**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

**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

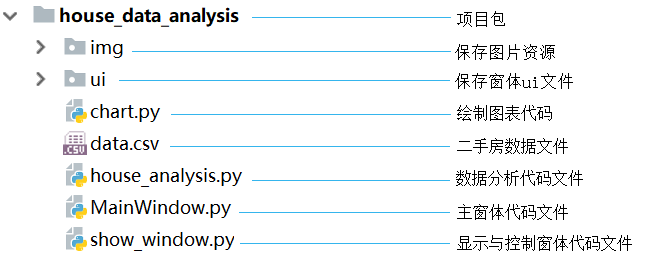
2.教师机 1 台

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

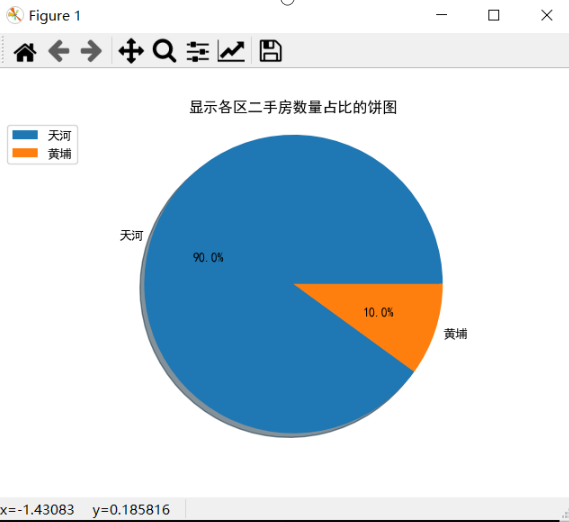
main.show()

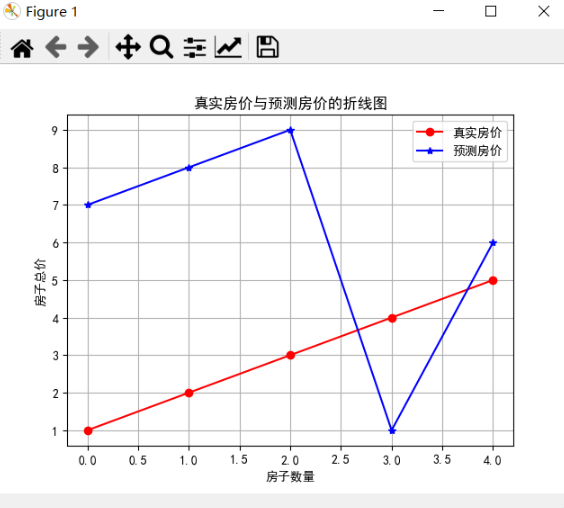
sys.exit(app.exec\_()) # 当窗口创建完成，需要结束主循环过程

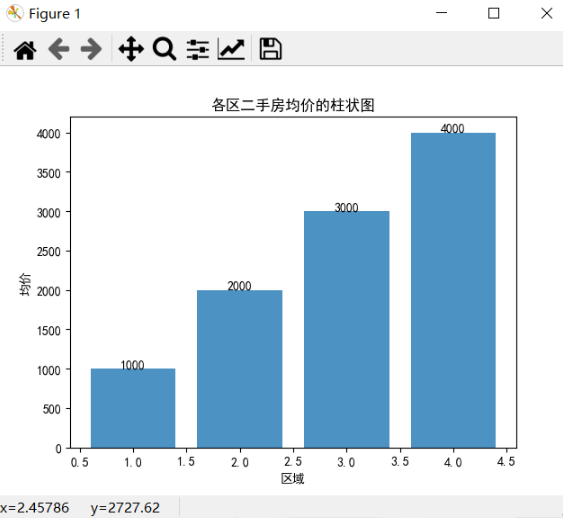
3、**系统预览**

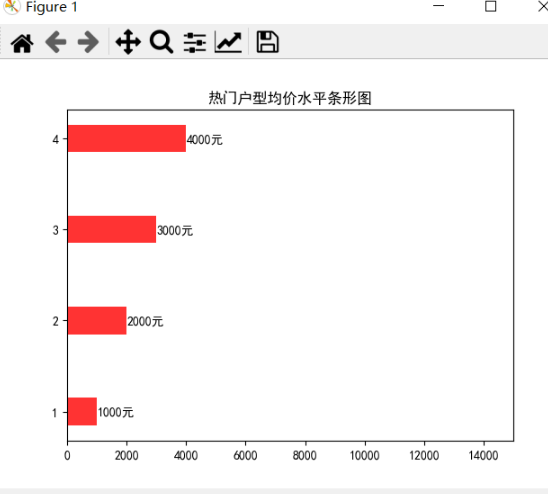


**五、实验结果与分析**









**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**