爬虫

spider中内容（对空值直接删除）

```
import scrapy
import re
from lianjia.items import LianJiaItem # 引入item对象
class LianJiaSpider(scrapy.Spider):
    name = "lianjia" # 爬虫的名字是 lianjia
    allowed_domains = ["dl.lianjia.com/"]  #允许爬取的网站域名
    start_urls = []
    for x in range(1, 101):
        url = 'https://dl.lianjia.com/ershoufang/pg{}/'.format(x)
        start_urls.append(url)

    def parse(self, response): #解析爬取的内容
        item = LianJiaItem() #生成一个在 items.py 中定义好的 xuetangitem 对象用于接收
爬取的数据

        # 总价
        # /html/body/div[4]/div[1]/ul/li[1]/div[1]/div[6]/div[1]/span
        # /html/body/div[4]/div[1]/ul/li[2]/div[1]/div[6]/div[1]/span

        # 单价
        # /html/body/div[4]/div[1]/ul/li[1]/div[1]/div[6]/div[2]/span
        # /html/body/div[4]/div[1]/ul/li[2]/div[1]/div[6]/div[2]/span
        for each in response.xpath('/html/body/div[4]/div[1]/ul/li'):
            item['total_price'] =
each.xpath("div[1]/div[6]/div[1]/span/text()").get()
            temp = each.xpath("div[1]/div[6]/div[2]/span/text()").get()
            # print('**********************************************************')
            # print(temp)
            # print(re.findall(r"\d+\.?\d*", temp)[0])
            # print('**********************************************************')
            item['unit_price'] = re.findall(r"\d+\.?\d*", temp)[0]
            # item['unit_price'] =
each.xpath("div[1]/div[6]/div[2]/span/text()").get()
            if(item['total_price'] and item['unit_price']): #去掉值为空的数据
                yield(item) #返回 item 数据给到 pipelines 模块
```

在pipeline中添加对csv文件写入的处理

```
import csv
# -*- coding: utf-8 -*-

# Define your item pipelines here
#
# Don't forget to add your pipeline to the ITEM_PIPELINES setting
# See: http://doc.scrapy.org/en/latest/topics/item-pipeline.html
```

```
class LianjiaPipeline(object):
    def open_spider(self, spider):
        try: #打开 json 文件
            self.file = open('Data.csv', "w", encoding="utf-8")

        except Exception as err:
            print(err)
    def process_item(self, item, spider):
        writer = csv.writer(self.file)
        writer.writerow([item['total_price'], item['unit_price']])
        # self.file.write('"{}",'.format(item['total_price']))
        # self.file.write('"{}",'.format(item['unit_price']))
        return item
    def close_spider(self,spider):
        self.file.close() #关闭文件
```

爬取的数据结果

Total_price	Unit_price
535	24370
155	11051
210	9503
70	14532
143	12707
198	29084
279	28279
198	21438
92	15901
510	51516
276	19509
130	15282
83	11893
330	25515
108	22123
95	10796
220	22236
198	21254

数据分析

In [7]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib notebook
```

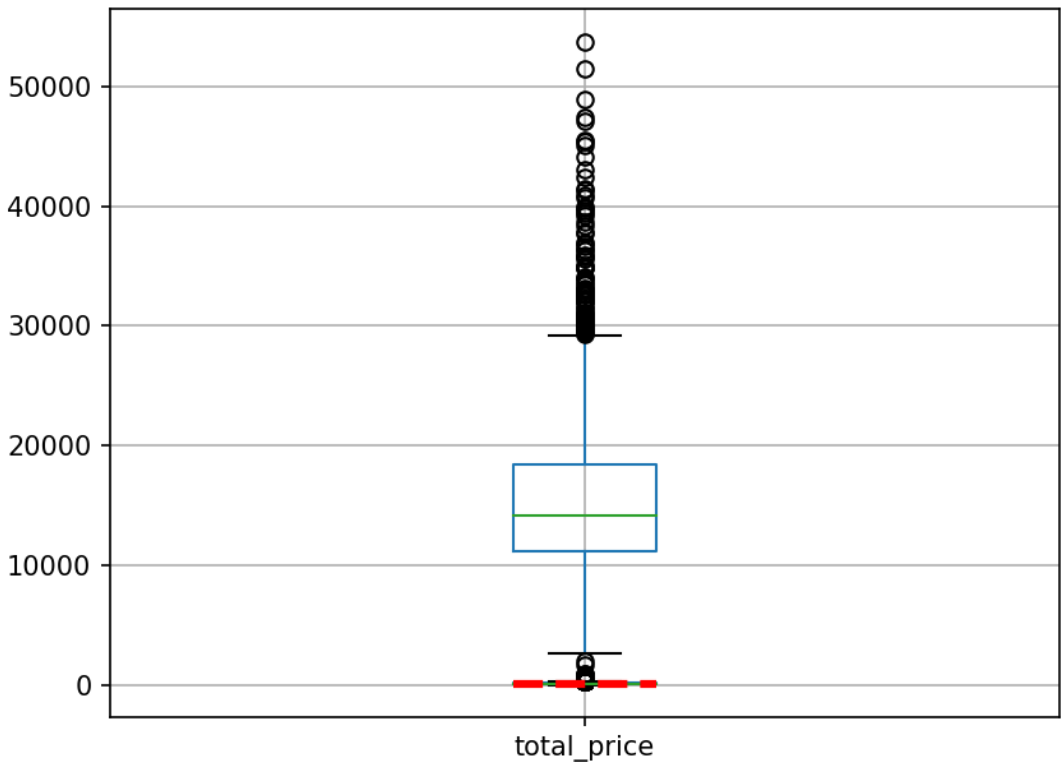
统计量分析

In [4]:
```
df = pd.read_csv('./lianjia/lianjia/Data.csv')
df.describe()
```

Out[4]:

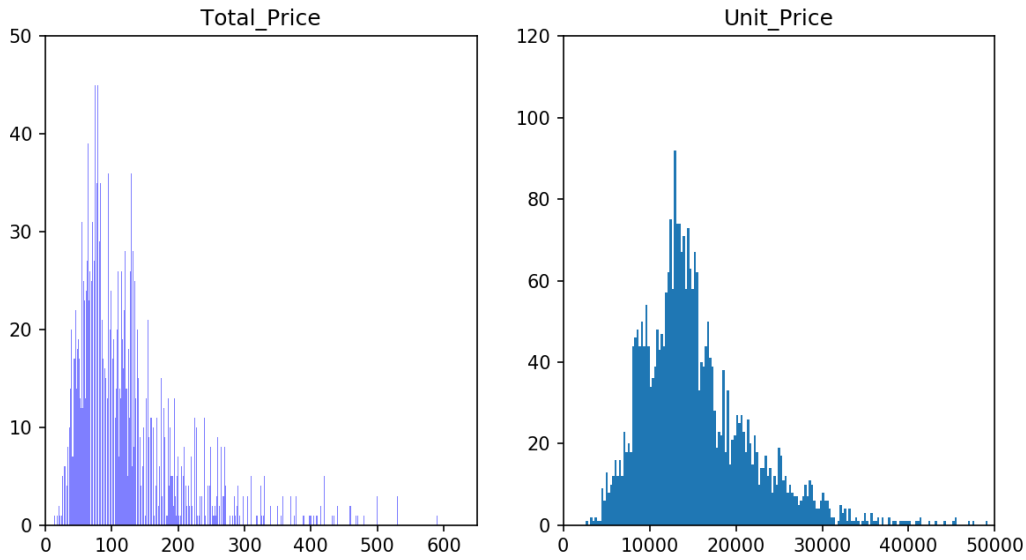
	total_price	unit_price
count	2998.000000	2998.000000
mean	128.540560	15362.356237
std	102.576454	6377.058592
min	14.000000	2596.000000
25%	71.500000	11125.250000
50%	100.000000	14123.000000
75%	150.000000	18400.250000
max	2000.000000	53770.000000

In [9]:
```
fig1 = plt.figure()
df.boxplot(column='unit_price')  #生成箱型图
#展示均值线

f = df.boxplot(column='total_price',meanline=True,showmeans=True,return_type='dict')

for mean in f['means']:
    mean.set(color='r', linewidth=3)

plt.show()
```



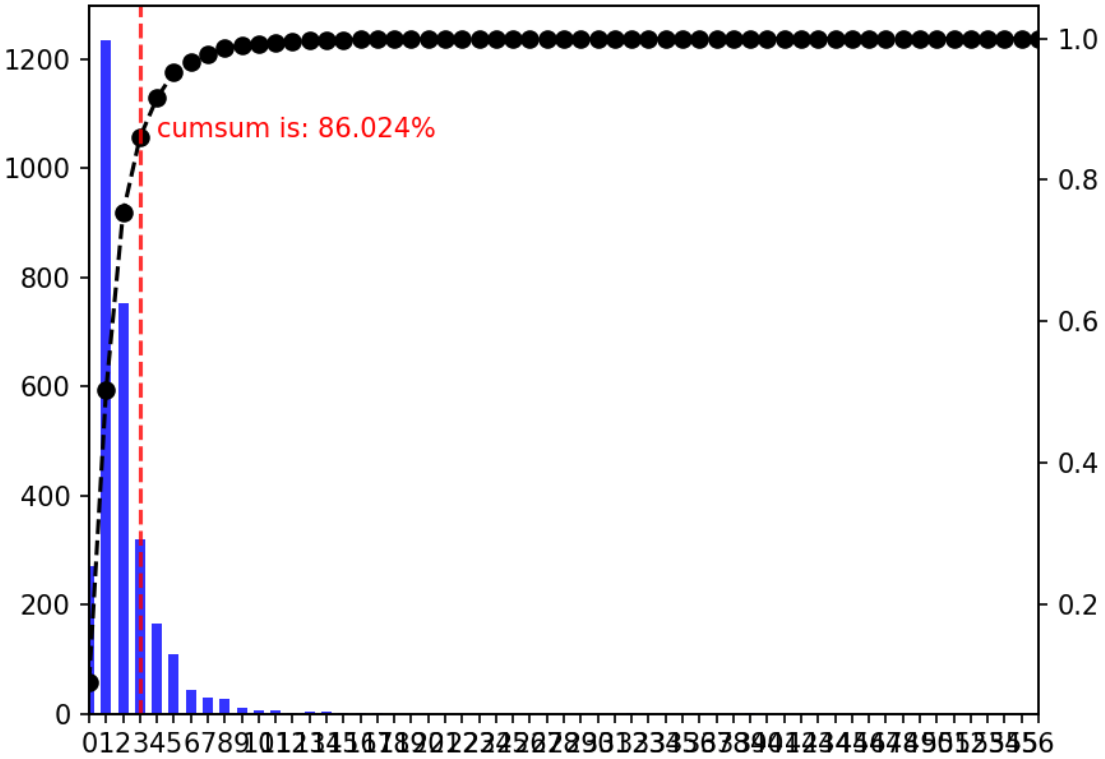
分布分析

In [19]:
```
#生成2个子图
fig2 = plt.figure()
ax1 = fig2.add_subplot(121)  #展示总价信息
ax2 = fig2.add_subplot(122)  #展示单价信息

def count_elements(scores): #定义转换函数，统计每个数值对应多少个
    scorescount = {}  #定义一个字典对象
    for i in scores:
        scorescount[int(i)] = scorescount.get(int(i), 0) + 1 #累加每个整数数值的个数
    return scorescount

#part1 展示总价和单价的分布直方图

counted1 = count_elements(df["total_price"])
ax1.set_title("Total_Price")
ax1.bar(list(counted1.keys()),counted1.values(),0.8,alpha=0.5,color='b')

# counted2 = count_elements(df["unit_price"])
ax2.set_title("Unit_Price")
# ax2.bar(list(counted2.keys()),counted2.values(),0.8,alpha=0.5,color='r')
# 使用频率分布直方图的形式显示
ax2.hist(df['unit_price'], bins = 200)

ax1.axis([0, 650, 0, 50])
ax2.axis([0, 50000, 0, 120])
plt.show()
#part1 展示总价和单价的分布直方图 结束
# ax1.axis([0, 1000, 0, ])
# ax1.hist(sample_int, bins = 200)
```



二八定律验证

In [21]:
```
sections = list(np.arange(0,2900,50))
mylabels = list(np.arange(25,2850,50))  #生成x轴上的标签
result1 = pd.cut(df.total_price,sections,labels=mylabels)
result2=result1.value_counts().sort_index()
fig3 = plt.figure()
df_counts = pd.Series(result2.values)  #频数，每个区间中房子的个数
# print("----------df_counts----------")
# print(df_counts)

df_freq = df_counts/df_counts.sum()   #频率，每个区间的房子个数/总个数，即每个区间的占比
# print("----------df_freq----------")
# print(df_freq)

cum_ratio = df_freq.cumsum()          #累计频率，累计百分比
# print("----------df_cum_freq----------")
# print(cum_ratio)

df_counts.plot(kind = 'bar', color = 'b', alpha = 0.8, width = 0.6)  #频数的直方图
key = cum_ratio[cum_ratio>0.8].index[0]  #找到大于80%的累计频率对应的索引号
key_num = df_counts.index.tolist().index(key)  #找到对应的索引序号
print('超过80%累计占比的节点值：' ,(key+1)*50)
print('超过80%累计占比的节点值索引位置为：' ,key_num)
print('------')
cum_ratio.plot(style = '--ko', secondary_y=True)  #累计频率的曲线图
plt.axvline(key_num,  color = 'r', linestyle = '--', alpha = 0.8) #把80%占比的参考线画出来，直接是key_num，因为它是X轴的索引值
plt.text(key_num+1,cum_ratio[key],'cumsum is: %.3f%%' % (cum_ratio[key]*100), color = 'r')  # 文字提示信息

plt.show()
```



超过80%累计占比的节点值： 200
超过80%累计占比的节点值索引位置为： 3

基本符合二八定律

In [ ]: