- 这是北京市的短租数据集,我将对这个数据集进行数据分析并将数据可视化

一、数据分析和可视化

1.1数据分析

导入所依赖的库

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

忽略无关警告

```
# Ignore irrelevant warnings
import warnings
warnings.filterwarnings('ignore')
```

读取数据集中的内容

```
listings = pd.read_csv("data/listings.csv",index_col=0)
```

查看数据内容

```
listings
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type
id								
44054	Modern and Comfortable Living in CBD	192875	East Apartments	NaN	朝阳区 / Chaoyang	39.89503	116.45163	Entire home/apt
100213	The Great Wall Box Deluxe Suite A团园长城小院东院套房	527062	Joe	NaN	密云县 / Miyun	40.68434	117.17231	Private room
128496	Heart of Beijing: House with View 2	467520	Cindy	NaN	东城区	39.93213	116.42200	Entire home/apt
161902	cozy studio in center of Beijing	707535	Robert	NaN	东城区	39.93357	116.43577	Entire home/apt
162144	nice studio near subway, sleep 4	707535	Robert	NaN	朝阳区 / Chaoyang	39.93668	116.43798	Entire home/apt
...	...	...	...	...	...	...	...	...
33948728	望京西门子,798,央美附近 温馨如家大床民宿	256112655	旭	NaN	朝阳区 / Chaoyang	39.98671	116.47394	Entire home/apt
33948787	04简约舒适电梯房/工体/三里屯/东大桥/朝阳医院/世贸天阶/国贸	147335664	Pony	NaN	朝阳区 / Chaoyang	39.92560	116.44735	Entire home/apt
33950006	临近地铁温馨网红风小屋一居室	141786513	昊	NaN	朝阳区 / Chaoyang	39.89733	116.50473	Entire home/apt
33950535	3. 老国展,三元桥地铁,静安东里大床房	213500128	晓征	NaN	朝阳区 / Chaoyang	39.95988	116.45187	Private room
33954414	密码锁自行入住,隐私安全,丰台宋家庄交通枢纽站,去往北京站北京南站,天安门故宫,长城水魔方,...	252799678	超	NaN	丰台区 / Fengtai	39.84714	116.43481	Entire home/apt

28452 rows × 15 columns

属性名　记录的内容

- id　用于标识房间的号码，一个房间有独一份的id
- name 房东给房间起的名字
- host_id　房东的的编号
- host_name　房东的名字
- neighbourhood_group　所属组织
- neighbourhood　地区
- latitude 维度
- longitude　经度
- room_type　房间的类型
- price　价格
- minimum_nights 最少要住几夜
- number_of_reviews　有多少评论
- last_review　上一次评论
- reviews_per_month　平均每月评论数
- calculated_host_listings_count 有多少间房

- availability_365　一年有多少天可出租

查看数据前五行

```
listings.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimu
id										
44054	Modern and Comfortable Living in CBD	192875	East Apartments	NaN	朝阳区 / Chaoyang	39.89503	116.45163	Entire home/apt	792	1
100213	The Great Wall Box Deluxe Suite A团园长城小院东院套房	527062	Joe	NaN	密云县 / Miyun	40.68434	117.17231	Private room	1201	1
128496	Heart of Beijing: House with View 2	467520	Cindy	NaN	东城区	39.93213	116.42200	Entire home/apt	389	3
161902	cozy studio in center of Beijing	707535	Robert	NaN	东城区	39.93357	116.43577	Entire home/apt	376	1
162144	nice studio near subway, sleep 4	707535	Robert	NaN	朝阳区 / Chaoyang	39.93668	116.43798	Entire home/apt	537	1

查看数据集的信息概述

```
listings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 28452 entries, 44054 to 33954414
Data columns (total 15 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   name                            28451 non-null  object 
 1   host_id                         28452 non-null  int64  
 2   host_name                       28452 non-null  object 
 3   neighbourhood_group             0 non-null      float64
 4   neighbourhood                   28452 non-null  object 
 5   latitude                        28452 non-null  float64
 6   longitude                       28452 non-null  float64
 7   room_type                       28452 non-null  object 
 8   price                           28452 non-null  int64  
 9   minimum_nights                  28452 non-null  int64  
 10  number_of_reviews               28452 non-null  int64  
 11  last_review                     17294 non-null  object 
 12  reviews_per_month               17294 non-null  float64
 13  calculated_host_listings_count  28452 non-null  int64  
 14  availability_365                28452 non-null  int64  
dtypes: float64(4), int64(6), object(5)
memory usage: 2.9+ MB
```

对数据进行整理

```
#发现neighbourhood_group一栏全部都是缺失,因此舍弃这一列
listings = listings.drop(columns=['neighbourhood_group'],axis=1)
listings
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	name	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number
id										
44054	Modern and Comfortable Living in CBD	192875	East Apartments	朝阳区 / Chaoyang	39.89503	116.45163	Entire home/apt	792	1	89
100213	The Great Wall Box Deluxe Suite A团园长城小院东院套房	527062	Joe	密云县 / Miyun	40.68434	117.17231	Private room	1201	1	2
128496	Heart of Beijing: House with View 2	467520	Cindy	东城区	39.93213	116.42200	Entire home/apt	389	3	259
161902	cozy studio in center of Beijing	707535	Robert	东城区	39.93357	116.43577	Entire home/apt	376	1	26
162144	nice studio near subway, sleep 4	707535	Robert	朝阳区 / Chaoyang	39.93668	116.43798	Entire home/apt	537	1	37
...	...	...	...	...	...	...	...	...	...	...
33948728	望京西门子,798,央美附近 温馨如家大床民宿	256112655	旭	朝阳区 / Chaoyang	39.98671	116.47394	Entire home/apt	396	1	0
33948787	04简约舒适电梯房/工体/三里屯/东大桥/朝阳医院/世贸天阶/国贸	147335664	Pony	朝阳区 / Chaoyang	39.92560	116.44735	Entire home/apt	1302	3	0
33950006	临近地铁温馨网红风小屋一居室	141786513	昊	朝阳区 / Chaoyang	39.89733	116.50473	Entire home/apt	329	1	0
33950535	3. 老国展,三元桥地铁,静安东里大床房	213500128	晓征	朝阳区 / Chaoyang	39.95988	116.45187	Private room	188	1	0
33954414	密码锁自行入住,隐私安全,丰台宋家庄交通枢纽站,去往北京站北京南站,天安门故宫,长城水魔方,...	252799678	超	丰台区 / Fengtai	39.84714	116.43481	Entire home/apt	295	1	0

28452 rows × 14 columns

```
#对于neighbourhood这一列,我们只需要地区的中文名称即可,可以将后面的英文名称删除
listings['neighbourhood'] = listings['neighbourhood'].str.split('/').str[0]
listings
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	name	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number
id										
44054	Modern and Comfortable Living in CBD	192875	East Apartments	朝阳区	39.89503	116.45163	Entire home/apt	792	1	89
100213	The Great Wall Box Deluxe Suite A团园长城小院东院套房	527062	Joe	密云县	40.68434	117.17231	Private room	1201	1	2
128496	Heart of Beijing: House with View 2	467520	Cindy	东城区	39.93213	116.42200	Entire home/apt	389	3	259
161902	cozy studio in center of Beijing	707535	Robert	东城区	39.93357	116.43577	Entire home/apt	376	1	26
162144	nice studio near subway, sleep 4	707535	Robert	朝阳区	39.93668	116.43798	Entire home/apt	537	1	37
...	...	...	...	...	...	...	...	...	...	...
33948728	望京西门子,798,央美附近 温馨如家大床民宿	256112655	旭	朝阳区	39.98671	116.47394	Entire home/apt	396	1	0
33948787	04简约舒适电梯房/工体/三里屯/东大桥/朝阳医院/世贸天阶/国贸	147335664	Pony	朝阳区	39.92560	116.44735	Entire home/apt	1302	3	0
33950006	临近地铁温馨网红风小屋一居室	141786513	昊	朝阳区	39.89733	116.50473	Entire home/apt	329	1	0
33950535	3. 老国展,三元桥地铁,静安东里大床房	213500128	晓征	朝阳区	39.95988	116.45187	Private room	188	1	0
33954414	密码锁自行入住,隐私安全,丰台宋家庄交通枢纽站,去往北京站北京南站,天安门故宫,长城水魔方,...	252799678	超	丰台区	39.84714	116.43481	Entire home/apt	295	1	0

28452 rows × 14 columns

```
#根据listings.info()的信息,name列存在缺失值,name缺失,则这条数据是没有意义的,应该将此行删除
listings = listings[listings['name'].notnull()]
listings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 28451 entries, 44054 to 33954414
Data columns (total 14 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
```

```
 0   name                            28451 non-null  object
 1   host_id                         28451 non-null  int64
 2   host_name                       28451 non-null  object
 3   neighbourhood                   28451 non-null  object
 4   latitude                        28451 non-null  float64
 5   longitude                       28451 non-null  float64
 6   room_type                       28451 non-null  object
 7   price                           28451 non-null  int64
 8   minimum_nights                  28451 non-null  int64
 9   number_of_reviews               28451 non-null  int64
 10  last_review                     17294 non-null  object
 11  reviews_per_month               17294 non-null  float64
 12  calculated_host_listings_count  28451 non-null  int64
 13  availability_365                28451 non-null  int64
dtypes: float64(3), int64(6), object(5)
memory usage: 2.7+ MB
```

- 对于其他缺失值,我们可以可视化时处理
- 由于每个房子的name值都是不一样的,对name可视化时没有意义的,因此name不进行可视化

可视化hostID和hostname

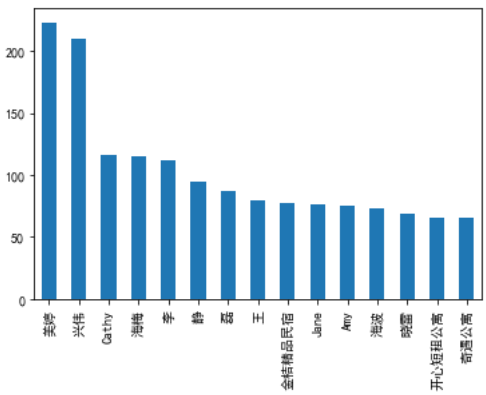
- hostID和hostName指的是短租房的id编号和短租房的名称

```
#由于hostID和hostname是相关联的,因此,对其中一个可视化即可
host_name = listings['host_name']
#进行频数统计
host_name_count = host_name.value_counts()
host_name_count
```

```
美婷         223
兴伟         210
Cathy      116
海梅         115
李          112
          ...
斓             1
允成             1
Chanys       1
钱多多            1
范范             1
Name: host_name, Length: 6627, dtype: int64
```

```
%matplotlib inline
#解决中文乱码问题!
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False
#取前15的数据,做柱状图可视化处理
host_name_count.head(15).plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x13d42c58>
```



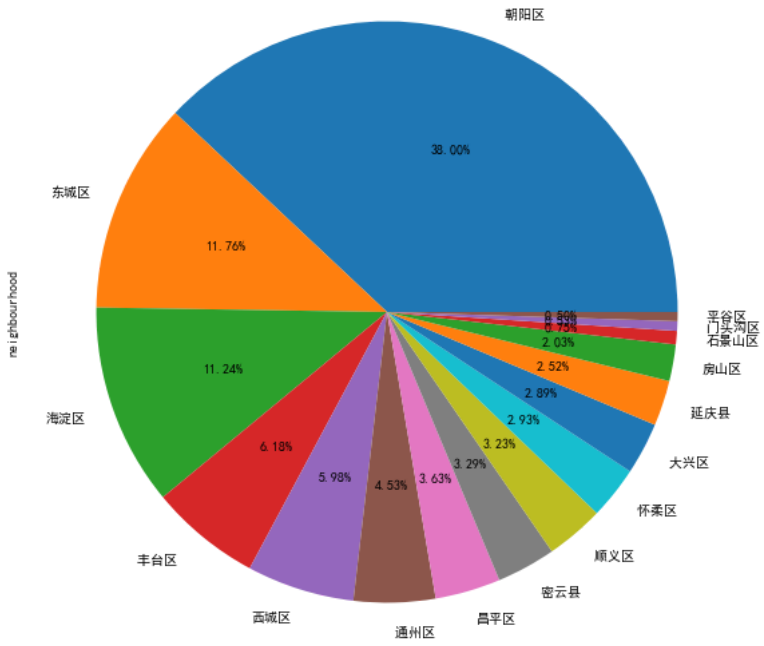
1.2 数据可视化

可视化neighbourhood

- 短租房所处的地区

```
#直接对neighbourhood做饼图统计
neighbourhood = listings['neighbourhood']
neighbourhood_count = neighbourhood.value_counts()
neighbourhood_count.plot.pie(figsize=(30,10),autopct='%.2f%%')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x13aa8aa8>
```



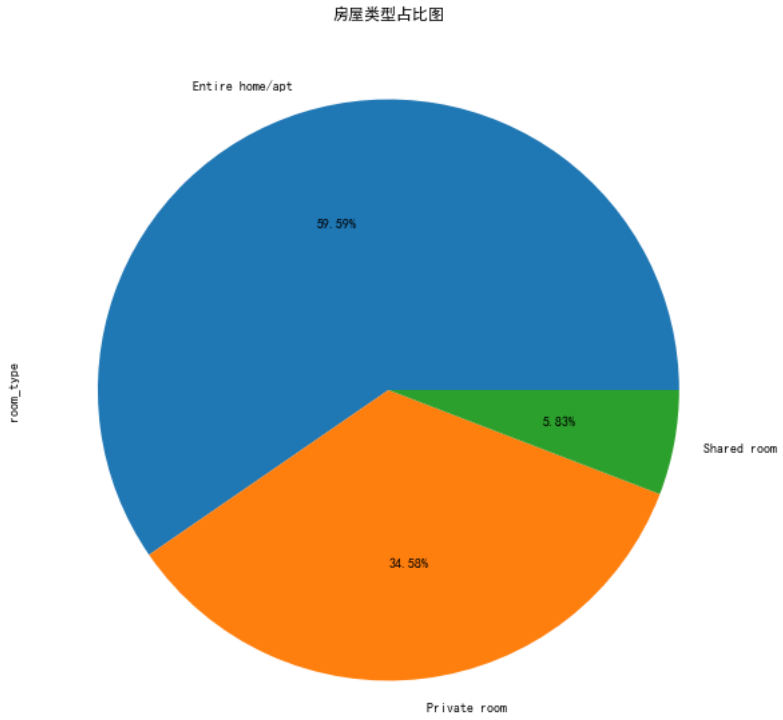
latitude和longitude是指房屋所在的经纬度坐标,没有可视化的必要

可视化room_type

- room_tyoe是指房屋类型

```
room_type = listings['room_type']
room_type_count = room_type.value_counts()
room_type_count.plot.pie(figsize=(30,10),autopct='%.2f%%')
plt.title('房屋类型占比图')
```

```
Text(0.5, 1.0, '房屋类型占比图')
```

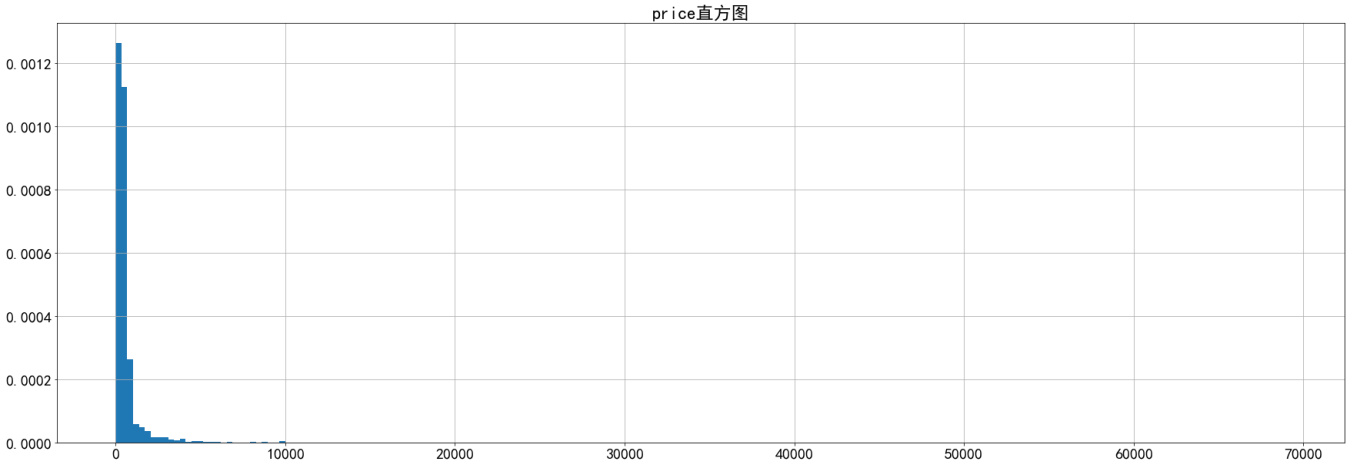


对price数据进行可视化

- price为当前的短租房租金价格
- 绘制price直方图

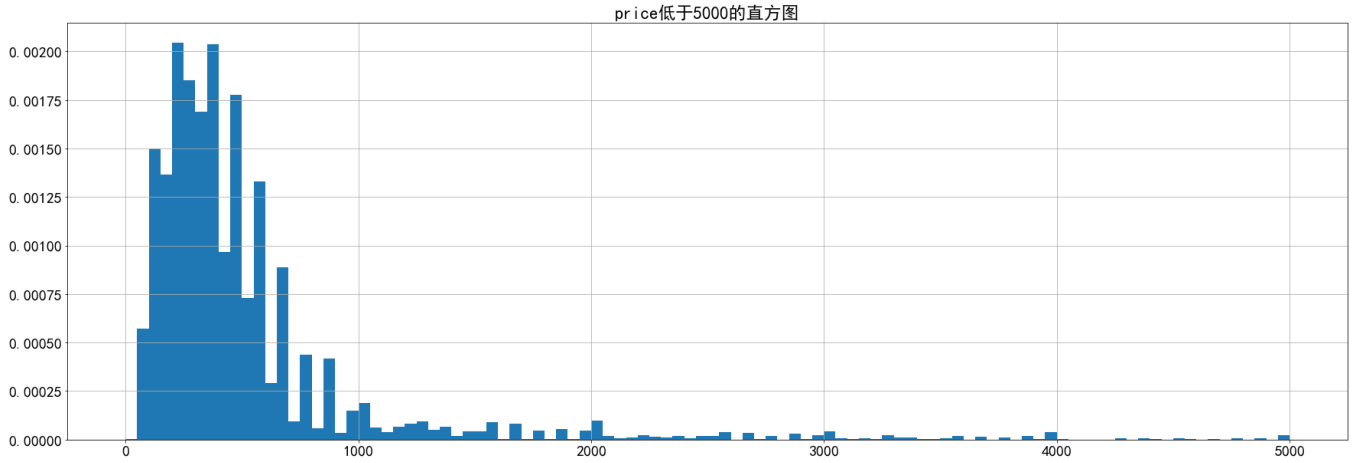
```
price = listings['price']
price.hist(xlabelsize=20,ylabelsize=20,figsize=(30,10),bins=200,density=True)
plt.title('price直方图',fontsize=23)
```

```
Text(0.5, 1.0, 'price直方图')
```



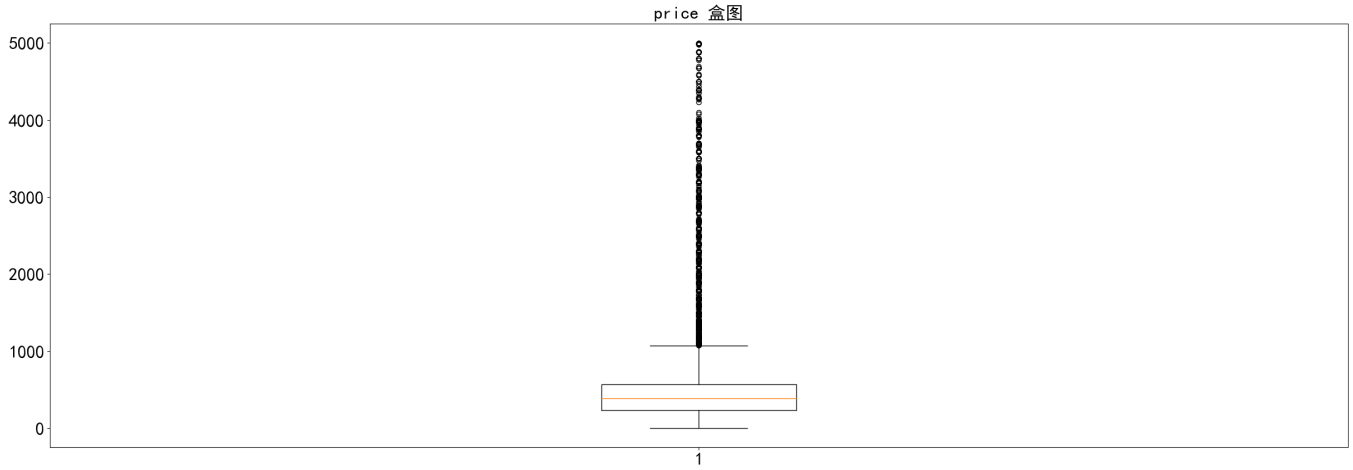

```
# 由上图可知,短租房价格一般在0到5000之间,因此,我们可以再检查0-5000之间的价格分布
price_litter = price[price < 5000]
price_litter.hist(xlabelsize=20,ylabelsize=20,figsize=(30,10),bins=100,density=True)
plt.title('price低于5000的直方图',fontsize=23)
```

```
Text(0.5, 1.0, 'price低于5000的直方图')
```



- 绘制price的盒图

```
figure,ax = plt.subplots(nrows=1,ncols=1,figsize=(30,10),dpi=100)
ax.boxplot(price_litter,notch=False,vert=True)
ax.tick_params(labelsize=23)
ax.set_title('price 盒图',fontsize=23)
plt.show()
```

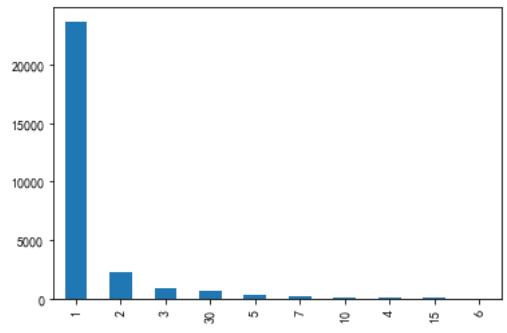


minimum_nights数据可视化

```
- minimum_nights为最少预定天数
```

```
minimum_nights = listings['minimum_nights']
minimum_nights_count = minimum_nights.value_counts()
minimum_nights_count.head(10).plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x15960850>
```

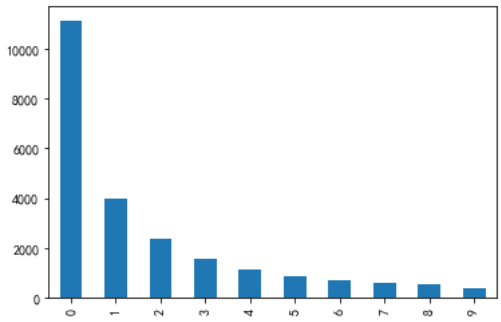


number_of_reviews数据可视化

- number_of_reviews为评论的数量

```
number_of_reviews = listings['number_of_reviews']
number_of_reviews_count = number_of_reviews.value_counts()
number_of_reviews_count.head(10).plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x15b177a8>
```

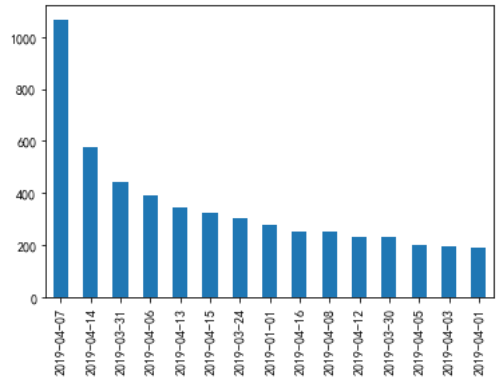


last_review数据可视化和缺失值处理

- last_reviews是最新评论时间,如果缺失,我们可以选择直接舍弃

```
last_review = listings['last_review'].dropna()
last_review_count = last_review.value_counts()
last_review_count.head(15).plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x15fcb5b0>
```

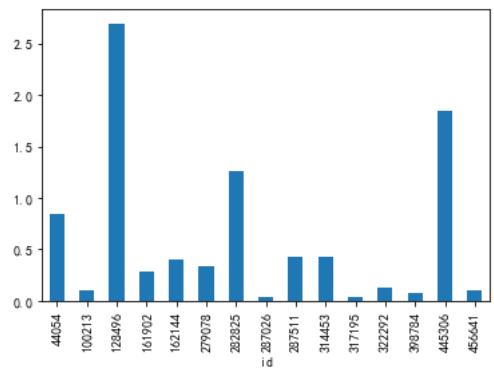


reviews_per_month数据可视化

- reviews_per_month是指平均每个月收到的评论数量,此处若有数据值缺失,我们可以将缺失值填充为0

```
reviews_per_month = listings['reviews_per_month']
reviews_per_month = reviews_per_month.fillna(0)
reviews_per_month_count = reviews_per_month.value_counts()
reviews_per_month.head(15).plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x16034820>
```

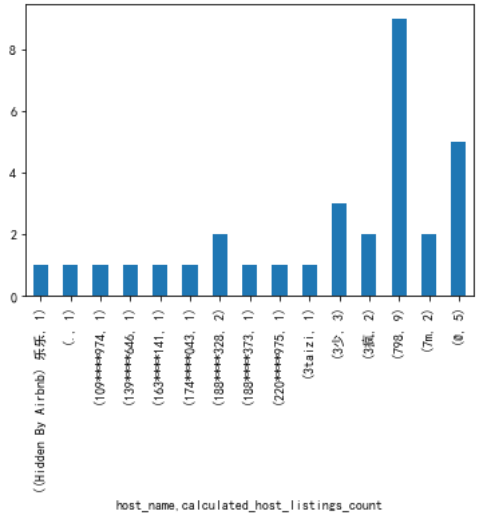


calculated_host_listings_count数据可视化

- calculated_host_listings_count是指每个房主拥有的房间数量

```
calculated_host_listings_count = listings.groupby('host_name')['calculated_host_listings_count']
calculated_host_listings_count_sum = calculated_host_listings_count.value_counts()
calculated_host_listings_count_sum.head(15).plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1650d0e8>
```

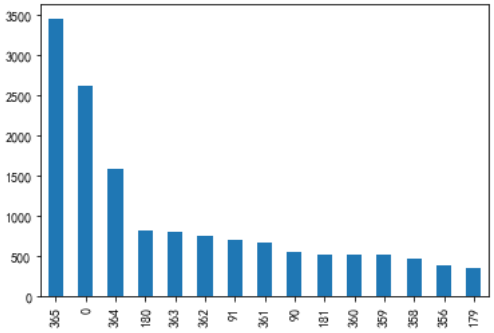


availability_365数据可视化

- availability_365是指可以预定天数

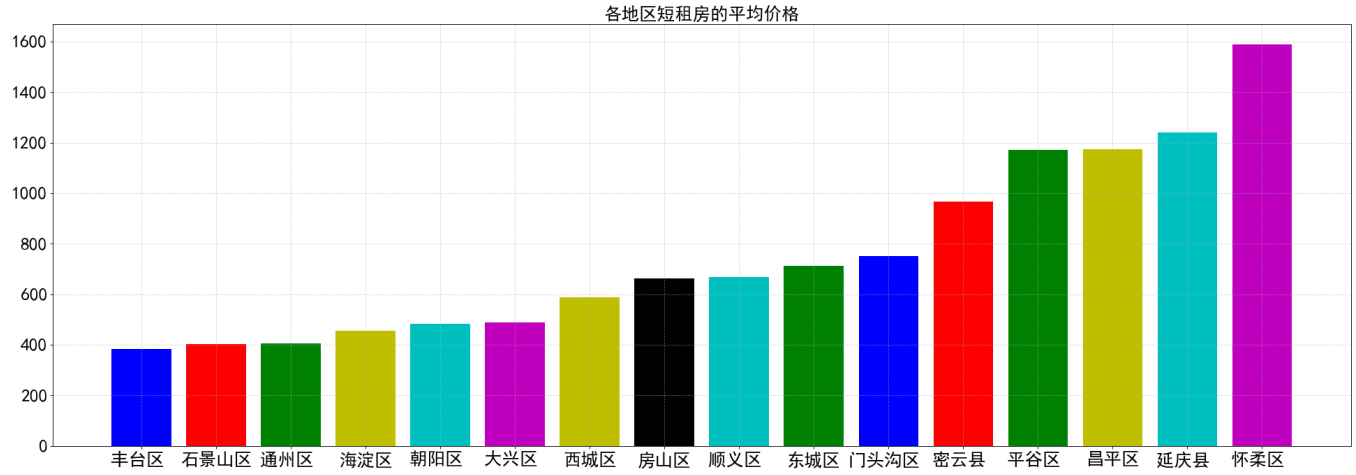
```
availability_365 = listings['availability_365']
availability_365_count = availability_365.value_counts()
availability_365_count.head(15).plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1655b1a8>
```



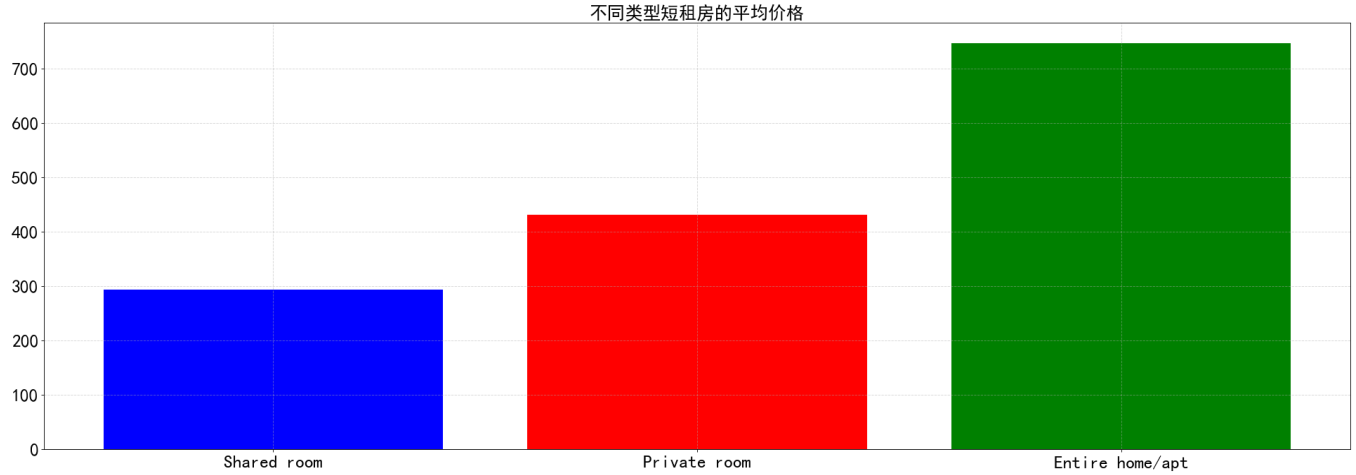
查看不同地区短租房的平均价格

```
plt.figure(figsize=(30,10),dpi=80)
neighbourhood_host_mean_price = listings.groupby('neighbourhood')['price'].mean()
data = neighbourhood_host_mean_price.sort_values()
x_index = data.index.tolist()
x_ticks = range(len(x_index))
counts = data.tolist()
plt.bar(x_ticks,counts,color=['b','r','g','y','c','m','y','k','c','g'])
plt.xticks(x_ticks,x_index,fontsize=23)
plt.yticks(fontsize=23)
plt.title("各地区短租房的平均价格",fontsize=23)
plt.grid(linestyle="--",alpha=0.5)
plt.show()
```



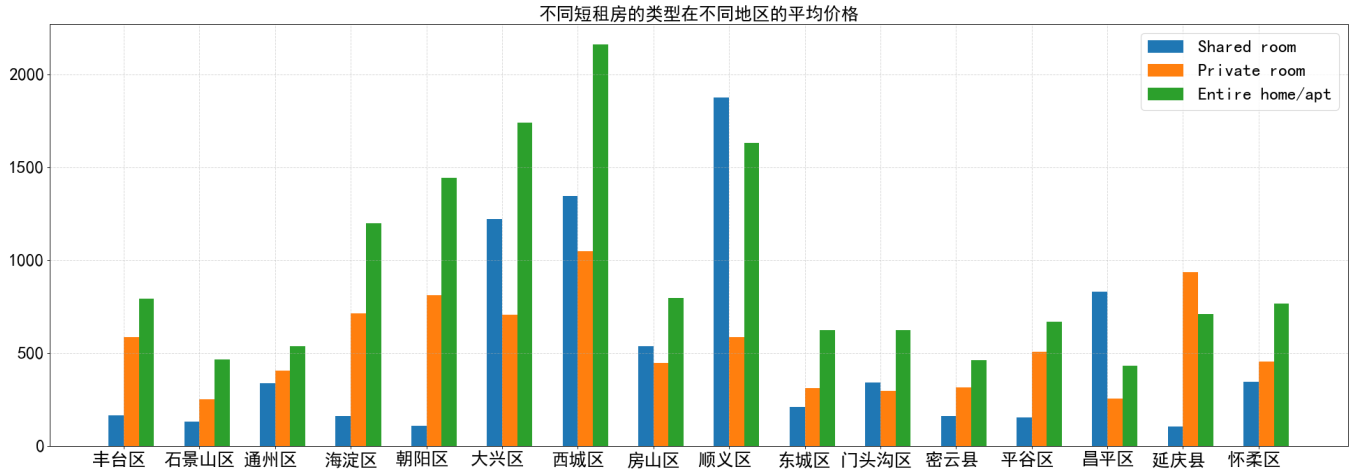
查看不同类型的短租房均价

```
plt.figure(figsize=(30,10),dpi=80)
room_type_host_mean_price = listings.groupby('room_type')['price'].mean()
data = room_type_host_mean_price.sort_values()
x_index = data.index.tolist()
x_ticks = range(len(x_index))
counts = data.tolist()
plt.bar(x_ticks,counts,color=['b','r','g'])
plt.xticks(x_ticks,x_index,fontsize=23)
plt.yticks(fontsize=23)
plt.title("不同类型短租房的平均价格",fontsize=23)
plt.grid(linestyle="--",alpha=0.5)
plt.show()
```



查看不同短租房的类型在不同地区的平均价格

```
plt.figure(figsize=(30,10),dpi=80)
neighbourhood_host_mean_price = listings.groupby('neighbourhood')['price'].mean()
data1 = neighbourhood_host_mean_price.sort_values()
neighbourhood_index = data1.index.tolist()
x_ticks = range(len(neighbourhood_index))
listings['room_type']
shared_room_count = listings[listings['room_type']=='Shared room'].groupby('neighbourhood')['price'].mean().tolist()
private_room_count = listings[listings['room_type']=='Private room'].groupby('neighbourhood')['price'].mean().tolist()
entire_home_count = listings[listings['room_type']=='Entire home/apt'].groupby('neighbourhood')['price'].mean().tolist()
plt.bar(x_ticks,shared_room_count,width=0.2,label='Shared room')
plt.bar([i+0.2 for i in x_ticks],private_room_count,width=0.2,label="Private room")
plt.bar([i+0.4 for i in x_ticks],entire_home_count,width=0.2,label="Entire home/apt")
#显示图例
plt.legend(fontsize=23)
#修改x刻度
plt.xticks([i+0.1 for i in x_ticks],neighbourhood_index,fontsize=23)
plt.yticks(fontsize=23)
#添加标题
plt.title("不同短租房的类型在不同地区的平均价格",fontsize=23)
plt.grid(linestyle="--",alpha=0.5)
#显示图像
plt.show()
```




```
listings = listings.dropna(how="any",subset = ["last_review"])
listings['reviews_per_month'].fillna(0, inplace=True)
listings.isnull().sum()
```

```
name                              0
host_id                           0
host_name                         0
neighbourhood                     0
latitude                          0
longitude                         0
room_type                         0
price                             0
minimum_nights                    0
number_of_reviews                 0
last_review                       0
reviews_per_month                 0
calculated_host_listings_count    0
availability_365                  0
dtype: int64
```

对全部不重复属性值进行One-hot编码，并生成编码和解码字典，方便规则的转换。然后将所有元组转换成向量表示

```
listings.head(3)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	name	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_re
id										
44054	Modern and Comfortable Living in CBD	192875	East Apartments	朝阳区	39.89503	116.45163	Entire home/apt	792	1	89
100213	The Great Wall Box Deluxe Suite A团园长城小院东院套房	527062	Joe	密云县	40.68434	117.17231	Private room	1201	1	2
128496	Heart of Beijing: House with View 2	467520	Cindy	东城区	39.93213	116.42200	Entire home/apt	389	3	259

```
listings['host_id'].value_counts()
```

```
54436429     100
20788084      56
196377840     49
119706958     42
156143513     41
             ...
251278673      1
27031448       1
42717087       1
187806826      1
207783937      1
Name: host_id, Length: 7567, dtype: int64
```

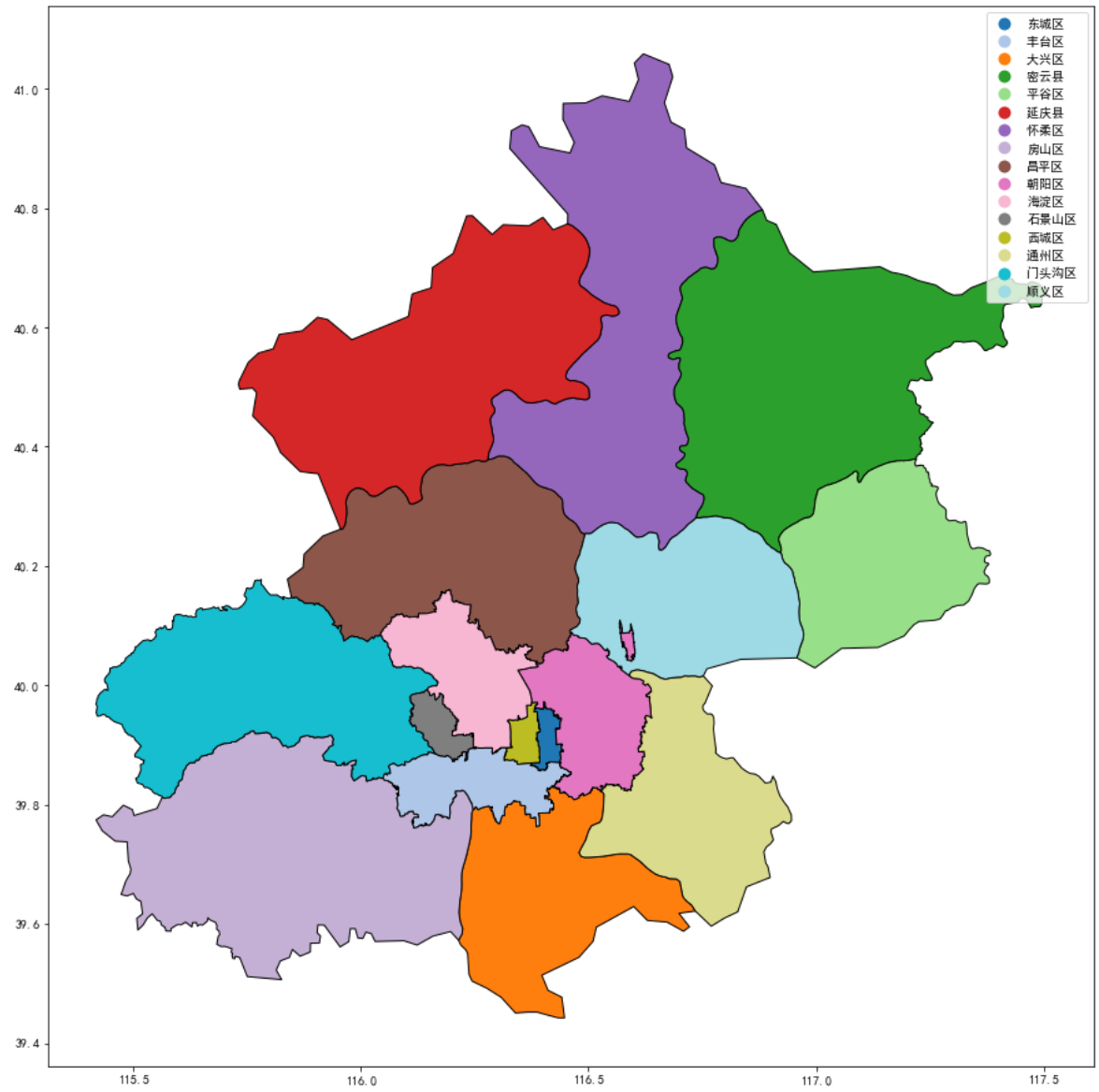
1.3地图可视化

```
import geopandas as gpd
from shapely.geometry import Point # 经纬度转换为点
import mapclassify
```

制作底图

```
data_map = gpd.GeoDataFrame.from_file('data/neighbourhoods.geojson')   #读取数据为geodataframe格式
data_map['neighbourhood'] = data_map['neighbourhood'].str.split('/').str[0]
data_map.plot('neighbourhood',cmap='tab20',legend=True,edgecolor='black',figsize=(15,20))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x19939fd0>
```



1.3.1房源标注

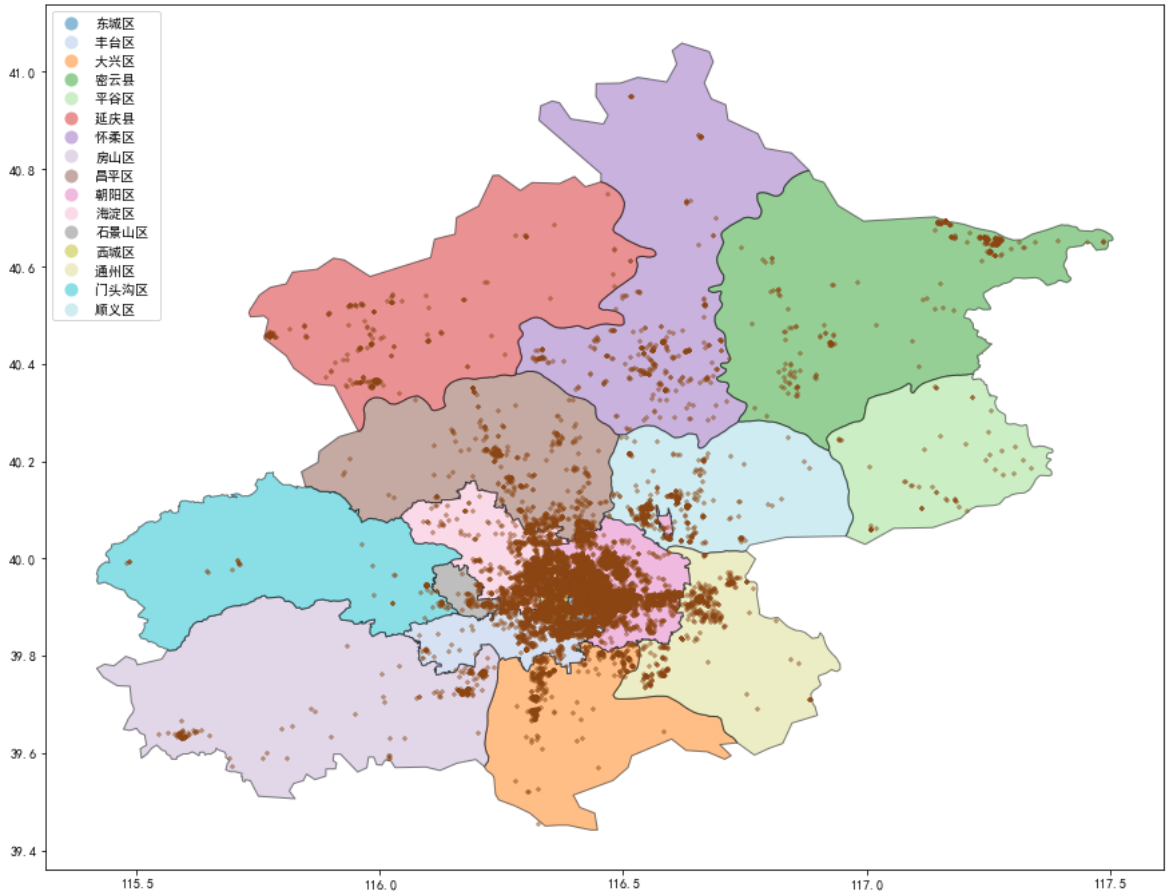
```
def data_to_gpd(data, label_longitude, label_latitude):
    data["geometry"] = list(zip(data[label_longitude],data[label_latitude]))
    data["geometry"] = data["geometry"].apply(Point)
    data = gpd.GeoDataFrame(data)
    del data[label_latitude]
    del data[label_longitude]
    return data

gpd_list = data_to_gpd(listings, "longitude", "latitude")
gpd_list.head(5)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	name	host_id	host_name	neighbourhood	room_type	price	minimum_nights	number_of_reviews	last_review	r
id										
44054	Modern and Comfortable Living in CBD	192875	East Apartments	朝阳区	Entire home/apt	792	1	89	2019-03-04	0
100213	The Great Wall Box Deluxe Suite A团园长城小院东院套房	527062	Joe	密云县	Private room	1201	1	2	2017-10-08	0
128496	Heart of Beijing: House with View 2	467520	Cindy	东城区	Entire home/apt	389	3	259	2019-02-05	2
161902	cozy studio in center of Beijing	707535	Robert	东城区	Entire home/apt	376	1	26	2016-12-03	0
162144	nice studio near subway, sleep 4	707535	Robert	朝阳区	Entire home/apt	537	1	37	2018-08-01	0

```
base =data_map.plot('neighbourhood',cmap='tab20',legend=True,edgecolor='black',figsize=(15,20),alpha=0.5)
gpd_list.plot(ax=base, color='saddlebrown', marker='+', markersize=10, alpha=0.5) #在底图上叠加房源数
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x19c4fa00>
```



1.3.2通过颜色深浅表示北京各区房源数量

首先计算各区的房源总数

```
house_num = gpd_list.groupby("neighbourhood").size() # Series格式
house_num = house_num.to_frame().reset_index() # 转换为dataframe格式
house_num.columns = ["neighbourhood", "count"] # 更改列名，方便操作
house_num
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	neighbourhood	count
0	东城区	2445
1	丰台区	997
2	大兴区	447
3	密云县	369
4	平谷区	51
5	延庆县	215
6	怀柔区	305
7	房山区	273
8	昌平区	532
9	朝阳区	7055
10	海淀区	2211
11	石景山区	115
12	西城区	1127
13	通州区	596
14	门头沟区	63
15	顺义区	493

将得到的house_num与neibourhood表合并

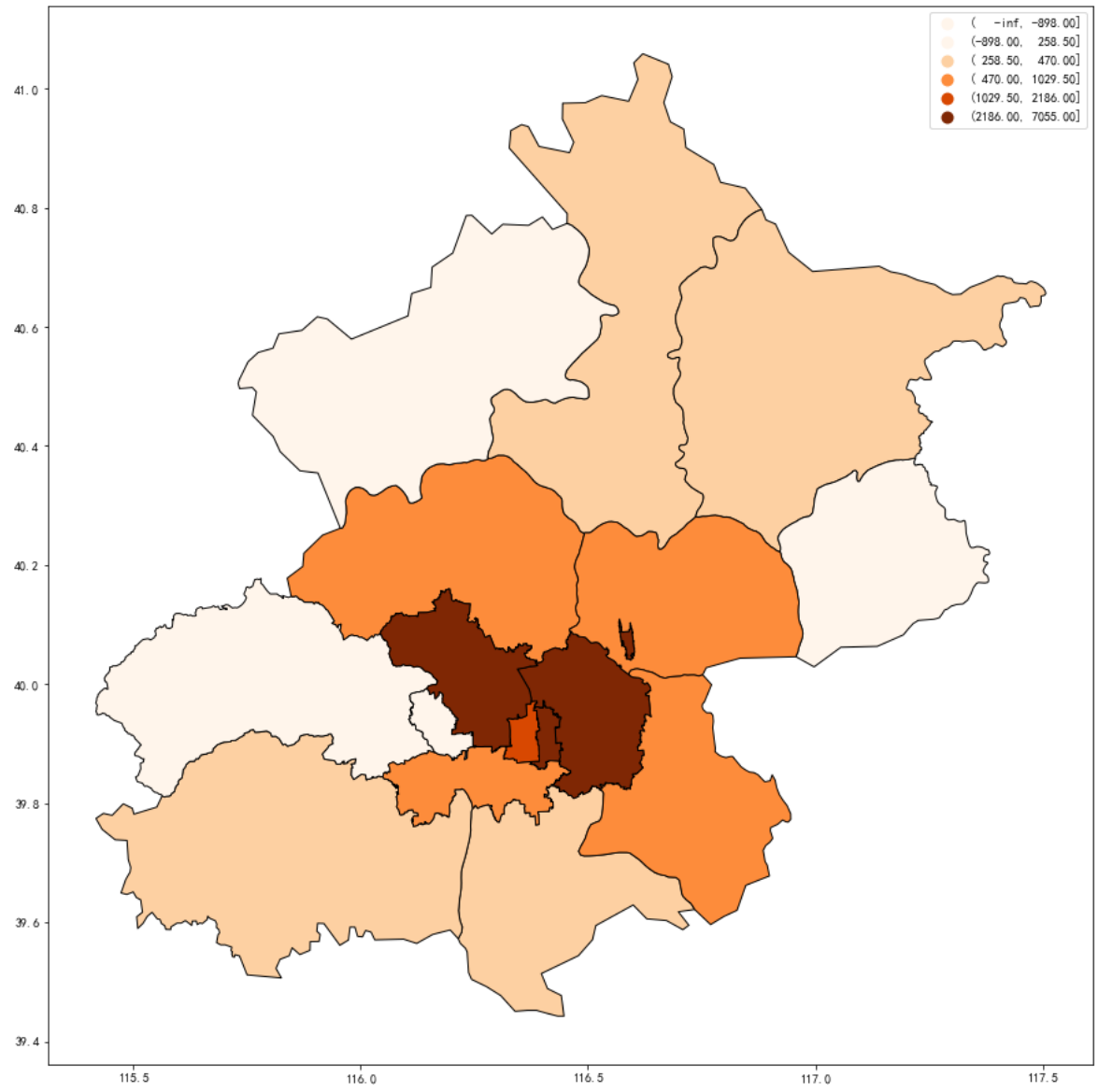
```
house_group = pd.merge(data_map, house_num, on="neighbourhood", how="left")
house_group
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	neighbourhood	neighbourhood_group	geometry	count
0	东城区	None	MULTIPOLYGON (((116.44231 39.90180, 116.44246 ...	2445
1	西城区	None	MULTIPOLYGON (((116.39155 39.89710, 116.39157 ...	1127
2	昌平区	None	MULTIPOLYGON (((116.04274 40.08406, 116.03750 ...	532
3	大兴区	None	MULTIPOLYGON (((116.73473 39.62160, 116.69884 ...	447
4	房山区	None	MULTIPOLYGON (((116.24662 39.79181, 116.24320 ...	273
5	怀柔区	None	MULTIPOLYGON (((116.27899 40.37974, 116.27943 ...	305
6	门头沟区	None	MULTIPOLYGON (((115.56297 39.81206, 115.56156 ...	63
7	密云县	None	MULTIPOLYGON (((116.88258 40.79685, 116.89160 ...	369
8	平谷区	None	MULTIPOLYGON (((117.38129 40.22541, 117.38304 ...	51
9	延庆县	None	MULTIPOLYGON (((116.27899 40.37974, 116.27895 ...	215
10	朝阳区	None	MULTIPOLYGON (((116.56870 40.10941, 116.56956 ...	7055
11	丰台区	None	MULTIPOLYGON (((116.43726 39.87080, 116.43910 ...	997
12	海淀区	None	MULTIPOLYGON (((116.32248 39.89564, 116.30838 ...	2211
13	顺义区	None	MULTIPOLYGON (((116.49032 40.25365, 116.49498 ...	493
14	通州区	None	MULTIPOLYGON (((116.58917 40.01649, 116.59197 ...	596
15	石景山区	None	MULTIPOLYGON (((116.16118 39.88551, 116.16068 ...	115

```
house_group.plot(column="count", cmap='Oranges', scheme="boxplot"
                           , edgecolor='black',legend=True, figsize=(15, 20))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x19c77940>
```



1.3.3通过颜色深浅表示北京各区房源平均价格

首先计算各区的房源平均价格

将得到的house_price与neighbourhood表合并

```
list(neighbourhood_host_mean_price)
```

```
[712.4085475194262,
 384.63139931740614,
 489.5419198055893,
 966.9625668449198,
 1172.2657342657342,
 1238.9860724233984,
 1589.7779111644659,
 663.0276816608997,
 1174.803675048356,
 484.395282146161,
 456.4610572411636,
 403.05164319248826,
 587.9870664315109,
 405.5480620155039,
 751.5855263157895,
 667.9717391304348]
```

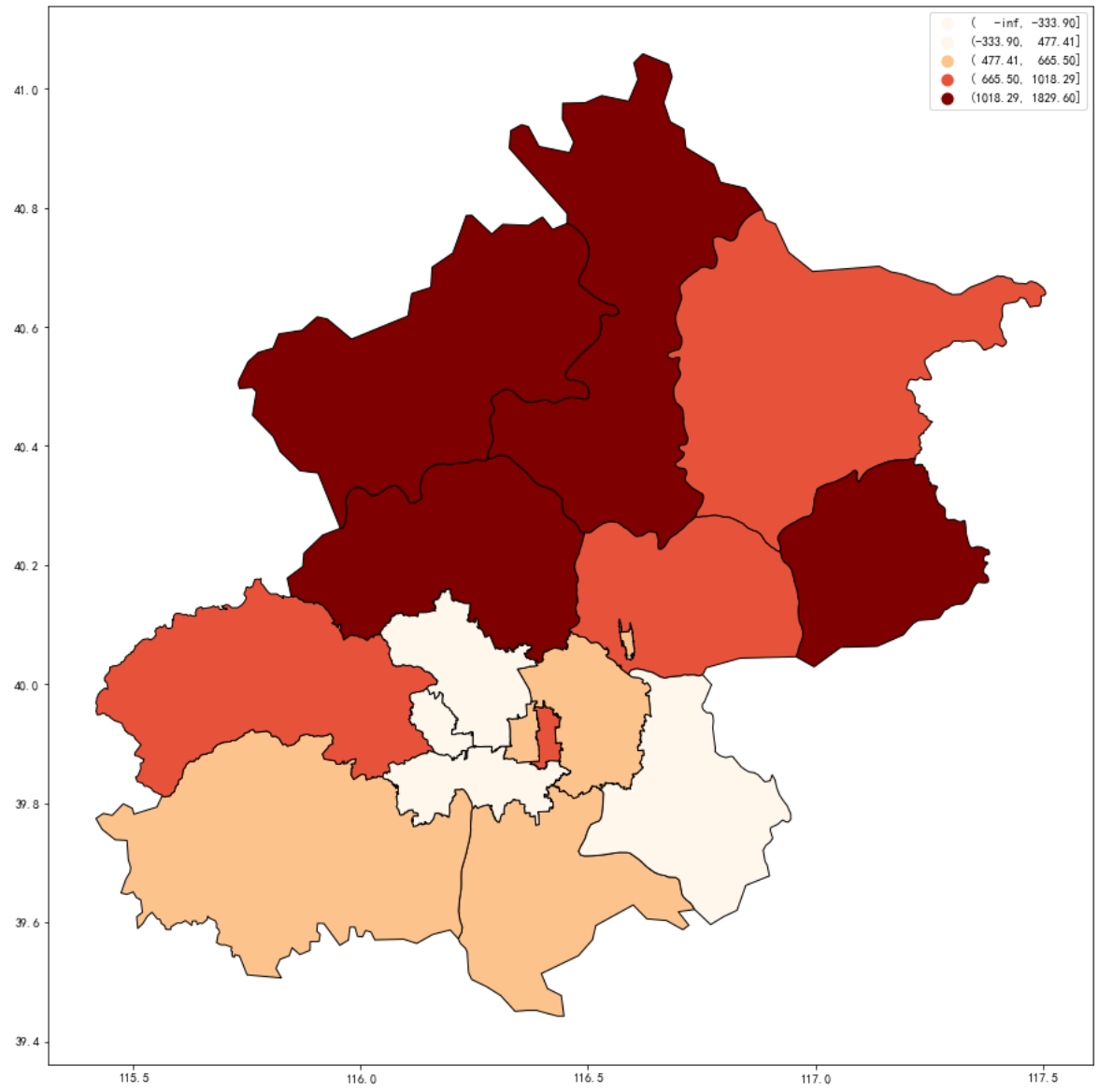
```
house_group_price = pd.merge(data_map,neighbourhood_host_mean_price, on="neighbourhood", how="left")
house_group_price
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	neighbourhood	neighbourhood_group	geometry	price
0	东城区	None	MULTIPOLYGON (((116.44231 39.90180, 116.44246 ...	712.408548
1	西城区	None	MULTIPOLYGON (((116.39155 39.89710, 116.39157 ...	587.987066
2	昌平区	None	MULTIPOLYGON (((116.04274 40.08406, 116.03750 ...	1174.803675
3	大兴区	None	MULTIPOLYGON (((116.73473 39.62160, 116.69884 ...	489.541920
4	房山区	None	MULTIPOLYGON (((116.24662 39.79181, 116.24320 ...	663.027682
5	怀柔区	None	MULTIPOLYGON (((116.27899 40.37974, 116.27943 ...	1589.777911
6	门头沟区	None	MULTIPOLYGON (((115.56297 39.81206, 115.56156 ...	751.585526
7	密云县	None	MULTIPOLYGON (((116.88258 40.79685, 116.89160 ...	966.962567
8	平谷区	None	MULTIPOLYGON (((117.38129 40.22541, 117.38304 ...	1172.265734
9	延庆县	None	MULTIPOLYGON (((116.27899 40.37974, 116.27895 ...	1238.986072
10	朝阳区	None	MULTIPOLYGON (((116.56870 40.10941, 116.56956 ...	484.395282
11	丰台区	None	MULTIPOLYGON (((116.43726 39.87080, 116.43910 ...	384.631399
12	海淀区	None	MULTIPOLYGON (((116.32248 39.89564, 116.30838 ...	456.461057
13	顺义区	None	MULTIPOLYGON (((116.49032 40.25365, 116.49498 ...	667.971739
14	通州区	None	MULTIPOLYGON (((116.58917 40.01649, 116.59197 ...	405.548062
15	石景山区	None	MULTIPOLYGON (((116.16118 39.88551, 116.16068 ...	403.051643

```
house_group_price.plot(column="price", cmap='OrRd', scheme="boxplot"
                            , edgecolor='black',legend=True, figsize=(15, 20))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x19a1a3a0>
```



二、频繁项集和关联规则挖掘

2.1提取数据集中可分析属性

name属性相当于标题，无需考虑；host_id属于编号，无需考虑；latitude和longitude是房子的经纬度，实际租房时并不需要考虑该属性; last_review、number_of reviews评论时间和数量属性，无需考虑。

```
## 创建需要分析的数据列表
SubDF = listings[['host_name','neighbourhood','room_type','price','reviews_per_month','calculated_host_listings_count','availability_365']]
```

```
## 删除price值小于等于0的元组
SubDF = SubDF[SubDF['price']>0]
```

```
SubDF.head(3)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	host_name	neighbourhood	room_type	price	reviews_per_month	calculated_host_listings_count	availability_365
id							
44054	East Apartments	朝阳区	Entire home/apt	792	0.85	9	341
100213	Joe	密云县	Private room	1201	0.10	4	0
128496	Cindy	东城区	Entire home/apt	389	2.70	1	93

对数据进行处理转换成可以进行关联规则挖掘的形式，对price,number_of_reviews,availability_365属性数据进行分层

```
SubDF['price']=pd.qcut(SubDF['price'],7)
def normalizef(x):
    x['price'] = "price_"+str(x['price'])
    x['reviews_per_month'] = 'reviewsPre_'+str(x['reviews_per_month'])
    x['calculated_host_listings_count'] = 'hostCount_'+str(x['calculated_host_listings_count'])
    s = x['availability_365']
    if (s > 30*11) and (s <= 365):
        s = 'availability_'+'11-12m'
    elif s > 30*10 and s <= 30*11:
        s = 'availability_'+'10-11m'
    elif s >30*9 and s<=30*10:
        s = 'availability_'+'9-10m'
    elif s >30*8 and s<=30*9:
        s = 'availability_'+'8-9m'
    elif s >30*7 and s<=30*8:
        s = 'availability_'+'7-8m'
    elif s >30*6 and s<=30*7:
        s = 'availability_'+'6-7m'
    elif s >30*5 and s<=30*6:
        s = 'availability_'+'5-6m'
    elif s >30*4 and s<=30*5:
        s = 'availability_'+'4-5m'
    elif s >30*3 and s<=30*4:
        s = 'availability_'+'3-4m'
    elif s >30*2 and s<=30*3:
        s = 'availability_'+'2-3m'
    elif s >30*1 and s<=30*2:
        s = 'availability_'+'1-2m'
    elif s >10*2 and s<=30:
        s = 'availability_'+'20-30d'
    elif s >10*1 and s<=10*2:
        s = 'availability_'+'10-20d'
    elif s >5 and s<=10:
        s = 'availability_'+'5-10d'
    elif s<=5:
        s = 'availability_'+'0-5d'
    x['availability_365'] = s
    return x
SubDF = SubDF.apply(normalizef, axis=1)
```

```
SubDF
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	host_name	neighbourhood	room_type	price	reviews_per_month	calculated_host_listings_count	availability_365
id							
44054	East Apartments	朝阳区	Entire home/apt	price_(678.0, 68983.0]	reviewsPre_0.85	hostCount_9	availability_11-12m
100213	Joe	密云县	Private room	price_(678.0, 68983.0]	reviewsPre_0.1	hostCount_4	availability_0-5d
128496	Cindy	东城区	Entire home/apt	price_(329.0, 396.0]	reviewsPre_2.7	hostCount_1	availability_3-4m
161902	Robert	东城区	Entire home/apt	price_(329.0, 396.0]	reviewsPre_0.28	hostCount_5	availability_9-10m
162144	Robert	朝阳区	Entire home/apt	price_(497.0, 678.0]	reviewsPre_0.4	hostCount_5	availability_11-12m
...	...	...	...	...	...	...	...
33889408	小田	朝阳区	Entire home/apt	price_(396.0, 497.0]	reviewsPre_1.0	hostCount_9	availability_10-20d
33890728	小田	朝阳区	Entire home/apt	price_(396.0, 497.0]	reviewsPre_1.0	hostCount_9	availability_10-20d
33891613	小田	朝阳区	Entire home/apt	price_(396.0, 497.0]	reviewsPre_1.0	hostCount_9	availability_5-10d
33892088	小田	朝阳区	Entire home/apt	price_(396.0, 497.0]	reviewsPre_1.0	hostCount_9	availability_10-20d
33925874	向鹏	石景山区	Entire home/apt	price_(329.0, 396.0]	reviewsPre_1.0	hostCount_2	availability_11-12m

17293 rows × 7 columns

对全部不重复属性值进行One-hot编码，并生成编码和解码字典，方便规则的转换。然后将所有元组转换成向量表示

```
strSet = SubDF['host_name'].unique().tolist()+SubDF['neighbourhood'].unique().tolist()+SubDF['room_type'].unique().tolist()+SubDF['price'].unique().tolist()+SubDF['reviews_per_month'].unique().tolist()+SubDF['calculated_host_listings_count'].unique().tolist()+SubDF['availability_365'].unique().tolist()
```

使用FP-growth 进行关联规则挖掘

```
import orangecontrib.associate.fpgrowth as oaf  #进行关联规则分析的包

strEncode = dict(zip(strSet,range(len(strSet)))) #编码字典
strDecode = dict(zip(strEncode.values(), strEncode.keys()))  #解码字典

def toNum(x):
    x['host_name'] = strEncode[x['host_name']]
    x['neighbourhood'] = strEncode[x['neighbourhood']]
    x['room_type'] = strEncode[x['room_type']]
    x['price'] = strEncode[x['price']]
   # x['minimum_nights'] =strEncode[x['minimum_nights']]
#     x['number_of_reviews'] = strEncode[x['number_of_reviews']]
    x['reviews_per_month'] = strEncode[x['reviews_per_month']]
    x['calculated_host_listings_count'] =strEncode[x['calculated_host_listings_count']]
    x['availability_365'] = strEncode[x['availability_365']]
    return x

# 将全部元组转化成向量表示
AnalysisDF = SubDF.apply(toNum, axis=1)
```

2.2关联规则挖掘

在支持度为0.03和置信度为0.7的条件下，进行关联规则的挖掘

```
itemsets = dict(oaf.frequent_itemsets(AnalysisDF.values.tolist(),  .03))
rules = oaf.association_rules(itemsets, .7)   #设置置信度
rules = list(rules)
print(len(rules))
```

```
19
```

```
# 查看频繁项集
itemsets
```

```
{frozenset({4963}): 10634,
 frozenset({4947}): 7055,
 frozenset({4947, 4963}): 4137,
 frozenset({5803}): 7053,
 frozenset({4947, 5803}): 2757,
 frozenset({4963, 5803}): 4663,
 frozenset({4947, 4963, 5803}): 1743,
 frozenset({4964}): 5714,
 frozenset({4947, 4964}): 2432,
 frozenset({4964, 5803}): 1957,
 frozenset({4947, 4964, 5803}): 775,
 frozenset({5749}): 3935,
 frozenset({4947, 5749}): 1615,
 frozenset({4963, 5749}): 2176,
 frozenset({4947, 4963, 5749}): 750,
 frozenset({4964, 5749}): 1564,
 frozenset({4947, 4964, 5749}): 767,
 frozenset({5749, 5803}): 929,
 frozenset({4963, 5749, 5803}): 546,
 frozenset({5807}): 2915,
 frozenset({4964, 5807}): 1042,
 frozenset({4947, 5807}): 1185,
 frozenset({4963, 5807}): 1754,
 frozenset({4947, 4963, 5807}): 650,
 frozenset({5749, 5807}): 727,
 frozenset({4971}): 2599,
 frozenset({4963, 4971}): 555,
 frozenset({4964, 4971}): 1964,
 frozenset({4971, 5803}): 845,
 frozenset({4964, 4971, 5803}): 623,
 frozenset({4947, 4971}): 1205,
 frozenset({4947, 4964, 4971}): 1021,
 frozenset({4971, 5749}): 798,
 frozenset({4964, 4971, 5749}): 554,
 frozenset({4969}): 2557,
 frozenset({4947, 4969}): 1188,
 frozenset({4969, 5803}): 1163,
 frozenset({4947, 4969, 5803}): 561,
 frozenset({4963, 4969}): 2336,
 frozenset({4947, 4963, 4969}): 1125,
 frozenset({4963, 4969, 5803}): 1082,
 frozenset({4947, 4963, 4969, 5803}): 532,
 frozenset({4970}): 2505,
 frozenset({4963, 4970}): 1453,
 frozenset({4964, 4970}): 1019,
 frozenset({4970, 5803}): 997,
 frozenset({4963, 4970, 5803}): 581,
 frozenset({4947, 4970}): 891,
 frozenset({4970, 5749}): 618,
 frozenset({4972}): 2491,
 frozenset({4947, 4972}): 1148,
 frozenset({4972, 5803}): 850,
 frozenset({4964, 4972}): 1561,
 frozenset({4947, 4964, 4972}): 693,
 frozenset({4972, 5749}): 811,
 frozenset({4964, 4972, 5749}): 604,
 frozenset({4966}): 2459,
 frozenset({4947, 4966}): 631,
 frozenset({4966, 5803}): 1166,
 frozenset({4963, 4966}): 2108,
 frozenset({4947, 4963, 4966}): 583,
 frozenset({4963, 4966, 5803}): 1008,
 frozenset({4949}): 2444,
 frozenset({4949, 4963}): 1707,
 frozenset({4949, 4964}): 644,
 frozenset({4949, 4966}): 615,
 frozenset({4949, 5803}): 850,
 frozenset({4949, 4963, 5803}): 626,
 frozenset({4949, 5807}): 535,
 frozenset({4968}): 2352,
 frozenset({4947, 4968}): 1017,
 frozenset({4968, 5803}): 1044,
 frozenset({4963, 4968}): 2079,
 frozenset({4947, 4963, 4968}): 948,
 frozenset({4963, 4968, 5803}): 920,
 frozenset({4967}): 2330,
 frozenset({4947, 4967}): 975,
 frozenset({4963, 4967}): 1978,
 frozenset({4947, 4963, 4967}): 859,
 frozenset({4967, 5803}): 988,
```

```
 frozenset({4963, 4967, 5803}): 845,
 frozenset({4951}): 2211,
 frozenset({4951, 5803}): 984,
 frozenset({4951, 4964}): 951,
 frozenset({4951, 4963}): 1075,
 frozenset({4951, 4963, 5803}): 562,
 frozenset({5752}): 2038,
 frozenset({4947, 5752}): 881,
 frozenset({4963, 5752}): 1076,
 frozenset({4964, 5752}): 869,
 frozenset({5752, 5803}): 701,
 frozenset({5810}): 2016,
 frozenset({4947, 5810}): 790,
 frozenset({4963, 5810}): 1311,
 frozenset({4964, 5810}): 602,
 frozenset({5804}): 1589,
 frozenset({4947, 5804}): 725,
 frozenset({4964, 5804}): 753,
 frozenset({4963, 5804}): 718,
 frozenset({5749, 5804}): 842,
 frozenset({5753}): 1531,
 frozenset({4964, 5753}): 587,
 frozenset({4947, 5753}): 653,
 frozenset({4963, 5753}): 866,
 frozenset({5753, 5803}): 611,
 frozenset({5748}): 1349,
 frozenset({4947, 5748}): 542,
 frozenset({5748, 5803}): 547,
 frozenset({4963, 5748}): 781,
 frozenset({4950}): 1127,
 frozenset({4950, 4963}): 716,
 frozenset({5051}): 1031,
 frozenset({4963, 5051}): 657,
 frozenset({4957}): 997,
 frozenset({4957, 4963}): 692,
 frozenset({5750}): 979,
 frozenset({4963, 5750}): 561,
 frozenset({4965}): 945,
 frozenset({4965, 4972}): 805,
 frozenset({5763}): 902,
 frozenset({4963, 5763}): 588,
 frozenset({5751}): 721,
 frozenset({5813}): 690,
 frozenset({4956}): 596,
 frozenset({5758}): 587,
 frozenset({5805}): 559,
 frozenset({5762}): 533,
 frozenset({4955}): 532}
```

打印挖掘出的规则，并使用Lift、Kulc和IR指标，评价关联规则

```
 def dealRules(rules,strDecode):
    returnRules = []
    for i in rules:
        temList = []
        pb = i[3]/i[6]
        pa = i[4]
        pab = i[3]*pa
        temStr = '( ';
        for j in i[0]:   #处理第一个frozenset
            temStr = temStr+strDecode[j]+', '
        temStr = temStr[:-2]+' )'
        temStr = temStr + ' ==> ('
        for j in i[1]:
            temStr = temStr+strDecode[j] + ', '
        temStr = temStr[:-2]+' )'
        temList.append(temStr); temList.append(i[2]); temList.append(i[3]); temList.append(i[6]);
#         temList.append(0.5*(i[3]+i[3]/i[5]))
        temList.append(0.5*(pab/pa+pab/pb))
        temList.append(abs(pa-pb)/(pa+pb-pab))
        temList.append(pa)
        temList.append(pb)
        temList.append(pab)
        returnRules.append(temList)
    return pd.DataFrame(returnRules,columns=('规则','支持度','置信度','Lift','Kulc','IR','SupA','SupB','SupAB'))

#printRules = dealRules(rules,strDecode)
```

```
pd.set_option('display.max_colwidth',None)
result = list(oaf.rules_stats(rules, itemsets, len(AnalysisDF)))
StateDF = dealRules(result,strDecode)
```

```
StateDF.sort_values(by='Lift' , ascending=False)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	规则	支持度	置信度	Lift	Kulc	IR	SupA	SupB	SupAB
18	(Shared room) ==> (price_(46.999, 168.0])	805	0.851852	5.913719	0.587508	0.587609	0.054646	0.144047	0.046551
2	(朝阳区 , price_(168.0, 248.0]) ==> (Private room)	1021	0.847303	2.564300	0.512993	0.764496	0.069681	0.330423	0.059041
13	(price_(168.0, 248.0]) ==> (Private room)	1964	0.755675	2.286995	0.549696	0.490628	0.150292	0.330423	0.113572
5	(price_(46.999, 168.0], hostCount_1) ==> (Private room)	604	0.744760	2.253960	0.425232	0.828070	0.046898	0.330423	0.034927
1	(price_(168.0, 248.0], availability_11-12m) ==> (Private room)	623	0.737278	2.231318	0.423154	0.820249	0.048864	0.330423	0.036026
0	(price_(396.0, 497.0], 朝阳区 , availability_11-12m) ==> (Entire home/apt)	532	0.948307	1.542135	0.499167	0.944668	0.032441	0.614931	0.030764
3	(price_(396.0, 497.0], 朝阳区) ==> (Entire home/apt)	1125	0.946970	1.539961	0.526381	0.883051	0.068698	0.614931	0.065055
9	(price_(497.0, 678.0], 朝阳区) ==> (Entire home/apt)	948	0.932153	1.515867	0.510651	0.898533	0.058810	0.614931	0.054820
4	(price_(396.0, 497.0], availability_11-12m) ==> (Entire home/apt)	1082	0.930353	1.512938	0.516051	0.883901	0.067253	0.614931	0.062569
6	(朝阳区 , price_(678.0, 68983.0]) ==> (Entire home/apt)	583	0.923930	1.502494	0.489377	0.936435	0.036489	0.614931	0.033713
14	(price_(396.0, 497.0]) ==> (Entire home/apt)	2336	0.913571	1.485648	0.566622	0.744081	0.147863	0.614931	0.135084
16	(price_(497.0, 678.0]) ==> (Entire home/apt)	2079	0.883929	1.437444	0.539717	0.759329	0.136009	0.614931	0.120222
10	(price_(497.0, 678.0], availability_11-12m) ==> (Entire home/apt)	920	0.881226	1.433049	0.483871	0.891430	0.060371	0.614931	0.053201
11	(朝阳区 , price_(329.0, 396.0]) ==> (Entire home/apt)	859	0.881026	1.432723	0.480902	0.898512	0.056381	0.614931	0.049673
7	(availability_11-12m, price_(678.0, 68983.0]) ==> (Entire home/apt)	1008	0.864494	1.405839	0.479642	0.877317	0.067426	0.614931	0.058289
15	(price_(678.0, 68983.0]) ==> (Entire home/apt)	2108	0.857259	1.394074	0.527746	0.744197	0.142196	0.614931	0.121899
12	(availability_11-12m, price_(329.0, 396.0]) ==> (Entire home/apt)	845	0.855263	1.390828	0.467363	0.895054	0.057133	0.614931	0.048864
17	(price_(329.0, 396.0]) ==> (Entire home/apt)	1978	0.848927	1.380524	0.517467	0.755871	0.134737	0.614931	0.114382
8	(availability_11-12m, 东城区) ==> (Entire home/apt)	626	0.736471	1.197648	0.397669	0.901087	0.049153	0.614931	0.036200

由上数据可以得到关联规则：

- (Shared room) ==> (price_(46.999, 168.0])， 其置信度为85.1%说明，合租房间价格在46.9至168元以内的概率为85.1%，同时其提升度5.91说明该条规则是很意义、普遍存在的，Kulc指标为0.58说明这条规则是接近中性的，同时IR为0.59可以认为这个规则是A、B是一般不平衡的，合租房间价格很可能在46.9-168之间，但是价格在这个区间的房源有很多种，不止包括合租；
- (朝阳区 , price_(168.0, 248.0]) ==> (Private room)，这两条规则置信度分别为0.847，可以说明在朝阳区价格在168-248之间的房源，84%以上都是单间，而他们的Lift值为2.56，说明这个规则也是有意义、较为普遍的，而0.76的IR值说明，这条规则是很不平衡的，即表明在朝阳区的单间房源还有更高或者更低的价格，依据常识更高的可能性会更大，这也给提醒租客在朝阳区租单人间时，最低价格很可能在168左右。
- (price_(168.0, 248.0], availability_11-12m) ==> (Private room) 该规则则表明在北京价格在168-248之间，且租住在11-12个月之间的的房源很可能是单人间，且很不平衡。
- (price_(396.0, 497.0], 朝阳区 , availability_11-12m) ==> (Entire home/apt)， 在朝阳区价格在396-497之间且出租11-12个月的房子，有94.8%的概率是整租公寓，这条规则Lift为1.54说明规则有意义，并且0.944的IR值表明该规则A、B很不平衡。

由于大部分关联规则都指向房源类型和评论数量，分析价值不大，因此去掉属性'room_type'，将支持度设为0.02，置信度设为0.6，挖掘关联规则

```
AnalysisDF2=AnalysisDF.drop(['room_type'],axis=1)
itemsets2 = dict(oaf.frequent_itemsets(AnalysisDF2.values.tolist(),  .01))
rules2 = oaf.association_rules(itemsets2, .6)   #设置置信度
rules2 = list(rules2)
result2 = list(oaf.rules_stats(rules2, itemsets2, len(AnalysisDF2)))
StateDF2 = dealRules(result2,strDecode)
StateDF2.sort_values(by='Lift' , ascending=False)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	规则	支持度	置信度	Lift	Kulc	IR	SupA	SupB	SupAB
1	(怀柔区) ==> (price_(678.0, 68983.0])	183	0.600000	4.219520	0.33721	0.83456	0.017637	0.142196	0.010582
0	(price_(497.0, 678.0], 海淀区) ==> (availability_11-12m)	177	0.606164	1.486233	0.31563	0.94322	0.016885	0.407853	0.010235

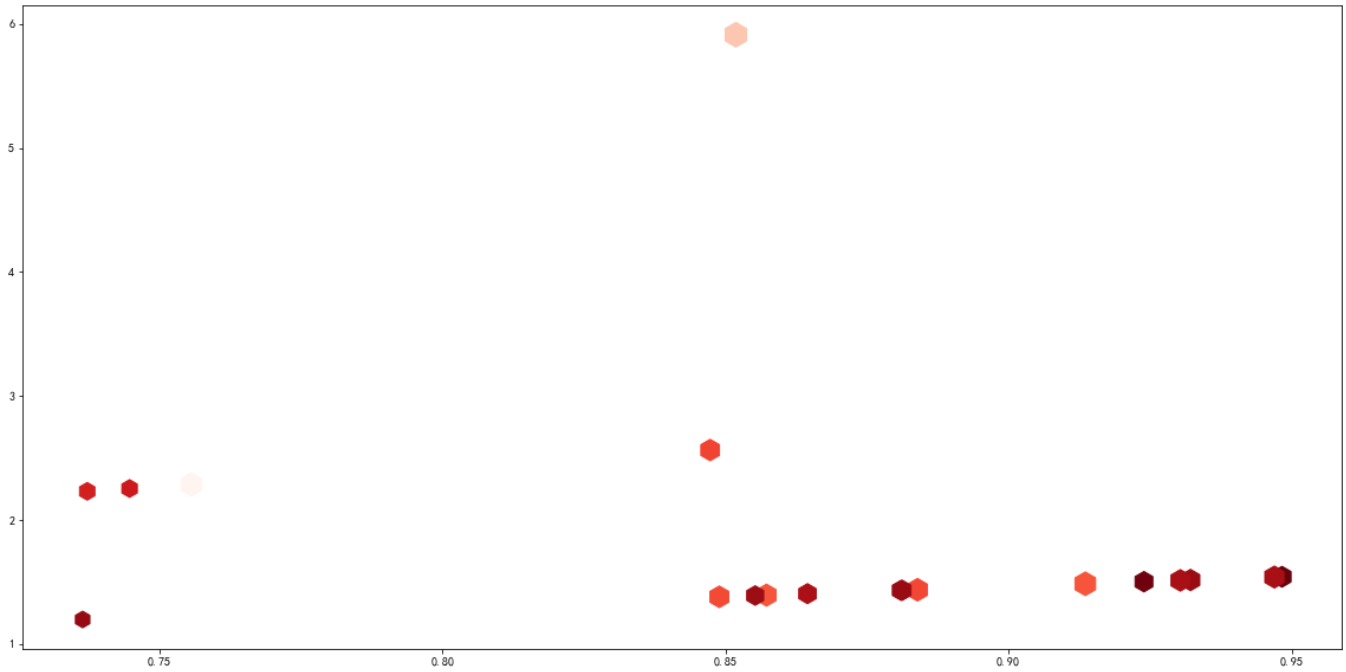
- (怀柔区) ==> (price_(678.0, 68983.0]) 该规则说明怀柔区的房子价格有60%的可能在678以上，且Lift为4.22，说明该规则有价值
- (price_(497.0, 678.0], 海淀区) ==> (availability_11-12m) 在海淀区且价格为497-678之间时，有61%的可能出租11-12个月，且规则很不平衡，意味着在海淀区高租金的情况下，很可能会出租很久。

2.3关联规则可视化展示

按置信度排降序排列，对规则进行可视化展示。其中x轴代表置信度，y轴代表提升度Lift，点的颜色代表IR的值，颜色越浅表示IR值越小,点的大小代表Kulc值得大小，值越大点越大

```
tempDF = StateDF.sort_values(by='置信度' , ascending=False)
colors = tempDF['IR']
area = np.pi * (tempDF['Kulc']*20)**2
# 画散点图
plt.rcParams['figure.figsize'] = (20, 10)
plt.scatter(x=tempDF['置信度'], y=tempDF['Lift'], s=area, c=colors,cmap=plt.cm.Reds, alpha=1, marker='h')
```

```
<matplotlib.collections.PathCollection at 0x1a7b6838>
```



三、分类与预测

3.1数据处理

- 对即将进行预测的数据进行数据的清洗和处理工作

查看数据集中的具体数据内容

```
listings = pd.read_csv("data/listings.csv")
listings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28452 entries, 0 to 28451
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              28452 non-null  int64  
 1   name                            28451 non-null  object 
 2   host_id                         28452 non-null  int64  
 3   host_name                       28452 non-null  object 
 4   neighbourhood_group             0 non-null      float64
 5   neighbourhood                   28452 non-null  object 
 6   latitude                        28452 non-null  float64
 7   longitude                       28452 non-null  float64
 8   room_type                       28452 non-null  object 
 9   price                           28452 non-null  int64  
 10  minimum_nights                  28452 non-null  int64  
 11  number_of_reviews               28452 non-null  int64  
 12  last_review                     17294 non-null  object 
 13  reviews_per_month               17294 non-null  float64
 14  calculated_host_listings_count  28452 non-null  int64  
 15  availability_365                28452 non-null  int64  
dtypes: float64(4), int64(7), object(5)
memory usage: 2.9+ MB
```

查看数据前5行

```
listings.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	mi
0	44054	Modern and Comfortable Living in CBD	192875	East Apartments	NaN	朝阳区 / Chaoyang	39.89503	116.45163	Entire home/apt	792	1
1	100213	The Great Wall Box Deluxe Suite A团园长城小院东院套房	527062	Joe	NaN	密云县 / Miyun	40.68434	117.17231	Private room	1201	1
2	128496	Heart of Beijing: House with View 2	467520	Cindy	NaN	东城区	39.93213	116.42200	Entire home/apt	389	3
3	161902	cozy studio in center of Beijing	707535	Robert	NaN	东城区	39.93357	116.43577	Entire home/apt	376	1
4	162144	nice studio near subway, sleep 4	707535	Robert	NaN	朝阳区 / Chaoyang	39.93668	116.43798	Entire home/apt	537	1

```
#发现neighbourhood_group一栏全部都是缺失,因此舍弃这一列
listings = listings.drop(columns=['neighbourhood_group'],axis=1)
listings
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	id	name	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	minimum_nights
0	44054	Modern and Comfortable Living in CBD	192875	East Apartments	朝阳区 / Chaoyang	39.89503	116.45163	Entire home/apt	792	1
1	100213	The Great Wall Box Deluxe Suite A团园长城小院东院套房	527062	Joe	密云县 / Miyun	40.68434	117.17231	Private room	1201	1
2	128496	Heart of Beijing: House with View 2	467520	Cindy	东城区	39.93213	116.42200	Entire home/apt	389	3
3	161902	cozy studio in center of Beijing	707535	Robert	东城区	39.93357	116.43577	Entire home/apt	376	1
4	162144	nice studio near subway, sleep 4	707535	Robert	朝阳区 / Chaoyang	39.93668	116.43798	Entire home/apt	537	1
...	...	...	...	...	...	...	...	...	...	...
28447	33948728	望京西门子,798,央美附近 温馨如家大床民宿	256112655	旭	朝阳区 / Chaoyang	39.98671	116.47394	Entire home/apt	396	1
28448	33948787	04简约舒适电梯房/工体/三里屯/东大桥/朝阳医院/世贸天阶/国贸	147335664	Pony	朝阳区 / Chaoyang	39.92560	116.44735	Entire home/apt	1302	3
28449	33950006	临近地铁温馨网红风小屋一居室	141786513	昊	朝阳区 / Chaoyang	39.89733	116.50473	Entire home/apt	329	1
28450	33950535	3. 老国展,三元桥地铁,静安东里大床房	213500128	晓征	朝阳区 / Chaoyang	39.95988	116.45187	Private room	188	1
28451	33954414	密码锁自行入住,隐私安全,丰台宋家庄交通枢纽站,去往北京站北京南站,天安门故宫,长城水魔方,东方医院	252799678	超	丰台区 / Fengtai	39.84714	116.43481	Entire home/apt	295	1

28452 rows × 15 columns

```
#对于neighbourhood这一列,我们只需要地区的中文名称即可,可以将后面的英文名称删除
listings['neighbourhood'] = listings['neighbourhood'].str.split('/').str[0]
listings
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	id	name	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	minimum_nights
0	44054	Modern and Comfortable Living in CBD	192875	East Apartments	朝阳区	39.89503	116.45163	Entire home/apt	792	1
1	100213	The Great Wall Box Deluxe Suite A团园长城小院东院套房	527062	Joe	密云县	40.68434	117.17231	Private room	1201	1
2	128496	Heart of Beijing: House with View 2	467520	Cindy	东城区	39.93213	116.42200	Entire home/apt	389	3
3	161902	cozy studio in center of Beijing	707535	Robert	东城区	39.93357	116.43577	Entire home/apt	376	1
4	162144	nice studio near subway, sleep 4	707535	Robert	朝阳区	39.93668	116.43798	Entire home/apt	537	1
...	...	...	...	...	...	...	...	...	...	...
28447	33948728	望京西门子,798,央美附近 温馨如家大床民宿	256112655	旭	朝阳区	39.98671	116.47394	Entire home/apt	396	1
28448	33948787	04简约舒适电梯房/工体/三里屯/东大桥/朝阳医院/世贸天阶/国贸	147335664	Pony	朝阳区	39.92560	116.44735	Entire home/apt	1302	3
28449	33950006	临近地铁温馨网红风小屋一居室	141786513	昊	朝阳区	39.89733	116.50473	Entire home/apt	329	1
28450	33950535	3. 老国展,三元桥地铁,静安东里大床房	213500128	晓征	朝阳区	39.95988	116.45187	Private room	188	1
28451	33954414	密码锁自行入住,隐私安全,丰台宋家庄交通枢纽站,去往北京站北京南站,天安门故宫,长城水魔方,东方医院	252799678	超	丰台区	39.84714	116.43481	Entire home/apt	295	1

28452 rows × 15 columns

```
#根据listings.info()的信息,name列存在缺失值,name缺失,则这条数据是没有意义的,应该将此行删除
listings = listings[listings['name'].notnull()]
listings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 28451 entries, 0 to 28451
Data columns (total 15 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              28451 non-null  int64
```

```
 1   name                            28451 non-null  object
 2   host_id                         28451 non-null  int64
 3   host_name                       28451 non-null  object
 4   neighbourhood                   28451 non-null  object
 5   latitude                        28451 non-null  float64
 6   longitude                       28451 non-null  float64
 7   room_type                       28451 non-null  object
 8   price                           28451 non-null  int64
 9   minimum_nights                  28451 non-null  int64
 10  number_of_reviews               28451 non-null  int64
 11  last_review                     17294 non-null  object
 12  reviews_per_month               17294 non-null  float64
 13  calculated_host_listings_count  28451 non-null  int64
 14  availability_365                28451 non-null  int64
dtypes: float64(3), int64(7), object(5)
memory usage: 2.9+ MB
```

```
###根据minimum_nights分组,查看相关结果
listings.groupby(by='minimum_nights').count()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	id	name	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	number_of_reviews
minimum_nights										
1	23679	23679	23679	23679	23679	23679	23679	23679	23679	23679
2	2276	2276	2276	2276	2276	2276	2276	2276	2276	2276
3	936	936	936	936	936	936	936	936	936	936
4	94	94	94	94	94	94	94	94	94	94
5	271	271	271	271	271	271	271	271	271	271
6	32	32	32	32	32	32	32	32	32	32
7	185	185	185	185	185	185	185	185	185	185
8	5	5	5	5	5	5	5	5	5	5
9	2	2	2	2	2	2	2	2	2	2
10	96	96	96	96	96	96	96	96	96	96
11	2	2	2	2	2	2	2	2	2	2
12	4	4	4	4	4	4	4	4	4	4
13	3	3	3	3	3	3	3	3	3	3
14	13	13	13	13	13	13	13	13	13	13
15	67	67	67	67	67	67	67	67	67	67
19	2	2	2	2	2	2	2	2	2	2
20	31	31	31	31	31	31	31	31	31	31
21	3	3	3	3	3	3	3	3	3	3
25	3	3	3	3	3	3	3	3	3	3
28	10	10	10	10	10	10	10	10	10	10
29	1	1	1	1	1	1	1	1	1	1
30	609	609	609	609	609	609	609	609	609	609
31	14	14	14	14	14	14	14	14	14	14
33	1	1	1	1	1	1	1	1	1	1
40	1	1	1	1	1	1	1	1	1	1
45	1	1	1	1	1	1	1	1	1	1
48	1	1	1	1	1	1	1	1	1	1
50	2	2	2	2	2	2	2	2	2	2
60	18	18	18	18	18	18	18	18	18	18
69	1	1	1	1	1	1	1	1	1	1
70	1	1	1	1	1	1	1	1	1	1
80	1	1	1	1	1	1	1	1	1	1
90	30	30	30	30	30	30	30	30	30	30
100	8	8	8	8	8	8	8	8	8	8
120	2	2	2	2	2	2	2	2	2	2
135	1	1	1	1	1	1	1	1	1	1
150	5	5	5	5	5	5	5	5	5	5
180	12	12	12	12	12	12	12	12	12	12
240	1	1	1	1	1	1	1	1	1	1
300	1	1	1	1	1	1	1	1	1	1
360	4	4	4	4	4	4	4	4	4	4
365	17	17	17	17	17	17	17	17	17	17
500	1	1	1	1	1	1	1	1	1	1
1000	2	2	2	2	2	2	2	2	2	2
1124	1	1	1	1	1	1	1	1	1	1

	id	name	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	number_of_reviews
minimum_nights										
1125	1	1	1	1	1	1	1	1	1	1

```
### 短租租期理论上应该不大于一年,因此大于365天的租期太长了,可以删除相关数据
listings = listings[listings['minimum_nights'] <= 365]
listings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 28446 entries, 0 to 28451
Data columns (total 15 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              28446 non-null  int64  
 1   name                            28446 non-null  object 
 2   host_id                         28446 non-null  int64  
 3   host_name                       28446 non-null  object 
 4   neighbourhood                   28446 non-null  object 
 5   latitude                        28446 non-null  float64
 6   longitude                       28446 non-null  float64
 7   room_type                       28446 non-null  object 
 8   price                           28446 non-null  int64  
 9   minimum_nights                  28446 non-null  int64  
 10  number_of_reviews               28446 non-null  int64  
 11  last_review                     17291 non-null  object 
 12  reviews_per_month               17291 non-null  float64
 13  calculated_host_listings_count  28446 non-null  int64  
 14  availability_365                28446 non-null  int64  
dtypes: float64(3), int64(7), object(5)
memory usage: 2.9+ MB
```

```
### 根据数据可视化结果,有些房价过高,我们可以查看价格高于10000的房源信息
listings[listings['price'] >=10000]
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	id	name	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	minimum_nights
1067	12689987	Artistic apartment with culture	68973377	晨斌	朝阳区	39.92300	116.57996	Entire home/apt	67104	1
1227	13418237	Great Wall Paradise Villa	6628610	Chris	怀柔区	40.46643	116.59467	Entire home/apt	10066	1
2012	15488817	Hotel apartment close to huge Mall	68973377	晨斌	朝阳区	39.91962	116.59173	Entire home/apt	63346	1
3427	18554424	东五环北欧风小两居	46923669	Shirley	朝阳区	39.85618	116.55519	Entire home/apt	10998	1
4199	19643684	观城9号(整院儿 圣泉寺 慕田峪 箭扣长城 红螺寺 响水湖)	47672470	经纬	顺义区	40.19383	116.87503	Entire home/apt	33471	1
4825	20404794	shannon的小屋	116443835	筱淋	朝阳区	39.92206	116.46648	Shared room	30002	1
4923	20532771	鸟巢水立方奥林匹克公园顶级豪宅	124507717	丰勇	朝阳区	39.98572	116.38590	Entire home/apt	19997	1
4988	20592852	秋苓美苑之水岸山舍	147144714	秋苓	怀柔区	40.29200	116.48337	Entire home/apt	13287	1
5167	20748712	大望路/九龙山大床房	141070198	洋	朝阳区	39.88798	116.47667	Entire home/apt	59997	1
5315	20851373	卢苑风景墅·风景尽收眼底	148180505	亚平	密云县	40.43884	116.86897	Entire home/apt	15998	1
6206	21651269	欧式皇家碧水宫殿	157265271	莱崔	昌平区	40.11721	116.29292	Entire home/apt	15803	1
6612	21942314	【温馨小窝窝】近地铁一号线五棵松/万寿路,距离北京西站3站地,15分钟。	48178909	Qing	海淀区	39.89523	116.28252	Shared room	59997	1
6631	21952658	【核桃树小院】-四九城儿老北京城中心的四合院 让您感受不同的老北京生活	160018763	耘	东城区	39.93781	116.40932	Private room	10797	1
7136	22457777	shannon的小屋	116443835	筱淋	朝阳区	39.92451	116.46530	Private room	30002	1
7279	22590967	御汤山现代欧式豪宅	157265271	莱崔	昌平区	40.18537	116.40322	Entire home/apt	12797	1
9173	24336898	慕田峪长城脚下赫家大院整院	180104890	桂华	怀柔区	40.42696	116.55851	Entire home/apt	11998	1
9512	24585516	60人轰趴Party包住宿+早晚餐自助烧烤&火锅2选1/KTV欢唱/麻将台球室/北京稻草坊艺术庄园	184698679	梦喆	昌平区	40.23816	116.36697	Entire home/apt	10287	1

	id	name	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	minimum_nights
9620	24634837	快乐阳光庄园	186175743	于春娟	密云县	40.65352	117.24925	Private room	49999	1
10170	24994830	良乡大学城两室温馨小屋	188806180	王	房山区	39.72157	116.15182	Entire home/apt	68828	1
10733	25434257	北京白河湾柏璟山水赋整院	184942283	Tank	怀柔区	40.64995	116.67208	Private room	28499	1
10766	25458469	北京景里高端四合院,后海、南锣鼓巷商圈,整套出租	189815103	秀华	东城区	39.93672	116.39088	Entire home/apt	12602	1
11837	26428751	北麓与懔言的结合	180077034	乐男	昌平区	40.32269	116.16036	Entire home/apt	10998	1
13152	27279622	【京遇】老北京城胡同大气独门四合院!前法国领事住宅!近雍和宫/南锣鼓巷/后海/国子监/簋街/故宫	45109145	Scarlett	东城区	39.93800	116.42069	Entire home/apt	12891	1
13504	27512515	1000平超大空间~团建聚会★活动开趴 独院水系山景别墅	139339588	东森家❤	昌平区	40.23415	116.37019	Entire home/apt	30197	1
13668	27587044	房源已下架	208158466	晶	昌平区	40.08912	116.29895	Private room	66667	30
13724	27624054	北京怀柔黄花城水长城超玩汇轰趴馆.水长城特色民宿	206627778	超	怀柔区	40.41151	116.32850	Private room	12877	1
13736	27639330	古北水镇首排观景小叠墅	208580528	Coco	密云县	40.64913	117.26936	Entire home/apt	19997	1
14697	28134193	此房不能租,不要询问了	212328505	陈	海淀区	39.94947	116.36246	Entire home/apt	68983	1
15024	28308986	四合院整套包院	132778194	欣	东城区	39.93536	116.41300	Private room	13582	1
15863	28687383	15号线南法信地铁口/机场T2T3高端商务套房/蜜月/求婚/生日长租优惠	214342283	瓜	顺义区	40.13874	116.60742	Entire home/apt	18890	1
16207	28803519	【北京站地铁3分钟.故宫周边最优惠.王府井商圈】溪流到家静雅民宿	216392612	容	东城区	39.90583	116.42199	Entire home/apt	65970	1
16862	29031102	享受乡村悠闲时光·北京琉璃山水民宿欢迎您	218791870	Mona	怀柔区	40.63209	116.64687	Entire home/apt	12797	1

	id	name	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	minimum_nights
17083	29138170	全A小筑	74938348	洋	朝阳区	39.89685	116.45925	Entire home/apt	59997	1
17723	29576476	雾灵山云岫谷豪华别墅整院可住26人—雾灵溪筑	222480988	淑珍	密云县	40.65356	117.39256	Entire home/apt	11112	1
19275	30355531	温暖又安全的小屋	227930711	晓霜	大兴区	39.81326	116.43608	Private room	30002	1
19812	30682794	瓦尔方法	135287911	Ainsley	海淀区	39.97819	116.37128	Private room	12998	5
19906	30721927	珠江嘉华欢聚	151448876	李	昌平区	40.12009	116.38571	Entire home/apt	15803	1
21583	31425063	醉懒.整院.团建亲子.雁栖湖青龙峡红螺寺怀北漂流	86212548	琳琳	怀柔区	40.44715	116.69868	Private room	15890	1
21809	31535043	水立方,鸟巢附近六人间男神床位	236331220	王林	昌平区	40.07817	116.42163	Shared room	67909	1
21815	31543015	这是个测试的房源	40276332	Lei	通州区	39.92220	116.64632	Shared room	34699	1
23723	32676949	豪华独栋花园别墅1600平米泳池会所	157265271	莱崔	昌平区	40.12041	116.38432	Entire home/apt	15803	1
24011	32800312	【无喧四合院•金融街店】【8室1厅】近金融街 故宫 天安门	27194544	Zhengfan	西城区	39.92547	116.37115	Entire home/apt	16977	1
24050	32817466	国贸双井苹果社区北区豪华南向大开间24小时热水,交通方便。24小时保安,月租房。此价格是月价格	163230736	泊毅	朝阳区	39.89898	116.47027	Entire home/apt	10602	30
24086	32825776	【有巢公邸】85平一房一厅两卫 拎包入住	246939481	立	朝阳区	39.96267	116.45624	Entire home/apt	25996	30
24186	32861042	【有巢公邸】大两居129㎡ 两室三卫 高速宽带入户 拎包入住	246939481	立	朝阳区	39.96230	116.45775	Entire home/apt	31599	28
24789	33032062	老北京传统四合院,独门独院400多平米。聚会,摄影,会议,活动,团建	248605544	Sheng	西城区	39.92645	116.37149	Entire home/apt	12978	1
27105	33642998	个人私宅独立卫浴	253442321	玉娟	东城区	39.94320	116.40040	Private room	10737	1

	id	name	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	minimum_nights
28422	33943498	王府井 东方广场 公寓 东方豪庭 公寓 标准大床 标准双床 订房前请务必咨询 不要直接下单重要提醒	215617282	軒	东城区	39.91070	116.41357	Entire home/apt	25003	30

```
### 27587044,31543015,28134193三个房源其中两个房源已经下架,还有一个是测试房源,因此必须删除
listings = listings[(listings['id'] != 27587044)]
listings = listings[(listings['id'] != 28134193)]
listings = listings[(listings['id'] != 28134193)]
listings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 28444 entries, 0 to 28451
Data columns (total 15 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              28444 non-null  int64  
 1   name                            28444 non-null  object 
 2   host_id                         28444 non-null  int64  
 3   host_name                       28444 non-null  object 
 4   neighbourhood                   28444 non-null  object 
 5   latitude                        28444 non-null  float64
 6   longitude                       28444 non-null  float64
 7   room_type                       28444 non-null  object 
 8   price                           28444 non-null  int64  
 9   minimum_nights                  28444 non-null  int64  
 10  number_of_reviews               28444 non-null  int64  
 11  last_review                     17290 non-null  object 
 12  reviews_per_month               17290 non-null  float64
 13  calculated_host_listings_count  28444 non-null  int64  
 14  availability_365                28444 non-null  int64  
dtypes: float64(3), int64(7), object(5)
memory usage: 2.9+ MB
```

```
### 对于房价大于10000的房源,其房屋的出租类型为shared room的可能性不大,,因此可以删除
listings = listings[~((listings['price'] >=10000)&(listings['room_type']=='Shared room'))]
listings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 28440 entries, 0 to 28451
Data columns (total 15 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              28440 non-null  int64  
 1   name                            28440 non-null  object 
 2   host_id                         28440 non-null  int64  
 3   host_name                       28440 non-null  object 
 4   neighbourhood                   28440 non-null  object 
 5   latitude                        28440 non-null  float64
 6   longitude                       28440 non-null  float64
 7   room_type                       28440 non-null  object 
 8   price                           28440 non-null  int64  
 9   minimum_nights                  28440 non-null  int64  
 10  number_of_reviews               28440 non-null  int64  
 11  last_review                     17289 non-null  object 
 12  reviews_per_month               17289 non-null  float64
 13  calculated_host_listings_count  28440 non-null  int64  
 14  availability_365                28440 non-null  int64  
dtypes: float64(3), int64(7), object(5)
memory usage: 2.9+ MB
```

```
### 查看是否有房源价格为0的房屋
listings[listings['price']<=0]
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	id	name	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	minimum_nights	num
5085	20670843	【胡同老宅-轻语竹林房】旅游绝佳地段】步行即到雍和宫、近故宫天安门、南锣鼓巷、美食簋街	129840905	Jing	东城区	39.94292	116.41323	Entire home/apt	0	2	81
5806	21246510	限时北京二环四合院别墅拍摄聚会 商务会议 娱乐同仁堂老宅近簋街、雍和宫、东直门、南锣鼓巷、恭王府	83233661	Eva	东城区	39.93677	116.42076	Entire home/apt	0	1	0
28234	33895187	测试房源mm2	185140389	Ning Host	朝阳区	39.98147	116.47109	Private room	0	1	0

```
### 房源价格应该大于0,若出现房源价格为0的,可以删除
listings = listings[listings['price']>0]
listings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 28437 entries, 0 to 28451
Data columns (total 15 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              28437 non-null  int64
 1   name                            28437 non-null  object
 2   host_id                         28437 non-null  int64
 3   host_name                       28437 non-null  object
 4   neighbourhood                   28437 non-null  object
 5   latitude                        28437 non-null  float64
 6   longitude                       28437 non-null  float64
 7   room_type                       28437 non-null  object
 8   price                           28437 non-null  int64
 9   minimum_nights                  28437 non-null  int64
 10  number_of_reviews               28437 non-null  int64
```

```
 11  last_review                     17288 non-null  object
 12  reviews_per_month               17288 non-null  float64
 13  calculated_host_listings_count  28437 non-null  int64
 14  availability_365                28437 non-null  int64
dtypes: float64(3), int64(7), object(5)
memory usage: 2.9+ MB
```

```
### 若number_of_reviews为0,而last_view不为空则代表数据值有异常
listings[(listings['number_of_reviews']==0)&(~listings['last_review'].isnull())]
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	id	name	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews

```
### 填补缺失值
listings['last_review'].fillna('0000-00-00',inplace=True)
listings['reviews_per_month'].fillna(0,inplace=True)
listings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 28437 entries, 0 to 28451
Data columns (total 15 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              28437 non-null  int64
 1   name                            28437 non-null  object
 2   host_id                         28437 non-null  int64
 3   host_name                       28437 non-null  object
 4   neighbourhood                   28437 non-null  object
 5   latitude                        28437 non-null  float64
 6   longitude                       28437 non-null  float64
 7   room_type                       28437 non-null  object
 8   price                           28437 non-null  int64
 9   minimum_nights                  28437 non-null  int64
 10  number_of_reviews               28437 non-null  int64
 11  last_review                     28437 non-null  object
 12  reviews_per_month               28437 non-null  float64
 13  calculated_host_listings_count  28437 non-null  int64
 14  availability_365                28437 non-null  int64
dtypes: float64(3), int64(7), object(5)
memory usage: 2.9+ MB
```

room_type只有3个指标，将其dummy化

```
listings['room_type'] = pd.factorize(listings['room_type'])[0]

room_dummies_listings = pd.get_dummies(listings['room_type'], prefix = listings[['room_type']].columns[0])
listings_new = pd.concat( [listings, room_dummies_listings], axis=1 )
listings_new.drop(columns='room_type',inplace=True)
listings_new.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	id	name	host_id	host_name	neighbourhood	latitude	longitude	price	minimum_nights	number_of_reviews
0	44054	Modern and Comfortable Living in CBD	192875	East Apartments	朝阳区	39.89503	116.45163	792	1	89
1	100213	The Great Wall Box Deluxe Suite A团园长城小院东院套房	527062	Joe	密云县	40.68434	117.17231	1201	1	2
2	128496	Heart of Beijing: House with View 2	467520	Cindy	东城区	39.93213	116.42200	389	3	259
3	161902	cozy studio in center of Beijing	707535	Robert	东城区	39.93357	116.43577	376	1	26
4	162144	nice studio near subway, sleep 4	707535	Robert	朝阳区	39.93668	116.43798	537	1	37

把16个区域（neighbourhood)转换成0-15这16个数字

```
listings_new['neighbourhood_num'] = pd.factorize(listings_new['neighbourhood'])[0]
listings_new.head(20)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	id	name	host_id	host_name	neighbourhood	latitude	longitude	price	minimum_nights	number_of_reviews
0	44054	Modern and Comfortable Living in CBD	192875	East Apartments	朝阳区	39.89503	116.45163	792	1	89
1	100213	The Great Wall Box Deluxe Suite A团园长城小院东院套房	527062	Joe	密云县	40.68434	117.17231	1201	1	2
2	128496	Heart of Beijing: House with View 2	467520	Cindy	东城区	39.93213	116.42200	389	3	259
3	161902	cozy studio in center of Beijing	707535	Robert	东城区	39.93357	116.43577	376	1	26
4	162144	nice studio near subway, sleep 4	707535	Robert	朝阳区	39.93668	116.43798	537	1	37
5	279078	Nice Apartment in Beijing	1455726	Fiona	东城区	39.93958	116.43485	403	1	29
6	282825	Large 2 BR Apt in a great location	1466681	Florence	朝阳区	39.93712	116.45089	637	3	107
7	287026	Studio in downtown Beijing #2	1456491	Vera	朝阳区	39.94115	116.44122	416	1	3
8	287511	High-flr studio in downtown Beijing	1456491	Vera	朝阳区	39.94032	116.44227	416	1	37
9	314453	Delicate Apartment, Dongzhimen Area	1455726	Fiona	东城区	39.93943	116.43362	396	1	34
10	317195	Sweet Apartment, Embassy Area	1455726	Fiona	东城区	39.93946	116.43545	550	1	3
11	322292	Cozy studio in downtown Beijing	1456491	Vera	朝阳区	39.93849	116.44066	436	1	11
12	398784	1BD 1BTH Serviced Apt in Fulicheng	192875	East Apartments	朝阳区	39.89403	116.44896	859	1	4
13	445306	Apartment in the heart of Beijing	2212388	Betty	东城区	39.91342	116.41837	906	1	155
14	456641	High-floor downtown studio #3	1456491	Vera	朝阳区	39.93848	116.44248	436	1	8
15	458561	Studio in downtown Beijing #5	1456491	Vera	朝阳区	39.93967	116.44128	483	1	5
16	491280	Cozy studio in center of Beijing #2	707535	Robert	东城区	39.93516	116.43682	376	1	4
17	498126	High-floor downtown studio #4	1456491	Vera	朝阳区	39.94059	116.44175	416	2	15

	id	name	host_id	host_name	neighbourhood	latitude	longitude	price	minimum_nights	number_of_reviews
18	537674	Quadrangle Courtyard (Siheyuan)	2562442	Chen	西城区	39.92412	116.37063	8898	1	11
19	554123	High-floor downtown studio #5	1456491	Vera	朝阳区	39.93879	116.44120	416	1	8

```
### 将新的数据存储到新的文件中
listings_new.to_csv("data/listings_cleaning.csv",index=False)
```

数据预测

SVM

复制一份数据表用于临时处理

```
listings_for_predict = listings_new.copy()
```

选择评论数量、房间类型、地区、年出租天数为参数，价格为预测值，生成训练表与测试表

```
X = listings_for_predict[['number_of_reviews', 'room_type_0', 'room_type_1', 'room_type_2', 'neighbourhood_num', 'availability_365']].values
y = listings_for_predict[['price']].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

设定 SVM 参数，以 rbf 为核，惩罚因子设定为 100，核系数设定为 0.1，使用 SVR 支持向量机做回归模型训练

```
from sklearn import svm
svr = svm.SVR(kernel='rbf', C=100, gamma=0.1)
svr.fit(X_train, y_train)
```

```
SVR(C=100, gamma=0.1)
```

对测试数据进行预测

```
y_test_pred = svr.predict(X_test)
```

输出预测结果和训练数据的均方误差

```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_test_pred)
```

```
2003540.451974532
```

输出预测结果和训练数据的决定系数

```
from sklearn.metrics import r2_score
r2_score(y_test, y_test_pred)
```

```
-0.0005807277792788668
```

发现上述预测模型实际结果误差太大，进行进一步优化。

将价格 log 化，并乘十倍后取整，再进行模型训练及预测

```
import math
listings_for_predict['price_log'] = listings_for_predict['price'].map(lambda x: round(10 * math.log(x)))

listings_for_predict.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	id	name	host_id	host_name	neighbourhood	latitude	longitude	price	minimum_nights	number_of_reviews
0	44054	Modern and Comfortable Living in CBD	192875	East Apartments	朝阳区	39.89503	116.45163	792	1	89
1	100213	The Great Wall Box Deluxe Suite A团园长城小院东院套房	527062	Joe	密云县	40.68434	117.17231	1201	1	2
2	128496	Heart of Beijing: House with View 2	467520	Cindy	东城区	39.93213	116.42200	389	3	259
3	161902	cozy studio in center of Beijing	707535	Robert	东城区	39.93357	116.43577	376	1	26
4	162144	nice studio near subway, sleep 4	707535	Robert	朝阳区	39.93668	116.43798	537	1	37

```
X = listings_for_predict[['number_of_reviews', 'room_type_0', 'room_type_1', 'room_type_2', 'neighbourhood_num', 'availability_365']].values
y = listings_for_predict[['price_log']].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

### 训练
svr.fit(X_train, y_train)

### 预测
y_test_pred = svr.predict(X_test)
```

输出预测结果和训练数据的均方误差

```
mean_squared_error(y_test, y_test_pred)
```

```
50.43975543010075
```

输出预测结果和训练数据的决定系数

```
r2_score(y_test, y_test_pred)
```

```
0.21482703794599456
```

此时误差已经在可接受范围内，得出该数据预测回归模型，储存在 svr 中。

逻辑回归

使用 SVM 预测中处理好的临时数据 listings_for_predict，进行模型训练及预测

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()

### 训练
lr.fit(X_train, y_train)

### 预测
y_test_pred = lr.predict(X_test)
```

输出均方误差和决定系数

```
mean_squared_error(y_test, y_test_pred)
```

```
66.34353023909986
```

```
r2_score(y_test, y_test_pred)
```

```
-0.03273986375967386
```

发现上述预测模型实际结果 R 方为负数，进行进一步优化。

对 avaliable_365 进行归一化处理，先化列，再化行

```
from sklearn.preprocessing import MinMaxScaler
listings_for_predict['scale_availability_365']=MinMaxScaler().fit_transform(listings_for_predict['availability_365'].values.reshape(-1,1)).reshape(1,-1)[0]
listings_for_predict.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	id	name	host_id	host_name	neighbourhood	latitude	longitude	price	minimum_nights	number_of_reviews
0	44054	Modern and Comfortable Living in CBD	192875	East Apartments	朝阳区	39.89503	116.45163	792	1	89
1	100213	The Great Wall Box Deluxe Suite A团园长城小院东院套房	527062	Joe	密云县	40.68434	117.17231	1201	1	2
2	128496	Heart of Beijing: House with View 2	467520	Cindy	东城区	39.93213	116.42200	389	3	259
3	161902	cozy studio in center of Beijing	707535	Robert	东城区	39.93357	116.43577	376	1	26
4	162144	nice studio near subway, sleep 4	707535	Robert	朝阳区	39.93668	116.43798	537	1	37

```
X = listings_for_predict[['number_of_reviews', 'room_type_0', 'room_type_1', 'room_type_2', 'neighbourhood_num',
'scale_availability_365']].values
y = listings_for_predict[['price_log']].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

### 训练
lr.fit(X_train, y_train)

### 预测
y_test_pred = lr.predict(X_test)
```

输出均方误差和决定系数

```
mean_squared_error(y_test, y_test_pred)
```

```
52.26248241912799
```

```
r2_score(y_test, y_test_pred)
```

```
0.18645346759881898
```

此时误差已经在可接受范围内，得出该数据预测回归模型，储存在 lr 中。

决策树

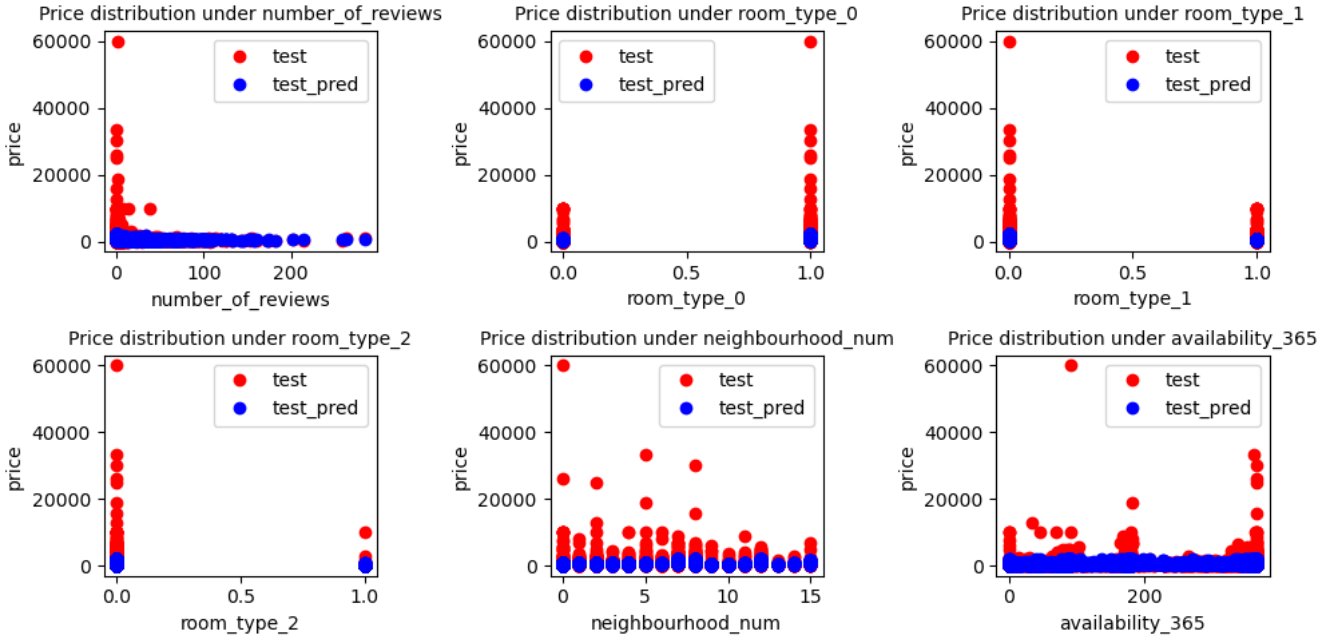
本模块采用决策树模型该数据集数据进行房价预测，主要涉及的数据内容特征与上述SVM相同

```
#读入数据
import pandas as pd
t_listings_new = pd.read_csv(r"data/listings_cleaning.csv")
X = t_listings_new[['number_of_reviews', 'room_type_0', 'room_type_1', 'room_type_2', 'neighbourhood_num', 'availability_365']].values
y = t_listings_new[['price']].values
x_tags = ['number_of_reviews', 'room_type_0', 'room_type_1', 'room_type_2', 'neighbourhood_num', 'availability_365']
from sklearn.model_selection import train_test_split
# 均方误差评价指标
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
from sklearn.tree import DecisionTreeRegressor
tree = DecisionTreeRegressor(max_depth=5)  # 树深的调整
tree.fit(X_train, y_train)
y_train_pred = tree.predict(X_train)
y_test_pred = tree.predict(X_test)
print('MSE train: %.3f, test: %.3f' % (mean_squared_error(y_train,y_train_pred),
                                       mean_squared_error(y_test,y_test_pred)))

print('R方 train: %.3f, test: %.3f' % (r2_score(y_train,y_train_pred),
                                       r2_score(y_test,y_test_pred)))
```

```
MSE train: 1829960.208, test: 1878760.818
R方 train: 0.060, test: 0.062
```

第一次拟合，未对数据进行处理，直接拟合，决策树最大深度随机设置为5，根据计算出的MSE和R方可以看出结果较差，预测房价和实际房价在各特征下数值分布在下方代码块中画出，我认为预测较差主要有两个原因，一是房价数据过于分散，决策树无法实现精准计算，二是决策树深度可以并不是最优。

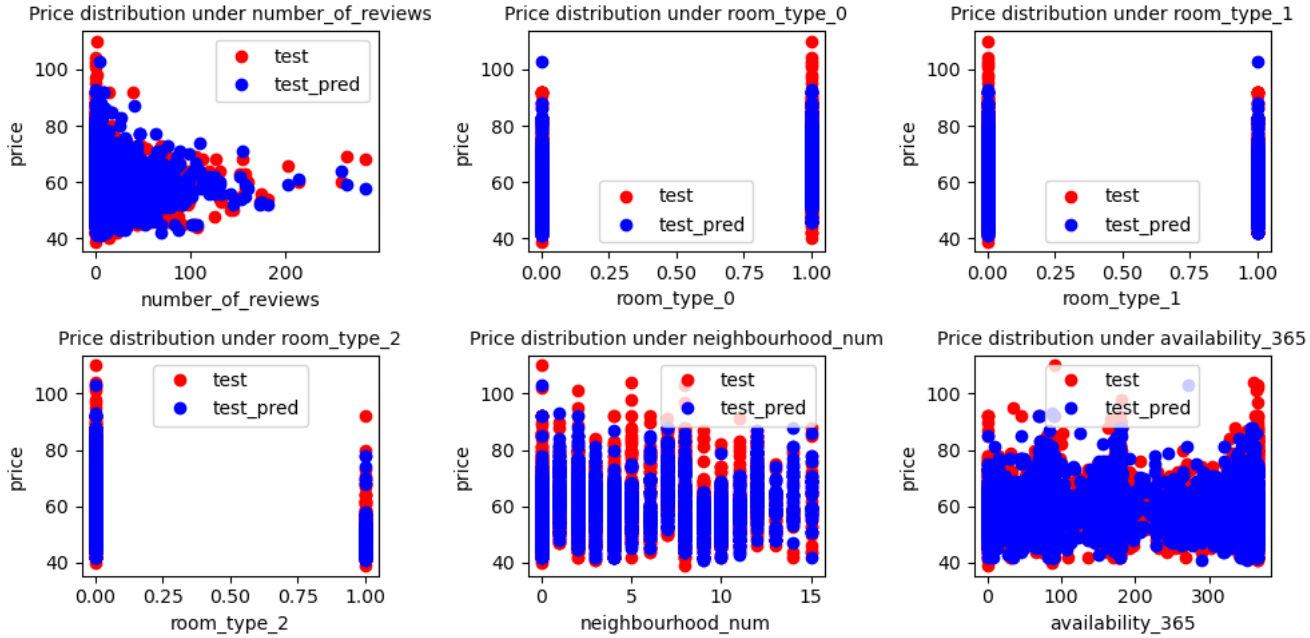
```
#预测房价和实际房价在各特征下数值分布
import matplotlib.pyplot as plt
plt.figure(figsize=(10,5), dpi=100)
for j in range(len(X_test[0])):

    plt.subplot(231+j)
    plt.title("Price distribution under {}".format(x_tags[j]),fontsize=10)
    plt.xlabel(x_tags[j])
    plt.ylabel("price")
    plt.plot([i[j] for i in X_test] ,y_test,"ro")
    plt.plot([i[j] for i in X_test],y_test_pred,"bo")
    plt.legend(['test','test_pred'])
plt.tight_layout()
plt.show()
```



针对难以精准预测的问题，根据常识，一般中低端房价价位一般有据可循，而高端房价则更加凭借出租者的想法，较难以预测，因此我将房价取对数乘以10并取整，这么做的目的是将高房价尽可能地映射到相近的值，而同时保存低房价的差异性。
为解决决策树的最大深度最优，我这次实验进一步尝试使得决策树的深度尽可能大，从而尽可能的拟合训练数据集。

```
import math
t_listings_new['price_log'] = t_listings_new['price'].map(lambda x: round(10 * math.log(x)))

t_listings_new.head()
X = t_listings_new[['number_of_reviews', 'room_type_0', 'room_type_1', 'room_type_2', 'neighbourhood_num', 'availability_365']].values
y = t_listings_new[['price_log']].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
tree = DecisionTreeRegressor()
### 训练
tree.fit(X_train, y_train)

### 预测

y_train_pred = tree.predict(X_train)
y_test_pred = tree.predict(X_test)
print('MSE train: %.3f, test: %.3f' % (mean_squared_error(y_train,y_train_pred),
                                       mean_squared_error(y_test,y_test_pred)))

print('R方 train: %.3f, test: %.3f' % (r2_score(y_train,y_train_pred),
                                       r2_score(y_test,y_test_pred)))


import matplotlib.pyplot as plt
plt.figure(figsize=(10,5), dpi=100)
for j in range(len(X_test[0])):

    plt.subplot(231+j)
    plt.title("Price distribution under {}".format(x_tags[j]),fontsize=10)
    plt.xlabel(x_tags[j])
    plt.ylabel("price")
    plt.plot([i[j] for i in X_test] ,y_test,"ro")
    plt.plot([i[j] for i in X_test],y_test_pred,"bo")
    plt.legend(['test','test_pred'])
plt.tight_layout()
plt.show()
```

```
MSE train: 19.736, test: 52.907
R方 train: 0.686, test: 0.176
```



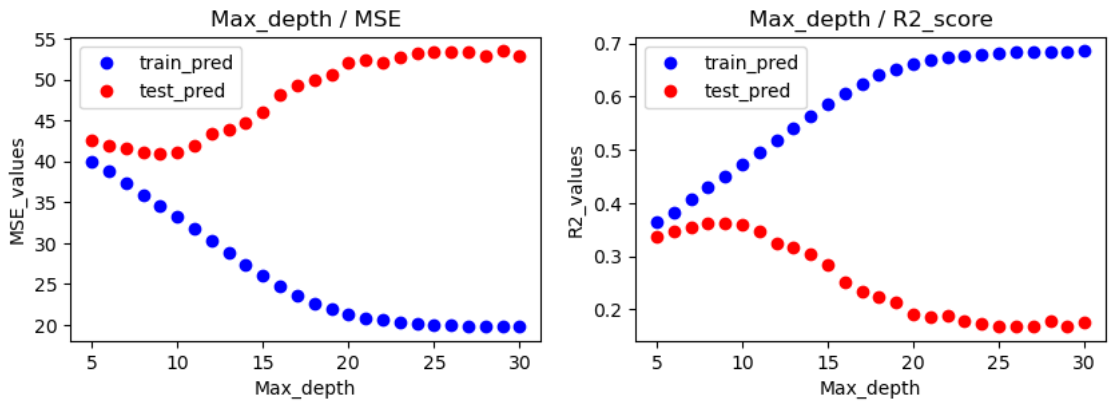
从上述结果可以看出，将房价取对数压缩预测空间，同时尽可能提升决策树深度，使得模型在训练集上能够获得较好的效果，但仍然可以看出该模型对于测试集的效果较差，这是很明显的过拟合现象，即决策树深度过大，既浪费了运算时间，同时在预测效果也不是很理想。于是对模型进行进一步改进。

```
mses = [[],[]]
r2s = [[],[]]

for i in range(5,31):
    tree = DecisionTreeRegressor(max_depth=i)  # 树深的调整
    tree.fit(X_train, y_train)
    y_train_pred = tree.predict(X_train)
    y_test_pred = tree.predict(X_test)
    print("决策树最大深度：%d"%(i))
    print('MSE train: %.3f, test: %.3f' % (mean_squared_error(y_train,y_train_pred),
                                           mean_squared_error(y_test,y_test_pred)))
    mses[0].append((mean_squared_error(y_train,y_train_pred)))
    mses[1].append(mean_squared_error(y_test,y_test_pred))
    print('R方 train: %.3f, test: %.3f' % (r2_score(y_train,y_train_pred),
                                           r2_score(y_test,y_test_pred)))
    r2s[0].append((r2_score(y_train,y_train_pred)))
    r2s[1].append(r2_score(y_test,y_test_pred))
```

```
决策树最大深度：5
MSE train: 39.980, test: 42.598
R方 train: 0.364, test: 0.337
决策树最大深度：6
MSE train: 38.889, test: 41.883
R方 train: 0.381, test: 0.348
决策树最大深度：7
MSE train: 37.289, test: 41.538
R方 train: 0.407, test: 0.353
决策树最大深度：8
MSE train: 35.892, test: 41.027
R方 train: 0.429, test: 0.361
决策树最大深度：9
MSE train: 34.585, test: 40.913
R方 train: 0.450, test: 0.363
决策树最大深度：10
MSE train: 33.232, test: 41.135
R方 train: 0.471, test: 0.360
决策树最大深度：11
MSE train: 31.788, test: 41.962
R方 train: 0.494, test: 0.347
决策树最大深度：12
MSE train: 30.313, test: 43.439
R方 train: 0.518, test: 0.324
决策树最大深度：13
MSE train: 28.887, test: 43.826
R方 train: 0.541, test: 0.318
决策树最大深度：14
MSE train: 27.402, test: 44.639
R方 train: 0.564, test: 0.305
决策树最大深度：15
MSE train: 26.045, test: 45.994
R方 train: 0.586, test: 0.284
决策树最大深度：16
MSE train: 24.769, test: 48.137
R方 train: 0.606, test: 0.251
```

```
决策树最大深度：17
MSE train: 23.668, test: 49.257
R方 train: 0.624, test: 0.233
决策树最大深度：18
MSE train: 22.615, test: 49.926
R方 train: 0.640, test: 0.223
决策树最大深度：19
MSE train: 21.887, test: 50.589
R方 train: 0.652, test: 0.213
决策树最大深度：20
MSE train: 21.312, test: 51.978
R方 train: 0.661, test: 0.191
决策树最大深度：21
MSE train: 20.834, test: 52.369
R方 train: 0.669, test: 0.185
决策树最大深度：22
MSE train: 20.573, test: 52.126
R方 train: 0.673, test: 0.189
决策树最大深度：23
MSE train: 20.331, test: 52.789
R方 train: 0.677, test: 0.178
决策树最大深度：24
MSE train: 20.162, test: 53.125
R方 train: 0.679, test: 0.173
决策树最大深度：25
MSE train: 20.040, test: 53.345
R方 train: 0.681, test: 0.170
决策树最大深度：26
MSE train: 19.958, test: 53.445
R方 train: 0.683, test: 0.168
决策树最大深度：27
MSE train: 19.901, test: 53.403
R方 train: 0.683, test: 0.169
决策树最大深度：28
MSE train: 19.847, test: 52.809
R方 train: 0.684, test: 0.178
决策树最大深度：29
MSE train: 19.813, test: 53.469
R方 train: 0.685, test: 0.168
决策树最大深度：30
MSE train: 19.786, test: 52.952
R方 train: 0.685, test: 0.176
```

```
t_dep = list(range(5,31))

plt.figure(figsize=(10,3),dpi=100)
plt.subplot(121)
plt.plot(t_dep,mses[0],'bo')
plt.plot(t_dep,mses[1],'ro')
plt.title("Max_depth / MSE")
plt.legend(['train_pred','test_pred'])
plt.xlabel("Max_depth")
plt.ylabel("MSE_values")
plt.subplot(122)
plt.plot(t_dep,r2s[0],'bo')
plt.plot(t_dep,r2s[1],'ro')
plt.title("Max_depth / R2_score")
plt.legend(['train_pred','test_pred'])
plt.xlabel("Max_depth")
plt.ylabel("R2_values")
plt.show()
```



为求解出最优的决策树深度，我画出了决策树最大深度与MSE,R方指标的关系图，从图中可以看出，当决策树深度在9左右时，模型对于测试集的预测效果很好，之后，过拟合逐渐严重。由此可以得出当决策树最大深度为9时，决策树模型获得了对数据集的最优拟合。

随机森林

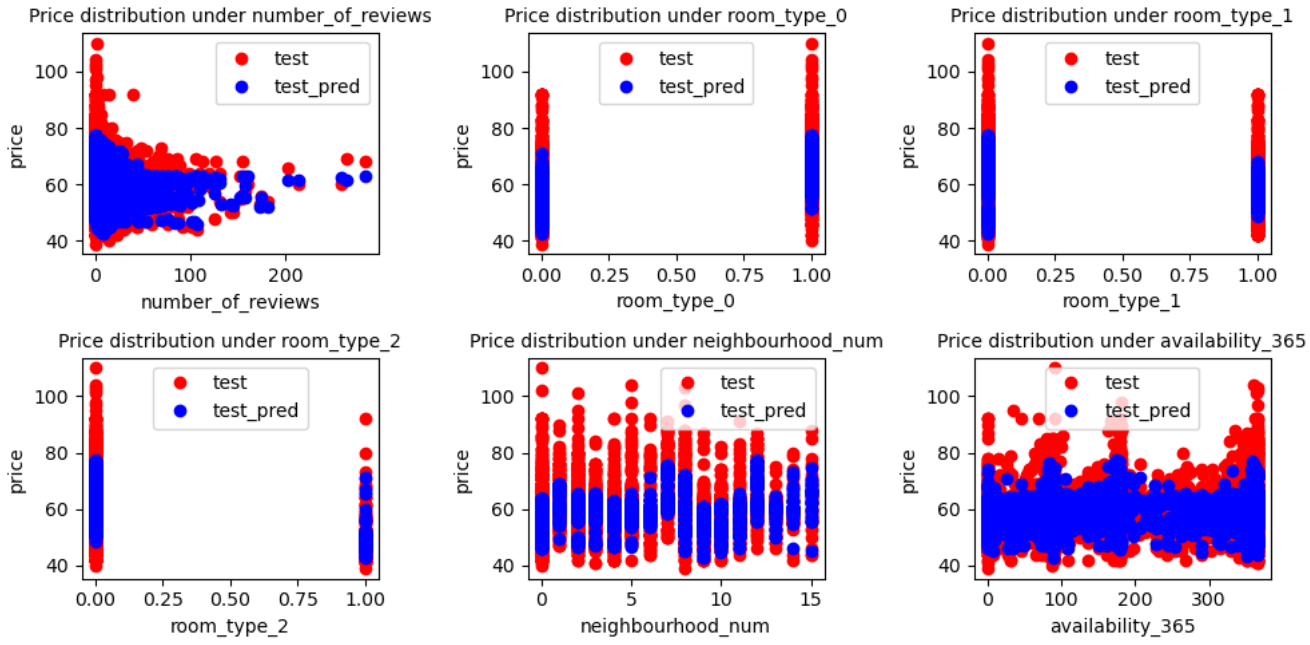
为进一步提升模型的性能，才有集成学习思想将决策树模型组合起来，既采用随机森林模型

```
# # Ignore irrelevant warnings
# import warnings
# warnings.filterwarnings('ignore')
from sklearn.ensemble import RandomForestRegressor
RF = RandomForestRegressor(max_depth=9)  # 其实有很多参数 这里默认
RF.fit(X_train, y_train.ravel())
y_train_pred = RF.predict(X_train)
y_test_pred = RF.predict(X_test)
print('MSE train: %.3f, test: %.3f' % (mean_squared_error(y_train,y_train_pred),
                                       mean_squared_error(y_test,y_test_pred)))

print('R方 train: %.3f, test: %.3f' % (r2_score(y_train,y_train_pred),
                                       r2_score(y_test,y_test_pred)))
```

```
MSE train: 33.427, test: 38.738
R方 train: 0.468, test: 0.397
```

在随机森林模型中，直接设置所有决策采用单棵最优决策树模型，设置深度为9，并默认学习器个数为100，获得较好的预测准确度。下图为预测房价和实际房价在各特征下数值分布。

```
plt.figure(figsize=(10,5), dpi=100)
for j in range(len(X_test[0])):

    plt.subplot(231+j)
    plt.title("Price distribution under {}".format(x_tags[j]),fontsize=10)
    plt.xlabel(x_tags[j])
    plt.ylabel("price")
    plt.plot([i[j] for i in X_test] ,y_test,"ro")
    plt.plot([i[j] for i in X_test],y_test_pred,"bo")
    plt.legend(['test','test_pred'])
plt.tight_layout()
plt.show()
```



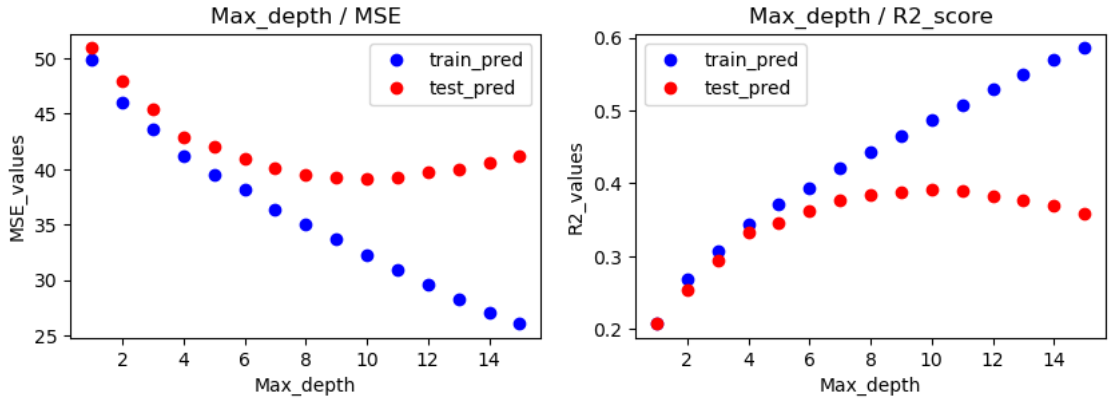

```
mses = [[],[]]
r2s = [[],[]]
t_dep = list(range(1,16))
for i in t_dep:
    RF = RandomForestRegressor(max_depth=i,n_estimators=10)  # 树深的调整
    RF.fit(X_train, y_train.ravel())
    y_train_pred = RF.predict(X_train)
    y_test_pred = RF.predict(X_test)
    print("决策树深度：%d"%(i))
    print('MSE train: %.3f, test: %.3f' % (mean_squared_error(y_train,y_train_pred),
                                           mean_squared_error(y_test,y_test_pred)))
    mses[0].append((mean_squared_error(y_train,y_train_pred)))
    mses[1].append(mean_squared_error(y_test,y_test_pred))
    print('R方 train: %.3f, test: %.3f' % (r2_score(y_train,y_train_pred),
                                           r2_score(y_test,y_test_pred)))
    r2s[0].append((r2_score(y_train,y_train_pred)))
```

```
    r2s[1].append(r2_score(y_test,y_test_pred))
```

```
决策树深度：1
MSE train: 49.850, test: 50.914
R方 train: 0.207, test: 0.207
决策树深度：2
MSE train: 46.043, test: 47.898
R方 train: 0.268, test: 0.254
决策树深度：3
MSE train: 43.569, test: 45.346
R方 train: 0.307, test: 0.294
决策树深度：4
MSE train: 41.221, test: 42.871
R方 train: 0.344, test: 0.333
决策树深度：5
MSE train: 39.479, test: 42.046
R方 train: 0.372, test: 0.345
决策树深度：6
MSE train: 38.171, test: 40.967
R方 train: 0.393, test: 0.362
决策树深度：7
MSE train: 36.356, test: 40.066
R方 train: 0.422, test: 0.376
决策树深度：8
MSE train: 35.002, test: 39.534
R方 train: 0.443, test: 0.385
决策树深度：9
MSE train: 33.689, test: 39.290
R方 train: 0.464, test: 0.388
决策树深度：10
MSE train: 32.210, test: 39.108
R方 train: 0.488, test: 0.391
决策树深度：11
MSE train: 30.954, test: 39.255
R方 train: 0.508, test: 0.389
决策树深度：12
MSE train: 29.602, test: 39.715
R方 train: 0.529, test: 0.382
决策树深度：13
MSE train: 28.290, test: 40.024
R方 train: 0.550, test: 0.377
决策树深度：14
MSE train: 27.087, test: 40.543
R方 train: 0.569, test: 0.369
决策树深度：15
MSE train: 26.048, test: 41.204
R方 train: 0.586, test: 0.359
```

为验证单棵决策树能否为随机森林模型带来最优，我默认学习器数量为10，并画出了随机森林模型的效能与决策树最大深度值的关系图，展示在下方。

```
plt.figure(figsize=(10,3),dpi=100)
plt.subplot(121)
plt.plot(t_dep,mses[0],'bo')
plt.plot(t_dep,mses[1],'ro')
plt.title("Max_depth / MSE")
plt.legend(['train_pred','test_pred'])
plt.xlabel("Max_depth")
plt.ylabel("MSE_values")
plt.subplot(122)
plt.plot(t_dep,r2s[0],'bo')
plt.plot(t_dep,r2s[1],'ro')
plt.title("Max_depth / R2_score")
plt.legend(['train_pred','test_pred'])
plt.xlabel("Max_depth")
plt.ylabel("R2_values")
plt.show()
```



可以看出当单棵决策树采取最优模型，即最大深度设置为9时，对数据集的拟合最优。

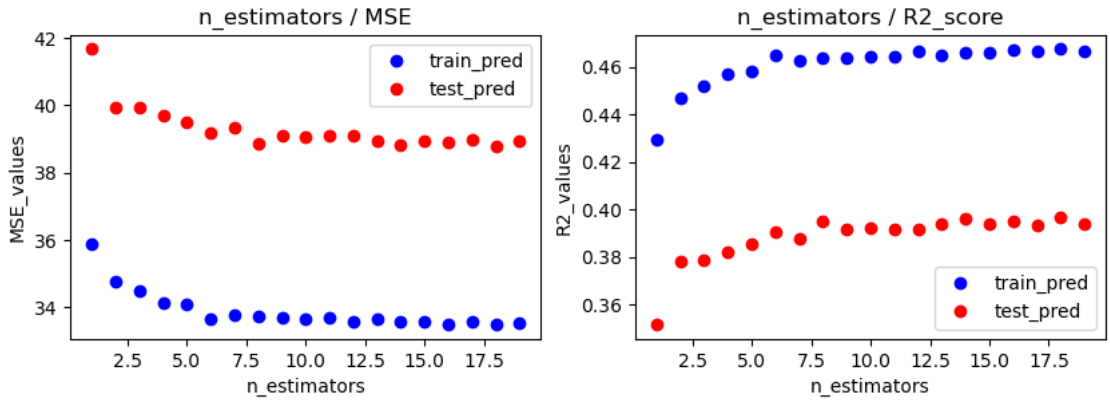
```
mses = [[],[]]
r2s = [[],[]]
t_dep = list(range(1,20))
for i in t_dep:
    RF = RandomForestRegressor(max_depth=9,n_estimators=i)  # 树深的调整
    RF.fit(X_train, y_train.ravel())
    y_train_pred = RF.predict(X_train)
    y_test_pred = RF.predict(X_test)
    print("决策树数量：%d"%(i))
    print('MSE train: %.3f, test: %.3f' % (mean_squared_error(y_train,y_train_pred),
                                           mean_squared_error(y_test,y_test_pred)))
    mses[0].append((mean_squared_error(y_train,y_train_pred)))
    mses[1].append(mean_squared_error(y_test,y_test_pred))
    print('R方 train: %.3f, test: %.3f' % (r2_score(y_train,y_train_pred),
                                           r2_score(y_test,y_test_pred)))
    r2s[0].append((r2_score(y_train,y_train_pred)))
    r2s[1].append(r2_score(y_test,y_test_pred))
```

```
决策树数量：1
MSE train: 35.881, test: 41.668
R方 train: 0.429, test: 0.351
决策树数量：2
MSE train: 34.760, test: 39.939
R方 train: 0.447, test: 0.378
决策树数量：3
MSE train: 34.468, test: 39.918
R方 train: 0.452, test: 0.379
决策树数量：4
MSE train: 34.136, test: 39.698
R方 train: 0.457, test: 0.382
决策树数量：5
MSE train: 34.066, test: 39.483
R方 train: 0.458, test: 0.385
决策树数量：6
MSE train: 33.641, test: 39.171
R方 train: 0.465, test: 0.390
决策树数量：7
MSE train: 33.785, test: 39.328
R方 train: 0.463, test: 0.388
决策树数量：8
MSE train: 33.727, test: 38.850
R方 train: 0.464, test: 0.395
决策树数量：9
MSE train: 33.705, test: 39.079
R方 train: 0.464, test: 0.392
决策树数量：10
MSE train: 33.666, test: 39.043
R方 train: 0.465, test: 0.392
决策树数量：11
MSE train: 33.686, test: 39.086
R方 train: 0.464, test: 0.392
决策树数量：12
MSE train: 33.550, test: 39.091
R方 train: 0.466, test: 0.391
决策树数量：13
MSE train: 33.645, test: 38.929
R方 train: 0.465, test: 0.394
决策树数量：14
MSE train: 33.556, test: 38.809
R方 train: 0.466, test: 0.396
决策树数量：15
MSE train: 33.554, test: 38.928
R方 train: 0.466, test: 0.394
决策树数量：16
```

```
MSE train: 33.483, test: 38.882
R方 train: 0.467, test: 0.395
决策树数量：17
MSE train: 33.549, test: 38.961
R方 train: 0.466, test: 0.394
决策树数量：18
MSE train: 33.478, test: 38.766
R方 train: 0.468, test: 0.397
决策树数量：19
MSE train: 33.547, test: 38.948
R方 train: 0.466, test: 0.394
```

同时为计算最优的学习器个数，采用类似方式，设定最大深度为9，画出学习器个数与评判指标的关系图。

```
plt.figure(figsize=(10,3),dpi=100)
plt.subplot(121)
plt.plot(t_dep,mses[0],'bo')
plt.plot(t_dep,mses[1],'ro')
plt.title("n_estimators / MSE")
plt.legend(['train_pred','test_pred'])
plt.xlabel("n_estimators")
plt.ylabel("MSE_values")
plt.subplot(122)
plt.plot(t_dep,r2s[0],'bo')
plt.plot(t_dep,r2s[1],'ro')
plt.title("n_estimators / R2_score")
plt.legend(['train_pred','test_pred'])
plt.xlabel("n_estimators")
plt.ylabel("R2_values")
plt.show()
```



可以看出，当决策树深度设置为9，学习器个数设置为10时，使得随机森林模型的拟合达到最优。