期末大作业

一．研究内容

基于京东研究笔记本电脑相关因素对其销量的影响，从中找出大部分用户的需求并根据需求推荐商品，通过对评论数据的分析，给用户提供有效信息，帮助用户找出最优选择。

二．资料收集——基于网络爬虫

京东笔记本电脑分项指标的csv文件

三．技术要求

（1）需要使用Selenium库对京东网站关于笔记本电脑的信息进行爬取

（2）需要使用bs4库中的BeautifulSoup对网页信息进行查找

（3）需要使用re库和jieba库对已有数据进行分解和精确查找

（4）需要使用numpy或pandas的工具对已有数据进行清洗（如数据有不完整和缺失）和处理

（5）采用合适的算法并用python实现

（6）通过使用matplotlib, wordcloud绘制直观、易懂的图形来给出结论

（7）提供GUI图形接口（如本例中，让用户通过GUI接口个性化选择对笔记本电脑各方面的需求，并给出最优的笔记本电脑及其相关信息）。

（8）通过使用win32com库提供语音播报选项，给用户提供更详细的信息

（9）使用os库判断图片路径是否存在

笔记本电脑各项指标与销量相关性研究

摘要：

笔记本电脑一直是各类社会人士必不可少的工具，但在电商平台上（京东）有着琳琅满目的各类商品，在无法试用并且看到实物的情况下，怎样才能给大众提供最优选择呢？本课题主要研究笔记本电脑各项指标对销量的影响以及他们之间的关系，通过对京东网站上关于笔记本电脑的商品介绍信息，销量数据以及评论数据进行分析，利用自行设计的算法，从中挖掘出笔记本电脑各项指标对销量的影响以及什么样的笔记本电脑销量会更高，并以直观，可视化的形式给出结论。

1. 问题分析

通过观察京东给出的商品介绍与其他有关信息（例如是否属于京东自营商品，是否被评为京品电脑），得出有可能影响销量的因素有：品牌，型号，价格，二手与否，京东自营与否，触控与否，处理器，运存，硬盘容量，显卡性能，显卡类别（独显/集显），显卡型号，是否被评为京品电脑，屏幕大小，好评率，通过对这些因素与销量的相关性以及影响程度的大小，最终分析得出用户最信赖的品牌以及最钟爱的配置，从而给用户提出合理建议。 2. 模型建立（或其他）

3. 问题求解

3.1 数据的爬取

- 利用Selenium库模拟chrome浏览器的特性，结合webdriver实现对动态页面的顺利爬取
- 利用Selenium库可以实现点击的功能来使网页显示出原本隐藏的数据（如：销量数据）
- 通过对比，商品摆放页中商品的简短介绍相对来说比商品详情页中的规格与包装显示的具体信息更加详细（这是由于部分商品详情页中商品为多种选择，规格包装中只写一个系列商品的共同属性），因此选择对商品摆放页进行爬取。
- 对商品摆放页采用点击下一页的方式将每一页的网页代码写入文件，并用特定的分隔符分隔。（共50页，1970件商品）
- 通过点击“只看当前商品评价”，在商品详情页中通过bs4库中的BeautifulSoup实现元素查找，获取每件商品真实的好评率和销量数据（通过信息查找以及相关经历，对于购买后不做评价的用户，京东会采取一段之间之后自动默认好评的措施，因此把评论数作为销量）
- 通过查看网页中的文件找到评论数据文件，从其请求链接中发现规律，对其进行爬取（考虑到爬取的网页数量庞大，差评网页数量较少，以及时间成本，共对筛选出的336件商品进行了每件10页（中评5页，差评5页）的爬取。
- 为防止被反爬，设置关闭webdriver的一些功能来防止被识别，使用time库的sleep实现一张页面约停留40秒。

3.2 数据的查找与筛选

- 使用bs4库，通过查找特定的标签实现对收集到的商品摆放页中的信息进行大致分类（分类后的文件为`p-name.txt`，`p-shop.txt`，`p-icons.txt`，`p-commit.txt`，`p-price.txt`）
- 由于观察商品详情页发现：从商品摆放页中爬取的销量数据`p-commit.txt`并不是此商品本身的销量，而是一系列产品销量的总和，因此又对商品详情页进行了爬取（共1970页），具体过程见3.1数据的爬取。
- 从`p-shop.txt`文件中提取商品是否为二手，店铺是否属于京东自营的信息（通过判断关键字是否在字符串中）

- 从p-icons.txt文件中提取商品折扣券的信息，以便将来获取相对更真实的价格数据
- 从p-name.txt文件中提取商品的全部属性（使用re表达式以及一些算法判断条件从商品简介中分别筛选有效信息，滤去冗余信息。）
- 对于只有两种选择的商品属性（如：京东自营与否，二手与否，触控与否），采用是，否分别对应1，0来减小储存空间，并且简化数据，利于分析。
- 对于字符串类型的数据，尽量转化成相同的格式或者长度，并且减少缺失值的出现。
- 对于缺失值的处理：在生成csv文件summary.csv后，使用DataFrame.drop_duplicates()对数据进行去重，使用DataFrame.dropna()删除销量，品牌（有些明显为错误信息），好评度为缺失值的行，使用DataFrame.replace()对每一列中同一个值的不同表达方式进行统一化，减少结论误差；在对每个属性单独分析时去除其列中的缺失值，并将类型原本应该是数字的列实现相应转化。
- 对价格的处理：通过判断是否满足优惠券或满减规则对价格进行优惠后处理，考虑到京东的销量数据是从商品上架以来的销量总和，并且各商品不同时期的变化程度不同以及无法获取一段时间以来价格变化和销量变化的数据，所以不考虑优惠力度对销量的影响。

4. 数据的标准化（或规格化） 对于要比较的两组总数不同的数据采用百分化，便于后续作图比较（减少数据数值间差距过大对图像直观性的影响）

3.3 算法设计和实现（可包含代码和图）

- 商品摆放页数据的爬取————通过对页面元素的观察找到"下一页"按钮，在每次爬取后点击（为保证网页爬取的顺利性，在爬取过程中先不对标签进行筛选）

```
chrome_options = Options()
chrome_options.add_argument("--headless")
chrome_options.add_argument("--disable-gpu")
driver = webdriver.Chrome(chrome_options=chrome_options)
text = []
for page in range(50):
    if page==0:
        driver.get("https://search.jd.com/Search?
keyword=%E7%AC%94%E8%AE%B0%E6%9C%AC%E7%94%B5%E8%84%91&wq=%E7%AC%94%E8%AE%B0%E6%9C%
AC%E7%94%B5%E8%84%91&page=1&s=1&click=0")
    text.append(driver.page_source)
    next_page = driver.find_element_by_xpath("//span[@class='p-
num']//a[@class='pn-next']")
    next_page.click()
    sleep(2)
with open("jd_data.txt", "w",encoding="utf-8") as f:
    f.write(str(text))
```

- 对销量数据的爬取————通过打开页面，点击勾选"只看当前商品评价"实现；设置错误处理机制（由于商品摆放页信息的爬取早于销量数据的爬取约2天，有些商品已经下架，导致部分网页链接失效，约有6个）来处理找不到指定标签的情况（也就是链接失效）；由于数据量较多，所以分开几个文件共同储存，分别为jd_data20.txt，jd_data21.txt，jd_data22.txt，jd_data23.txt，jd_data24.txt。

```
option = ChromeOptions()
option.add_experimental_option('excludeSwitches', ['enable-automation'])
```

```
driver = Chrome(options=option)
text = []
with open("href.txt", "r", encoding="utf-8") as f:
    hrefs = list(eval(f.read()))
for href in hrefs[0:1970]:
    try:
        sleep(5)
        driver.get("https:" + href + "#comment")
        sleep(20)
        element = driver.find_element_by_xpath("//label[@for='comm-curr-sku']")
        ActionChains(driver).move_to_element(element).click().perform()
        sleep(12)
        comments_number =
driver.find_element_by_xpath("//li[@clstag='shangpin|keycount|product|allpingjia']
")
        good_percentage = driver.find_element_by_xpath("//div[@class='percent-
con']")
        with open("jd_data24.txt","a",encoding="utf-8") as f:
            f.write(str(comments_number.text))
            f.write(str(good_percentage.text))
            f.write("------")
    except:
        with open("jd_data24.txt","a",encoding="utf-8") as f:
            f.write(' ')
            f.write("------")
```

- 使用BeautifulSoup对网页数据进行标签查找，大致分类后分别写入文件`p-name.txt`, `p-icons.txt`, `p-price.txt`, `p-shop.txt`, `p-commit.txt`, `href.txt`

```
def process_data(tagname):

    tags_list=[]
    for html in htmls:
        soup = BeautifulSoup(html, "lxml")
        tag_all = soup.find("div", {"class": "goods-list-v2 gl-type-1 J-goods-
list", "id": "J_goodsList"})
        if tagname == "href":
            tags = tag_all.find_all("div", {"class": "p-name"})
            for tag in tags:
                tags_list.append(tag.find("a")["href"])
        else:
            tags = tag_all.find_all("div", {"class": tagname})
            for tag in tags:
                processed_tags = tag.text.replace("\n", "").replace("￥",
"").replace("¥", "")
                if tagname=='p-price':
                    tags_list.append(processed_tags[0:7])
                else:
                    tags_list.append(processed_tags)
    with open(tagname+".txt", "w", encoding="utf-8") as f:
        f.write(str(tags_list))
```

```
    print(len(tags_list))

if __name__ == "__main__":
    with open("jd_data.txt", "r",encoding="utf-8") as f:
        htmls=f.read()
    htmls = list(eval(htmls))
    tag_finded = ["p-name", "p-icons", "p-price", "p-shop", "p-commit", "href"]
    for tag in tag_finded:
        process_data(tag)
```

- 通过构建类对不同的情况进行分类，以便实现代码复用，对每个文件的数据进行精准提取并创建新的数据分类文件；利用jieba分词功能找出品牌信息，利用正则表达式筛选出型号，折扣，运存，硬盘容量，显示器类型，显存，显示器型号，是否自营，是否二手，是否被认定为京品电脑，处理器，处理器版本，是否触控，屏幕大小。使用`replace()`将数据统一化，均以字符串列表的形式储存的文件中（便于后期数据汇总），分别为`brands.txt`，`models.txt`，`money_off.txt`，`discount.txt`，`internal_storage.txt`，`solid_storage.txt`，`displayer.txt`，`video_memory.txt`，`display.txt`，`logistic.txt`，`used.txt`，`Jingdong.txt`，`processor.txt`，`version.txt`，`touch_control.txt`，`screen_size.txt`，具体代码见文件`seperate_name_data.py`。
- 对销量数据的处理————对销量数据中的加号进行忽略，对“万”进行替换（变为乘10000），并分别将销量和好评率写入两个文件`sale.txt`和`percentage.txt`，详见`sale.py`
- 将以上所有文件的数据整合生成文件`summary.csv`
- 对文件缺失值，冗余行，品牌相同值的删除与替换，并重新将索引排序并整合生成新的文件`summary2.0.csv`

```
import pandas as pd
import numpy as np
df = pd.read_csv("summary.csv", encoding="utf_8_sig")
df.Price = df.Price.astype(int)
df.drop_duplicates(inplace=True)
df.Brands=df.Brands.replace('机械','机械革命')
df.Brands=df.Brands.replace('APPLE','Apple')
df['Solid State Storage']=df['Solid State Storage'].replace('128GG+1T','128G+1T')
df.money_off=df.money_off.replace(' ','')
df.discount=df.discount.replace(' ','')
df.Sale=df.Sale.replace(' ',np.nan)
df['Favorable rate']=df['Favorable rate'].replace(' ',np.nan)
df.Brands=df.Brands.astype(str)
for index,brand in enumerate(df.Brands):
    if brand.isalpha()==True:
        df.iloc[index,0]=brand
    else:
        df.iloc[index,0]=np.nan
df=df.dropna(axis=0,subset=["Brands",'Sale','Favorable rate'])
df.Sale=df.Sale.astype(int)
df['Favorable rate']=df['Favorable rate'].astype(int)
df=df.drop_duplicates(['Brands','Model','Price','Internal Storage','Solid State
Storage','Display','Shop','Display_type','Video Memory','Processor','Screen
size','Sale'],keep='last')
df = df.reset_index(drop=True)
```

- 使用聚类分析补充部分缺失值：通过将同一类字符串用同一个数字代替将需要分析的数据都变成数字形式以满足聚类分析要求，目标是分析三个变量（处理器，运存，固态硬盘储存，价格）之间的关系并且补充缺失值，因此分别取其中一个作为训练标签，其他三个作为训练集来分析，结果是运存受处理器，固态硬盘储存以及价格的影响最大（数据集正确率约为70%），而硬盘储存和处理器受其他因素影响并不大（正确率约为50%~60%）

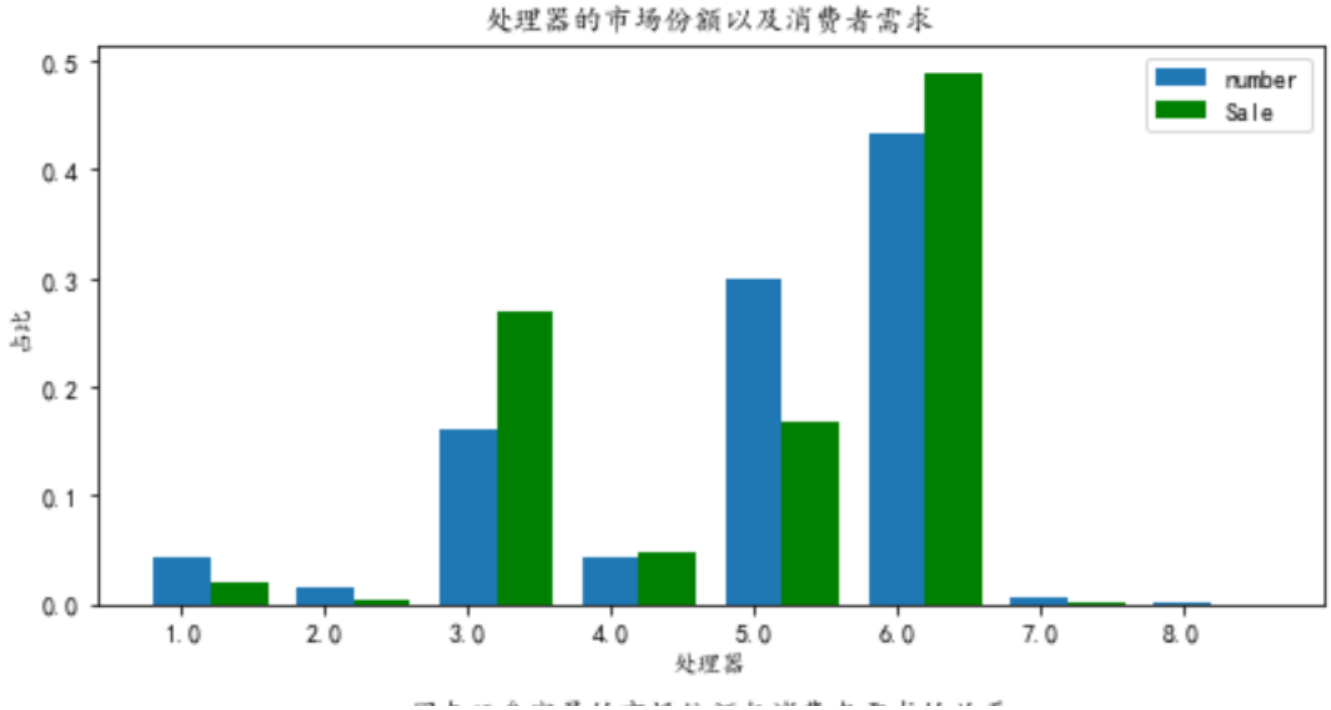
```
import pandas as pd
from sklearn import neighbors
from sklearn.model_selection import train_test_split
import sklearn

data = pd.read_csv(r"summary4.0.csv",header=None,names=
['Brands','Model','Price','money_off','discount','Internal Storage','Solid State
Storage','Display','Shop','Shop Type','Video
Memory','JingDong','href','used','Display_type','Processor','Version','Touch
control','Screen size','Sale','Favorable rate'])
df1=pd.DataFrame(data)
df1=df1[(df1['Internal Storage'].isnull()==False)&
(df1['Processor'].isnull()==False)&(df1['Solid State Storage'].isnull()==False)]
df1['Internal Storage']=df1['Internal Storage'][1:].str[:-1].astype(int)
df1['Solid State Storage']=df1['Solid State Storage'][1:].str[:-1].astype(int)
df1=df1.reset_index(drop=True)
for index,i in enumerate(df1['Processor']):
    if i=='i3':
        df1['Processor'][index]=1
    elif i=='R3':
        df1['Processor'][index]=2
    elif i=='R5':
        df1['Processor'][index]=3
    elif i=='R7':
        df1['Processor'][index]=4
    elif i=='i7':
        df1['Processor'][index]=5
    elif i=='i5':
        df1['Processor'][index]=6
    elif i=='i9':
        df1['Processor'][index]=7
    elif i=='R9':
        df1['Processor'][index]=8
data=df1.iloc[1:,[2,5,6]]
labels=df1.iloc[1:,-6].astype(int)
clf = neighbors.KNeighborsClassifier(40, weights='distance')
train_X , test_X, train_y ,test_y = train_test_split(data,labels)
clf.fit(train_X ,train_y)
print(clf.score(test_X,test_y))
for i in range(df.shape[0]):
    if pd.isna(df.iloc[i,-6]) and not pd.isna(df.iloc[i,5]) and  not
pd.isna(df.iloc[i,6]):
        df.iloc[i,-6]=clf.predict([[df.iloc[i,2],int(df.iloc[i,5]
[:-1]),int(df.iloc[i,6][:-1])]])
```

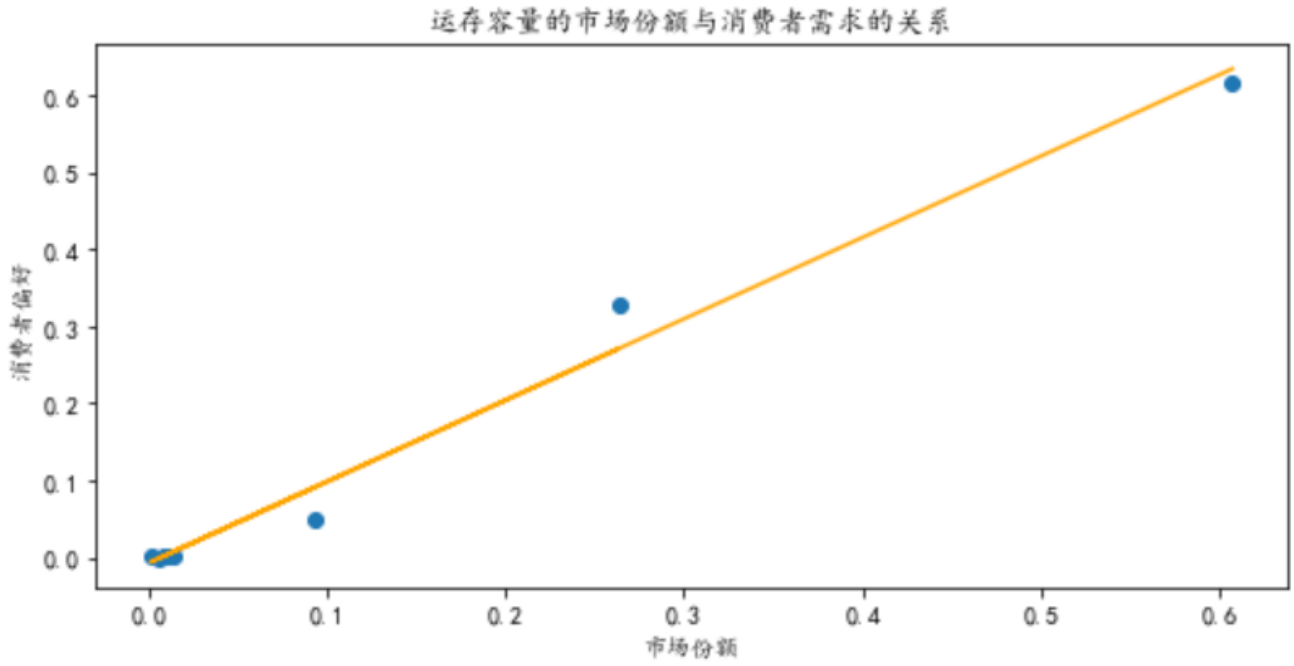
3.3 数据可视化处理（可包含代码和图）

- 各变量市场份额与实际销量的关系统计图绘制————柱形图大多采用数据百分化的方式（见文件数据分析和部分爬虫.ipynb）

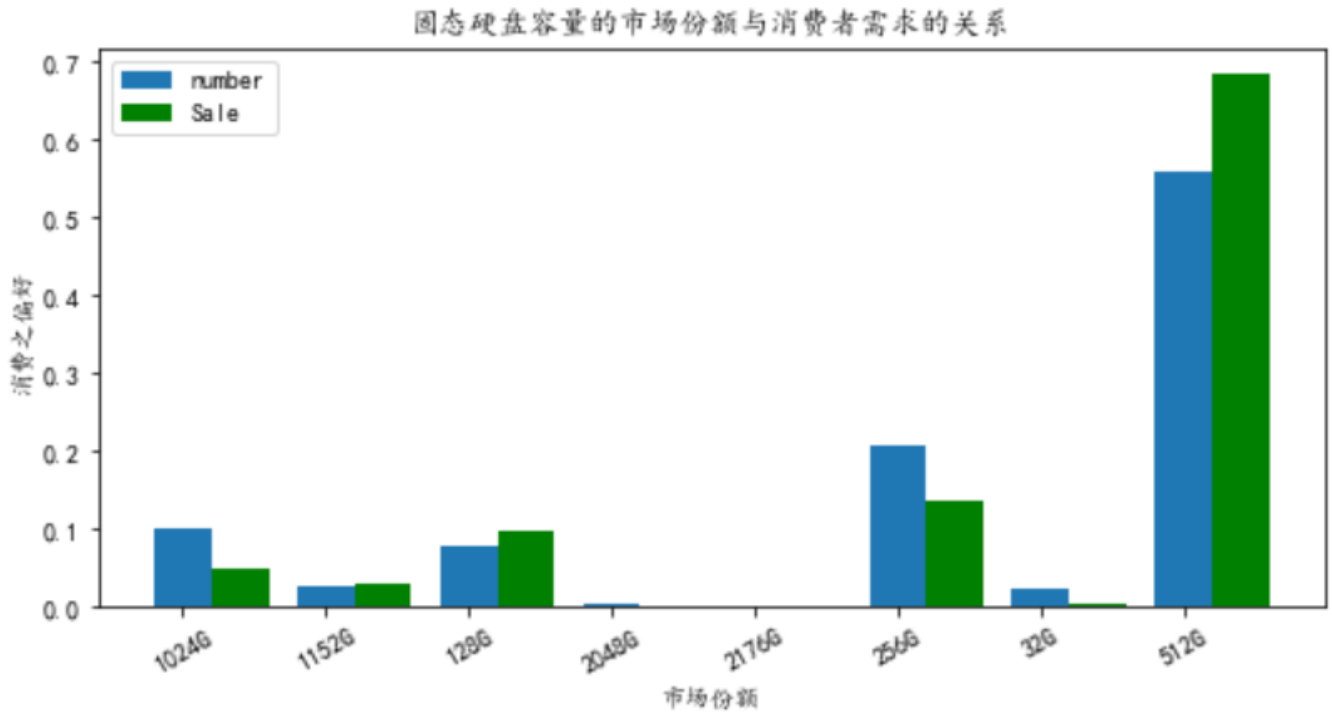
1. 由此图可得，大众更偏向于i5和处理器，并且R5处理器供不应求，说明大众越来越能够接受锐龙系列的处理器；相反，i7这样更高端的处理器却没有R5受欢迎程度高



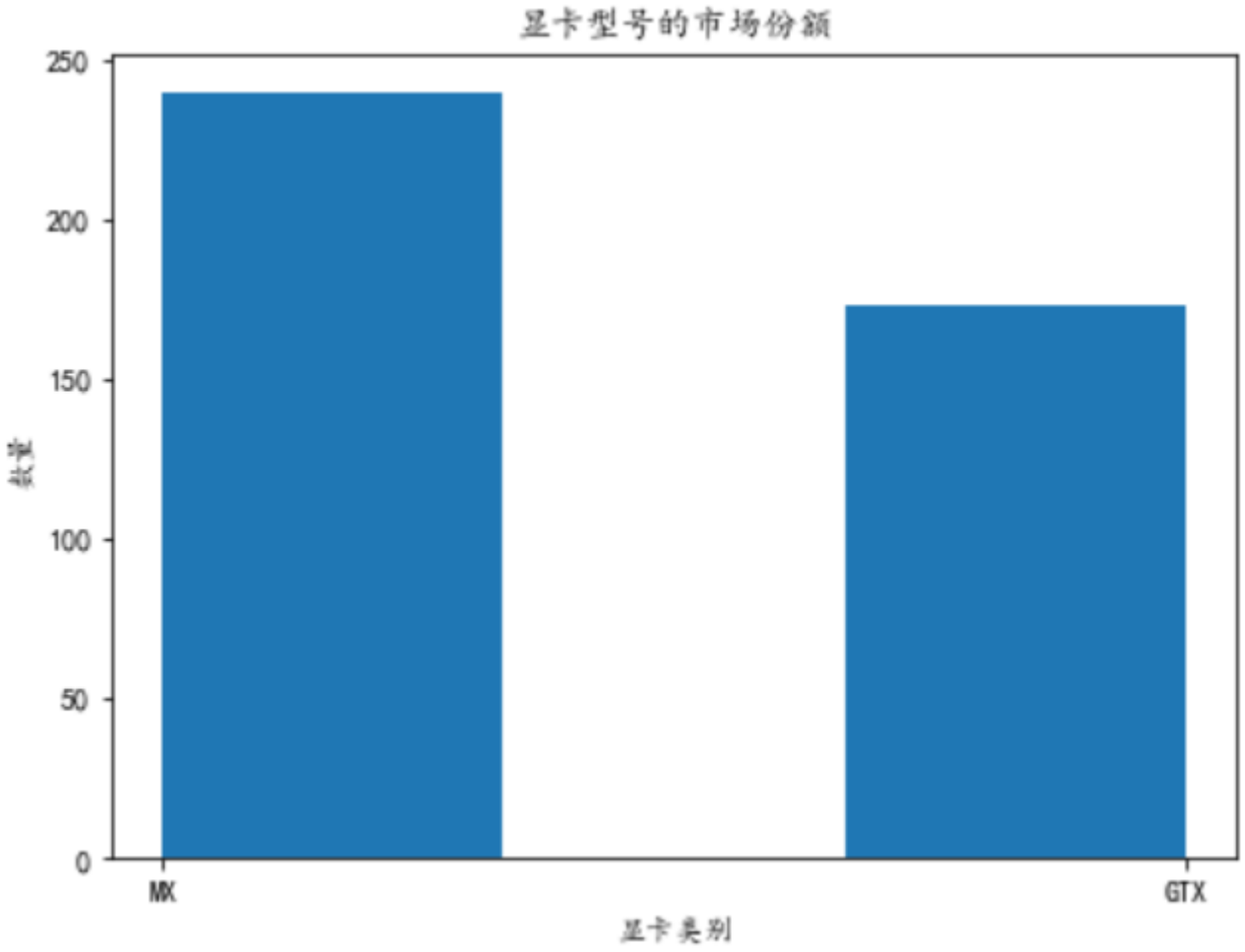
2. 由此图可以看出，经过拟合，市场份额占比大的销量也会越高



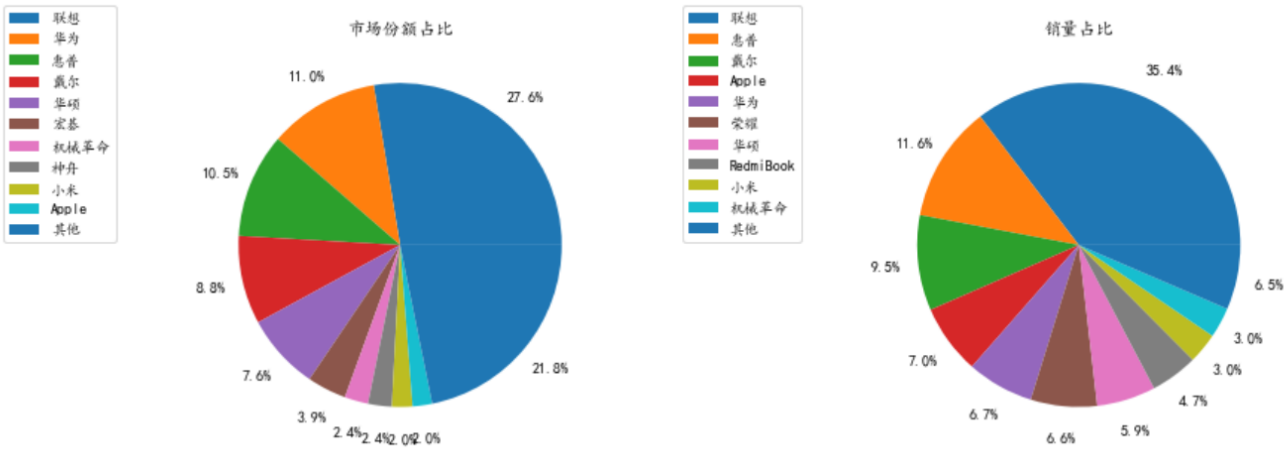
3. 由此可以看出512G的固态硬盘最受欢迎



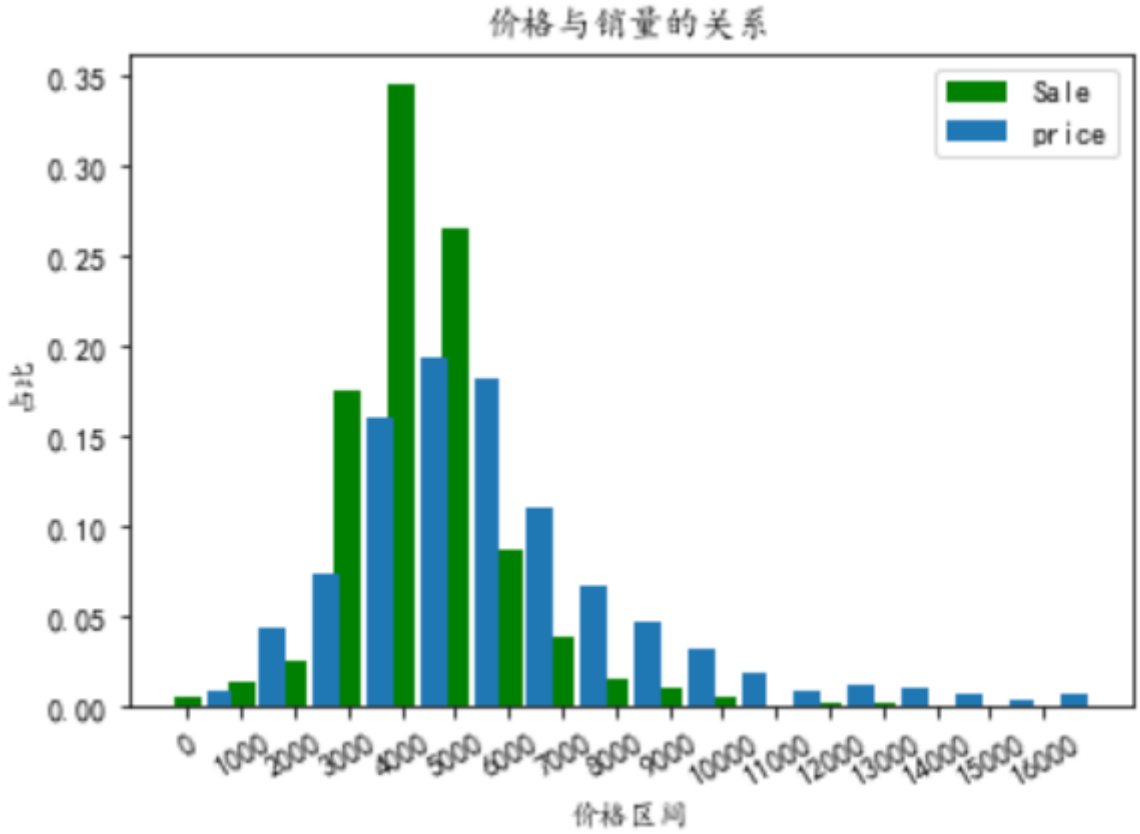
4. 通过直方图可得MX系列的显卡占市场份额更高，但是由于缺失数据较多，所以后面在提供给用户的选项中不做研究



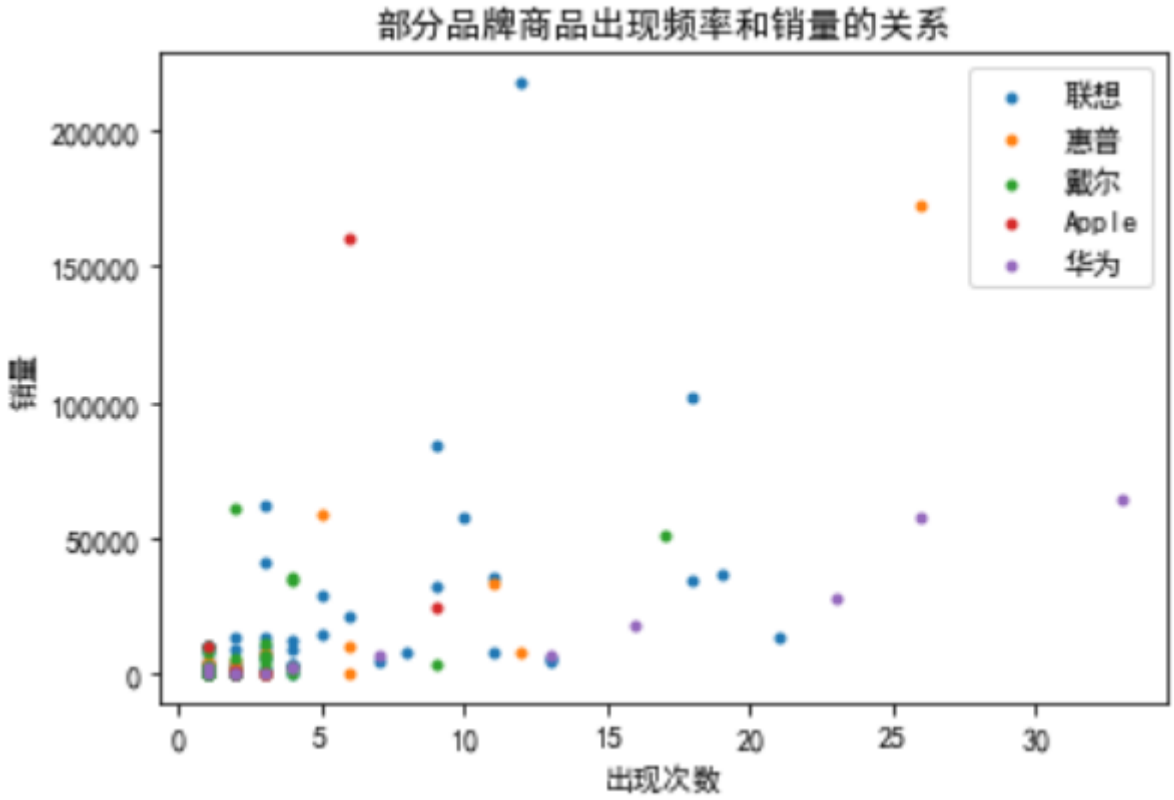
- 品牌市场份额和大众喜爱度对比：通过以下两幅图可以得到，虽然有些品牌的商品在市场中出现频率较高，但是销售量却不是很高，因此可以看出用户对品牌的信赖程度较强。



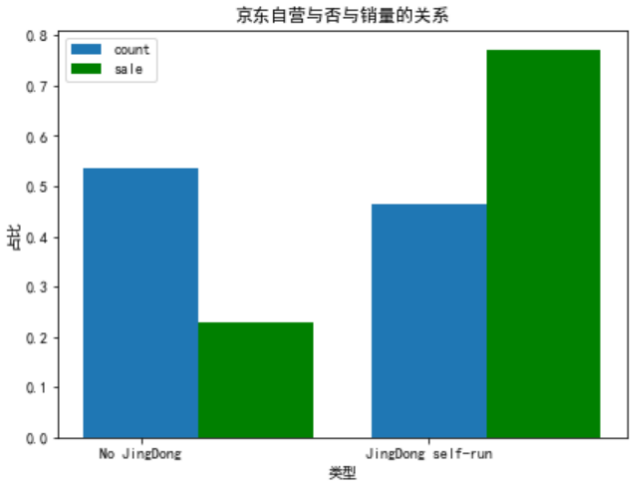
- 价格的市场份额以及群众接受度：通过这个图可以得到，用户最能接受的价格区间在3500-5500之间，与市场份额趋势大致相同，在4500左右销量极高

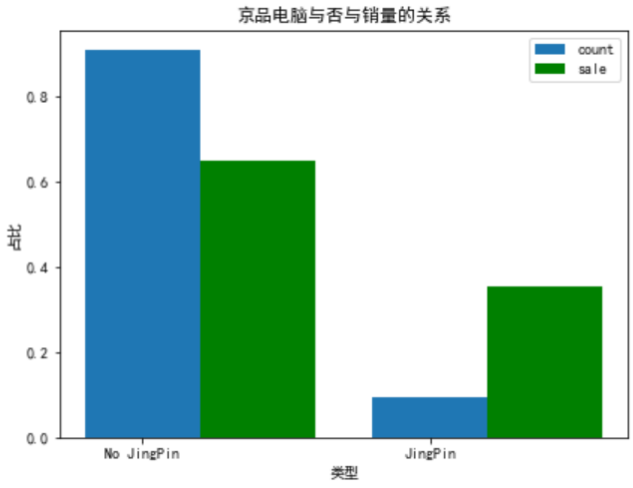


- 进一步分析同一品牌商品出现频率和销量的关系：从图中易看出数据杂乱分布，因此商品出现的频率对销量没有太大影响。

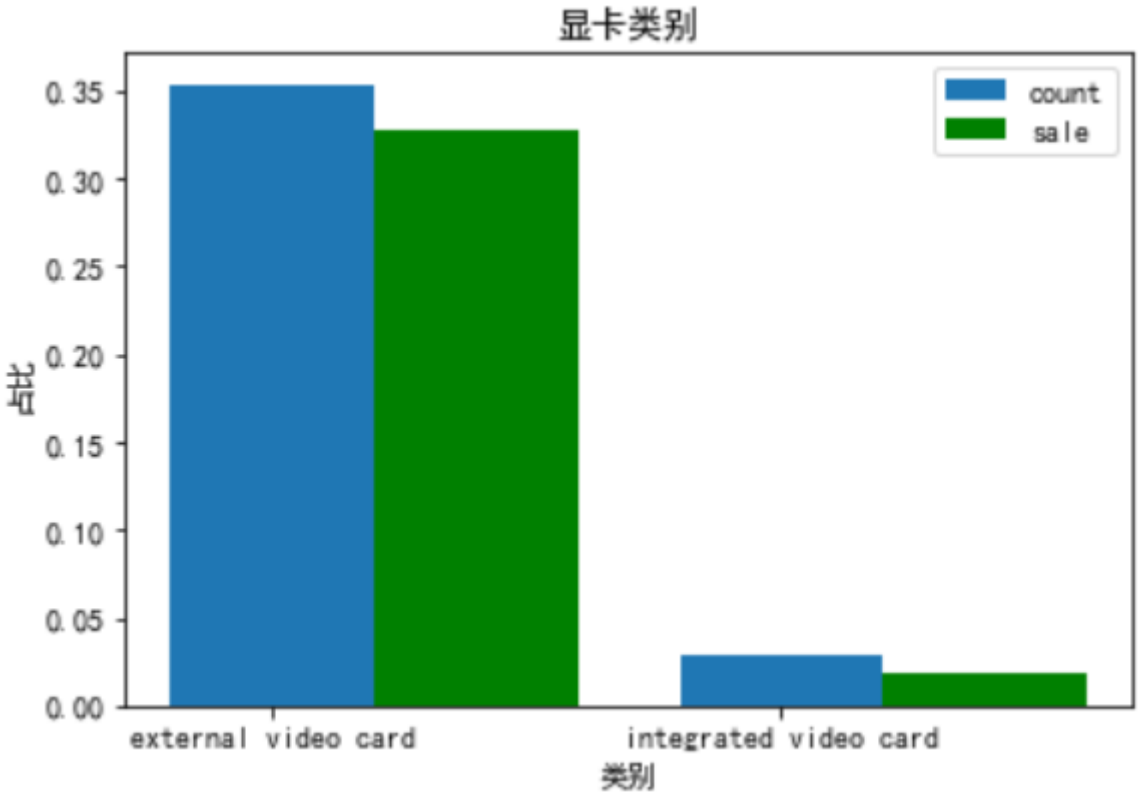


- 是否为京东自营或者京品电脑和销量的关系：以下两图足以看出消费者对京东的信赖程度（无论是在物流还是品质方面），因此在消费者对物流不做要求时首先考虑京东物流





- 是否为独显对销量的影响：从图中可以看出，大多数人都偏向于独显

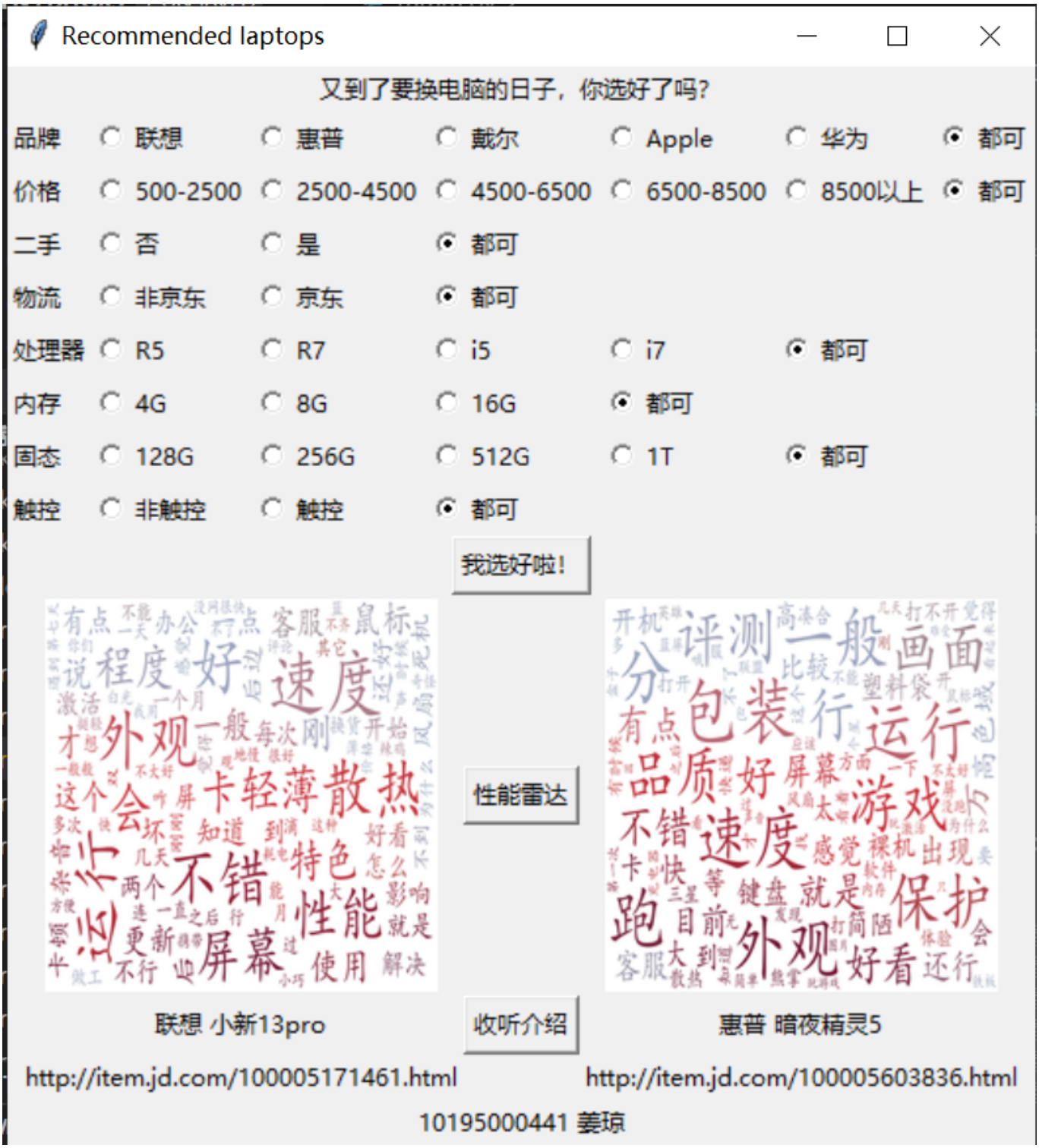


3.4 分析评论数据

- 通过前面的分析，得出满足大多数用户条件的性能和品牌需求，之后在GUI中给用户提供选择，根据用户选择和分析结果综合考虑给用户推荐最合适的产品；通过对GUI每个选项的循环组合配对，找到了每种选择对应的最优商品，并且通过程序找到他们的索引（共计336件），之后再对评论信息进行爬取，整合差评和中评，经过使用jieba库的分词和worcloud库生成词云，展现给用户，让用户真正意识到产品的问题所在，而不是被好评蒙蔽双眼（对部分评论爬取的文件见`comments.py`，中评见`comment_median.txt`，差评见`comment_bad.txt`，综合所有评价见`summar_comment.txt`）

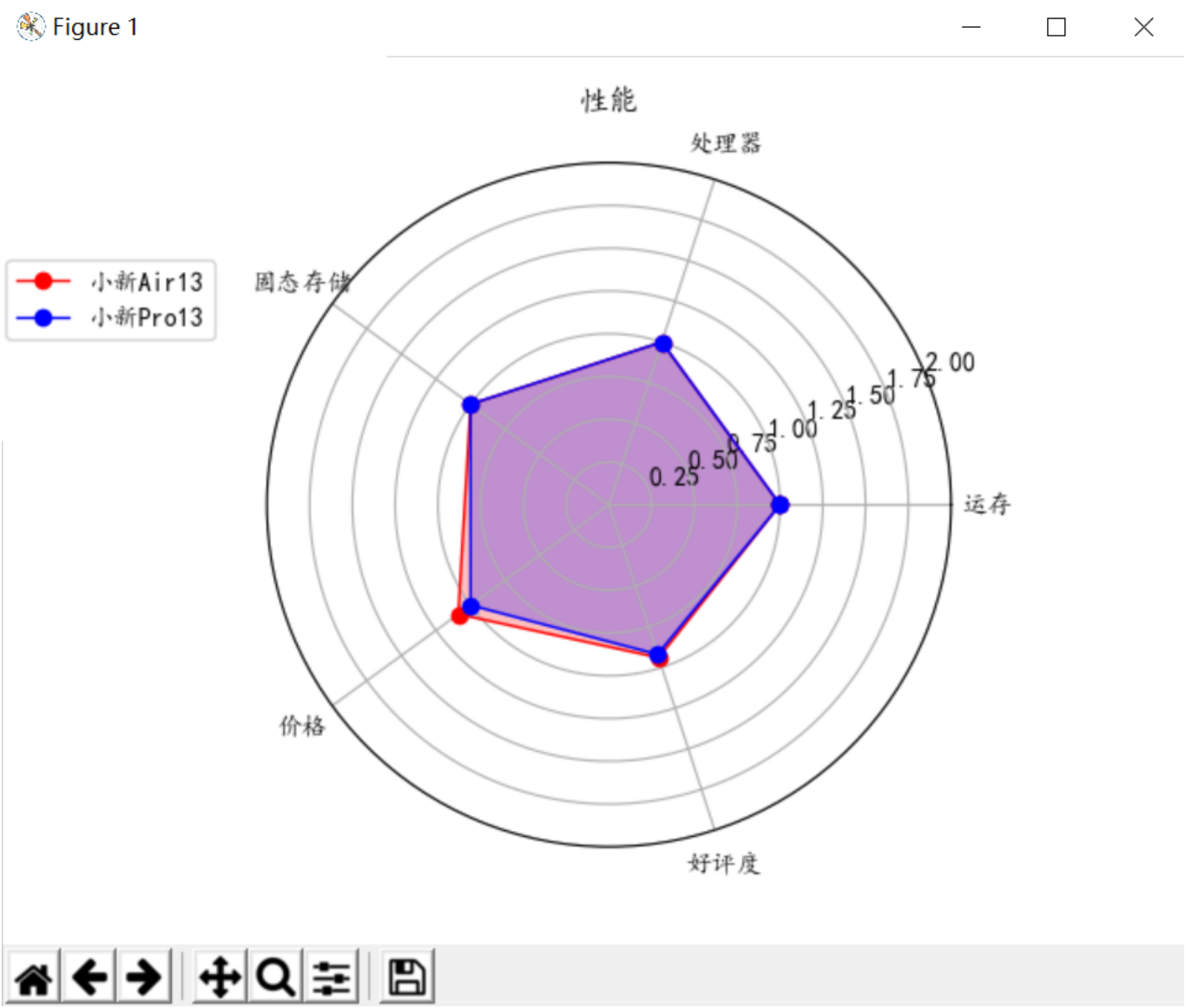
对GUI的展示

1. 用户初始选择界面——第一次展示的商品是分析后筛选出对于大众的最优选择



2. 点击"我选好啦"按钮会按照选择自动更新词云（词云的生成见文件`wordcloud1.py`，由于词云文件是事先生成的，所以需要提前下载所有图片）

3. 点击“性能雷达”可以显示出性能比对图（其中默认中间的位置是大众所需）



4. 点击“收听介绍”可以听到讲解，此处无法显示图片，程序自带声音

4 结论

通过图像可以得出：

- 消费者更倾向于的消费区间是3500-5500，更偏爱的处理器是i5（R5还有一定的提升空间），在当今时代，人们追求更快的运行速度，对运存（大多数人偏向于8G，还有很多人偏向于16G）和硬盘储存（大多数人偏向于512G）的需求都较高；在物流方面，人们对京东自营的信任远超过非自营，对京品电脑的信赖程度超过市场供给量，可见选择京东购物的用户对京东的信任