数据主要包括2014年5月至2015年5月美国King County的房屋销售价格以及房屋的基本信息。

数据分为训练数据和测试数据，分别保存在kc_train.csv和kc_test.csv两个文件中。

其中训练数据主要包括10000条记录，14个字段，主要字段说明如下：

第一列“销售日期”：2014年5月到2015年5月房屋出售时的日期

第二列“销售价格”：房屋交易价格，单位为美元，是目标预测值

第三列“卧室数”：房屋中的卧室数目

第四列“浴室数”：房屋中的浴室数目

第五列“房屋面积”：房屋里的生活面积

第六列“停车面积”：停车坪的面积

第七列“楼层数”：房屋的楼层数

第八列“房屋评分”：King County房屋评分系统对房屋的总体评分

第九列“建筑面积”：除了地下室之外的房屋建筑面积

第十列“地下室面积”：地下室的面积

第十一列“建筑年份”：房屋建成的年份

第十二列“修复年份”：房屋上次修复的年份

第十三列"纬度"：房屋所在纬度

第十四列“经度”：房屋所在经度

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
from scipy.stats import norm
import xgboost as xgb
```

In [2]:

```
#给每一列数据进行命名
columns_names = [
    'saleDate',
    'bedroomNum',
    'bathroomNum',
    'houseArea', #房屋面积”：房屋里的生活面积
    'parkingArea',
    'floorNum',
    'houseRank',
    'coveredArea', # “建筑面积”：除了地下室之外的房屋建筑面积
    'basementArea',
    'buildYear',
    'repairedYear',
    'longitude',
    'latitude',
    'salePrice'
]

#读取训练数据
data = pd.read_csv("C://kc_train.csv", names=columns_names)
```

一. 数据清洗（分为3步进行）

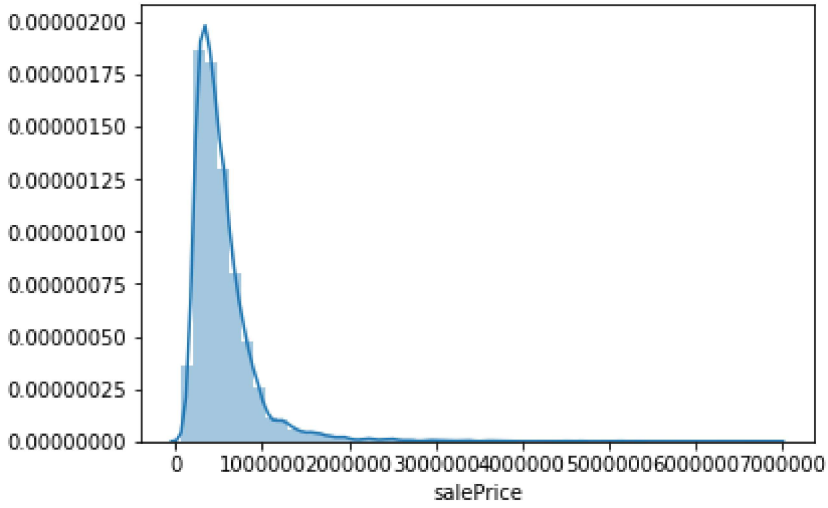
In [3]:

```
#对因变量salePrice进行分析
print(data['salePrice'].describe())
sns.distplot(data['salePrice'])
plt.show()

#分析salePrice的偏态和峰度
print("Skewness: %f" % data['salePrice'].skew())
print("Kurtosis: %f" % data['salePrice'].kurt())
```

```
count     1.000000e+04
mean      5.428749e+05
std       3.729258e+05
min       7.500000e+04
25%       3.225000e+05
50%       4.507000e+05
75%       6.450000e+05
max       6.885000e+06
Name: salePrice, dtype: float64

D:\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462: UserWarning: The 'norm
ed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```



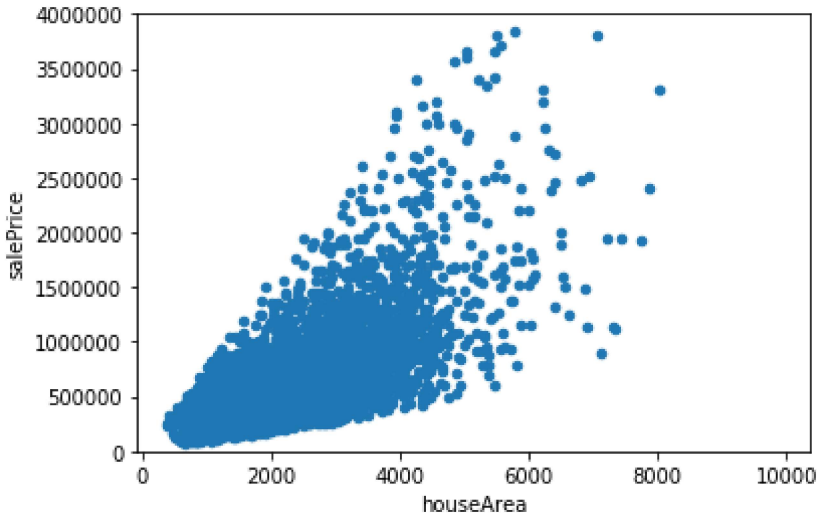

```
Skewness: 3.898737
Kurtosis: 29.356202
```

Flexibly plot a univariate distribution of observations.

1. 根据我们的直觉，会觉得房价与住房面积，地下室面积，排名，建筑年份有关，根据这样的直觉对房价与这些变量的关系进行逐一分析，但是需要强调，这只是我们的主观臆断

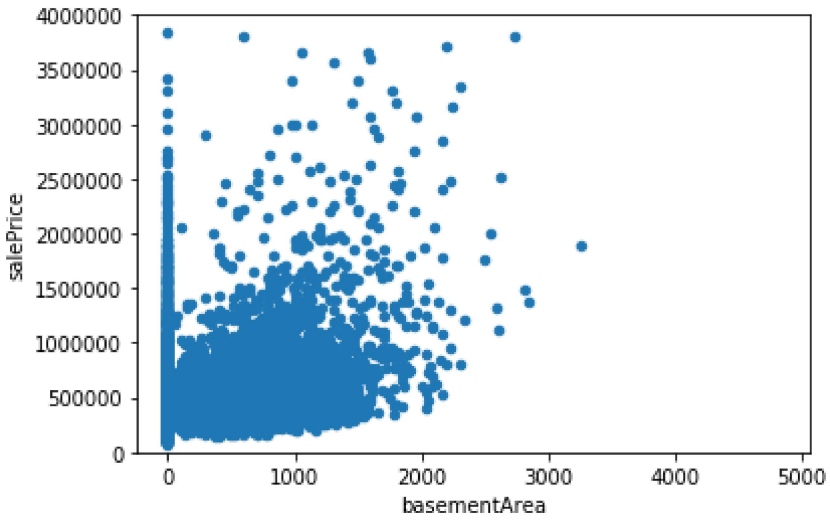
In [4]:

```
#scatter plot houseArea/saleprice
#salePrice和houseArea具有线性关系
var = 'houseArea'
data1 = pd.concat([data['salePrice'], data[var]], axis=1)
data1.plot.scatter(x=var, y='salePrice', ylim=(0,4000000))
plt.show()
```



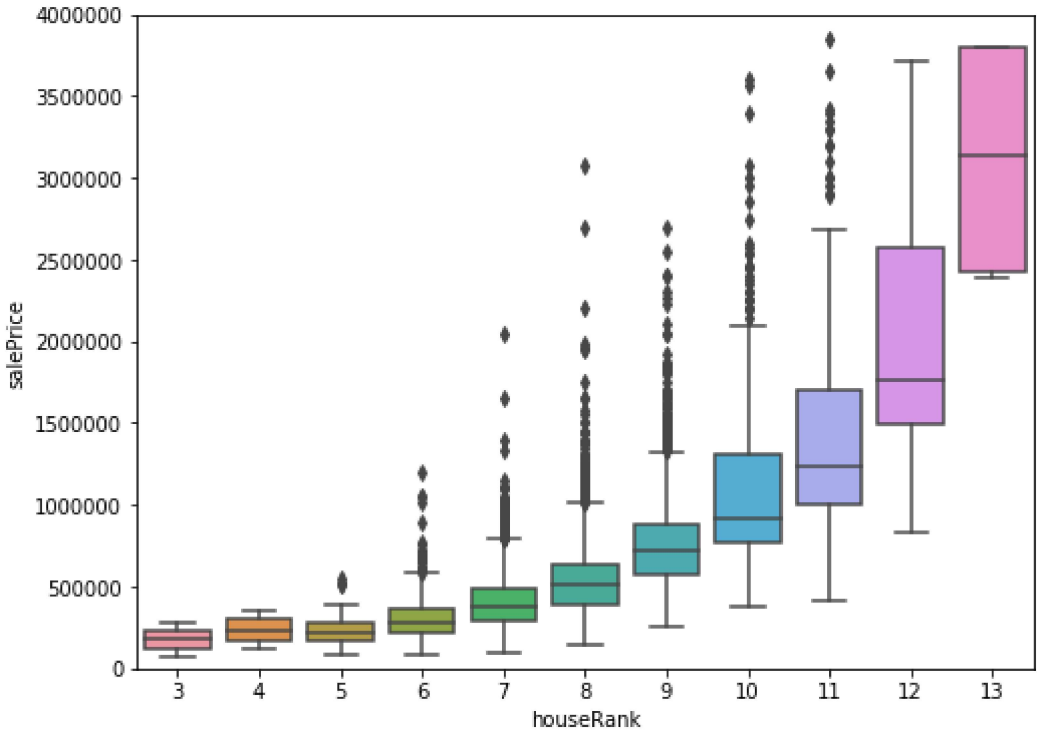
In [5]:

```
#scatter plot basementArea/salePrice
#salePrice和houseArea具有强线性关系，突然间具有指数关系
var = 'basementArea'
data1 = pd.concat([data['salePrice'], data[var]], axis=1)
data1.plot.scatter(x=var, y='salePrice', ylim=(0, 4000000))
plt.show()
```



In [6]:

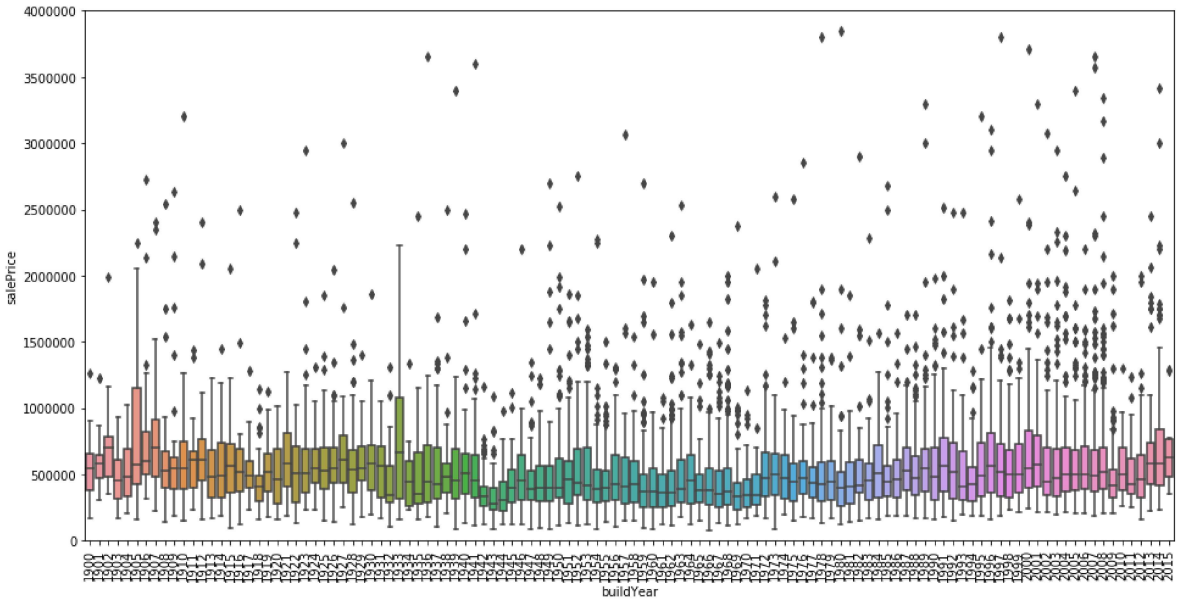
```
#box plot houseRank /saleprice
#salePrice and houseRank seems to have positive ralation
var = 'houseRank'
data1 = pd.concat([data['salePrice'], data[var]], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var, y="salePrice", data=data1)
fig.axis(ymin=0, ymax=4000000)
plt.show()
```



A box plot (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable. The box shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution, except for points that are determined to be “outliers” using a method that is a function of the inter-quartile range.

In [7]:

```
#plot bulidYear/saleprice
#'salePrice' is more prone to spend more money in new stuff than in old relics.
var = 'buildYear'
data1 = pd.concat([data['salePrice'], data[var]], axis=1)
f, ax = plt.subplots(figsize=(16, 8))
fig = sns.boxplot(x=var, y="salePrice", data=data1)
fig.axis(ymin=0, ymax=4000000)
plt.xticks(rotation=90)
plt.show()
```

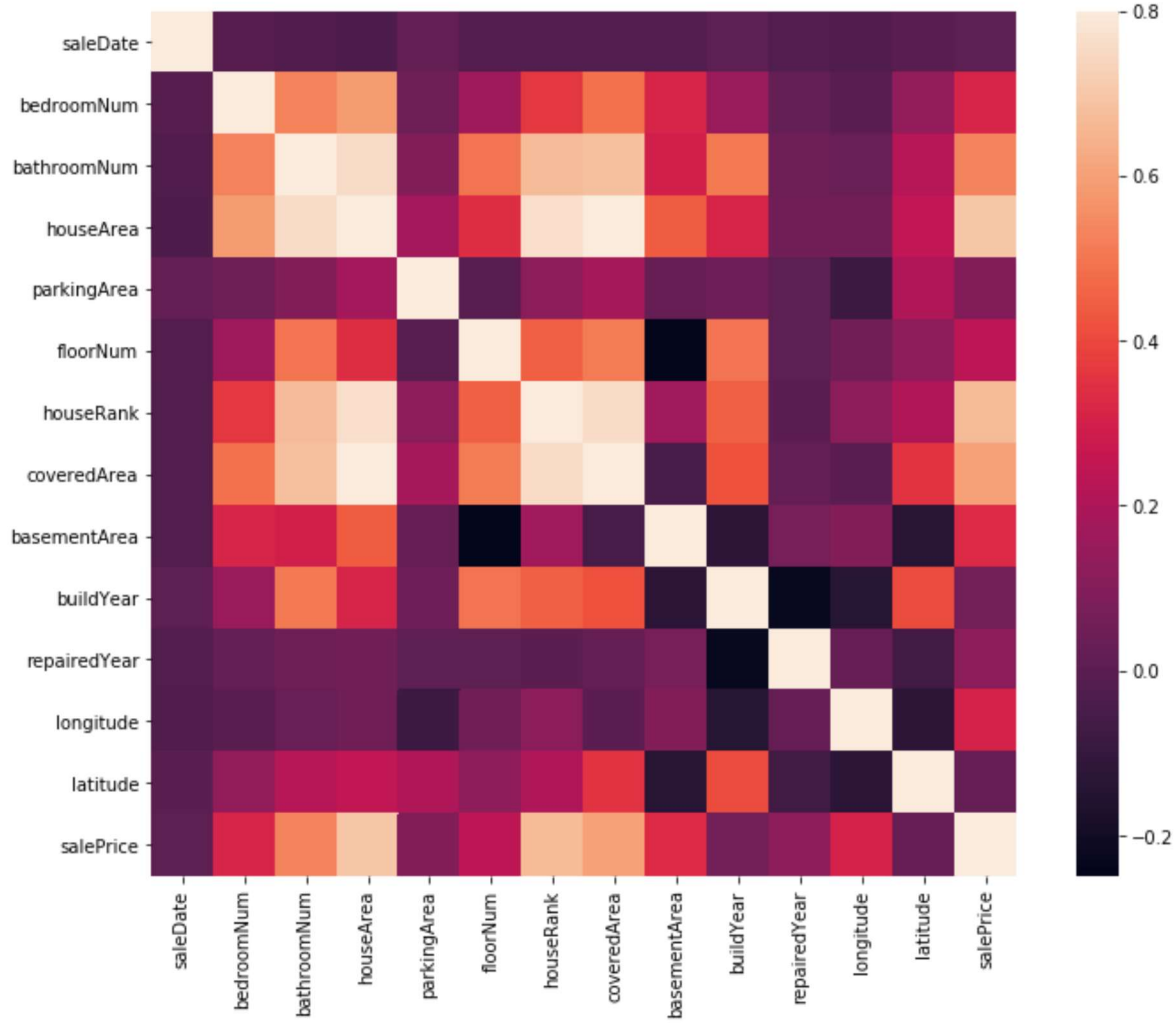


2. 接下来我们应该冷静下来，做一些看起来更加智慧的工作。我们应该考虑所有的特征，进行地毯式搜索，不应该忽略我们看起来不重要的因素

In [8]:

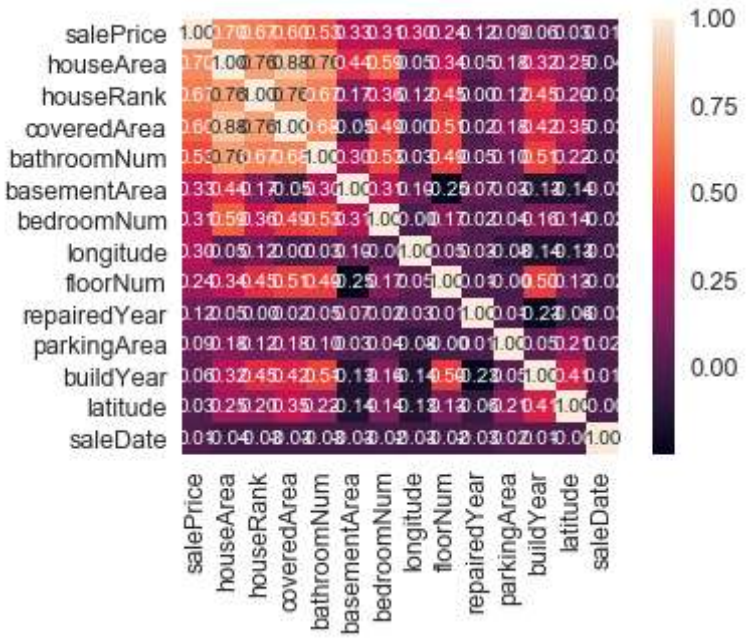
```
#correlation matrix
corrmat = data.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True)
plt.show()#将训练数据划分为训练集和测试集
from sklearn.cross_validation import train_test_split
x_train, x_test, y_train, y_test = train_test_split(
    data[['houseArea', 'houseRank', 'coveredArea', 'bathroomNum', 'basementArea', 'bedroomNum', 'lor
    data[['salePrice']],
    test_size=0.25,
    random_state=12
)

#feature归一化
from sklearn.preprocessing import StandardScaler
ss_x = StandardScaler()
x_train = ss_x.fit_transform(x_train)
x_test = ss_x.transform(x_test)
```



In [11]:

```
#saleprice correlation matrix
#从相关矩阵，我们看到主观猜想可能会有问题
#不过前5个因素我们猜中了2个，其中前2都押中了
k = 14  #number of variables for heatmap
cols = corrmat.nlargest(k, 'salePrice')['salePrice'].index
cm = np.corrcoef(data[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size': 10}, yticklabels=cols.values, xticklabels=cols.values)
plt.show()
```



In [12]:

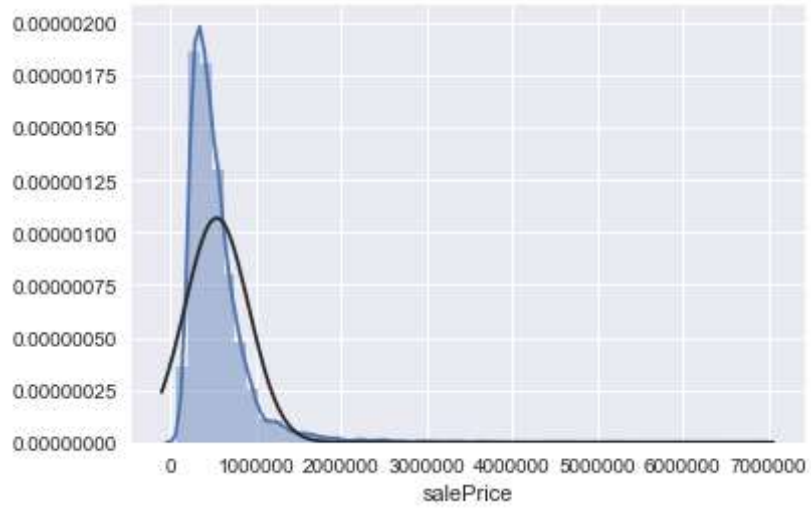
```
#scatterplot
#从相关矩阵我们看到，houseArea和coverArea相关性达到了0.88，取houseArea忽略coverArea
#做出salePrice和其他7个因素的图形
sns.set()
cols = ['salePrice', 'houseArea', 'houseRank', 'coveredArea','bathroomNum','basementArea', 'longitud
sns.pairplot(data[cols], size = 2.5)
plt.show()
```

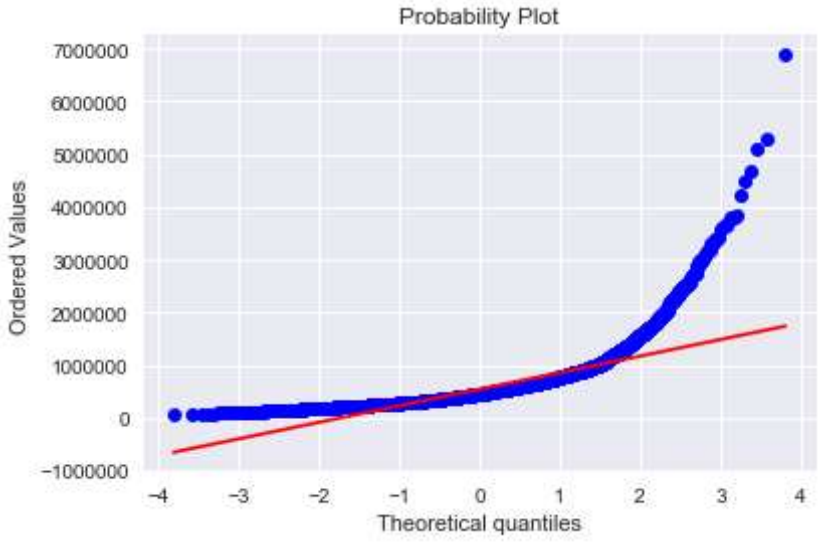
3. 正态，同性

In [13]:

```
#1)process salePrice
#histogram and normal probability plot
#由画出的图形可知，salePrice不正常。 它显示“峰值”，正偏斜并且不跟随对角线。
sns.distplot(data['salePrice'], fit=norm)
fig = plt.figure()
res = stats.probplot(data['salePrice'], plot=plt)
plt.show()
```

```
D:\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462: UserWarning: The 'norm
ed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```





Calculate quantiles for a probability plot, and optionally show the plot.

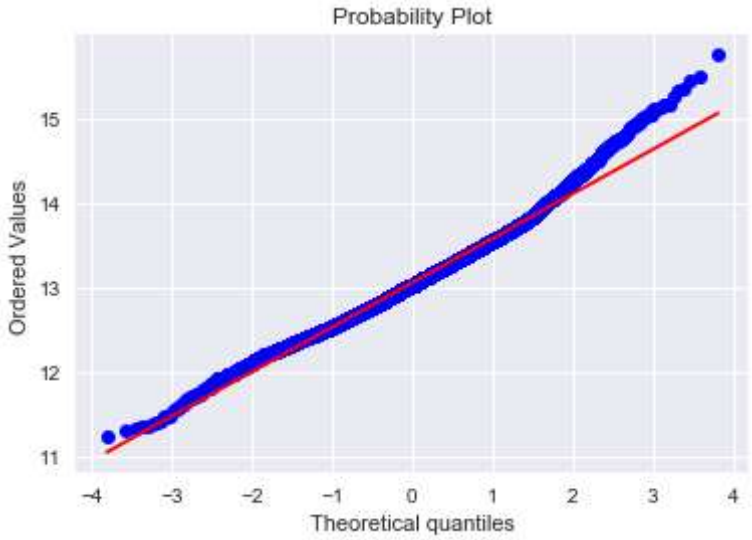
Generates a probability plot of sample data against the quantiles of a specified theoretical distribution (the normal distribution by default). probplot optionally calculates a best-fit line for the data and plots the results using Matplotlib or a given plot function.

正态概率图 - 数据分布应紧密跟随代表正态分布的对角线。

In [14]:

```
#applying log transformation
#此时scalePrice满足正态分布，跟随对角线
data['salePrice'] = np.log(data['salePrice'])
sns.distplot(data['salePrice'], fit=norm)
fig = plt.figure()
res = stats.probplot(data['salePrice'], plot=plt)
plt.show()
```

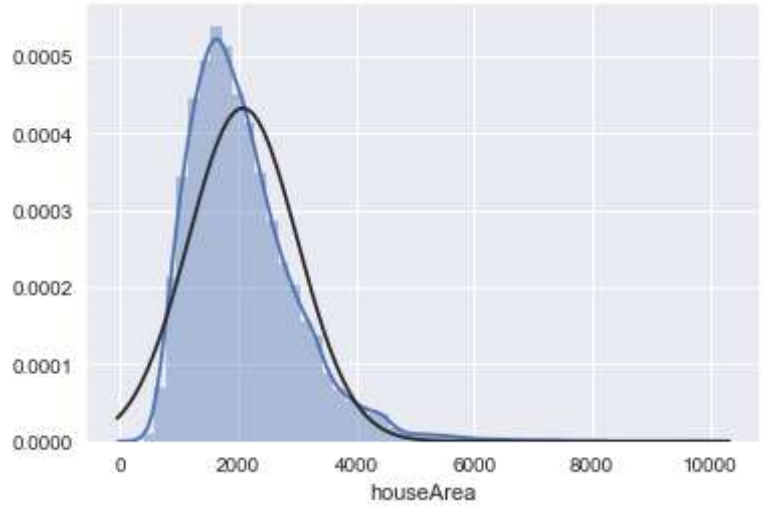
```
D:\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462: UserWarning: The 'norm
ed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

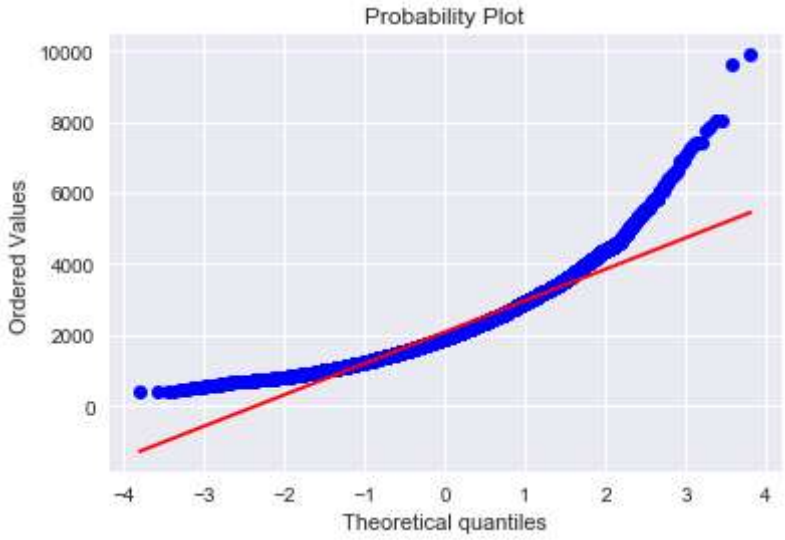


In [15]:

```
#2)process houseArea
#histogram and normal probability plot
sns.distplot(data['houseArea'], fit=norm)
fig = plt.figure()
res = stats.probplot(data['houseArea'], plot=plt)
plt.show()
```

```
D:\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462: UserWarning: The 'norm
ed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

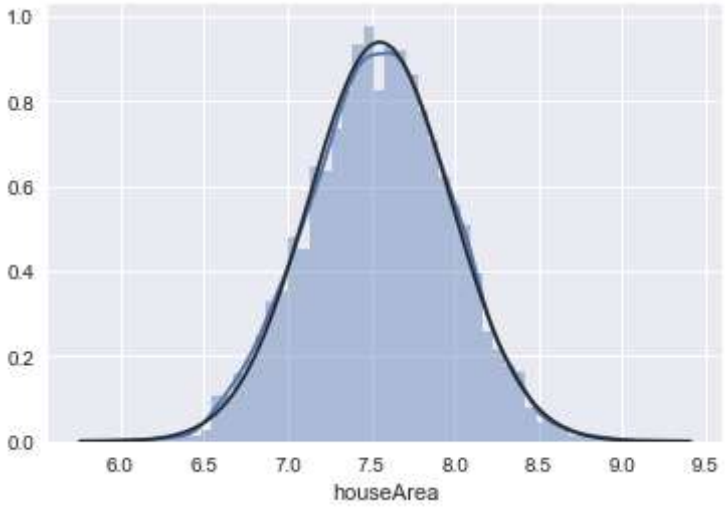


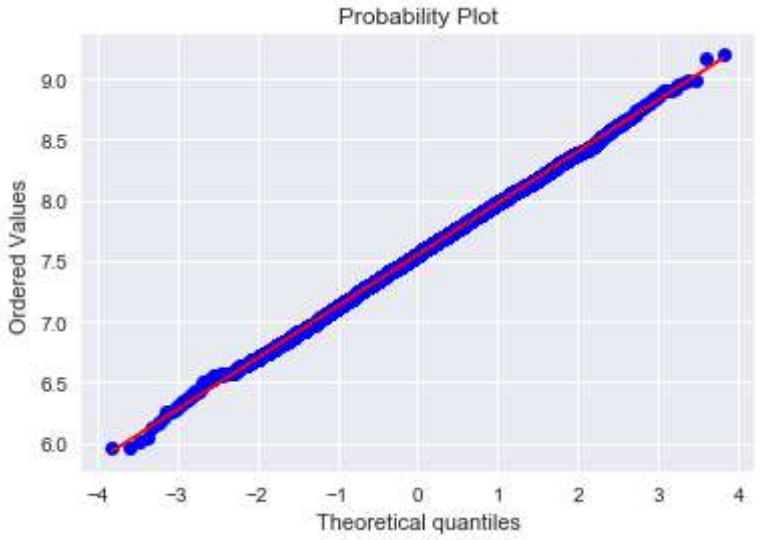


In [16]:

```
#data transformation
data['houseArea'] = np.log(data['houseArea'])
sns.distplot(data['houseArea'], fit=norm)
fig = plt.figure()
res = stats.probplot(data['houseArea'], plot=plt)
plt.show()
```

```
D:\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462: UserWarning: The 'norm
ed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

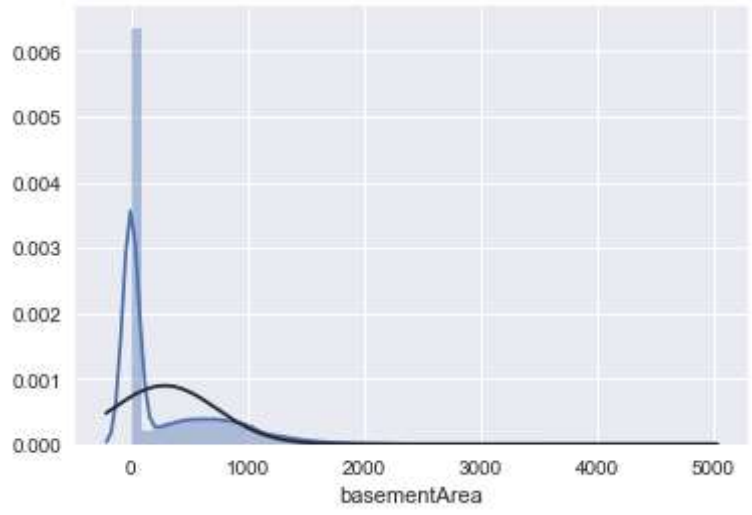


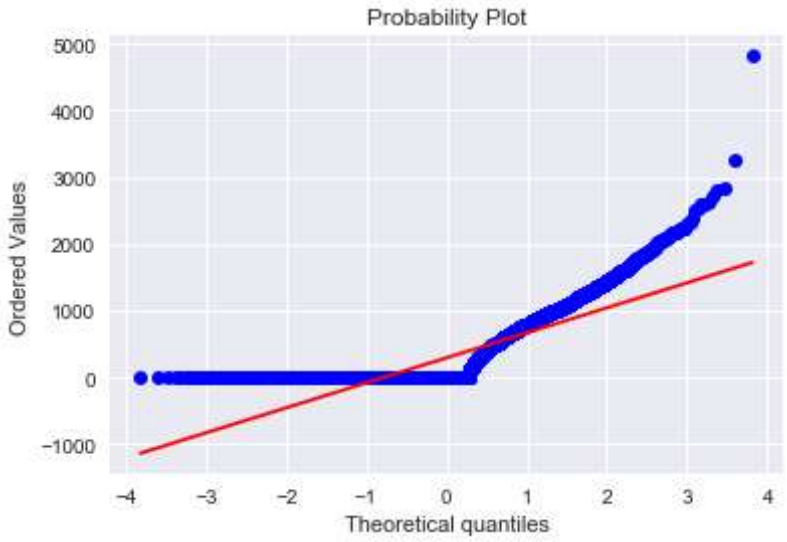


In [17]:

```
#3)process basementArea
#histogram and normal probability plot
#由图形可知，大量的房子并没有地下室
#一个大问题，因为零值不允许我们进行日志转换。
#为了在这里应用日志转换，我们将创建一个变量来获得有或没有地下室（二进制变量）的效果。
#然后，我们将对所有非零观测值进行对数变换，忽略值为零的观测值。
#这样我们就可以转换数据，而不会失去有或没有地下室的影响。
sns.distplot(data['basementArea'], fit=norm)
fig = plt.figure()
res = stats.probplot(data['basementArea'], plot=plt)
plt.show()
```

```
D:\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462: UserWarning: The 'norm
ed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

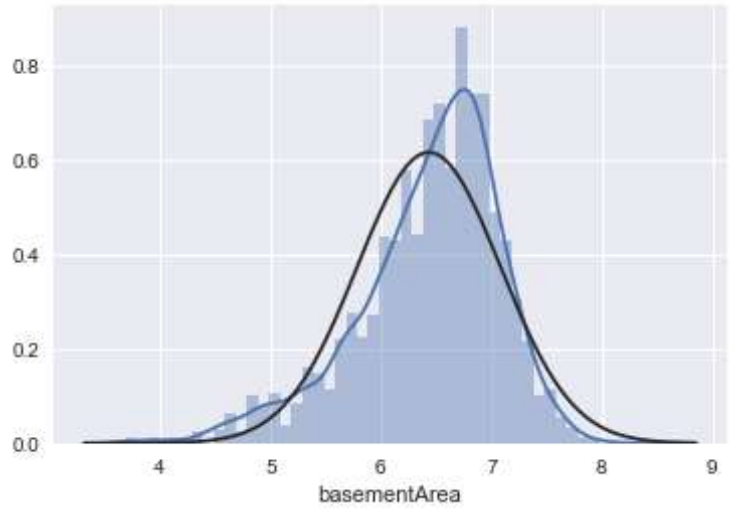


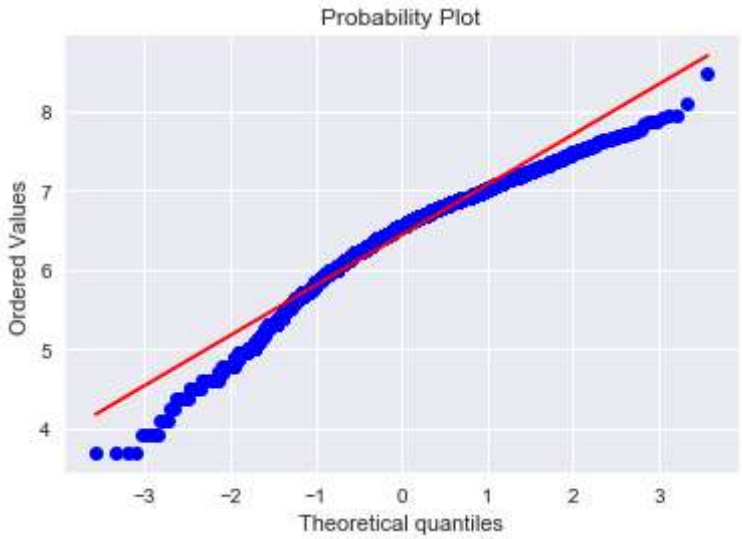


In [18]:

```
#create column for new variable (one is enough because it's a binary categorical feature)
#if area>0 it gets 1, for area==0 it gets 0
data['HasBsmt'] = pd.Series(len(data['basementArea']), index=data.index)
data['HasBsmt'] = 0
data.loc[data['basementArea']>0,'HasBsmt'] = 1
#transform data
data.loc[data['HasBsmt']==1,'basementArea'] = np.log(data['basementArea'])
#histogram and normal probability plot
sns.distplot(data[data['basementArea']>0]['basementArea'], fit=norm)
fig = plt.figure()
res = stats.probplot(data[data['basementArea']>0]['basementArea'], plot=plt)
plt.show()
```

```
D:\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: RuntimeWarning: divide by ze
ro encountered in log
  import sys
D:\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462: UserWarning: The 'norm
ed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

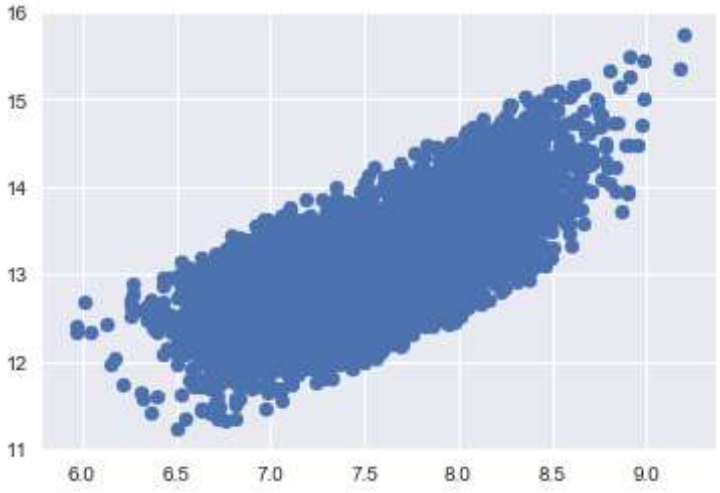


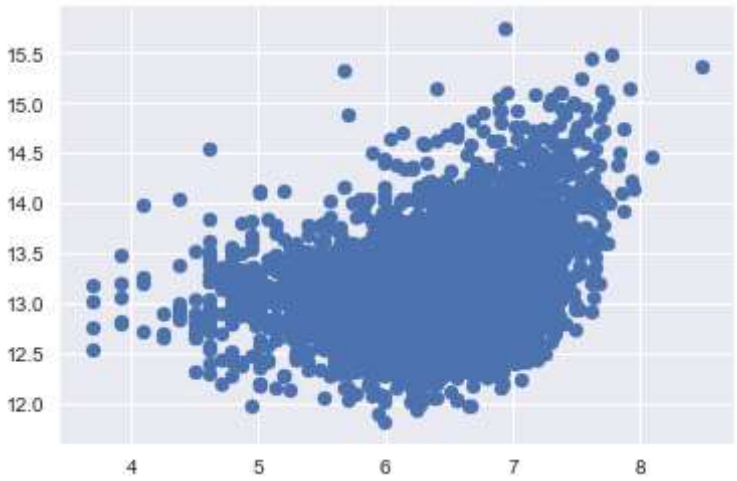


In [19]:

```
#scatter plot
plt.scatter(data['houseArea'], data['salePrice'])
plt.show()

plt.scatter(data[data['basementArea']>0]['basementArea'], data[data['basementArea']>0]['salePrice'])
plt.show()
```





二. 采用特征筛选，选取重要的一些特征(不要主观臆断)

Filter法更好，但是最后效果并不理想

In [20]:

```
#1. Wrapper 法
x= data.drop(['salePrice'],axis =1,inplace=False)
x = np.array(x)
y = data[['salePrice']]
y = np.array(y)
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE
model1 = LinearRegression()
rfe = RFE(model1, 8)
rfe = rfe.fit(x, y)
print(rfe.support_)
print(rfe.ranking_)
```

```
D:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shap
e of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

[False  True  True  True False  True  True False False False False  True
  True  True]
[6 1 1 1 7 1 1 4 2 3 5 1 1 1]
```

In [21]:

```
from sklearn.model_selection import cross_val_score
x_test1 = data[['bedroomNum', 'bathroomNum', 'houseArea', 'floorNum', 'houseRank','longitude','latit
x_test1 = np.array(x_test1)
y_test1 = data[['salePrice']]
y_test1 = np.array(y_test1)
model2 = LinearRegression()
model2.fit(x_test1,y_test1)
scores = -cross_val_score(model2, x_test1, y_test1, cv=5, scoring= 'neg_mean_absolute_error')
print("Wrapper 法:", np.mean(scores))
```

```
Wrapper 法: 0.22233246700994277
```

In [22]:

```
#2. Filter法
from sklearn.feature_selection import SelectKBest, f_classif
selector = SelectKBest(f_classif, k=8)
a = selector.fit(x, y)
print(np.array(a.scores_),'\n',a.get_support())
```

```
D:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shap
e of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

[1.01172447 1.58116643 2.9169126  4.156027   0.85029036 1.4149715
 5.35734239 3.99941001 1.2627663  1.22412787 1.23153039 2.63561764
 1.02468174 1.201535  ]
 [False  True  True  True False  True  True  True  True False False  True
 False False]
```

In [23]:

```
x_test2 = data[['bedroomNum', 'bathroomNum', 'houseArea', 'floorNum', 'houseRank', 'coveredArea', 'b
x_test2 = np.array(x_test2)
y_test2 = data[['salePrice']]
y_test2 = np.array(y_test2)
model3 = LinearRegression()
model3.fit(x_test2,y_test2)
scores = -cross_val_score(model3, x_test2, y_test2, cv=5, scoring= 'neg_mean_absolute_error')
print("Filter法:", np.mean(scores))
```

Filter法: 0.22090997598352585

三. 数据的预处理

In [24]:

```
#将训练数据划分为训练集和测试集
from sklearn.cross_validation import train_test_split
x_train, x_test, y_train, y_test = train_test_split(
    data[['houseArea', 'houseRank', 'coveredArea', 'bathroomNum', 'basementArea', 'bedroomNum', 'lor
    data[['salePrice']],
    test_size=0.25,
    random_state=12
)

#feature归一化
from sklearn.preprocessing import StandardScaler
ss_x = StandardScaler()
x_train = ss_x.fit_transform(x_train)
x_test = ss_x.transform(x_test)
```

```
D:\Anaconda3\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWarning: T
his module was deprecated in version 0.18 in favor of the model_selection module int
o which all the refactored classes and functions are moved. Also note that the inter
face of the new CV iterators are different from that of this module. This module wil
l be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

四. 采用线性回归方法，simplest is the best

特征降维和添加相关性强的属性的高次方效果不好

In [25]:

```
#线性回归预测
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x_train, y_train)
lr_predict = lr.predict(x_test)
a = lr.intercept_
b = lr.coef_
print('y = {} + {} * X'.format(a,b))


#评估
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error, explained_variance_s
print('The value of mean_absolute_error  LinearRegression is: ', mean_absolute_error(y_test, lr_pred
print('The value of R-squared  LinearRegression is: ', r2_score(y_test, lr_predict))
print('The mean squared error of LinearRegression is: ', mean_squared_error(y_test, lr_predict))
```

```
y = [13.05008984] + [[ 0.10830947  0.21989395  0.07830677  0.06403723  0.04933601 -
0.03255442
   0.17945922  0.01913419  0.01131348  0.01456901 -0.12957067 -0.00089501
   0.02129878]] * X
The value of mean_absolute_error  LinearRegression is:  0.20146538352634386
The value of R-squared  LinearRegression is:  0.7657466848688663
The mean squared error of LinearRegression is:  0.06969716901799257
```

最后的误差是100万会有7万的误差

五. 采用随机森林方法

采用默认参数

In [26]:

```
#1. default parameters
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor()
rfr.fit(x_train, y_train)
rfr_predict = rfr.predict(x_test)

#评估
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error, explained_variance_s
print('The value of mean_absolute_error  LinearRegression is: ', mean_absolute_error(y_test, rfr_pre
print('The mean squared error of LinearRegression is: ', mean_squared_error(y_test, rfr_predict))
```

```
D:\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A col
umn-vector y was passed when a 1d array was expected. Please change the shape of y t
o (n_samples,), for example using ravel().
  after removing the cwd from sys.path.

The value of mean_absolute_error  LinearRegression is:  0.14435697009191223
The mean squared error of LinearRegression is:  0.042015251526460705
```

调参过程

In [ ]:

```
#2. try to adjust some paras
#{'n_estimators': 120} 0.8540299570950255
from sklearn.grid_search import GridSearchCV
param_test1 = {'n_estimators':list(range(60, 151, 10))}
gsearch1 = GridSearchCV(estimator = RandomForestRegressor(),
                param_grid = param_test1,
                iid=False, cv=5)
gsearch1.fit(x_test, y_test)
print(gsearch1.grid_scores_, gsearch1.best_params_, gsearch1.best_score_)


#params: {'max_d {'max_depth': 15, 'min_samples_leaf': 2, 'min_samples_split': 3} 0.855263006999689
param_test2 = {'max_depth':list(range(5,20,2)), 'min_samples_split':list(range(2, 8, 1)), 'min_sampl
gsearch2 = GridSearchCV(estimator = RandomForestRegressor(n_estimators=120,
                                oob_score=True),
                param_grid = param_test2,
                iid=False, cv=5)
gsearch2.fit(x_test, y_test)
print(gsearch2.grid_scores_, gsearch2.best_params_, gsearch2.best_score_)


#{'max_features': 8} 0.8563327363495251
param_test3 = {'max_features':list(range(6,13,1))}
gsearch3 = GridSearchCV(estimator =  RandomForestRegressor(n_estimators=120, max_depth=15, min_sampl
                                min_samples_leaf=2 ,oob_score=True, random_state=10),
                param_grid = param_test3,
                iid=False, cv=5)
gsearch3.fit(x_test, y_test)
print(gsearch3.grid_scores_, gsearch3.best_params_, gsearch3.best_score_)
```

最优化的参数运行结果

In [28]:

```
#3. 综合考虑参数
rfr = RandomForestRegressor(n_estimators=120, max_depth=15, min_samples_split=3,
                                min_samples_leaf=2 ,oob_score=True, random_state=10,
                                max_features=8)
rfr.fit(x_train, y_train)
rfr_predict = rfr.predict(x_test)

print('The value of mean_absolute_error  LinearRegression is: ', mean_absolute_error(y_test, rfr_pre
print('The mean squared error of LinearRegression is: ', mean_squared_error(y_test, rfr_predict))
```

```
D:\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: DataConversionWarning: A col
umn-vector y was passed when a 1d array was expected. Please change the shape of y t
o (n_samples,), for example using ravel().
  """

The value of mean_absolute_error  LinearRegression is:  0.13386053683232788
The mean squared error of LinearRegression is:  0.03635544148587335
```

最后的误差是100W会有3.7W的误差

六. gradient boosting tree

默认参数

In [29]:

```
#1. default parameters
from sklearn.ensemble import GradientBoostingRegressor
gbr = GradientBoostingRegressor()
gbr.fit(x_train, y_train)
gbr_predict = gbr.predict(x_test)

#评估
print('The value of mean_absolute_error  LinearRegression is: ', mean_absolute_error(y_test, gbr_pre
print('The mean squared error of LinearRegression is: ', mean_squared_error(y_test, gbr_predict))
```

```
D:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shap
e of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

The value of mean_absolute_error  LinearRegression is:  0.13952390917589513
The mean squared error of LinearRegression is:  0.037709130043416896
```

调参过程

In [ ]:

```
#2. try to adjust some paras
#{'n_estimators': 280} 0.869689580604794
param_test1 = {'n_estimators':list(range(150, 500, 10))}
gsearch1 = GridSearchCV(estimator = GradientBoostingRegressor(learning_rate=0.1),
                param_grid = param_test1,
                iid=False, cv=5)
gsearch1.fit(x_test, y_test)
print(gsearch1.grid_scores_, gsearch1.best_params_, gsearch1.best_score_)

#{'max_depth': 3, 'min_samples_split': 2} 0.8693885689236183
param_test2 = {'max_depth':list(range(3, 20, 2)), 'min_samples_split':list(range(2, 10, 1))}
gsearch2 = GridSearchCV(estimator=GradientBoostingRegressor(learning_rate=0.1, n_estimators=280),
                param_grid = param_test2,
                iid=False, cv=5)
gsearch2.fit(x_test, y_test)
print(gsearch2.grid_scores_, gsearch2.best_params_, gsearch2.best_score_)

#{'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 2} 0.869488325512437
param_test3 = {'max_depth':list(range(3, 8, 1)),'min_samples_split':list(range(2, 5, 1)), 'min_sampl
gsearch3 = GridSearchCV(estimator = GradientBoostingRegressor(learning_rate=0.1, n_estimators=280),
                        param_grid = param_test3,
                        iid=False, cv=5)
gsearch3.fit(x_test, y_test)
print(gsearch3.grid_scores_, gsearch3.best_params_, gsearch3.best_score_)
```

最优化的参数运行结果

In [30]:

```
#3. 综合考虑参数
gbr = GradientBoostingRegressor(learning_rate=0.1, n_estimators=280, max_depth=5, min_samples_leaf=3
gbr.fit(x_train, y_train)
gbr_predict = gbr.predict(x_test)
print('The value of mean_absolute_error  LinearRegression is: ', mean_absolute_error(y_test, gbr_pre
print('The mean squared error of LinearRegression is: ', mean_squared_error(y_test, gbr_predict))
```

```
D:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shap
e of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

The value of mean_absolute_error  LinearRegression is:  0.1283800059114201
The mean squared error of LinearRegression is:  0.03353636776454574
```

提高泛化能力

In [31]:

```
#为了提高泛化能力，步长缩小一半，最大迭代次数增加2倍
gbr = GradientBoostingRegressor(learning_rate=0.05, n_estimators=560, max_depth=5, min_samples_leaf=
gbr.fit(x_train, y_train)
gbr_predict = gbr.predict(x_test)
print('The value of mean_absolute_error  LinearRegression is: ', mean_absolute_error(y_test, gbr_pre
print('The mean squared error of LinearRegression is: ', mean_squared_error(y_test, gbr_predict))
```

```
D:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shap
e of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

The value of mean_absolute_error  LinearRegression is:  0.12799881737039653
The mean squared error of LinearRegression is:  0.033682626307807106
```

最后的误差100W会有3.4W的差距

七. xgboost

默认参数

In [32]:

```
#1. default params
xlf = xgb.XGBRegressor()
xlf.fit(x_train, y_train)
xlf_predict = xlf.predict(x_test)

from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error, explained_variance_s
print('The value of mean_absolute_error  LinearRegression is: ', mean_absolute_error(y_test, xlf_pre
print('The mean squared error of LinearRegression is: ', mean_squared_error(y_test, xlf_predict))
```

```
The value of mean_absolute_error  LinearRegression is:  0.14103485417372905
The mean squared error of LinearRegression is:  0.03837361435310901
```

调参过程

In [ ]:

```
#2. try to adjust some paras
from sklearn.grid_search import GridSearchCV
#{'n_estimators': 350} 0.8703262482985921
param_test1 = {'n_estimators':list(range(300,400,10))}
gsearch1 = GridSearchCV(estimator = xgb.XGBRegressor(),
                param_grid = param_test1,
                iid=False, cv=5)
gsearch1.fit(x_test, y_test)
print(gsearch1.grid_scores_, gsearch1.best_params_, gsearch1.best_score_)

#{'max_depth': 5, 'min_child_weight': 5} 0.8730765254359296
param_test2 = {
 'max_depth':list(range(3,10,2)),
 'min_child_weight':list(range(1,6,2))
}
gsearch2 = GridSearchCV(estimator = xgb.XGBRegressor( learning_rate =0.1, n_estimators=350),
 param_grid = param_test2,iid=False, cv=5)
gsearch2.fit(x_test, y_test)
print(gsearch2.grid_scores_, gsearch2.best_params_, gsearch2.best_score_)


#{'max_depth': 4, 'min_child_weight': 6} 0.8742104396845903
param_test2 = {
 'max_depth':[4,5,6],
 'min_child_weight':[4,5,6]
}
gsearch2 = GridSearchCV(estimator = xgb.XGBRegressor( learning_rate =0.1, n_estimators=350),
 param_grid = param_test2,iid=False, cv=5)
gsearch2.fit(x_test, y_test)
print(gsearch2.grid_scores_, gsearch2.best_params_, gsearch2.best_score_)

#{'gamma': 0.0} 0.8742104396845903
param_test3 = {
 'gamma':[i/10.0 for i in range(0,5)]
}
gsearch3 = GridSearchCV(estimator = xgb.XGBRegressor( learning_rate =0.1, n_estimators=350, max_dept

gsearch3.fit(x_test, y_test)
print(gsearch3.grid_scores_, gsearch3.best_params_, gsearch3.best_score_)

#{'colsample_bytree': 0.8, 'subsample': 0.7} 0.8757734283545815
param_test4 = {
 'subsample':[i/10.0 for i in range(6,10)],
 'colsample_bytree':[i/10.0 for i in range(6,10)]
}
from sklearn.grid_search import GridSearchCV
gsearch4 = GridSearchCV(estimator = xgb.XGBRegressor( learning_rate =0.1, n_estimators=350, max_dept

gsearch4.fit(x_test, y_test)
print(gsearch4.grid_scores_, gsearch4.best_params_, gsearch4.best_score_)



#{'reg_alpha': 1} 0.8792720701821735
param_test5 = {
 'reg_alpha':[1e-5, 1e-2, 0.1, 1, 100]
}
gsearch5 = GridSearchCV(estimator = xgb.XGBRegressor( learning_rate =0.1, n_estimators=350, max_dept
```

```
gsearch5.fit(x_test, y_test)
print(gsearch5.grid_scores_, gsearch5.best_params_, gsearch5.best_score_)

#{'reg_alpha': 0.9} 0.8805302243446643
param_test5 = {
 'reg_alpha':[0.8, 0.9, 1, 1.1, 1.2, 1.3, 1.4, 1.5]
}
gsearch5 = GridSearchCV(estimator = xgb.XGBRegressor( learning_rate =0.1, n_estimators=350, max_dept

gsearch5.fit(x_test, y_test)
print(gsearch5.grid_scores_, gsearch5.best_params_, gsearch5.best_score_)

#{'reg_lambda': 1} 0.8805302243446643
param_test5 = {
 'reg_lambda':[0.8, 0.9, 1, 1.1, 1.2, 1.3, 1.4, 1.5]
}
gsearch5 = GridSearchCV(estimator = xgb.XGBRegressor( learning_rate =0.1, n_estimators=350, max_dept

gsearch5.fit(x_test, y_test)
print(gsearch5.grid_scores_, gsearch5.best_params_, gsearch5.best_score_)
```

优化后的结果

In [33]:

```
#3. 优化以后的结果
xlf = xgb.XGBRegressor(learning_rate =0.1, n_estimators=350, max_depth = 4, min_child_weight = 6, ga
xlf.fit(x_train, y_train)
xlf_predict = xlf.predict(x_test)

from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error, explained_variance_s
print('The value of mean_absolute_error  LinearRegression is: ', mean_absolute_error(y_test, xlf_pre
print('The mean squared error of LinearRegression is: ', mean_squared_error(y_test, xlf_predict))
```

```
The value of mean_absolute_error  LinearRegression is:  0.1258569132347789
The mean squared error of LinearRegression is:  0.03215896150581125
```

最后的误差100W会有3.2W误差

八. 集成学习，将3个模型进行综合

In [35]:

```
#1. 进行简单的平均，mmp,性能变差
com_predict = (gbr_predict + rfr_predict + xlf_predict) / 3
print('The value of mean_absolute_error  LinearRegression is: ', mean_absolute_error(y_test, com_pre
print('The mean squared error of LinearRegression is: ', mean_squared_error(y_test, com_predict))
```

```
The value of mean_absolute_error  LinearRegression is:  0.12526586233615858
The mean squared error of LinearRegression is:  0.032481692298115705
```

In [ ]:

```
#2. 进行穷举搜索
min1 = min3 = 1
index1 = index2 = 0
for i in range(10):
    for j in range(10):
        if(i + j <= 10):
            com_predict = (i/10) * gbr_predict + (j/10) * rfr_predict + (10 - i - j)/10 * xlf_predi
            if mean_absolute_error(y_test, com_predict) < min1 and mean_squared_error(y_test, com_pr
                    min1 = mean_absolute_error(y_test, com_predict)
                    min2 = mean_squared_error(y_test, com_predict)
                    index1 = i
                    index2 = j
print(index1, index2)
```

3 1

九. 全部数据集用于训练, 获取最后的模型

In [37]:

```
x = data[['houseArea', 'houseRank', 'coveredArea', 'bathroomNum', 'basementArea', 'bedroomNum', 'lor
y = data[['salePrice']]
x = ss_x.transform(x)

rfr = RandomForestRegressor(n_estimators=120, max_depth=15, min_samples_split=3,
                            min_samples_leaf=2 ,oob_score=True, random_state=10,
                            max_features=8)
rfr.fit(x, y)

gbr = GradientBoostingRegressor(learning_rate=0.1, n_estimators=280, max_depth=5,
                            min_samples_leaf=3, min_samples_split=2)
gbr.fit(x, y)

xlf = xgb.XGBRegressor(learning_rate =0.1, n_estimators=350, max_depth = 4,
                            min_child_weight = 6, gamma = 0,colsample_bytree = 0.8, subsample =
                            reg_alpha = 0.9, reg_lambda = 1)
xlf.fit(x, y)
```

```
D:\Anaconda3\lib\site-packages\ipykernel_launcher.py:8: DataConversionWarning: A col
umn-vector y was passed when a 1d array was expected. Please change the shape of y t
o (n_samples,), for example using ravel().

D:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shap
e of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[37]:

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=0.8, gamma=0, learning_rate=0.1, max_delta_step=0,
       max_depth=4, min_child_weight=6, missing=None, n_estimators=350,
       n_jobs=1, nthread=None, objective='reg:linear', random_state=0,
       reg_alpha=0.9, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=True, subsample=0.7)
```

十. 回守襄阳

In [38]:

```
columns_names = [
    'saleDate',
    'bedroomNum',
    'bathroomNum',
    'houseArea',
    'parkingArea',
    'floorNum',
    'houseRank',
    'coveredArea',
    'basementArea',
    'buildYear',
    'repairedYear',
    'longitude',
    'latitude'
]

data = pd.read_csv("C://kc_test.csv", names=columns_names)
data['houseArea'] = np.log(data['houseArea'])
x = data[['houseArea', 'houseRank', 'coveredArea', 'bathroomNum', 'basementArea', 'bedroomNum', 'lor
x = ss_x.transform(x)


rfr_predict = rfr.predict(x)
gbr_predict = gbr.predict(x)
xlf_predict = xlf.predict(x)
com_predict = rfr_predict * 0.3 + gbr_predict * 0.1 + xlf_predict * 0.6

data['salePrice'] = np.exp(com_predict)
data.to_csv("D://kc_testResult.csv")
```