

Kaggle 竞赛——房价预测(House Prices)

算法及代码详述

Gaiss

2019.7

目 录

一、 数据来源及任务描述.....	3
a) 数据来源.....	3
b) 任务描述.....	3
二、 数据分析.....	3
a) 数据描述.....	3
b) 离群点处理.....	5
c) 偏度校正.....	8
d) 缺失值处理.....	10
e) 变量属性变换.....	12
三、 特征工程.....	13
四、 构建模型.....	14
五、 进行预测.....	17
六、 结果评价.....	17
七、 附录（变量解释）	18

一、数据来源及任务描述

a) 数据来源

本项目使用 kaggle 经典赛题——房价预测(House Prices: Advanced Regression Techniques) 进行实践。赛题链接如下：

<https://www.kaggle.com/c/house-prices-advanced-regression-techniques/leaderboard>

b) 任务描述

影响房价的因素有很多，在本题的数据集中有 79 个变量几乎描述了爱荷华州艾姆斯 (Ames, Iowa) 住宅的方方面面。最终目标要求预测出每间房屋的价格，对于测试集中的每一个 Id，给出变量 SalePrice 相应的值。关于 79 个变量的具体信息和中文解释附在文末附录中。

二、数据分析

a) 数据描述

首先我们导入数据并查看，其中训练集有 79 个预测变量和 1 个响应变量(SalePrice)：

```
train.columns
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

测试集有 79 个预测变量：

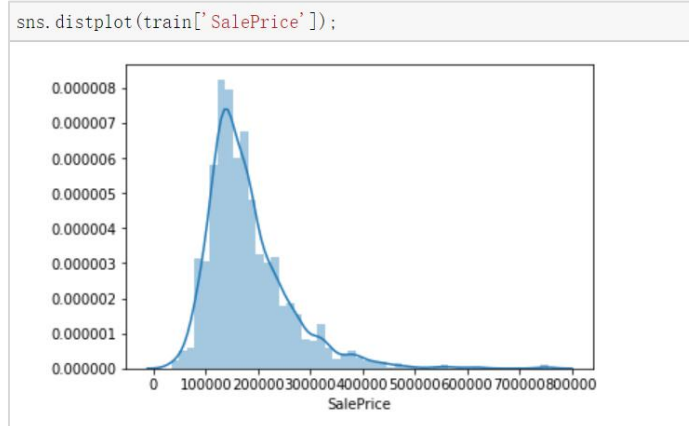
```
test.columns
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
      'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
      'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
      'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
      'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
      'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
      'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
      'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
      'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
      'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
      'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
      'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
      'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
      'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
      'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
      'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
      'SaleCondition'],
      dtype=object)
```

我们先着眼于响应变量 **SalePrice**。

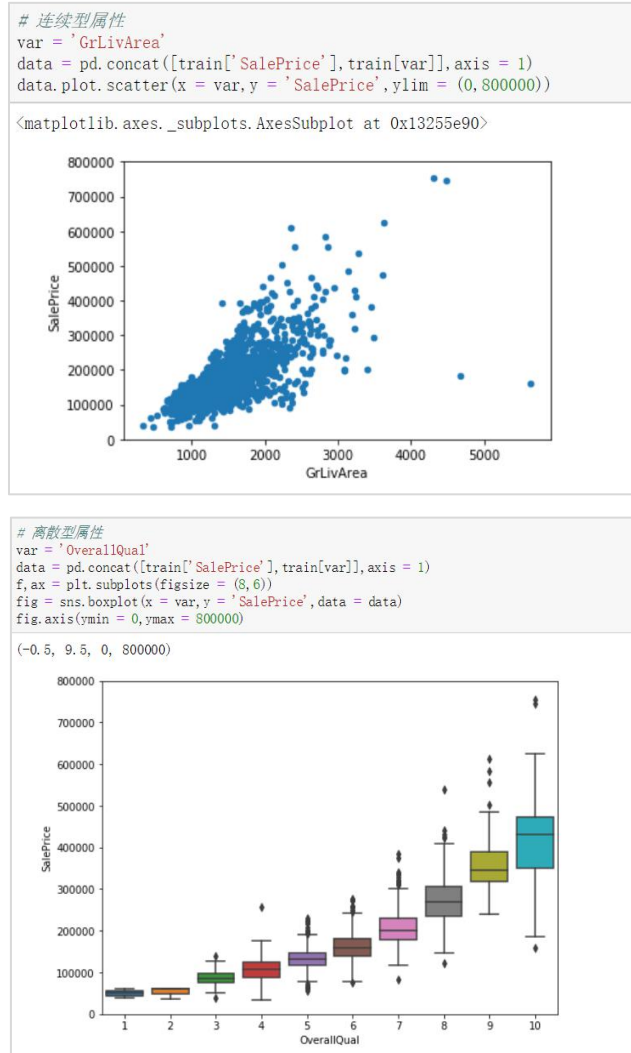
查看 **SalePrice** 的描述性统计量：

```
train['SalePrice'].describe()
count      1460.000000
mean       180921.195890
std        79442.502883
min        34900.000000
25%        129975.000000
50%        163000.000000
75%        214000.000000
max        755000.000000
Name: SalePrice, dtype: float64
```

画出 **SalePrice** 的分布图：



然后观察预测变量，本实验的数据分为连续变量及分类变量，我们先分别取一个连续变量以及分类变量：

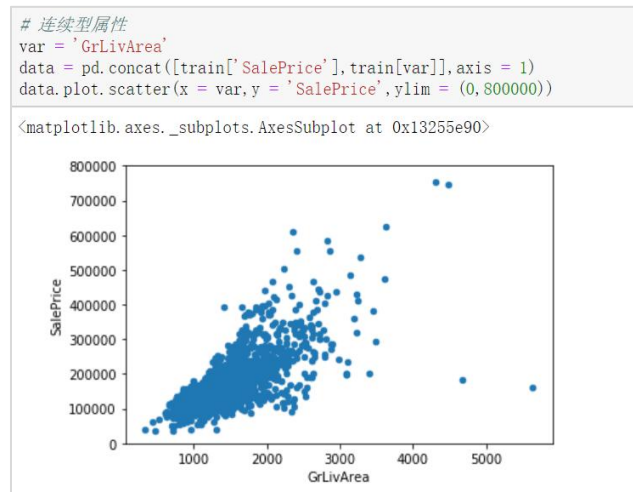


我们现在对响应变量和预测变量的分布有了直观上的大致认识，接下来我们将用各种方式处理这些原始数据。

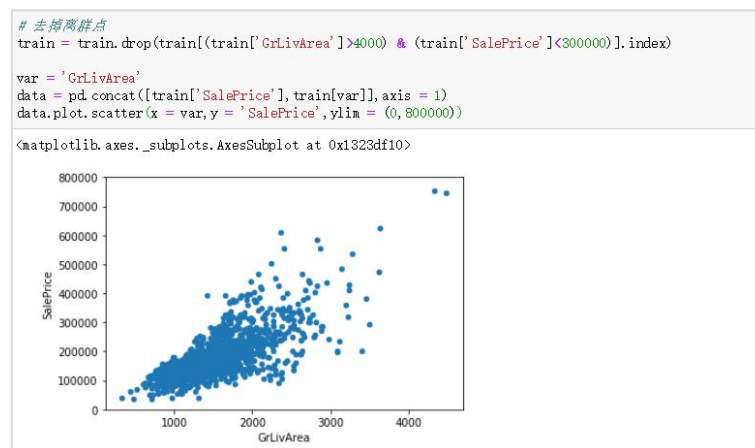
b) 离群点处理

先分析连续型变量的离群点：

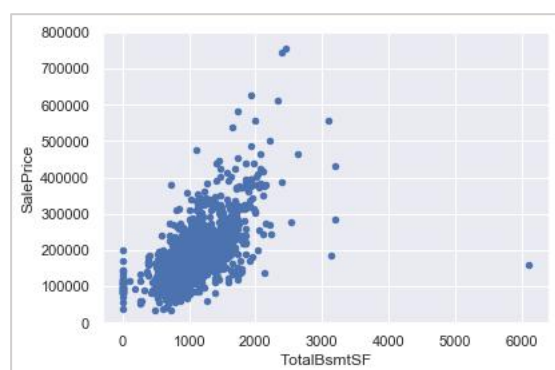
画出 GrLivArea 变量与房价关系的散点图：



由图可以看出，当 GrLivArea 大于 4000 然后 SalePrice 小于 300000 的两个点为离群点，因此需要 drop 那两个离群点数据：

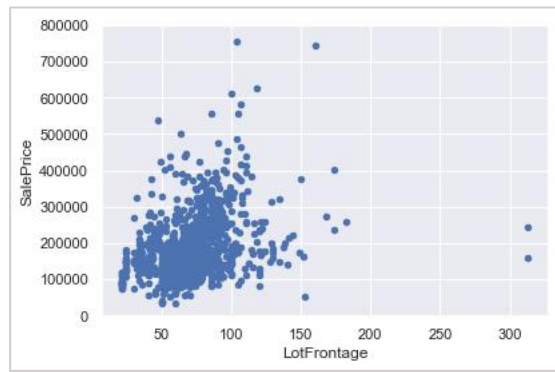


画出 TotalBsmtSF 变量与房价关系的散点图：



由图可以看出，当 TotalBsmtSF 大于 300 的点为离群点，因此同样需要 drop 那个离群点数据。

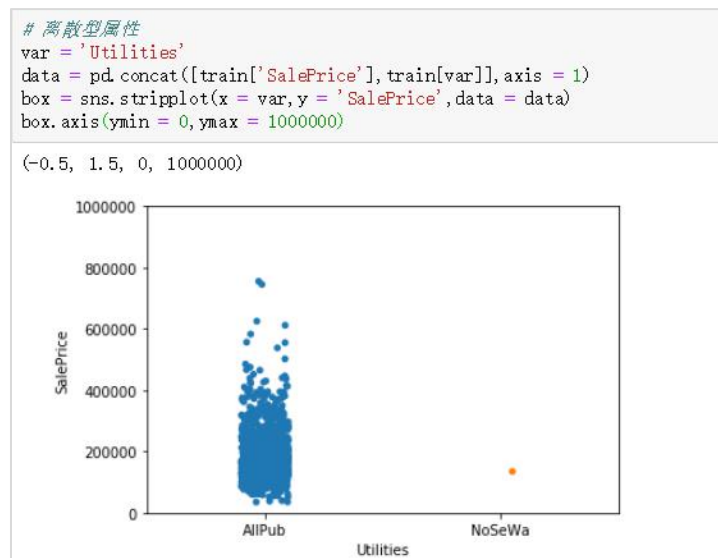
画出 LotFrontage 变量与房价关系的散点图：



由图可以看出，当 LotFrontage 大于 300 的两个点为离群点，因此同样需要 drop 那两个离群点数据。

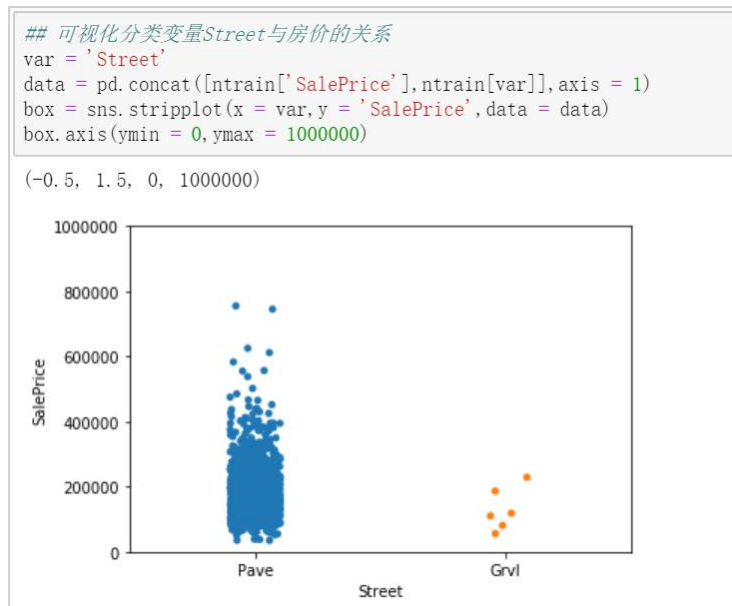
接着分析分类变量的离群点：

画出 Utilities 变量与房价关系的散点图：



可以看出，大部分的数据处在 AllPub 这个属性，该变量的属性变化对 SalePrice 的影响很小，因此该变量可以 drop 掉。

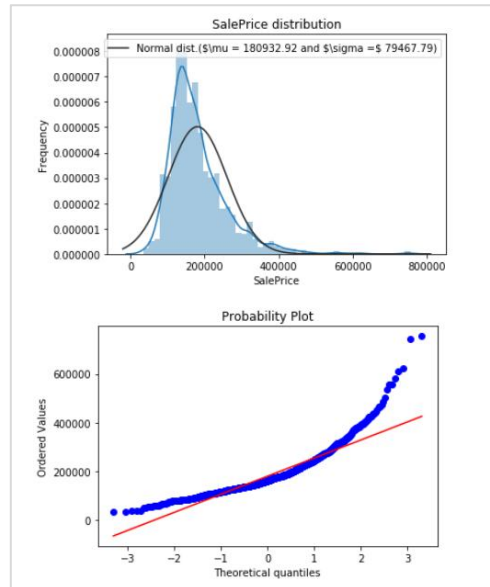
画出 Street 变量与房价关系的散点图：



可以看出，大部分的数据处在 Pave 这个属性，该变量的属性变化对 SalePrice 的影响很小，因此该变量可以 drop 掉。

c) 偏度校正

先观察因变量 SalePrice 的分布。由于在一般统计中，我们都会假设因变量 y 服从正态分布，因此当因变量 y 服从正态分布时，模型的拟合会比较好。偏度是统计数据分布偏斜方向和程度的度量，是统计数据分布非对称程度的数字特征，直观看来就是密度函数曲线尾部的相对长度。当数据样本的偏度小于 0 时，数据出现左侧长尾，当偏度大于 0 时，就会出现右侧长尾，若数据样本服从正态分布时，其偏度为 0。其定义为： $\text{Skew}(X) = E\left[\left(\frac{X-\mu}{\sigma}\right)^3\right]$ 。



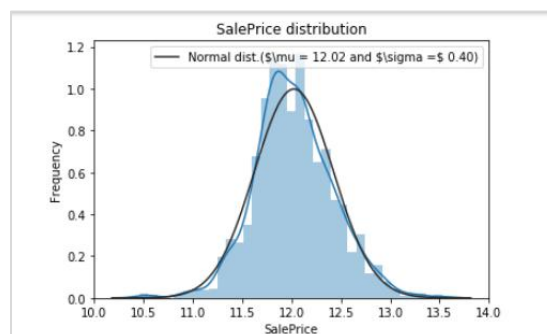
由图可得，可得变量 **SalePrice** 的偏度大于 0，为右侧长尾。因此我们通过一个递增函数 $\log(1+x)$ 对该变量进行变换，使其分布近似正态分布。

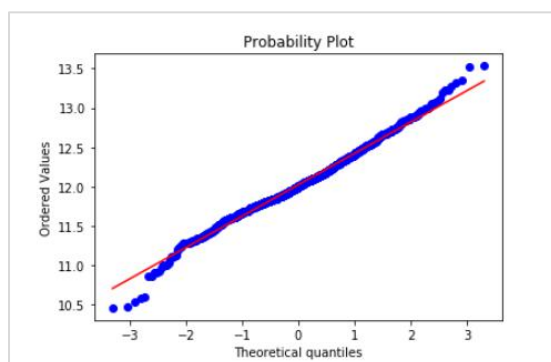
```
# 进行对数变换  $\log(1+x)$ 
train['SalePrice'] = np.log(train['SalePrice'] + 1)

# 进行变换后的分布图以及QQ图
sns.distplot(train['SalePrice'], fit = norm);

(mu, sigma) = norm.fit(train['SalePrice'])
plt.legend(['Normal dist. ( $\mu = {:.2f}$  and  $\sigma = {:.2f}$ )',
           loc = 'best'])
plt.ylabel('Frequency')
plt.title('SalePrice distribution')

# QQ图
fig = plt.figure()
res = stats.probplot(train['SalePrice'], plot = plt)
plt.show()
```





d) 缺失值处理

首先查看数据的缺失情况：

```
# missing data
total = all_data.isnull().sum().sort_values(ascending = False)
percent = (all_data.isnull().sum() / all_data.isnull().count()).sort_values(ascending = False)
missing_data = pd.concat([total, percent], axis = 1, keys = ['Total', 'Percent'])
missing_data.head(35)
```

	Total	Percent
PoolQC	2909	0.996574
MiscFeature	2814	0.964029
Alley	2721	0.932169
Fence	2348	0.804385
FireplaceQu	1420	0.486468
LotFrontage	486	0.166495
GarageQual	159	0.054471
GarageFinish	159	0.054471
GarageYrBlt	159	0.054471

（注：图片过长仅截取部分）

有 34 个变量有不同程度的缺失，接下来我们结合每个缺失的变量的具体意义分 4 种情况进行缺失值处理：

1. 删除的数据：

"PoolQC", "MiscFeature", "Alley", 这三个变量的数据有超过 85% 的缺失率，已经不能为预测提供有效的信息了，故删除之。

```
## 删除缺失值过多的数据
all_data = all_data.drop(["PoolQC", "MiscFeature", "Alley"], axis = 1)
```

2. 置 None 的数据:

“MiscFeature”, “Fence”, “FireplaceQu”, “GarageType”, “GarageFinish”, “GarageQual”, “GarageCond”, “BsmtQual”, “BsmtCond”, “BsmtExposure”, “BsmtFinType1”, “BsmtFinType2”, “MasVnrType”, “MSSubClass”, 这些数据都是为分类变量, 表示房屋的某种属性, 如围栏、壁炉、车库情况等, 缺失值很可能是因为这些房屋根本不拥有这些属性, 即没有围栏, 没有车库, 没有壁炉等, 因此采取置 None 添补缺失值的办法。

```
# 缺失值处理
# 特征
all_data['MiscFeature'] = all_data['MiscFeature'].fillna('None')
```

(此处仅以“MiscFeature”为例展示)

3. 置 0 的数据:

“GarageYrBlt”, “GarageArea”, “GarageCars”, “BsmtFinSF1”, “BsmtFinSF2”, “BsmtUnfSF”, “TotalBsmtSF”, “BsmtFullBath”, “BsmtHalfBath”, “MasVnrArea”这些数据都为连续型数据, 表示房屋某些属性的面积或个数等, 如砖石饰面面积、地下面积的总面积、车库容量大小等, 缺失值仍然可能是因为这些房屋根本不拥有这些属性, 因此采取置 0 添补缺失值的办法。

```
for col in ('GarageYrBlt', 'GarageArea', 'GarageCars'):
    all_data[col] = all_data[col].fillna(0)
```

(此处仅以“GarageYrBlt”, “GarageArea”, “GarageCars”为例展示)

4. 置该属性的众数:

“MSZoning”, “Electrical”, “KitchenQual”, “Exterior1st”, “Exterior2nd”, “SaleType”, 这些数据为连续型变量表示房屋某些属性的类型, 如电气系统、销售类型、房屋外部遮盖物材质等, 缺失值可能是用户并不清楚自己的房屋属于哪种类型, 因此采取更有代表性的众数来添补缺失值。

```
# 分区属性
all_data['MSZoning'] = all_data['MSZoning'].fillna(all_data['MSZoning'].mode()[0])
```

(此处仅以“MSZoning”为例展示)

5. 用特定值进行填补:

“LotFrontage”, “Functional”这些数据为非数字的类别型变量, 根据经验常识来选取

符合常理的分类来添补缺失值。“LotFrontage”表示街道的长度,选用“Neighborhood”的各分组的中位值来填补缺失值;“Functional”表示家庭功能评级,选用“Typ”(标准)来填补缺失值。

```
# 到街道的距离
all_data['LotFrontage'] = all_data.groupby('Neighborhood')['LotFrontage'].transform(lambda x: x.fillna(x.median()))

# 家庭功能评价
all_data['Functional'] = all_data['Functional'].fillna('Typ')
```

完成上述步骤后再次检查数据的缺失情况。

```
# 查看是否还有缺失数据
all_data_na = (all_data.isnull().sum() / len(all_data)) * 100
all_data_na = all_data_na.drop(all_data_na[all_data_na == 0].index).sort_values(ascending = False)
missing_data = pd.DataFrame({'Missing Ratio': all_data_na})
missing_data.head()
```

Missing Ratio

可以看到所有缺失值都被已经填补完毕!

e) 变量属性变换

观察数据可得, MSSubClass 变量的取值为 20, 30, 40 等, 虽然取值为数值型, 但是实际为分类型数据, 同理 OverallCond(表示评分), YrSoldn(表示年份), MoSold(表示月份)也是分类型数据, 因此我们需要将这些数据进行离散化处理。

```
# 看似为连续值的数值型数据离散化
all_data['MSSubClass'] = all_data['MSSubClass'].apply(str)

all_data['OverallCond'] = all_data['OverallCond'].astype(str)

all_data['YrSold'] = all_data['YrSold'].astype(str)
all_data['MoSold'] = all_data['MoSold'].astype(str)
```

然后将分类数据用 1, 2, 3, 4, ...进行标记, 便于放进模型进行拟合:

```
# 将分类变量分成1, 2, 3...等数值
cols = ['FireplaceQu', 'BsmtQual', 'BsmtCond', 'GarageQual',
        'GarageCond', 'ExterQual', 'ExterCond', 'HeatingQC',
        'KitchenQual', 'BsmtFinType1', 'BsmtFinType2',
        'Functional', 'Fence', 'BsmtExposure', 'GarageFinish',
        'LandSlope', 'LotShape', 'PavedDrive', 'CentralAir',
        'MSSubClass', 'OverallCond', 'YrSold', 'MoSold']

for c in cols:
    lbl = LabelEncoder()
    lbl.fit(list(all_data[c].values))
    all_data[c] = lbl.transform(all_data[c].values)

all_data.shape
```

三、特征工程

首先可以依据常理融合几个特征构成可能具有一定意义的新的特征。

将建造年份和修缮年份相加得新变量'YrBltAndRemod'，值越大说明房屋越新，可能房价会越高。

将地下面积、一层面积、二层面积相加得新变量'TotalSF'，值越大说明房屋总面积越大，可能房价会越高。

将地下竣工面积、一层面积、二层面积相加得新变量'TotalSF'，值越大说明房屋总可用面积越大，可能房价会越高。

将地上完整卫生间、0.5×地上半卫生间、地下完整卫生间、0.5×地下半卫生间相加得新变量'Total_Bathrooms'，值越大说明房屋总卫生间面积越大，可能房价会越高。

将开放式门廊面积、三季门廊面积、封闭门廊面积、屏风门廊面积、木甲板面积相加得新变量'Total_porch_sf'，值越大说明房屋总门廊面积越大，可能房价会越高。

具体操作过程如下：

```
# 增加新特征
all_data['TotalSF'] = all_data['TotalBsmtSF'] + all_data['1stFlrSF'] + all_data['2ndFlrSF']
all_data['YrBltAndRemod'] = all_data['YearBuilt'] + all_data['YearRemodAdd']
all_data['Total_sqr_footage'] = (all_data['BsmtFinSF1'] + all_data['BsmtFinSF2'] +
                                all_data['1stFlrSF'] + all_data['2ndFlrSF'])

all_data['Total_Bathrooms'] = (all_data['FullBath'] + (0.5 * all_data['HalfBath']) +
                                all_data['BsmtFullBath'] + (0.5 * all_data['BsmtHalfBath']))

all_data['Total_porch_sf'] = (all_data['OpenPorchSF'] + all_data['3SsnPorch'] +
                                all_data['EnclosedPorch'] + all_data['ScreenPorch'] +
                                all_data['WoodDeckSF'])
```

然后我们再统计所有变量的偏度：

```
# 特征的偏度
numeric_feats = all_data.dtypes[all_data.dtypes != 'object'].index

skewed_feats = all_data[numeric_feats].apply(lambda x: skew(x.dropna())).sort_values(ascending = False)
skewness = pd.DataFrame({'skew': skewed_feats})
skewness.head()
```

	skew
MiscVal	21.947195
PoolArea	16.898328
LotArea	12.822431
LowQualFinSF	12.088761
3SsnPorch	11.376065

使用 Box-Cox 变换对高偏度数据进行处理，使其近似服从正态分布。

```
skewness = skewness[abs(skewness) > 0.75]
print('{} transform'.format(skewness.shape[0]))

skewed_features = skewness.index
lam = 0.15
for feat in skewed_features:
    all_data[feat] = boxcoxlp(all_data[feat], lam)
```

63 transform

综上，我们创建了 5 个新变量并对其施行了合理的变换。

四、构建模型

分别构造 lasso 模型、elastic net 弹性网络模型、Kernel Ridge 模型、Gradient boosting 模型、xgboost 模型：

```
# 定义lasso模型（使用一范数作为正则化项）
lasso = make_pipeline(RobustScaler(), Lasso(alpha = 0.0005, random_state = 1))

# 定义elastic net弹性网络模型（弹性网络实际上是结合了岭回归和lasso的特点，同时使用了L1和L2作为正则化项）
ENe = make_pipeline(RobustScaler(), ElasticNet(alpha = 0.0005, l1_ratio = .9, random_state = 3))

# 带核函数的岭回归
KRR = KernelRidge(alpha = 0.6, kernel = 'polynomial', degree = 2, coef0 = 2.5)

# Gradient boosting
GBoost = GradientBoostingRegressor(n_estimators = 3000, learning_rate = 0.05,
                                    max_depth = 4, max_features = 'sqrt',
                                    min_samples_leaf = 15, min_samples_split = 10,
                                    loss = 'huber', random_state = 5)

# xgboost
xgboost = xgb.XGBRegressor(colsample_bytree = 0.4603, gamma = 0.0468,
                           learning_rate = 0.05, max_depth = 3,
                           min_child_weight = 1.7817, n_estimators = 2200,
                           reg_alpha = 0.4640, reg_lambda = 0.8571,
                           subsample = 0.5213, silent = 1,
                           nthread = -1)
```

使用 RMSE 作为评价指标，定义 K 折交叉验证的模型评分函数：


```
## 机器学习
# 设置k折交叉验证的参数
n_folds = 5
# 创建模型评分函数，根据不同模型的表现打分
# rmsle表示均方根对数误差 (Root Mean Squared Logarithmic Error)
# cv表示交叉验证 (Cross-validation)
def rmsle_cv(model):
    kf = KFold(n_folds, shuffle = True, random_state = 42).get_n_splits(train.values)
    rmse = np.sqrt(-cross_val_score(model, train.values, y_train, scoring = 'neg_mean_squared_error', cv = kf))
    return rmse
```

使用训练集数据，利用模型评分函数查看使用以上各单个模型的训练效果：

```
# 打印一范数LASSO模型的得分
score = rmsle_cv(lasso)
print('\nLasso Score: {:.4f} ({:.4f})\n'.format(score.mean(), score.std()))
```

Lasso Score:0.1115(0.0074)

```
# 打印elastic net弹性网络模型的得分
score = rmsle_cv(ENe)
print('\nElasticNet Score: {:.4f} ({:.4f})\n'.format(score.mean(), score.std()))
```

ElasticNet Score:0.1115(0.0074)

```
# 打印Kernel Ridge回归模型的得分
score = rmsle_cv(KRR)
print('\nKernel Ridge Score: {:.4f} ({:.4f})\n'.format(score.mean(), score.std()))
```

Kernel Ridge Score:0.1150(0.0079)

```
# 打印gbr梯度提升回归模型的得分
score = rmsle_cv(GBoost)
print('\nGradient Boosting Score: {:.4f} ({:.4f})\n'.format(score.mean(), score.std()))
```

Gradient Boosting Score:0.1154(0.0074)

```
# 打印xgboost模型的得分
score = rmsle_cv(xgboost)
print('\nXgBoost Score: {:.4f} ({:.4f})\n'.format(score.mean(), score.std()))
```

XgBoost Score:0.1154(0.0063)

可以看到，这 5 个模型的得分均在 0.11-0.12 的范围内，标准差均在 0.006-0.008 的范围内，总体上 5 个模型的拟合效果都较好。接下来的问题就是，应该把哪一个模型作为最终的选择。如果单独看每一个模型，毫无疑问是 lasso 或 elastic net 弹性网络的效果最好（得分最低），但如果只把 lasso 或 elastic net 弹性网络的结果作为最终结果，就造成了另外 4 个模型的结果的浪费，这并不是我们所期望的——我们期望的是以结果为导向，最大限度地利用已有资源。于是就有了集成学习的概念。

集成学习，顾名思义，通过将多个单个学习器集成/组合在一起，使它们共同完成学习任务，有时也被称为“多分类器系统(multi-classifier system)”、“基于委员会的学习

(Committee-based learning)”，多个学习器可以互相帮助，各取所长，就有可能一起合作把一个学习任务完成得比较漂亮。

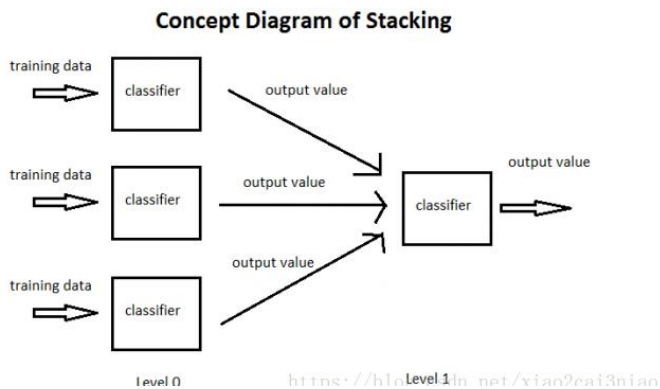
将上述各模型进行集成学习，首先我们先进行平均(Average)模型。第一步是对 5 个模型做出来的结果取平均，从而得到我们的预测值，此时，平均模型的误差值为 0.1080。第二步，为了防止这 5 个模型中的某一个对其他 4 个模型有潜在的干扰，导致误差值被提高，我们依次将 5 个模型中的任意 4 个进行平均，得到 5 个新的平均模型，并查看误差值。事实证明我们是多虑了，新的 5 个平均模型的误差值的最小值为 0.1082。因此，我们还是选择对 5 个模型进行平均。

```
# 对五个模型做集成
averaged_models_five = averagemodels(models = (lasso, ENe, KRR, xgboost, GBoost))

score = rmsle_cv(averaged_models_five)
print('\nAverage base models Score: {:.4f} ({:.4f})\n'.format(score.mean(), score.std()))
```

Average base models Score:0.1080(0.0072)

其次，我们再使用堆积模型(stacking)的方法进行建模。我们分两个阶段，第一阶段为 base-models，我们用了 3 个模型，分别是 elastic net 弹性网络模型，Gradient boosting 模型，Kernel Ridge 模型，在第一阶段取出来的值作为第二阶段的 3 个特征，即经过了第一阶段，每一个样本都有 3 个值输出，该 3 个值为交叉验证的验证集。然后我们再将这 3 个值作为特征，传到第二阶段，如图所示：



然后得到最终的结果：误差值为 0.1080，虽然略逊于平均模型，但也是很好的结果了。

```
stacked_models = stackingmodels(base_models = (ENe, GBoost, KRR),
                                meta_model = lasso)

score = rmsle_cv(stacked_models)
print('\nStacking models Score: {:.4f} ({:.4f})\n'.format(score.mean(), score.std()))
```

Stacking models Score:0.1080(0.0076)

最后，我们将 5 个模型的平均模型与堆积模型再做一次平均，得到的误差值为 0.1078。

至此，模型训练完毕。

```
# 对平均模型以及堆积模型再做平均
averaged_models_new = averagemodels(models = (averaged_models_five, stacked_models))

score = rmsle_cv(averaged_models_new)
print('\nStack&Average models Score:{:.4f} ({:.4f})\n'.format(score.mean(), score.std()))

Stack&Average models Score:0.1078(0.0074)
```

五、 进行预测

将训练好的模型带入测试集数据中求出预测房价，并导出结果按格式为第一列'id'、第二列'SalePrice'保存成 csv 文件，上传到 kaggle 网站检验。

```
# 进行预测
averaged_models_new.fit(train.values, y_train)
y_pre_averaged_new = averaged_models_new.predict(test.values)
y_pre_averaged_new = np.exp(y_pre_averaged_new)-1
y_pre_averaged_new
```

```
# 输出数据
prediction_averaged_new = pd.DataFrame(y_pre_averaged_new, columns=['SalePrice'])
result_averaged_new = pd.concat([test_ID, prediction_averaged_new], axis=1)
result_averaged_new.columns
```


```
# 保存预测结果
result_averaged_new.to_csv('./Predictions.csv', index=False)
```

六、 结果评价

Your most recent submission				
Name	Submitted	Wait time	Execution time	Score
Predictions(1).csv	just now	0 seconds	0 seconds	0.11542
Complete				
Jump to your position on the leaderboard				

471


ElsieChen



0.11542


7

1m

Your Best Entry 

You advanced 13 places on the leaderboard!

Your submission scored 0.11542, which is an improvement of your previous score of 0.11546. Great job!

 Tweet this!

在 kaggle 网站上，我们的分数是 0.11542，暂时的排名是 471，预测的结果还不错，但是还有需要解决的问题，如直接将分类变量置为 1，2，3...是否会影响拟合的效果？因为不同类别对 SalePrice 的影响不一定是成正比的关系；以及如何生成更为有效，能提高拟合效果的新特征？

另外，本文对缺失值的填充方法之一是简单插补，即有中位数或众数来替换变量中的缺失值，这种方法虽然简单，但是对于非 MCAR（非完全随机删失）的数据会产生有偏的结果，很可能会低估标准差、曲解变量间的相关性。如能解决上述问题，相信能更好地提高模型的效果。

七、 附录

79 个变量的具体信息和中文解释如下：

MSSubClass: 表示要出售的住宅的类型

20	平房（1 层）	1946	全部翻新过
30	平房（1 层）	1945	老房
40	平房（1 层）	适合所有年龄段的阁楼（即将建成\竣工）	
45	1 层半房	没建完的	适合各个年龄段
50	1 层半房	竣工	适合各个年龄段
60	2 层房	1946	翻新过
70	2 层房	1945	老房
75	2 层半房	适合各个年龄段	
80	复式住宅或者多层住宅		

85	分离式门厅
90	连栋式的两栋住宅 适合各种风格以及各种年龄段
120	1 层住宅（计划单元开发）1946 新的
150	1 层半的住宅（计划单元开发） 适合各个年龄
160	2 层住宅（计划单元开发）1946 新的
180	多层住宅（计划单元开发） 包括分离式梯/门厅
190	双户转换，适合各个年龄段

MSZoning: 表示销售的一般分区分类

A	农业
C	商业
FV	浮村住宅
I	工业
RH	入住密度很大的住宅
RL	入住密度很低的住宅
RP	入住密度很低的住宅庭院
RM	入住密度中等的住宅

LotFrontage: 与房屋相连的街道的延长英尺，就是街道的长度

LotArea: 房屋占地面积（平方英尺）

Street: 连接房屋的道路类型

Grvl	砾石
Pave	铺平的

Alley: 连接房屋的胡同道路类型

Grvl	砾石
Pave	铺平的
NA	没有胡同

LotShape: 房屋的形状

Reg	规则的
IR1	轻微不规则
IR2	中度不规则
IR3	不规则

LandContour: 房屋的平坦度

Lvl	几乎平坦
Bnk	倾斜 - 从街道级快速显著上升到房屋
HLS	山坡 - 从一侧到另一侧明显的坡度
Low	凹陷

Utilities: 可用的设施类型

AllPub	所有设施
NoSewr	电、气、水（化粪池）
NoSeWa	只有电和气
EL0	只有电

LotConfig: 房屋布局

Inside	内部地段
Corner	街角地段
CulDSac	死胡同

FR2	房屋两面临街
FR3	房屋三面临街

LandSlope: 房屋的倾斜度

Gt1	轻微倾斜
Mod	中度倾斜
Sev	严重倾斜

Neighborhood: 埃姆斯市范围内的实际位置

Blmngtn	布卢明顿高地
Blueste	蓝斯特姆
BrDale	布里亚代尔
BrkSide	布鲁克赛德
ClearCr	克里克河
CollgCr	克里克大学
Crawfor	克劳福德
Edwards	爱德华兹
Gilbert	吉尔伯特
IDOTRR	爱荷华州交通和铁路
MeadowV	草甸村
Mitchel	米契尔
Names	埃姆斯北部
NoRidge	北岭
NPkVill	北园别墅
NridgHt	北岭高地
NWAmes	埃姆斯的西北部
OldTown	老城区
SWISU	爱荷华州立大学西南部

Sawyer	索耶
SawyerW	索耶西部
Somerst	萨默塞特
StoneBr	石溪
Timber	林地
Veenker	韦恩克

Condition1: 附近的多种条件

Artery	临近主干道
Feedr	临近支线街道
Norm	附近条件一般
RRNn	在南北铁路 200 英尺以内
RRAn	邻近南北铁路
PosN	附近有积极的 有益的室外景点—公园，绿地等.
PosA	邻近有益的室外景点
RRNe	在东西铁路 200 英尺以内
RR Ae	邻近东西铁路

Condition2: 附近的多种条件（如果有不至一种条件）

Artery	临近主干道
Feedr	临近支线街道
Norm	附近条件一般
RRNn	在南北铁路 200 英尺以内
RRAn	邻近南北铁路
PosN	附近有积极的 有益的室外景点—公园，绿地等.
PosA	邻近有益的室外景点
RRNe	在东西铁路 200 英尺以内
RR Ae	邻近东西铁路

BldgType: 住宅类型

1Fam	单独的独户
2FmCon	双户转换，但建成一户住宅的样子
Duplx	双层公寓
TwnhsE	联排别墅末端房屋
TwnhsI	联排别墅里面房屋

HouseStyle: 住宅样式

1Story	平房
1.5Fin	一层半房，第二层完成
1.5Unf	一层半房，第二层未完成
2Story	2层
2.5Fin	2层半房，第二层完成
2.5Unf	2层半房，第二层未完成
SFoyer	分门厅
SLvl	错层式

OverallQual: 对房子的整体材料和装修进行评级

10	非常豪华
9	豪华
8	非常好
7	好
6	中等以上
5	中等
4	中等以下
3	一般

2	差
1	很差

OverallCond: 评估房子的整体状况

10	非常豪华
9	豪华
8	非常好
7	好
6	中等以上
5	中等
4	中等以下
3	一般
2	差
1	很差

YearBuilt: 原施工日期

YearRemodAdd: 改造日期（如无改造或增加，与施工日期相同）

RoofStyle: 屋顶种类

Flat	平坦的
Gable	三角式屋顶
Gambrel	谷仓型屋顶
Hip	斜脊
Mansard	折线形屋顶
Shed	棚式

RoofMatl: 屋顶材料

ClyTile	粘土或瓦
CompShg	标准（复合）瓦
Membran	膜结构
Metal	金属
Roll	轧辊材料
Tar&Grv	砾石 + 焦油
WdShake	木制瓦
WdShngl	木制复合瓦

Exterior1st: 房屋外部遮盖物

AsbShng	石棉瓦
AsphShn	沥青瓦
BrkComm	普通砖
BrkFace	饰面砖
CBlock	煤渣砖
CemntBd	水泥板
HdBoard	硬质纤维板
ImStucc	仿灰泥
MetalSd	金属壁板
Other	其他
Plywood	胶合板
PreCast	预制板
Stone	石板
Stucco	灰泥
VinylSd	乙烯基壁板
Wd Sdng	木壁板
WdShing	木制扣板

Exterior2nd: 房屋外部遮盖物(如果不止一种材料)

AsbShng	石棉瓦
AsphShn	沥青瓦
BrkComm	普通砖
BrkFace	饰面砖
CBlock	煤渣块
CemntBd	水泥板
HdBoard	硬质纤维板
ImStucc	仿灰泥
MetalSd	金属壁板
Other	其他
Plywood	胶合板
PreCast	预制板
Stone	石板
Stucco	灰泥
VinylSd	乙烯基壁板
Wd Sdng	木壁板
WdShing	木制扣板

MasVnrType: 砌体贴面类型

BrkCmn	普通砖
BrkFace	饰面砖
CBlock	煤渣砖
None	没有
Stone	石板

MasVnrArea: 砖石饰面面积(平方英尺)

ExterQual: 评估外部材料的质量

Ex	非常好
Gd	好
TA	中等/标准
Fa	一般
Po	差

ExterCond: 评估外部材料的现状

Ex	非常好
Gd	好
TA	中等/标准
Fa	一般
Po	差

Foundation: 基地材料

BrkTil	砖和瓦
CBlock	煤渣砖
PConc	浇混凝土
Slab	(石、木等坚硬物质的) 厚板
Stone	石头
Wood	木头

BsmtQual: 评估地下室高度

Ex	非常高 (100+英寸)
Gd	高 (90-99 英寸)
TA	中等/标准 (80-89 英寸)

Fa	一般（70-79 英寸）
Po	低（<70 英寸）
NA	无地下室

BsmtCond: 评估地下室总体情况

Ex	豪华
Gd	好
TA	中等/标准 - 有轻微潮湿
Fa	一般 - 潮湿，有裂缝或者下沉
Po	差 - 有严重裂缝，下沉或者潮湿
NA	无地下室

BsmtExposure: 指户外或花园水平的墙壁

Gd	良好的暴露
Av	中等的暴露（通常评级中等或以上的分离式楼层或门厅）
Mn	最小暴露
No	无暴露
NA	无地下室

BsmtFinType1: 地下室竣工面积的评级

GLQ	良好的生活区
ALQ	标准的生活区
BLQ	低于标准的生活区
Rec	标准的娱乐室
LwQ	低质量
Unf	未完工
NA	无地下室

BsmtFinSF1: 类型 1 竣工平方英尺

BsmtFinType2: 地下室竣工面积的评级(如果有多种类型)

GLQ	良好的生活区
ALQ	标准的生活区
BLQ	低于标准的生活区
Rec	标准的娱乐室
LwQ	低质量
Unf	未完工
NA	无地下室

BsmtFinSF2: 类型 2 竣工平方英尺

BsmtUnfSF: 未完工的地下室面积

TotalBsmtSF: 地下面积的总面积

Heating: 暖气设备类型

Floor	地板炉
GasA	煤气强制热空气炉
GasW	煤气热水或蒸汽热
Grav	重力炉
OthW	燃气以外的热水或蒸汽热
Wall	壁炉

HeatingQC: 暖气设备质量及状况

Ex	豪华
Gd	好
TA	中等/标准
Fa	一般
Po	差

CentralAir: 中央空调

N	有
Y	无

Electrical: 电气系统

SBrkr	标准断路器和 Romex 电缆
FuseA	保险丝盒超过 60 安培和所有 Romex 布线（标准）
FuseF	60 安培保险丝盒和大部分 Romex 布线（一般）
FuseP	60 安培保险丝盒和大部分旋钮 + 管接线（差）
Mix	混合

1stFlrSF: 一楼平方英尺

2ndFlrSF: 二楼平方英尺

LowQualFinSF: 低质量成品平方英尺（所有楼层）

GrLivArea: 地面以上居住面积（平方英尺）

BsmtFullBath: 地下室设施齐全的浴室

BsmtHalfBath: 地下室仅具便池、面盆设备的卫生间

FullBath: 地面以上的设施齐全的卫生间

HalfBath: 地面以上仅具便池、面盆设备的卫生间

Bedroom: 地面以上卧室（不包括地下室卧室）

Kitchen: 地面以上厨房

KitchenQual: 厨房质量

Ex	豪华
Gd	好
TA	中等/标准
Fa	一般
Po	差

TotRmsAbvGrd: 地面以上房间总数（不含卫生间）

Functional: 家庭功能（假设是标准的，除非有必要进行扣减）

Typ	标准功能
Min1	较小扣减 1
Min2	较小扣减 2
Mod	适中扣减
Maj1	较大扣减 1
Maj2	较大扣减 2
Sev	严重损坏
Sal	仅供救助

Fireplaces: 壁炉数量

FireplaceQu: 壁炉质量

Ex	豪华 - 豪华砌体壁炉
Gd	好 - 主层砌石壁炉
TA	标准 - 主生活区预制壁炉或地下室的砖砌壁炉
Fa	一般 - 地下室预制壁炉
Po	差 - 本·富兰克林炉
NA	无壁炉

GarageType: 车库位置

2Types	多个类型的车库
Attchd	附属于家
Basment	地下室车库
BuiltIn	内置（房子的车库部分 - 通常有车库上方的房间）
CarPort	简易车库
Detchd	与家分离
NA	无车库

GarageYrBltn: 建造车库的年份

GarageFinish: 车库的内部装修

Fin	完成
RFin	粗略完成
Unf	未完成
NA	无车库

GarageCars: 车库容量大小

GarageArea: 车库面积 (平方英尺)

GarageQual: 车库质量

Ex	豪华
Gd	好
TA	中等/标准
Fa	一般
Po	差
NA	无车库

GarageCond: 车库状况

Ex	豪华
Gd	好
TA	中等/标准
Fa	一般
Po	差
NA	无车库

PavedDrive: 铺面车道

Y	全铺面
P	部分铺装
N	泥土/砂砾

WoodDeckSF: 木甲板面积 (平方英尺)

OpenPorchSF: 开放式门廊面积 (平方英尺)

EnclosedPorch: 封闭门廊面积 (平方英尺)

3SsnPorch: 三季门廊面积 (平方英尺)

ScreenPorch: 屏风门廊面积 (平方英尺)

PoolArea: 游泳池面积 (平方英尺)

PoolQC: 游泳池质量

Ex 豪华

Gd 好

TA 中等/标准

Fa 一般

NA 无游泳池

Fence: 栅栏质量

GdPrv 良好的隐私

MnPrv 最低的隐私

GdWo 好木材

MnWw 最小木材/线材

NA 无栅栏

MiscFeature: 其他类别未涵盖的杂项特征

Elev 有电梯

Gar2	有第二车库 (如果没有在车库部分描述)
Othr	其他
Shed	有棚 (超过 100 SF)
TenC	有网球场
NA	没有

MiscVal: 杂项特征的值

MoSold: 销售的月份 (MM)

YrSold: 销售的年份 (YYYY)

SaleType: 销售类型

WD	保修契据 - 常规
CWD	保修契据 - 现金
VWD	保修契据 - VA 贷款
New	房屋刚刚建成和销售
COD	法庭官员契据/遗产
Con	合同 15%定金定期支付
ConLw	合同低首付和低利息
ConLI	合同低利息
ConLD	合同低首付和低利息
Oth	其他

SaleCondition: 销售条件

Normal	正常销售
Abnorml	异常销售 - 交易, 止损, 卖空
AdjLand	毗邻土地购买

Alloca 分配 - 两个具有独立契约的相互关联的财产，典型的带有车位的公寓

Family 家庭成员之间的销售

Partial 上次评估时未完成房屋（与新住宅相关）