Data science programming of Taobao Commodity Based on Python

Group W
MB95510 Ren Xuguang and MB95525 Bao Rong

1. Introduction

1.1 Object

(1) Taobao commodity: sofa

(2) Quantity: 4400 items in 100 pages

(3) Screening Conditions: TMall, Sales Volume from High to Low, Price above 500 yuan.

1.2 Purpose

(1) Text analysis of commodity title, word cloud visualization; (2)Statistical analysis of sales corresponding to different keywords word; (3)Analysis of commodity price distribution; (4)Analysis of sales distribution of goods; (5)Average sales distribution of commodities in different price ranges; (6)Analysis of the impact of commodity price on sales volume; (7)Analysis of the impact of commodity price on sales; (8)Average sales distribution of goods in different provinces.

1.3 Steps and codes

(1) Data collection: Python crawls commodity data of taobao.com (2) Data cleaning and processing (3) Text analysis: Jieba segmentation, wordcloud visualization (4) Visualization of data histogram (5) Histogram visualization hist (6) Scatter visualization (7) Data regression analysis visual regplotv

1.4 Libraries

Requests, Retrying, Re, Missingno, Jieba, Matplotlib, Wordcloud, Imread, Seaborn, Pyecharts, etc

2. Collect the Data by Web Crawler

2.1 Basic process: send requests; get the corresponding content; interpret HTML content: use Re library or third-party library; export data as Excel document.

2.2 Detailed process:

2.2.1 Import libraries

2.2.2 Requests (cookie, user-agent)

https://world.taobao.com/robots.txt

```
User-Agent: *
Disallow: /
```

Unfortunately, according to the Robots Exclusion Standard, the search pages of Taobao does not allow crawling. Our project just discuss technology for academic purpose and ensure non-commercial use. With many strategies for coping with anti-spider, we successfully anti anti-scraping and got the data. How to deal with the anti-scraping mechanisms was the biggest challenge for our project.

(1) Time our program using time.clock() to find out how long it takes to execute our solution. (2) We observed that there were 44 products shown in every webpage and then we decided to crawl 1-100 pages. (3) Retrying is a general-purpose retrying library, so we use it to simplify the task of adding retry behavior. (4) Using requests library: make requests using HTTP methods such as GET. (5) Map: similar to map in python but the iterables are collected immediately rather than lazily, func is executed asynchronously and several calls to func may be made concurrently. With a lot of strategies

```
start = time.clock() # Time counting starts

plist = [] # plist is the Number of URLs on pages 1-100
for i in range(1, 101): # the range of webpage number
    j = 44 * (i-1) # 44 products shown in every webpage
    plist.append(j)

listno = plist
datatmsp = pd.DataFrame(columns=[])

while True:
    @retry(stop_max_attempt_number = 8) # Setting the max number of retrying
    def network_programming(num):
        url = 'https://s.taobao.com/search?initiative_id=tbindexz_20170306 \
        &ie=utf8&spm=a21bo.2017.201856-taobao-item.2&sourceId=tb.index \
        &search_type=item&ssid=s5-e&commend=all&imgfile=&q=%E6%B2%99%E5%8F%91 \
        &suggest=history_1&_input_charset=utf-8&wq=shafa&suggest_query=shafa \
        &source=suggest&bcoffset=4&ntoffset=4&p4ppushleft=1%2C48&s=' + str(num)
        web = requests.get(url, headers=headers) #using Requests library
        web.encoding = 'utf-8'
        return web
```

```
# Multithreading
def multithreading():
    # the unsuccessfully-crawled webpages everytime
    number = listno
    event = []

    with ThreadPoolExecutor(max_workers=10) as executor:
        for result in executor.map(network_programming, number, chunksize=10):
            event.append(result)
    return event
```

(6) Customize our requests by modifying headers: Because there may be anti-crawlers on Taobao, it is necessary to use cookies to build the header to pretend to be a browser.

```
# Hiding: modifying the headers.
headers = {'User-Agent':'Mozilla/5.0 (X11; Linux x86_64) \
    AppleWebKit/537.36 (KHTML, like Gecko) \
    Chrome/64.0.3282.167 Safari/537.36'}
```

2.2.3 Regular expression (Re library)

(1) Regular expression can be used to check if a string contains the specified search pattern and findall() function returns a list containing all matches.

```
# Getting data from Json by multithreading
listpg = []
event = multithreading()
for i in event:
    json = re.findall('"auctions":(.*?),"recommendAuctions"', i.text)
    if len(json):
        table = pd.read_json(json[0])
        datatmsp = pd.concat([datatmsp, table], axis=0, ignore_index=True)

        pg = re.findall('"pageNum":(.*?),"p4pbottom_up"', i.text)[0]
        listpg.append(pg)
        # #Recording the page numbers of successfully-crawled webpages everytime
```

(2) Turning the page numbers of successfully crawled webpages into the num value in the URL. (3) Recording the page numbers of unsuccessfully crawled webpages in the list for cyclic crawling. (4) Stop cycling when the number of unsuccessfully crawled webpages is 0.

```
lists = []
for a in listpg:
    b = 44 * (int(a)-1)
    lists.append(b)

listn = listno

# Clearing the failed array every cycle
listno = []
for p in listn:
    if p not in lists:
        listno.append(p)

if len(listno) == 0:
    break
```

2.2.4 Export

Finally, we export the data as Excel document.

```
datatmsp.to_excel('./data/datastmsp.xls', index=False)
# Exporting data to excel

end = time.clock() # Time counting stops
print("爬取完成，用时: ", end-start, 's')
```

3. Data Cleaning and Data Processing

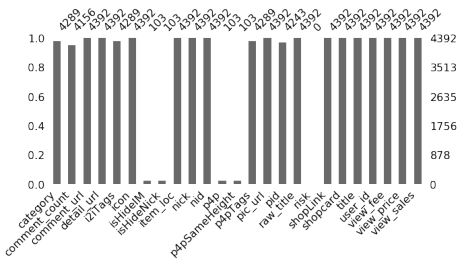
3.1 Reading data and general understanding.

3.2 Using visualization tool (Missingno library) to observe data loss.

3.3 Using data.dropna() to delete columns with more than half of the missing values.

3.4 Deleting duplicated rows by pandas.DataFrame.

```
In [4]: datatmsp = pd.read_excel('./data/datastmsp.xls') #Reading data

In [5]: datatmsp.shape
Out[5]: (4392, 25)

In [6]: import missingno as msno
        # Missingno: visualization tool to observe data loss
        msno.bar(datatmsp.sample(len(datatmsp)), figsize=(10, 4))

        # Deleting columns with more than half of the missing values
        half_count = len(datatmsp) / 2
        datatmsp = datatmsp.dropna(thresh = half_count, axis = 1)

        # Deleting duplicated rows
        datatmsp = datatmsp.drop_duplicates()
```



3.5 Check the first five data

3.6 According to the purpose of this project, the following attributes are needed for this analysis: item_loc, raw_title, view_price, view_sales

```
# Taking out these four columns of data
data = datatmsp[['item_loc', 'raw_title', 'view_price', 'view_sales']]
data.head()
```

	item_loc	raw_title	view_price	view_sales
0	广东 佛山	北欧风格小户型实木架客厅L型布艺沙发组合	3480.0	328人付款
1	广东 佛山	凯哲家具欧式真皮沙发 头层牛皮雕花实木客厅整装转角沙发组合	7400.0	111人付款
2	广东 佛山	雅居汇 欧式真皮转角沙发实木雕花客厅家具组合美式真皮转角沙发	18800.0	25人付款
3	广东 江门	斯帝罗兰简约现代真皮沙发组合整装客厅转角头层牛皮三人沙发家具	8699.0	93人付款
4	浙江 杭州	！【新款】现顾家kuka北欧布艺沙发客厅大小户型现代简约组合2037	2999.0	142人付款

3.7 Organizing these four kinds of data.

Firstly, split the provinces and cities listed in item_loc to get two columns: 'province' and 'city'. Special case: Since the provinces and cities of the municipality directly under the central government are the same, it was judged according to the character length. Secondly, extract the numbers in the 'view_sales' column to get the 'sales' column. Thirdly, view data type.

```
# Spliting the provinces and cities listed in item_loc to get two columns: 'province' and 'city'
data['province'] = data.item_loc.apply(lambda x: x.split()[0])

data['city'] = data.item_loc.apply(lambda x: x.split()[0] if len(x) < 4 else x.split()[1])

# Extracting the numbers in the 'view_sales' column to get the 'sales' column
data['sales'] = data.view_sales.apply(lambda x: x.split('人')[0])

# Viewing data type of every column
data.dtypes
```

3.8 Convert data types as category data type in order to save time and space as well as improve computing efficiency. Then delete unused data columns.

```
# Converting data types
data['sales'] = data.sales.astype('int')
list_col = ['province', 'city']
for i in list_col:
    data[i] = data[i].astype('category')

# Deleting unused columns
data = data.drop(['item_loc', 'view_sales'], axis=1)
```

3.9 After that, we successfully cleaned and organized the data via Web crawler. The demo was as following:

```
data.head()
```

	raw_title	view_price	province	city	sales
0	北欧风格小户型实木架客厅L型布艺沙发组合	3480.0	广东	佛山	328
1	凯哲家具欧式真皮沙发 头层牛皮雕花实木客厅整装转角沙发组合	7400.0	广东	佛山	111
2	雅居汇 欧式真皮转角沙发实木雕花客厅家具组合美式真皮转角沙发	18800.0	广东	佛山	25
3	斯帝罗兰简约现代真皮沙发组合整装客厅转角头层牛皮三人沙发家具	8699.0	广东	江门	93
4	！【新款】现顾家kuka北欧布艺沙发客厅大小户型现代简约组合2037	2999.0	浙江	杭州	142

4. Data Mining and Analysis

4.1 Use word breaker jieba library and achieve Word segmentation for each title by lcut function.

4.2 Filter every element (str) in title_s (list of lists form) to remove unnecessary words from the stop words list then import the stopwords list.

```
title = data.raw_title.values.tolist()

import jieba
title_s = []
for line in title:
    title_cut = jieba.lcut(line)
    title_s.append(title_cut)

stopwords = pd.read_excel('./data/stopwords.xlsx')
stopwords = stopwords.stopword.values.tolist()
```

4.3 Remove stopwords and duplicates.

```
# Removing stopwords:
title_clean = []
for line in title_s:
    line_clean = []
    for word in line:
        if word not in stopwords:
            line_clean.append(word)
    title_clean.append(line_clean)
```

```
# Removing duplicates
title_clean_dist = []
for line in title_clean:
    line_dist = []
    for word in line:
        if word not in line_dist:
            line_dist.append(word)
    title_clean_dist.append(line_dist)
```

4.4 Converting title_clean_dist to only one list; Turning the list allwords_clean_dist to DataFrame; Classifying and summarizing the words filtered and de duplicated.

```
allwords_clean_dist = []
for line in title_clean_dist:
    for word in line:
        allwords_clean_dist.append(word)

df_allwords_clean_dist = pd.DataFrame({'allwords': allwords_clean_dist})
word_count = df_allwords_clean_dist.allwords.value_counts().reset_index()
word_count.columns = ['word', 'count']

word_count.head()
```

	word	count
0	沙发	2897
1	组合	2800
2	客厅	2570
3	小户型	2348
4	简约	1908

5. Word Cloud Visualization

5.1 Use libraries: Matplotlib, wordcloud and imread, to visualize word cloud as following.

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from imageio import imread # Drawing pattern outline


plt.figure(figsize = (20, 10))

pic = imread("./images/shafa.jpg")

w_c = WordCloud(font_path = "./data/simhei.ttf",  # Setting font
                background_color = 'white', # The default is black. It's set to white here.
                mask = pic, # Modeling cover
                max_font_size = 60, # Font maximum
                margin = 1)

wc = w_c.fit_words({x[0] : x[1] for x in word_count.head(100).values}) # Taking the first 100 words for visualization

plt.imshow(wc, interpolation = 'bilinear') # Graph optimization
plt.axis("off") # Hiding axises
plt.show()
```

5.2 Conclusion: (1)Words such as '沙发', '组合', '客厅', '小户型' and '简约' are frequently used. (2)Among the words describing sofa material, '布艺沙发' appears the most frequently. (3)Among the words describing sofa style, '简约' appears the most frequently. (4)Among the words used to describe the type of house the sofa is suitable for, '小户型' appears the most frequently.

5.3 Prediction: (1) As for sofa material, due to the large sales volume of fabric sofa, that is, the demand is large, it is expected that the enterprise will increase the supply in response to the purchase demand. (2) About sofa style, simple style is very popular. More enterprises will produce simple style sofas, and more businesses will sell simple style sofas. (3) About the type of houses the sofa is suitable for, it is very popular for small family and more businesses will sell sofa suitable for small family.

6. the Impact of Different Keywords on Sales

Using Numpy to process the data we need.

```
import numpy as np

w_s_sum = []
for w in word_count.word:
    i = 0
    s_list = []
    for t in title_clean_dist:
        if w in t:
            try:
                s_list.append(data.sales[i])
            except:
                s_list.append(0)
        i += 1
    w_s_sum.append(sum(s_list))

df_w_s_sum = pd.DataFrame({'w_s_sum': w_s_sum})
```

df_w_s_sum.head()

	w_s_sum
0	354926
1	347685
2	342496
3	314679
4	275670

word_count.head()

	word	count
0	沙发	2897
1	组合	2800
2	客厅	2570
3	小户型	2348
4	简约	1908

Combining data tables (concatenation) using Pandas

```
df_word_sum = pd.concat([word_count, df_w_s_sum], axis = 1, ignore_index = True)
df_word_sum.columns = ['word', 'count', 'w_s_sum']
```

df_word_sum.head(30)

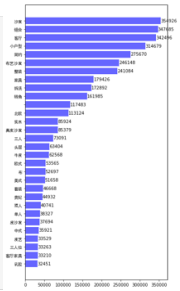
	word	count	w_s_sum
0	沙发	2897	354926
1	组合	2800	347685
2	客厅	2570	342496
3	小户型	2348	314679
4	简约	1908	275670
5	布艺沙发	1711	246148
6	整装	1508	241084
7	家具	1164	179426
8	转角	1129	161985

```
df_word_sum.sort_values('w_s_sum', inplace = True, ascending = True)
# ascendingly sort according to the value of 'w_s_sum'
df_w_s = df_word_sum.tail(30)
df_w_s
```

	word	count	w_s_sum
28	乳胶	274	32451
23	客厅家具	367	33210
25	三人位	326	33263
26	皮艺	300	33529
17	中式	514	35921
36	皮沙发	189	37694
24	单人	364	38327

Visualizing data by Numpy and Matplotlib.

```
import matplotlib
from matplotlib import pyplot as plt
font = matplotlib.font_manager.FontProperties(fname='./data/simhei.ttf') # Importing font

index = np.arange(df_w_s.word.size)

plt.figure(figsize = (6, 12))
plt.barh(index,             # Giving the position on the y-axis
         df_w_s.w_s_sum,
         color = 'blue',
         align = 'center',
         alpha = 0.6)       # Tone adjustment

plt.yticks(index,  list(df_w_s.word), fontproperties = font)  # Setting y-axis


for y,x in zip(index, df_w_s.w_s_sum):
    plt.text(x,y,"%.0f" %x, ha = 'left', va = 'center')

plt.show()
```



7. the Price Distribution of Commodities

7.1 Analyze and visualize the price distribution of goods. In order to visualizing more intuitive, we only considering products with price less than 20,000.

```
data_p = data[data['view_price'] < 20000]

plt.figure(figsize = (7, 5))
plt.hist(data_p['view_price'], bins = 20, color = 'purple')
plt.xlabel('Price')
plt.ylabel('Quantity of goods on sale')
plt.title('Price distribution of goods on sale')
plt.show()
```

data.head()

	raw_title	view_price	province	city	sales
0	北欧风格小户型实木架客厅L型布艺沙发组合	3480.0	广东	佛山	328
1	凯铭家具欧式真皮沙发 头层牛皮雕花实木客厅整装转角沙发组合	7400.0	广东	佛山	111
2	雅居汇 欧式真皮转角沙发实木雕花客厅家具组合美式真皮转角沙发	18800.0	广东	佛山	25
3	斯帝罗兰简约现代真皮沙发组合整装客厅转角头层牛皮三人沙发家具	8699.0	广东	江门	93
4	【新款】现丽家kuka北欧布艺沙发客厅大小户型现代简约组合2037	2999.0	浙江	杭州	142

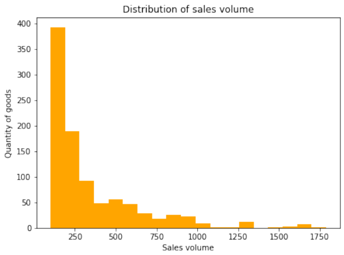
7.2 Conclusion: (1)The largest number of sofas on sale with a price between ￥1,000-2,000. (2)The prices of sofas are mainly between ￥0-4,000. (3)The quantity of goods shows a downward trend with the price rising. The higher the price, the less goods are on sale.

7.3 Prediction: As a necessity of daily life, the price elasticity of demand is relatively small. Most middle-class consumers will tend to buy low-end sofas, and businesses will also focus on selling sofas with a price range of ￥ 1000-2000. In the future, there will still be businesses selling high-quality and high-grade expensive sofas to high-end consumers, but the number of sales is limited.

8. the Sales Distribution of Commodities

8.1 In order to make the visualization more intuitive, only considering products with sales more than 100, less than 20,000.

```
data_s = data[(data['sales'] > 100) & (data['sales'] < 2000)]
print('Proportion of goods which sales volume is in the range of 100 to 2000: %0.3f' %(len(data_s)/len(data)))

plt.figure(figsize = (7, 5))
plt.hist(data_s['sales'], bins = 19, color = 'orange') # bins =19 : width for every bin is (2,000–100)/19 = 100
plt.xlabel('Sales volume')
plt.ylabel('Quantity of goods')
plt.title('Distribution of sales volume')
plt.show()
```



8.2 Conclusion: (1) Goods with a sales volume in the range of 100 to 2000 accounted for only 27.1%, of which the sales of 100-200 is the most, followed by 200-300. (2) When the sales volume is between 100 and 500, the quantity of goods shows a downward trend with the sales volume, and the trend is steep, sales of most goods are very low. (3) There are few goods with sales volume of more than 1000.

8.3 Prediction: Our conclusion is in line with the 28 law of Pareto, an Italian economist. That is to say, 20% of all kinds of goods bring 80% of the sales volume, so Pareto believes that only this part should be retained, and the rest should be discarded. According to this Law and the actual data we analyze, we predict that in the future, the sales of those small amount of goods will be more, and some of the goods with low sales will be eliminated by the market.

9. Average Sales Volume Distribution of Commodities in Different Price Ranges

9.1 According to the prices of goods, the price range is divided into 10 equal parts, and match each commodity with the corresponding price range.

```
data['price'] = data.view_price.astype('int')
data['price_range'] = pd.qcut(data.price, 10)

data.head()
```

	raw_title	view_price	province	city	sales_volume	price	price_range
0	北欧风格小户型实木架客厅L型布艺沙发组合	3480.0	广东	佛山	328	3480	(2980.0, 3699.0]
1	凯哲家具欧式真皮沙发 头层牛皮雕花实木客厅整装转角沙发组合	7400.0	广东	佛山	111	7400	(6600.0, 99002.0]
2	雅居汇 欧式真皮转角沙发实木雕花客厅家具组合美式真皮转角沙发	18800.0	广东	佛山	25	18800	(6600.0, 99002.0]
3	斯帝罗兰简约现代真皮沙发组合整装客厅转角头层牛皮三人沙发家具	8699.0	广东	江门	93	8699	(6600.0, 99002.0]
4	！【新款】现顾家kuka北欧布艺沙发客厅大小户型现代简约组合2037	2999.0	浙江	杭州	142	2999	(2980.0, 3699.0]

```
df_s_g = data[['sales_volume', 'price_range']]
df_s_g
```

	sales_volume	price_range
0	328	(2980.0, 3699.0]
1	111	(6600.0, 99002.0]
2	25	(6600.0, 99002.0]
3	93	(6600.0, 99002.0]
4	142	(2980.0, 3699.0]
...	...	...
4387	230	(39.999, 588.1]
4388	125	(1813.8, 2368.0]
4389	31	(2980.0, 3699.0]
4390	942	(39.999, 588.1]
4391	1323	(39.999, 588.1]

3502 rows × 2 columns

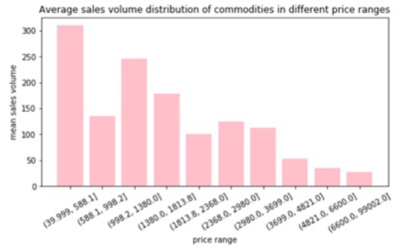
Then, calculate the average sales volume of goods belonging to the same price range.

```
df_s_g = data[['sales_volume', 'price_range']].groupby('price_range').mean().reset_index()
df_s_g
```

	price_range	sales_volume
0	(39.999, 588.1]	309.817664
1	(588.1, 998.2]	134.505714
2	(998.2, 1380.0]	246.216524
3	(1380.0, 1813.8]	177.845272
4	(1813.8, 2368.0]	101.056338
5	(2368.0, 2980.0]	123.955923
6	(2980.0, 3699.0]	112.200000
7	(3699.0, 4821.0]	52.799427
8	(4821.0, 6600.0]	35.366762
9	(6600.0, 99002.0]	27.405714

After calculating, we use matplotlib to visualize average sales volume distribution of commodities in different price ranges.

```
index = np.arange(df_s_g.price_range.size)
plt.figure(figsize = (8, 4))
plt.bar(index, df_s_g.sales_volume, color = 'pink')
plt.xticks(index, df_s_g.price_range, rotation = 30)
plt.xlabel('price range')
plt.ylabel('mean sales volume')
plt.title('Average sales volume distribution of commodities in different price ranges')
plt.show()
```



9.2 Conclusion: (1)The average sales volume of goods with price between ￥40-588 is the highest, followed by ￥998-1380, and the lowest with price over ￥6600; (2) Overall, it shows a downward trend.

9.3 Prediction: In the future, the commodities with price range of ￥ 40-588 will continue to dominate, and most of the middle and low-end sofas will dominate the market. Businesses can find their own position and choose whether to benefit from the sales of middle-class consumers or provide expensive high-end sofas for a few people.

10. the Influence of Price on Sales Volume

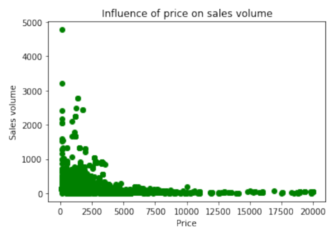
10.1 In order to make the visualization more intuitive, only considering products with price less than 20,000.

```
data_p
```

	raw_title	view_price	province	city	sales
0	北欧风格小户型实木架客厅L型布艺沙发组合	3480.0	广东	佛山	328
1	凯哲家具欧式真皮沙发 头层牛皮雕花实木客厅整装转角沙发组合	7400.0	广东	佛山	111
2	雅居汇 欧式真皮转角沙发实木雕花客厅家具组合美式真皮转角沙发	18800.0	广东	佛山	25
3	斯帝罗兰简约现代真皮沙发组合整装客厅转角头层牛皮三人沙发家具	8699.0	广东	江门	93
4	！【新款】现顾家kuka北欧布艺沙发客厅大小户型现代简约组合2037	2999.0	浙江	杭州	142
...	...	...	...	...	...
4387	奶茶甜品店咖啡西餐厅洽谈桌椅组合现代简约休闲皮艺双人卡座沙发	280.0	浙江	湖州	230
4388	北欧简约布艺沙发大小户型客厅转角储物可拆洗L/U型整装组合家具	1980.0	广东	佛山	125
4389	美式乡村复古工艺意大利进口油蜡皮头层真牛皮沙发1+2+3组合	3499.0	广东	佛山	31
4390	办公沙发简约会客接待商务三人位沙发办公室家具时尚沙发茶几组合	558.0	广东	佛山	942
4391	洽谈桌椅组合简约休闲甜品奶茶店西餐咖啡厅双人卡座办公室布沙发	200.0	浙江	湖州	1323

3449 rows × 5 columns

```
fig, ax = plt.subplots()
ax.scatter(data_p['view_price'], data_p['sales'], color='green')
ax.set_xlabel('Price')
ax.set_ylabel('Sales volume')
ax.set_title('Influence of price on sales volume')
plt.show()
```



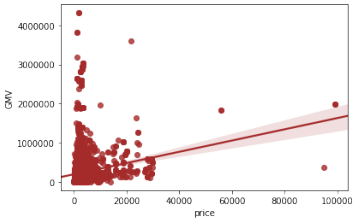
10.2 Conclusion: (1) General trend: the lower the price of a commodity, the higher its sales volume. (2) Commodity with a price range of ￥0-4000 are easier to sell.

10.3 Prediction: Most businesses will choose to provide middle and low-end sofas to middle-class consumers, of course, there will be a relatively small number of businesses dedicated to selling expensive high-end sofas for specific consumers

11. the Influence of Price on Sales

11.1 Use Seaborn library to visualize and analyze the influence of price on sales.

```
data['GMV'] = data['price'] * data['sales']

import seaborn as sns
sns.regplot(x = 'price', y = 'GMV', data = data, color = 'brown')
```



11.2 Conclusion: (1) General trend: as can be seen from the linear regression fitting line, with the increase of price, the total sales shows an upward trend. (2) The prices of most commodities are low, and the total sales is also low.

11.3 Prediction: low and medium-end sofas with relatively low prices are favored by consumers. According to the conclusion, we can see that most consumers are not very sensitive to the price, and businesses can get benefits by increasing the price appropriately.

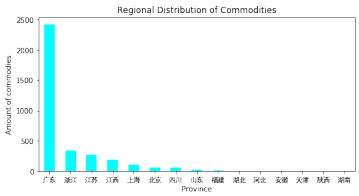
12. the Regional Distribution of Commodities

12.1 Use Numpy and Matplotlib to visualize and analyze the regional distribution of commodities.

```
data.province.value_counts()
```

```
广东    2421
浙江     339
江苏     271
江西     191
上海     106
北京      65
```

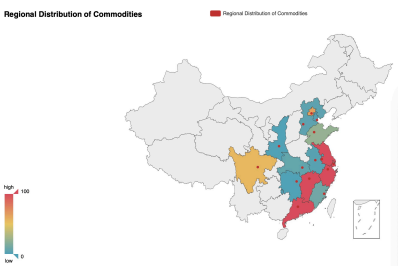
```
plt.figure(figsize = (8,4))
data.province.value_counts().plot(kind = 'bar', color = 'cyan')
plt.xticks(rotation = 0, fontproperties = font)
plt.xlabel('Province')
plt.ylabel('Amount of commodies')
plt.title('Regional Distribution of Commodities')
plt.show()
```



12.2 We realize visualization of geographic information and commodities information by import map from pyecharts library. It has the dynamic characteristic and interactive feature. So that can make our visualization more interesting attractive and efficient.

```
from pyecharts import Map

attr = r_d_c.index
value = r_d_c.values

map = Map("Regional Distribution of Commodities", width = 1200, height = 600)
map.add("Regional Distribution of Commodities", attr, value, maptype ="china",  is_visualmap = True, visual_text_color =

map.render(path="Regional Distribution of Commodities.html")
```

```
Interactive page:
file:///C:/Users/vince/Desktop/fintech/Introduction%20to%20Data%20Science%20Programming/Project/Regional%20Distrib
```



12.3 Conclusion: (1) The number of commodities from Guangdong Province is the largest, followed by Zhejiang and Jiangsu. (2) In particular, the number of goods from Guangdong far exceeds that from Jiangsu, Zhejiang, Shanghai and other places, indicating that for the sub-category of sofa, stores in Guangdong are dominant. (3) The number of commodities from Jiangsu, Zhejiang and Shanghai is almost the same.

12.4. Prediction: the enterprises of sofa production and sales in mainland China will still be mainly concentrated in the southeast coastal areas, and Guangdong Province is dominant with obvious advantages. According to the theory of industrial cluster in economics, more sofa manufacturers will invest factories and sell sofa in southeast coastal areas.