**广 州 商 学 院**

**实验报告册**

信息技术与工程 学院 软件工程 专业2003年级3班

2022 - 2022 学年第 2 学期

课程名称：数据分析与处理实验

姓 名： 王弦

学 号： 202006120123

**目 录**

实验一 ………………………………………………………………... 1

实验二 ………………………………………………………………... 8

实验三 …………………………………………………………………13

实验四 ………………………………………………………………..27

**广 州 商 学 院**

**实验报告（第 1 次）**

实验名称 客户价值分析 实验时间 2022/3/23

同组同学 小组分工 独自完成

**一、实验目的**

（1）熟练掌握Python常用数据结构

（2）熟练掌握数据分析相关库的运用

（3）熟练应用RFM模型，进行聚类分析

（4）熟练应用K-means聚类算法

（5）能用Python实现客户价值分析

**二、实验仪器设备或材料**

1.学生PC机 57 台

2.教师机 1 台

3.教师机安装有极域教师机端、学生PC机安装有极域学生端

4.教学机和学生机需要安装第三方模块 ：pandas、numpy、matplotlib、sklearn。

**三、实验原理**

1、教学机与学生机需要安装PyCharm和Anaconda3、数据分析常用的第三方库。

2、学生需要已经学习Python程序设计语言，熟悉Python的基本语法和结构。

3、学生在自己的电脑上要安装第三方模块 ：pandas、numpy、matplotlib、sklearn。

**四、实验内容与步骤**

**1、项目文件结构**

****

**2、源代码**

**data\_clean.py**

import pandas as pd

import numpy as np

aa =r'./data/TB201812.xls'

resultfile=r'./data/data.xls'

df = pd.DataFrame(pd.read\_excel(aa))

df1=df[['订单付款时间','买家会员名','买家实际支付金额','数据采集时间']]

#去除空值，订单付款时间非空值才保留

#去除买家实际支付金额为0的记录

df1=df1[df1['订单付款时间'].notnull() & df1['买家实际支付金额'] !=0]

#R值：R表示客户最近活跃时间与数据采集点时间距离，时间相减变为天：

#R代表客户最近的活跃时间距离数据采集点的时间距离，

#R越大，表示客户越久未发生交易，R越小，表示客户越近有交易发生。

#R越大则客户越可能会“沉睡”，流失的可能性越大。在这部分客户中，可能有些优质客户，值得通过一定的营销手段进行激活。

#F值：F代表客户过去某段时间内的活跃频率。F越大，则表示客户同本公司的交易越频繁，是非常忠诚的客户；

#F越小，则表示客户不够活跃，且可能是竞争对手的常客。针对F较小、且消费额较大的客户，需要推出一定的竞争策略，将这批客户从竞争对手中争取过来。

#M值：M表示客户每次消费金额的多少，可以用最近一次消费金额，也可以用过去的平均消费金额，根据分析的目的不同，可以有不同的标识方法。

#一般来讲，单次交易金额较大的客户，支付能力强，价格敏感度低，是较为优质的客户，而每次交易金额很小的客户，可能在支付能力和支付意愿上相对较低。

#R:最近消费时间距数据采集时间的间隔

#利用to\_datetime转换为时间格式：'yyyy-MM-dd HH:mm:ss'

df1['R'] = (pd.to\_datetime(df1['数据采集时间']) - pd.to\_datetime(df1['订单付款时间'])).values/np.timedelta64(1, 'D')

df1=df1[['订单付款时间','买家会员名','买家实际支付金额','R']]

df2=df1.groupby('买家会员名').agg({'R': 'min','买家实际支付金额':'sum'})

#F:消费频次，客户一定时间内的购买次数。

df2['F']=df1.groupby(["买家会员名"])['买家会员名'].size()

df2.to\_excel(resultfile) #导出结果

**data\_transform.py**

import pandas as pd

#标准化处理

datafile = r'./data/data.xls' #需要进行标准化的数据文件；

transformfile = r'./data/transformdata.xls' #标准化后的数据存储路径文件；

data = pd.read\_excel(datafile)

data=data[['R','F','买家实际支付金额']]

data = (data - data.mean(axis = 0))/(data.std(axis = 0)) #简洁的语句实现了标准化变换，类似地可以实现任何想要的变换。

data.columns=['R','F','M'] #表头重命名。

data.to\_excel(transformfile, index = False) #数据写入

**data\_kmeans.py**

#-\*- coding: utf-8 -\*-

import pandas as pd

import numpy as np

from pandas import to\_datetime

#引入sklearn框架，导入K均值聚类算法

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

#from sklearn.manifold import

inputfile = r'./data/transformdata.xls' #待聚类的数据文件

outputfile=r'./data/data\_type.xls'

#读取数据并进行聚类分析

data = pd.read\_excel(inputfile) #读取数据

#利用K-Means聚类算法对客户数据进行客户分群，聚成4类

k = 4 #需要进行的聚类类别数

iteration=500

kmodel = KMeans(n\_clusters = k,max\_iter=iteration)

kmodel.fit(data) #训练模型

r1=pd.Series(kmodel.labels\_).value\_counts()

r2=pd.DataFrame(kmodel.cluster\_centers\_)

r=pd.concat([r2,r1],axis=1)

r.columns=list(data.columns)+[u'聚类数量']

r3 = pd.Series(kmodel.labels\_,index=data.index)

r = pd.concat([data,r3], axis=1)

r.columns = list(data.columns)+[u'聚类类别']

r.to\_excel(outputfile)

kmodel.cluster\_centers\_

kmodel.labels\_

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

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

for i in range(k):

cls=data[r[u'聚类类别']==i]

cls.plot(kind='kde',linewidth=2,subplots=True,sharex=False)

plt.suptitle('客户群=%d;聚类数量=%d' %(i,r1[i]))

plt.legend()

plt.show()

**data\_view.py**

#-\*- coding: utf-8 -\*-

#对数据进行基本的探索

#返回缺失值个数以及最大最小值

import pandas as pd

datafile= r'./data/TB201812.xls' #原始数据,第一行为属性标签

resultfile = r'./data/view.xls' #数据探索结果表

data = pd.read\_excel(datafile, encoding = 'utf-8') #读取原始数据，指定UTF-8编码（需要用文本编辑器将数据装换为UTF-8编码）

data=data[['订单付款时间','买家会员名','买家实际支付金额','数据采集时间']]

view = data.describe(percentiles = [], include = 'all').T #包括对数据的基本描述，percentiles参数是指定计算多少的分位数表（如1/4分位数、中位数等）；T是转置，转置后更方便查阅

view['null'] = len(data)-view['count'] #describe()函数自动计算非空值数，需要手动计算空值数

view = view[['null', 'max', 'min']]

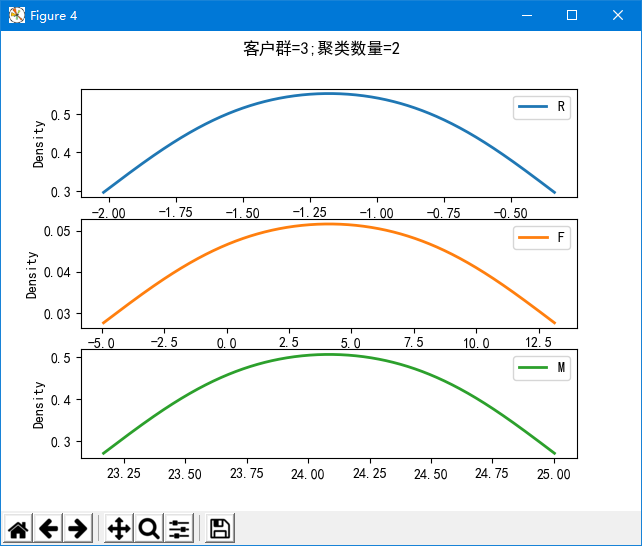
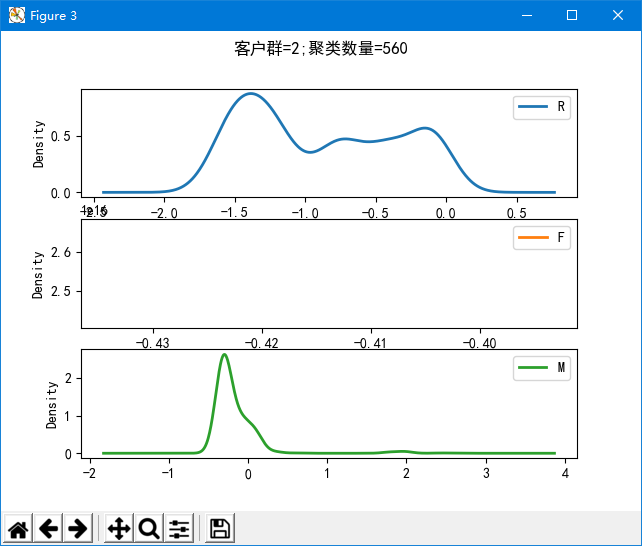
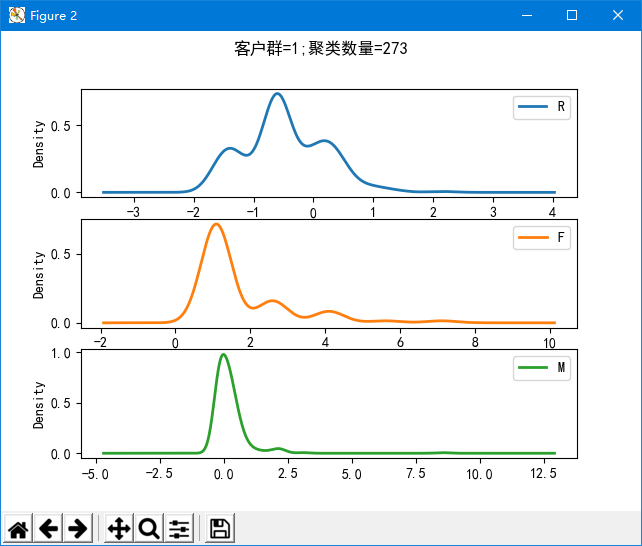
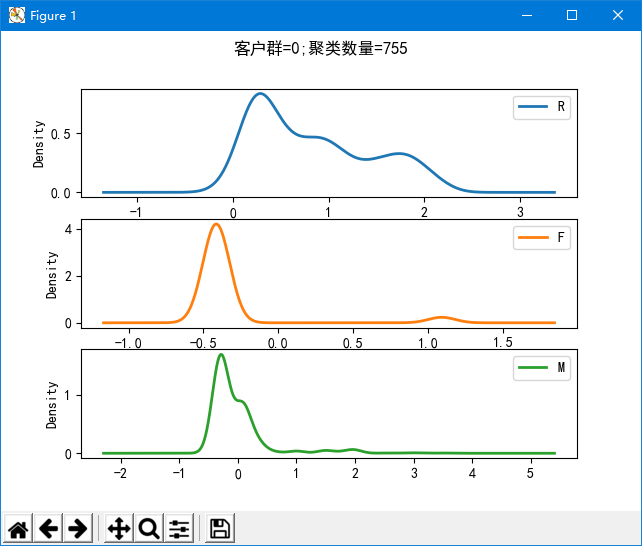
view.columns = [u'空值数', u'最大值', u'最小值'] #表头重命名

'''这里只选取部分探索结果。

describe()函数自动计算的字段有count（非空值数）、unique（唯一值数）、top（频数最高者）、freq（最高频数）、mean（平均值）、std（方差）、min（最小值）、50%（中位数）、max（最大值）'''

view.to\_excel(resultfile) #导出结果

**3、系统预览**



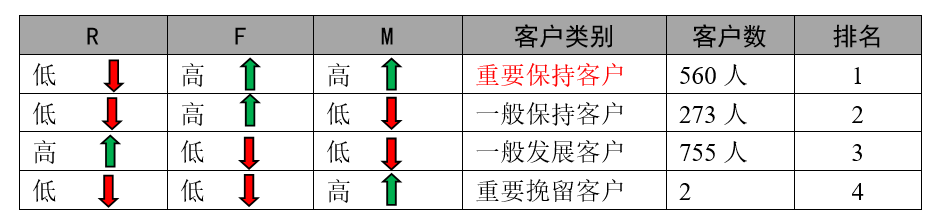
客户群0

客户群1

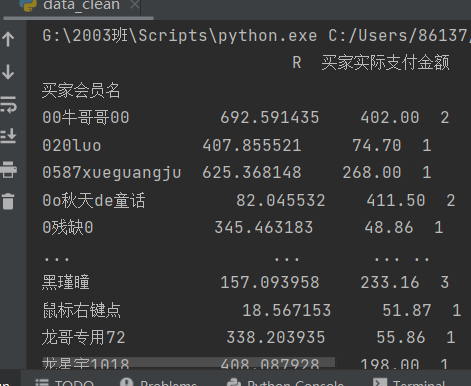
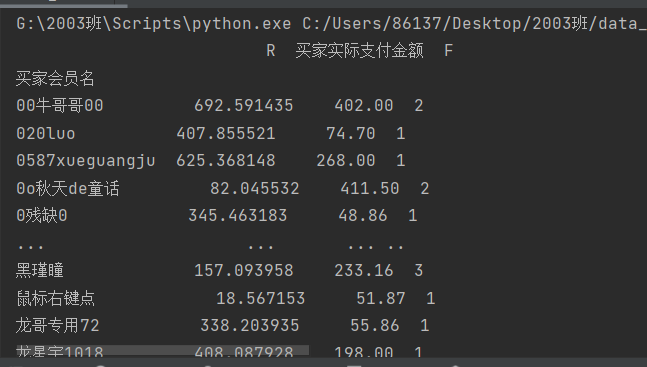
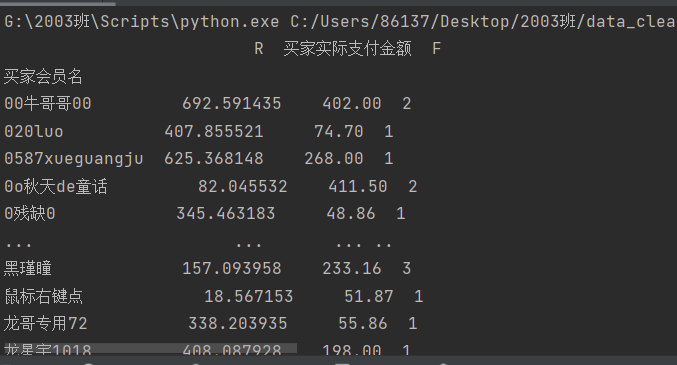
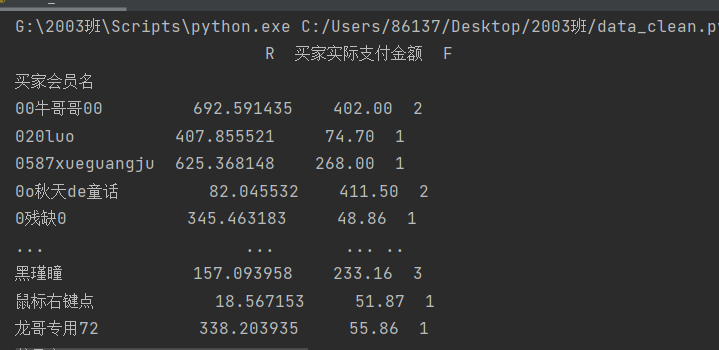
客户群2

客户群3

**4、客户价值分析结果解析**



**五、实验结果与分析**



**六、结论与体会**

结论与体会：本次实验学习了Python里数据分析库的运用和聚类分析，实验比较难理解，需要多练习。

**七、教师评语**

实验目的明确，设计合理，实验数据正确;可看出在实验中操作步骤和过程完整，同时将知识能够很好地运用将课堂知识运用。

**广 州 商 学 院**

**实验报告（第 2 次）**

实验名称 销售收入分析与预测 实验时间 2022/4/10

同组同学 小组分工 独自完成

1. **实验目的**

（1）明确做数据预处理的作用

（2）熟练掌握对数据重复值、缺失值、异常值的检测和处理方法

（3）能够对实际案例数据进行数据清洗工作

（4）熟练掌握线性回归与最小二乘法

1. **实验仪器设备或材料**

1、教学机与学生机需要安装第三方模块pandas、numpy、matplotlib、sklearn

2、教学机与学生机需要安装PyCharm和Anaconda3

1. **实验原理**

1.学生PC机 57 台

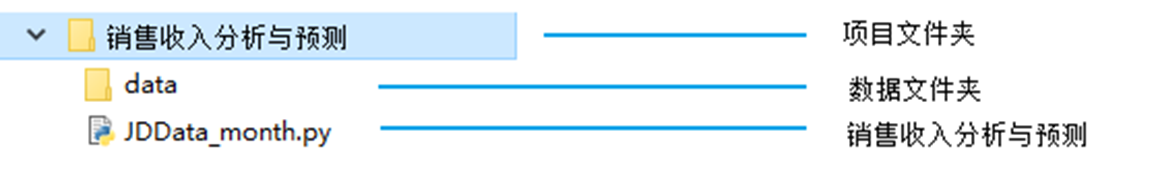
2.教师机 1 台

3.教师机安装有极域教师机端、学生PC机安装有极域学生端

4.教学机和学生机需要安装 PyCharm和Anaconda3。

**四、实验内容与步骤**

**1、项目文件结构**

****

**2、源代码**

**JDData\_month.py**

import pandas as pd

import numpy as np

from sklearn import linear\_model

import matplotlib.pyplot as plt

aa =r'.\data\JDdata.xls'

bb=r'.\data\JDcar.xls'

resultfile1=r'result1.xls'

resultfile2=r'result2.xls'

dfaa = pd.DataFrame(pd.read\_excel(aa))

dfbb=pd.DataFrame(pd.read\_excel(bb))

df1=dfaa[['业务日期','金额']]

df2=dfbb[['投放日期','支出']]

#去除空日期和金额为0的记录

df1=df1[df1['业务日期'].notnull() & df1['金额'] !=0]

df2=df2[df2['投放日期'].notnull() & df2['支出'] !=0]

df1['业务日期'] = pd.to\_datetime(df1['业务日期'])

df2['投放日期'] = pd.to\_datetime(df2['投放日期'])

dfData = df1.set\_index('业务日期',drop=True)

dfCar=df2.set\_index('投放日期',drop=True)

# 按月度统计并显示销售金额

dfData\_month=dfData.resample('M').sum().to\_period('M')

# 按月度统计并显示广告费支出金额

dfCar\_month=dfCar.resample('M').sum().to\_period('M')

dfData\_month.to\_excel(resultfile1) #导出结果

dfCar\_month.to\_excel(resultfile2) #导出结果

clf=linear\_model.LinearRegression(fit\_intercept=True,normalize=False)

#x为广告费用，y为销售收入

x=pd.DataFrame(dfCar\_month['支出'])

y=pd.DataFrame(dfData\_month['金额'])

# 图表字体为华文细黑，字号为10

plt.rc('font', family='SimHei', size=10)

plt.figure("销售收入分析")

plt.scatter(x, y, color='red') #真实值散点图

#plt.show()

#绘制拟合图

clf.fit(x,y) #拟合线性模型

k=clf.coef\_ #获取回归系数（斜率w1,w2,w3,...,wn）

b=clf.intercept\_ #获取截距w0

#7月预计投入60000元广告费（x0）

x0=60000

#预测7月销售收入（y0），y0=截距+X值\*斜率

y0=b+x0\*k

print(y0)

#使用线性模型进行预测y值

y\_pred =clf.predict(x)

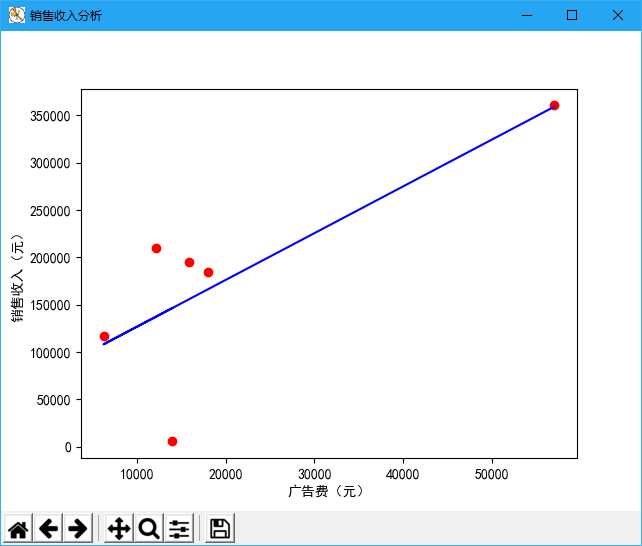
plt.plot(x,y\_pred, color='blue', linewidth=1.5) #预测回归线

plt.ylabel(u'销售收入（元）')

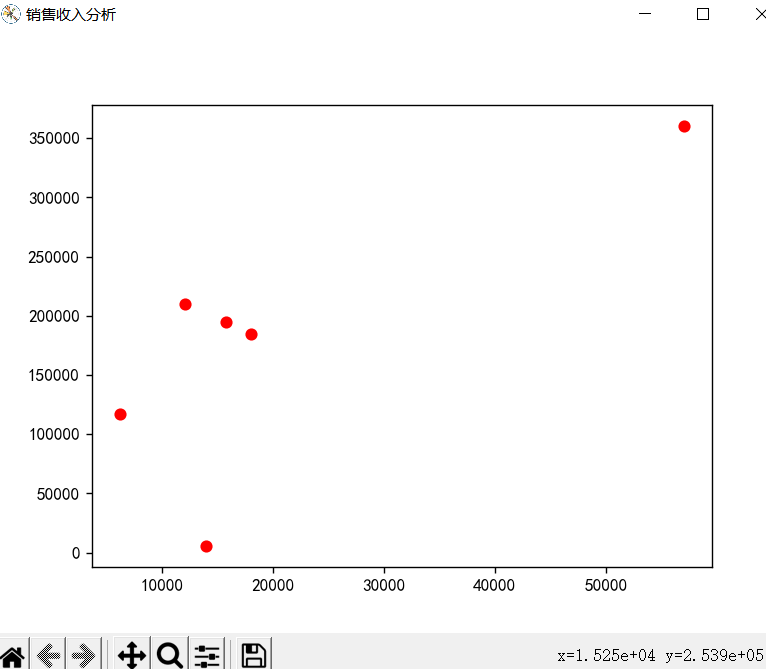
plt.xlabel(u'广告费（元）')

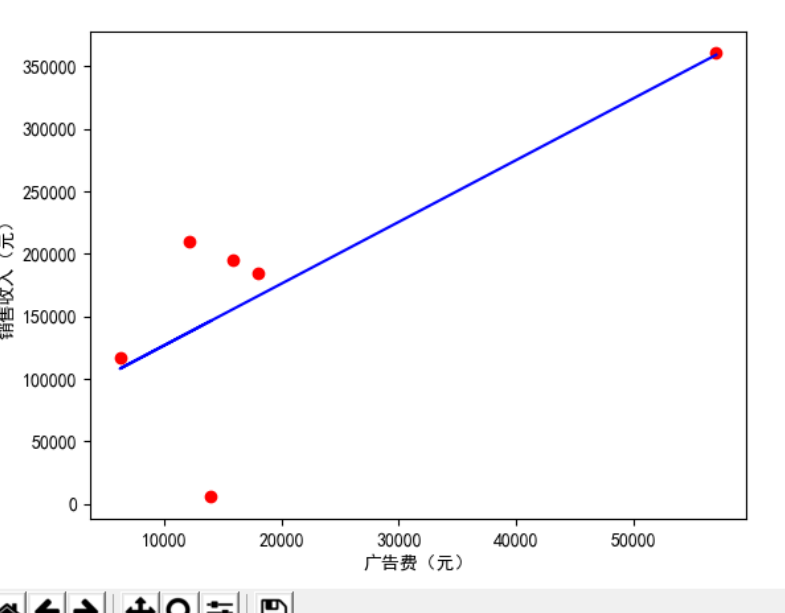
plt.show()

3、**系统预览**



1. **实验结果与分析**





**六、结论与体会**

结论与体会：本次实验学习了各种对数据的处理的方法，实验遇到了一些困难，通过上网查资料解决。

1. **教师评语**

实验中实验数据准确，记录完整，图形设计合理，表明在实验中善于发现问题，分析问题，并结合理论知识解释和解决问题。

**广 州 商 学 院**

**实验报告（第 3 次）**

实验名称 二手房数据分析预测系统 实验时间 2022.5.5

同组同学 小组分工 独自完成

**一、实验目的**

（1）掌握对简单图形和复杂图形的数据可视化方法

（2）熟练利用Python对数据进行可视化处理

（3）掌握使用PyQt5搭建应用窗体

**二、实验原理**

1、教学机与学生机需要安装PyCharm和Anaconda3

2、教学机与学生机需要安装第三方模块PyQt5、pyqt5-tools、matplotlib、sklearn、

pandas

**三、实验仪器材料**

1.学生PC机 57 台

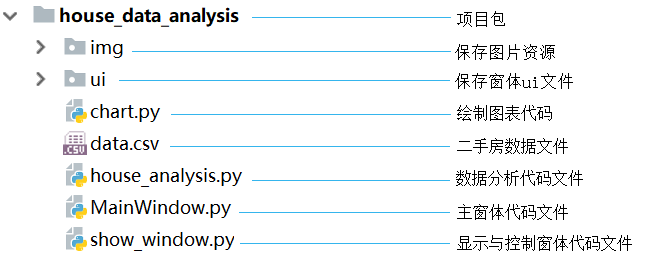
2.教师机 1 台

3.教师机安装有极域教师机端、学生PC机安装有极域学生端

4.教学机和学生机需要安装 PyCharm和Anaconda3

**四、实验内容**

**1、项目文件结构**

****

**2、源代码**

**MainWindow.py**

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_MainWindow(object):

def setupUi(self, MainWindow):

MainWindow.setObjectName("MainWindow")

MainWindow.resize(703, 390)

MainWindow.setMinimumSize(QtCore.QSize(695, 390))

MainWindow.setMaximumSize(QtCore.QSize(710, 695))

MainWindow.setContextMenuPolicy(QtCore.Qt.CustomContextMenu)

MainWindow.setToolButtonStyle(QtCore.Qt.ToolButtonTextUnderIcon)

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

self.label = QtWidgets.QLabel(self.centralwidget)

self.label.setGeometry(QtCore.QRect(0, 0, 711, 321))

self.label.setText("")

self.label.setPixmap(QtGui.QPixmap("../img/背景图.png"))

self.label.setScaledContents(True)

self.label.setObjectName("label")

MainWindow.setCentralWidget(self.centralwidget)

self.toolBar = QtWidgets.QToolBar(MainWindow)

self.toolBar.setAcceptDrops(True)

self.toolBar.setIconSize(QtCore.QSize(48, 48))

self.toolBar.setObjectName("toolBar")

MainWindow.addToolBar(QtCore.Qt.TopToolBarArea, self.toolBar)

self.btn\_1 = QtWidgets.QAction(MainWindow)

icon = QtGui.QIcon()

icon.addPixmap(QtGui.QPixmap("../img/图标-1.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.btn\_1.setIcon(icon)

font = QtGui.QFont()

font.setPointSize(6)

self.btn\_1.setFont(font)

self.btn\_1.setObjectName("btn\_1")

self.btn\_2 = QtWidgets.QAction(MainWindow)

icon1 = QtGui.QIcon()

icon1.addPixmap(QtGui.QPixmap("../img/图标-2.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.btn\_2.setIcon(icon1)

font = QtGui.QFont()

font.setPointSize(6)

self.btn\_2.setFont(font)

self.btn\_2.setObjectName("btn\_2")

self.btn\_3 = QtWidgets.QAction(MainWindow)

icon2 = QtGui.QIcon()

icon2.addPixmap(QtGui.QPixmap("../img/图标-3.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.btn\_3.setIcon(icon2)

font = QtGui.QFont()

font.setPointSize(6)

self.btn\_3.setFont(font)

self.btn\_3.setObjectName("btn\_3")

self.btn\_4 = QtWidgets.QAction(MainWindow)

icon3 = QtGui.QIcon()

icon3.addPixmap(QtGui.QPixmap("../img/图标-4.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.btn\_4.setIcon(icon3)

font = QtGui.QFont()

font.setPointSize(6)

self.btn\_4.setFont(font)

self.btn\_4.setObjectName("btn\_4")

self.btn\_5 = QtWidgets.QAction(MainWindow)

icon4 = QtGui.QIcon()

icon4.addPixmap(QtGui.QPixmap("../img/图标-5.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.btn\_5.setIcon(icon4)

font = QtGui.QFont()

font.setPointSize(1)

self.btn\_5.setFont(font)

self.btn\_5.setObjectName("btn\_5")

self.toolBar.addSeparator()

self.toolBar.addAction(self.btn\_1)

self.toolBar.addSeparator()

self.toolBar.addAction(self.btn\_2)

self.toolBar.addSeparator()

self.toolBar.addAction(self.btn\_3)

self.toolBar.addSeparator()

self.toolBar.addAction(self.btn\_4)

self.toolBar.addSeparator()

self.toolBar.addAction(self.btn\_5)

self.toolBar.addSeparator()

self.retranslateUi(MainWindow)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):

\_translate = QtCore.QCoreApplication.translate

MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))

self.toolBar.setWindowTitle(\_translate("MainWindow", "toolBar"))

self.btn\_1.setText(\_translate("MainWindow", "各区二手房均价分析"))

self.btn\_1.setToolTip(\_translate("MainWindow", "各区二手房均价分析"))

self.btn\_2.setText(\_translate("MainWindow", "各区二手房数量所占比例"))

self.btn\_2.setToolTip(\_translate("MainWindow", "各区二手房数量所占比例"))

self.btn\_3.setText(\_translate("MainWindow", "全市二手房装修程度分析"))

self.btn\_3.setToolTip(\_translate("MainWindow", "全市二手房装修程度分析"))

self.btn\_4.setText(\_translate("MainWindow", "热门户型均价分析"))

self.btn\_4.setToolTip(\_translate("MainWindow", "热门户型均价分析"))

self.btn\_5.setText(\_translate("MainWindow", "二手房售价预测"))

self.btn\_5.setToolTip(\_translate("MainWindow", "二手房售价预测"))

**house\_analysis.py**

import pandas # 导入数据统计模块

from sklearn.svm import LinearSVR # 导入回归函数

data = pandas.read\_csv('data.csv') # 读取csv数据文件

del data['Unnamed: 0'] # 将索引列删除

data.dropna(axis=0, how='any', inplace=True) # 删除data数据中的所有空值

data['单价'] = data['单价'].map(lambda d: d.replace('元/平米', '')) # 将单价“元/平米”去掉

data['单价'] = data['单价'].astype(float) # 将房子单价转换为浮点类型

data['总价'] = data['总价'].map(lambda z: z.replace('万', '')) # 将总价“万”去掉

data['总价'] = data['总价'].astype(float) # 将房子总价转换为浮点类型

data['建筑面积'] = data['建筑面积'].map(lambda p: p.replace('平米', '')) # 将建筑面价“平米”去掉

data['建筑面积'] = data['建筑面积'].astype(float) # 将建筑面积转换为浮点类型

# 获取各区二手房均价分析

def get\_average\_price():

group = data.groupby('区域') # 将房子区域分组

average\_price\_group = group['单价'].mean() # 计算每个区域的均价

region = average\_price\_group.index # 区域

average\_price = average\_price\_group.values.astype(int) # 区域对应的均价

return region, average\_price # 返回区域与对应的均价

# 获取各区房子数量比例

def get\_house\_number():

group\_number = data.groupby('区域').size() # 房子区域分组数量

region = group\_number.index # 区域

numbers = group\_number.values # 获取每个区域内房子出售的数量

percentage = numbers / numbers.sum() \* 100 # 计算每个区域房子数量的百分比

return region, percentage # 返回百分比

# 获取全市二手房装修程度对比

def get\_renovation():

group\_renovation = data.groupby('装修').size() # 将房子装修程度分组并统计数量

type = group\_renovation.index # 装修程度

number = group\_renovation.values # 装修程度对应的数量

return type, number # 返回装修程度与对应的数量

# 获取二手房热门户型均价

def get\_house\_type():

house\_type\_number = data.groupby('户型').size() # 房子户型分组数量

sort\_values = house\_type\_number.sort\_values(ascending=False) # 将户型分组数量进行降序

top\_five = sort\_values.head(5) # 提取前5组户型数据

house\_type\_mean = data.groupby('户型')['单价'].mean() # 计算每个户型的均价

type = house\_type\_mean[top\_five.index].index # 户型

price = house\_type\_mean[top\_five.index].values # 户型对应的均价

return type, price.astype(int) # 返回户型与对应的数量

# 获取价格预测

def get\_price\_forecast():

data\_copy = data.copy() # 拷贝数据

print(data\_copy[['户型', '建筑面积']].head())

data\_copy[['室', '厅', '卫']] = data\_copy['户型'].str.extract('(\d+)室(\d+)厅(\d+)卫')

data\_copy['室'] = data\_copy['室'].astype(float) # 将房子室转换为浮点类型

data\_copy['厅'] = data\_copy['厅'].astype(float) # 将房子厅转换为浮点类型

data\_copy['卫'] = data\_copy['卫'].astype(float) # 将房子卫转换为浮点类型

print(data\_copy[['室','厅','卫']].head()) # 打印“室”、“厅”、“卫”数据

del data\_copy['小区名字']

del data\_copy['户型']

del data\_copy['朝向']

del data\_copy['楼层']

del data\_copy['装修']

del data\_copy['区域']

del data\_copy['单价']

data\_copy.dropna(axis=0, how='any', inplace=True) # 删除data数据中的所有空值

# 获取“建筑面积”小于300平米的房子信息

new\_data = data\_copy[data\_copy['建筑面积'] < 300].reset\_index(drop=True)

print(new\_data.head()) # 打印处理后的头部信息

# 添加自定义预测数据

new\_data.loc[2505] = [None, 88.0, 2.0, 1.0, 1.0]

new\_data.loc[2506] = [None, 136.0, 3.0, 2.0, 2.0]

data\_train=new\_data.loc[0:2504]

x\_list = ['建筑面积', '室', '厅', '卫'] # 自变量参考列

data\_mean = data\_train.mean() # 获取平均值

data\_std = data\_train.std() # 获取标准偏差

data\_train = (data\_train - data\_mean) / data\_std # 数据标准化

x\_train = data\_train[x\_list].values # 特征数据

y\_train = data\_train['总价'].values # 目标数据，总价

linearsvr = LinearSVR(C=0.1) # 创建LinearSVR()对象

linearsvr.fit(x\_train, y\_train) # 训练模型

x = ((new\_data[x\_list] - data\_mean[x\_list]) / data\_std[x\_list]).values # 标准化特征数据

new\_data[u'y\_pred'] = linearsvr.predict(x) \* data\_std['总价'] + data\_mean['总价'] # 添加预测房价的信息列

print('真实值与预测值分别为：\n', new\_data[['总价', 'y\_pred']])

y = new\_data[['总价']][2490:] # 获取2490以后的真实总价

y\_pred = new\_data[['y\_pred']][2490:] # 获取2490以后的预测总价

return y,y\_pred # 返回真实房价与预测房价

**chart.py**

import matplotlib # 导入图表模块

import matplotlib.pyplot as plt # 导入绘图模块

# 避免中文乱码

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

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

# 显示饼图

def pie\_chart(size,label,title):

"""

绘制饼图

size:各部分大小

labels:设置各部分标签

labeldistance:设置标签文本距圆心位置，1.1表示1.1倍半径

autopct：设置圆里面文本

shadow：设置是否有阴影

startangle：起始角度，默认从0开始逆时针转

pctdistance：设置圆内文本距圆心距离

"""

plt.figure() # 图形画布

plt.pie(size, labels=label,labeldistance=1.05,

autopct="%1.1f%%", shadow=True, startangle=0, pctdistance=0.6)

plt.axis("equal") # 设置横轴和纵轴大小相等，这样饼才是圆的

plt.title(title, fontsize=12)

plt.legend(bbox\_to\_anchor=(0.03, 1)) # 让图例生效，并设置图例显示位置

plt.show() # 显示饼图

# 显示预测房价折线图

def broken\_line(y,y\_pred,title):

'''

y:y轴折线点，也就是房子总价

y\_pred,预测房价的折线点

color：折线的颜色

marker：折点的形状

'''

plt.figure() # 图形画布

plt.plot(y, color='r', marker='o',label='真实房价') # 绘制折线，并在折点添加蓝色圆点

plt.plot(y\_pred, color='b', marker='\*',label='预测房价')

plt.xlabel('房子数量')

plt.ylabel('房子总价')

plt.title(title) # 表标题文字

plt.legend() # 显示图例

plt.grid() # 显示网格

plt.show() # 显示图表

# 显示均价条形图

def average\_price\_bar(x,y, title):

plt.figure() # 图形画布

plt.bar(x,y, alpha=0.8) # 绘制条形图

plt.xlabel("区域") # 区域文字

plt.ylabel("均价") # 均价文字

plt.title(title) # 表标题文字

# 为每一个图形加数值标签

for x, y in enumerate(y):

plt.text(x, y + 100, y, ha='center')

plt.show() # 显示图表

# 显示装修条形图

def renovation\_bar(x,y, title):

plt.figure() # 图形画布

plt.bar(x,y, alpha=0.8) # 绘制条形图

plt.xlabel("装修类型") # 区域文字

plt.ylabel("数量") # 均价文字

plt.title(title) # 表标题文字

# 为每一个图形加数值标签

for x, y in enumerate(y):

plt.text(x, y + 10, y, ha='center')

plt.show() # 显示图表

# 显示热门户型的水平条形图

def bar(price,type, title):

"""

绘制水平条形图方法barh

参数一：y轴

参数二：x轴

"""

plt.figure() # 图形画布

plt.barh(type, price, height=0.3, color='r', alpha=0.8) # 从下往上画水平条形图

plt.xlim(0, 15000) # X轴的均价0~15000

plt.xlabel("均价") # 均价文字

plt.title(title) # 表标题文字

# 为每一个图形加数值标签

for y, x in enumerate(price):

plt.text(x + 10, y,str(x) + '元', va='center')

plt.show() # 显示图表

**show\_window.py**

from PyQt5.QtWidgets import QMainWindow, QApplication

from img.MainWindow import Ui\_MainWindow # 导入主窗体文件中的ui类

import sys # 导入系统模块

import house\_analysis # 导入自定义房子数据分析模块

import chart # 导入自定义绘图模块

# 主窗体初始化类

class Main(QMainWindow, Ui\_MainWindow):

def \_\_init\_\_(self):

super(Main, self).\_\_init\_\_()

self.setupUi(self)

# 显示各区二手房均价分析图

def show\_average\_price(self):

region, average\_price= house\_analysis.get\_average\_price() # 获取房子区域与均价

chart.average\_price\_bar(region,average\_price,'各区二手房均价分析')

# 显示各区二手房数量所占比例

def show\_house\_number(self):

region, percentage = house\_analysis.get\_house\_number() # 获取房子区域与数量百分比

chart.pie\_chart(percentage,region,'各区二手房数量所占比例') # 显示图表

# 显示全市二手房装修程度分析

def show\_renovation(self):

type, number = house\_analysis.get\_renovation() # 获取全市房子装修程度

chart.renovation\_bar(type,number,'全市二手房装修程度分析') # 显示图表

# 显示热门户型均价分析图

def show\_type(self):

type, price = house\_analysis.get\_house\_type() # 获取全市二手房热门户型均价

chart.bar(price,type,'热门户型均价分析')

# 显示二手房售价预测折线图

def show\_total\_price(self):

true\_price,forecast\_price = house\_analysis.get\_price\_forecast() # 获取预测房价

chart.broken\_line(true\_price,forecast\_price,'二手房售价预测') # 绘制及显示图表

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication(sys.argv)

# 主窗体对象

main = Main()

# 显示各区二手房均价分析图，按钮事件

main.btn\_1.triggered.connect(main.show\_average\_price)

# 显示各区二手房数量所占比例图，按钮事件

main.btn\_2.triggered.connect(main.show\_house\_number)

# 显示全市二手房装修程度分析图，按钮事件

main.btn\_3.triggered.connect(main.show\_renovation)

# 显示热门户型均价分析图，按钮事件

main.btn\_4.triggered.connect(main.show\_type)

# 显示全市二手房户售价预测图，按钮事件

main.btn\_5.triggered.connect(main.show\_total\_price)

# 显示主窗体

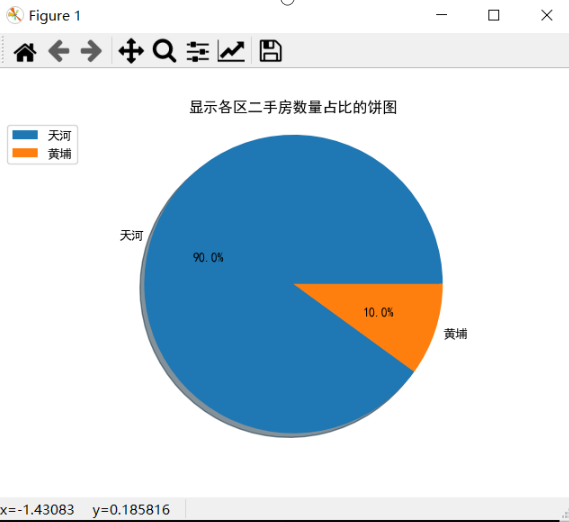
main.show()

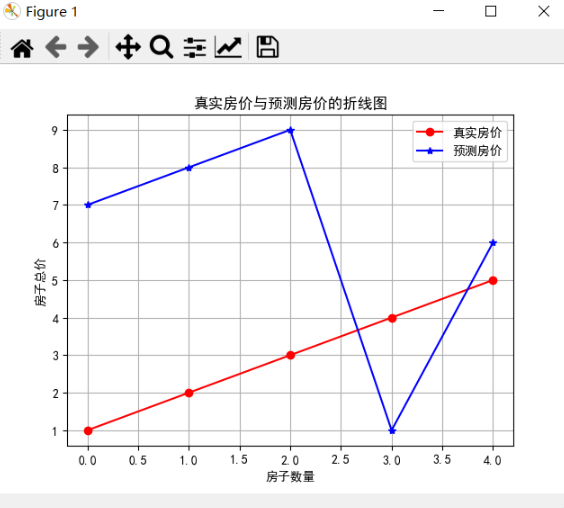
sys.exit(app.exec\_()) # 当窗口创建完成，需要结束主循环过程

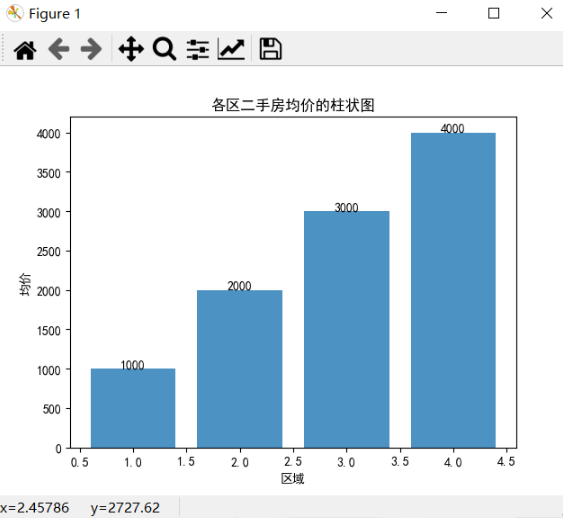
3、**系统预览**

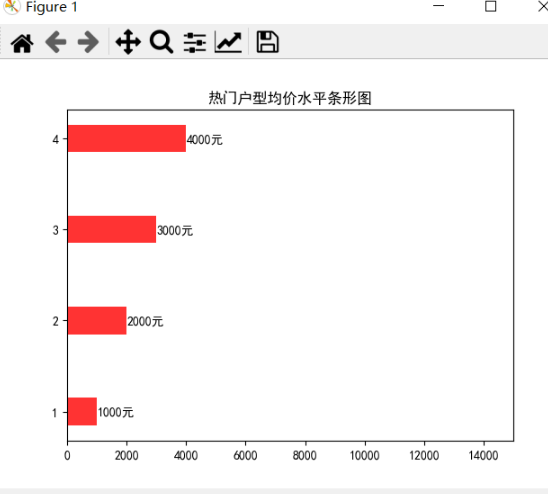


**五、实验结果与分析**









**六、结论与体会**

学习了对简单图形和复杂图形的数据可视化方法。

**七、教师评语**

实验表明该生具有一定的研究、分析和解决问题的能力，能够及时完成任务;并且一定的独特见解，实验数据和结果正确，完成质量好。

**广 州 商 学 院**

**实验报告（第 4 次）**

实验名称 智能停车场运营分析系统 实验时间 2022.05.25

同组同学 小组分工 独自完成

**一、实验目的**

（1）掌握数据分析方法

（2）掌握项目数据分析流程

（3）掌握使用Pygame搭建应用窗体

**二、实验仪器设备或材料**

1.学生PC机 57 台

2.教师机 1 台

3.教师机安装有极域教师机端、学生PC机安装有极域学生端

4.教学机和学生机需要安装 PyCharm和Anaconda3

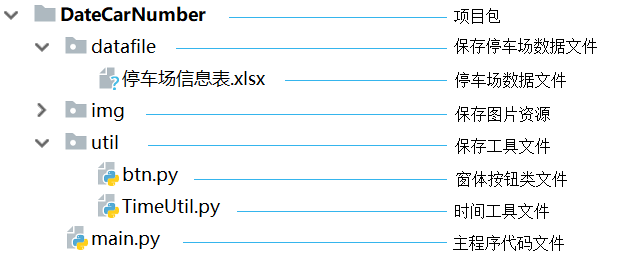
**三、实验原理**

1、教学机与学生机需要安装PyCharm和Anaconda3

2、教学机与学生机需要安装第三方模块pygame、matplotlib、pandas

**四、实验内容与步骤**

**1、项目文件结构**

****

1. **源代码**

**main.py**

import pygame # 导入pygame模块

from util import btn # 导入自定义的按钮

import matplotlib.pyplot as plt # 导入绘制图表的模块

from matplotlib.ticker import FuncFormatter # 绘图模块格式化类

from util.TimeUtil import \* # 导入自定义的时间处理模块

import pandas as pd # 导入pandas模块

plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签

plt.rcParams['axes.unicode\_minus']=False #用来正常显示负号

# 窗体大小

size = 340, 484

# 定义颜色

WHITE = (255, 255, 255)

BLUE = (72, 61, 139)

excelFile = r'datafile/停车场信息表.xlsx'

# 读取文件内容

pi\_table = pd.read\_excel(excelFile, sheet\_name='data')

# pygame初始化

pygame.init()

# 设置窗体名称

pygame.display.set\_caption('智能停车场运营分析系统')

# 图标

ic\_launcher = pygame.image.load('img/ic\_launcher.png')

# 设置图

pygame.display.set\_icon(ic\_launcher)

# 设置窗体大小

screen=pygame.display.set\_mode(size)

# 设置背景颜色

screen.fill(WHITE)

# 车位每天利用率

def lyl():

# 获取列表中state(车辆状体)列1为出停车场

tcdf = pi\_table.loc[pi\_table['state'] == 1]

# 循环的开始与结束时间

start = '2018-01-01'

end = '2018-03-31'

# 转换开始与结束时间类型

datestart = datetime.datetime.strptime(start, '%Y-%m-%d')

dateend = datetime.datetime.strptime(end, '%Y-%m-%d')

VALUE = [] # 数据列表

DATE = [] # 日期列表

while datestart <= dateend:

# 判断当前天 出车库的车辆多少

kk = tcdf[tcdf['timeout'].str.contains(datestart.strftime('%Y-%m-%d'))]

DATE.append(datestart.strftime('%Y-%m-%d')) # 将日期添加至列表中

yh =100- kk['rps'].mean() # 计算每天车位使用率

VALUE.append(yh) # 添加至数据列表中

# 按照天循环日期

datestart += datetime.timedelta(days=1)

# 绘制折线图 填充数据

plt.plot(DATE, VALUE)

# yticks格式化方法

def to\_percent(temp, position):

return '%1.0f' % (temp) + '%'

# 格式化yticks，以百分比的方式显示

plt.gca().yaxis.set\_major\_formatter(FuncFormatter(to\_percent))

plt.xticks([]) # 隐藏x轴刻度

plt.xlabel('2018-01-01至2018-03-31') # 显示日期范围

plt.title('车位利用率') # 设置标题

plt.show() # 显示图表

pass

# 周繁忙统计

def fmtj():

# 获取列表中rps（车位剩余）列为0的所有数据

fmdfs = pi\_table.loc[pi\_table['state'] ==1]

# 转换数据为列表

fmdfs=fmdfs.values

# x轴数据

WEEK = ['周一','周二','周三','周四','周五','周六','周日']

WEEK1 = 0 # 星期一

WEEK2 = 0 # 星期二

WEEK3 = 0 # 星期三

WEEK4 = 0 # 星期四

WEEK5 = 0 # 星期五

WEEK6 = 0 # 星期六

WEEK7 = 0 # 星期日

# 循环数据列表

for fmdf in fmdfs:

# 判断数据是星期几

week\_numbeer= get\_week\_numbeer(fmdf[1])

# 星期一

if week\_numbeer==0:

WEEK1 = WEEK1+1

pass

# 星期二

if week\_numbeer==1:

WEEK2 = WEEK2+1

pass

# 星期三

if week\_numbeer==2:

WEEK3 = WEEK3+1

pass

# 星期四

if week\_numbeer==3:

WEEK4 = WEEK4+1

pass

# 星期五

if week\_numbeer==4:

WEEK5 = WEEK5+1

pass

# 星期六

if week\_numbeer==5:

WEEK6 = WEEK6+1

pass

# 星期日

if week\_numbeer==6:

WEEK7 = WEEK7+1

pass

pass

# 数据信息

WEEK\_VAULE=[WEEK1,WEEK2,WEEK3,WEEK4,WEEK5,WEEK6,WEEK7]

plt.title("周繁忙统计") # 设置标题

plt.pie(WEEK\_VAULE, labels=WEEK, autopct='%1.1f%%') # 绘制饼图

plt.axis('equal') # 该行代码使饼图长宽相等

# 显示图例

plt.legend(loc="upper right", fontsize=10, bbox\_to\_anchor=(1.1, 1.05), borderaxespad=0.3)

plt.show() # 显示图表

# 每日接待车辆统计

def cljd():

# 获取列表中state(车辆状体)列1为出停车场

tcdf = pi\_table.loc[pi\_table['state'] == 1]

# 循环的开始与结束时间

start = '2018-01-01'

end = '2018-03-31'

# 转换开始与结束时间类型

datestart = datetime.datetime.strptime(start, '%Y-%m-%d')

dateend = datetime.datetime.strptime(end, '%Y-%m-%d')

VALUE=[] # 数据列表

DATE=[] # 日期列表

# 循环日期

while datestart <= dateend:

# 判断当前天 出车库的车辆多少

kk = tcdf[tcdf['timeout'].str.contains(datestart.strftime('%Y-%m-%d'))]

# 设置x轴数据按照天划分

DATE.append(datestart.strftime('%Y-%m-%d'))

# 日期对应的出车库车辆数

VALUE.append(len(kk))

# 按照天循环日期

datestart += datetime.timedelta(days=1)

#绘制折线图 填充数据

plt.plot(DATE,VALUE)

plt.xticks([])#隐藏x轴刻度

plt.xlabel('2018-01-01至2018-03-31') # 显示日期范围

# 设置标题

plt.title('每日接待车辆统计')

plt.show() # 显示图表

pass

# 停车高峰时间

def tcgf():

# 图表标题

plt.title("停车高峰时间所占比例")

# 设置x轴数据

labels = ['0-3点','3-6点','6-9点','9-12点','12-15点','15-18点',

'18-21点','21-00点']

# 根据时间获取y轴数据 判断包含

kk0 = pi\_table[pi\_table['timein'].str.contains(' 00:')]

kk1 = pi\_table[pi\_table['timein'].str.contains(' 01:')]

kk2 = pi\_table[pi\_table['timein'].str.contains(' 02:')]

kk3 = pi\_table[pi\_table['timein'].str.contains(' 03:')]

kk4 = pi\_table[pi\_table['timein'].str.contains(' 04:')]

kk5 = pi\_table[pi\_table['timein'].str.contains(' 05:')]

kk6 = pi\_table[pi\_table['timein'].str.contains(' 06:')]

kk7 = pi\_table[pi\_table['timein'].str.contains(' 07:')]

kk8 = pi\_table[pi\_table['timein'].str.contains(' 08:')]

kk9 = pi\_table[pi\_table['timein'].str.contains(' 09:')]

kk10 = pi\_table[pi\_table['timein'].str.contains(' 10:')]

kk11 = pi\_table[pi\_table['timein'].str.contains(' 11:')]

kk12 = pi\_table[pi\_table['timein'].str.contains(' 12:')]

kk13 = pi\_table[pi\_table['timein'].str.contains(' 13:')]

kk14 = pi\_table[pi\_table['timein'].str.contains(' 14:')]

kk15 = pi\_table[pi\_table['timein'].str.contains(' 15:')]

kk16 = pi\_table[pi\_table['timein'].str.contains(' 16:')]

kk17 = pi\_table[pi\_table['timein'].str.contains(' 17:')]

kk18 = pi\_table[pi\_table['timein'].str.contains(' 18:')]

kk19 = pi\_table[pi\_table['timein'].str.contains(' 19:')]

kk20 = pi\_table[pi\_table['timein'].str.contains(' 20:')]

kk21 = pi\_table[pi\_table['timein'].str.contains(' 21:')]

kk22 = pi\_table[pi\_table['timein'].str.contains(' 22:')]

kk23 = pi\_table[pi\_table['timein'].str.contains(' 23:')]

# 设置数据信息

x = [(len(kk0)+len(kk1)+len(kk2)),(len(kk3)+len(kk4)+len(kk5)),

(len(kk6)+len(kk7)+len(kk8)),( len(kk9)+len(kk10)+len(kk11)),

(len(kk12)+len(kk13)+len(kk14)),(len(kk15)+len(kk16)+len(kk17)),

(len(kk18)+len(kk19)+len(kk20)),(len(kk21)+len(kk22)+len(kk23))]

# 设置饼图,autopct保留1位小数点

plt.pie(x, labels=labels, autopct='%1.1f%%')

plt.axis('equal') # 该行代码使饼图长宽相等

plt.legend(loc="upper right", fontsize=10, bbox\_to\_anchor=(1.1, 1.05), borderaxespad=0.3) # 显示图例

plt.show() # 显示图表

pass

# 收入分析（月）

def ysrfx():

srdf = pi\_table.loc[pi\_table['state'] == 1]

# 筛选每月数据

kk1 = srdf[srdf['timeout'].str.contains('2018-01')]

kk2 = srdf[srdf['timeout'].str.contains('2018-02')]

kk3 = srdf[srdf['timeout'].str.contains('2018-03')]

# 计算价格和

price1 = kk1['price'].sum()

price2 = kk2['price'].sum()

price3 = kk3['price'].sum()

labels\_x = ['1月', '2月', '3月']

y = [price1, price2, price3]

# 设置柱状图

plt.bar(labels\_x,y)

# 为每一个图形加数值标签

for x, y in enumerate(y):

plt.text(x, y + 300, str(y)+'元', ha='center')

# x,y轴显示文字

plt.xlabel('月份')

plt.ylabel('元')

# 设置标题

plt.title("2018年1-3月收入分析-总收入："+str(price1+ price2+price3)+"元")

plt.show() # 显示图表

# 停车时间分布

def sjfb():

# 图表标题

plt.title("停车时间分布图")

# 设置x轴信息

labels\_x = ['1小时','2小时','3-5小时','6-10小时','11-12小时','12小时以上']

# 获取表中数据判断车辆停车时间

df1 = pi\_table.loc[(pi\_table['price'] == 3)] # 停车1小时

df2 = pi\_table.loc[(pi\_table['price'] == 6)] # 停车2小时

# 停车3-5小时

df3 = pi\_table.loc[(pi\_table['price']>6)&(pi\_table['price']<=15)]

# 停车6-10小时

df4 = pi\_table.loc[(pi\_table['price']>15)&(pi\_table['price']<=30)]

# 停车11-12小时

df5 = pi\_table.loc[(pi\_table['price']>30)&(pi\_table['price']<=36)]

df6 = pi\_table.loc[(pi\_table['price']>36)] # 停车12小时以上

# 停车各时间段停车数量

y=[len(df1),len(df2),len(df3),len(df4),len(df5),len(df6)]

plt.bar(labels\_x,y) # 绘制条形图

# 为每一个图形加数值标签

for x, y in enumerate(y):

plt.text(x, y + 30, str(y)+'台', ha='center')

plt.show() # 显示条形图窗体

# 主线程

Running =True

while Running:

# 创建停车时间分布按钮

btn1 = btn.Button(screen, (90, 50), 140, 60, BLUE, WHITE, "停车时间分布", 20)

# 绘制停车时间分布的按钮

btn1.draw\_button()

# 创建停车高峰时间按钮

btn2 = btn.Button(screen, (90, 130), 140, 60, BLUE, WHITE, "停车高峰时间", 20)

# 绘制停车高峰时间的按钮

btn2.draw\_button()

# 创建周繁忙统计按钮

btn3 = btn.Button(screen, (90, 210), 140, 60, BLUE, WHITE, "周繁忙统计", 20)

# 绘制周繁忙统计的按钮

btn3.draw\_button()

# 创建月收入分析按钮

btn4 = btn.Button(screen, (250, 50), 140, 60, BLUE, WHITE, "月收入分析", 20)

# 绘制月收入分析的按钮

btn4.draw\_button()

# 创建接待车辆统计按钮

btn5 = btn.Button(screen, (250, 130), 140, 60, BLUE, WHITE, "接待车辆统计", 20)

# 绘制接待车辆统计的按钮

btn5.draw\_button()

# 创建车位利用率按钮

btn6 = btn.Button(screen, (250, 210), 140, 60, BLUE, WHITE, "车位利用率", 20)

# 绘制车位利用率的按钮

btn6.draw\_button()

# 更新主窗口

pygame.display.update()

for event in pygame.event.get():

# 关闭页面游戏退出

if event.type == pygame.QUIT:

plt.close()

# 退出

pygame.quit()

exit()

# 判断点击

elif event.type == pygame.MOUSEBUTTONDOWN:

# 鼠标点击位置，判断单击“停车时间分布”按钮

if 20 <= event.pos[0] and event.pos[0] <= 90+70 \

and 20 <= event.pos[1] and event.pos[1] <= 50+30:

sjfb() # 停车时间分布

pass

# 判断单击“月收入分析”按钮

elif 180 <= event.pos[0] and event.pos[0] <= 250+70 \

and 20 <= event.pos[1] and event.pos[1] <= 50+30:

ysrfx() # 收入分析（月）

pass

# 判断单击“停车高峰时间”按钮

elif 20 <= event.pos[0] and event.pos[0] <= 90+70 \

and 100 <= event.pos[1] and event.pos[1] <= 130+30:

tcgf() # 停车高峰时间

pass

# 判断单击“接待车辆统计”按钮

elif 180 <= event.pos[0] and event.pos[0] <= 250+70 \

and 100 <= event.pos[1] and event.pos[1] <= 130 + 30:

cljd() # 每日接待车辆统计

pass

# 判断单击“周繁忙统计”按钮

elif 20 <= event.pos[0] and event.pos[0] <= 90+70 \

and 180 <= event.pos[1] and event.pos[1] <= 210 + 30:

fmtj() # 周繁忙统计

pass

# 判断单击“车位利用率”按钮

elif 180 <= event.pos[0] and event.pos[0] <= 250+70 \

and 180 <= event.pos[1] and event.pos[1] <= 210 + 30:

lyl() # 车位每天利用率

pass

**btn.py**

import pygame

# 自定义按钮

class Button():

# msg为要在按钮中显示的文本

def \_\_init\_\_(self,screen,centerxy,width, height,button\_color,text\_color, msg,size):

"""初始化按钮的属性"""

self.screen = screen

# 按钮宽高

self.width, self.height = width, height

# 设置按钮的rect对象颜色为深蓝

self.button\_color = button\_color

# 设置文本的颜色为白色

self.text\_color = text\_color

# 设置文本为默认字体，字号为20

self.font = pygame.font.SysFont('SimHei', size)

# 设置按钮大小

self.rect = pygame.Rect(0, 0, self.width, self.height)

# 创建按钮的rect对象，并设置按钮中心位置

self.rect.centerx = centerxy[0]

self.rect.centery= centerxy[1]

# 渲染图像

self.deal\_msg(msg)

def deal\_msg(self, msg):

"""将msg渲染为图像，并将其在按钮上居中"""

# render将存储在msg的文本转换为图像

self.msg\_img = self.font.render(msg, True, self.text\_color, self.button\_color)

# 根据文本图像创建一个rect

self.msg\_img\_rect = self.msg\_img.get\_rect()

# 将该rect的center属性设置为按钮的center属性

self.msg\_img\_rect.center = self.rect.center

def draw\_button(self):

# 填充颜色

self.screen.fill(self.button\_color, self.rect)

# 将该图像绘制到屏幕

self.screen.blit(self.msg\_img, self.msg\_img\_rect)

**TimeUtil.py**

# 引入模块

import datetime

# 返回 星期几标记 0代表星期一 1代表星期二...6代表星期天

def get\_week\_numbeer(date):

date = datetime.datetime.strptime(date, "%Y-%m-%d %H:%M:%S")

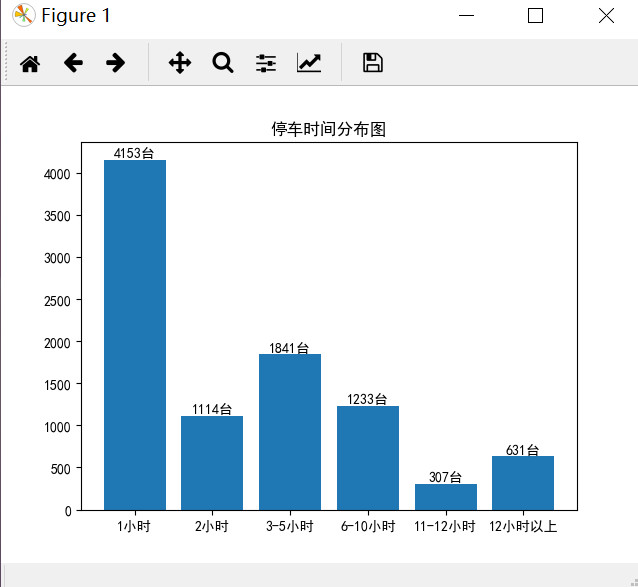
day = date.weekday()

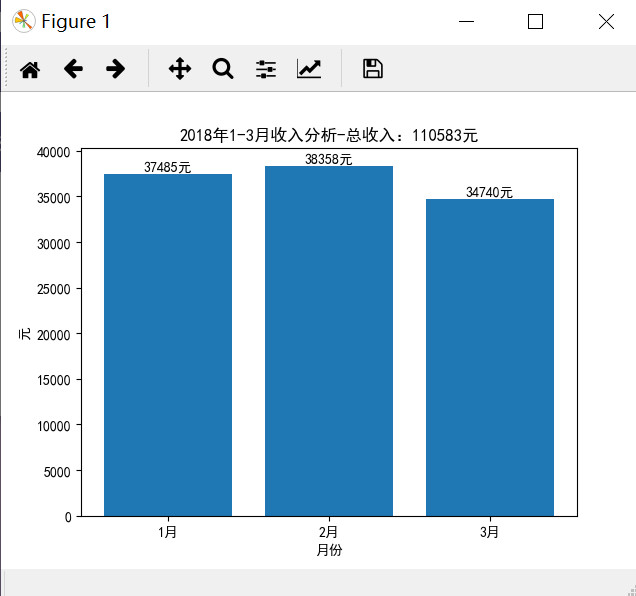
return day

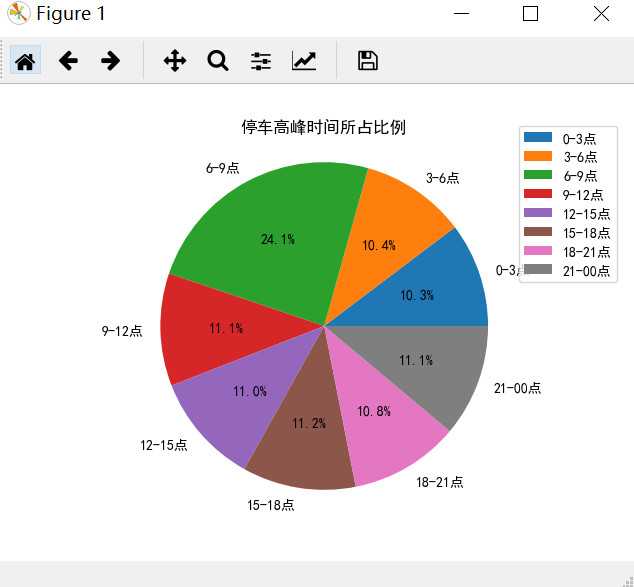
1. **系统预览**

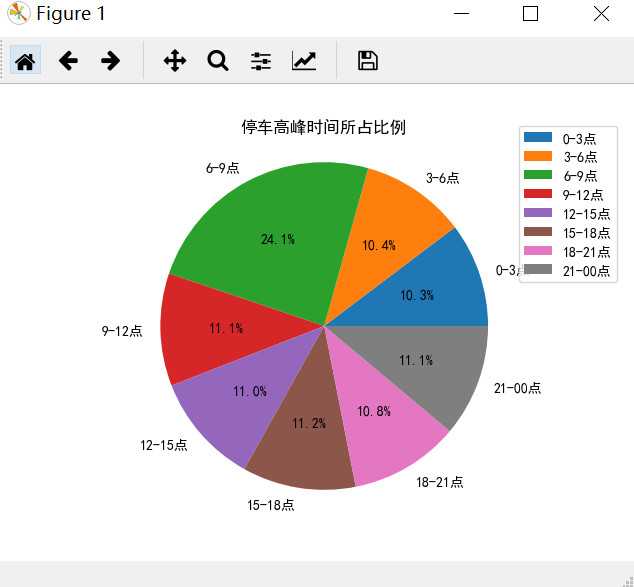
****

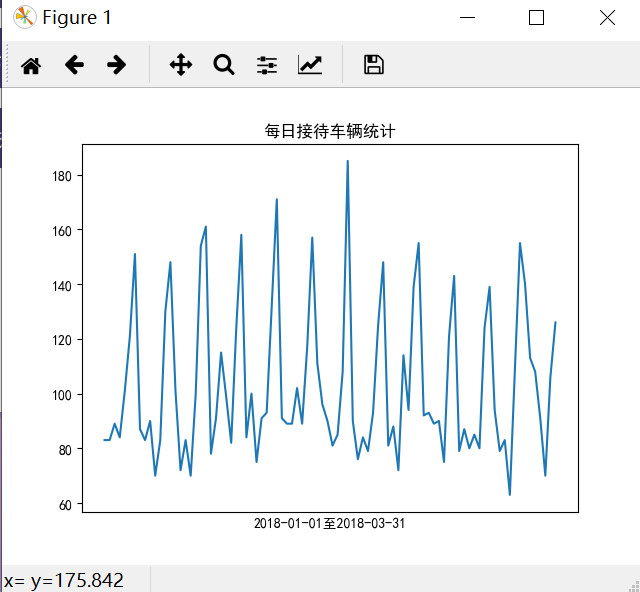
**五、实验结果与分析**

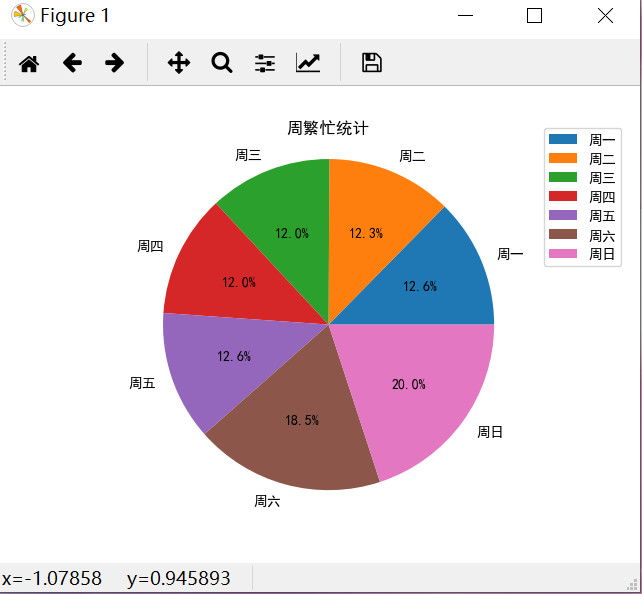
****

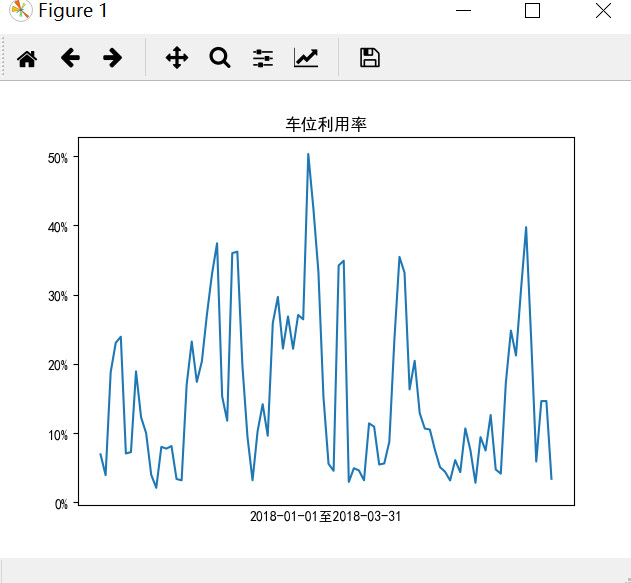
****

****

****

****

****

****

**六、结论与体会**

结论：学习了用Pygame搭建应用窗体。

体会：不够熟练要多练习。

**七、教师评语**

实验中操作过程和结果合理完整，完成及时，质量较好;表明学生能够将课 堂上学习的。