# 基于 Python 爬虫的南京二手房信息可视化和智能预测系统

**程序名称：**南京二手房智能小帮手

**程序版本：**v1.0

**程序作者：*****

**程序编程语言和编译器：**Python，Jupyter Notebook

**需求分析：**

如今江苏南京的发展十分迅速，政府每年制定更具吸引力的人才引进政策，大量的知名商业中心和国际型企业入驻江苏南京，南京的智能制造和尖端科技发展日新月异，因此南京的房价也节节攀升，许多年轻人在南京工作后期望成家立业，而关乎最大的房产价格。房产价格实时在变化，二手房源也在逐渐增多，年轻人对选择二手房的类型、区域、楼层和价格等选择都存在困惑，他们也无法时时刻刻去了解二手房的大盘信息，同时，年轻人也期望得到一套完整的算法，输入二手房的楼层、朝向、区域和面积等属性能精准的预测二手房单价所属的波动区间，方便做下一步的购房计划。此外，各大二手房信息平台的竞争也颇为激烈，期望通过别的二手房网站获取完整的房源信息和大盘信息供参考研究。



**程序基本流程介绍：**

文本设计了基于 Python 的南京二手房信息可视化和智能预测系统，该系统可爬取链家南京各区域的二手房源信息，包含信息字段：单价，总价，房型，楼层，产年，结构，面积等，并保存未 Json 文件。随后通过 DataFrame 读取并可视化数据集，采用检索和求均值等算法做关联可视化，让用户了解各区域的平均房价、各房型的平均房价等图表信息。最后，程序还将字段数据清洗后进行 One-hot 编码，采用机器学习决策树算法，训练学习生成能通过房产各属性预测房价区间的分类算法模型。

**程序各功能详细分析（伪代码，源码附页）：**

**1. 爬虫访问：**

爬虫访问主要分为两个阶段：构造访问 URL 链接和 Requests 访问，涉及外接库为 requests。

阶段一，我发现不同的区域和页面，链接采用了统一的 URL 构造方法：

https://nj.lianjia.com/ershoufang/gulou/pg2 中，“nj”代表南京，“ershoufang”代表二手房，” gulou”代表鼓楼，” pg2”代表 Page(页面 2)。因此，我们需要爬取南京 11 个区域，100 页的二手房信息，即构造区域 List 并循环遍历 100 即可构造需要的 1100 个 URL 链接。（此处由于无需登入访问，因此在 V2.0 可以用多线程爬虫做优化。

阶段一伪代码：

```
Def Generate_URL():
    Create List_region [11 Regions in Nanjing];
    for i in List_region:
        for j in page_num:
            url = 'https://nj.lianjia.com/ershoufang/' + region + '/pg'+ str(i) + '/'
```

阶段二，需要用 Requests 模拟访问阶段一生成的 1100 个 URL 链接：

分析原网页我们发现，无需进行模拟登录，因此 Headers 直接传入浏览器属性参数即可，无需 Cookies，本步骤的实现方法较为简单，直接采用 Requests.get 方法即可。

阶段二伪代码：

```
Def Request_URLs(url):
    Headers = {输入你的浏览器属性}
    R = requests.get(url,headers)
```

**2. 页面解析（bs4）：**

HTML 的页面构造较为简单，我们获取 URL 的 text 信息后，只需要采用 bs4 的 BeautifulSoup 功能提取我们需要的信息即可，定位方式是“特定属性（data-lj_action_source_type）和标签”。最终取出['区域','房型','面积','朝向','装修','楼层','始建年份','房屋类型','总价','单价']信息。

页面解析伪代码：

```
    Def Analyse_URLs(r):
        soup = BeautifulSoup(r.text, 'lxml')
        results = soup.findall(属性表达式定位)
        for items in results:
            根据标签取值并存储
```

**3. 数据预处理（pandas+re）：**

首先，我们用 Pandas.Dataframe 可视化下我们的数据集，总共是 26268 条房源信息，由于数据没法长时间在 Jupyter Notebook 中存储，因此我们存储为 Json 文件，方便我们后期直接读取和分析。

Out[46]:

| | 区域 | 房型 | 面积 | 朝向 | 装修 | 楼层 | 始建年份 | 房屋类型 | 总价 | 单价 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | gulou | 2室2厅 | 56.91平米 | 南北 | 简装 | 低楼层(共5层) | 1989年建 | 板楼 | 450万 | 单价79073元/平米 |
| 1 | gulou | 4室1厅 | 127.8平米 | 南 | 简装 | 中楼层(共17层) | 1999年建 | 板楼 | 855万 | 单价66902元/平米 |
| 3 | gulou | 3室1厅 | 75.5平米 | 南北 | 精装 | 中楼层(共7层) | 1995年建 | 板楼 | 598万 | 单价79206元/平米 |
| 4 | gulou | 3室1厅 | 71.63平米 | 南北 | 精装 | 低楼层(共7层) | 1995年建 | 板楼 | 345万 | 单价48165元/平米 |
| 5 | gulou | 1室1厅 | 38.91平米 | 南 | 简装 | 高楼层(共6层) | 1991年建 | 板楼 | 355万 | 单价91237元/平米 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 30035 | gaochun | 3室2厅 | 97.07平米 | 南北 | 其他 | 中楼层(共6层) | 2016年建 | 板楼 | 58万 | 单价5976元/平米 |
| 30036 | gaochun | 3室2厅 | 97.07平米 | 南北 | 其他 | 中楼层(共6层) | 2016年建 | 板楼 | 55万 | 单价5667元/平米 |
| 30039 | gaochun | 3室2厅 | 148平米 | 南北 | 其他 | 低楼层(共9层) | 2014年建 | 板塔结合 | 233万 | 单价15744元/平米 |
| 30041 | gaochun | 3室2厅 | 118.18平米 | 南北 | 其他 | 高楼层(共6层) | 2013年建 | 板楼 | 85万 | 单价7193元/平米 |
| 30045 | gaochun | 4室2厅 | 113.63平米 | 南 | 其他 | 中楼层(共23层) | 2018年建 | 板楼 | 96万 | 单价8449元/平米 |

26268 rows × 10 columns

其次，不论是可视化还是做决策树机器学习算法的训练学习模型，我们都需要对数据集进行预处理操作，主要采用 Pandas+Re 的处理手段，通过 pd.groupby()函数我们发现，有些变量/字段存在小样本的情况，比如朝向存在”南南北东东西”，房型存在”7 室 3 厅”等内容，而这些内容对于我们的分析和建模起不到效果（训练学习的效果很差）因此我打算直接做删除操作。

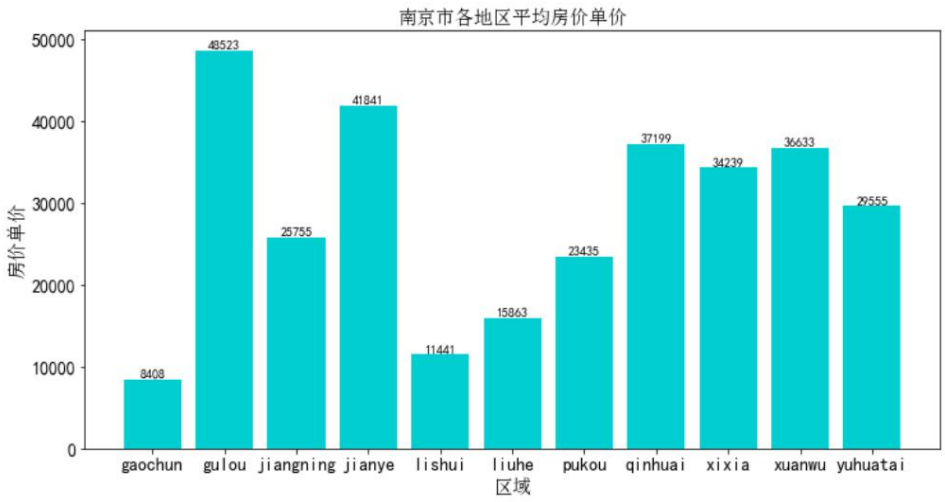
此外，我们需要统一我们各解释变量的数据格式，数据建模阶段采用 One-Hot 编码，目前可视化阶段主要是把”单价 79073 元/平米”等数据采用正则表达式提取为”79073”的数值型变量。　正则表达式提取数字的示例：

```
pattern = re.compile("\d+")
df["单价"] = df["单价"].map(lambda x:int(re.findall(pattern,x)[0]))
```

**4. 数据可视化（matplotlib.pyplot）：**

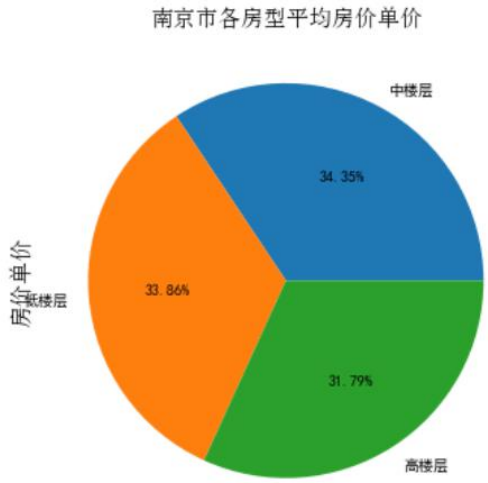
本文主要采用条形图和扇形图两种可视化方式帮助购房者了解目前二手房的大盘情况，主要制作了三张图表。本文采用 matplotlib.pyplot 库，导入相关字体，创建相应画布，借鉴网上生成图表方式，调整画布长宽、颜色、数据标签等参数。

南京市各地区平均房价单价可视化：




南京市各房型平均房价总价（万元）可视化：


南京市各房型平均房价单价可视化：



**5. 决策时模型建立（sklearn）：**

本文采用主流的机器学习模型库 sklearn 调用决策树模型，由于树模型可视化较为复杂（实际工业界运用广泛），因此本文主要通过训练学习的准确率提升图可视化模型。决策树是很好的分类和回归模型，基于规则训练学习特征，并采用梯度下降的方法无限逼近最终真值，信息增益作为特征选择的方法和损失函数计算表达式。

本文采用的基本均为离散型特征，因此回归树的建立并没有意义，因此我们将单价分为 9 个类别，每一万为一个档次，同时其他特征变量采用 One-Hot 编码。

| | 区域 | 房型 | 面积 | 朝向 | 装修 | 楼层 | 始建年份 | 房屋类型 | 总价 | 单价 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 56.91平米 | 0 | 0 | 0 | 1989年建 | 0 | 450 | 8 |
| **3** | 0 | 1 | 75.5平米 | 0 | 1 | 1 | 1995年建 | 0 | 598 | 8 |
| **4** | 0 | 1 | 71.63平米 | 0 | 1 | 0 | 1995年建 | 0 | 345 | 5 |
| **5** | 0 | 2 | 38.91平米 | 1 | 0 | 2 | 1991年建 | 0 | 355 | 9 |
| **7** | 0 | 0 | 87.34平米 | 1 | 1 | 0 | 2000年建 | 0 | 490 | 6 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **30027** | 10 | 1 | 88.12平米 | 0 | 2 | 2 | 2018年建 | 0 | 90 | 2 |
| **30028** | 10 | 1 | 97平米 | 0 | 1 | 2 | 2016年建 | 0 | 63 | 1 |
| **30029** | 10 | 0 | 96.8平米 | 1 | 0 | 1 | 2016年建 | 0 | 106 | 2 |
| **30030** | 10 | 4 | 97平米 | 1 | 1 | 1 | 2016年建 | 0 | 78 | 1 |
| **30033** | 10 | 4 | 97平米 | 1 | 2 | 1 | 2016年建 | 0 | 66 | 1 |

16132 rows × 10 columns

根据日常中的逻辑和常识，我们特征变量选取['区域','房型','朝向','装修','楼层','房屋类型']，单价的分类作为输出变量/被解释变量，构建决策树模型。

模型构建第一步：

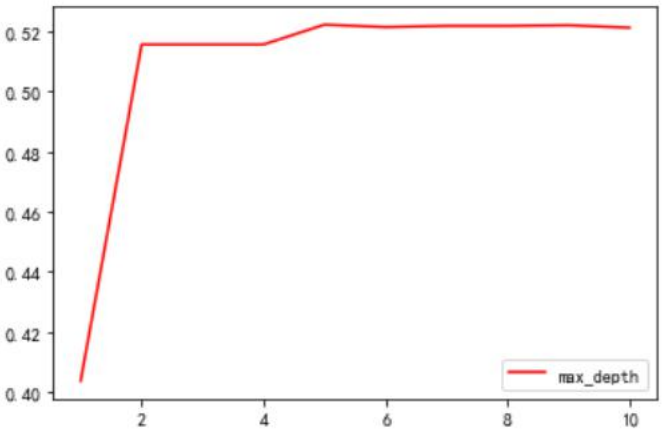
划分训练集和数据集，Xtrain, Xtest, Ytrain, Ytest = train_test_split(X,y,test_size=0.3)，采用 Sklearn 的内置函数划分 30%的数据集为测试数据集。

模型构建第二步：

采用 DecisionTreeClassifier 函数调用决策树模型，针对训练集训练学习模型，最终得到训练后的模型，评估测试集的预测准确度。

模型构建第三步：

优化模型，我们针对 max_depth 最遍历和选择，选择准确度最高的参数。最佳参数是最大深度是 5。




模型构建第四步：

输出每个特征遍历的重要性程度。

[('区域', 0.5295094486134014),

('房型', 0.12493411094168208),

('朝向', 0.09702636270752445),

('装修', 0.06813846745591162),

('楼层', 0.09081565530047828),

('房屋类型', 0.08957595498100189)]

因此，决定二手房单价的特征重要性排序是区域，房型，朝向，装修，楼层和房屋类型。区域的权重极高，和常识相似，一般一些 CBD 区域的房价十分高，而偏远地区的房价自然较低。房型和朝向也是购房者主要考虑的因素之一，对于装修、房屋类型和楼层等消费者不是很在意。

**源代码（含注释）：**

```
import requests,json,time
from bs4 import BeautifulSoup
import pandas as pd
import re
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties
from sklearn import tree
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
import pandas as pd

def crab_data(page_num):#构造 URL
    global data
    global all_data_list
    all_data_list = []
    #选取了 11 个区域构成 URL
    regions = ['gulou','jianye','qinhuai','xuanwu','yuhuatai','xixia','jiangning','pukou','liuhe','lishui','gaochun']
    for region in regions:
        #循环 100 页构造 URL
        for i in range(1, page_num):
            url = 'https://nj.lianjia.com/ershoufang/' + region + '/pg'+ str(i) + '/'
            time.sleep(0.3)
            try:
                html_analysis(url)
            except:
                print(region,i,"页，爬取失败")
    data = pd.DataFrame(all_data_list)

def store_data(output):#存储数据
    global all_data_list
```

```
        all_data_list.append(output)

    def html_analysis(url):#访问和 HTML 解析
            headers={
          'user-agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.61 Safari/537.36'
        }
            #采用 requests.get 访问 url
            r = requests.get(url, headers=headers)
            soup = BeautifulSoup(r.text, 'lxml')
            results = soup.find_all(attrs={'data-lj_action_source_type':'链家_PC_二
手列表页卡片'})
            for item in results:
                output = []
                # 从 url 中获得区域内容
                output.append(url.split('/')[-3])

                # 获得户型、面积、朝向等信息
                information = item.find('div', 'houseInfo').text.replace(' ', '').split('|')
                for i in range(7):
                    try:
                        output.append(information[i])
                    except:
                        output.append("Null")

                # 获得总价
                output.append(item.find('div', 'totalPrice').text)

                # 获得单价
                output.append(item.find('div', 'unitPrice').text)
                store_data(output)
    crab_data(101)

    data.to_json("estate_nanjing_data.json")#存储为 Json 文件
```

```
df = pd.read_json("estate_nanjing_data.json")#打开存储的 Json 文件，打开为 Dataframe 格式
df.columns = ['区域','房型','面积','朝向','装修','楼层','始建年份','房屋类型','总价','单价']#添加列标签
df=df[~df['房屋类型'].isin(['Null'])]#删除房屋类型是 NULL 的选项
pattern = re.compile("\d+")#正则表达式提取数字
df["单价"] = df["单价"].map(lambda x:int(re.findall(pattern,x)[0]))
df["总价"] = df["总价"].map(lambda x:int(re.findall(pattern,x)[0]))
#删除小样本量的房型/朝向/装修/楼层/房屋类型，即保留大样本特征
df=df[df['房型'].isin(["1 室 1 厅","2 室 1 厅","2 室 2 厅","3 室 1 厅","3 室 2 厅","4 室 2 厅"])]
df=df[df['朝向'].isin(["东","南","西","北","南北","东西"])]
df=df[df['装修'].isin(["毛坯","简装","精装"])]
df["楼层"] = df["楼层"].map(lambda x : x[0:3])
df=df[df['楼层'].isin(["中楼层","低楼层","高楼层"])]
df=df[df['房屋类型'].isin(["塔楼","板塔结合","板楼"])]
from pylab import mpl
mpl.rcParams['font.sans-serif']=['SimHei'] #用来指定默认字体 SimHei 为黑体
mpl.rcParams['axes.unicode_minus']=False
#南京市各地区平均房价单价可视化
region_price_list = df['单价'].groupby(df['区域']).mean().tolist()
region_list = df['单价'].groupby(df['区域']).mean().index.tolist()
plt.figure(figsize=(12,6))
font_set = FontProperties(fname=r"c:\windows\fonts\simsun.ttc", size=15)
plt.bar(region_list,region_price_list,color="#00CED1")
plt.title(u'南京市各地区平均房价单价',fontsize=20,fontproperties=font_set)
plt.xlabel(u'区域',fontsize=18,fontproperties=font_set)
plt.ylabel(u'房价单价',fontsize=18,fontproperties=font_set)
plt.tick_params(labelsize=14)
for a,b in zip(region_list,region_price_list):
    plt.text(a,b+70,"%.0f"%b,ha="center",va="bottom",fontsize=10)
plt.show()
#南京市各房型平均房价总价可视化
```

```
region_fangxing_list = df['总价'].groupby(df['房型']).mean().tolist()
region_list = df['总价'].groupby(df['房型']).mean().index.tolist()
plt.figure(figsize=(6,5))
font_set = FontProperties(fname=r"c:\windows\fonts\simsun.ttc", size=15)
plt.bar(region_list,region_fangxing_list,color="#00CED1")
plt.title(u'南京市各房型平均房价总价',fontsize=20,fontproperties=font_set)
plt.xlabel(u'房型',fontsize=18,fontproperties=font_set)
plt.ylabel(u'房价总价',fontsize=18,fontproperties=font_set)
plt.tick_params(labelsize=14)
for a,b in zip(region_list,region_fangxing_list):
    plt.text(a,b+2,"%.0f"%b,ha="center",va="bottom",fontsize=10)

plt.show()
#南京市各房型平均房价单价可视化
region_louceng_list = df['单价'].groupby(df['楼层']).mean().tolist()
region_list = df['单价'].groupby(df['楼层']).mean().index.tolist()
plt.figure(figsize=(6,6))
plt.pie(region_louceng_list,labels=region_list,autopct='%3.2f%%')
plt.title(u'南京市各房型平均房价单价',fontsize=20,fontproperties=font_set)
plt.xlabel(u'楼层',fontsize=18,fontproperties=font_set)
plt.ylabel(u'房价单价',fontsize=18,fontproperties=font_set)
plt.tick_params(labelsize=14)
plt.show()
#机器学习建模
def mean_price_category(value):
    if(value<10000):
        return 1
    if(value<20000):
        return 2
    if(value<30000):
        return 3
    if(value<40000):
        return 4
    if(value<50000):
```

```
            return 5
        if(value<60000):
            return 6
        if(value<70000):
            return 7
        if(value<80000):
            return 8
        else:
            return 9
    df["单价"] = df["单价"].map(lambda x:mean_price_category(x))
    #将多分类变量转换为数值型变量
    labels = df["区域"].unique().tolist()
    df["区域"] = df["区域"].apply(lambda x: labels.index(x))
    labels = df["房型"].unique().tolist()
    df["房型"] = df["房型"].apply(lambda x: labels.index(x))
    labels = df["朝向"].unique().tolist()
    df["朝向"] = df["朝向"].apply(lambda x: labels.index(x))
    labels = df["装修"].unique().tolist()
    df["装修"] = df["装修"].apply(lambda x: labels.index(x))
    labels = df["楼层"].unique().tolist()
    df["楼层"] = df["楼层"].apply(lambda x: labels.index(x))
    labels = df["房屋类型"].unique().tolist()
    df["房屋类型"] = df["房屋类型"].apply(lambda x: labels.index(x))
    df = df.drop("面积",axis=1).drop("始建年份",axis=1).drop("总价",axis=1)#删除不必要的列特征
    df_target = df["单价"]#选择目标
    df_features = df.drop("单价",axis=1)
    df_features = df_features.values
    print(df_features)
    Xtrain, Xtest, Ytrain, Ytest = train_test_split(df_features,df_target,test_size=0.3)#划分测试集训练集
    clf = tree.DecisionTreeClassifier(criterion="entropy")#构建模型
    clf = clf.fit(Xtrain, Ytrain)
    score = clf.score(Xtest, Ytest) #返回预测的准确度 accuracy
```

```
print(score)
feature_name = ['区域','房型','朝向','装修','楼层','房屋类型']#添加特征名
print(clf.feature_importances_)#特征重要性
print([*zip(feature_name,clf.feature_importances_)])#展现重要性
test = []
for i in range(10):#遍历不同的深度
    clf = tree.DecisionTreeClassifier(max_depth=i+1
                                      ,criterion="entropy"
                                      ,random_state=30
                                      ,splitter="random"
                                      )
    clf = clf.fit(Xtrain, Ytrain)
    score = clf.score(Xtest, Ytest)
    test.append(score)
plt.plot(range(1,11),test,color="red",label="max_depth")
plt.legend()
plt.show()
```