Python 大作业 电商信息获取、处理与可视化

班级：2017211314　　学号：2017213508　　学生：蒋雪枫

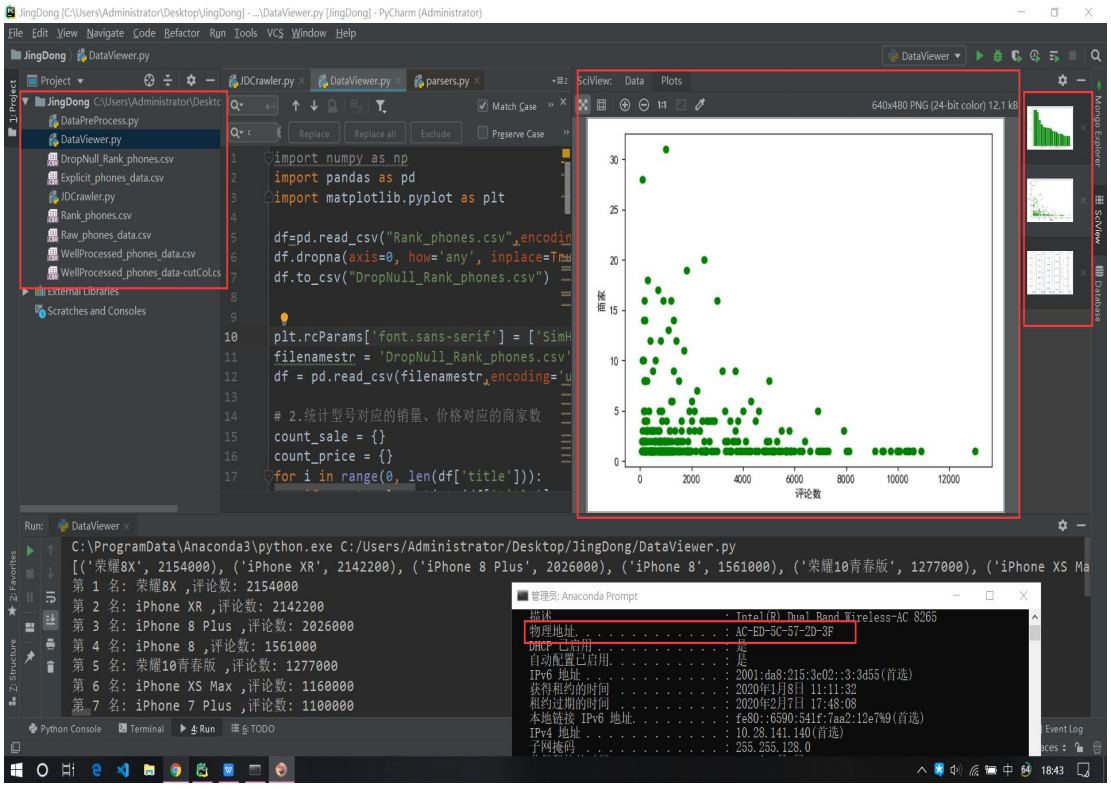
综述：

在本次大作业中，我们要按照销量数从大到小来爬取淘宝的手机销售数据，按照评价数从大到小来爬取京东的销售数据。这两个任务的区别一个是在于平台，一个是在于构造 url 的时候的微小区别。基本功能其实相对高级功能比较简单，因为高级功能是一定要进入一个商品的“内部页”并进行进一步的分析，但基本功能则不一定非要进入，同时京东的数据爬取也比淘宝的要友善许多。通过分析，我认为高级功能地完整实现是很困难的，或者说无法实现，因为我们对于一个“内部的”商品页，不能查看不同型号对应的不同销量，一个“内部”商品页可以有多个型号，但是它只给了你一个评论数/销售数字段，那到时候又要怎么统计如 iPhoneXr 128G 和 iPhoneXr 64G 的累计的销量/评价数信息呢？所以我们可以去做的附加功能是把信息爬取得尽可能完整一些。另外，这里数据爬的并不是特别多，两个版本都爬了 1300 左右条。(淘宝 45*30 pages，京东 60*22 pages)首先按老师上课所讲的内容来完成本次任务，会遇到很多问题，所以我本次是借助了上课所讲的知识，再额外查看与学习了一些资料才完成的本次任务，花费了不少时间与精力，但感觉还是很有意思的，这里用到的额外参考资料——如何绕开淘宝的身份认证实现流畅数据爬虫，都会在后面的核心代码部分进行解说。同时，全部代码上传在 Github，老师感兴趣可以去看看，链接是

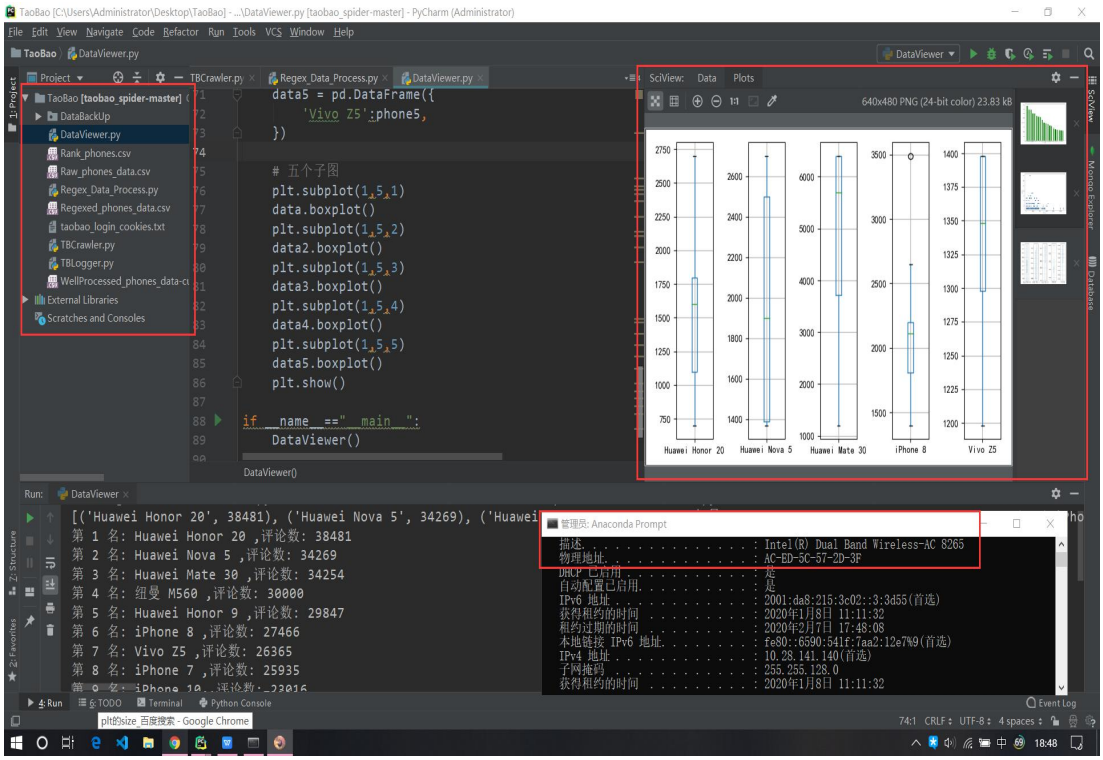
https://github.com/Sprinter1999/PythonAdvanced 。

一、老师要求的运行效果必需截图

京东高级版本：

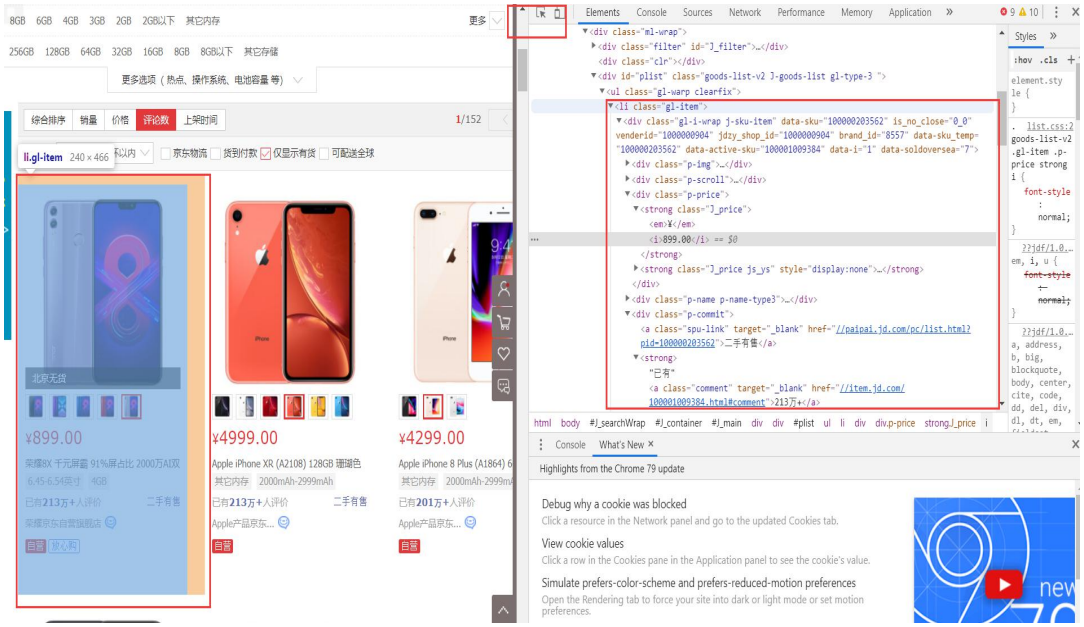


淘宝版本截图：



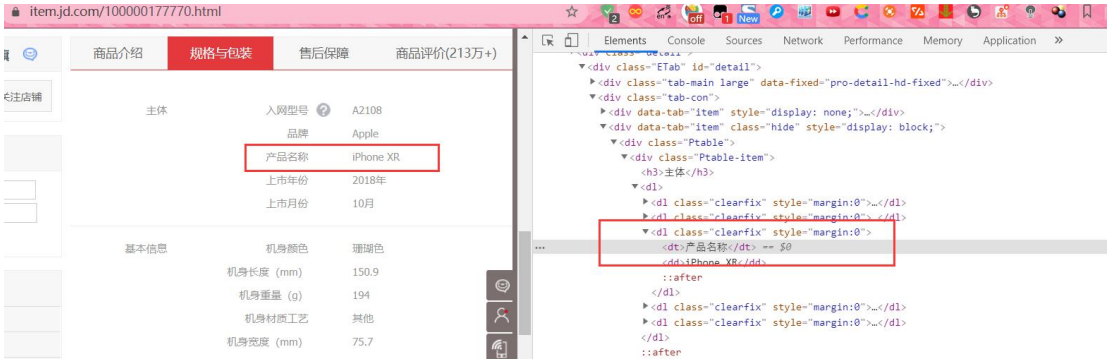
二、京东高级版本核心代码说明与注解

虽然京东高级版本我是在爬了淘宝之后做的，但是我想先进行说明：对于京东，我们分两步进行数据的爬取，第一步是在“有很多商品”的页面来获取每个商品对应的 id，也就获取了对应的链接，并且保存下来，用于第二步进入每一个内部商品页的具体数据获取。



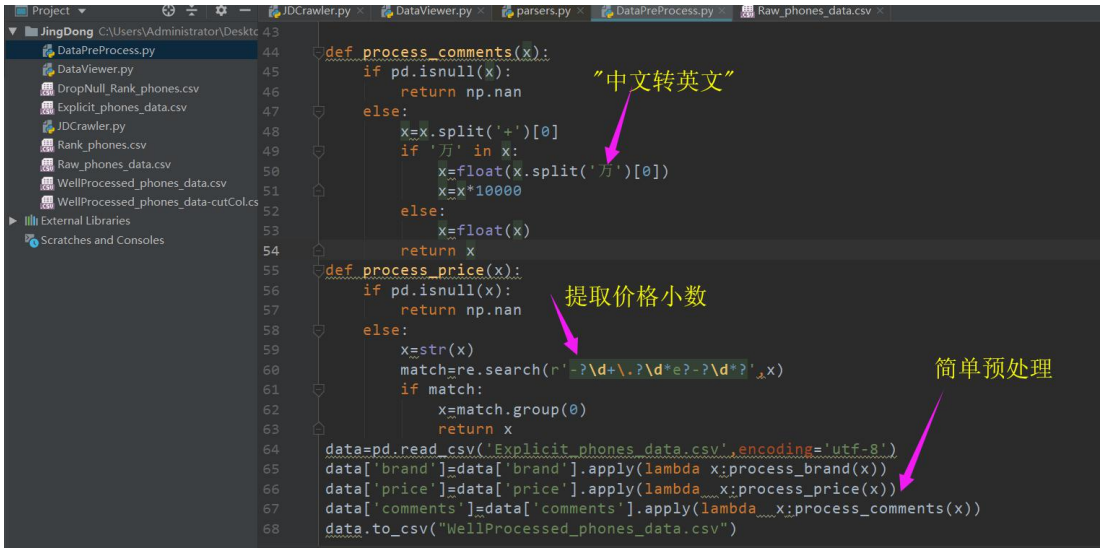
```
for i in range(22):
    #   JD按照评论数排序
    browser.get("https://list.jd.com/list.html?cat=9987,653,655&page={}&sort=sort_commentcount_desc".format(
    wait = WebDriverWait(browser, 10)
    page=html.fromstring(browser.page_source)
    print("Start visiting No.{} external pages,progress:".format(i))
    count=0
    print("[[",end='')
    for each in page.xpath("//div[@class='gl-i-wrap j-sku-item']"):
        count+=1
        if count%6==0:
            print('===',end='')
        phone_id=each.xpath("./@data-sku")[-1]
        price=each.xpath(".//div[@class='p-price']/strong[1]/i/text()")[:]
        comments=each.xpath(".//a[@class='comment']/text()")[-1]
        phone_list.append(list((phone_id,price,comments)))
    print("]]",end='')
    print("\nNo.{} external pages have been visited.".format(i))
browser.close()
```

而在一个具体的商品页中，比较详细的信息在这个规格与包装里面都有。这里我想说的是，京东的每个商品页都比较友好，它的 Restful 风格的商品构造页仅仅只有商品 id 的这个标号，这个商品 id 也就是前面获取到的。



商品页内部的核心处理逻辑：

```
for each in phone_list:
    url='https://item.jd.com/{}.html'.format(each[0])
    r=requests.get(url)
    if r.ok==False:
        print('不允许访问,出错!')
        break
    page=html.fromstring(r.text)
    dic_temp={}
    list_temp=page.xpath("//dl[@class='clearfix'][./dt[text()='电池容量（mAh）']]/dd/text()")
    list_temp.insert(0,np.nan)
    dic_temp['battery']=list_temp[-1]
    list_temp=page.xpath("//dl[@class='clearfix'][./dt[text()='上市年份']]/dd/text()")
    list_temp.insert(0,np.nan)
    dic_temp['year']=list_temp[-1]
    list_temp=page.xpath("//dl[@class='clearfix'][./dt[text()='上市月份']]/dd/text()")
    list_temp.insert(0,np.nan)
    dic_temp['month']=list_temp[-1]
    list_temp=page.xpath("//dl[@class='clearfix'][./dt[text()='品牌']]/dd/text()")
    list_temp.insert(0,np.nan)
    dic_temp['brand']=list_temp[-1]
    # //*[@id="detail"]/div[2]/div[2]/div[1]/div[1]/dl/dl[3]/dd
    list_temp=page.xpath("//dl[@class='clearfix'][./dt[text()='产品名称']]/dd/text()")
    list_temp.insert(0,np.nan)
    dic_temp['model']=list_temp[-1]
```



爬取的数据截图如下：

第一列有效数据是指商品 id。

下面是作图部分的完整代码：

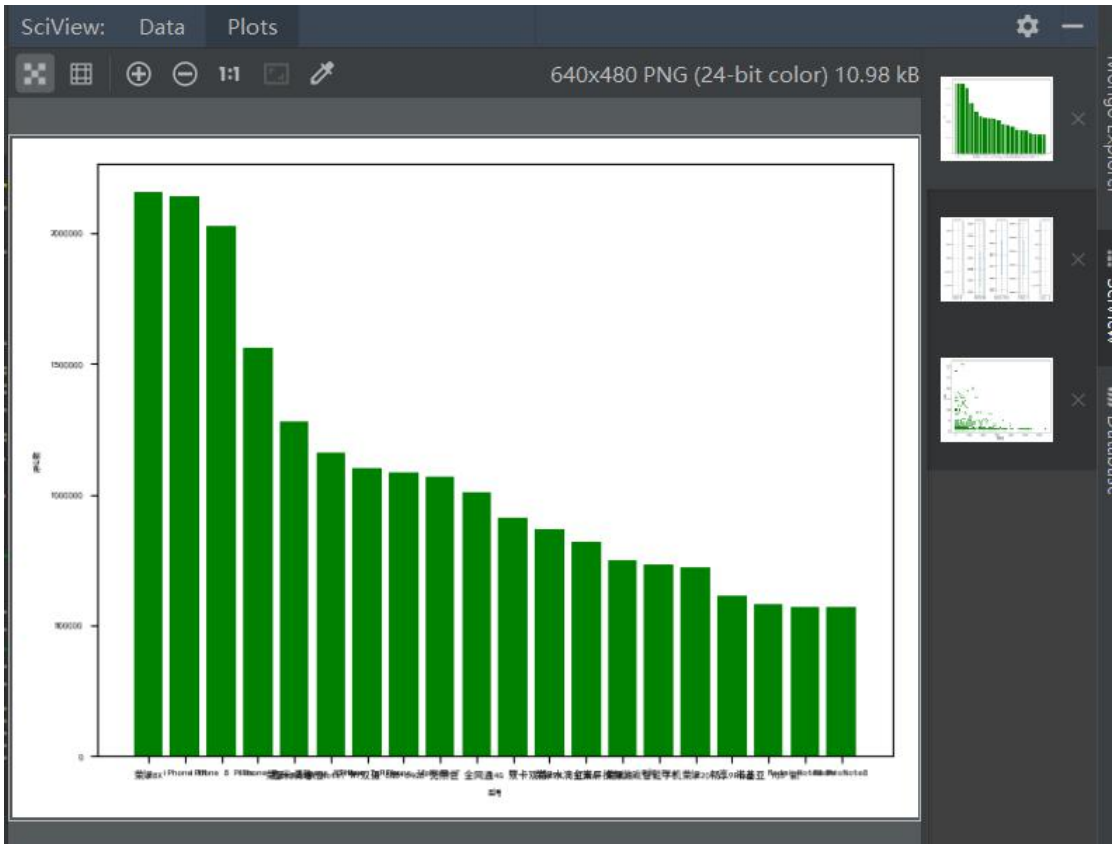
（最开头还有一个去空操作，这是因为有时候压根商品页就没数据，根本无法统计）

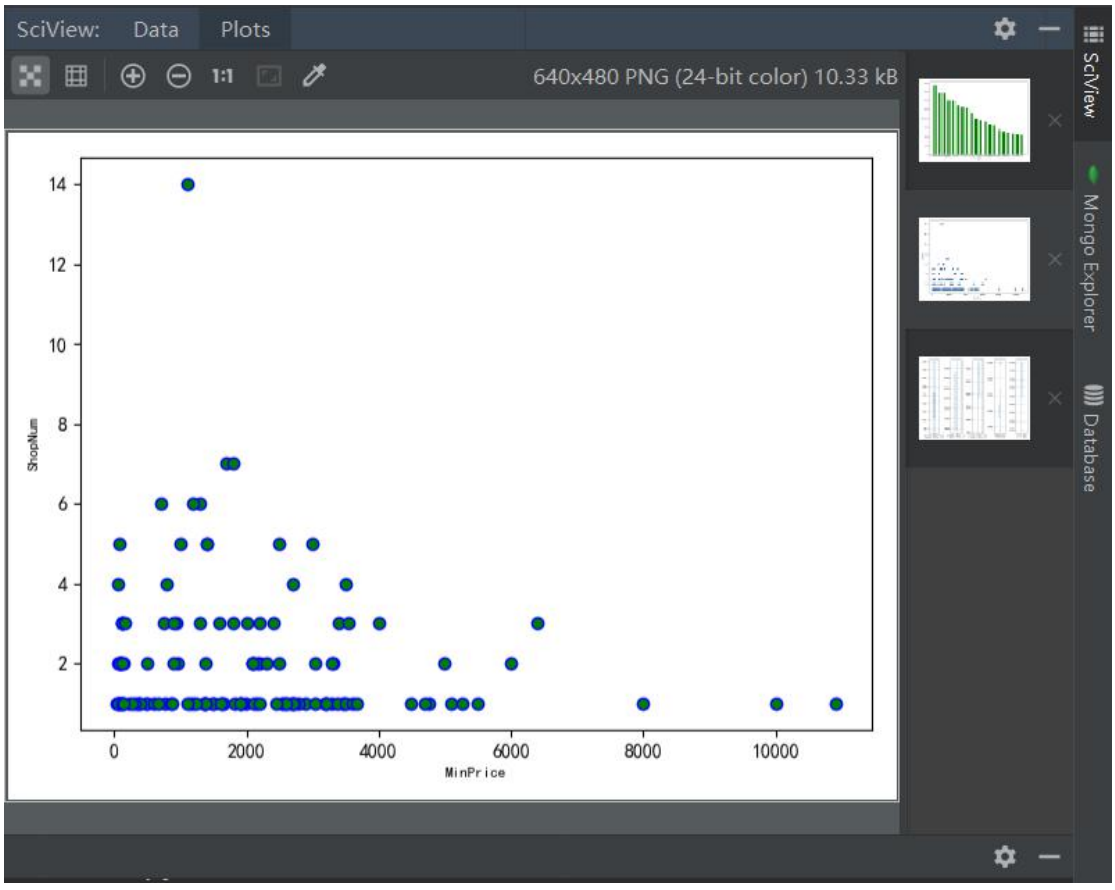
```
import pandas as pd
import matplotlib.pyplot as plt
#去空
df=pd.read_csv("Rank_phones.csv",encoding='gbk')
df.dropna(axis=0, how='any', inplace=True)
df.to_csv("DropNull_Rank_phones.csv")



plt.rcParams['font.sans-serif'] = ['SimHei']
filenamestr = 'DropNull_Rank_phones.csv'
df = pd.read_csv(filenamestr,encoding='utf-8')

# 2.统计型号对应的销量、价格对应的商家数
```

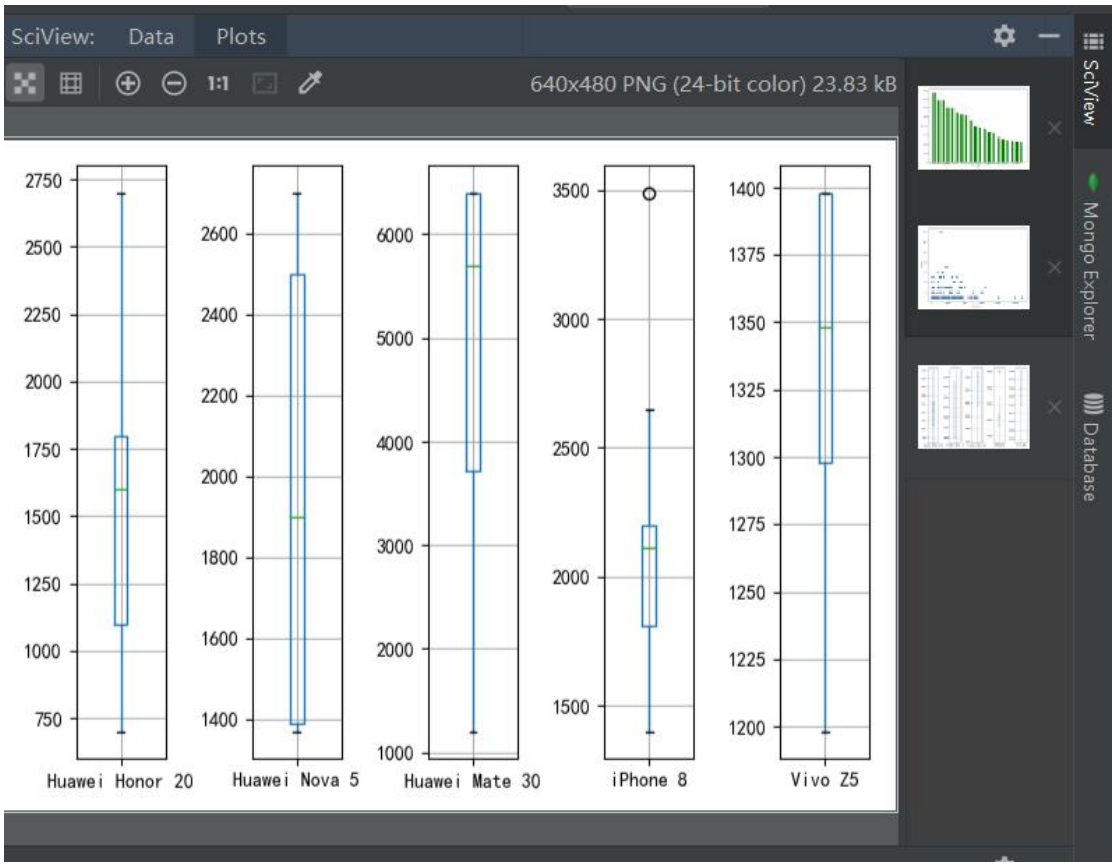
```
count_sale = {}
count_price = {}
for i in range(0, len(df['title'])):
    if not(count_sale.get(str(df['title'].values[i]))):
        count_sale[str(df['title'].values[i])] = 0
    if not(count_price.get(int(df['price'].values[i]))):
        count_price[int(df['price'].values[i])] = 0

    count_sale[str(df['title'].values[i])] += int(df['comments'].values[i])
    count_price[int(df['price'].values[i])] += 1

# 3.筛选销量前 20 的手机型号，并作图
count_sale2 = sorted(count_sale.items(), key=lambda x: x[1], reverse=True)
print(count_sale2)
x ,y= [],[]
plt.xlabel("型号",fontsize=4)
plt.ylabel("评论数",fontsize=4)
plt.tick_params(labelsize=5)
for i in range(0,20):
    print("第",i+1,"名:",count_sale2[i][0],",评论数:",count_sale2[i][1])
    x.append(count_sale2[i][0])
    y.append(count_sale2[i][1])
plt.bar(x, y,color='g')
plt.show()

# 4.画价格对应的散点图
plt.xlabel("评论数",fontsize=10,fontweight='bold')
plt.ylabel("商家",fontsize=10,fontweight='bold')
price = count_price.keys()
values = count_price.values()
plt.scatter(price,values,color='g')
plt.show()

# 5.画箱型图
phone1,phone2,phone3,phone4,phone5 = [],[],[],[],[]
# Honor 20
for i in range(0, len(df['title'])):
    if df['title'].values[i] == 'Honor 8X': #这是因为通过打印信息已知前五名
        phone1.append(df['price'].values[i])
    if df['title'].values[i] == 'iPhone XR':
        phone2.append(df['price'].values[i])
    if df['title'].values[i] == 'iPhone 8 Plus':
        phone3.append(df['price'].values[i])
    if df['title'].values[i] == 'iPhone 8':
```

```
            phone4.append(df['price'].values[i])
        if df['title'].values[i] == 'Honor 10':
            phone5.append(df['price'].values[i])


print(phone1, phone5)
data = pd.DataFrame({
        'Honor 8X':phone1,
})
data2 = pd.DataFrame({
        'iPhone XR':phone2,
})
data3 = pd.DataFrame({
        'iPhone 8 Plus':phone3,
})
data4 = pd.DataFrame({
        'iPhone 8':phone4,
})
data5 = pd.DataFrame({
        'Honor 10':phone5,
})

# 五个子图
plt.subplot(1,5,1)
data.boxplot()
plt.subplot(1,5,2)
data2.boxplot()
plt.subplot(1,5,3)
data3.boxplot()
plt.subplot(1,5,4)
data4.boxplot()
plt.subplot(1,5,5)
data5.boxplot()
plt.show()
```

分析图如下:


SciView:
Data
Plots
1:1
640x480 PNG (24-bit color) 10.98 kB

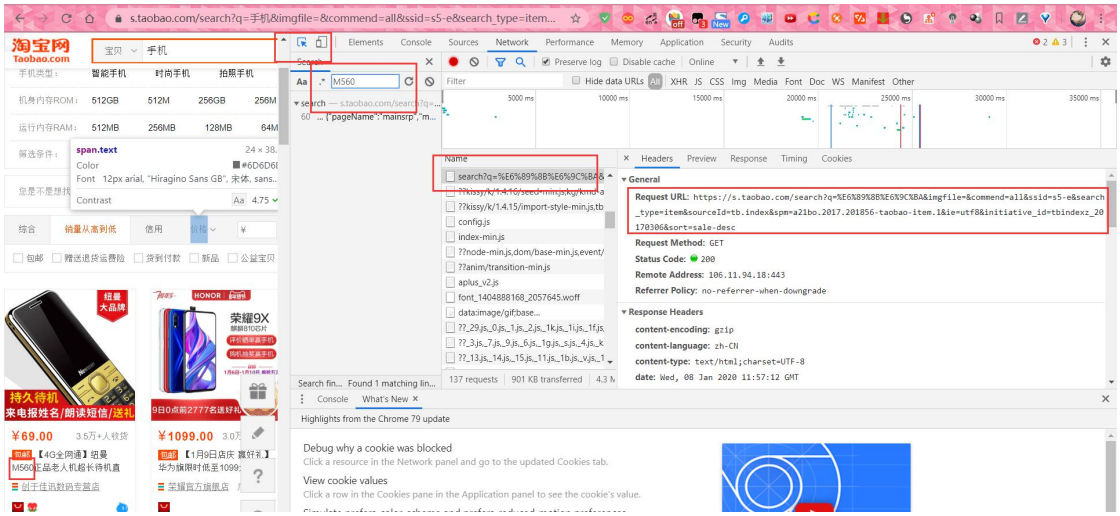

SciView:
Data
Plots
1:1
640x480 PNG (24-bit color) 10.33 kB
ShopNum
MinPrice
0
2000
4000
6000
8000
10000
2
4
6
8
10
12
14
SciView
Mongo Explorer
Database



三、淘宝版本代码说明与注解

淘宝部分就相对更加复杂了，主要是难在数据到底来自哪里和为什么要我滑块扫码...这两点上，由于代码思路和上面差不太多，这里就着重讲解不同的地方。我是先做的淘宝，当时还不太会 selenium...但之后才觉得，真香！

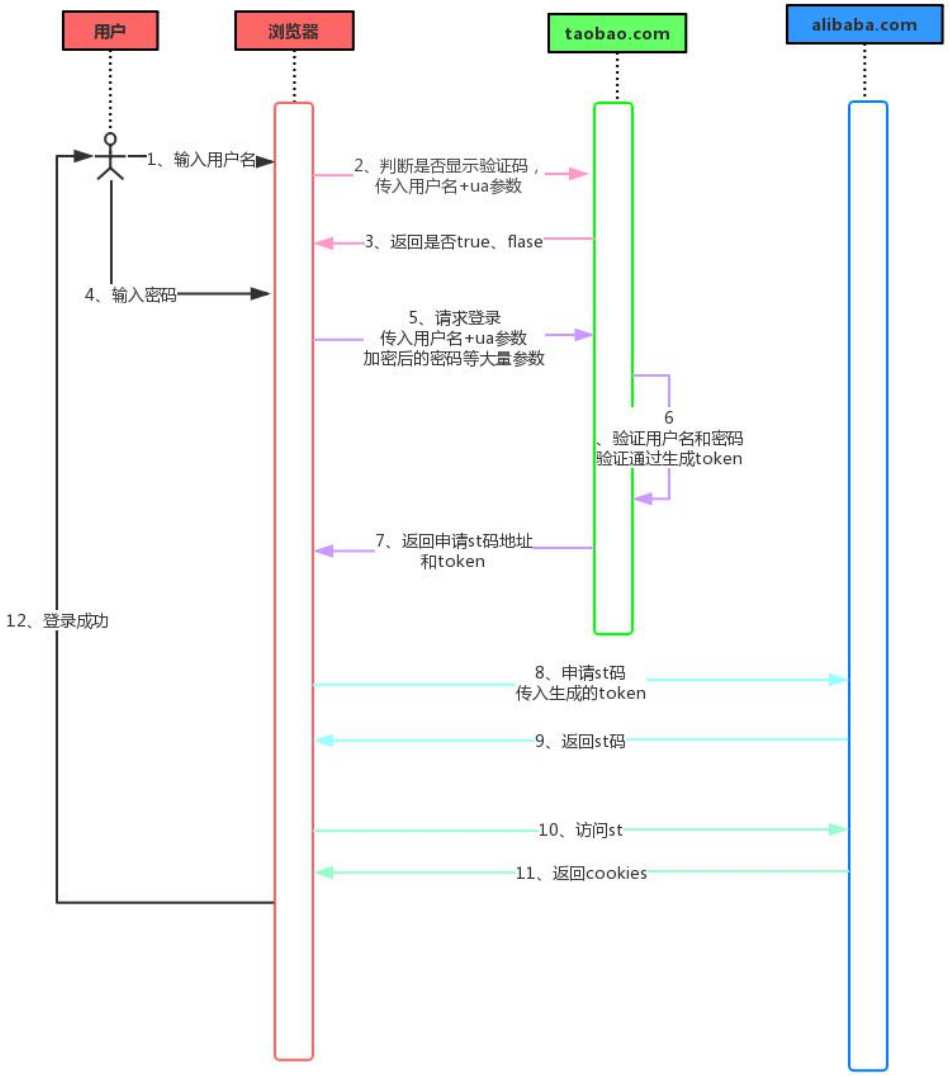
一、数据来源不同了



二、如何实现顺畅无阻的爬虫？

一种方法是使用临时 cookie 让程序员手动保存，但往往这个 cookie 的生命周期比较短暂，只有 5~15min 左右，于是我放弃了这种方法。另一种是用 selenium，上网查看了

官方文档与一些技术博客，我发现真香，但是期末周比较忙，所以当时就没去学 selenium，然后在 DDL 的今天才发现真好用。**当时我参考了一篇模拟登录的文章，来自一篇公众号——裸睡的猪，**这里简单介绍以下如何实现的模拟登录。



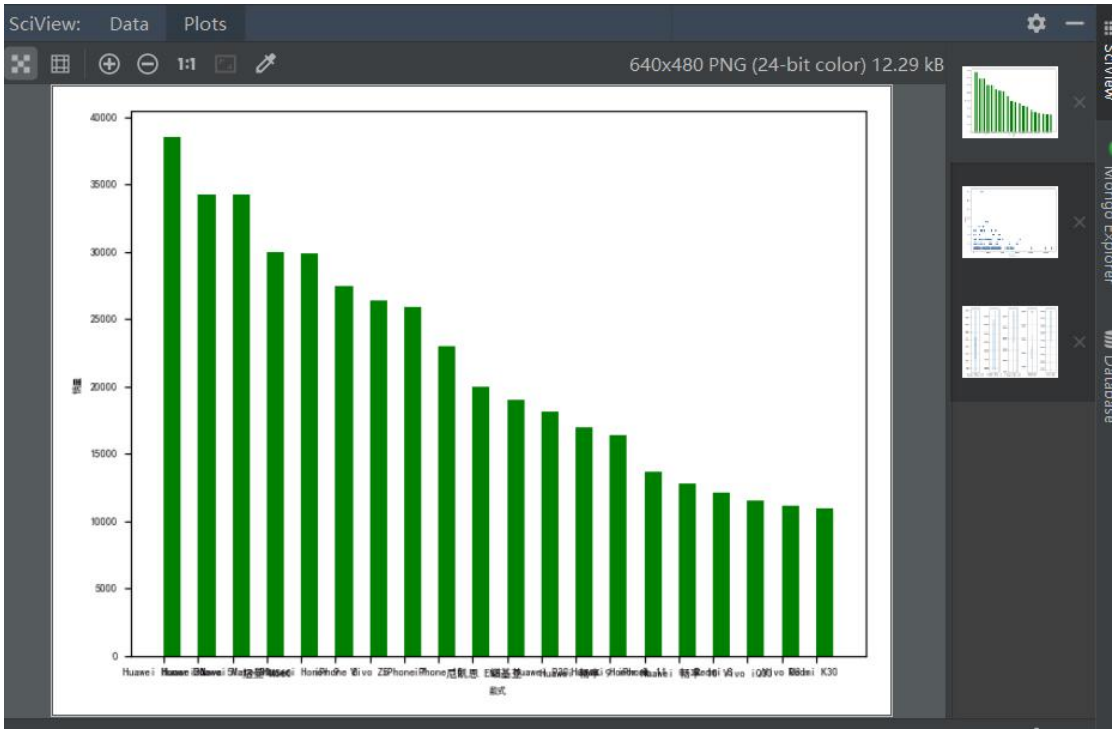
1.输入用户名后，浏览器会向淘宝（taobao.com）发起一个 post 的请求，判断是否出现滑块验证。
2.用户输入密码后，浏览器向淘宝（taobao.com）又发起一个 post 请求，验证用户名密码是否正确，如果正确则返回一个 token。
3.浏览器拿着 token 去阿里巴巴（alibaba.com）交换 st 码。
4.浏览器获取 st 码之后，拿着 st 码获取 cookies，登录成功。
不过这一点并不是大作业的核心，就不放出代码了，老师感兴趣可以去我上面的链接查看源代码或者去公众号看看相关的分析。
爬虫业务逻辑：

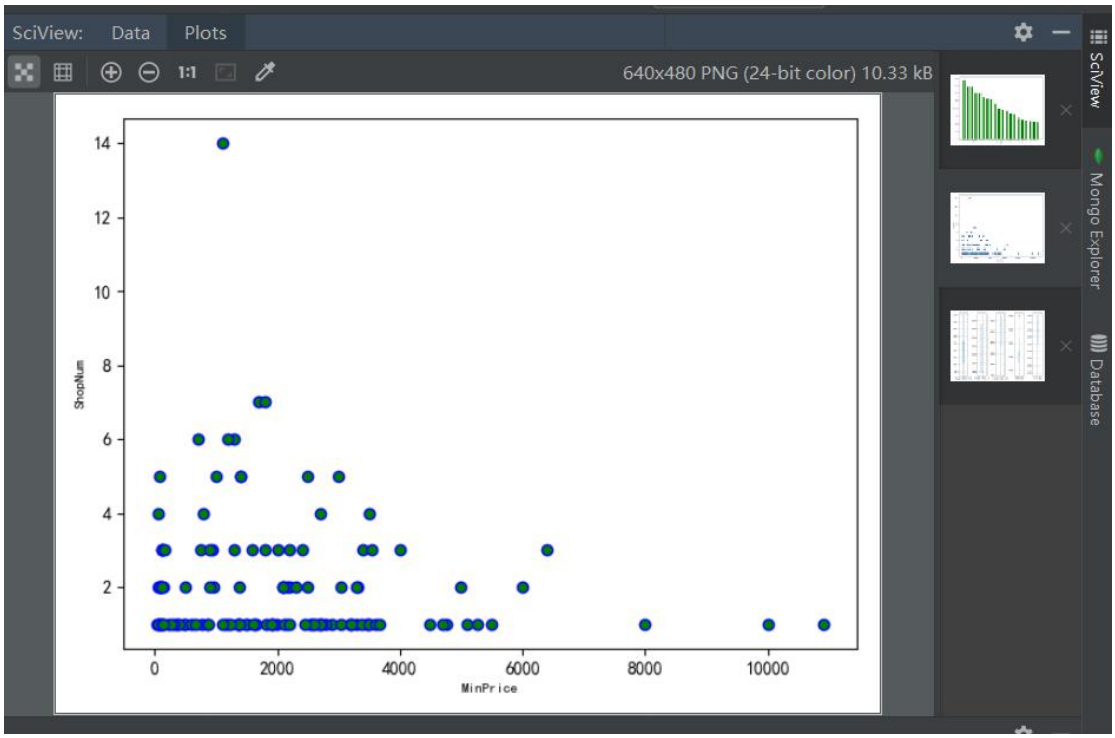
```
def spider_goods(self, page):
    s = page * 44
    # 搜索链接，q 参数表示搜索关键字，s=page*44 数据开始索引 一页 44 个数据

search_url=f'https://s.taobao.com/search?q={self.q}&imgfile=&commend=all&ssid=s5-e&s
```

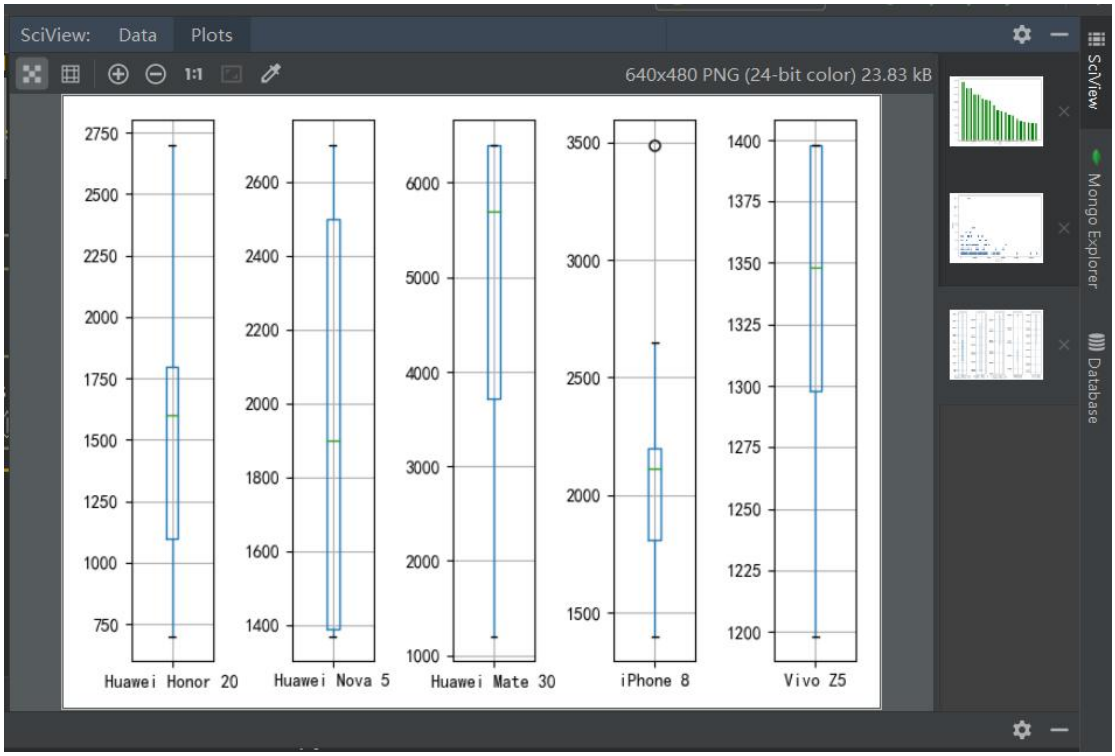
```
earch_type=item&sourceId=tb.index&spm=a21bo.2017.201856-taobao-item.1&ie=utf8&in
itiative_id=tbindexz_20170306&sort=sale-desc&s={s}&data-key=s&data-value={s + 44}'
    proxies = {'http': '10.28.234.209:8080',
               }
    # 请求头
    headers = {
        'referer': 'https://www.taobao.com/',
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36'
    }
    response = req_session.get(search_url, headers=headers, proxies=proxies,
                               verify=False, timeout=self.timeout)
    # print(response.text)
    goods_match = re.search(r'g_page_config = (.*?)}};', response.text)
    # 没有匹配到数据
    if not goods_match:
        print('提取页面中的数据失败！')
        print(response.text)
        raise RuntimeError
    goods_str = goods_match.group(1) + '}}'
    goods_list = self._get_goods_info(goods_str)
    self._save_excel(goods_list)
    # print(goods_str)

def _get_goods_info(self, goods_str):
    """
    解析 json 数据，并提取标题、价格、商家地址、销量、评价地址
    """
    goods_json = json.loads(goods_str)
    goods_items = goods_json['mods']['itemlist']['data']['auctions']
    goods_list = []
    for goods_item in goods_items:
        goods = {'title': goods_item['raw_title'],
                 'price': goods_item['view_price'],
                 'location': goods_item['item_loc'],
                 'sales': goods_item['view_sales'],
                 'comment_url': goods_item['comment_url']}
        goods_list.append(goods)
        print(goods)
    return goods_list
```

数据可视化和京东的可视化也基本没有区别，也不过多介绍了。

最后的结果分析图：


SciView:
Data
Plots
640x480 PNG (24-bit color) 12.29 kB
40000
35000
30000
25000
20000
15000
10000
5000
0
SciView
Mongo Explorer
Database


SciView:
Data
Plots
640x480 PNG (24-bit color) 10.33 kB
ShopNum
MinPrice
14
12
10
8
6
4
2
0
2000
4000
6000
8000
10000
SciView
Mongo Explorer
Database



附录：

两个电商站点前 1300 条左右的数据的 Top20:

淘宝:

第 1 名: Huawei Honor 20 ,评论数: 38481
第 2 名: Huawei Nova 5 ,评论数: 34269
第 3 名: Huawei Mate 30 ,评论数: 34254
第 4 名: 纽曼 M560 ,评论数: 30000
第 5 名: Huawei Honor 9 ,评论数: 29847
第 6 名: iPhone 8 ,评论数: 27466
第 7 名: Vivo Z5 ,评论数: 26365
第 8 名: iPhone 7 ,评论数: 25935
第 9 名: iPhone 10 ,评论数: 23016
第 10 名: 尼凯恩 EN3 ,评论数: 20000
第 11 名: 诺基亚 ,评论数: 19029
第 12 名: Huawei P30 ,评论数: 18102
第 13 名: Huawei 畅享 9 ,评论数: 16927
第 14 名: Huawei Honor 8 ,评论数: 16392
第 15 名: iPhone 11 ,评论数: 13668
第 16 名: Huawei 畅享 10 ,评论数: 12771
第 17 名: Redmi 8 ,评论数: 12107
第 18 名: Vivo iQOO ,评论数: 11567
第 19 名: Vivo U3 ,评论数: 11169
第 20 名: Redmi K30 ,评论数: 10898

京东:

第 1 名: 荣耀 8X ,评论数: 2154000
第 2 名: iPhone XR ,评论数: 2142200
第 3 名: iPhone 8 Plus ,评论数: 2026000
第 4 名: iPhone 8 ,评论数: 1561000
第 5 名: 荣耀 10 青春版 ,评论数: 1277000
第 6 名: iPhone XS Max ,评论数: 1160000
第 7 名: iPhone 7 Plus ,评论数: 1100000
第 8 名: iPhone 11 ,评论数: 1083500
第 9 名: iphone 7 ,评论数: 1070000
第 10 名: 红米 Redmi Note7 AI 双摄 6GB+64GB 亮黑色 全网通 4G 双卡双待 水滴全面屏拍照游戏智能手机 ,评论数: 1010000
第 11 名: E2 ,评论数: 913000
第 12 名: 荣耀 9X ,评论数: 865300
第 13 名: 红米 6 ,评论数: 820000
第 14 名: 荣耀 20i ,评论数: 748500
第 15 名: P30 Pro ,评论数: 731900
第 16 名: 荣耀 20 ,评论数: 723700
第 17 名: 畅享 9Plus ,评论数: 614400
第 18 名: 诺基亚 105 新 ,评论数: 580000
第 19 名: Redmi Note8 Pro ,评论数: 570800
第 20 名: Redmi Note8 ,评论数: 570300