基于网络搜索行为
对洪山区商品房价格的短期预测

实验分析过程及结果说明

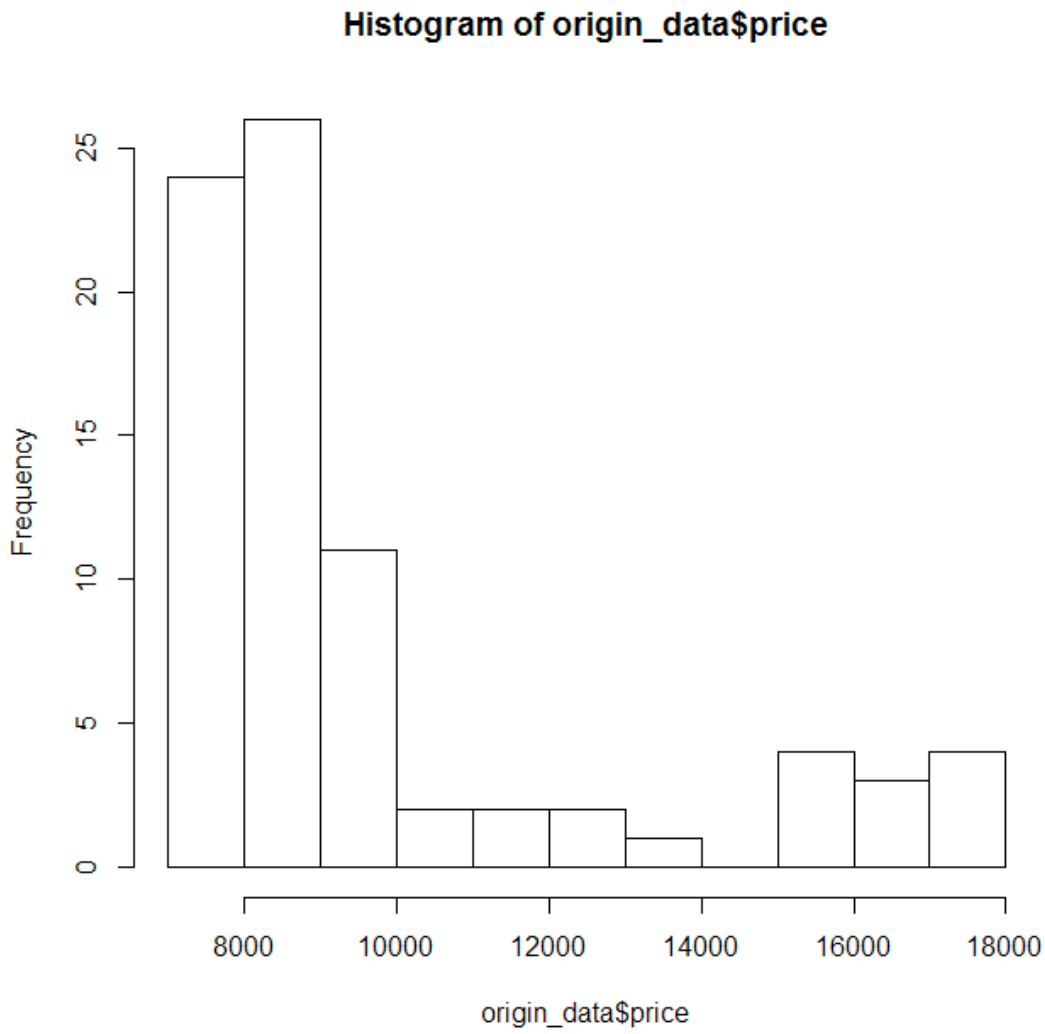
蒲东齐

```
#---------------------------------------------------------------
# 切换工作目录
#---------------------------------------------------------------
setwd("……")
```
填写地址，切换到一个合适的工作目录

```
#---------------------------------------------------------------
# 导入相关软件包
#---------------------------------------------------------------
```
主要是构建模型需要的。
```
library(MASS)
library(randomForest)
library(rpart)
library(plyr)
library(e1071)
library(glmnet)
library(ipred)
library(maptree)

#---------------------------------------------------------------
# 构造评价函数
#---------------------------------------------------------------
```
构造了 MSE 和 NMSE 两个评价误差的函数，在之后模型完成之后可以直接调用并保存数据。
```
MSE <- function(origin, predict){
    n <- length(origin)
    mse <- sum(origin - predict) ^2/n
    return(mse)
}

NMSE <- function(origin, predict){
    return(MSE(origin, predict)/var(origin))
}

#---------------------------------------------------------------
# 读入整理好的原始数据,并划分训练集测试集
#---------------------------------------------------------------
```
上面已经切换过工作目录到你选择的地址，现在要把整理好的原始数据 origin..csv 放到你选择的地址里。
```
origin_data <- read.csv('origin_utf8.csv', header = T)
```
（已经手动的清理了一些变量，如把数据量为零的变量删除）
read.csv 表示以 csv 格式读入数据文件 origin..csv，header = T 表示第一行数据为变量名称。
```
#按照训练集占 75%，测试集占 25%的原则划分
train_sub <- sample(nrow(origin_data), nrow(origin_data)*0.75)
```

train_data <- origin_data[train_sub,]
test_data <- origin_data[-train_sub,]
按照行号随机抽取 75%的行号作为训练集，25%的行号作为测试集。

#--
Linear Models
#--
【模型简介】
在回归分析中，如果有两个或两个以上的自变量，就称为多元回归。一元一次的回归曲线是用最小二乘法确定各个变量的系数，推广到多元其实也是一样的，最小二乘法（又称最小平方法）是一种数学优化技术。它通过最小化误差的平方和寻找数据的最佳函数匹配。利用最小二乘法可以简便地求得未知的数据，并使得这些求得的数据与实际数据之间误差的平方和为最小。简单说就是使得方程真实值和预测值的残差平方和最小。
对于一元线性回归模型, 假设从总体中获取了 n 组观察值（X1，Y1），（X2，Y2），…，（Xn，Yn）。对于平面中的这 n 个点，可以使用无数条曲线来拟合。要求样本回归函数尽可能好地拟合这组值。综合起来看，这条线处于样本数据的中心位置最合理。 选择最佳拟合曲线的标准可以确定为：使总的拟合误差（即总残差）达到最小。
hist(origin_data$price)
liu



shapiro.test(origin_data$price)
对房产真实价格数据进行正态检验，结果如下：
Shapiro-Wilk normality test

data: origin_data$price
W = 0.7098, p-value = 3.335e-11
因为 P 值小于 0.05，故拒绝愿假设，房产真实价格数据不是正态分布。

model_lm <- lm(price~., data = train_data[,-1])
建立简单的多元线性回归模型，模型名称 model_lm，lm 表示采用线性回归模型，price~.,表示 price 作为因变量，其余 K1 至 K23 作为自变量，data = train_data[,-1]表示数据集为训练集，并且去掉第一列月份数据。
summary(model_lm)
输出模型 model_lm 的摘要信息，如下：
Call:
lm(formula = price ~ ., data = train_data[, -1])

Residuals:

Min	1Q	Median	3Q	Max
-922.20	-256.36	-36.95	288.38	1622.40

Coefficients:

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	7.541e+03	9.311e+02	8.099	1.55e-09	***
K1	-1.566e-01	1.288e-01	-1.216	0.23222	
K2	2.271e-01	4.381e-01	0.518	0.60750	
K3	-4.561e-01	2.878e-01	-1.585	0.12198	
K4	-4.960e-02	4.382e-01	-0.113	0.91051	
K5	2.117e-01	2.764e-01	0.766	0.44878	
K6	7.875e-01	5.048e-01	1.560	0.12772	
K7	4.432e-01	2.427e-01	1.826	0.07640	.
K8	4.167e-01	5.275e-01	0.790	0.43481	
K9	-1.238e-01	5.207e-01	-0.238	0.81346	
K10	-1.421e+00	8.045e-01	-1.766	0.08614	.
K11	2.391e-02	5.420e-01	0.044	0.96507	
K12	-2.947e-01	6.096e-01	-0.483	0.63182	
K13	3.275e-03	9.053e-02	0.036	0.97134	
K14	-5.745e-02	6.360e-02	-0.903	0.37252	
K15	-5.295e-01	2.376e-01	-2.229	0.03235	*
K16	6.357e-01	2.952e-01	2.154	0.03822	*
K17	-3.984e-02	9.075e-02	-0.439	0.66340	
K18	4.433e-03	1.020e-01	0.043	0.96560	
K19	1.780e-01	7.426e-02	2.397	0.02200	*
K20	-3.549e-01	5.344e-01	-0.664	0.51101	
K21	6.167e-03	7.420e-02	0.083	0.93424	
K22	6.140e-01	2.170e-01	2.830	0.00767	**
K23	-1.398e-01	3.209e-01	-0.436	0.66581	

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 576.9 on 35 degrees of freedom

Multiple R-squared: 0.9807, Adjusted R-squared: 0.968

F-statistic: 77.3 on 23 and 35 DF, p-value: < 2.2e-16

由上述输出可知，模型系数显著性状况不佳，许多系数的 P 值大于 0.05，即不够显著，但从 R 方来看为 0.9807，很高，调整后 R 方为 0.968，回归方程的显著性检验 F 统计量为 2.2e-16，小于 0.05，说明回归方程整体显著。

model_lm_step <- stepAIC(model_lm, direction = 'backward')

对上述简单多元线性回归模型进行逐步回归，模型名称 model_lm_step，stepAIC 表示根据 AIC 信息准则进行逐步回归，model_lm 表示对 model_lm 模型进行逐步回归，direction = 'backward'表示采用向后回归方法。其逐步回归过程如下：

Start: AIC=767.39

price ~ K1 + K2 + K3 + K4 + K5 + K6 + K7 + K8 + K9 + K10 + K11 +
K12 + K13 + K14 + K15 + K16 + K17 + K18 + K19 + K20 + K21 +
K22 + K23

	Df	Sum of Sq	RSS	AIC
- K13	1	436	11647962	765.39
- K18	1	628	11648154	765.39
- K11	1	648	11648174	765.39
- K21	1	2299	11649825	765.40
- K4	1	4265	11651791	765.41
- K9	1	18812	11666338	765.49
- K23	1	63140	11710666	765.71
- K17	1	64121	11711647	765.71
- K12	1	77766	11725293	765.78
- K2	1	89403	11736930	765.84
- K20	1	146748	11794275	766.13
- K5	1	195290	11842816	766.37
- K8	1	207726	11855253	766.43
- K14	1	271565	11919091	766.75
<none>			11647526	767.39
- K1	1	491865	12139391	767.83
- K6	1	810053	12457579	769.36
- K3	1	835945	12483471	769.48
- K10	1	1037715	12685241	770.43
- K7	1	1109564	12757090	770.76
- K16	1	1543763	13191289	772.73
- K15	1	1653154	13300680	773.22
- K19	1	1912093	13559620	774.36
- K22	1	2664461	14311987	777.55

Step: AIC=765.39
price ~ K1 + K2 + K3 + K4 + K5 + K6 + K7 + K8 + K9 + K10 + K11 +
K12 + K14 + K15 + K16 + K17 + K18 + K19 + K20 + K21 + K22 +
K23

	Df	Sum of Sq	RSS	AIC
- K18	1	377	11648339	763.40
- K11	1	388	11648350	763.40
- K21	1	2406	11650368	763.41
- K4	1	3831	11651793	763.41
- K9	1	18832	11666794	763.49
- K17	1	70048	11718010	763.75
- K12	1	79944	11727906	763.80
- K23	1	87662	11735624	763.84

```
- K2      1     92220 11740182 763.86
- K20     1    146341 11794303 764.13
- K5      1    195019 11842981 764.37
- K8      1    220269 11868231 764.50
- K14     1    284699 11932660 764.82
<none>                11647962 765.39
- K1      1    500675 12148637 765.88
- K6      1    816402 12464364 767.39
- K3      1    910534 12558496 767.83
- K10     1   1148645 12796607 768.94
- K7      1   1711664 13359626 771.48
- K15     1   1752092 13400054 771.66
- K19     1   1922587 13570548 772.41
- K22     1   2811166 14459128 776.15
- K16     1   4189091 15837053 781.52

Step:  AIC=763.4
price ~ K1 + K2 + K3 + K4 + K5 + K6 + K7 + K8 + K9 + K10 + K11 +
    K12 + K14 + K15 + K16 + K17 + K19 + K20 + K21 + K22 + K23

        Df Sum of Sq      RSS    AIC
- K11    1       537 11648876 761.40
- K21    1      2312 11650651 761.41
- K4     1      6262 11654601 761.43
- K9     1     19938 11668277 761.50
- K17    1     69709 11718047 761.75
- K12    1     80705 11729044 761.80
- K23    1     87287 11735625 761.84
- K2     1     91880 11740219 761.86
- K20    1    150213 11798552 762.15
- K5     1    197617 11845956 762.39
- K8     1    220075 11868414 762.50
- K14    1    287353 11935692 762.83
<none>              11648339 763.40
- K1     1    505344 12153683 763.90
- K6     1    847793 12496132 765.54
- K3     1    919541 12567879 765.88
- K10    1   1153692 12802031 766.97
- K15    1   1792596 13440935 769.84
- K7     1   1857604 13505943 770.13
- K19    1   1926760 13575099 770.43
- K22    1   2895110 14543449 774.49
- K16    1   4433120 16081459 780.42
```

Step: AIC=761.4
price ~ K1 + K2 + K3 + K4 + K5 + K6 + K7 + K8 + K9 + K10 + K12 +
    K14 + K15 + K16 + K17 + K19 + K20 + K21 + K22 + K23

	Df	Sum of Sq	RSS	AIC
- K21	1	1937	11650812	759.41
- K4	1	6792	11655668	759.43
- K9	1	20047	11668922	759.50
- K17	1	70126	11719002	759.75
- K12	1	80810	11729686	759.81
- K23	1	87123	11735999	759.84
- K2	1	91527	11740403	759.86
- K20	1	149886	11798762	760.15
- K5	1	197671	11846546	760.39
- K8	1	228512	11877388	760.54
- K14	1	291140	11940016	760.85
<none>			11648876	761.40
- K1	1	540419	12189295	762.07
- K6	1	848967	12497843	763.55
- K3	1	922487	12571363	763.89
- K10	1	1279157	12928033	765.54
- K15	1	1803137	13452012	767.89
- K7	1	1861077	13509952	768.14
- K19	1	2049192	13698068	768.96
- K22	1	3288226	14937102	774.07
- K16	1	4432584	16081459	778.42

Step: AIC=759.41
price ~ K1 + K2 + K3 + K4 + K5 + K6 + K7 + K8 + K9 + K10 + K12 +
    K14 + K15 + K16 + K17 + K19 + K20 + K22 + K23

	Df	Sum of Sq	RSS	AIC
- K4	1	5895	11656707	757.44
- K9	1	18952	11669764	757.50
- K17	1	72601	11723413	757.77
- K12	1	79698	11730510	757.81
- K23	1	88932	11739745	757.86
- K2	1	94009	11744822	757.88
- K20	1	155118	11805930	758.19
- K5	1	202812	11853625	758.43
- K8	1	227057	11877869	758.55
- K14	1	290769	11941581	758.86
<none>			11650812	759.41
- K1	1	647821	12298633	760.60

```
- K6      1    915944 12566756 761.87
- K3      1    933321 12584134 761.95
- K10     1   1308365 12959178 763.69
- K7      1   1870294 13521107 766.19
- K15     1   1933644 13584457 766.47
- K19     1   2255969 13906781 767.85
- K22     1   4414251 16065063 776.36
- K16     1   5331860 16982673 779.64

Step:  AIC=757.44
price ~ K1 + K2 + K3 + K5 + K6 + K7 + K8 + K9 + K10 + K12 + K14 +
    K15 + K16 + K17 + K19 + K20 + K22 + K23

        Df Sum of Sq      RSS    AIC
- K9     1     24306 11681013 755.56
- K17    1     67573 11724280 755.78
- K12    1     74346 11731053 755.81
- K2     1     97912 11754620 755.93
- K20    1    150243 11806950 756.19
- K23    1    164683 11821390 756.27
- K5     1    209019 11865727 756.49
- K8     1    272882 11929589 756.80
- K14    1    292062 11948769 756.90
<none>               11656707 757.44
- K1     1    667559 12324267 758.72
- K6     1    971701 12628408 760.16
- K3     1   1121709 12778416 760.86
- K7     1   1872418 13529125 764.23
- K10    1   2073824 13730531 765.10
- K15    1   2291663 13948370 766.03
- K19    1   2490313 14147020 766.86
- K22    1   4467792 16124499 774.58
- K16    1   5584063 17240770 778.53

Step:  AIC=755.56
price ~ K1 + K2 + K3 + K5 + K6 + K7 + K8 + K10 + K12 + K14 +
    K15 + K16 + K17 + K19 + K20 + K22 + K23

        Df Sum of Sq      RSS    AIC
- K2     1     81535 11762548 753.97
- K17    1     84851 11765863 753.99
- K12    1    111675 11792688 754.12
- K20    1    140932 11821945 754.27
- K23    1    179910 11860923 754.46
```

```
- K5      1      193695 11874708 754.53
- K8      1      253361 11934374 754.83
- K14     1      275152 11956165 754.93
<none>                  11681013 755.56
- K1      1      706741 12387754 757.03
- K6      1      951668 12632681 758.18
- K3      1     1123674 12804687 758.98
- K10     1     2053439 13734452 763.11
- K15     1     2267437 13948450 764.03
- K19     1     2516439 14197452 765.07
- K7      1     2559763 14240776 765.25
- K22     1     5410413 17091426 776.02
- K16     1     5562025 17243038 776.54

Step:  AIC=753.97
price ~ K1 + K3 + K5 + K6 + K7 + K8 + K10 + K12 + K14 + K15 +
    K16 + K17 + K19 + K20 + K22 + K23

          Df Sum of Sq      RSS      AIC
- K17     1     77070 11839618 752.36
- K12     1     77473 11840022 752.36
- K20     1     97764 11860312 752.46
- K8      1    203300 11965848 752.98
- K23     1    206942 11969491 753.00
- K5      1    372117 12134665 753.81
<none>                 11762548 753.97
- K14     1    530724 12293272 754.57
- K6      1    879121 12641670 756.22
- K1      1    931203 12693751 756.47
- K3      1   1107540 12870088 757.28
- K10     1   2005529 13768077 761.26
- K15     1   2190325 13952873 762.05
- K19     1   2503552 14266101 763.36
- K7      1   2553865 14316414 763.56
- K22     1   5329000 17091549 774.02
- K16     1   6110757 17873305 776.66

Step:  AIC=752.36
price ~ K1 + K3 + K5 + K6 + K7 + K8 + K10 + K12 + K14 + K15 +
    K16 + K19 + K20 + K22 + K23

          Df Sum of Sq      RSS      AIC
- K12     1    168034 12007652 751.19
- K20     1    177129 12016747 751.23
```

```
- K23      1       208799 12048417 751.39
- K8       1       212808 12052426 751.41
<none>                    11839618 752.36
- K5       1       450716 12290334 752.56
- K14      1       716959 12556577 753.82
- K1       1       970087 12809705 755.00
- K3       1      1084594 12924212 755.53
- K6       1      1187199 13026817 755.99
- K10      1      1946677 13786295 759.34
- K7       1      2608453 14448071 762.10
- K19      1      2799219 14638837 762.88
- K15      1      4592235 16431853 769.69
- K22      1      5562417 17402035 773.08
- K16      1      6485663 18325281 776.13

Step:  AIC=751.19
price ~ K1 + K3 + K5 + K6 + K7 + K8 + K10 + K14 + K15 + K16 +
    K19 + K20 + K22 + K23

          Df Sum of Sq          RSS       AIC
- K23      1       197015 12204667 750.15
- K8       1       206966 12214619 750.20
- K20      1       258553 12266205 750.44
<none>                    12007652 751.19
- K5       1       574771 12582423 751.95
- K14      1       766431 12774083 752.84
- K1       1       998341 13005994 753.90
- K3       1      1096592 13104244 754.34
- K6       1      1158344 13165996 754.62
- K19      1      2656397 14664049 760.98
- K7       1      2972683 14980335 762.24
- K15      1      4524367 16532019 768.05
- K22      1      5397939 17405591 771.09
- K10      1      5404471 17412123 771.11
- K16      1      6346307 18353959 774.22

Step:  AIC=750.15
price ~ K1 + K3 + K5 + K6 + K7 + K8 + K10 + K14 + K15 + K16 +
    K19 + K20 + K22

          Df Sum of Sq          RSS       AIC
- K8       1       375434 12580101 749.94
<none>                    12204667 750.15
- K5       1       424346 12629013 750.16
```

```
- K20     1      830350 13035018 752.03
- K14     1     1001252 13205920 752.80
- K3      1     1310849 13515516 754.17
- K6      1     1760318 13964985 756.10
- K1      1     2298516 14503183 758.33
- K7      1     2778309 14982976 760.25
- K19     1     2883445 15088112 760.66
- K15     1     4327352 16532019 766.05
- K10     1     5305364 17510031 769.44
- K16     1     6871444 19076112 774.50
- K22     1     7718819 19923486 777.06

Step:  AIC=749.94
price ~ K1 + K3 + K5 + K6 + K7 + K10 + K14 + K15 + K16 + K19 +
    K20 + K22

          Df Sum of Sq        RSS       AIC
<none>                    12580101 749.94
- K20     1     544568 13124670 750.44
- K5      1     627948 13208050 750.81
- K14     1     984412 13564514 752.38
- K6      1    1513537 14093638 754.64
- K3      1    1516443 14096544 754.65
- K1      1    2242555 14822657 757.61
- K7      1    2805762 15385863 759.81
- K19     1    3231155 15811256 761.42
- K15     1    4087934 16668036 764.54
- K10     1    4932299 17512401 767.45
- K22     1    7344438 19924539 775.07
- K16     1    7637820 20217921 775.93
```

有上述输出可以看出，逐步回归后保留的变量为 K1 + K3 + K5 + K6 + K7 + K10 + K14 + K15 + K16 + K19 + K20 + K22

summary(model_lm_step)

输出模型 model_lm_step 的摘要信息，如下：

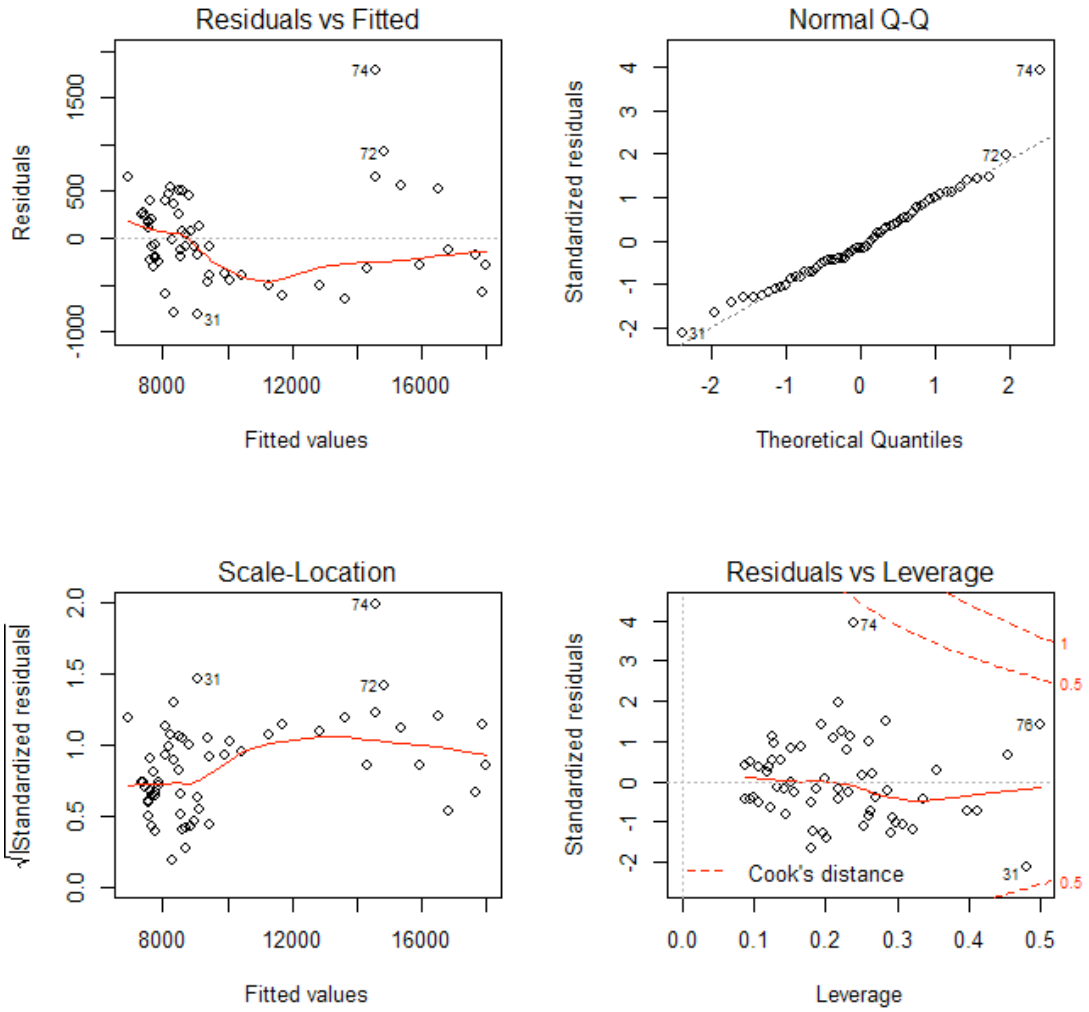
```
Call:
lm(formula = price ~ K1 + K3 + K5 + K6 + K7 + K10 + K14 + K15 +
    K16 + K19 + K20 + K22, data = train_data[, -1])

Residuals:
     Min       1Q   Median       3Q      Max
-804.09 -301.82  -85.83  261.21 1792.96

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
```

(Intercept) 7294.37835 642.86343 11.347 6.30e-15 ***
K1 -0.21129 0.07379 -2.864 0.006290 **
K3 -0.53024 0.22518 -2.355 0.022857 *
K5 0.28063 0.18519 1.515 0.136538
K6 0.82596 0.35110 2.353 0.022981 *
K7 0.46296 0.14454 3.203 0.002470 **
K10 -1.40823 0.33160 -4.247 0.000104 ***
K14 -0.08111 0.04275 -1.897 0.064082 .
K15 -0.52281 0.13523 -3.866 0.000345 ***
K16 0.64204 0.12149 5.285 3.36e-06 ***
K19 0.19258 0.05603 3.437 0.001257 **
K20 -0.46423 0.32898 -1.411 0.164938
K22 0.59873 0.11553 5.182 4.76e-06 ***

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 523 on 46 degrees of freedom
Multiple R-squared: 0.9791, Adjusted R-squared: 0.9737
F-statistic: 180 on 12 and 46 DF, p-value: < 2.2e-16
由上述输出可知，模型系数显著性状况较好，许多系数的 P 值小于 0.05，即系数显著，从 R 方来看为 0.9791，也比较高，调整后 R 方为 0.9737，回归方程的显著性检验 F 统计量为 2.2e-16，小于 0.05，说明回归方程整体显著。

opar <- par()
par()是所有的绘图变量 然后保存在 opar 里面 准备以后恢复
par(mfrow=c(2,2))
4 个区域绘图
plot(model_lm_step)
par <- opar
恢复原始绘图参数
绘制逐步回归后的诊断图，如下：



简单说一下第一行第二个图，QQ 图呈现一条线，说明方程拟合效果好

pred_lm_test <- predict(model_lm_step, test_data[,-c(1,2)])
利用模型 model_lm_step 对测试集数据进行预测，结果记作 pred_lm_test，predict 表示进行模型预测，model_lm_step 表示采用的预测模型为 model_lm_step，test_data[,-c(1,2)表示预测的数据集为测试集，并且去掉月份、价格数据。
MSE_lm <- MSE(test_data[,2], pred_lm_test)
NMSE_lm <- NMSE(test_data[,2], pred_lm_test)
利用函数计算逐步回归后模型的 MSE、NMSE，并记作 MSE_lm、NMSE_lm

#--
LASSO Models
#--
【模型简介】
普通线性模型遇到困难的情况下，比如模型精度不够，或者难以解释，lasso 回归通过对最小二乘估计加入惩罚约束，使某些系数的估计为 0，从而达到特征选择或者压缩变量的目的。有一种最小角回归（LARS）是求解 lasso 的具体算法。因为 lasso 不把所有的变量都放入模

型中进行拟合，而是有选择的把变量放入模型从而得到更好的性能参数。通过参数控制模型的复杂度，从而避免过度拟合。过程大家看一下很容易明白，筛选出变量建立模型求出模型系数，预测和线性模型就一样了。

```
lasso_train_data_x <- as.matrix(train_data[,c(3:25)])
lasso_train_data_y <- as.matrix(train_data[,2])
lasso_test_data_x <- as.matrix(test_data[,c(3:25)])
lasso_test_data_y <- as.matrix(test_data[,2])
```

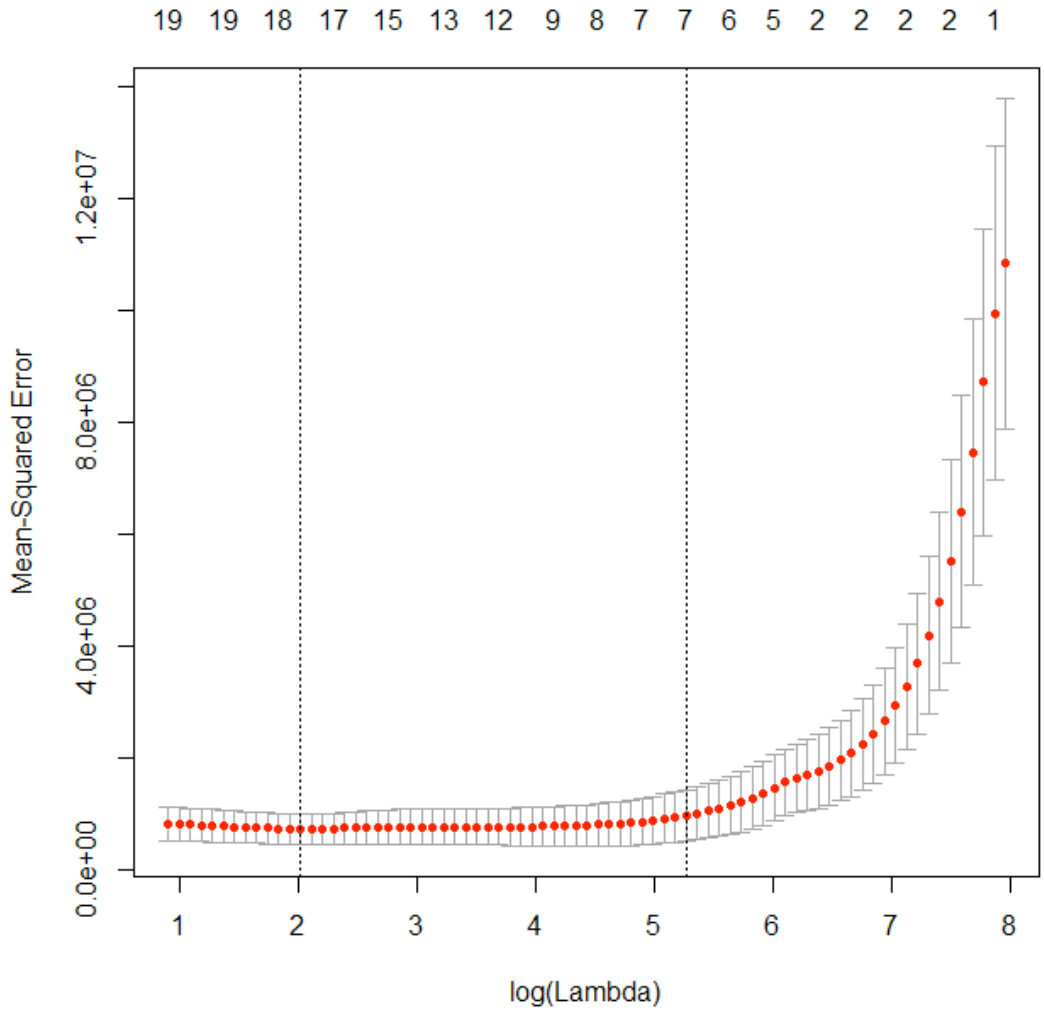
因为 lasso 模型要求输入的数据是矩阵格式，所以先把测试集、训练集的因变量、自变量拆分并转换成矩阵。

```
set.seed(8888)
model_lasso_cv <- cv.glmnet(lasso_train_data_x, lasso_train_data_y,
                            family = "gaussian", type.measure = "mse")
```

进行 lasso 交叉验证，目的是为例获得合适的 lambda 数值，结果记做 model_lasso_cv，cv.glmnet 表示进行交叉验证，lasso_train_data_x, lasso_train_data_y,表示采用训练集的自变量和因变量，family = "gaussian"表示适用于一维连续因变量，type.measure = "mse"表示使用拟合因变量与实际应变量的 mean squred error（即 MSE）

```
plot(model_lasso_cv)
```

绘制交叉验证图，如下：



如图所示，交叉验证，对于每一个 lambda 值，在红点所示目标参量的均值左右，都可以得到一个目标参量的置信区间。 两条虚线分别指示了两个特殊的 lambda 值，随着 lambda 的减少，参与建模的变量逐渐变多，误差也逐渐缩小，第一条虚线即为 lambda.min，第二条虚线即为 lambda.1se

model_lasso_cv$lambda.min　求值

12.35219

model_lasso_cv$lambda.1se

65.91994

lambda.min 是指在所有的 lambda 值中，得到最小目标参量均值的那一个。而 lambda.1se 是指在 lambda.min 一个方差范围内得到最简单模型的那一个 lambda 值。 因为 lambda 值到达一定大小之后，继续增加模型自变量个数即缩小 lambda 值，并不能很显著的提高模型性能，lambda.1se 给出的就是一个具备优良性能但是自变量个数最少的模型。

#建立 lasso 回归模型

model_lasso <- glmnet(lasso_train_data_x, lasso_train_data_y,

family = "gaussian", standardize = TRUE)

建立 lasso 模型，记作 model_lasso，glmnet 表示利用 glmnet 函数建立模型，lasso_train_data_x,

lasso_train_data_y, 表示采用训练集的自变量和因变量，family = "gaussian"表示适用于一维连续因变量，standardize = TRUE 表示对原始数据进行标准化（非必须的）

model_lasso

查看模型结果输出，如下：

Call: glmnet(x = lasso_train_data_x, y = lasso_train_data_y, family = "gaussian", standardize = TRUE)

	Df	%Dev	Lambda
[1,]	0	0.0000	2892.0000
[2,]	1	0.1389	2635.0000
[3,]	2	0.2678	2401.0000
[4,]	2	0.3760	2188.0000
[5,]	2	0.4658	1993.0000
[6,]	2	0.5404	1816.0000
[7,]	2	0.6023	1655.0000
[8,]	2	0.6537	1508.0000
[9,]	2	0.6964	1374.0000
[10,]	2	0.7318	1252.0000
[11,]	2	0.7612	1141.0000
[12,]	2	0.7857	1039.0000
[13,]	2	0.8059	947.0000
[14,]	2	0.8228	862.8000
[15,]	2	0.8367	786.2000
[16,]	2	0.8483	716.3000
[17,]	2	0.8580	652.7000
[18,]	2	0.8660	594.7000
[19,]	3	0.8728	541.9000
[20,]	3	0.8788	493.7000
[21,]	5	0.8852	449.9000
[22,]	5	0.8964	409.9000
[23,]	6	0.9062	373.5000
[24,]	6	0.9146	340.3000
[25,]	6	0.9215	310.1000
[26,]	6	0.9273	282.5000
[27,]	6	0.9321	257.4000
[28,]	6	0.9360	234.6000
[29,]	6	0.9393	213.7000
[30,]	7	0.9433	194.7000
[31,]	7	0.9477	177.4000
[32,]	7	0.9513	161.7000
[33,]	7	0.9544	147.3000
[34,]	7	0.9569	134.2000
[35,]	7	0.9589	122.3000
[36,]	8	0.9607	111.4000

```
[37,]  8 0.9623  101.5000
[38,]  8 0.9636   92.5200
[39,]  8 0.9646   84.3000
[40,]  8 0.9655   76.8100
[41,]  9 0.9662   69.9900
[42,]  9 0.9669   63.7700
[43,] 11 0.9674   58.1000
[44,] 11 0.9682   52.9400
[45,] 11 0.9689   48.2400
[46,] 12 0.9695   43.9500
[47,] 12 0.9704   40.0500
[48,] 12 0.9712   36.4900
[49,] 12 0.9718   33.2500
[50,] 12 0.9723   30.3000
[51,] 12 0.9727   27.6000
[52,] 13 0.9731   25.1500
[53,] 13 0.9735   22.9200
[54,] 13 0.9738   20.8800
[55,] 13 0.9740   19.0300
[56,] 14 0.9743   17.3400
[57,] 15 0.9747   15.8000
[58,] 16 0.9751   14.3900
[59,] 17 0.9757   13.1100
[60,] 17 0.9765   11.9500
[61,] 17 0.9772   10.8900
[62,] 17 0.9777    9.9210
[63,] 17 0.9781    9.0390
[64,] 17 0.9785    8.2360
[65,] 17 0.9788    7.5050
[66,] 17 0.9791    6.8380
[67,] 18 0.9793    6.2300
[68,] 18 0.9795    5.6770
[69,] 19 0.9796    5.1730
[70,] 19 0.9798    4.7130
[71,] 19 0.9799    4.2940
[72,] 19 0.9801    3.9130
[73,] 19 0.9801    3.5650
[74,] 19 0.9802    3.2490
[75,] 19 0.9803    2.9600
[76,] 19 0.9803    2.6970
[77,] 19 0.9804    2.4570
[78,] 19 0.9804    2.2390
[79,] 20 0.9805    2.0400
[80,] 20 0.9805    1.8590
```

[81,] 20 0.9805 1.6940
[82,] 20 0.9805 1.5430
[83,] 21 0.9806 1.4060
[84,] 22 0.9806 1.2810
[85,] 22 0.9806 1.1670
[86,] 22 0.9806 1.0640
[87,] 22 0.9806 0.9692
[88,] 22 0.9806 0.8831

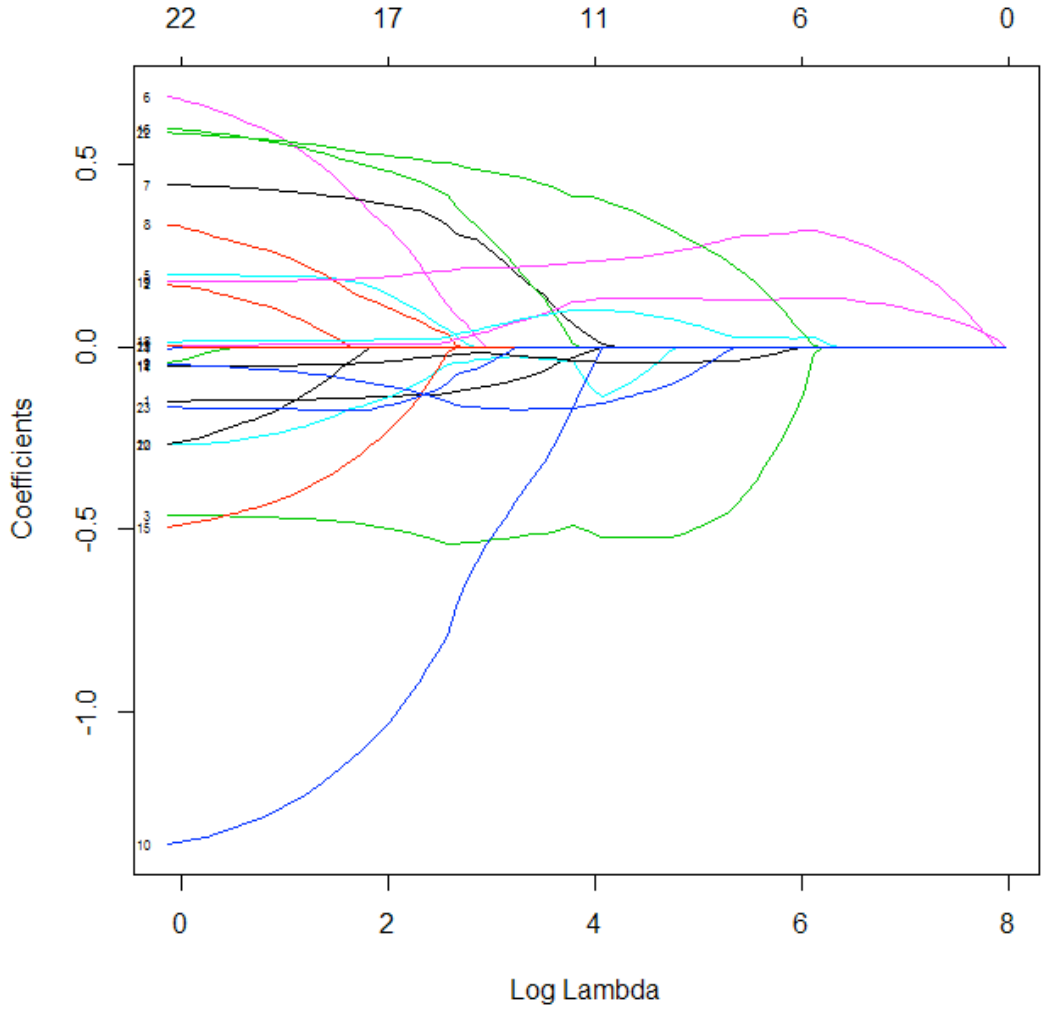
lasso 回归复杂度调整的程度由参数 lambda 来控制，lambda 越大对变量较多的线性模型的惩罚力度就越大，从而最终获得一个变量较少的模型。

每一行代表了一个模型。列 Df 是自由度，代表了非零的线性模型拟合系数的个数。列%Dev 代表了由模型解释的残差的比例，类似线性模型的模型拟合的 R2(R-squred)。它在 0 和 1 之间，越接近 1 说明模型的表现越好。列 lambda 就是每个模型对应的 lambda 值

从模型的输出结果看，随着 lambda 的逐渐减少，参与建模的变量逐渐变多，最后用了 22 个变量，%Dev 达到 0.9806。

```
plot(model_lasso, xvar = "lambda", label = TRUE)
```

绘制 lambda 图，如下：



图中的每一条曲线代表了每一个自变量系数的变化轨迹，纵坐标是系数的值，下横坐标是log(lambda)，上横坐标是此时模型中非零系数的个数。各个变量的系数随着 lambda 的变小而变大。

coef(model_lasso, s = 65.91994)
提取 lambda.1se 时的模型系数，如下：
(Intercept) 9590.85803438
K1 -0.01452518
K2 .
K3 -0.73609023
K4 .
K5 .
K6 .
K7 .
K8 .
K9 .
K10 .

K11 .
K12 -0.21477606
K13 0.16153771
K14 .
K15 .
K16 .
K17 -0.19325702
K18 0.18218623
K19 0.21246195
K20 .
K21 .
K22 0.29292224
K23 .

系数中为点的就是系数为 0，有数字的就是保留下来的模型变量的系数。

```
pred_lasso_test <- predict(model_lasso_cv, newx = lasso_test_data_x,
                                        type = "class", s = "lambda.1se")
```

利用 model_lasso_cv 对测试集数据进行预测，结果记作 pred_ lasso _test，predict 表示进行模型预测，model_lasso_cv 表示采用的预测模型为 model_lasso_cv，newx = lasso_test_data_x 表示预测的数据集为测试集自变量矩阵，type = "class"表示预测方式为数值，s = "lambda.1se"表示采用的模型系数为 lambda.1se 时候的模型系数。

```
MSE_lasso <- MSE(lasso_test_data_y, pred_lasso_test)
NMSE_lasso <- NMSE(lasso_test_data_y, pred_lasso_test)
```

利用函数计算 lasso 模型的 MSE、NMSE，并记作 MSE_ lasso、NMSE_ lasso

```
#-----------------------------------------------------------------
# Regression Trees
#-----------------------------------------------------------------
```

【模型简介】

训练模型的时候根据对回归树用平方误差最小化准则，对分类树用基尼指数最小化准则，进行特征选择，生成二叉树，节点的每次分裂都把原样本空间划分为互不相交的两个子集。每次都根据某个局部标准，选择最好的划分。以此不断分叉，分到不能分为止，这样就得到了一个训练好的回归树，在进行预测的时候，变量的数值代入到回归树中，根据不同的叶子规则得到不同的预测值。

```
model_rpart <- rpart(price~., data = train_data[,-1], method = 'anova')
```

建立回归树模型，记作 model_rpart，rpart 表示建立模型，price~.,表示 price 作为因变量，其余 K1 至 K23 作为自变量，data = train_data[,-1]表示数据集为训练集，并且去掉第一列月份数据，method = 'anova'表示根据树末端的数据类型选择相应变量分割方法，连续型即为 anova。

```
summary(model_rpart)
```

查看模型输出摘要，如下：

Call:

```
rpart(formula = price ~ ., data = train_data[, -1], method = "anova",
      cp = 0.01)
  n= 59

          CP nsplit  rel error     xerror       xstd
1 0.88418495      0 1.00000000 1.0735310 0.2136811
2 0.03716285      1 0.11581505 0.4885576 0.1886565
3 0.01224671      2 0.07865221 0.4154203 0.1660036
4 0.01000000      3 0.06640550 0.3545943 0.1350514

Variable importance
K22  K7 K15 K18 K13 K19  K4 K10 K11  K5  K6
 18  16  16  16  15  14   1   1   1   1   1

Node number 1: 59 observations,    complexity param=0.8841849
  mean=10070.03, MSE=1.022594e+07
  left son=2 (46 obs) right son=3 (13 obs)
  Primary splits:
      K22 < 7730    to the left,  improve=0.8841849, (0 missing)
      K7  < 9051.5  to the left,  improve=0.8386171, (0 missing)
      K13 < 27242.5 to the left,  improve=0.8386171, (0 missing)
      K15 < 7800.5  to the left,  improve=0.8386171, (0 missing)
      K19 < 6572.5  to the left,  improve=0.8346294, (0 missing)
  Surrogate splits:
      K7  < 9051.5  to the left,  agree=0.966, adj=0.846, (0 split)
      K13 < 27242.5 to the left,  agree=0.966, adj=0.846, (0 split)
      K15 < 7800.5  to the left,  agree=0.966, adj=0.846, (0 split)
      K18 < 6872    to the left,  agree=0.966, adj=0.846, (0 split)
      K19 < 6572.5  to the left,  agree=0.949, adj=0.769, (0 split)

Node number 2: 46 observations,    complexity param=0.03716285
  mean=8471.522, MSE=757121.9
  left son=4 (19 obs) right son=5 (27 obs)
  Primary splits:
      K4  < 3048.5  to the left,  improve=0.6437846, (0 missing)
      K5  < 7417    to the left,  improve=0.6437846, (0 missing)
      K6  < 2835    to the left,  improve=0.6437846, (0 missing)
      K7  < 3550    to the left,  improve=0.6437846, (0 missing)
      K10 < 2297    to the left,  improve=0.6437846, (0 missing)
  Surrogate splits:
      K5  < 7417    to the left,  agree=1, adj=1, (0 split)
      K6  < 2835    to the left,  agree=1, adj=1, (0 split)
      K7  < 3550    to the left,  agree=1, adj=1, (0 split)
      K10 < 2297    to the left,  agree=1, adj=1, (0 split)
```

```
      K11 < 2087.5    to the left,    agree=1, adj=1, (0 split)

Node number 3: 13 observations
  mean=15726.31, MSE=2695934

Node number 4: 19 observations
  mean=7639.263, MSE=21598.09

Node number 5: 27 observations,       complexity param=0.01224671
  mean=9057.185, MSE=444287.6
  left son=10 (19 obs) right son=11 (8 obs)
  Primary splits:
      K7   < 7322       to the left,   improve=0.6159519, (0 missing)
      K15 < 6772       to the left,   improve=0.5737824, (0 missing)
      K4   < 5431.5    to the left,   improve=0.5620565, (0 missing)
      K18 < 5054       to the left,   improve=0.5418258, (0 missing)
      K13 < 16048      to the left,   improve=0.4610160, (0 missing)
  Surrogate splits:
      K15 < 6772       to the left,   agree=0.963, adj=0.875, (0 split)
      K18 < 5054       to the left,   agree=0.926, adj=0.750, (0 split)
      K19 < 6066       to the left,   agree=0.926, adj=0.750, (0 split)
      K1   < 3871.5    to the left,   agree=0.889, adj=0.625, (0 split)
      K4   < 5224.5    to the left,   agree=0.889, adj=0.625, (0 split)

Node number 10: 19 observations
  mean=8717.737, MSE=60266.09

Node number 11: 8 observations
  mean=9863.375, MSE=432737
```

以上描述了模型各个节点的情况

```
pred_lm_test <- predict(model_rpart, test_data[,-c(1,2)])
```

利用模型 model_rpart 对测试集数据进行预测，结果记作 pred_lm_test，predict 表示进行模型预测，model_rpart 表示采用的预测模型为 model_rpart，test_data[,-c(1,2)表示预测的数据集为测试集，并且去掉月份、价格数据。

```
MSE_rpart <- MSE(test_data[,2], pred_lm_test)
NMSE_rpart <- NMSE(test_data[,2], pred_lm_test)
```

利用函数计算 回归树 模型的 MSE、NMSE，并记作 MSE_ rpart、NMSE_ rpart。

```
draw.tree(model_rpart)
#------------------------------------------------------------
# bagging
#------------------------------------------------------------
```

【模型简介】

Bootstraping 是一种有放回的抽样方法（可能抽到重复的样本），从原始样本集中抽取训练集。

每轮从原始样本集中使用 Bootstraping 的方法抽取 n 个训练样本（在训练集中，有些样本可能被多次抽取到，而有些样本可能一次都没有被抽中）。共进行 k 轮抽取，得到 k 个训练集。（k 个训练集之间是相互独立的），每次使用一个训练集得到一个模型，k 个训练集共得到 k 个模型。（注：这里并没有具体的分类算法或回归方法，我们可以根据具体问题采用不同的分类或回归方法，如决策树、感知器等），对分类问题：将上步得到的 k 个模型采用投票的方式得到分类结果；对回归问题，计算上述模型的均值作为最后的结果。（所有模型的重要性相同）

其实有一篇博文写的很好，这个也是在其中摘录的，大家可以看看

http://www.cnblogs.com/liuwu265/p/4690486.html

所以 adaboost 和 bagging 还是有本质区别的。

```
set.seed(8888)
model_bagging <- bagging(price ~ ., data = train_data[,-1],
                                    nbagg = 25, coob = TRUE)
```

建立袋装模型，记作 model_bagging，bagging 表示建立模型，price~.,表示 price 作为因变量，其余 K1 至 K23 作为自变量，data = train_data[,-1]表示数据集为训练集，并且去掉第一列月份数据，nbagg = 25 表示迭代 25 次，coob = TRUE 表示计算装袋错误。

```
model_bagging
```

查看模型输出如下：

```
Bagging regression trees with 25 bootstrap replications

Call: bagging.data.frame(formula = price ~ ., data = train_data[, -1],
      nbagg = 25, coob = TRUE)

Out-of-bag estimate of root mean squared error:  1379.294
Out-of-bag estimate of root mean squared error:  1379.294
```

就是 coob = TRUE 的作用。

```
pred_bagging_test <- predict(model_bagging,  test_data[,-c(1,2)])
```

利用模型 model_bagging 对测试集数据进行预测，结果记作 pred_bagging_test，predict 表示进行模型预测，model_bagging 表示采用的预测模型为 model_bagging，test_data[,-c(1,2)表示预测的数据集为测试集，并且去掉月份、价格数据。

```
MSE_bagging <- MSE(test_data[,2], pred_bagging_test)
NMSE_bagging <- NMSE(test_data[,2], pred_bagging_test)
```

利用函数计算 bagging 模型的 MSE、NMSE，并记作 MSE_ bagging、NMSE_ bagging。

```
#-------------------------------------------------------------
# Random Forest
#-------------------------------------------------------------
```

【模型简介】

随机森林是一个用随机方式建立的，包含多个决策树的分类器。对于分类其输出的类别是由各个树输出的类别的众数而定，对于回归就是各个树的平均值。简单说 Bagging + 决策树 = 随机森林。随机性主要体现在两个方面：数据的随机性选取，以及待选特征的随机选取。数据的随机选取：从原始的数据集中采取有放回的抽样，构造子数据集，第二，利用子数据集来构建子决策树。特征的随机选取，随机森林中的子树的每一个分裂过程并未用到所有的待

选特征，而是从所有的待选特征中随机选取一定的特征，之后再在随机选取的特征中选取最优的特征。这样能够使得随机森林中的决策树都能够彼此不同，提升系统的多样性，从而提升性能。

```
set.seed(8888)
model_rf <- randomForest(price~., data = train_data[,-1],
                         importance = TRUE, proximity = TRUE,
                         ntree = 500, mtry = 4)
```

建立随机森林模型，记作 model_rf，randomForest 表示建立随机森林模型，price~.,表示 price 作为因变量，其余 K1 至 K23 作为自变量，data = train_data[,-1]表示数据集为训练集，并且去掉第一列月份数据，importance = TRUE 表示计算重要性得分，proximity = TRUE 表示调用随机森林，ntree = 500 表示设置生长数为 500 棵，mtry=4 表示每一个分裂节点处样本预测器的个数为 4 个。

```
model_rf
```

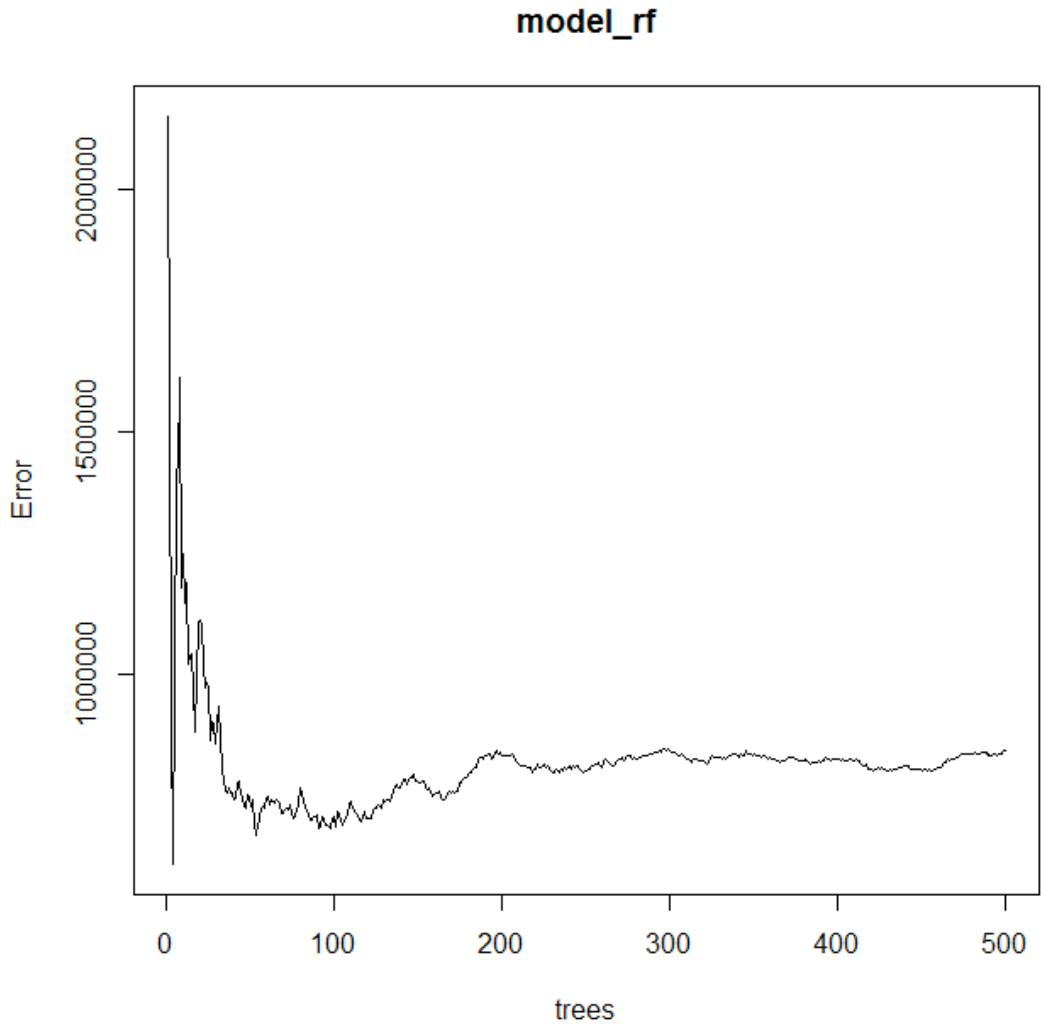
查看模型结果输出，如下：

```
Call:
 randomForest(formula = price ~ ., data = train_data[, -1], importance = TRUE,          proximity
= TRUE, ntree = 500, mtry = 4)
                Type of random forest: regression
                      Number of trees: 500
No. of variables tried at each split: 4

              Mean of squared residuals: 845033.1
                        % Var explained: 91.74
```

由上述输出可以看出，模型均方误差为 845033.1，可解释的方差比例为 91.74%。

```
plot(model_rf)
```

对随机森林的结果进行作图，如下：



从图中可以看出，随着树的增加，模型的误差率逐渐下降，随后略上升并保持平稳。

importance(model_rf)

查看变量重要性得分，如下：

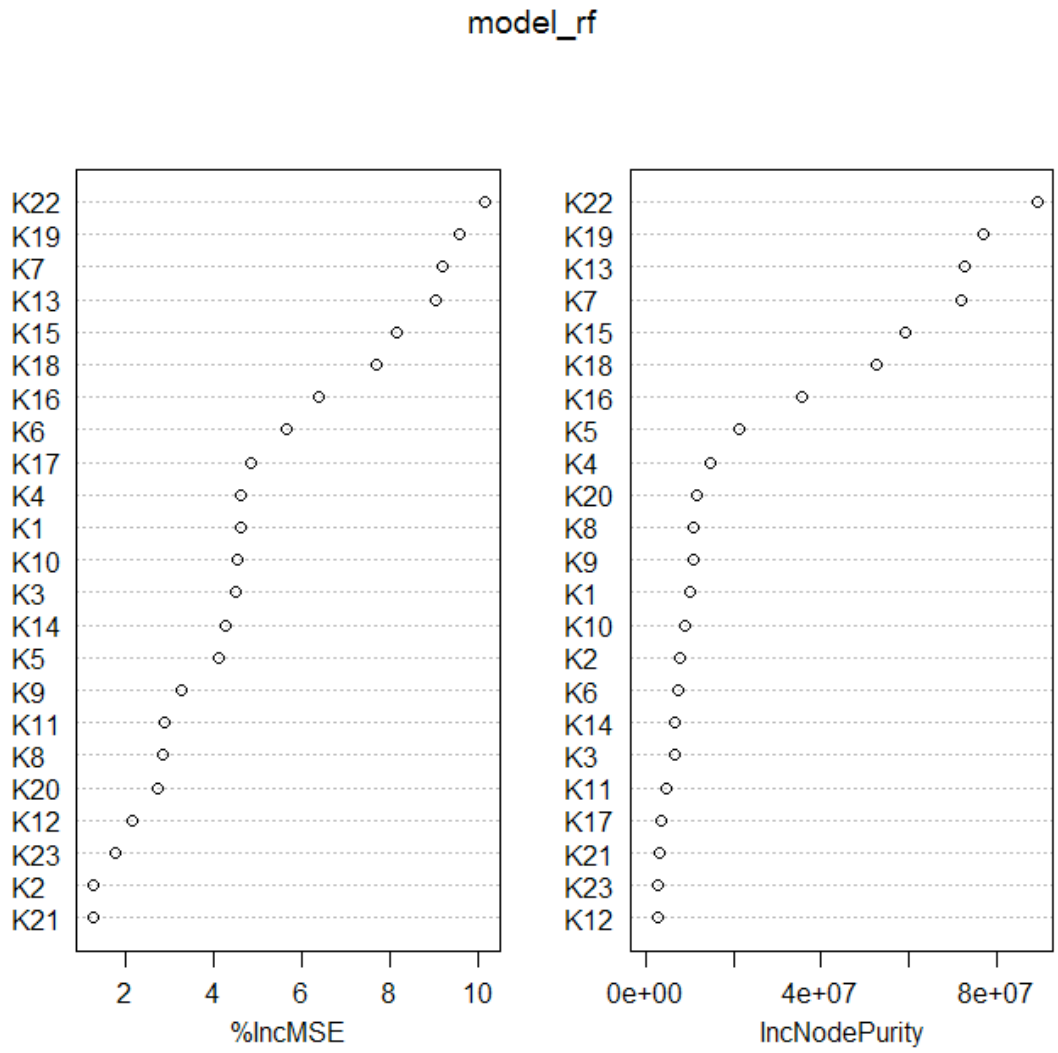
	%IncMSE	IncNodePurity
K1	4.603629	10115286
K2	1.291151	7558460
K3	4.48612	6509118
K4	4.610686	14882364
K5	4.120671	21329205
K6	5.648198	7374789
K7	9.183312	71967457
K8	2.857043	10929501
K9	3.261924	10823609
K10	4.539573	8873328
K11	2.873499	4592134
K12	2.170433	2708425
K13	9.018837	72859666

K14	4.265003	6666635
K15	8.150087	59185431
K16	6.378096	35617915
K17	4.854133	3484960
K18	7.69279	52706549
K19	9.551547	77057139
K20	2.718308	11582834
K21	1.257351	2893080
K22	10.126595	89435892
K23	1.787184	2802132

分别按照%IncMSE 和 IncNodePurity 列出了各个指标对模型的重要性影响程度，得分越高，重要程度越大。

varImpPlot(model_rf)

变量重要性程度绘图如下：



pred_rf_test <- predict(model_rf, test_data[,-c(1,2)])

利用模型 model_rf 对测试集数据进行预测，结果记作 pred_rf_test，predict 表示进行模型预测，model_rf 表示采用的预测模型为 model_rf，test_data[,-c(1,2)表示预测的数据集为测试集，

并且去掉月份、价格数据。

```
MSE_rf <- MSE(test_data[,2], pred_rf_test)
NMSE_rf <- NMSE(test_data[,2], pred_rf_test)
```

利用函数计算 rf 模型的 MSE、NMSE，并记作 MSE_ rf、NMSE_ rf。

```
#------------------------------------------------------------
# All MSE and NMSE
#------------------------------------------------------------
MSE_all <- cbind(MSE_lm, MSE_lasso, MSE_rpart, MSE_bagging,
                 MSE_rf)
NMSE_all <- cbind(NMSE_lm, NMSE_lasso, NMSE_rpart, NMSE_bagging,
                  NMSE_rf)
MSE_and_NMSE <- rbind(MSE_all, NMSE_all)
```

将所有模型的 MSE，NMSE 进行合并，如下表所示

	lm	lasso	rpart	bagging	rf
MSE	2.072152e+06	9.196795e+05	1.034961e+05	6.443617e+05	5.836782e+05
NMSE	2.615442e-01	1.160807e-01	1.306314e-02	8.133048e-02	7.367109e-02

```
#------------------------------------------------------------
# Full Data Predict
#------------------------------------------------------------
origin_data_pred <- origin_data[,-c(1,2)]

pred_lm_origin <- predict(model_lm, origin_data_pred)
pred_lasso_cv_origin <- as.numeric(predict(model_lasso_cv, as.matrix(origin_data_pred)))
pred_rpart_origin <- predict(model_rpart, origin_data_pred)
pred_bagging_origin <- predict(model_bagging, origin_data_pred)
pred_rf_origin <- predict(model_rf, origin_data_pred)

origin_data_pred_full <- cbind(origin_data[,c(1,2)], pred_lm_origin,
                               pred_lasso_cv_origin, pred_rpart_origin,
                               pred_bagging_origin, pred_rf_origin)
```

将所有模型对全部数据的拟合放在一起，如下表所示：

	month	price	pred_lm_origin	pred_lasso_cv_origin	pred_rpart_origin	pred_bagging_origin	pred_rf_origin
1	2011年2月	7368	7490.001	7533.208	7639.263	7710.313	7495.894
2	2011年3月	7432	7468.95	7911.415	7639.263	7710.313	7508.478
3	2011年4月	7551	7863.318	8164.201	7639.263	7710.313	7662.83
4	2011年5月	7567	7584.479	7937.704	7639.263	7710.313	7614.944
5	2011年6月	7607	6893.042	7901.7	7639.263	7710.313	7635.033

6	2011年7月	7632	7345.625	7862.618	7639.263	7710.313	7633.933
7	2011年8月	7672	7194.628	7672.049	7639.263	7710.313	7639.925
8	2011年9月	7737	7391.439	7664.722	7639.263	7710.313	7688.434
9	2011年10月	7732	7657.225	7729.977	7639.263	7710.313	7609.611
10	2011年11月	7691	7648.238	7943.931	7639.263	7710.313	7670.226
11	2011年12月	7540	8368.295	8279.085	7639.263	7710.313	7616.777
12	2012年1月	7608	7636.763	7827.52	7639.263	7710.313	7650.372
13	2012年2月	7474	8151.287	7627.812	7639.263	7710.313	7539.055
14	2012年3月	7623	7962.922	7826.75	7639.263	7822.728	7659.663
15	2012年4月	7594	7539.793	7833.842	7639.263	7822.728	7808.782
16	2012年5月	7567	7855.292	7540.637	7639.263	7710.313	7596.099
17	2012年6月	7588	7740.213	7612.401	7639.263	7710.313	7596.981
18	2012年7月	7685	7509.344	7382.428	7639.263	7822.728	7672.645
19	2012年8月	7657	7592.413	7374.313	7639.263	7710.313	7683.95
20	2012年9月	7636	7914.09	7480.054	7639.263	7710.313	7651.516
21	2012年10月	7687	7526.838	7496.941	7639.263	7710.313	7710.997
22	2012年11月	7734	7723.855	7433.344	7639.263	7755.636	7757.16
23	2012年12月	7843	7837.698	7457.941	7639.263	7710.313	7808.456
24	2013年1月	7890	7827.503	7398.44	7639.263	7710.313	7817.945
25	2013年2月	8031	7615.45	7311.751	7639.263	7755.636	7860.048
26	2013年3月	8126	8238.164	7135.089	7639.263	7710.313	7715.628
27	2013年4月	8166	7942.968	7265.048	7639.263	7880.334	7858.901
28	2013年5月	8189	8098.523	7631.006	7639.263	7710.313	7647.317
29	2013年6月	8278	8749.21	9210.916	8717.737	8405.754	8621.798
30	2013年7月	8363	8530.042	9093.707	8717.737	8454.829	8453.351
31	2013年8月	8302	8828.967	9218.907	8717.737	8408.765	8524.823
32	2013年9月	8290	8412.672	9162.911	8717.737	8454.829	8435.721
33	2013年10月	8436	8560.526	9205.735	8717.737	8408.765	8503.238
34	2013年11月	8487	8050.921	8988.669	8717.737	8638.863	8575.296
35	2013年12月	8675	8648.546	9227.267	8717.737	8454.829	8689.673
36	2014年1月	8728	8653.813	8294.661	8717.737	8914.019	8712.196
37	2014年2月	8617	8661.814	8467.832	8717.737	8247.276	8634.481
38	2014年3月	8661	8312.772	9112.395	8717.737	8864.945	8715.762
39	2014年4月	8829	8593.059	9021.389	8717.737	8937.47	8918.248

40	2014年5月	8893	9019.128	8892.176	8717.737	8937.47	8829.528
41	2014年6月	8892	8876.752	9158.943	8717.737	8914.019	9025.833
42	2014年7月	8735	8682.614	8976.235	8717.737	8996.543	8808.117
43	2014年8月	8800	8712.077	8954.48	8717.737	8996.543	8850.024
44	2014年9月	8714	8725.239	9236.242	8717.737	8996.543	8884.898
45	2014年10月	8714	8508.311	9056.318	8717.737	8955.043	8822.277
46	2014年11月	8804	8536.122	9346.137	8717.737	9082.206	9554.35
47	2014年12月	8886	8850.117	8899.092	8717.737	9041.182	9064.24
48	2015年1月	9099	8695.592	8702.172	8717.737	8986.544	9070.658
49	2015年2月	8786	8297.448	8670.197	8717.737	9082.206	8986.709
50	2015年3月	8923	9302.093	8626.405	8717.737	8202.916	8625.517
51	2015年4月	9034	9268.637	9645.472	8717.737	9082.206	9343.359
52	2015年5月	8974	9019.815	9344.336	8717.737	8825.428	9075.832
53	2015年6月	9028	8397.318	8922.685	8717.737	9170.187	9092.8
54	2015年7月	9085	7794.919	10070.812	9863.375	9411.49	9902.649
55	2015年8月	9248	8950.812	9646.876	9863.375	9667.53	9635.558
56	2015年9月	9256	8962.552	10041.759	9863.375	9463.184	9515.171
57	2015年10月	9380	9438.623	9696.104	9863.375	9408.546	9475.638
58	2015年11月	9541	9736.561	9812.319	9863.375	9463.184	9547.935
59	2015年12月	9418	10624.865	9684.346	9863.375	8914.412	9398.087
60	2016年1月	9611	10057.046	9552.91	9863.375	9411.49	9559.936
61	2016年2月	9806	9572.848	9635.705	9863.375	9411.49	9300.702
62	2016年3月	10030	10417.567	10507.449	9863.375	9369.99	10233.608
63	2016年4月	10762	11308.224	12453.056	9863.375	11463.947	12223.737
64	2016年5月	11079	11917.589	12146.195	9863.375	10547.683	11954.5
65	2016年6月	11661	12517.404	12861.298	15726.308	14266.688	13529.853
66	2016年7月	12340	12822.75	12886.881	15726.308	14622.177	12775.006
67	2016年8月	12960	13882.201	13011.992	15726.308	14062.249	13071.049
68	2016年9月	13982	14104.759	13967.869	15726.308	15586.942	14059.883
69	2016年10月	15225	14442.91	14375.872	15726.308	15586.942	15181.301
70	2016年11月	15635	15812.967	15138.289	15726.308	15586.942	15548.742
71	2016年12月	15937	15452.336	14194.574	15726.308	14127.141	15133.751
72	2017年1月	15762	14839.045	13512.303	15726.308	15344.637	15335.46
73	2017年2月	16050	13987.716	12421.839	9863.375	9652.688	10344.119

74	2017年3月	16363	14740.6	13222.036	15726.308	14174.623	14933.641
75	2017年4月	16689	16852.568	16387.117	15726.308	15586.942	16603.969
76	2017年5月	17072	16781.606	17024.491	15726.308	15344.637	16524.911
77	2017年6月	17302	17815.24	17172.021	15726.308	15586.942	16912.083
78	2017年7月	17474	17419.654	16942.844	15726.308	14823.505	16645.371
79	2017年8月	17701	18039.333	16152.764	15726.308	15366.391	16642.251
平均误差率			-0.11%	0.74%	0.14%	-0.44%	0.13%

附：

k1=二手房出售

k2=二手房交易

k3=二手房网

k4=房贷利率

k5=公积金

k6=公积金管理中心

k7=公积金提取

k8=户型

k9=户型图大全

k10=建材

k11=建材市场

k12=建材网

k13=武汉房价

k14=武汉公积金

k15=武汉楼盘

k16=武汉楼市

k17=武汉搜房网

k18=武汉装修

k19=武汉装修公司

k20=中国建材网

k21=租房

k22=租房合同

k23=最新房贷利率