广州商学院
Guangzhou College of Commerce

实验指导书

实验课程：数据分析与处理实验

编写教师：邹锋

适用专业：软件工程

编写日期：2022.2

前　言

本实验课程是与《数据分析与处理实验》课程的理论教学内容相配套而开设的。数据分析是软件工程专业中一门重要的专业选修课程，是实现数据价值的重要工具，数据分析的目的是把隐藏在一大批看来杂乱无章的数据中的信息集中和提炼出来，从而找出所研究对象的内在规律。数据分析可帮助人们做出判断，以便采取适当行动。数据分析是有组织有目的地收集数据、分析数据，使之成为信息的过程。通过该课程学习，从一些基础的数据分析开始，逐渐延伸到数据分析综合实验，其目标是培养学生掌握数据分析技术，能够独立完成数据分析处理工作。课程内容是进入大数据处理、数据挖掘、以数据为中心人工智能领域的必备实践基础。本课程完整地讲解了数据分析与处理的实践操作技能，培养独立完成大数据项目分析与处理的能力。为毕业后从事大数据分析与开发相关工作奠定基础，重点培养学生运用当代最优秀第三方库资源，快速分析和解决问题的能力。

本实验指导书选用了 4 个实验项目。主要客户价值分析、销售收入分析与预测、二手房数据分析预测系统、智能停车场运营分析系统；实验指导书中给出了实验目的、实验设备、实验内容，学生在做实验前务必先认真预习实验指导书，注意应用相关原理解决实际的数据分析问题。

由于编者水平有限，难免在本实验指导书中出现错误或不妥之处，望读者指正。

实验一 客户价值分析

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。

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第三方模块 ： pandas、numpy、matplotlib、sklearn。

四、实验内容

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
```

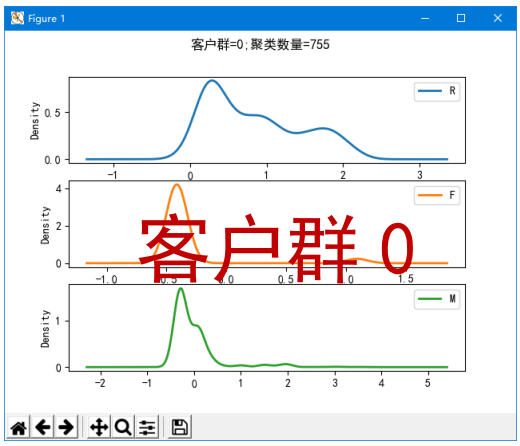
```
#from sklearn.manifold import

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()
```

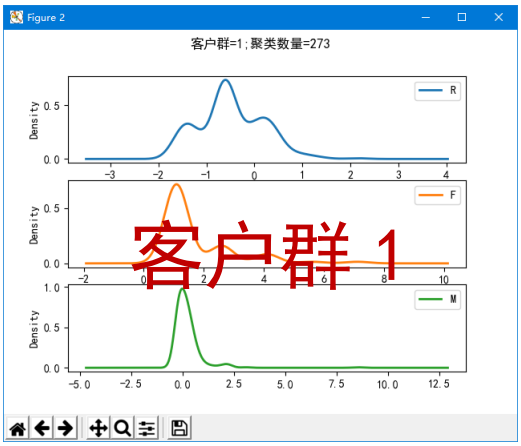
data_view.py

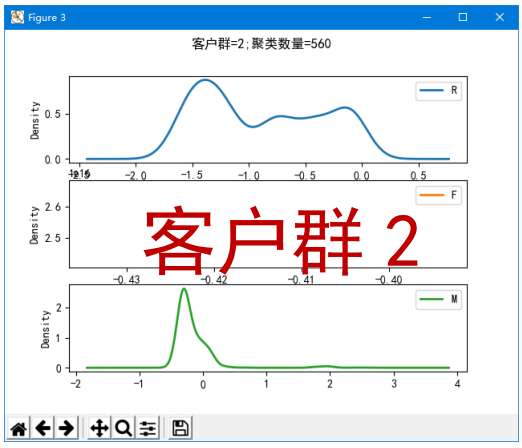
```
#-*- coding: utf-8 -*-
#对数据进行基本的探索
#返回缺失值个数以及最大最小值

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']]
```

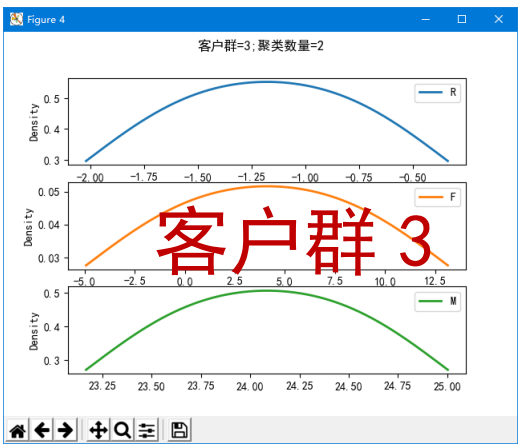
```
view = data.describe(percentiles = [], include = 'all').T #包括对数据的基本描述，percentiles 参数是指定计算多少的分位数表（如 1/4 分位数、中位数等）；T 是转置，转置后更方便查阅
view['null'] = len(data)-view['count'] #describe()函数自动计算非空值数，需要手动计算空值数

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









4、客户价值分析结果解析

R	F	M	客户类别	客户数	排名
低	高	高	重要保持客户	2	
高	低	低	一般发展客户	755	

低	高	高	重要保持客户	273	
低	低	低	一般挽留客户	560	

实验二 、销售收入分析与预测

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

二、实验原理

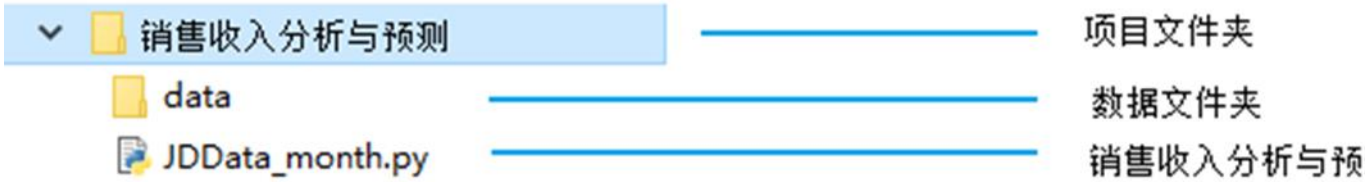
1、教学机与学生机需要安装第三方模块 pandas、numpy、matplotlib、sklearn
2、教学机与学生机需要安装 PyCharm 和 Anaconda3

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

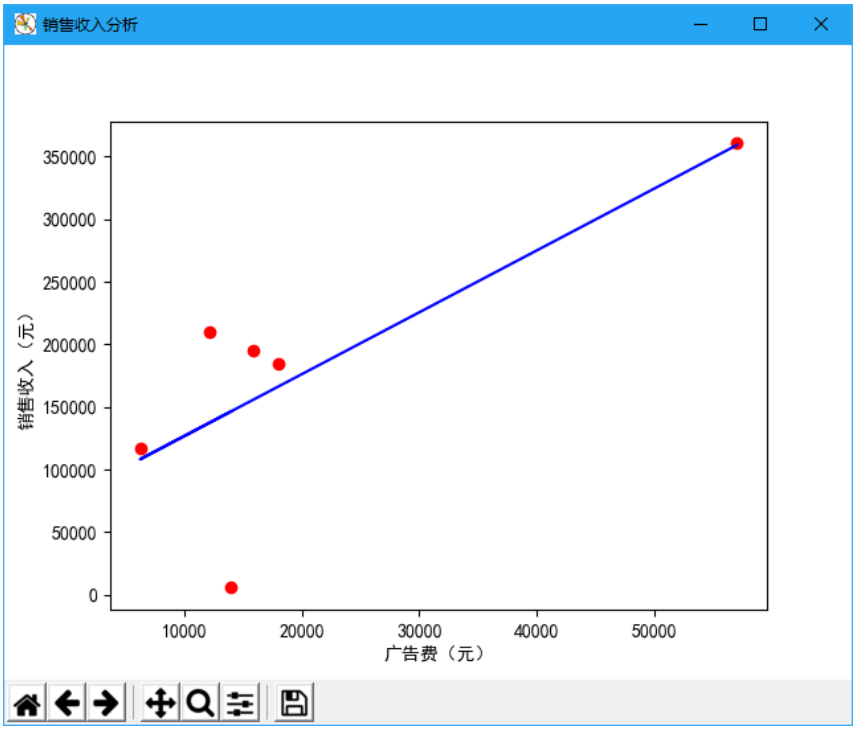
JDData_month.py

```
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



实验三　二手房数据分析预测系统

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

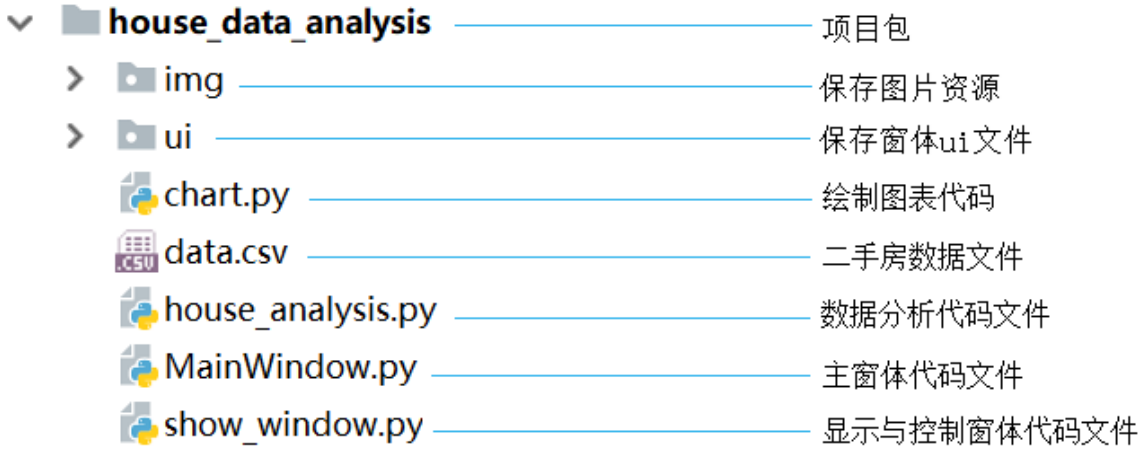
1、教学机与学生机需要安装 PyCharm 和 Anaconda3
2、教学机与学生机需要安装第三方模块 PyQt5、pyqt5-tools、matplotlib、sklearn、pandas

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**


二手房数据分析预测系统
各区二手房均价分析
各区二手房数量所占比例
全市二手房装修程度分析
二手房数据分析预测系统

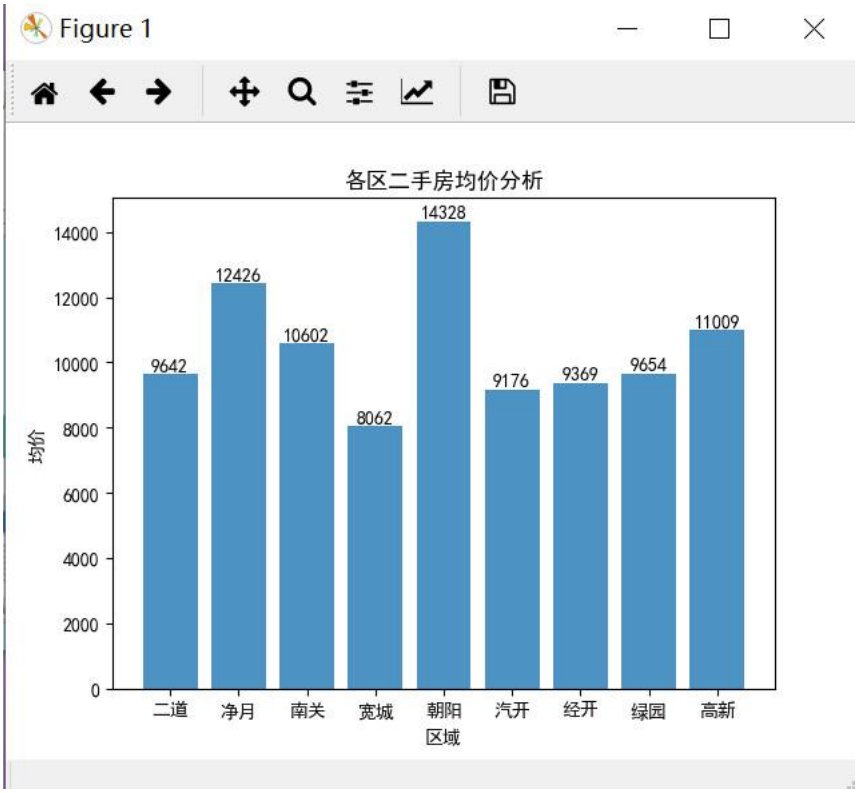


Figure 1
各区二手房均价分析
均价
区域
二道
净月
南关
宽城
朝阳
汽开
经开
绿园
高新
9642
12426
10602
8062
14328
9176
9369
9654
11009

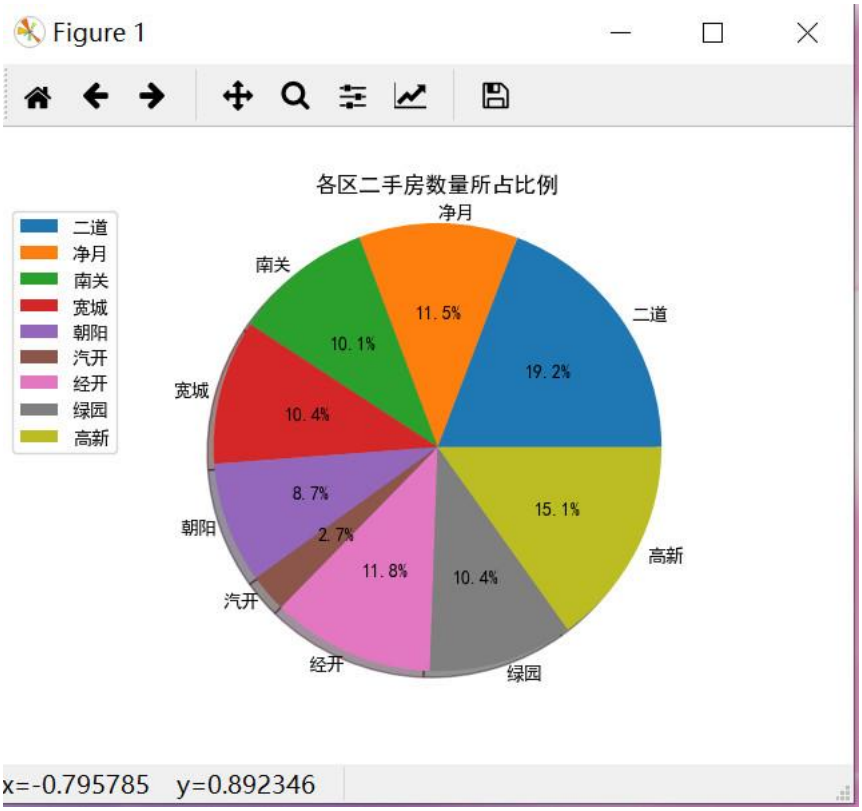


Figure 1
各区二手房数量所占比例
二道
净月
南关
宽城
朝阳
汽开
经开
绿园
高新
19.2%
11.5%
10.1%
10.4%
8.7%
2.7%
11.8%
10.4%
15.1%
x=-0.795785 y=0.892346

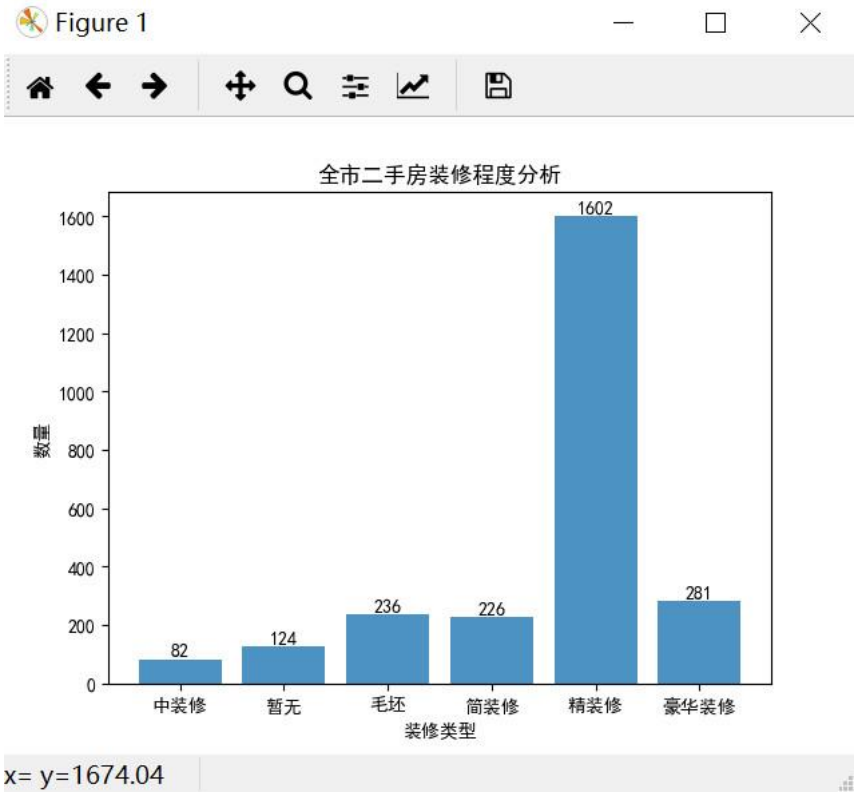


Figure 1
全市二手房装修程度分析
数量
装修类型
中装修
暂无
毛坯
简装修
精装修
豪华装修
82
124
236
226
1602
281
x= y=1674.04

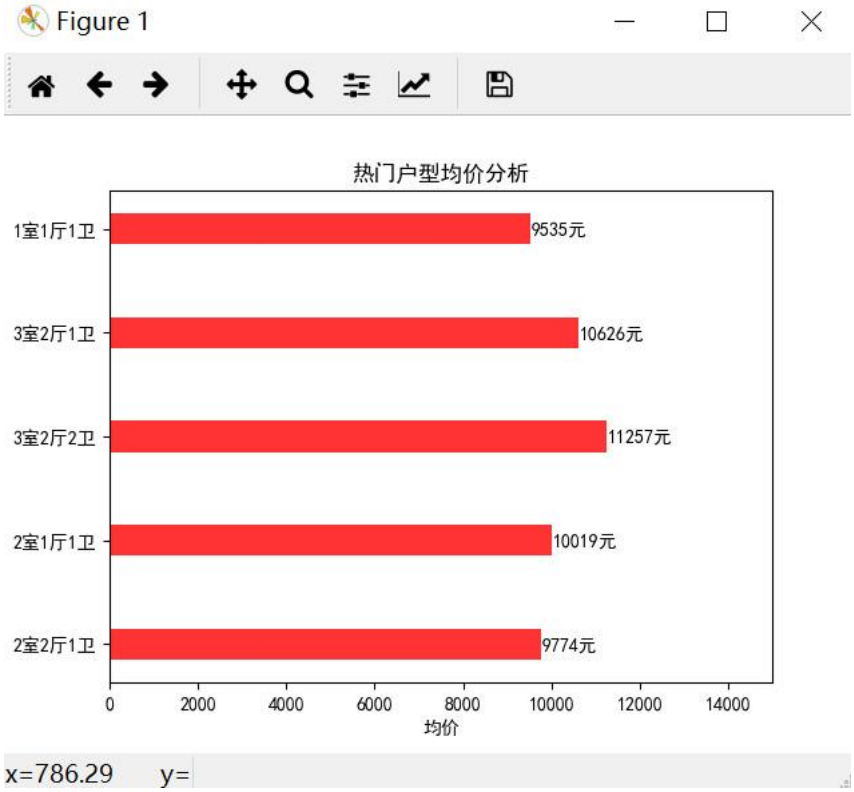


Figure 1
热门户型均价分析
1室1厅1卫
3室2厅1卫
3室2厅2卫
2室1厅1卫
2室2厅1卫
9535元
10626元
11257元
10019元
9774元
0
2000
4000
6000
8000
10000
12000
14000
均价
x=786.29
y=

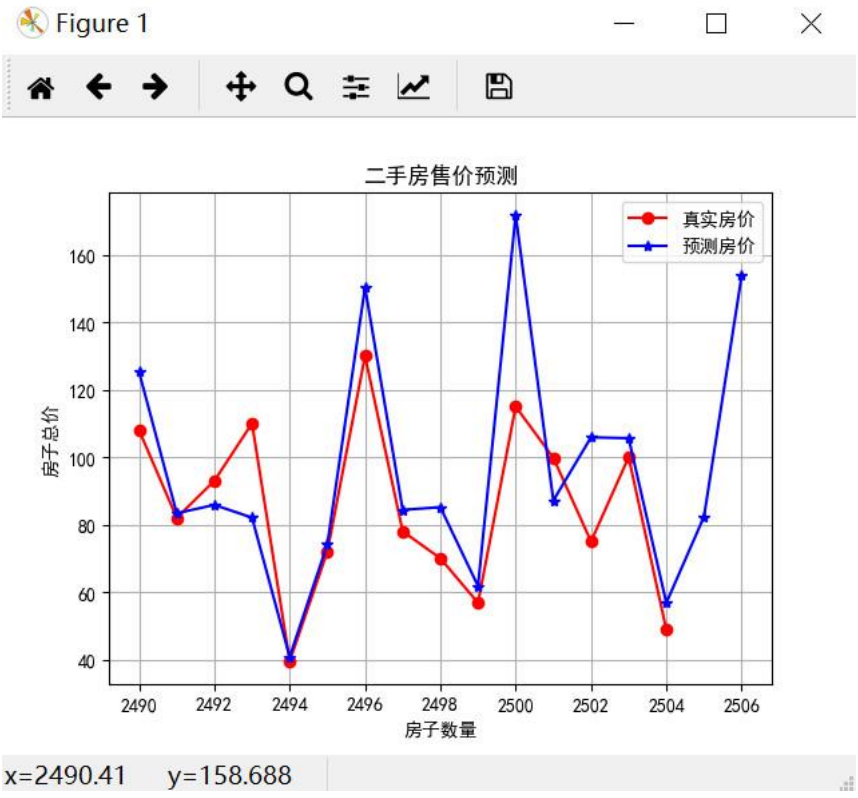


Figure 1
二手房售价预测
真实房价
预测房价
房子总价
160
140
120
100
80
60
40
2490
2492
2494
2496
2498
2500
2502
2504
2506
房子数量
x=2490.41
y=158.688

实验四　智能停车场运营分析系统

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

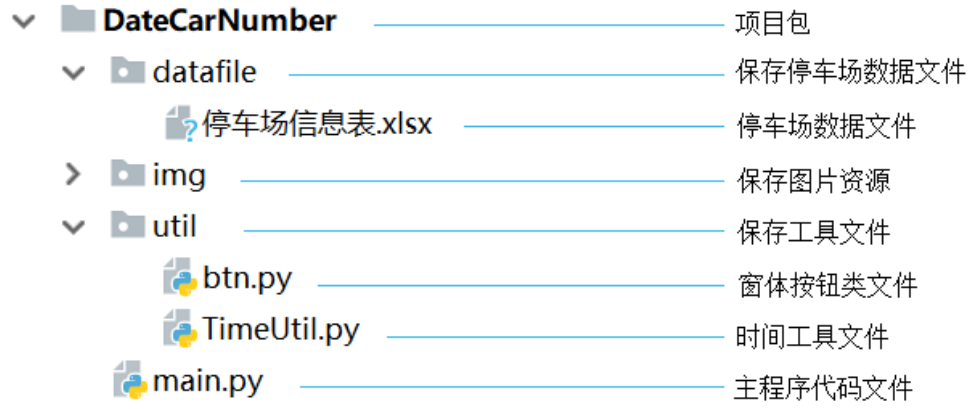
1、教学机与学生机需要安装 PyCharm 和 Anaconda3
2、教学机与学生机需要安装第三方模块 pygame、matplotlib、pandas
3、内置模块：time、datetime、sys

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

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
```

3、**系统预览**

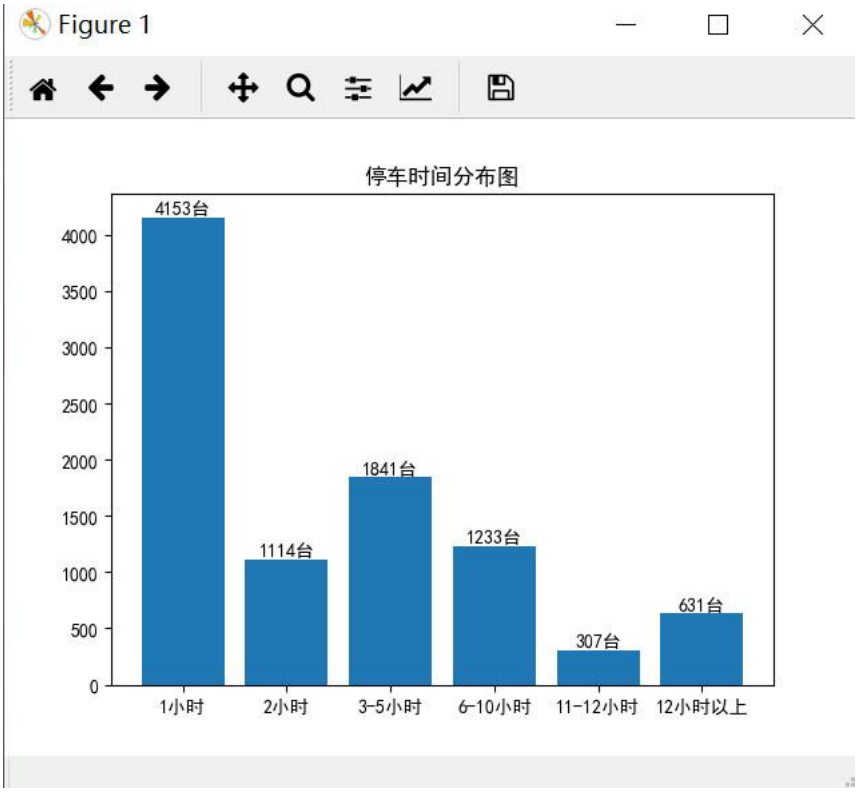


Figure 1
停车时间分布图
4153台
1114台
1841台
1233台
307台
631台
0
500
1000
1500
2000
2500
3000
3500
4000
1小时
2小时
3-5小时
6-10小时
11-12小时
12小时以上

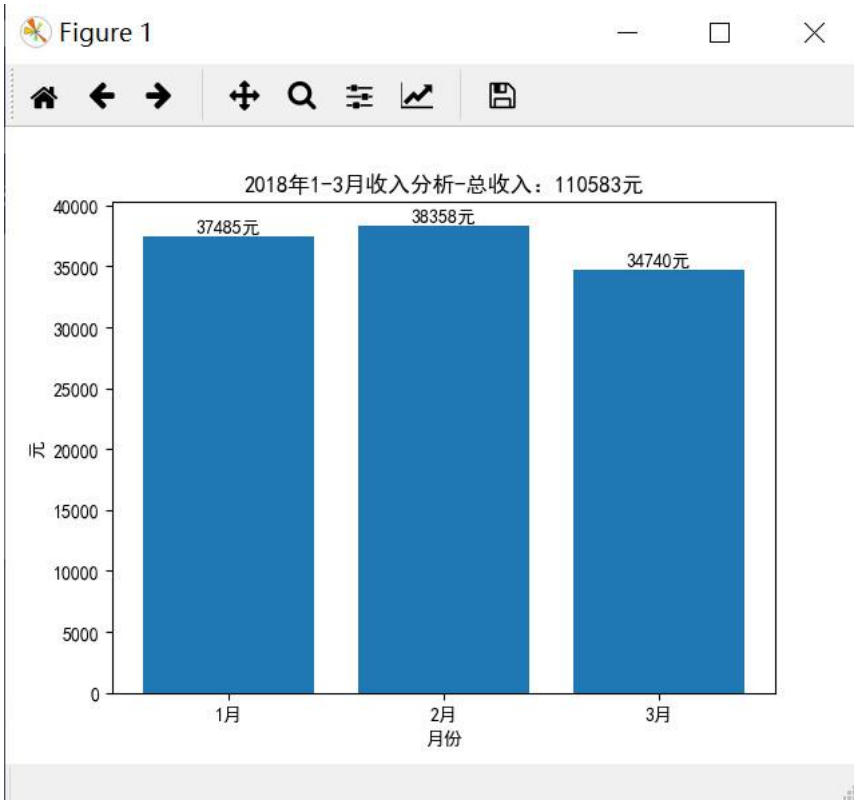


Figure 1
2018年1-3月收入分析-总收入：110583元
37485元
38358元
34740元
0
5000
10000
15000
20000
25000
30000
35000
40000
1月
2月
3月
月份

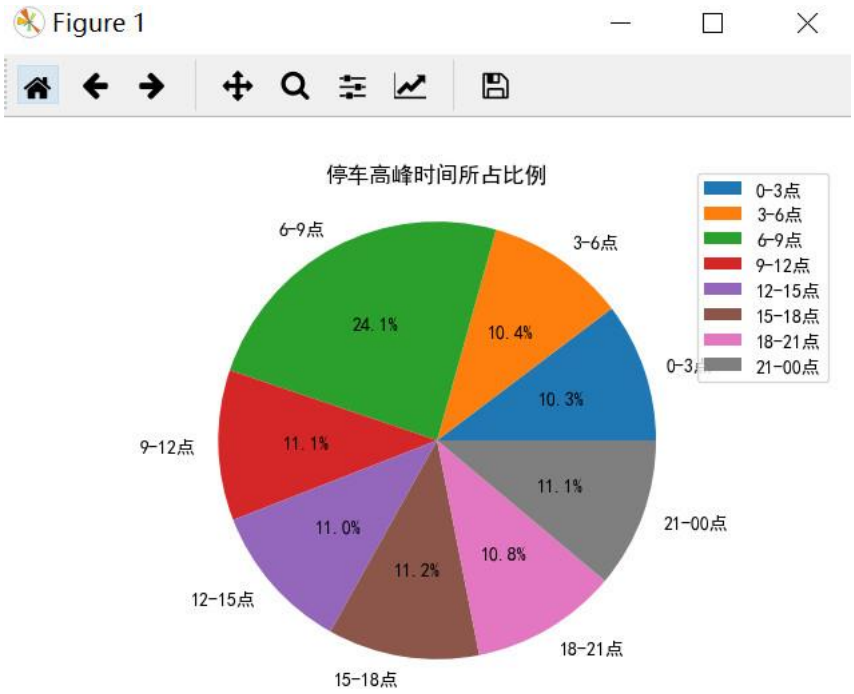


Figure 1
停车高峰时间所占比例
0-3点
3-6点
6-9点
9-12点
12-15点
15-18点
18-21点
21-00点
24.1%
10.4%
10.3%
11.1%
11.1%
10.8%
11.2%
11.0%

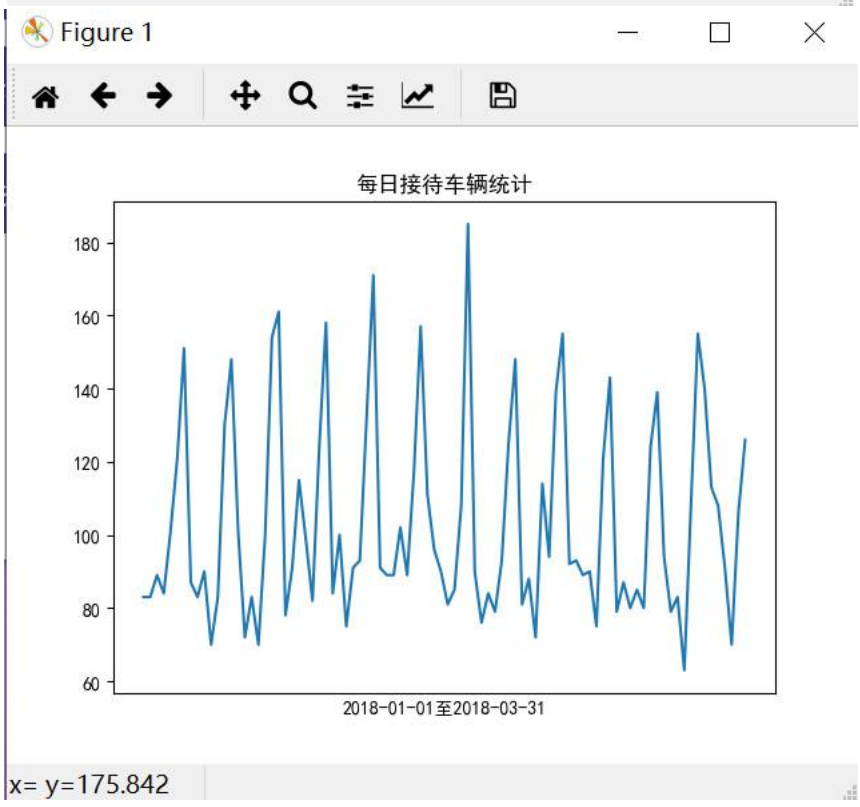


Figure 1
每日接待车辆统计
180
160
140
120
100
80
60
2018-01-01至2018-03-31
x= y=175.842

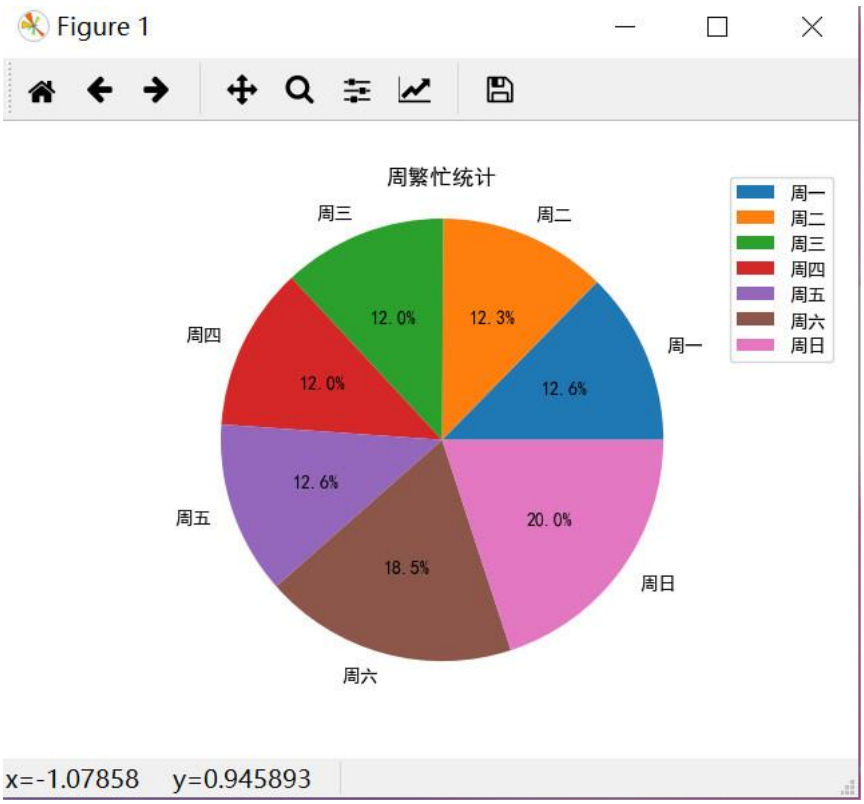


Figure 1
周繁忙统计
周一
周二
周三
周四
周五
周六
周日
12.6%
12.3%
12.0%
12.0%
12.6%
18.5%
20.0%
x=-1.07858 y=0.945893

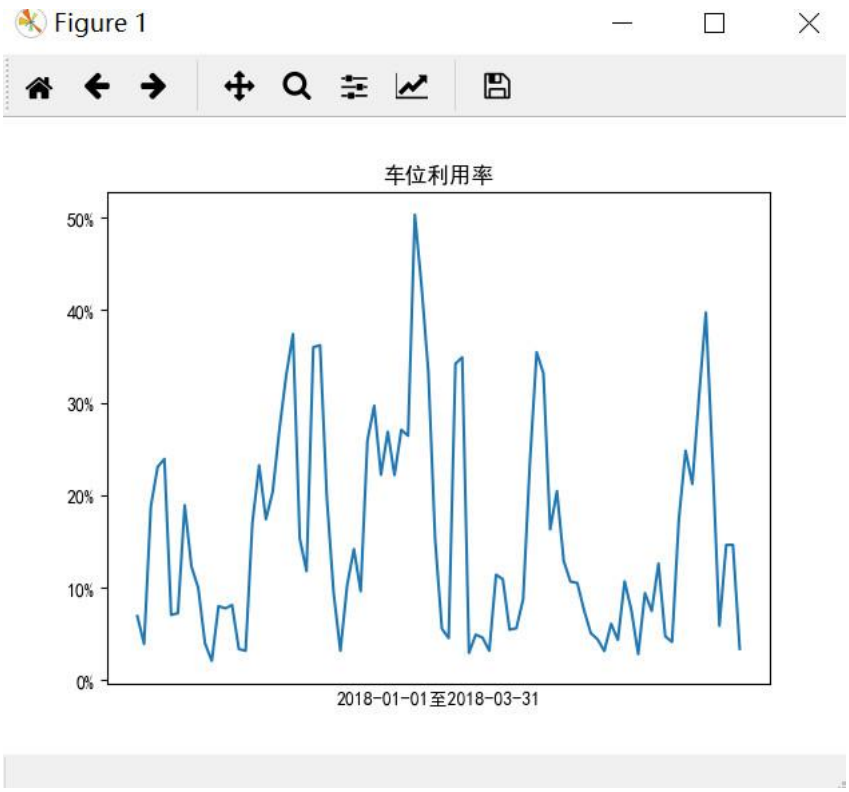


Figure 1
车位利用率
50%
40%
30%
20%
10%
0%
2018-01-01至2018-03-31