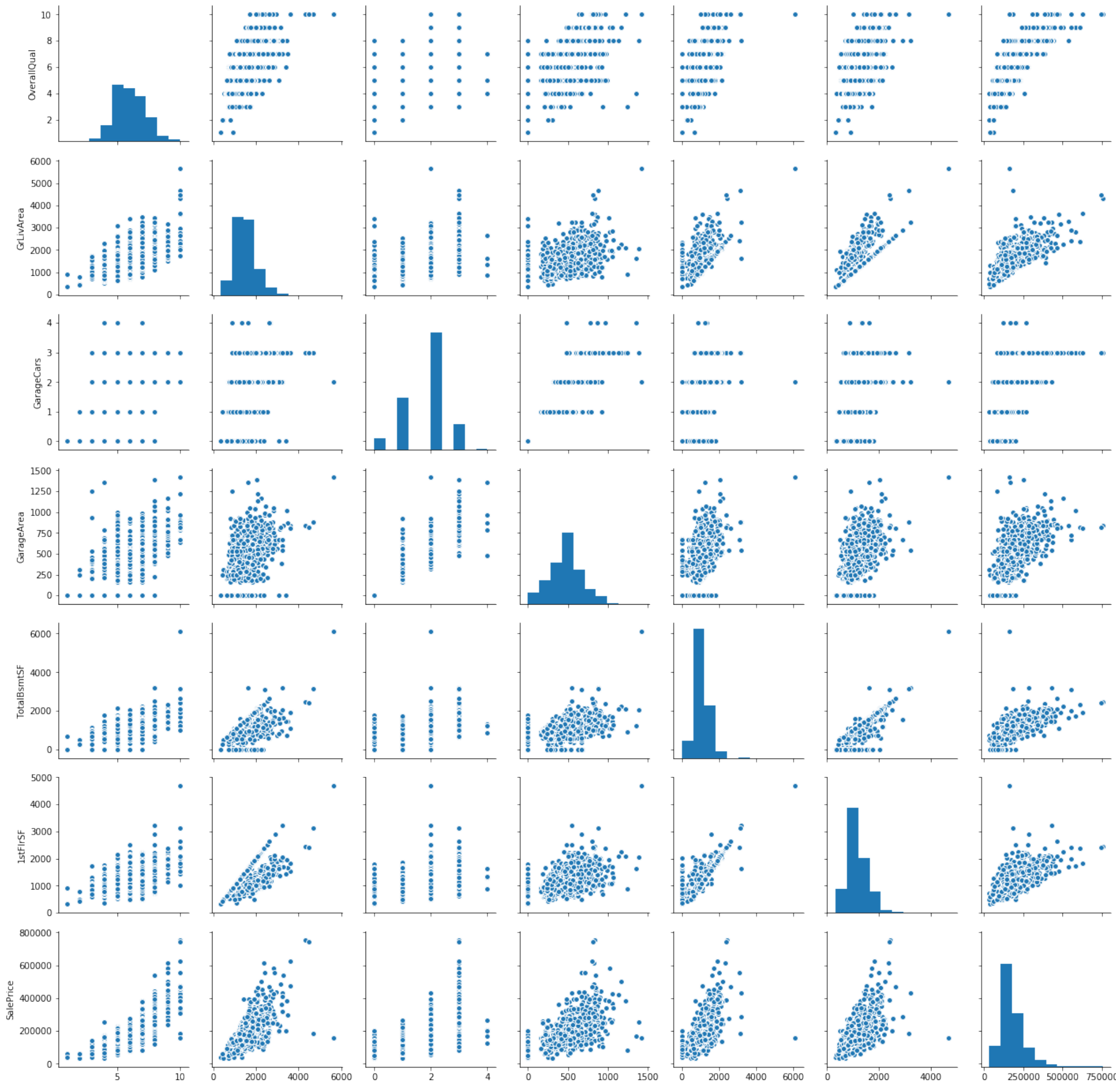
In [89]:
```python
tcf = tcf[tcf.index != 'SalePrice']
tcf
```

```
Out[89]: OverallQual     0.790982
         GrLivArea       0.708624
         GarageCars      0.640409
         GarageArea      0.623431
         TotalBsmtSF     0.613581
         1stFlrSF        0.605852
         Name: SalePrice, dtype: float64
```

```
In [90]: tcf.shape
```

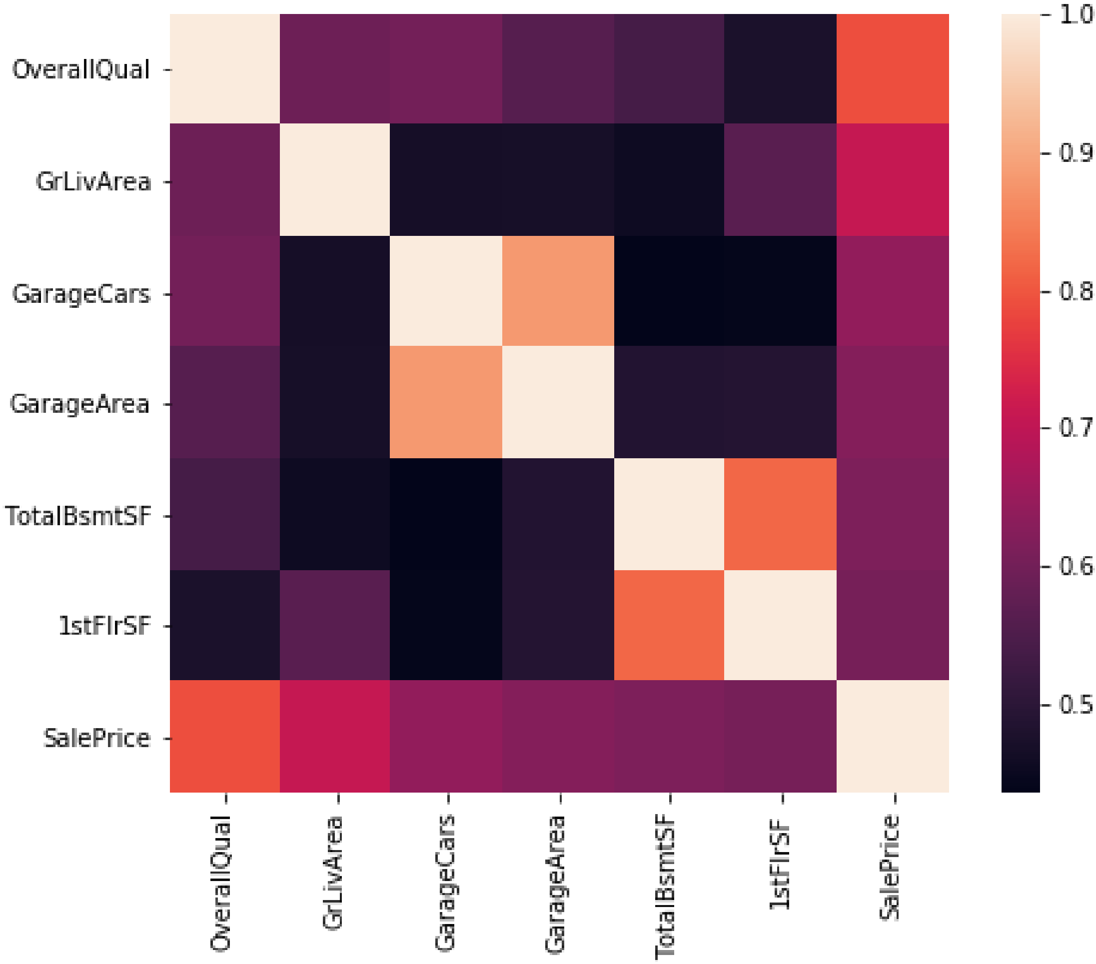
```
Out[90]: (6,)
```

```
In [91]: cols = tcf.index.values.tolist() + ['SalePrice']
         sns.pairplot(d[cols], size=2.5)
         plt.show()
```



In [94]:
```
# Build the correlation matrix
matrix = d[cols].corr()
f, ax = plt.subplots(figsize=(8, 6))
sns.heatmap(matrix, vmax=1.0, square=True)
```

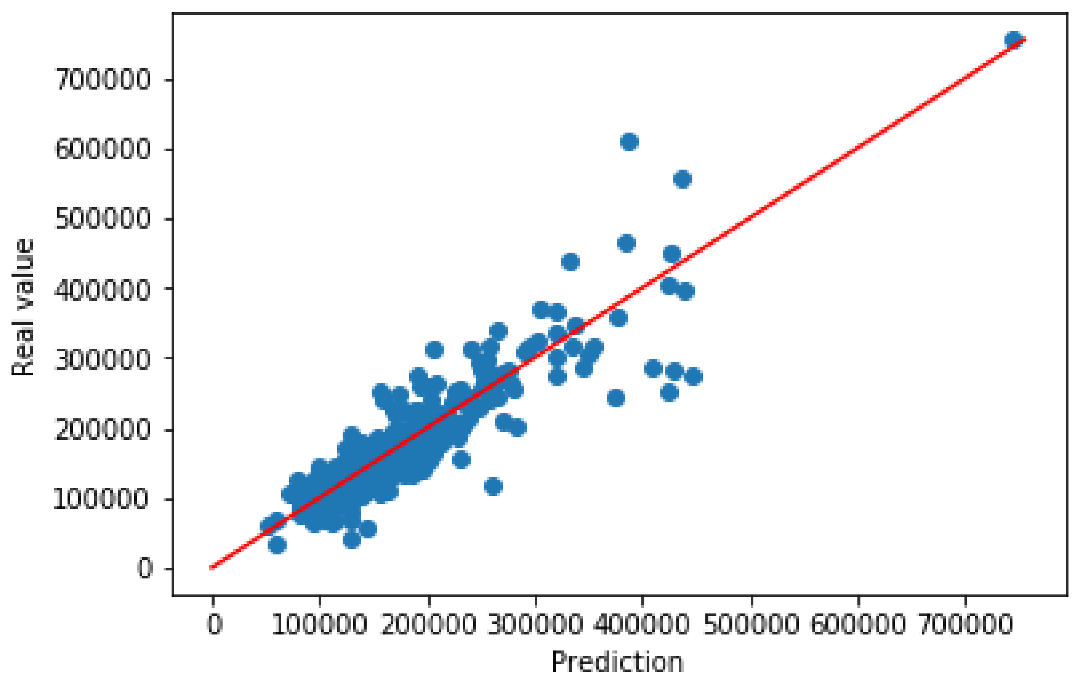
Out[94]: <matplotlib.axes._subplots.AxesSubplot at 0x1ad3bf9e550>



In [146]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
X=d1.loc[:,d1.columns!='SalePrice']
y = d1['SalePrice']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Out[146]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
                       oob_score=False, random_state=42, verbose=0, warm_start=False)

In [147]:
```
y_pred = model.predict(X_test)

# Build a plot
plt.scatter(y_pred, y_test)
plt.xlabel('Prediction')
plt.ylabel('Real value')

# Now add the perfect prediction line
diagonal = np.linspace(0, np.max(y_test), 100)
plt.plot(diagonal, diagonal, '-r')
plt.show()
```



In [148]:
```
from sklearn.metrics import mean_squared_log_error, mean_absolute_error

print('MAE:\t$%.2f' % mean_absolute_error(y_test, y_pred))
print('MSLE:\t%.5f' % mean_squared_log_error(y_test, y_pred))
```

```
MAE:         $26348.38
MSLE:         0.04555
```

```
In [149]: #Score/Accuracy
          print("Accuracy --> ", model.score(X_test, y_test)*100)
```

```
Accuracy -->  0.684931506849315
```

```
In [153]: #Train the model
          from sklearn import linear_model
          model = linear_model.LinearRegression()
```

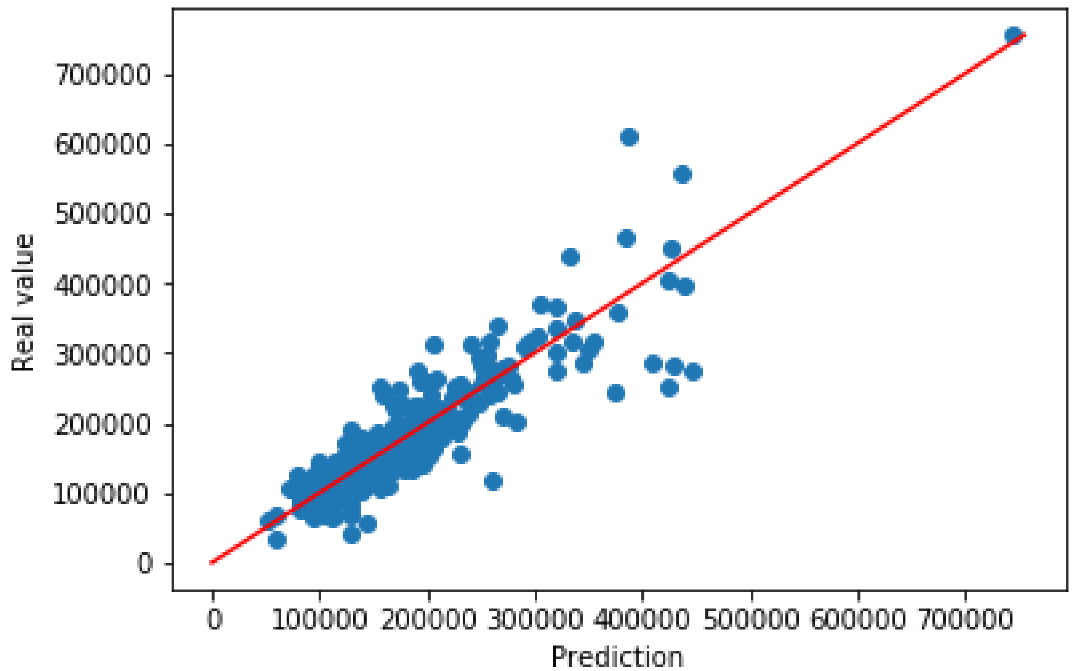
```
In [154]: #Fit the model
          model.fit(X_train, y_train)
```

```
Out[154]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```
In [157]: #Score/Accuracy
          print("Accuracy --> ", model.score(X_test, y_test)*100)
```

```
Accuracy -->  79.24553693088554
```

```
In [159]: # Build a plot
          plt.scatter(y_pred, y_test)
          plt.xlabel('Prediction')
          plt.ylabel('Real value')

          # Now add the perfect prediction line
          diagonal = np.linspace(0, np.max(y_test), 100)
          plt.plot(diagonal, diagonal, '-r')
          plt.show()
```



In [150]:
```
#Train the model
from sklearn.ensemble import GradientBoostingRegressor
GBR = GradientBoostingRegressor(n_estimators=100, max_depth=4)
```

In [151]:
```
#Fit
GBR.fit(X_train, y_train)
```

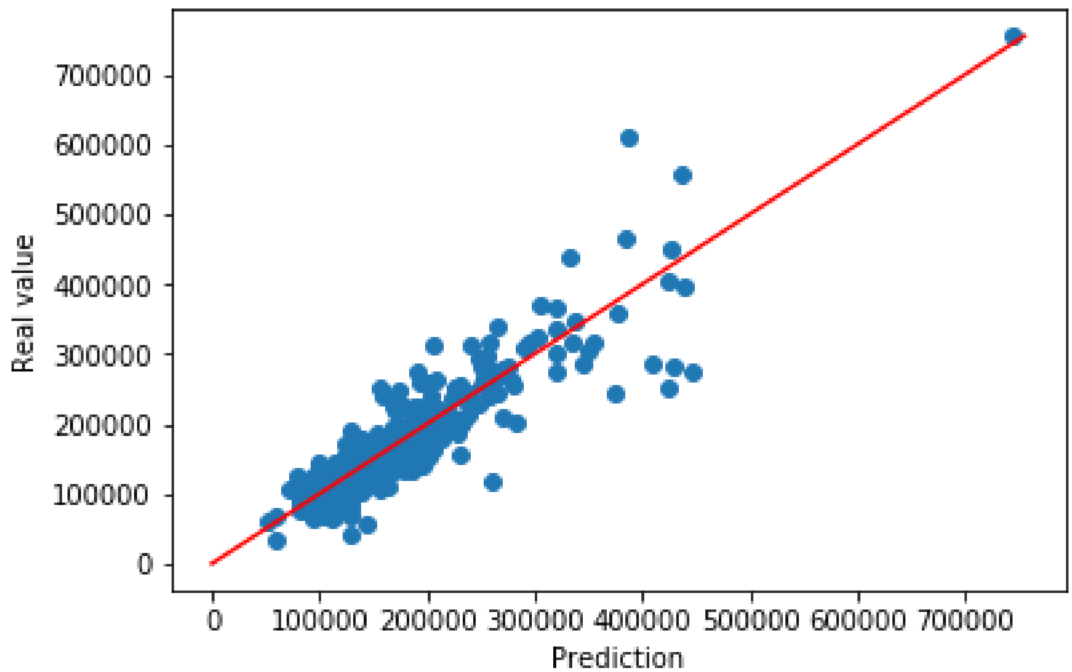
Out[151]:
```
GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
             learning_rate=0.1, loss='ls', max_depth=4, max_features=None,
             max_leaf_nodes=None, min_impurity_decrease=0.0,
             min_impurity_split=None, min_samples_leaf=1,
             min_samples_split=2, min_weight_fraction_leaf=0.0,
             n_estimators=100, presort='auto', random_state=None,
             subsample=1.0, verbose=0, warm_start=False)
```

In [152]:
```
print("Accuracy --> ", GBR.score(X_test, y_test)*100)
```

```
Accuracy -->  87.72567683930332
```

In [158]:
```
# Build a plot
plt.scatter(y_pred, y_test)
plt.xlabel('Prediction')
plt.ylabel('Real value')
```

```
# Now add the perfect prediction line
diagonal = np.linspace(0, np.max(y_test), 100)
plt.plot(diagonal, diagonal, '-r')
plt.show()
```




```
In [ ]:
```