第 1 章

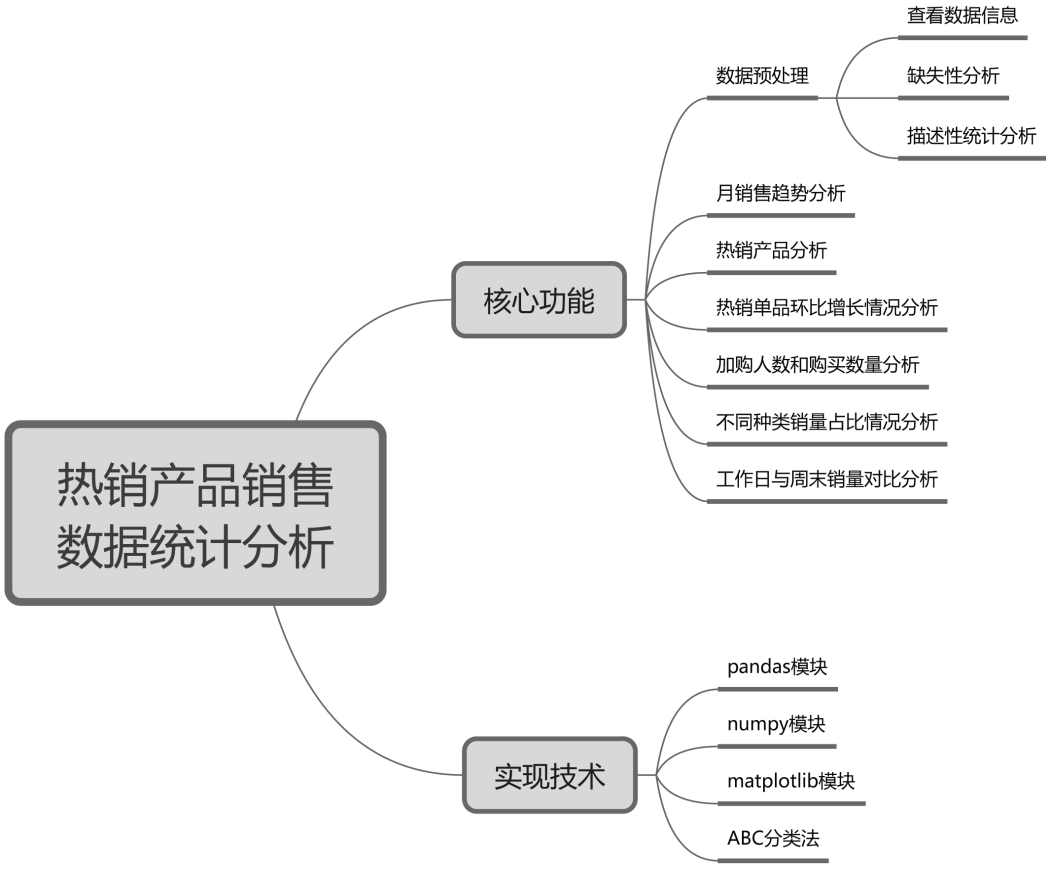
热销产品销售数据统计分析

——pandas + numpy + matplotlib + ABC 分类法

随着电商行业的快速发展，销售数据的规模和复杂性也在不断增加。通过使用 Python 进行销售数据分析，我们可以从海量数据中挖掘出有价值的信息，如销售趋势、热门产品、客户消费行为等。本章将使用 pandas 模块、numpy 模块、matplotlib 模块并结合 ABC 分类法实现热销产品销售数据统计分析。

项目微视频

本项目的核心功能及实现技术如下：



1.1 开 发 背 景

现如今，销售数据分析已成为企业营销和决策的关键因素，而深入挖掘销售数据中的热销产品将会更大程度地提升销售业绩，从而为企业带来更大的收益。同时，销售增长速度的环比分析是企业经营管理中

常用的指标，用于衡量销售业绩的增长速度。本章将使用 pandas 模块、numpy 模块、matplotlib 模块并结合 ABC 分类法实现月销售趋势分析、热销产品分析、热销单品环比增长情况分析等。

1.2 系 统 设 计

1.2.1 开发环境

本项目的开发及运行环境如下：

- ☑ 操作系统：推荐 Windows 10、Windows 11 或更高版本。
- ☑ 编程语言：Python 3.12。
- ☑ 开发环境：PyCharm。
- ☑ 第三方模块：pandas（2.1.4）、openpyxl（3.1.2）、numpy（1.26.3）、matplotlib（3.8.2）。

1.2.2 分析流程

热销产品销售数据统计分析首要任务是数据准备，接着进行数据预处理工作，即查看数据信息、缺失性分析和描述性统计分析，以确保数据质量，然后进行数据统计分析。

本项目分析流程如图 1.1 所示。

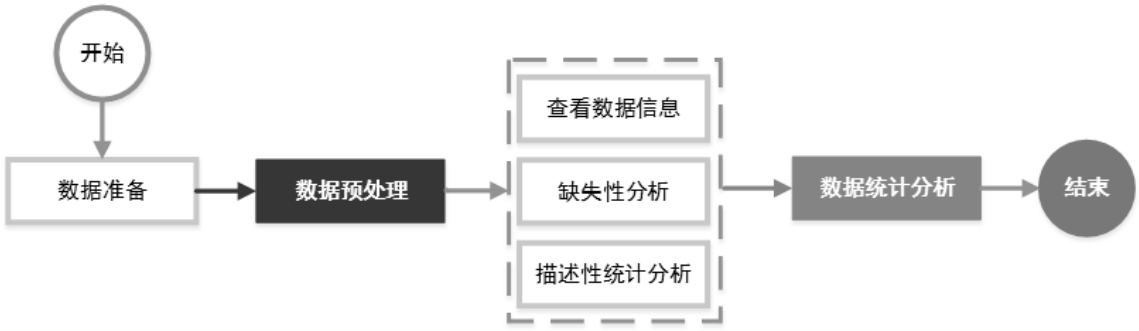


图 1.1 热销产品销售数据统计分析流程

1.2.3 功能结构

本项目的功能结构已经在章首页中给出。本项目实现的具体功能如下：

- ☑ 数据预处理：查看数据信息、缺失性分析、描述性统计分析。
- ☑ 月销售趋势分析：按月统计分析成交商品件数和成交码洋。
- ☑ 热销产品分析：采用 ABC 分类法和累计贡献率对产品进行分类，将其划分为 A 类、B 类和 C 类，随后对分类结果进行可视化展示。
- ☑ 热销单品环比增长情况分析：对热销单品的成交商品件数按月进行环比分析。
- ☑ 加购人数和购买数量分析：对热销产品中的 A 类产品加入购物车人数和实际成交商品件数进行对比分析。
- ☑ 不同种类销量占比情况分析：分析热销产品中不同种类销量占比情况。
- ☑ 工作日与周末销量对比分析：对比分析工作日五天和周末两天的销量。

1.3 技 术 准 备

1.3.1 技术概览

Python 数据分析相关模块非常多，作为第一个项目，我们采用 Python 数据分析最基本的三大模块，它们也是 Python 数据分析必备三剑客，即 pandas 模块、numpy 模块和 matplotlib 模块。这三大模块基本可以实现数据分析所需的大部分功能，并且各自分工明确。其中，pandas 模块主要用于数据处理和统计分析，numpy 模块主要用于数组计算和科学计算，matplotlib 模块主要用于绘制图表，实现数据可视化。

关于这三大模块，此处不进行详细介绍。《Python 数据分析从入门到精通（第 2 版）》一书对它们进行了详细的讲解，对这些知识不太熟悉的读者，可以参考该书对应的内容。

除此之外，对于热销产品的分析，我们使用 ABC 分类法，下面将对其进行必要的介绍，以确保读者可以顺利完成本项目。

1.3.2 ABC 分类法

顾名思义，热销产品销售数据统计分析核心任务就是找出哪些产品属于热销产品，然后对这些热销产品进一步分析。那么，通过猜测或人工筛选显然是不合理的。

在该项目中，我们将使用科学的分析方法，即 ABC 分类法（帕累托分析法），将销售数据按产品维度进行分析，对比不同产品的销售情况，将产品划分为 A 类、B 类和 C 类。我们首先来了解什么是 ABC 分类法。

ABC 分类法全称为 ABC 分类库存控制法，又称为帕累托分析法或 ABC 分析法、ABC 管理法，通常被称为 80/20 法则，该法则是由意大利经济学家“帕累托”提出的。80/20 法则认为：原因和结果、投入和产出、努力和报酬之间本来存在着无法解释的不平衡。例如，一家公司 80%的利润常常来自 20%的产品。下面简单介绍一下相关算法，如下所示：

$$\text{累计贡献率（\%）}=\frac{\text{累加销售收入}}{\text{销售总收入}}\times 100\%$$

上述公式得出的计算结果，我们称之为“累计贡献率”。累计贡献率对应的是产品，通过累计贡献率，可以将产品划分为 A 类、B 类和 C 类，如表 1.1 所示。

表 1.1 ABC 分类法

类 别	重 要 程 度	占 比	说 明
A 类产品	非常重要	80%	数量占比少，价值占比大
B 类产品	比较重要	10%	没有 A 类产品重要，处于 A 类和 C 类之间
C 类产品	一般重要	10%	数量占比大但价值占比很小

说明

真正的比例不一定正好是 80%：20%。80/20 法则表明在多数情况下该关系很可能是不平衡的，并且接近于 80/20。

另外需要说明一点，在实际应用中，上述算法无须手工计算，因为 pandas 模块已经为我们提供了现成

的函数，即 cumsum()函数和 sum()函数。其中，cumsum()函数用于计算序列的累积和，也就是从第一条记录开始计算，每一条记录都与上一条记录相加，直到最后一条记录。cumsum()函数在上述算法中用于计算“累加销售收入”，举个简单的例子：

```
import pandas as pd
import numpy as np
data=np.array([1,3,5,7,9])
data_cumsum=pd.Series(data).cumsum()
print(data_cumsum)
```

运行程序，结果如下：

```
0    1
1    4
2    9
3   16
4   25
```

sum()函数用于求和，在上述算法中用于计算销售总收入，例如对上述举例进行求和，代码如下：

```
print(data.sum())
```

运行程序，结果如下：

```
25
```

在数据统计分析过程中，这两个函数应用非常广泛，它们可以帮助我们更好地了解数据的趋势和变化，因此读者必须掌握它们。

1.4 前 期 工 作

1.4.1 开发环境设置

数据分析过程中，我们经常需要在 PyCharm 的控制台中显示数据。然而，默认情况下，这些数据可能会出现不对齐的现象。为了解决这个问题，我们需要对 PyCharm 控制台字体进行一些简单的设置。运行 PyCharm，选择 File→Settings 命令，打开 Settings 对话框，然后按照图 1.2 所示的步骤进行设置即可。

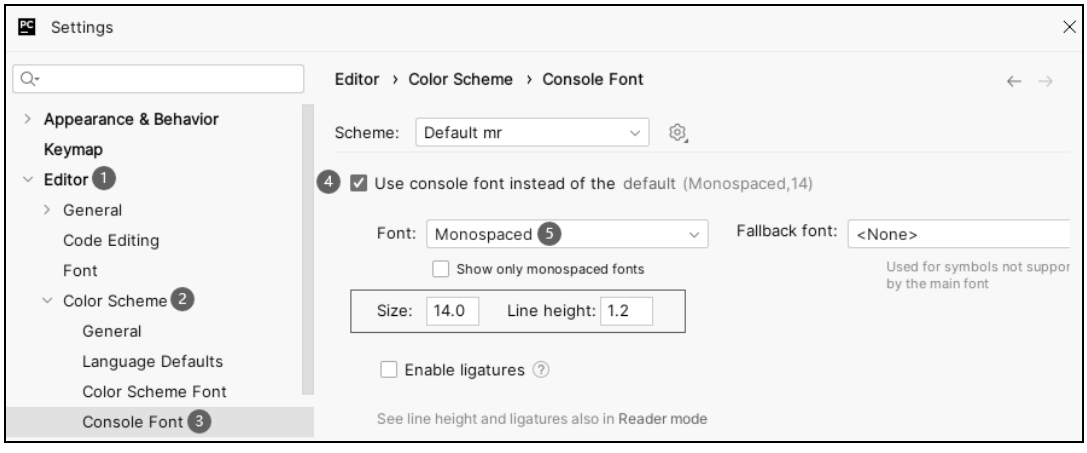


图 1.2 设置字体

1.4.2 安装第三方模块

按照惯例，我们通常使用 pip 命令来安装第三方模块，当然也可以在 PyCharm 开发环境中进行安装。本项目所需模块在前面已经进行了介绍，这里应逐一进行安装。

例如，安装 pandas 模块，在系统“搜索”文本框中输入 cmd，打开“命令提示符”窗口，然后输入如下安装命令：

```
pip install pandas
```

如果需要多个模块，也可以同时安装，则安装命令如下：

```
pip install module1 module2 module3
```

如果需要指定特定版本的模块，则安装命令如下：

```
pip install module1==1.0.0 module2>=2.0.0 module3<=3.0.0
```

1.4.3 新建项目目录

开发项目前，应当创建一个项目目录，用于保存项目所需的 Python 脚本文件，具体步骤如下：运行 PyCharm，右击工程目录（如 PycharmProjects），在弹出的快捷菜单中选择 New 下的 Directory，然后输入名称“热销产品销售数据统计分析”，最后按 Enter 键确认，这样项目目录就创建成功了，如图 1.3 所示。

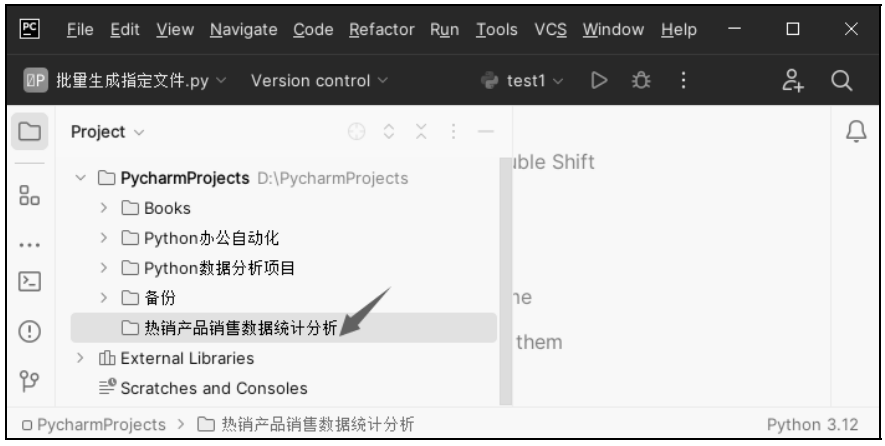


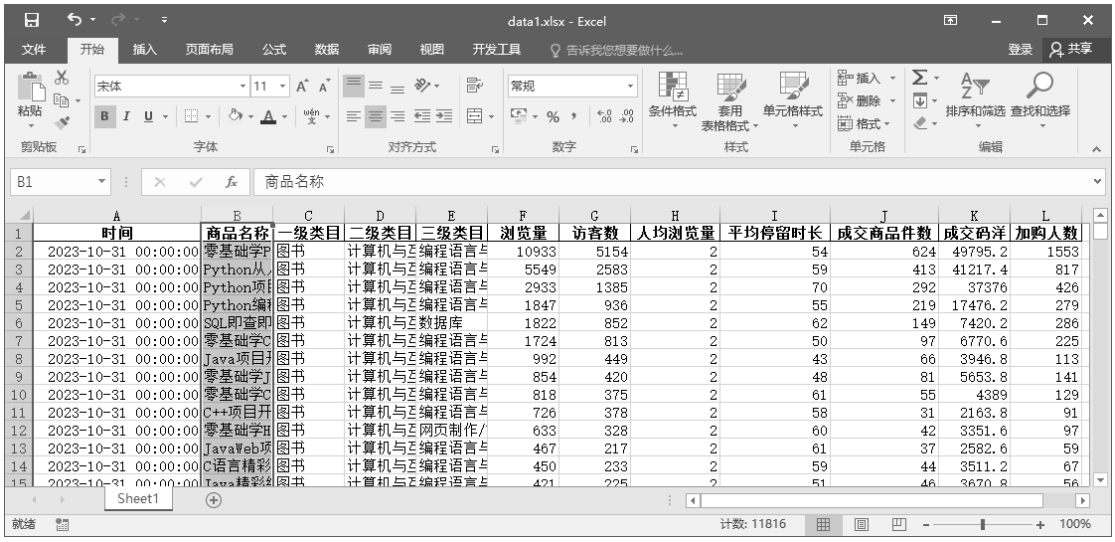
图 1.3 新建项目目录

1.4.4 数据准备

热销产品销售数据统计分析的数据来源于 Excel 文件，具体介绍如下：

- ☑ 文件名：data1.xlsx。
- ☑ 字段：时间、商品名称、一级类目、二级类目、三级类目、浏览量、访客数、人均浏览量、平均停留时长、成交商品件数、成交码洋、加购人数。
- ☑ 记录：11815 条产品销售数据。

部分数据截图，如图 1.4 所示。



	A	B	C	D	E	F	G	H	I	J	K	L
1	时间	商品名称	一级类目	二级类目	三级类目	浏览量	访客数	人均浏览量	平均停留时长	成交商品件数	成交码洋	加购人数
2	2023-10-31 00:00:00	零基础学P	图书	计算机与互	编程语言与	10933	5154	2	54	624	49795.2	1553
3	2023-10-31 00:00:00	Python从	图书	计算机与互	编程语言与	5549	2583	2	59	413	41217.4	817
4	2023-10-31 00:00:00	Python项	图书	计算机与互	编程语言与	2933	1385	2	70	292	37376	426
5	2023-10-31 00:00:00	Python编	图书	计算机与互	编程语言与	1847	936	2	55	219	17476.2	279
6	2023-10-31 00:00:00	SQL即查即	图书	计算机与互	数据库	1822	852	2	62	149	7420.2	286
7	2023-10-31 00:00:00	零基础学C	图书	计算机与互	编程语言与	1724	813	2	50	97	6770.6	225
8	2023-10-31 00:00:00	Java项目开	图书	计算机与互	编程语言与	992	449	2	43	66	3946.8	113
9	2023-10-31 00:00:00	零基础学J	图书	计算机与互	编程语言与	854	420	2	48	81	5653.8	141
10	2023-10-31 00:00:00	零基础学C	图书	计算机与互	编程语言与	818	375	2	61	55	4389	129
11	2023-10-31 00:00:00	C++项目开	图书	计算机与互	编程语言与	726	378	2	58	31	2163.8	91
12	2023-10-31 00:00:00	零基础学H	图书	计算机与互	网页制作/	633	328	2	60	42	3351.6	97
13	2023-10-31 00:00:00	Java Web项	图书	计算机与互	编程语言与	467	217	2	61	37	2582.6	59
14	2023-10-31 00:00:00	C语言精彩	图书	计算机与互	编程语言与	450	233	2	59	44	3511.2	67
15	2023-10-31 00:00:00	Java精彩	图书	计算机与互	编程语言与	421	225	2	51	46	3670.8	56

图 1.4　部分数据截图

说明

data1.xlsx 位于资源包项目所在文件夹下的 data 文件夹中，开发本项目前，应将 data 文件夹复制到项目目录中，如图 1.5 所示。

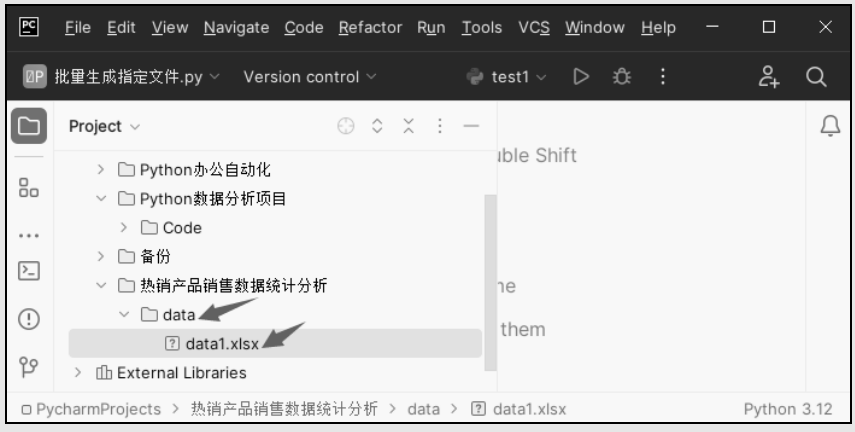


图 1.5　将 data 文件夹复制到项目目录

1.5　数据预处理

1.5.1　查看数据信息

面对一个全新的数据集，在尚未了解其内容的情况下，我们不应冒然进行数据处理和分析。那么，我们应如何着手呢？答案肯定是，我们首先应该查看数据的基本信息，以了解数据的概况，包括数据的行数、列数以及所包含的字段等。由于该项目的数据源自 Excel 文件，如果数据量庞大，直接在 Excel 中打开查看显然是不可取的。在这种情况下，我们可以使用 pandas 模块提供的相关函数来预览数据。

例如，查看 Excel 文件 data1.xlsx 的数据信息，实现过程如下（源码位置：资源包\Code\01\data_view.py）。

（1）运行 PyCharm，在项目目录下新建一个 Python 文件，并将其命名为 data_view.py。

（2）导入相关模块，代码如下：

```
# 导入 pandas 模块
import pandas as pd
```

（3）读取 Excel 文件，并查看数据的行数、列数以及字段。这主要使用 DataFrame 对象的 shape 属性和 columns 属性来实现，代码如下：

```
# 读取 Excel 文件
df=pd.read_excel('./data/data1.xlsx')
# 查看数据维数（行数和列数）和字段
print(df.shape)
print(df.columns)
```

运行程序，结果如图 1.6 所示。

```
(11815, 12)
Index(['时间', '商品名称', '一级类目', '二级类目', '三级类目', '浏览量', '访客数', '人均浏览量', '平均停留时长',
       '成交商品件数', '成交码洋', '加购人数'],
      dtype='object')
```

图 1.6　查看数据

从运行结果中得知：该数据集包含 11815 行和 12 列，同时列出了每一列的名称，即数据中包含的字段。

（4）使用 DataFrame 对象的 head()方法和 tail()方法分别查看前 5 条和最后 5 条数据，代码如下：

```
# 查看前 5 条和最后 5 条数据
print(df.head())
print(df.tail())
```

运行程序，结果如图 1.7 所示。

```
         时间                 商品名称 一级类目      二级类目  ... 平均停留时长  成交商品件数      成交码洋  加购人数
0 2023-10-31      零基础学Python（全彩版）   图书  计算机与互联网  ...      54     624  49795.2  1553
1 2023-10-31  Python从入门到项目实践（全彩版）   图书  计算机与互联网  ...      59     413  41217.4   817
2 2023-10-31  Python项目开发案例集锦（全彩版）   图书  计算机与互联网  ...      70     292  37376.0   426
3 2023-10-31      Python编程锦囊（全彩版）   图书  计算机与互联网  ...      55     219  17476.2   279
4 2023-10-31        SQL即查即用（全彩版）   图书  计算机与互联网  ...      62     149   7420.2   286

[5 rows x 12 columns]
             时间                 商品名称 一级类目  ... 成交商品件数    成交码洋  加购人数
11810 2023-09-01      JSP项目开发实战入门（全彩版）   图书  ...       2  139.6     3
11811 2023-09-01       C#精彩编程200例（全彩版）   图书  ...       2  179.6     4
11812 2023-09-01        玩转C语言程序设计（全彩版）   图书  ...       0    0.0     1
11813 2023-09-01  Android精彩编程200例（全彩版）   图书  ...       2  179.6     6
11814 2023-09-01         Java开发详解（全彩版）   图书  ...       2  238.0     2

[5 rows x 12 columns]
```

图 1.7　前 5 条和最后 5 条数据

说明

从运行结果中得知：数据出现了不对齐以及显示“...”（即省略了部分数据）的现象，那么如何解决这一问题呢？

pandas 模块提供了专门用于处理数据显示格式的函数 set_option()，可以更改的参数如下：

☑ display.max_columns：设置 DataFrame 对象的最大显示列数，默认值为 20。
☑ display.max_rows：设置 DataFrame 对象的最大显示行数，默认值为 60。
☑ display.max_colwidth：设置 DataFrame 对象每列的最大长度，默认值为 50。
☑ display.precision：设置 DataFrame 对象中数字的显示精度，默认值为 6。

除此之外，还可以设置 DataFrame 对象的显示格式。其中：float_format 参数用于设置浮点数的显示格式，如“%.2f”表示保留 2 位小数；date_dayfirst 参数用于设置日期的显示格式，默认为 False，表示以月/日/年的格式显示日期，如果设置为 True，则以日/月/年的格式显示日期。

在上述代码前加入如下代码：

```
# 设置数据显示的编码格式为东亚宽度，以实现列对齐
pd.set_option('display.unicode.east_asian_width', True)
pd.set_option('display.width',10000)          # 显示宽度
pd.set_option('display.max_columns',1000)     # 最大列数
```

再次运行程序，之前的问题得到解决，结果如图 1.8 所示。

	时间	商品名称	一级类目	二级类目	三级类目	浏览量	访客数	人均浏览量	平均停留时长	成交商品件数	成交码洋	加购人数
0	2023-10-31	零基础学Python（全彩版）	图书	计算机与互联网	编程语言与程序设计	10933	5154	2	54	624	49795.2	1553
1	2023-10-31	Python从入门到项目实践（全彩版）	图书	计算机与互联网	编程语言与程序设计	5549	2583	2	59	413	41217.4	817
2	2023-10-31	Python项目开发案例集锦（全彩版）	图书	计算机与互联网	编程语言与程序设计	2933	1385	2	70	292	37376.0	426
3	2023-10-31	Python编程锦囊（全彩版）	图书	计算机与互联网	编程语言与程序设计	1847	936	2	55	219	17476.2	279
4	2023-10-31	SQL即查即用（全彩版）	图书	计算机与互联网	数据库	1822	852	2	62	149	7420.2	286
	时间	商品名称	一级类目	二级类目	三级类目	浏览量	访客数	人均浏览量	平均停留时长	成交商品件数	成交码洋	加购人数
11810	2023-09-01	JSP项目开发实战入门（全彩版）	图书	计算机与互联网	编程语言与程序设计	27	10	3	121	2	139.6	3
11811	2023-09-01	C#精彩编程200例（全彩版）	图书	计算机与互联网	编程语言与程序设计	25	21	1	39	2	179.6	4
11812	2023-09-01	玩转C语言程序设计（全彩版）	图书	计算机与互联网	编程语言与程序设计	23	13	2	77	0	0.0	1
11813	2023-09-01	Android精彩编程200例（全彩版）	图书	计算机与互联网	移动开发	20	14	1	177	2	179.6	6
11814	2023-09-01	Java开发详解（全彩版）	图书	计算机与互联网	编程语言与程序设计	14	10	1	42	2	238.0	2

图 1.8　显示前 5 条和最后 5 条数据

1.5.2　缺失性分析

在数据分析过程中，首先需要清晰地了解数据的情况，包括查看数据中是否存在缺失值以及列数据类型是否正常。下面使用 DataFrame 对象的 info()方法查看数据的摘要信息和缺失情况，代码如下（源码位置：资源包\Code\01\data_view.py）：

```
print(df.info())
```

运行程序，结果如图 1.9 所示。

从运行结果中得知：记录总数为 11815，如“时间”字段显示为 11815 non-null，这意味着在该字段中有 11815 条非空数据，其他字段也都显示为 11815 non-null。因此，结论是数据中没有缺失值。

说明

当出现大型数据集时，如果字段数达到上百个，那么使用前面的方法会显示很多信息，使得结果看起来十分烦琐。info()方法提供了简短摘要信息的功能，但主要通过将 verbose 参数值设置为 False 来实现，代码如下：

```
print(df.info(verbose=False))
```

运行程序，结果如图 1.10 所示。

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11815 entries, 0 to 11814
Data columns (total 12 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   时间      11815 non-null  datetime64[ns]
 1   商品名称    11815 non-null  object
 2   一级类目    11815 non-null  object
 3   二级类目    11815 non-null  object
 4   三级类目    11815 non-null  object
 5   浏览量     11815 non-null  int64
 6   访客数     11815 non-null  int64
 7   人均浏览量   11815 non-null  int64
 8   平均停留时长  11815 non-null  int64
 9   成交商品件数  11815 non-null  int64
 10  成交码洋    11815 non-null  float64
 11  加购人数    11815 non-null  int64
dtypes: datetime64[ns](1), float64(1), int64(6), object(4)
memory usage: 1.1+ MB
None
```

图 1.9　查看数据的摘要信息和缺失情况

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11815 entries, 0 to 11814
Columns: 12 entries, 时间 to 加购人数
dtypes: datetime64[ns](1), float64(1), int64(6), object(4)
memory usage: 1.1+ MB
None
```

图 1.10　数据简短摘要信息

与图 1.9 相比，摘要信息看起来简洁清爽多了。

1.5.3　描述性统计分析

为了快速检视统计信息并从中识别异常数据，如空数据或值为 0 的数据，我们可以使用 DataFrame 对象的 describe()方法来查看每列数据的基本统计量。这些统计量主要包括计数、平均值、标准差、最小值、第一四分位数（1/4 分位数）、中位数（1/2 分位数）、第三四分位数（3/4 分位数）以及最大值。通过分析这些统计量，我们可以发现数据中的异常情况，代码如下（源码位置：资源包\Code\01\data_view.py）：

```
print(df.describe().T)
```

运行程序，结果如图 1.11 所示。

	count	mean	min	25%	50%	75%	max	std
时间	11815	2023-07-04 06:46:13.355903488	2023-01-01 00:00:00	2023-04-06 00:00:00	2023-07-05 00:00:00	2023-10-03 00:00:00	2023-12-31 00:00:00	NaN
浏览量	11815.0	266.673127	0.0	56.0	117.0	249.0	11554.0	495.506533
访客数	11815.0	126.068811	0.0	30.0	58.0	118.0	5415.0	230.523978
人均浏览量	11815.0	2.018282	0.0	2.0	2.0	2.0	9.0	0.447521
平均停留时长	11815.0	75.209564	0.0	58.0	71.0	87.0	377.0	28.093542
成交商品件数	11815.0	16.588828	0.0	3.0	7.0	17.0	1007.0	31.714549
成交码洋	11815.0	1350.848667	0.0	238.0	558.4	1197.0	80358.6	2849.94013
加购人数	11815.0	31.088955	0.0	6.0	13.0	29.0	1733.0	62.131456

图 1.11　描述性统计分析

从运行结果中得知：数据的整体统计分布情况，包括记录数、均值、最小值、第一四分位数（25%）、中位数（50%）、第三四分位数（75%）、最大值和标准差。例如，在“成交商品件数”这一字段中，有 25%的数据小于或等于 3.0，50%的数据小于或等于 7.0，而 75%的数据小于或等于 17.0。同时，我们注意到“成交商品件数”和“成交码洋”等字段的最小值为 0，这可能表明客户没有进行购买、浏览或加入购物车的行为，因此这部分数据可能被视为异常值或无效数据。

接下来统计数据中值为 0 的记录数量，代码如下：

```
print(df[df == 0].count())
```

运行程序，结果如图 1.12 所示。

图 1.12　统计数据为 0 的记录数

从运行结果中得知：“成交商品件数”和“成交码洋”两个字段中，值为 0 的记录数最多，均为 745 条，而其他字段也有少量值为 0 的记录。然而，这些值为 0 的数据并不影响我们对热销产品的分析，因此这里选择不对其进行处理。

综上所述，数据质量良好，接下来进入数据统计分析环节。

1.6　数据统计分析

1.6.1　月销售趋势分析

月销售趋势分析主要实现按月份统计分析销售额和销量，并通过双折线图直观地体现销售的整体趋势，如图 1.13 所示。

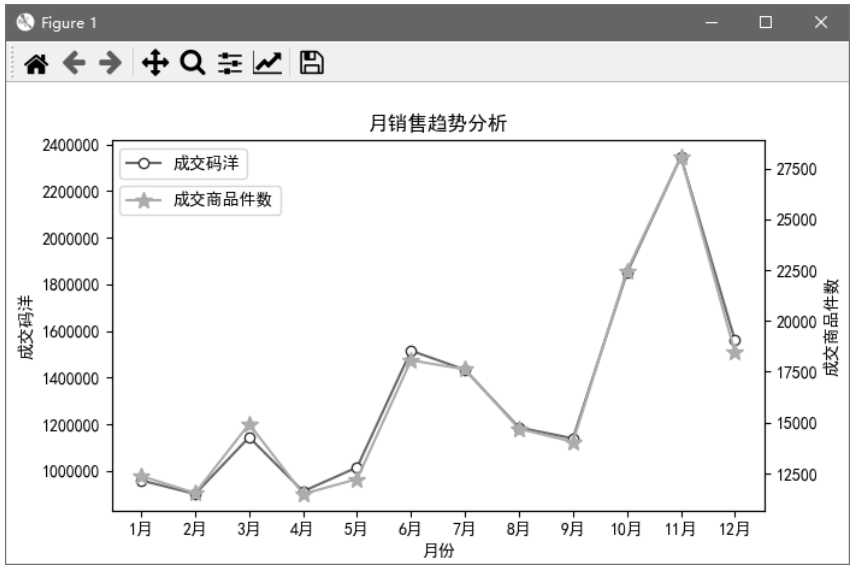


图 1.13　月销售趋势分析

从运行结果中得知：销售数据呈明显上升趋势，其中 6 月、10 月和 11 月是销售旺季。为了实现月销售趋势分析，首先使用 pandas 模块的 resample()方法和 to_period()方法按月统计销售数据，然后通过 matplotlib 模块的 plot()函数绘制双 *y* 轴图表，其中左侧 *y* 轴体现“成交码洋”，右侧 *y* 轴体现“成交商品件数”，实现过程如下（源码位置：资源包\Code\01\data_month_sales.py）。

（1）在项目目录下新建一个 Python 文件，并将其命名为 data_month_sales.py。

（2）导入相关模块，代码如下：

```
import pandas as pd                                    # 导入 pandas 模块
import matplotlib.pyplot as plt                        # 导入 matplotlib 模块
```

（3）读取 Excel 文件并抽取指定的数据，代码如下：

```
# 读取 Excel 文件
df=pd.read_excel('./data/data1.xlsx')
# 抽取数据
df=df[['时间','成交商品件数','成交码洋']]
```

（4）按月统计数据，首先按“时间”对数据进行升序排序并将“时间”设置为索引，然后使用 resample()方法和 to_period()方法按月统计数据，代码如下：

```
# 排序并设置时间为索引
df1=df.sort_values(by=['时间']).set_index('时间')
# 按月统计数据
df2=df1.resample('M').sum().to_period('M')
print(df2)
```

运行程序，查看按月统计后的数据，如图 1.14 所示。

时间	成交商品件数	成交码洋
2023-01	12376	958763.6
2023-02	11537	900500.2
2023-03	14933	1144057.4
2023-04	11493	911718.8
2023-05	12228	1014847.8
2023-06	18066	1515419.0
2023-07	17619	1433418.2
2023-08	14698	1185811.0
2023-09	14059	1138865.0
2023-10	22467	1848853.4
2023-11	28063	2347063.0
2023-12	18458	1560959.6

图 1.14　按月统计数据

（5）数据处理完成后，接下来的任务是使用 matplotlib 模块的 plot()函数绘制一个双 *y* 轴的图表。具体思路是：首先使用 add_subplot()函数创建一个子图，然后通过 twinx()函数为该子图添加一个与主 *y* 轴共享 *x* 轴的第二个 *y* 轴，并确保这个新 *y* 轴的刻度显示在子图的右侧，代码如下：

```
# 设置月份
month=['1 月','2 月','3 月','4 月','5 月','6 月','7 月','8 月','9 月','10 月','11 月','12 月']
# x 轴数据
x = range(len(month))
# y 轴数据
y1=df2[['成交码洋']]
y2=df2[['成交商品件数']]
# 解决中文乱码问题
plt.rcParams['font.sans-serif']=['SimHei']
# 设置画布大小
fig = plt.figure(figsize=(7,4))
# 创建子图表
ax1 = fig.add_subplot(111)
# x 轴刻度及标签
plt.xticks(x,labels=month)
# x 轴标签
plt.xlabel('月份')
# 取消科学记数法
ax1.get_yaxis().get_major_formatter().set_scientific(False)
# 第一个折线图
ax1.plot(x,y1,marker='o', mec='r', mfc='w',label=u'成交码洋')
ax1.set_ylabel('成交码洋')                              # 第一个 y 轴标签
ax1.legend(loc=2)                                      # 显示图例
```

```
# 第二个折线图
ax2 = ax1.twinx()                                          # 共享 x 轴添加一条 y 轴
ax2.plot(x,y2,color='orange',marker='*', ms=10,label=u'成交商品件数')
ax2.set_ylabel('成交商品件数')                               # 第二个 y 轴标签
ax2.legend(loc=2,bbox_to_anchor=(0, 0.9))                  # 显示图例
plt.title("月销售趋势分析")                                  # 标题
plt.show()                                                 # 显示图表
```

1.6.2 热销产品分析（ABC 分类法）

热销产品分析主要使用 ABC 分类法，该方法通过计算产品累计贡献率，将产品划分为 A 类、B 类和 C 类，并随后进行数据可视化，以便直观地观察热销产品的分布情况，如图 1.15 所示。

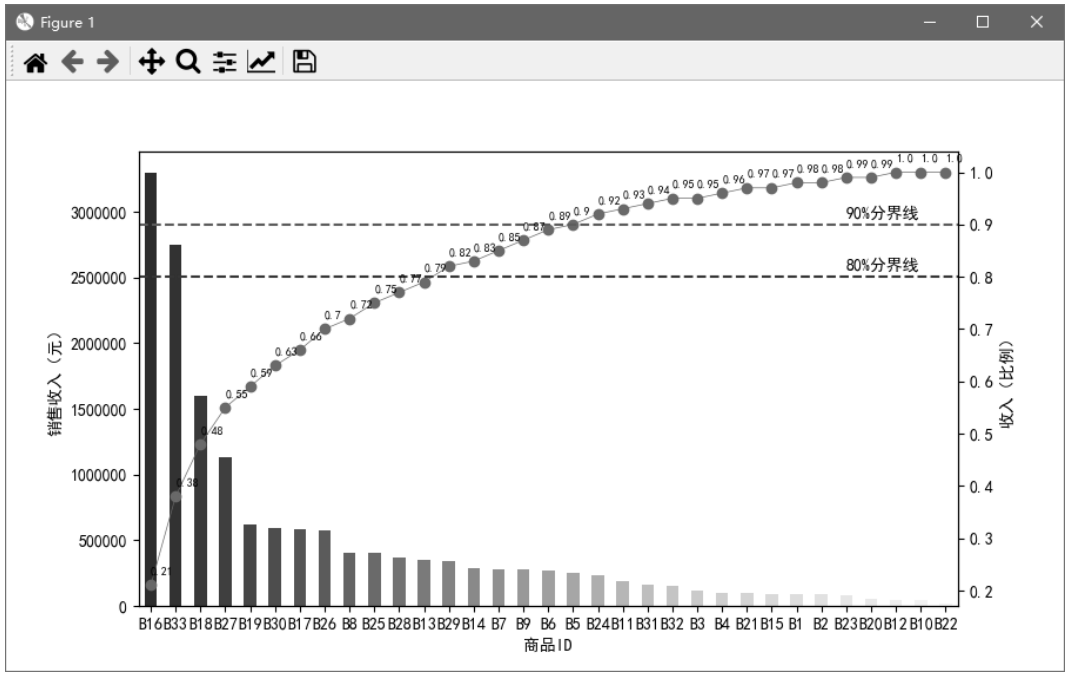


图 1.15 热销产品分析

从运行结果中得知：销售收入排名前三的产品分别是 B16、B33 和 B18，销售收入排名后三的产品分别是 B12、B10 和 B22。此外，各产品销售收入之间的差异巨大。

综上所述，根据 ABC 分类法，其中：A 类产品贡献率为 80%，包括 B16、B33、B18、B27、B19、B30、B17、B26、B8、B25、B28 和 B13，这些产品属于热销产品，需要重点营销；B 类产品贡献率为 10%，包括 B29、B14、B7、B9、B6 和 B5，这些产品比较重要，需要关注，并进一步挖掘增长点；C 类产品贡献率为 10%，包括 B24、B11、B31、B32、B3 等，这些产品一般重要，需要发挥潜力。

下面介绍热销产品分析的实现过程（源码位置：资源包\Code\01\data_hot_all.py）。

（1）在项目目录下新建一个 Python 文件，并将其命名为 data_hot_all.py。

（2）导入相关模块，代码如下：

```
import pandas as pd                                        # 导入 pandas 模块
import matplotlib.pyplot as plt                            # 导入 matplotlib 模块
import numpy as np                                         # 导入 numpy 模块
```

（3）按照“商品名称”进行分组统计数据。首先读取 Excel 文件，然后抽取指定的数据，最后使用 groupby() 函数实现分组统计求和，代码如下：

```
# 读取 Excel 文件
df=pd.read_excel('./data/data1.xlsx')
# 抽取数据
df=df[['商品名称','成交商品件数','成交码洋','加购人数']]
# 按照商品名称进行分组统计
# 通过 reset_index()函数将 groupby()的分组结果重新设置索引
df1 = df.groupby("商品名称").sum().reset_index()
```

（4）由于“商品名称”过于冗长，可能会影响图表的可视化效果，因此我们可以为“商品名称”设置“商品 ID”。这主要通过 for 循环和 DataFrame 对象的 loc 属性来实现，代码如下：

```
# 设置商品 ID
row_count = df1.shape[0]                                    # 行数
for i in range(row_count):                                  # 遍历行数
    df1.loc[i,'商品 ID']='B'+str(i+1)                       # 为商品 ID 赋值
```

（5）计算累计贡献率。首先将“商品 ID”设置为 DataFrame 的索引，然后按照“成交码洋”的降序对数据进行排序，最后使用 cumsum()函数和 sum()函数计算累计贡献率，代码如下：

```
df1 = df1.set_index('商品 ID').sort_values(by='成交码洋',ascending=False)    # 设置索引并降序排序
df1['累计贡献率']=(df1['成交码洋'].cumsum()/df1['成交码洋'].sum()).round(2)  # 计算累计贡献率
print(df1.head())                                                            # 输出前 5 条数据
```

运行程序，输出前 5 条数据，以检验计算结果，如图 1.16 所示。

商品ID	商品名称	成交商品件数	成交码洋	加购人数	累计贡献率
B16	Python从入门到项目实践（全彩版）	33016	3294996.8	72354	0.21
B33	零基础学Python（全彩版）	34495	2752701.0	72836	0.38
B18	Python项目开发案例集锦（全彩版）	12474	1596672.0	20069	0.48
B27	零基础学C语言（全彩版）	16223	1132365.4	28946	0.55
B19	SQL即查即用（全彩版）	12436	619312.8	25706	0.59

图 1.16　累计贡献率

（6）按照 ABC 分类法进行分类。首先使用 cut()函数切分数据并标记类别，代码如下：

```
# 使用 cut()函数标记类别
df1['类别']=pd.cut(df1['累计贡献率'], [0,0.8,0.9,1], labels=[u"A 类",u"B 类",u"C 类"])
```

运行程序，输出前 5 条和最后 5 条数据，以便查看分类后的数据，如图 1.17 所示。

商品ID	商品名称	成交商品件数	成交码洋	加购人数	累计贡献率	类别
B16	Python从入门到项目实践（全彩版）	33016	3294996.8	72354	0.21	A类
B33	零基础学Python（全彩版）	34495	2752701.0	72836	0.38	A类
B18	Python项目开发案例集锦（全彩版）	12474	1596672.0	20069	0.48	A类
B27	零基础学C语言（全彩版）	16223	1132365.4	28946	0.55	A类
B19	SQL即查即用（全彩版）	12436	619312.8	25706	0.59	A类
商品ID	商品名称	成交商品件数	成交码洋	加购人数	累计贡献率	类别
B23	零基础学ASP.NET（全彩版）	993	79241.4	1810	0.99	C类
B20	Visual Basic精彩编程200例（全彩版）	718	57296.4	1644	0.99	C类
B12	Java开发详解（全彩版）	411	48909.0	1124	1.00	C类
B10	JSP项目开发实战入门（全彩版）	618	43136.4	1189	1.00	C类
B22	玩转C语言程序设计（全彩版）	437	21762.6	1200	1.00	C类

图 1.17　分类后的数据

（7）将分类数据保存为 Excel 文件，以方便日后使用，代码如下：

```
# 导出为 Excel 文件
df1.to_excel('data/data1_hot.xlsx')
```

（8）数据可视化。我们绘制双 y 轴图表，主要使用 pandas 模块提供的内置绘图函数。其中，柱形图用于展示“销售收入”，而曲线图用于展示“收入比例”。在曲线图上，我们添加累计贡献率的文本标签，同时绘制两条水平线，分别作为累计贡献率 80%和 90%的分界线，从而使得累计贡献率在图表中一目了然，代码如下：

```
fig = plt.figure(figsize=(9,5))                                    # 设置画布大小
plt.rcParams['font.sans-serif']=['SimHei']                         # 解决中文乱码问题
# 取消科学记数法
plt.gca().get_yaxis().get_major_formatter().set_scientific(False)
# 绘制第一个柱形图
# 定义颜色为渐变蓝色
colors = plt.colormaps['Blues'](1-np.arange(len(df1))/len(df1))
df1['成交码洋'].plot(kind='bar',color=colors)
plt.ylabel(u'销售收入（元）')                                       # y 轴标签
# 绘制第二个累计贡献率曲线图
p=df1['累计贡献率']
# secondary_y 设置第二个 y 轴，linestyle 为线型，marker 为标记样式
p.plot(secondary_y=True, style='-o', linewidth=0.5)
# 在曲线图上添加文本标签
for a in range(p.shape[0]):
    plt.annotate(format(p.iloc[a], '.2'), xy=(a, p.iloc[a]), xytext=(a, p.iloc[a]+0.02),fontsize=7)
plt.ylabel(u'收入（比例）')                                         # 右侧 y 轴标签
# 添加 80%和 90%分界线
plt.axhline(0.8,color='red',linestyle='--')
plt.axhline(0.9,color='green',linestyle='--')
# 在分界线上添加文本
plt.text(x=28,y=0.81, s='80%分界线')
plt.text(x=28,y=0.91, s='90%分界线')
plt.show()                                                         # 显示图表
```

1.6.3 热销单品环比增长情况分析

通过 1.6.2 节内容，我们了解到热销产品中第一位的是 B16。接下来，我们将分析该产品的环比增长情况，并通过双 y 轴图表来展示。在图表中，柱形图将展示“成交商品件数”，而折线图则展示“环比增长率”。具体如图 1.18 所示。

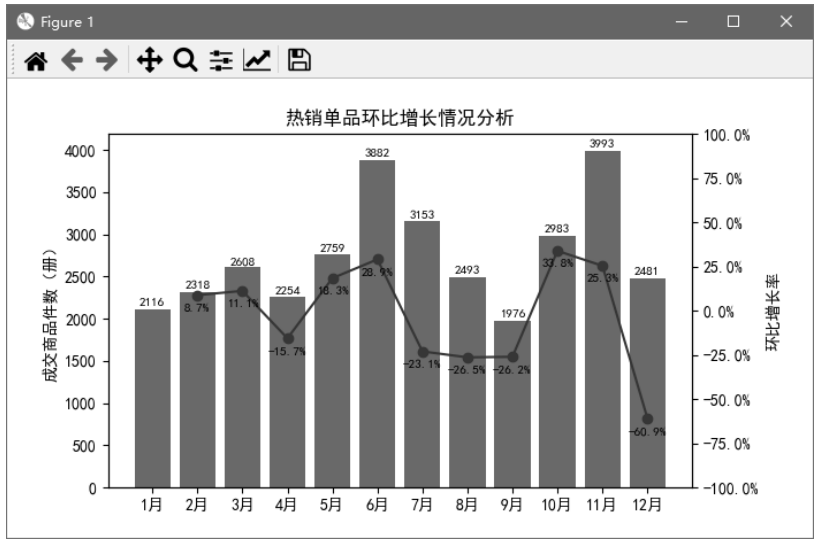


图 1.18 热销单品环比增长情况分析

下面介绍热销单品环比增长情况分析的实现过程（源码位置：资源包\Code\01\data_hot_one.py）。

（1）在项目目录下新建一个 Python 文件，并将其命名为 data_hot_one.py。

（2）导入相关模块，代码如下：

```
import pandas as pd                                  # 导入 pandas 模块
import matplotlib.pyplot as plt                      # 导入 matplotlib 的 pyplot 子模块
import matplotlib.ticker as mtick                    # 导入 matplotlib 的 ticker 子模块
```

（3）筛选指定数据，然后按月统计数据，代码如下：

```
df=pd.read_excel('./data/data1.xlsx')                # 读取 Excel 文件
df1=df[df['商品名称']=='Python 从入门到项目实践（全彩版）']  # 筛选数据
df1=df1[['时间','成交商品件数']]                      # 抽取数据
df1=df1.set_index('时间').sort_values(by=['时间'])    # 将时间设置为索引并进行升序排序
df2=df1.resample('M').sum().to_period('M')           # 按月统计数据
print(df2)
```

（4）计算环比增长率。环比增长率反映本月比上个月增长了多少，公式如下：

$$\text{环比增长率}=\frac{(\text{本月数}-\text{上月数})}{\text{上月数}}\times 100\%$$

在 pandas 中，我们可以使用 shift()方法将数据移至上一条，从而得到上个月的数据，然后通过上述计算公式得到环比增长率，代码如下：

```
df2['成交商品件数']=df2.sum(axis=1)                                  # 求和运算
df2['rate']=((df2['成交商品件数']-df2['成交商品件数'].shift())/df2['成交商品件数'])*100 # 环比增长率
print(df2.head())                                                    # 输出前 5 条数据
```

运行程序，输出前 5 条数据，看一下环比增长率，如图 1.19 所示。

时间	成交商品件数	rate
2023-01	2116	NaN
2023-02	2318	8.714409
2023-03	2608	11.119632
2023-04	2254	-15.705413
2023-05	2759	18.303733

图 1.19　环比增长率

说明

在运行结果中出现了 NaN，NaN 是“Not a Number”的缩写，即不是一个数，表示空值。之所以会出现空值，是因为没有上月数，也就是上一年 12 月份的数据。

（5）数据可视化。绘制双 y 轴图表，其中：使用 matplotlib 模块的 bar()函数绘制柱形图，以展示“成交商品件数”；使用 matplotlib 模块的 plot()函数绘制折线图，以展示环比增长率，代码如下：

```
# 设置月份
month=['1 月','2 月','3 月','4 月','5 月','6 月','7 月','8 月','9 月','10 月','11 月','12 月']
# x 轴数据
x = range(len(month))
# y 轴数据
y1=df2['成交商品件数']
y2=df2['rate']
fig = plt.figure(figsize=(7,4))                      # 创建并设置画布大小
plt.rcParams['font.sans-serif']=['SimHei']           # 解决中文乱码问题
plt.rcParams['axes.unicode_minus'] = False           # 用来正常显示负号
ax1 = fig.add_subplot(111)                           # 添加子图
plt.title('热销单品环比增长情况分析')                  # 图表标题
plt.xticks(x,labels=month)                           # x 轴刻度及标签
ax1.bar(x,y1,label=u"成交商品件数")                    # 柱形图
ax1.set_ylabel('成交商品件数（册）')                   # y 轴标签
# 为柱形图添加文本标签
for a,b in zip(x,y1):
    plt.text(a, b+20, b, ha='center', va= 'bottom',fontsize=8)
```

```
ax2 = ax1.twinx()                                    # 添加一条 y 轴坐标轴
ax2.plot(x,y2,color='r',linestyle='-',marker='o')    # 折线图
# 设置右侧 y 轴刻度为百分比格式
fmt = '%.1f%%'                                       # 将小数转换为百分比，保留 1 位小数
yticks = mtick.FormatStrFormatter(fmt)
ax2.yaxis.set_major_formatter(yticks)
ax2.set_ylim(-100,100)                               # 右侧 y 轴坐标范围
ax2.set_ylabel(u"环比增长率")                         # 右侧 y 轴标签
# 为折线图添加文本标签
for a,b in zip(x,y2):
    plt.text(a, b-10, '%.1f%%' % b, ha='center', va= 'bottom',fontsize=8)
plt.subplots_adjust(right=0.85)                      # 调整图表距右的空白
plt.show()                                           # 显示图表
```

说明

matplotlib.ticker 模块主要用于设置坐标轴刻度的格式和位置。它提供了一些常用的刻度格式，如科学记数法、百分比和日期等，并允许用户自定义刻度格式。此外，它还可以设置刻度的位置、间隔和标签等，以满足不同用户的需求。常用的刻度和格式化如下：

- ☑ AutoLocator：自动选择刻度间隔。
- ☑ FixedLocator：指定固定的刻度位置。
- ☑ MultipleLocator：指定刻度间隔。
- ☑ NullLocator：不显示刻度。
- ☑ FormatStrFormatter：指定格式化字符串。

1.6.4 加购人数和购买数量分析

加购人数和购买数量分析主要分析热销产品中 A 类产品的加入购物车人数和实际成交商品件数的对比情况，并通过结合柱形图和折线图来展示分析结果，如图 1.20 所示。

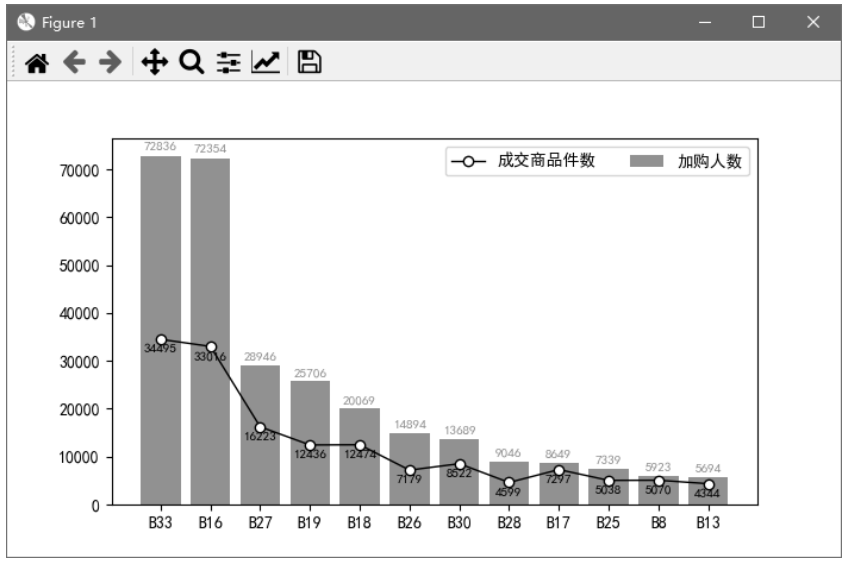


图 1.20 加购人数和购买数量分析

从运行结果中得知：不同产品的购买数量及加购人数呈现基本同步变化的趋势。排名前三的产品分别是 B33、B16 和 B27，其中 B33 和 B16 的购买数量差异不大，但它们与其他产品的购买数量相比，却有着

显著的差异，是其他产品的三倍或更多。

下面介绍加购人数和购买数量分析的实现过程（源码位置：资源包\Code\01\data_addcart.py）。

（1）在项目目录下新建一个 Python 文件，并将其命名为 data_addcart.py。

（2）导入相关模块，代码如下：

```
import pandas as pd                                    # 导入 pandas 模块
import matplotlib.pyplot as plt                        # 导入 matplotlib 模块
```

（3）在热销产品中筛选 A 类产品，代码如下：

```
# 读取 Excel 文件
df=pd.read_excel('./data/data1_hot.xlsx')
# 筛选 A 类产品
df=df[df['类别']=='A 类']
# 抽取数据并按照“加购人数”进行降序排序
df1=df[['商品 ID','成交商品件数','加购人数']].sort_values(by='加购人数',ascending=False)
```

（4）分别绘制折线图和柱形图，以便对实际成交商品件数和加购人数进行对比分析。其中：使用 matplotlib 模块的 plot()函数绘制折线图，用于展示成交商品件数；使用 matplotlib 模块的 bar()函数绘制柱形图，用于展示加购人数，代码如下：

```
# x 轴数据
x = range(len(df1['商品 ID']))
# y 轴数据
y1=df1['成交商品件数']
y2=df1['加购人数']
plt.rcParams['font.sans-serif']=['SimHei']                    # 解决中文乱码问题
fig = plt.figure(figsize=(7,4))                               # 创建画布并设置画布大小
# x 轴刻度及标签
plt.xticks(x,labels=df1['商品 ID'])
plt.plot(x,y1,color='black',linewidth=1,marker='o',mfc='w')   # 折线图
plt.bar(x,y2,color='CornflowerBlue')                          # 柱形图
# 为折线图添加文本标签
for a,b in zip(x,y1):
    plt.text(a,b-3000,b,ha = 'center',va = 'bottom',fontsize=8)
# 为柱形图添加文本标签
for a,b in zip(x,y2):
    plt.text(a,b+800,b,color='CornflowerBlue',ha = 'center',va = 'bottom',fontsize=8)
# 显示图例（2 列）
plt.legend(['成交商品件数', '加购人数'],ncol=2)
plt.show()                                                    # 显示图表
```

1.6.5 不同种类产品的销量占比情况分析

接下来，我们对热销产品中不同种类产品的销量占比情况进行分析。我们首先为每种产品打上标签，然后按照这些标签对产品进行分组并统计每一组的销量，最后通过饼形图展示不同种类产品的销量占比情况，如图 1.21 所示。

从运行结果中得知：在热销产品中，Python 的占比达到了 44.5%，几乎占到总数的一半。因此，建议对这些产品实施重点营销策略，并继续拓展其周边产品，以增加产品种类的多样化。

下面介绍不同种类产品销量占比情况分析的实现过程（源码位置：资源包\Code\01\data_type.py）。

（1）在项目目录下新建一个 Python 文件，并将其命名为 data_type.py。

（2）导入相关模块，代码如下：

```
import pandas as pd                                    # 导入 pandas 模块
import matplotlib.pyplot as plt                        # 导入 matplotlib 模块
```

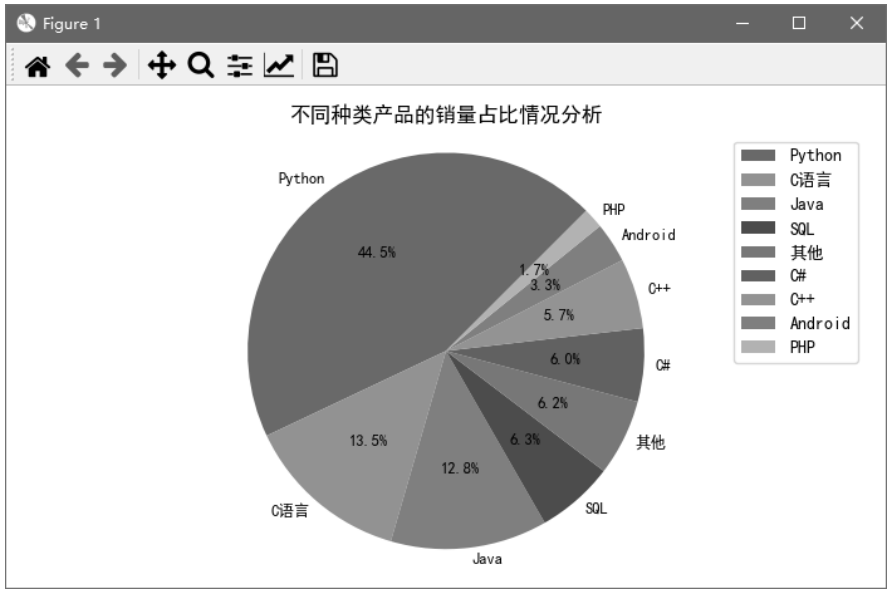


图 1.21　不同种类产品的销量占比情况分析

（3）为产品打标签。我们通过“商品名称”字符串中包含的关键字为产品打标签。我们首先定义标签函数 add_tag()，在该函数中使用 if 语句判断“商品名称”字符串中包含的关键字，然后应用该函数为产品打标签，代码如下：

```
# 读取 Excel 文件
df=pd.read_excel('./data/data1_hot.xlsx')
# 为产品打标签
# 定义标签函数
def add_tag(data):
    global tag
    tag='其他'
    if 'Android' in data:
        tag='Android'
    if 'C#' in data:
        tag='C#'
    if 'C 语言' in data:
        tag='C 语言'
    if 'C++' in data:
        tag='C++'
    if 'Java' in data:
        tag='Java'
    if 'PHP' in data:
        tag='PHP'
    if 'Python' in data:
        tag='Python'
    if 'SQL' in data:
        tag='SQL'
    return tag
# 应用 add_tag()函数为产品打标签
df['tag'] = df['商品名称'].apply(add_tag)
# 设置数据显示的编码格式为东亚宽度，以使列对齐
pd.set_option('display.unicode.east_asian_width', True)
pd.set_option('display.width',10000)                    # 显示宽度
pd.set_option('display.max_columns',1000)               # 最大列数
print(df.head())                                        # 输出前 5 条数据
```

运行程序，输出前 5 条数据，查看打标签后的结果，如图 1.22 所示。

	商品ID	商品名称	成交商品件数	成交码洋	加购人数	累计贡献率	类别	tag
0	B16	Python从入门到项目实践（全彩版）	33016	3294996.8	72354	0.21	A类	Python
1	B33	零基础学Python（全彩版）	34495	2752701.0	72836	0.38	A类	Python
2	B18	Python项目开发案例集锦（全彩版）	12474	1596672.0	20069	0.48	A类	Python
3	B27	零基础学C语言（全彩版）	16223	1132365.4	28946	0.55	A类	C语言
4	B19	SQL即查即用（全彩版）	12436	619312.8	25706	0.59	A类	SQL

图 1.22　打标签后的数据

（4）分组统计数据。按照 tag 标签对数据进行分组统计求和，并根据“成交商品件数”进行降序排序，代码如下：

```
# 抽取数据
df=df[['tag','成交商品件数']]
# 按照 tag 标签分组统计并进行降序排序
df1=df.groupby('tag').sum().sort_values('成交商品件数',ascending=False)
```

（5）数据可视化。使用 matplotlib 模块的 pie()函数绘制饼形图，以便分析不同种类产品的销量占比情况，代码如下：

```
plt.rcParams['font.sans-serif']=['SimHei']                    # 解决中文乱码问题
plt.figure(figsize=(7,4))                                     # 创建画布并设置画布大小
labels = df1.index                                            # 饼图标签
sizes = df1['成交商品件数']                                    # 饼图数据
plt.pie(sizes,                                                # 饼图数据
        labels=labels,                                        # 添加区域水平标签
        labeldistance=1.06,                                   # 设置各扇形标签（图例）与圆心的距离
        autopct='%.1f%%',                                     # 设置百分比的格式，这里保留一位小数
        startangle=45,                                        # 设置饼图的初始角度
        radius = 0.5,                                         # 设置饼图的半径
        center = (0.2,0.2),                                   # 设置饼图的原点
        textprops = {'fontsize':9, 'color':'k'},              # 设置文本标签的属性值
        pctdistance=0.6)                                      # 设置百分比标签与圆心的距离
# 设置 x，y 轴刻度一致，保证饼图为圆形
plt.axis('equal')
plt.title('不同种类产品的销量占比情况分析')                    # 图表标题
plt.legend()                                                  # 图例
plt.tight_layout()                                            # 图形元素自适应
plt.show()                                                    # 显示图表
```

1.6.6　工作日与周末销量对比分析

下面进行工作日与周末销量对比分析。我们首先对“日期”进行处理，从日期中找到工作日和周末，并对它们进行标记，然后对工作日 5 天和周末两天的数据进行分组统计求和，最后绘制双折线图以对这些数据进行对比分析，如图 1.23 所示。

从运行结果中得知：我们虽然凭直觉认为周末休息时购买的人数可能会更多，但经过实际对比分析发现，工作日的销量高于周末的销量。这种差异可能是由于工作日有 5 天，而周末只有两天。

下面介绍工作日与周末销量对比分析的实现过程（源码位置：资源包\Code\01\data_workday_weekend.py）。

（1）在项目目录下新建一个 Python 文件，并将其命名为 data_workday_weekend.py。

（2）导入相关模块，代码如下：

```
import pandas as pd                                           # 导入 pandas 模块
import matplotlib.pyplot as plt                               # 导入 matplotlib 模块
```

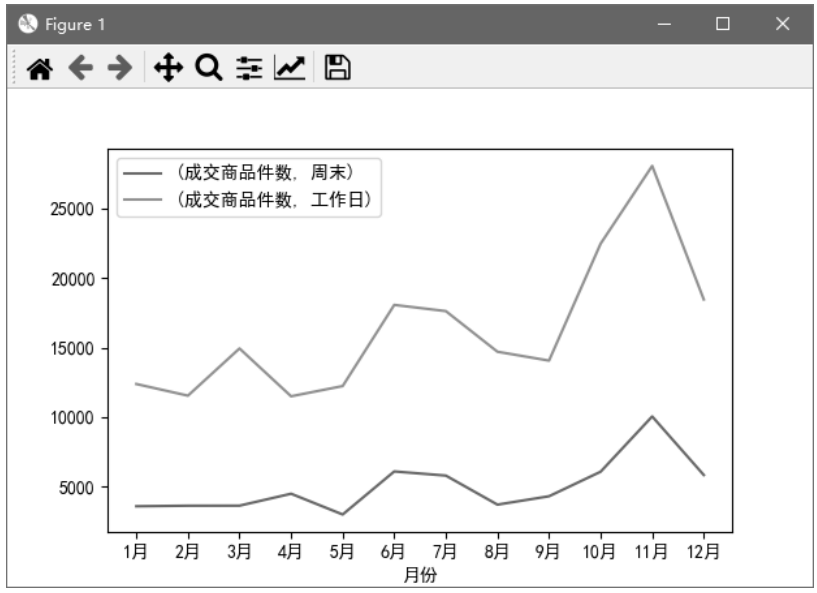


图 1.23　工作日与周末销量对比分析

（3）对日期数据进行处理。我们主要使用 dt 对象的 month 属性和 dayofweek 属性获取月份和星期。通过 dayofweek 属性获取的星期是以数字形式表示的，其中 0 表示星期一，以此类推。然后，我们依据星期进行判断，将数字小于 5 的标记为工作日，将数字大于 5 的标记为周末，代码如下：

```
# 读取 Excel 文件
df=pd.read_excel('./data/data1.xlsx')
# 获取并添加月份和星期（星期一=0，星期日=6）
df['月份'],df['星期']=df['时间'].dt.month,df['时间'].dt.dayofweek
# 标记工作日和周末
df.loc[df[df['星期']<5].index,'标记']='工作日'
df.loc[df[df['星期']>=5].index,'标记']='周末'
```

运行程序，查看标记后的数据，如图 1.24 所示。

（4）数据分组统计。首先抽取指定的数据，然后按照月份和标记分组统计求和，代码如下：

```
# 抽取数据
df=df[['月份','成交商品件数','标记']]
# 按月份和标记分组统计求和
df1=df.groupby(['月份','标记']).sum()
print(df1.head(6))                    # 输出前 6 条数据
```

运行程序，看一下分组统计后前 3 个月的数据，如图 1.25 所示。

	时间	商品名称	一级类目	...	月份	星期	标记
0	2023-10-31	零基础学Python（全彩版）	图书	...	10	1	工作日
1	2023-10-31	Python从入门到项目实践（全彩版）	图书	...	10	1	工作日
2	2023-10-31	Python项目开发案例集锦（全彩版）	图书	...	10	1	工作日
3	2023-10-31	Python编程锦囊（全彩版）	图书	...	10	1	工作日
4	2023-10-31	SQL即查即用（全彩版）	图书	...	10	1	工作日

图 1.24　标记后的数据

		成交商品件数
月份	标记	
1	周末	3583
	工作日	8793
2	周末	3623
	工作日	7914
3	周末	3626
	工作日	11307

图 1.25　分组统计后前 3 个月的数据

（5）绘制双折线图，主要使用 DataFrame 对象自带的绘图函数 plot()直接对处理后的数据进行绘图，方便快捷，代码如下：

```
# 设置月份
month=['1 月','2 月','3 月','4 月','5 月','6 月','7 月','8 月','9 月','10 月','11 月','12 月']
# 解决中文乱码问题
plt.rcParams['font.sans-serif']=['SimHei']
# 将最内层的行索引转换为列索引然后绘制折线图
df1.unstack().plot(stacked=True)
plt.xticks(range(1,13,1),month)          # 设置 x 轴刻度及标签
plt.legend()                             # 显示图例
plt.show()                               # 显示图表
```

1.7 项 目 运 行

通过前述步骤，我们已经设计并完成了“热销产品销售数据统计分析”项目的开发。“热销产品销售数据统计分析”项目目录包含 7 个 Python 脚本文件，如图 1.26 所示。

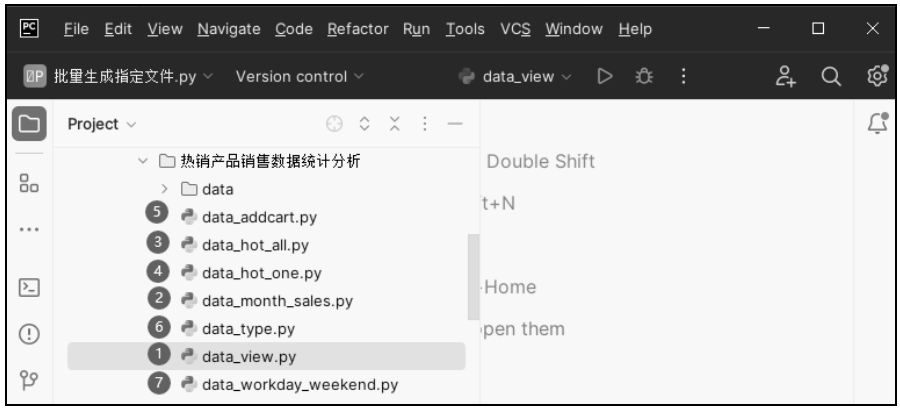


图 1.26　项目目录

接下来，我们按照开发过程运行脚本文件，以检验我们的开发成果。例如，运行 data_view.py，双击该文件，右侧“代码窗口”将显示全部代码，然后右击，在弹出的快捷菜单中选择 Run 'data_view'命令（见图 1.27），即可运行程序。

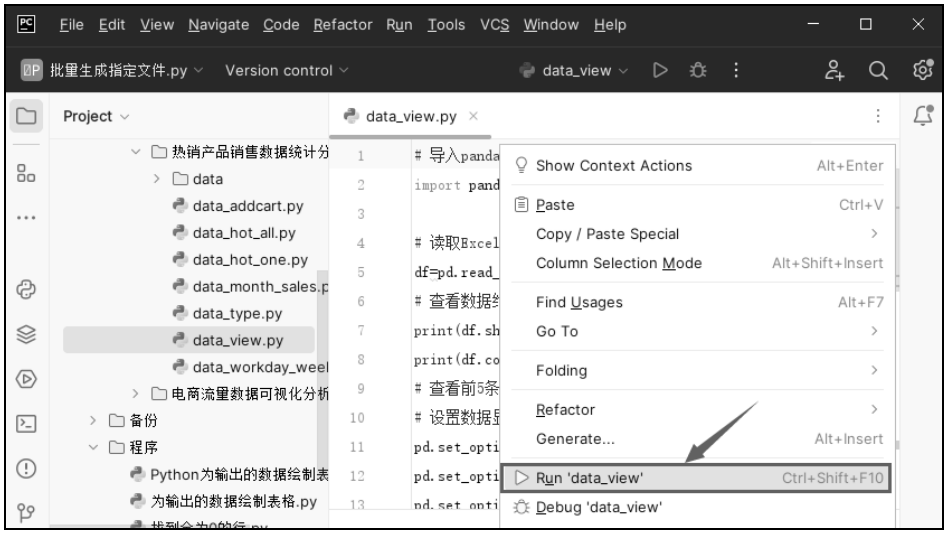


图 1.27　运行 data_view.py

其他脚本文件按照图 1.26 给出的顺序运行，这里就不再赘述了。

1.8 源 码 下 载

本章虽然详细地讲解了如何通过 pandas 模块、numpy 模块和 matplotlib 模块实现“热销产品销售数据统计分析”的各个功能，但给出的代码都是代码片段，而非完整的源代码。为了方便读者学习，本书提供了用于下载完整源代码的二维码。

源码下载