绘制不同品类之间的散点图

# 导入 pandas 库，并使用 pd 作为别名

import pandas as pd

# 导入 numpy 库，并使用 np 作为别名

import numpy as np

# 从 matplotlib 库导入 pyplot 模块，并使用 plt 作为别名

from matplotlib import pyplot as plt

# 定义一个名为 linear\_test 的函数，接受一个文件名作为参数

def linear\_test(filename):

# 使用 pandas 的 read\_excel 方法读取指定的 Excel 文件，并将其内容赋值给 df 变量

df=pd.read\_excel(filename)

# 初始化一些空列表，用于之后存放不同列的数据

listq = []

lists = []

listc = []

listj = []

listl = []

listy = []

# 使用一个循环遍历 df 的每一行

for i in range(331968):

# 将每一行的各列数据依次添加到对应的列表中

listq.append(df.values[i, 0])

lists.append(df.values[i, 1])

listc.append(df.values[i, 2])

listj.append(df.values[i, 3])

listl.append(df.values[i, 4])

listy.append(df.values[i, 5])

# 将处理后的列表赋值给相应的变量，以便后续使用

a=listq

b=lists

c=listc

d=listj

e=listl

f=listy

# 创建一个字典，其中键为列名，值为相应的列表

dict1={"qie":a,"shuisheng":b,"huacai":c,"shiyongjun":d,"lajiao":e,"huaye":f}

# 使用 pandas 将字典转换为 DataFrame 对象，并赋值给 s 变量

s = pd.DataFrame(dict1)

# 打印 DataFrame 的列名

print(s.columns)

# 创建一个figure对象，赋值给 fig 变量

fig = plt.figure()

# 在 figure 中添加一个子图，并赋值给 ax1 变量

ax1 = fig.add\_subplot(1, 1, 1)

# 为子图设置标题为 'Linear Test'

ax1.set\_title('Linear Test')

# 为子图设置 x 轴标签为 'category'，y 轴标签也为 'category'

ax1.set\_xlabel('category')

ax1.set\_ylabel('category')

# 在子图中创建一个散点图，以 a 为 x 轴，b 为 y 轴，并用黑色展示散点

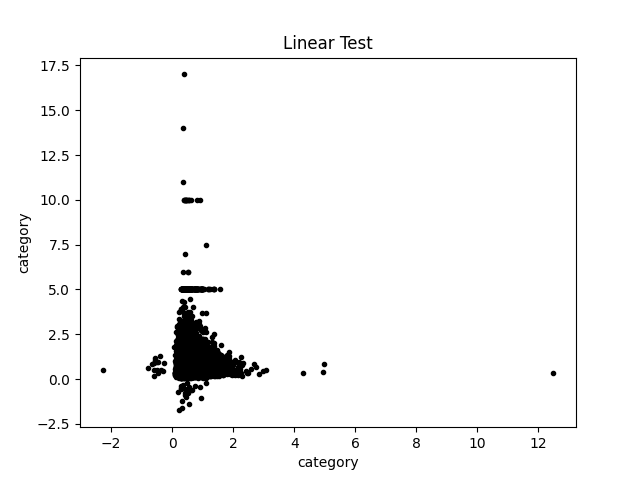
ax1.scatter(a, b, c='k', marker='.')

# 将 figure 保存为名为 'linear\_test.png' 的图片文件

plt.savefig('linear\_test.png')

# 显示图片，如果是在 jupyter notebook 中运行，这一步是必须的，否则图片不会在 notebook 中显示

plt.show()



绘制了品类之间的散点图，发现不同品类之间呈现一定的线性关系，因此采用正态性检测先进行分析

运用皮尔逊相关系数

# 创建一个空的 Python 数据结构，这里用它来保存 Excel 文件的每一列的均值和标准差

dict1 = {}

# 从 Excel 文件中读取数据，这里假设文件名为 'data.xlsx'

s = pd.read\_excel('data.xlsx')

# 对每列数据进行遍历，计算每列的均值和标准差，并将结果存入 dict1 中

for i in s.columns:

mean = s[i].mean() # 计算均值

std = s[i].std() # 计算标准差

print('均值为：%.2f，标准差为：%.2f' % (mean, std)) # 打印均值和标准差

print('------') # 打印分隔线

# 对每列数据按照升序排序，并打印排序后的前五行

s.sort\_values(by=i, inplace=True)

print(s.head())

# 重置索引，原来的索引会被替换为一个新的、从0开始的索引

s\_r = s.reset\_index(drop=False)

print("----------\n", s\_r.head())

# 为每行数据生成一个 p 值，该值等于该行的索引减去0.5再除以总行数

s\_r['p'] = (s\_r.index - 0.5) / len(s\_r)

# 为每列数据生成一个 q 值，该值等于该列数据值减去该列的均值再除以该列的标准差

s\_r['q'] = (s\_r[i] - mean) / std

print(s\_r.head()) # 打印前五行

print('------') # 打印分隔线

# 计算每列数据的描述统计信息，包括四分之一位数和四分之三位数，并打印结果

st = s[i].describe()

x1 ,y1 = 0.25, st['25%'] # 四分之一位数

x2 ,y2 = 0.75, st['75%'] # 四分之三位数

print('四分之一位数为：%.2f，四分之三位数为：%.2f' % (y1,y2)) # 打印四分之一位数和四分之三位数

print('------') # 打印分隔线

# 通过 matplotlib 创建图形，图形包含一个散点图，显示每对数据之间的关系，并打印出来

fig = plt.figure(figsize = (10,6))

ax1 = fig.add\_subplot(2,1,1) # 创建子图1

ax1.scatter(dict1[i],dict1[i]) # 在子图上绘制散点图

plt.tight\_layout() # 使图形布局紧凑

plt.show() # 显示图形

# 通过 matplotlib 创建第二个图形，包含一个直方图和一个核密度估计图，显示每列数据的分布情况，并打印出来

fig = plt.figure(figsize=(10, 6))

ax2 = fig.add\_subplot(3,1,2) # 创建子图2

s.hist(i,bins=30, alpha=0.5, ax=ax2) # 在子图上绘制直方图

s.plot(i,kind='kde', secondary\_y=True, ax=ax2) # 在子图上绘制核密度估计图

plt.grid() # 显示网格线

plt.show() # 显示图形

# 通过 matplotlib 创建第三个图形，包含一个 QQ 图，显示每列数据是否符合正态分布，并打印出来

fig = plt.figure(figsize=(10, 6))

ax3 = fig.add\_subplot(4,1,3) # 创建子图3

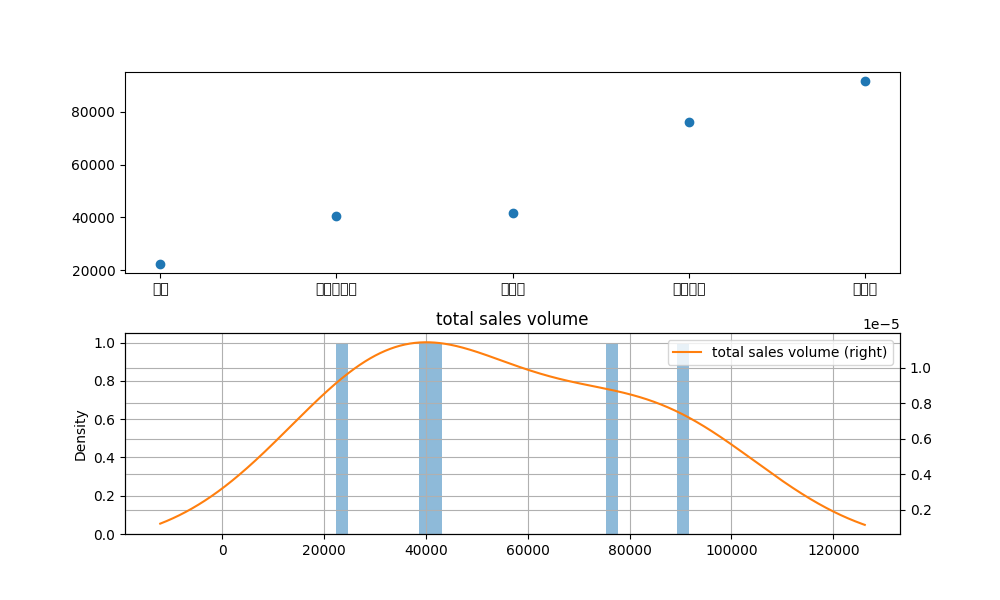
ax3.plot(s\_r['p'],s\_r[i],'k.',alpha = 0.1) # 在子图上绘制 QQ 图的主线部分

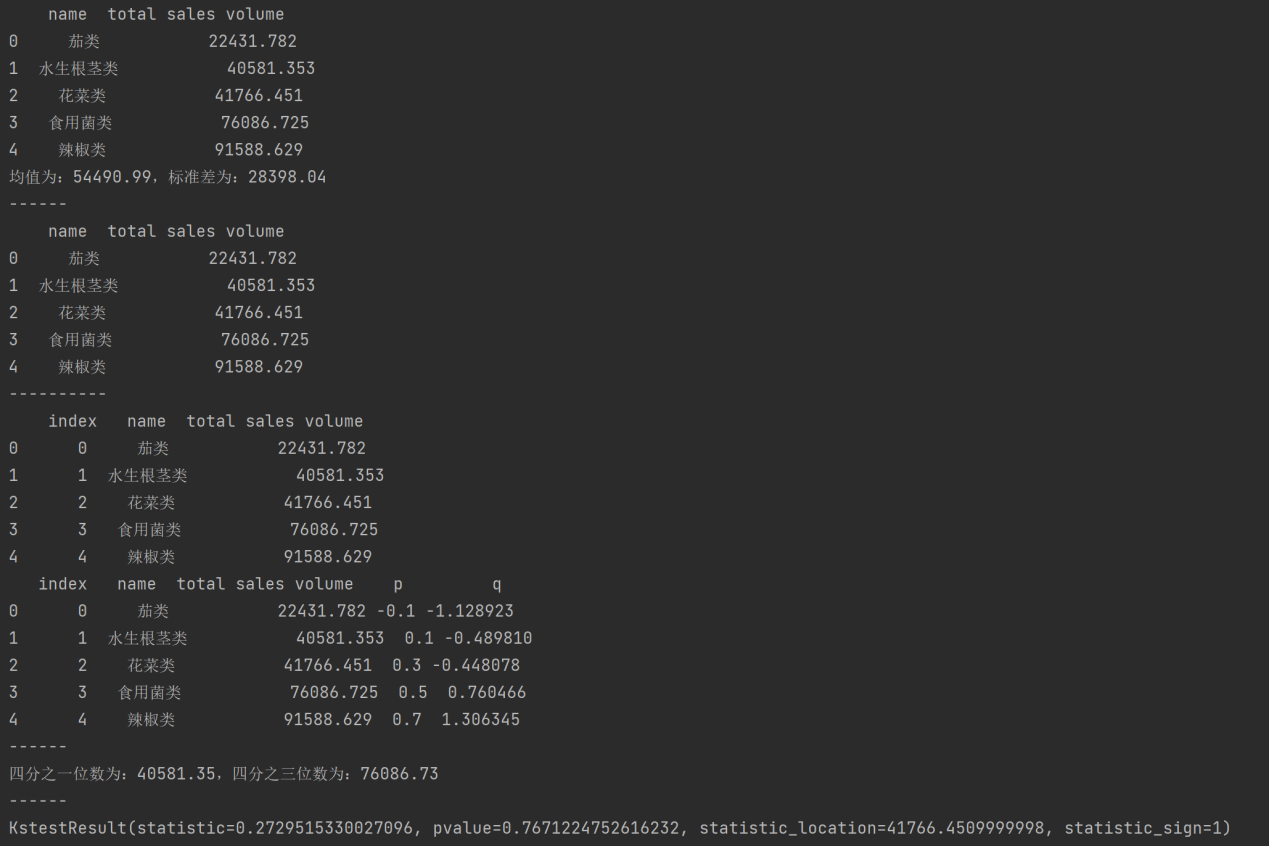
ax3.plot([x1,x2],[y1,y2],'-r') # 在子图上绘制 QQ 图中的辅助线部分

plt.grid() # 显示网格线

linear\_test(r"C:\Users\阿韩想养二哈\Desktop\工作簿2.xlsx")

data(r"C:\Users\阿韩想养二哈\Desktop\工作簿2.xlsx")





相关性矩阵+热力图

# 导入 pandas 库，用于数据处理和分析

import pandas as pd

# 导入 numpy 库，用于数值计算

import numpy as np

# 导入 seaborn 库，用于数据可视化

import seaborn as sns

# 导入 matplotlib.pyplot 库，用于数据可视化

import matplotlib.pyplot as plt

# 定义一个函数，用于读取指定的Excel文件

# 该函数接受一个文件名作为参数，不返回任何值（None）

def data(filename):

# 使用 pandas 的 read\_excel 函数读取 Excel 文件，将数据保存到 DataFrame 中

# 这里用 df 作为 DataFrame 的变量名，是一个常用的约定

df = pd.read\_excel(filename)

# 打印 DataFrame 的内容，方便查看读取的数据是否正确

print(df)

# 计算 DataFrame 中各列之间的相关性，返回一个相关性矩阵

# method='pearson' 表示使用皮尔逊相关系数计算相关性

# 将相关性矩阵保存到 rho 变量中，以备后续使用

rho =df.corr(method='pearson')

print(rho)

# 设置 matplotlib 的全局参数，让图形中的字体都使用 SimHei 字体

# 这可以让图形中的中文正确显示

plt.rcParams['font.family'] = ['SimHei']

# 设置 matplotlib 的全局参数，让图形中的负号使用正常的字体，而不是默认的斜体

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

# 使用 seaborn 的 heatmap 函数绘制相关性矩阵的热力图

# annot=True 表示在热力图上显示数字

sns.heatmap(rho, annot=True)

# 设置图形的标题，使用指定的字体大小

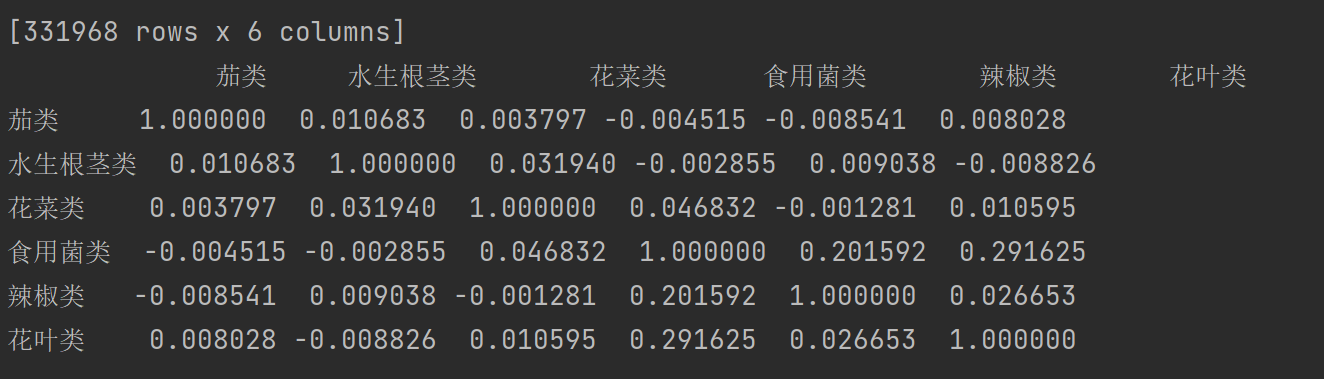
plt.title('Heat Map', fontsize=18)

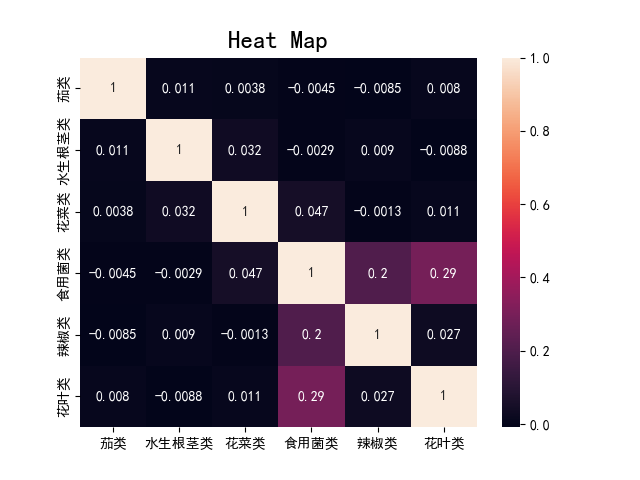
# 将图形保存为一个 PNG 图片文件，设置图片的 DPI 为 300

plt.savefig('heatmap1.png', dpi=300)

# 显示图形，让用户看到绘制的结果

plt.show()  
  
data(r"C:\Users\阿韩想养二哈\Desktop\工作簿2.xlsx")





运用斯皮尔曼相关系数

# 导入 pandas 库，pandas 是一个用于数据操作和分析的库

import pandas as pd

# 导入 numpy 库，numpy 是一个用于数值计算的库

import numpy as np

# 导入 seaborn 库，seaborn 是一个基于 matplotlib 的数据可视化库

import seaborn as sns

# 导入 matplotlib.pyplot 库，这是数据可视化的库

import matplotlib.pyplot as plt

# 定义一个函数，用于读取 Excel 文件中的数据

# 这个函数接受一个参数 'filename'，代表要读取的 Excel 文件的文件名

def data(filename):

# 使用 pandas 的 read\_excel 函数读取 Excel 文件，将数据保存到 DataFrame 中

# df 是 DataFrame 的实例，用于进行后续的数据操作

df = pd.read\_excel(filename)

# 打印读取到的 DataFrame 的内容，方便调试和查看数据是否正确读取

print(df)

# 使用 pandas 的 corr 方法计算 DataFrame 中各列之间的相关性，返回一个相关性矩阵

# method='spearman' 表示使用斯皮尔曼等级相关系数计算相关性，这比默认的皮尔逊相关系数更为稳健

# 将相关性矩阵保存到 rho 变量中，以备后续使用

rho = df.corr(method='spearman')

print(rho)

# 设置 matplotlib 的全局参数，让图形中的字体都使用 SimHei 字体

# 这可以让图形中的中文正确显示

plt.rcParams['font.family'] = ['SimHei']

# 设置 matplotlib 的全局参数，让图形中的负号使用正常的字体，而不是默认的斜体

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

# 使用 seaborn 的 heatmap 函数绘制相关性矩阵的热力图

# annot=True 表示在热力图上显示数字

sns.heatmap(rho, annot=True)

# 设置图形的标题，使用指定的字体大小

plt.title('Heat Map', fontsize=18)

# 将图形保存为一个 PNG 图片文件，设置图片的 DPI 为 300（每英寸的像素点数为300）

plt.savefig('heatmap1.png', dpi=300)

# 显示图形，让用户看到绘制的结果

plt.show()  
data(r"C:\Users\阿韩想养二哈\Desktop\工作簿2.xlsx")



