

《商务统计与数据分析》期末考核作业

论文题目： 商务统计与课程分析期末作业

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | **学号** | **承担工作介绍** | **分数** |
| **陈一铭** | **2020150176057** | **题目解答** |  |
| **方鸿杰** | **2020150176056** | **题目解答** |  |
| **何岑昊** | **2020150176050** | **题目解答** |  |

**教师评语：**

**教师签名：**

**2020 秋双学士《商务统计与数据分析》期末课程作业**

## 1、 课程作业要求：

1. 按规定完成相应的数据分析任务，装订成册（加封面，加注名字和学号）后提交；
2. 字体使用宋体小四号，英文使用 Times New Roman；单倍行距；
3. 独立完成作业，**作业内容需包括任务相关的 python 代码和结果**；
4. 请在规定期限内提交作业，否则按照未提交作业计算成绩；
5. 如果文献引用参考，请加注参考文献来源；
6. 注意整个作业手册和绘制的图表的美观性以及专业性，这里会有10**分评价成绩**，每一页右上角加注页码；
7. 整个课业报告页数不少于 20 页；
8. 请将本作业要求作为封面之后的第一页装订在报告里面。

## 2、数据分析任务

* 1. **餐饮团购活动.xlsx**这个数据集里记录了某团购网站上某段时间内餐饮团购活动内容的具体信息。**餐饮团购店铺.xlsx**这个数据集中记录了团购店铺的相关数据，可以从这个数据集了解到每家餐饮店铺的特征数据。基于这两个数据，进行相应的数据整理与清洗，试着回答以下问题：(合计20分)

1. 有多少家店铺参加了团购活动？
2. 最受消费者喜爱的Top5餐饮类型是哪些？
3. 销售最多的店铺Top10是哪些？
4. 团购优惠的额度是否和购买数量存在关系？什么关系？
5. 不同行政区域，口碑良好（选择评分大于 4 分）的商家数量？

# 使用python对**京东超市销售数据.csv** 数据集中的合适变量进行基本的描述性统计分析，主要包括数据的频数分析、数据的集中趋势分析、数据离散程度分析、数据的分布、变量间的相关性以及一些基本的统计图形比如直方图和盒状图，并对结果进行管理学意义上的讨论。（小组为单位完成；合计20分）；

* 1. 成都市青羊区锦江区是公认的小学初中教育资源最好的行政区，因此二手房价相对较高。课程作业文件夹中包含**青羊区二手房.csv** 格式数据文档文件一份。数据为青羊区挂牌二手房的相关数据。要求：整理与描述青羊区挂牌二手房的相关信息，建立二手房房价的预测模型，揭示影响二手房房价的主要因素并对结果进行管理学意义上的阐述（小组为单位完成；合计20分）；
  2. 基于纽约地铁站人流量数据（**Newyork.csv**），使用FineBI或Python进行数据的可视化描述天气状况如何影响地铁客流?这种影响效应是否存在差异性（在不同的条件下影响效应具有差异）？输出图表结果并对结果进行较为详细的阐述（小组为单位完成；合计 20 分）
  3. **51job.csv**存放了51job 网站成都地区的关于数据分析相关的职位的信息，试使用python的numpy 和 pandas 包对数据进行整理清洗与分析，回答如下问题：（个人完成；合计 20 分）

1. 哪个区对数据分析相关的岗位需求最高？
2. 对数据分析相关有需求的企业具有什么特征？
3. 薪资最高的 10 家企业？
4. 数据分析相关的职位有哪些并进行排序？
5. 对岗位职位进行可视化词云分析，并对结果进行阐述；

**第一题：餐饮团购活动.xlsx**这个数据集里记录了某团购网站上某段时间内餐饮团购活动内容的具体信息。**餐饮团购店铺.xlsx**这个数据集中记录了团购店铺的相关数据，可以从这个数据集了解到每家餐饮店铺的特征数据。基于这两个数据，进行相应的数据整理与清洗，试着回答以下问题：(合计20分)

1.有多少家店铺参加了团购活动？

2.最受消费者喜爱的Top5餐饮类型是哪些？

3.销售最多的店铺Top10是哪些？

4.团购优惠的额度是否和购买数量存在关系？什么关系？

5.不同行政区域，口碑良好（选择评分大于4分）的商家数量？

Main.py 主函数

from get\_participation\_restaurant import \*

from get\_favour\_catering\_type import \*

from get\_analysis\_restaurant\_data import \*

from get\_restaurant\_sale\_num import \*

from show\_discount\_relationship import \*

from show\_good\_restaurant\_num import \*

if \_\_name\_\_ == '\_\_main\_\_':

Excel = xlwt.Workbook()

purchase\_data\_sheet = read\_purchase\_data()

# write\_analysis\_purchase\_data(Excel, purchase\_data\_sheet)

restaurant\_data\_sheet = read\_restaurant\_data()

# write\_analysis\_restaurant\_data(Excel, restaurant\_data\_sheet)

print\_participation\_restaurant\_num(purchase\_data\_sheet)

print\_top5\_catering\_type(purchase\_data\_sheet)

print\_top10\_restaurant\_sale(purchase\_data\_sheet)

show\_discount\_relationship\_with\_sale\_num(purchase\_data\_sheet)

show\_good\_restaurant\_num\_with\_area(restaurant\_data\_sheet)

print('Finished!')

get\_analysis\_restaurant\_data.py 从列表中获得餐馆和餐馆优惠信息

import xlrd # xlrd 2.0.1版本只支持.xls文件,需要安装1.2.0版本（pip install xlrd==1.2.0）

import xlwt

import re

def read\_purchase\_data():

sheet = xlrd.open\_workbook('./餐饮团购活动.xlsx').sheet\_by\_name('Sheet1')

sheet\_rows = sheet.nrows

new\_sheet = []

for row in range(1, sheet\_rows):

sheet\_row\_dict = {}

sheet\_row\_dict['团购名'] = sheet.cell(row, 0).value

sheet\_row\_dict['店名'] = sheet.cell(row, 1).value

sheet\_row\_dict['购买人数'] = int(sheet.cell(row, 4).value)

evaluation = sheet.cell(row, 5).value

if evaluation == '暂无评价':

sheet\_row\_dict['团购评价'] = 0

else:

sheet\_row\_dict['团购评价'] = float(evaluation)

discount\_value = sheet.cell(row, 8).value

market\_value = sheet.cell(row, 9).value

if discount\_value == '' or market\_value == '':

discount\_info = sheet.cell(row, 3).value

discount\_value = re.search(r'仅售(.\*?)元', discount\_info).group(1)

market\_value = re.search(r'价值(.\*?)元', discount\_info).group(1)

sheet\_row\_dict['团购价'] = float(discount\_value)

sheet\_row\_dict['市场价'] = float(market\_value)

new\_sheet.append(sheet\_row\_dict)

# print(sheet\_row\_dict)

return new\_sheet

def read\_restaurant\_data():

sheet = xlrd.open\_workbook('./餐饮团购店铺.xlsx').sheet\_by\_name('Sheet1')

sheet\_rows = sheet.nrows

new\_sheet = []

for row in range(1, sheet\_rows):

sheet\_row\_dict = {}

sheet\_row\_dict['店名'] = sheet.cell(row, 0).value

sheet\_row\_dict['关键词'] = sheet.cell(row, 1).value

address = sheet.cell(row, 6).value

if address == '':

continue

city = re.search(r'(.\*?)市', address).group(1)

area = re.search(r'市(.\*?)(区|县|市|其他)', address).group(1)

if area == '':

area = '其他'

else:

area = area + address[re.search(r'市(.\*?)(区|县|市|其他)', address).span()[1] - 1]

sheet\_row\_dict['城市'] = city

sheet\_row\_dict['行政区'] = area

sheet\_row\_dict['评分'] = float(sheet.cell(row, 3).value)

new\_sheet.append(sheet\_row\_dict)

# print(sheet\_row\_dict)

return new\_sheet

def write\_analysis\_purchase\_data(Excel, new\_sheet):

sheet = Excel.add\_sheet('Sheet1')

sheet.write(0, 0, '店名')

sheet.write(0, 1, '团购名')

sheet.write(0, 2, '购买人数')

sheet.write(0, 3, '团购评价')

sheet.write(0, 4, '团购价')

sheet.write(0, 5, '市场价')

row = 1

for sheet\_row in new\_sheet:

sheet.write(row, 0, sheet\_row['店名'])

sheet.write(row, 1, sheet\_row['团购名'])

sheet.write(row, 2, sheet\_row['购买人数'])

sheet.write(row, 3, sheet\_row['团购评价'])

sheet.write(row, 4, sheet\_row['团购价'])

sheet.write(row, 5, sheet\_row['市场价'])

row += 1

Excel.save('餐饮团购数据整理.xls')

return

Excel.save('餐饮团购数据整理.xls')

return

def write\_analysis\_restaurant\_data(Excel, new\_sheet):

sheet = Excel.add\_sheet('Sheet2')

sheet.write(0, 0, '店名')

sheet.write(0, 1, '关键词')

sheet.write(0, 2, '城市')

sheet.write(0, 3, '行政区')

sheet.write(0, 4, '评分')

row = 1

for sheet\_row in new\_sheet:

sheet.write(row, 0, sheet\_row['店名'])

sheet.write(row, 1, sheet\_row['关键词'])

sheet.write(row, 2, sheet\_row['城市'])

sheet.write(row, 3, sheet\_row['行政区'])

sheet.write(row, 4, sheet\_row['评分'])

row += 1

Excel.save('餐饮团购数据整理.xls')

return

get\_participation\_restaurant.py 得到参加活动的餐馆数量

# 统计参加活动的餐馆

def get\_participation\_restaurant(purchase\_data\_sheet):

num = 1

first\_restaurant = purchase\_data\_sheet[0]['店名']

for data in purchase\_data\_sheet:

if data['店名'] != first\_restaurant:

num += 1

first\_restaurant = data['店名']

return num

def print\_participation\_restaurant\_num(purchase\_data\_sheet):

print('参与活动的店铺数量：%d' % (get\_participation\_restaurant(purchase\_data\_sheet)))

get\_favour\_catering\_type.py 得到最受欢迎的餐馆种类

# 得到top5受欢迎的餐饮类型

# 将团购活动的购买人数排序输出，得到一个按购买人数从高到低排列的 <团购名, 购买人数> 序列通过排列的序列和自定义的餐饮类型，匹配得到每种餐饮类型的总购买人数。通过排序后的每种餐饮类型的总购买人数序列，得到top5受欢迎的餐饮类型。

def get\_sorted\_sale\_num(purchase\_data\_sheet):

sale\_num = {}

sorted\_sale\_num = {}

for data in purchase\_data\_sheet:

sale\_num[data['团购名']] = int(data['购买人数'])

for item in sorted(sale\_num.items(), key=lambda item: item[1], reverse=True):

sorted\_sale\_num[item[0]] = item[1]

return sorted\_sale\_num

def get\_favour\_catering\_type(sorted\_sale\_num):

types = ['自助', '串串', '火锅', '烧肉', '涮', '鸡公煲', '烧烤', '鱼', '海鲜']

type\_sale\_num = {}

sorted\_type\_sale\_num = {}

for catering\_type in types:

type\_sale\_num[catering\_type] = 0

for data in sorted\_sale\_num:

for catering\_type in types:

if catering\_type in data:

type\_sale\_num[catering\_type] += sorted\_sale\_num[data]

for item in sorted(type\_sale\_num.items(), key=lambda item: item[1], reverse=True):

sorted\_type\_sale\_num[item[0]] = item[1]

return sorted\_type\_sale\_num

def print\_top5\_catering\_type(purchase\_data\_sheet):

print('\ntop5受欢迎的餐饮类型:')

sorted\_type\_sale\_num = get\_favour\_catering\_type(get\_sorted\_sale\_num(purchase\_data\_sheet))

i = 0

for key, value in sorted\_type\_sale\_num.items():

if i in range(5):

print('Top %d : ' % (i + 1) + key + ' -> 总人数: %d' % value)

i += 1

else:

break

get\_restaurant\_sale\_num.py 获得餐馆销售数量

# 得到销售Top10的店铺

def get\_sorted\_restaurant\_sale\_num(purchase\_data\_sheet):

num = 1

restaurant\_sale\_num = {}

sorted\_restaurant\_sale\_num = {}

first\_restaurant = purchase\_data\_sheet[0]['店名']

restaurant\_sale\_num[first\_restaurant] = purchase\_data\_sheet[0]['购买人数']

for data in purchase\_data\_sheet:

if data['店名'] != first\_restaurant:

num += 1

first\_restaurant = data['店名']

restaurant\_sale\_num[first\_restaurant] = data['购买人数']

else:

restaurant\_sale\_num[first\_restaurant] += data['购买人数']

for item in sorted(restaurant\_sale\_num.items(), key=lambda item: item[1], reverse=True):

sorted\_restaurant\_sale\_num[item[0]] = item[1]

return sorted\_restaurant\_sale\_num

def print\_top10\_restaurant\_sale(purchase\_data\_sheet):

print('\n销售最多的店铺Top10:')

sorted\_restaurant\_sale\_num = get\_sorted\_restaurant\_sale\_num(purchase\_data\_sheet)

# print(sorted\_restaurant\_sale\_num)

i = 0

for key, value in sorted\_restaurant\_sale\_num.items():

if i in range(10):

print('Top %d : ' % (i + 1) + key + ' -> 总人数: %d' % value)

i += 1

else:

break

show\_discount\_relationship.py 绘制销售数量与打折的关系散点图

import matplotlib.pyplot as plt

def show\_discount\_relationship\_with\_sale\_num(purchase\_data\_sheet):

discounts = []

sale\_nums = []

for data in purchase\_data\_sheet:

discount = data['团购价'] / data['市场价']

sale\_num = data['购买人数']

discounts.append(discount)

sale\_nums.append(sale\_num)

plt.xlim(xmax=1, xmin=0)

plt.ylim(ymax=5000, ymin=0)

plt.plot(discounts, sale\_nums, '.')

plt.show()

return

show\_good\_restaurant\_num.py 绘制不同行政区良好商家数量的条形图

import matplotlib.pyplot as plt

import matplotlib as mpl

import numpy as np

def show\_good\_restaurant\_num\_with\_area(restaurant\_data\_sheet):

good\_restaurant\_num\_with\_area = {}

for data in restaurant\_data\_sheet:

if data['行政区'] not in good\_restaurant\_num\_with\_area.keys():

good\_restaurant\_num\_with\_area[data['行政区']] = 0

if data['评分'] >= 4:

good\_restaurant\_num\_with\_area[data['行政区']] += 1

y = good\_restaurant\_num\_with\_area

x = np.arange(len(y.keys()))

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

plt.bar(x, height=y.values(), align='center', color='green', width=0.5)

plt.xticks(x, y.keys(), size='small', rotation=60)

for a, b in zip(x, y.values()):

plt.text(a, b + 0.05, '%.0f' % b, ha='center', va='bottom', fontsize=7)

plt.show()

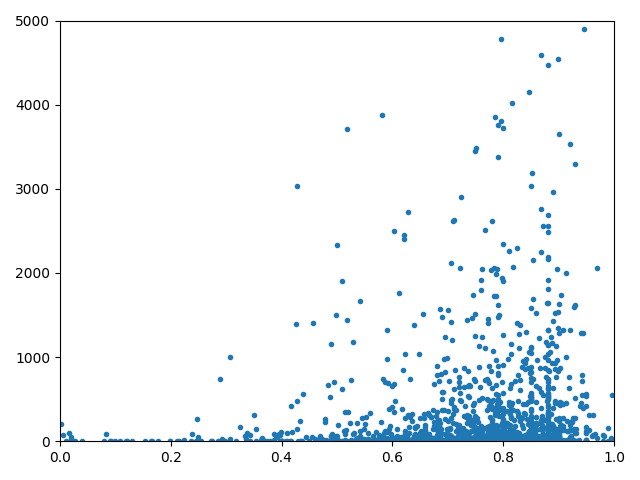
#print(good\_restaurant\_num\_with\_area)

return

（1）（2）（3）运行结果：



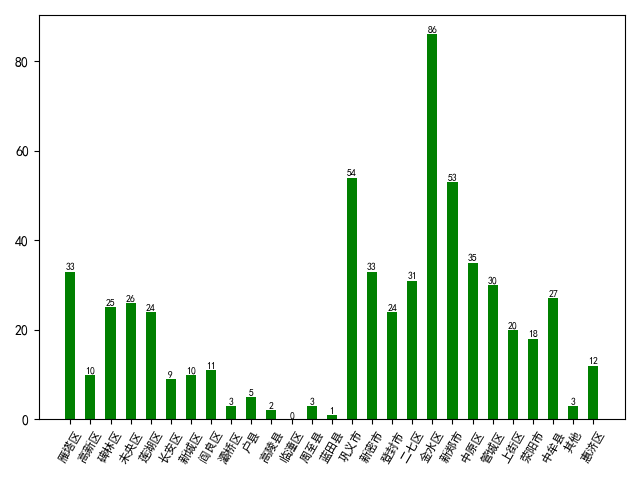
（4）运行结果：



注：纵轴为销售数量，横轴为销售数量

结论：不存在关系，虽然打8折到9折时的购买数量相比不打折或其他更高折扣时的购买数量要多。但从散点图上来看并不存在明显关系

（5）运行结果：



注：横轴为行政区，纵轴为良好商家数量

第二题：使用python对**京东超市销售数据.csv** 数据集中的合适变量进行基本的描述性统计分析，主要包括数据的频数分析、数据的集中趋势分析、数据离散程度分析、数据的分布、变量间的相关性以及一些基本的统计图形比如直方图和盒状图，并对结果进行管理学意义上的讨论。（小组为单位完成；合计20分）；

Main.py 主函数

import numpy as np

import pandas as pd

import seaborn as sns

from matplotlib import pyplot as plt

sns.set(font="SimHei", font\_scale=1)

pd.set\_option("max\_rows", None)

pd.set\_option("max\_columns", None)

# 京东超市销售数据分析

if \_\_name\_\_ == '\_\_main\_\_':

# 获取数据

jd\_data = pd.read\_csv("./京东超市销售数据.csv")

jd\_data.head()

jd\_data[["每笔订单利润", "销售额预测", "每名客户销售额", "利润", "销售额"]] \

= jd\_data[["每笔订单利润", "销售额预测", "每名客户销售额", "利润", "销售额"]]\

.applymap(lambda x: x.replace(",", "")).astype("float")

# 数值变量的集中趋势和离散趋势

jd\_data["利润率"] = jd\_data["利润"] / jd\_data["每名客户销售额"]

jd\_data.describe()

jd\_data.corr()

# 销售额和利润率频数分析

pd.pivot\_table(jd\_data,

index=["地区"],

values=["销售额"],

aggfunc={"销售额": [np.max, np.mean]}

).plot(kind='barh')

pd.pivot\_table(jd\_data,

index=["地区"],

values=["利润率"],

aggfunc={"利润率": [np.max, np.mean]}

).plot(kind='barh')

plt.show()

# 地区和销售额的相关性（散点图和小提琴图表示）

sns.stripplot(x="地区", y="销售额",

data=jd\_data, jitter=True,

hue="装运状态", dodge=True,

palette="Accent")

plt.show()

sns.violinplot(x=jd\_data["地区"], y=jd\_data["销售额"],

palette="Greens")

plt.show()

# 销售对象和物品类别相关性（直方图表示）

sns.catplot(x="类别", col="细分",

data=jd\_data, kind="count",

height=8, aspect=.7,

palette="Set2")

plt.show()

#散点图总集

sns.pairplot(jd\_data, palette="Greens")

plt.figure(figsize=(40, 40), dpi=150)

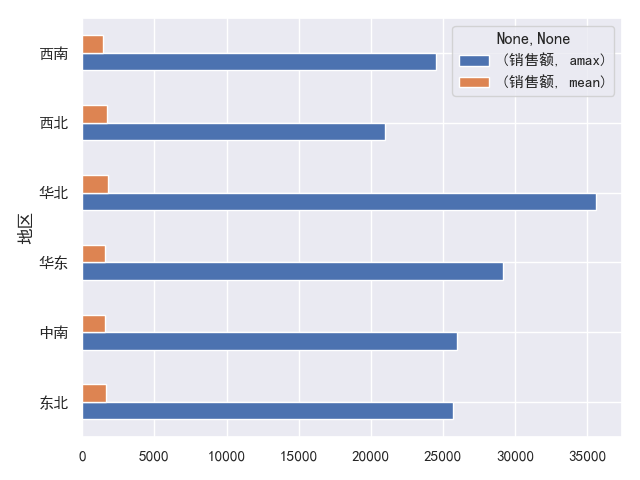
plt.show()

数据变量的集中趋势和离散趋势：

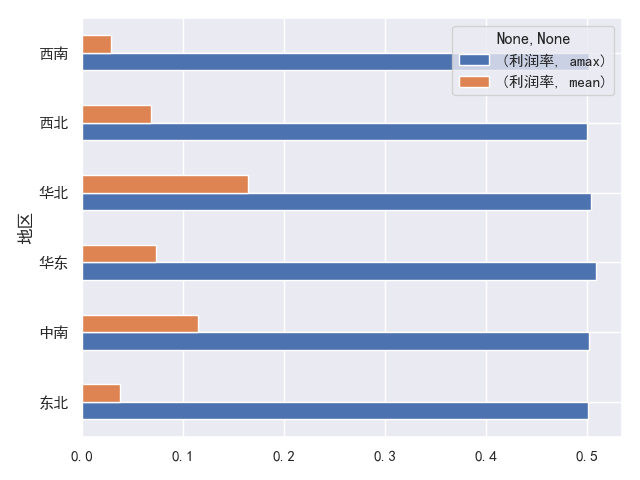


销售额和利润率频数分析：

销售额：

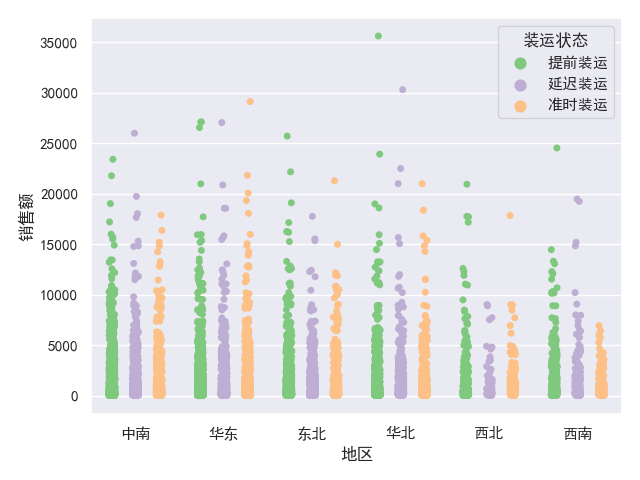


利润率：

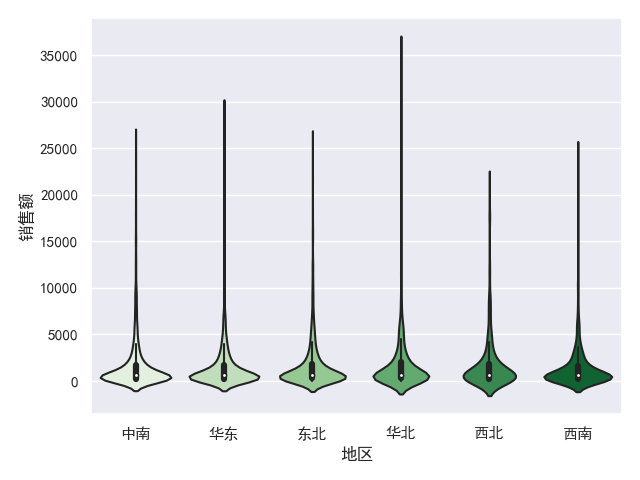


地区和销售额的相关性（散点图和小提琴图表示）：

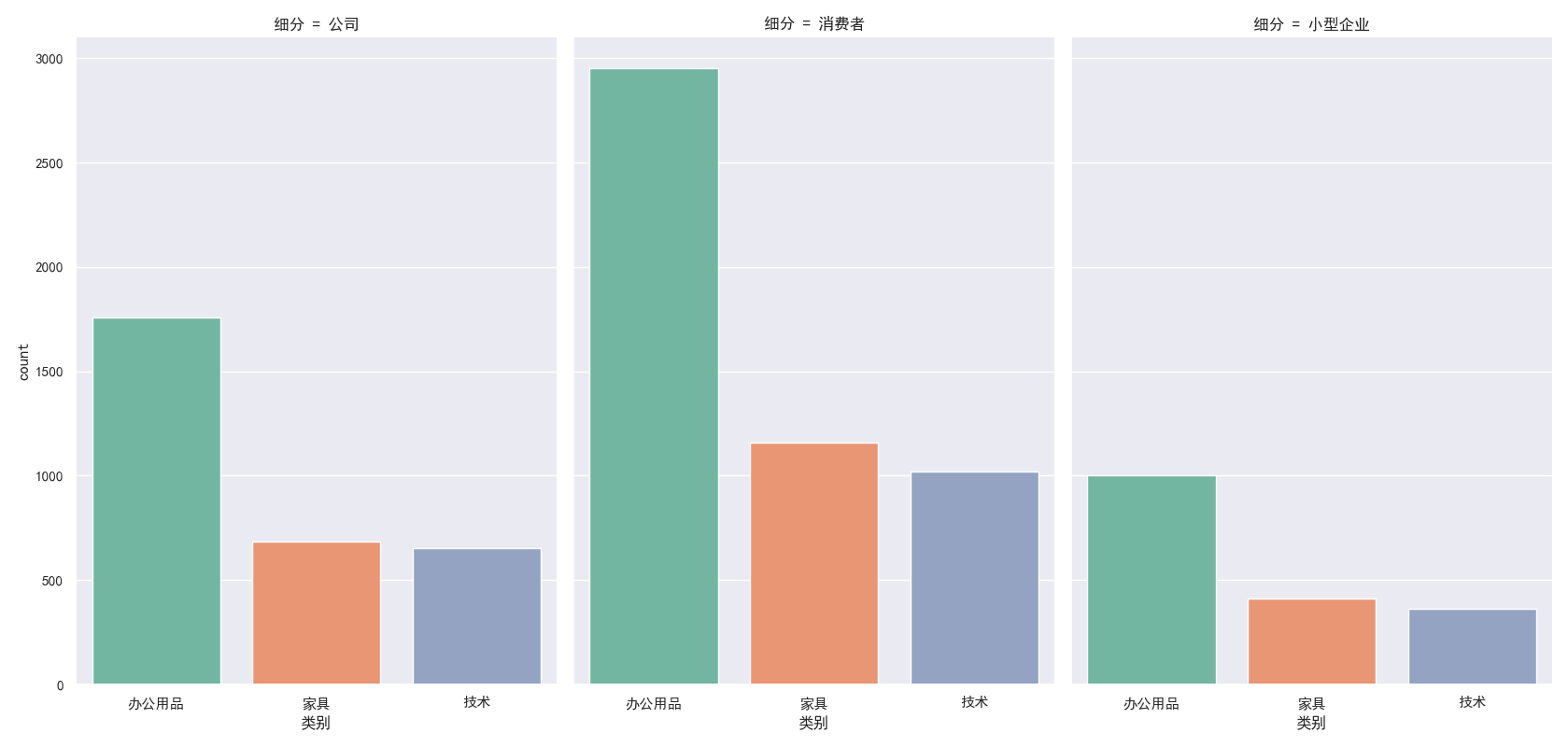
散点图：



小提琴图：



销售对象和物品类别相关性（直方图表示）：



所有数值变量之间的散点图：



第三题：成都市青羊区锦江区是公认的小学初中教育资源最好的行政区，因此二手房价相对较高。课程作业文件夹中包含**青羊区二手房.csv** 格式数据文档文件一份。数据为青羊区挂牌二手房的相关数据。要求：整理与描述青羊区挂牌二手房的相关信息，建立二手房房价的预测模型，揭示影响二手房房价的主要因素并对结果进行管理学意义上的阐述（小组为单位完成；合计20分）；

Get\_data.py 获取csv数据：

# 数据读取

qyq\_house\_data = pd.read\_csv("./青羊区二手房.csv",

names=["title", "room", "area", "height", "dic", "year", "addr", "cost", "price"],

encoding='gbk')

# 删去空行

qyq\_house\_data.dropna(axis=0, how='any', inplace=True)

# 去除换行符和制表符

qyq\_house\_data.title = qyq\_house\_data.title.apply(lambda x: x.replace("\t", "").replace("\n", ""))

qyq\_house\_data.head()

qyq\_house\_data.room = qyq\_house\_data.room.apply(lambda x: [x, "d室d厅"][x == "独栋"])

qyq\_house\_data.room = qyq\_house\_data.room.apply(lambda x: [x, "l室l厅"][x == "联排"])

qyq\_house\_data.room = qyq\_house\_data.room.apply(lambda x: [x, "s室s厅"][x == "双拼"])

qyq\_house\_data.room = qyq\_house\_data.room.apply(lambda x: [x, "j室j厅"][x == "叠加"])

print(qyq\_house\_data.room.apply(lambda x: x))

qyq\_house\_data["room"] = qyq\_house\_data.room.apply(lambda x: x[0])

qyq\_house\_data["hall"] = qyq\_house\_data.room.apply(lambda x: x[2])

# 研究非别墅区,提取数字,处理定性变量

normal\_house = qyq\_house\_data.sort\_values(by="area", axis=0,

ascending=True, inplace=False,

kind='quicksort', na\_position='last'

).head(-103)

normal\_house["room"] = normal\_house.room.apply(lambda x: int(x[0]))

normal\_house["hall"] = normal\_house.room.apply(lambda x: int(x[2]))

normal\_house["area"] = normal\_house.area.apply(lambda x: x.replace("㎡", "")).astype("float")

normal\_house["layers"] = normal\_house.height.apply(lambda x: int(x.split("（共")[-1][:-2]))

normal\_house["height"] = normal\_house.height.apply(lambda x: x[0])

normal\_house["year"] = normal\_house.year.apply(lambda x: int(re.findall(r"\d+\.?\d\*", x)[0]))

normal\_house["addr\_1"] = normal\_house.addr.apply(lambda x: x.split("-")[0])

normal\_house["addr\_2"] = normal\_house.addr.apply(lambda x: x.split("-")[-1])

normal\_house["cost"] = normal\_house.cost.astype("float")

normal\_house["price"] = normal\_house.price.apply(lambda x: int(re.findall(r"\d+\.?\d\*", x)[0]))

normal\_house["price\_cor"] = normal\_house.cost / normal\_house.area \* 10000

# 采用四分点和中点来映射层高

height\_state = {'高': 0.75, '中': 0.5, '低': 0.25}

normal\_house["height\_state"] = normal\_house.height.map(height\_state) \* normal\_house.layers

# 独热编码

normal\_house = normal\_house.join([pd.get\_dummies(normal\_house.height), pd.get\_dummies(normal\_house.dic), pd.get\_dummies(normal\_house.addr\_1)])

normal\_house.head(1)

# 相关系数

normal\_house.corr().dropna(thresh=1).dropna(axis=1, thresh=1).sort\_values(by="price\_cor")

Build\_model.py 建立模型：

#数据集划分与标准化

from sklearn.preprocessing import StandardScaler, MinMaxScaler

from sklearn.metrics import \*

from sklearn.linear\_model import LinearRegression

from sklearn.svm import SVR

import random

scaler = StandardScaler()

mms = MinMaxScaler()

normal\_house = normal\_house.sort\_values(by="price\_cor").reset\_index(drop=True)

# 注意不能保留总售价，会出现完全共线性

x = scaler.fit\_transform(normal\_house.loc[:, ("area", "year", "shi", "ting", "layers", "height\_state", "中", "低", "高", "东北向",

"东南向", "东向", "东西向", "北向", "南北向", "南向", "西北向", "西南向", "西向", "万家湾", "光华", "八宝街",

"内光华", "内金沙", "外光华", "外金沙", "天府广场", "府南新区", "杜甫草堂", "浣花小区", "石人小区", "苏坡",

"草市街", "西南财大", "贝森", "长顺街", "顺城街", "骡马市")])

y = normal\_house.loc[:, "price\_cor"]

# 划分训练集与测试集；测试集选取全部数据的20%

train = []

test = []

for i in range(normal\_house.shape[0]):

if random.random() > 0.2:

train.append(i)

else:

test.append(i)

x\_test = x[test]

x\_train = x[train]

y\_test = y[test]

y\_train = y[train]

# 多元线性回归

linreg = LinearRegression()

lin\_fbs = linreg.fit(x\_train, y\_train)

y\_lin = lin\_fbs.predict(x\_test)

y\_hat\_2 = lin\_fbs.predict(x\_train)

fig = plt.figure(figsize=(16, 16), dpi=150)

plt.subplots\_adjust(wspace=0.1, hspace=0.3)

x\_label = np.arange(1, len(y\_test))

plt.subplot(211)

plt.plot(x\_label, y\_test[:-1], linestyle='--', linewidth=2, label='y\_test 实际')

plt.plot(x\_label, y\_lin[:-1], linestyle='--', linewidth=1, label='y\_test 预测')

plt.title('Prediction\n $Price$')

plt.legend()

plt.subplot(212)

plt.plot(x\_label, y\_test[:-1] - y\_lin[:-1], linestyle='--', linewidth=2, label='y\_test 实际')

plt.title('Prediction\n $e$')

plt.legend()

# 回测

R\_square\_3 = r2\_score(y\_train, y\_hat\_2)

EVS\_3 = explained\_variance\_score(y\_train, y\_hat\_2)

MSE\_3 = mean\_squared\_error(y\_train, y\_hat\_2)

# 预测

R\_square\_4 = r2\_score(y\_test, y\_lin)

EVS\_4 = explained\_variance\_score(y\_test, y\_lin)

MSE\_4 = mean\_squared\_error(y\_test, y\_lin)

title = ["area", "year", "shi", "ting", "layers", "height\_state", "中", "低", "高", "东北向", "东南向", "东向", "东西向", "北向",

"南北向", "南向", "西北向", "西南向", "西向", "万家湾", "光华", "八宝街", "内光华", "内金沙", "外光华", "外金沙", "天府广场", "府南新区", "杜甫草堂",

"浣花小区", "石人小区", "苏坡", "草市街", "西南财大", "贝森", "长顺街", "顺城街", "骡马市"]

coef = lin\_fbs.coef\_.tolist()

c = {

"title": title,

"coef": coef

}

pd.DataFrame(c).sort\_values(by="coef", ascending=False).reset\_index(drop=True)

# 支持向量机

svr\_rbf = SVR(kernel='rbf', C=100)

svr\_rbf.fit(x\_train, y\_train)

y\_svr = svr\_rbf.predict(x\_test)

y\_hat\_1 = svr\_rbf.predict(x\_train)

fig = plt.figure(figsize=(16, 16), dpi=150)

plt.subplots\_adjust(wspace=0.1, hspace=0.3)

x\_label = np.arange(1, len(y\_test))

plt.subplot(211)

plt.plot(x\_label, y\_test[:-1], linestyle='--', linewidth=2, label='y\_test 实际')

plt.plot(x\_label, y\_svr[:-1], linestyle='--', linewidth=1, label='y\_test 预测')

plt.title('Prediction\n $Price$')

plt.legend()

plt.subplot(212)

plt.plot(x\_label, y\_test[:-1] - y\_svr[:-1], linestyle='--', linewidth=2, label='y\_test 实际')

plt.title('Prediction\n $e$')

plt.legend()

# 回测

R\_square\_1 = r2\_score(y\_train, y\_hat\_1)

EVS\_1 = explained\_variance\_score(y\_train, y\_hat\_1)

MSE\_1 = mean\_squared\_error(y\_train, y\_hat\_1)

# 预测

R\_square\_2 = r2\_score(y\_test, y\_svr)

EVS\_2 = explained\_variance\_score(y\_test, y\_svr)

MSE\_2 = mean\_squared\_error(y\_test, y\_svr)

第四题：基于纽约地铁站人流量数据（**Newyork.csv**），使用FineBI或Python进行数据的可视化描述天气状况如何影响地铁客流?这种影响效应是否存在差异性（在不同的条件下影响效应具有差异）？输出图表结果并对结果进行较为详细的阐述（小组为单位完成；合计20分）

**第五题：51job.csv**存放了51job 网站成都地区的关于数据分析相关的职位的信息，试使用python的numpy 和 pandas 包对数据进行整理清洗与分析，回答如下问题：（个人完成；合计 20 分）

find\_the\_most\_nead\_job.py 找到对数据分析需求最高的岗位：

import re

import pandas as pd

def find\_the\_most\_nead\_job ():

'''哪个区对数据分析相关的岗位需求最高'''

data = pd.read\_excel('51job.xlsx', sheet\_name = '51job', header = None)

selected\_data = data[data[1].str.contains('数据分析')].copy() #筛选数据分析相关岗位

need\_person\_list = [] #需求人数

for row in selected\_data.itertuples():

strlist = row[8].split('-')

if len(strlist) == 1:

strlist.append(strlist[0])

zui\_shao\_ren\_shu = int(re.sub("[^0-9\.]", "", strlist[0]))

zui\_duo\_ren\_shu = int(re.sub("[^0-9\.]", "", strlist[1]))

need\_person = (zui\_shao\_ren\_shu + zui\_duo\_ren\_shu) // 2

need\_person\_list.append(need\_person)

selected\_data[10] = need\_person\_list

need\_person = {}

for row in selected\_data.itertuples():

if row[4] in need\_person:

need\_person[row[4]] += row[11]

else:

need\_person[row[4]] = row[11]

sorted\_QuYuXuQiu = sorted(need\_person.items(),key=lambda d: d[1],reverse=True)

print(sorted\_QuYuXuQiu[0])

（1）从数据中，我们可以看到成都对数据分析岗位的需求最大。