沈阳航空航天大学

**学生实验报告**



|  |  |
| --- | --- |
| 课程名称 | 数据分析编程基础B |
| 实验名称 | 实验四 科学计算可视化实验 |
| 专业班级 | 物联网2202 |
| 学生学号 | 223428010210 |
| 学生姓名 | 陈梓欣 |
| 指导教师 | 李济瀚 |
| 实验时间 | 2023-2024学年第二学期 |
| 实验地点 | 机械馆410-1 |

## 实验目的

熟悉实验环境，分析和了解numpy库和matplotlib库的程序可视化编写与运用；培养学生函数在程序可视化编写过程中逻辑分析与实际动手的编程能力。

## 二、实验要求

1. 实验学时：2学时。
2. 使用Python环境编写和调试程序。
3. 了解numpy库的可视化编写与使用。
4. 了解matplotlib库的可视化编写与使用。
5. 实验应通过调试，并获得正确的结果。
6. 对程序中重要的内容做必要的注释。

## 三、实验准备

文件的打开与读取的方法。

## 四、实验程序（要求：程序运行结果，程序代码写注释）

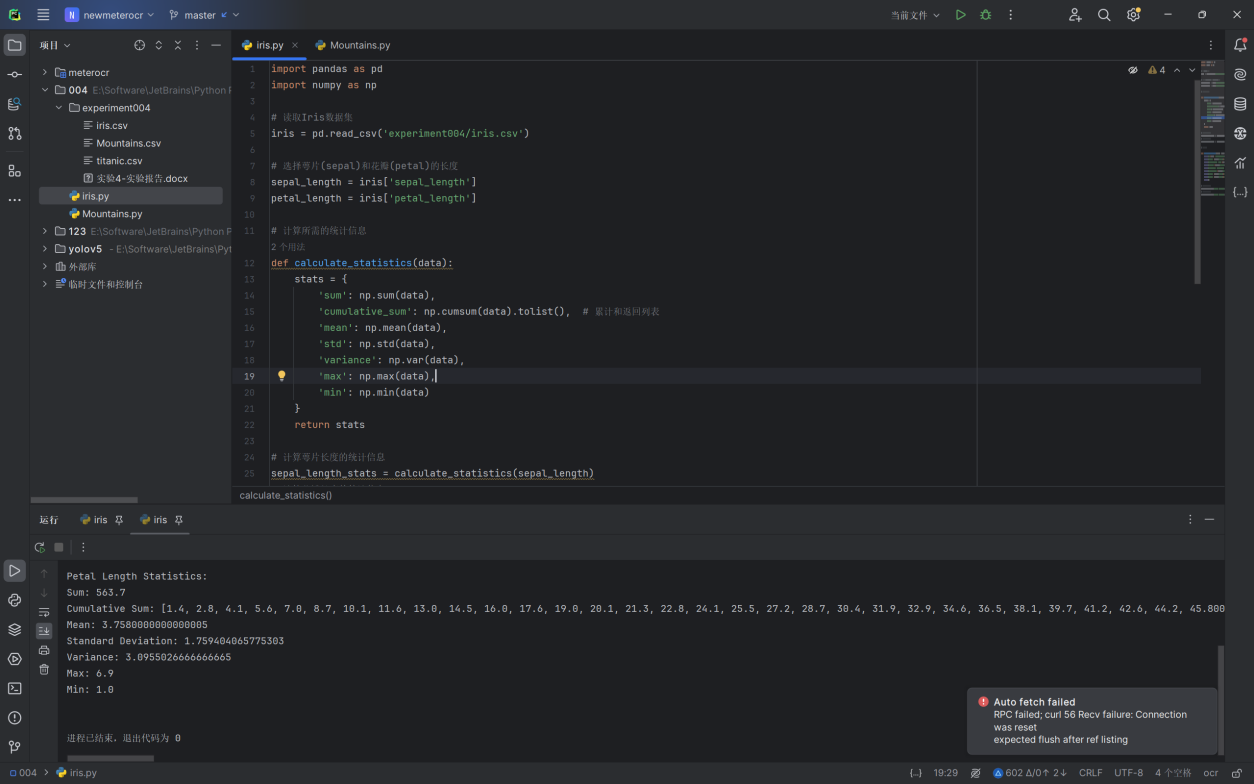
**一、编程题**

1. 读取iris数据集中鸢尾花的萼片，花瓣长度信息。

**要求：**对其进行求和，累计和，均值，标准差、方差、最大值和最小值。（数据集中sepal:萼片、petal:花瓣、species:种类），（数据集中species:种类、setoa:山鸢尾、versicolor:变色鸢尾、virginica：维吉尼鸢尾）。

程序代码及注释：

import pandas as pd  
import numpy as np  
  
# 读取Iris数据集  
iris = pd.read\_csv('experiment004/iris.csv')  
  
# 选择萼片(sepal)和花瓣(petal)的长度  
sepal\_length = iris['sepal\_length']  
petal\_length = iris['petal\_length']  
  
# 计算所需的统计信息  
def calculate\_statistics(data):  
 stats = {  
 'sum': np.sum(data),  
 'cumulative\_sum': np.cumsum(data).tolist(), # 累计和返回列表  
 'mean': np.mean(data),  
 'std': np.std(data),  
 'variance': np.var(data),  
 'max': np.max(data),  
 'min': np.min(data)  
 }  
 return stats  
  
# 计算萼片长度的统计信息  
sepal\_length\_stats = calculate\_statistics(sepal\_length)  
# 计算花瓣长度的统计信息  
petal\_length\_stats = calculate\_statistics(petal\_length)  
  
# 输出统计信息  
def print\_statistics(name, stats):  
 print(f"{name} Statistics:")  
 print(f"Sum: {stats['sum']}")  
 print(f"Cumulative Sum: {stats['cumulative\_sum']}")  
 print(f"Mean: {stats['mean']}")  
 print(f"Standard Deviation: {stats['std']}")  
 print(f"Variance: {stats['variance']}")  
 print(f"Max: {stats['max']}")  
 print(f"Min: {stats['min']}")  
 print()  
  
# 输出萼片长度的统计信息  
print\_statistics("Sepal Length", sepal\_length\_stats)  
# 输出花瓣长度的统计信息  
print\_statistics("Petal Length", petal\_length\_stats)

程序运行结果：

**图1 打印信息**

1. 读取titanic数据集中的乘客的性别、年龄等信息。

**要求**：实现显示其前50行的数据信息，实现调用函数显示数据信息总览，并实现数据中生还者的饼图绘制，利用均值填充数据中“年龄”的缺失值实现。

程序代码及注释：

import pandas as pd

import matplotlib.pyplot as plt

# 读取Titanic数据集

titanic = pd.read\_csv('experiment004/titanic.csv')

# 调整pandas显示设置，确保显示所有列

pd.set\_option('display.max\_columns', None) # 显示所有列

pd.set\_option('display.width', 1000) # 设置显示宽度

pd.set\_option('display.max\_colwidth', None) # 设置每列的最大宽度

# 显示前50行数据

print("前50行的数据信息：")

print(titanic.head(50))

# 定义一个函数显示数据信息总览

def data\_overview(df):

print("\n数据信息总览：")

print(df.info())

print("\n数据的描述性统计：")

print(df.describe(include='all'))

# 调用函数显示数据信息总览

data\_overview(titanic)

# 用均值填充年龄的缺失值

titanic['age'].fillna(titanic['age'].mean(), inplace=True)

# 绘制生还者的饼图

survived\_counts = titanic['survived'].value\_counts()

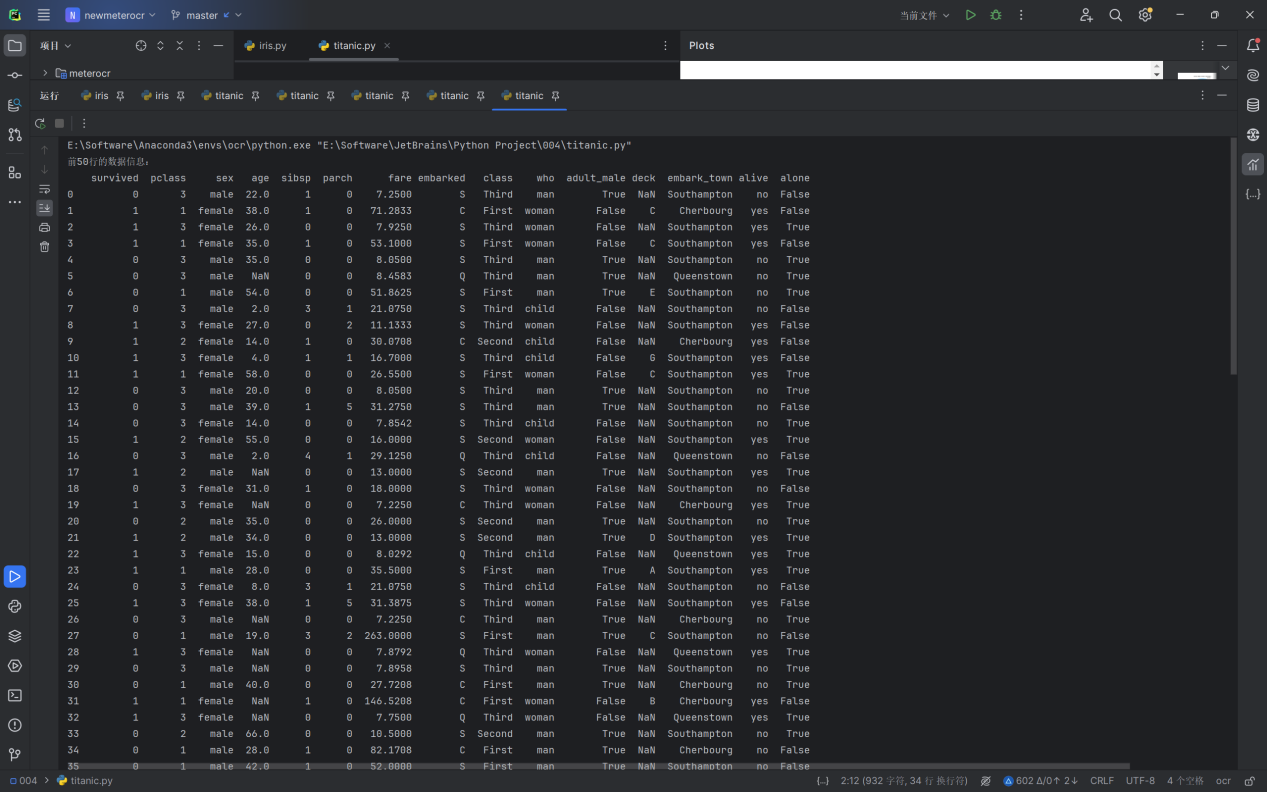
plt.figure(figsize=(8, 6))

plt.pie(survived\_counts, labels=['Not Survived', 'Survived'], autopct='%1.1f%%', startangle=140, colors=['lightcoral', 'lightskyblue'])

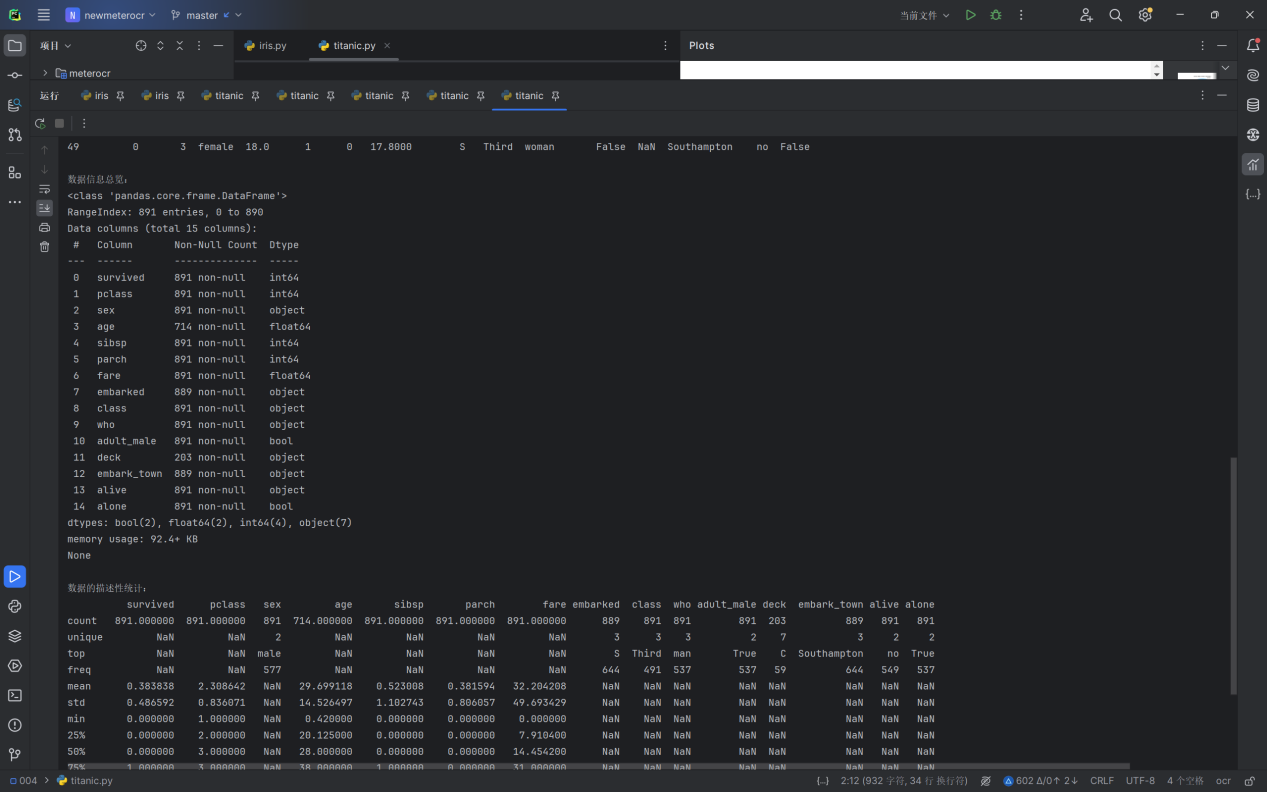
plt.title('Survival Rate of Titanic Passengers')

plt.axis('equal') # 确保饼图为圆形

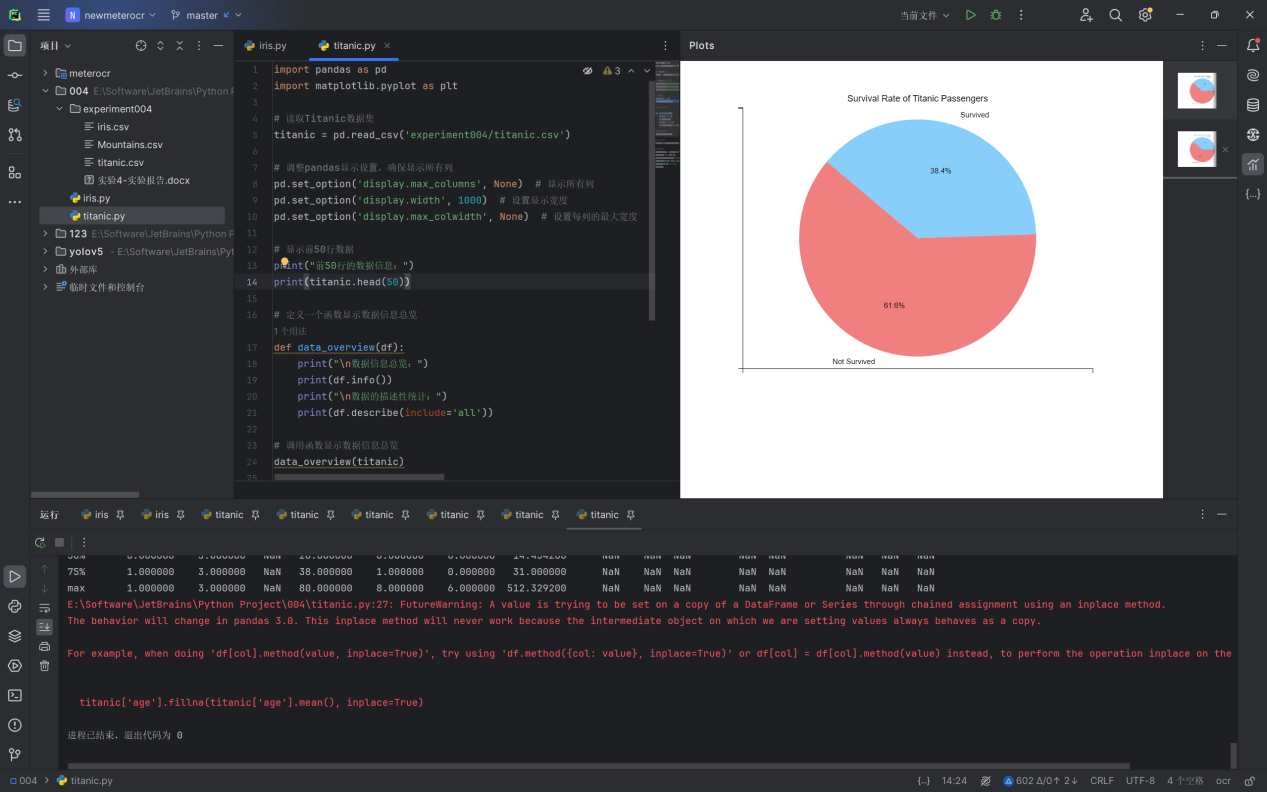
plt.show()

程序运行结果：

**图2 打印信息(1)**



**图3 打印信息(2)**



**图4 输出饼状图**

3. 读取Mountains数据集的信息，实现山峰可视化功能。

**要求：**编写过程中要求设置图片显示的主题样式，显示中文指定”SimHei”字体，可以运用def函数形式编写，实现对数据集中“Height(m)”：“Height”,“Ascents bef.2004”：“ Success”,“Failed attempts bef.2004”：“Failed”重构、清洗（用0填充“成功”或者“失败”的缺失项）以及可视化功能，用图形的方式显示山峰的数量和海拔，显示山峰的数量和年份，用子图形式显示海拔与登顶、登顶与排名。

程序代码及注释：

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import re

# 读取Mountains数据集，尝试使用不同的编码

try:

mountains = pd.read\_csv('experiment004/mountains.csv', encoding='utf-8')

except UnicodeDecodeError:

mountains = pd.read\_csv('experiment004/mountains.csv', encoding='latin1')

# 打印列名以确认其正确性

print("列名：", mountains.columns)

# 调整pandas显示设置，确保显示所有列

pd.set\_option('display.max\_columns', None)

pd.set\_option('display.width', 1000)

pd.set\_option('display.max\_colwidth', None)

# 设置图片显示的主题样式

sns.set\_theme(style="whitegrid")

# 设置中文字体为SimHei

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

# 清洗数据集中的非数字值

def clean\_numeric\_values(series):

# 移除所有非数字字符

series = series.apply(lambda x: re.sub(r'\D', '', str(x)))

# 将清理后的值转换为整数

series = series.apply(lambda x: int(x) if x.isdigit() else 0)

return series

# 重构和清洗数据集

def clean\_and\_restructure\_data(df):

df = df.rename(columns={

'Height (m)': 'Height',

'Ascents bef. 2004': 'Success',

'Failed attempts bef. 2004': 'Failed'

})

df['Success'] = clean\_numeric\_values(df['Success'])

df['Failed'] = clean\_numeric\_values(df['Failed'])

return df

# 检查重命名是否成功

mountains = clean\_and\_restructure\_data(mountains)

print("重命名后的列名：", mountains.columns)

# 可视化功能

def visualize\_data(df):

plt.figure(figsize=(14, 10))

# 图1: 山峰的数量和海拔

plt.subplot(2, 2, 1)

sns.histplot(df['Height'], bins=30, kde=True)

plt.title('山峰的数量和海拔')

plt.xlabel('海拔 (m)')

plt.ylabel('数量')

# 图2: 山峰的数量和年份（假设有Year列）

if 'Year' in df.columns:

plt.subplot(2, 2, 2)

sns.histplot(df['Year'], bins=30, kde=True)

plt.title('山峰的数量和年份')

plt.xlabel('年份')

plt.ylabel('数量')

# 图3: 海拔与登顶

plt.subplot(2, 2, 3)

sns.scatterplot(x='Height', y='Success', data=df)

plt.title('海拔与登顶')

plt.xlabel('海拔 (m)')

plt.ylabel('成功次数')

# 图4: 登顶与排名（假设有Rank列）

if 'Rank' in df.columns:

plt.subplot(2, 2, 4)

sns.scatterplot(x='Success', y='Rank', data=df)

plt.title('登顶与排名')

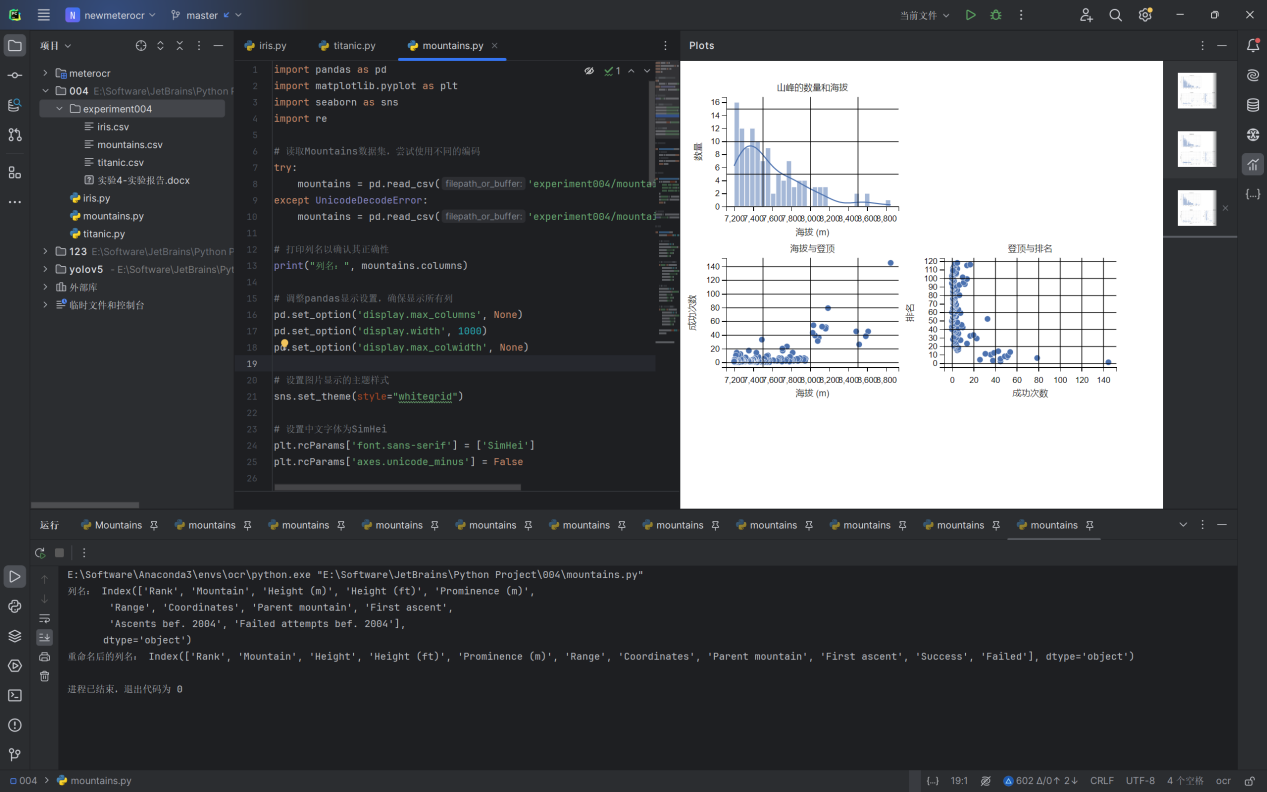
plt.xlabel('成功次数')

plt.ylabel('排名')

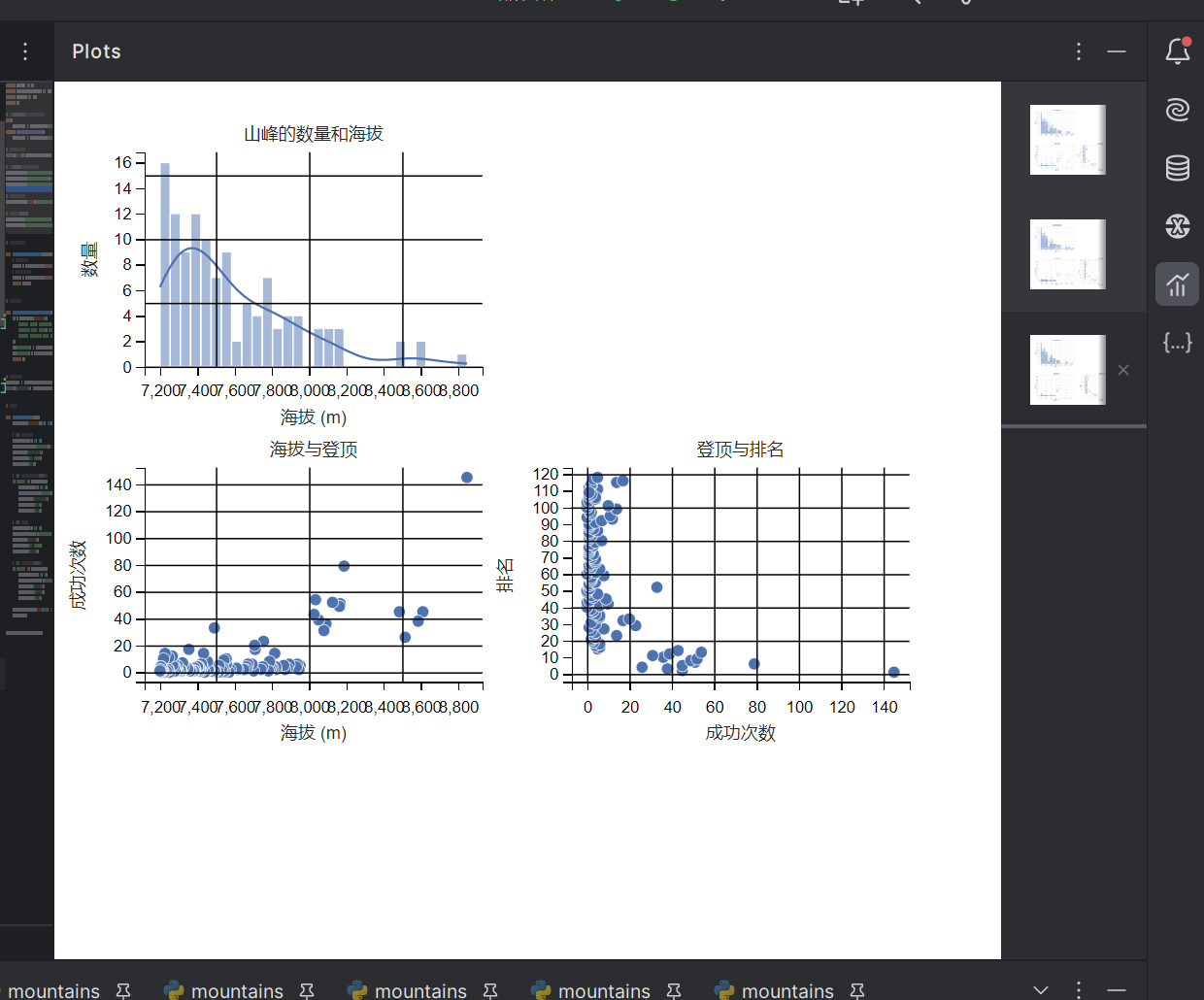
plt.tight\_layout(pad=4.0, w\_pad=5.0, h\_pad=5.0)

plt.show()

visualize\_data(mountains)

程序运行结果：

**图5 打印信息**



**图6 绘制子图**