中南大學

CENTRAL SOUTH UNIVERSITY

本科毕业设计（论文）

GRADUATION DESIGN (THSIS)

题　　目:	基于核方法的深圳市房租预测分析
学生姓名:	李易航
指导老师:	刘源远
学　　院:	数学与统计学院
专业班级:	应数 1602

本科生院制

2020 年 6 月

基于核方法的深圳市房租预测分析

摘要

房屋租金的影响因素众多，同时在房客和房东之间往往存在着严重的信息不对称。因此，找到合理的切入点对房租价格进行合理预测有着深远意义。而核方法能够在将低维非线性数据转换为高维线性数据的同时避免高维计算，故本文选择以此为出发点进行探讨。

本文首先综述了核方法的理论背景，以核岭回归为例阐述了存在的计算瓶颈，并总结了相关解决方法。然后基于 Python 的爬虫框架 Scrapy，爬取了深圳市的房屋租赁数据。针对其中的分类变量做了编码、拆分、删除、缺失值填补等工作，并随后将相应数值型变量进行了 z-score 标准化，由此得到的总样本数为 22593 个，能用于模型的变量个数为 16 个。随后对获取到的文本数据进行了词云分析，也基于爬取到的经纬度信息对房租进行了地理可视化。

在模型阶段，本文采用了核岭回归和支持向量回归，选取 RBF 核函数，以 10 折交叉检验进行网格搜索，选择了最优参数组合。最后根据最优参数组合在训练数据集上重新训练模型，并在测试集上进行预测，计算得到各自的预测均方根误差分别为 0.0516 和 0.3241，而模型的决定系数分别为 0.9969 和 0.8949。其中支持向量回归表现相对较差的原因是对房租有较大值的那些点预测效果较差。

关键词： 房租　核方法　数据爬取　核岭回归　支持向量回归

Kernel Based Methods for House Rent Prediction in ShenZhen City

ABSTRACT

It is of much significance to accurately predict house rent because of too many determinants of house rent and asymmetric information between tenants and landlords. In the meanwhile, this thesis chooses to start discussion from the point of kernel methods, since it can transform the low-dimensional non-linear data into the high-dimensional linear data while avoiding the high-dimensional computation.

This thesis first summarizes the theoretical background of the kernel methods, takes the KRR as an example to explain the existing calculation bottlenecks and summarizes the related solutions. Then based on Scrapy, a web crawler framework of Python, the housing rental data of Shenzhen was crawled. Among the categorical variables, coding, splitting, deleting, missing value filling, etc. were done, and after processing, the corresponding numerical variables were z-score standardized. Finally the total number of samples are 22593, the number of variables are 16. Subsequently, a word cloud analysis was performed on the text data, and the rent was also geo-visualized based on the latitude and longitude information.

For modeling, this thesis tries KRR and SVR. RBF kernel function and grid search with 10-fold cross validation were selected to tune the optimal parameters. Finally, the model was retrained on the training data set according to the combination of optimal parameters and predictions were made on the testing data set. The calculated root-mean-square errors of the respective predictions are 0.0516 and 0.3241, respectively, and the coefficients of determination of the model are 0.9969 and 0.8949, respectively. The reason for the relatively poor performance of SVR is the poor prediction for those points with a large rent.

Key Words: Rent Kernel Methods Data Crawl KRR SVR

目录

第1章　绪论

1.1　研究背景与研究意义

自改革开放以来，我国的住房租赁市场蓬勃发展，对推动城镇化进程，改善城镇居民的住房条件发挥了重要作用。而近年来由于更多政策的支持和置业成本的提高，租房行业更是进一步踏入了高速发展的新时期。一方面，国务院办公厅于 2016 年首次提出“购租并举”后，我国的住房租赁市场站上了新的风口。另一方面，租金收益率越低，置业成本越高，上海易居房地产研究院于 2019 年一季度发布的《50 城租金收益率研究报告》显示，国内 50 个主要城市的住房租金收益率为 2.4%，环比下降 4%，同比下降 7%。

在这样的特定历史环境下，越来越多的人尤其是年轻人或主动或被动地选择了租房，而租房也正成为中国房地产行业发展的下半场。其中，作为改革开放造就的年轻城市——深圳，不可避免的成为了这个舞台上的主角之一。2019 年 8 月 30 日，深圳市住建局发布了《深圳市人民政府关于规范住房租赁市场稳定住房租赁价格的意见》，提出“力争到 2022 年新增建设筹集租赁住房不少于 30 万套”等目标。2019 年 6 月 25 日，租房平台自如和新华社共同发布的《中国青年租住生活蓝皮书》指出，深圳市租房人群中本科学历及以上超过半数，约 60% 的人愿意选择长租平台，近 40% 的 95 后愿意把 4 成以上的收入作为租金。

然而，租房市场快速扩张的同时带来了租赁行为的不规范，阻碍了租房市场的健康发展[1]。同时房屋租金的决定因素众多，在房客和房东之间存在着较为严重的信息不对称。这导致在外漂泊的年轻人发出“找房子比找工作还难”的感叹。因此，找到合理的切入点对住房租赁价格进行合理预测有着深远的意义。

1.2　研究现状

1.2.1　影响房租的因素

影响房租价格的因素众多，但总体上可分为宏观因素与微观因素。在宏观层面上，曹文聪[2] 基于格兰杰因果关系检验分析，对空置、人口、收入、通货膨胀、房价等影响深圳市住宅租金的因素进行了回归分析，并提出了对住宅租赁市场管理的建议。汪业辉[3] 通过借鉴西方国家的租赁市场状况，基于市场理论和计量分析方法从定性和定量的角度分析了影响国内住房租赁市场的相关因素：当地常住人口、当地住宅完工面积、当地人

均可支配收入、当地 GDP、当地住宅平均销售价格、当地住宅待售面积等。

由于宏观层面的因素所决定的房屋租赁价格往往也是宏观层面的市场平均价格，本文将更多地着眼于微观层面上的影响因素，从而利用相关因素进行预测得到微观层面上的价格，即单个房源的出租价格。而在微观层面的影响因素上，现有的研究多以特征价格模型作为出发点进行探讨[4]。张蕾和幸华[5] 在对深圳市福田区的城中村住房租金的研究中，从租约、建筑、邻里、区位四个特征维度出发构建特征价格指标体系并提取相关变量建立特征价格模型，分析出整租与合租、有无阳台、是否邻近马路、楼层的高低、房间数量的多少、距商业活动中心的远近等对其研究对象房租价格影响的定量关系。王洪强、李小雪和张英婕[6] 从租金价格的空间分异特征着手，对 1688 个上海市出租屋样本进行分析，得出卫生间数量、居室个数、房屋面积、交通位势、中心位势以及医疗配套等因素对住宅租金的影响状况，并提出住宅租金价格以中心城区为核心向四周递减，具有较高空间正相关性等结论。

1.2.2 现有的预测方法

张倩[7] 基于 Data Castle 的住房月租金数据集，利用特征工程进行数据的清理、特征选择构造等，构建了随机森林模型，对房租价格的走势提供预测。谢勇、项薇等[8] 在 Data Castle 提供的数据集基础上划分出不同的数据集，分别应用 GBDT、Xgboost 和 LightGBM 三种机器学习算法对租金进行了预测，同时探索出影响房屋租金的主要因素主要包括面积、商圈位置、距离地铁的距离、所在建筑的总楼层数和同一小区房源数量等。马涛和刘宁宁[9] 提出堆叠集成策略，通过集成学习融合随机森林、极端随机森林、LightGBM 三个基模型，提高了预测的稳定性和精度。

相较而言关注于房租预测的研究并不多，故本文也适当参考借鉴了对于房价预测的研究。Sabyasachi Basu 和 Thomas G. Thibodeau[10] 基于地理编码的 5000 个达拉斯房屋成交数据，使用半对数特征价格方法和球形自相关函数来检验房价的空间自相关性，最后利用广义最小二乘法估计出对应参数。Limsombunchai[11] 基于 200 个克莱斯特彻奇市 (新西兰城市) 的住房数据，分析比较了传统的特征价格模型和神经网络模型在预测房价上的差异。Chao Wu、Xinyue Ye 等[12] 从大数据的视角出发，基于二手房交易平台的房价数据和从社交平台获取的签到数据，利用特征价格方法和地理加权回归方法进行了房价预测。

至此，站在前人的肩膀上看问题，要想做好房屋出租价格的预测，一方面需要尽可

能多地考虑相关影响因素，如在构建预测模型时需要把相关地理信息等融合进去，另一方面还应从大量的数据出发，由数据来挖掘出结论。值得注意的是，相较于现有的预测模型，借助于核方法来进行相关房租预测的研究相对较少。并且从前面的分析我们知道，影响房租价格的因素错综复杂，并非简单的线性关系，而核方法能够在将低维非线性数据转换为高维线性数据的同时避免高维计算[①]，故本文将以此为出发点进行探讨。

1.3 本文结构

本文主要分为六章，可简述如下：

第 1 章：介绍了问题的背景，以及合理作出房租预测的意义。同时综述了影响房租因素相关的文献和现存的预测方法，回答了为什么选择核方法来进行预测。

第 2 章：系统综述核方法理论背景及相关研究。将阐明核方法的基础理论，应用核方法时存在的挑战以及相应解决的方法。

第 3 章：基于 Python 的 Scrapy 网络爬虫框架，从链家租房网爬取深圳市的租房数据。并对爬取到的数据进行诸如编码、拆分、标准化、缺失值填补等预处理。其中在处理缺失值时延伸出分类子问题。

第 4 章：以描述性统计分析为主，进行了诸如文本词云分析以及地理信息可视化等操作。

第 5 章：尝试基于核方法的模型：核岭回归和支持向量回归。其中根据交叉检验选择最优参数组合，并相应计算出模型在测试集上的预测均方根误差 $rmse$ 以及模型的决定系数 R^2。

第 6 章：总结了全文的研究内容，并对今后的研究工作做了一些简单的展望。

[①]Cover 定理[13] 保证了这一转换，而核函数的使用避免了高维计算

第2章　核方法理论背景及相关研究

核方法被广泛应用于各种数据分析方法中，这一名称的由来源自于使用核函数将低维非线性的数据转换为高维线性数据，而这一转换的过程也称为核技巧。接下来将在2.1节中具体阐述核函数的理论基础：相关的性质和定义以及核函数的特征函数。而在2.2节中将给出再生核希尔伯特空间的定义以及 Moore-Aronszajn 定理。

同时如1.2.2节末尾所述，要想做好房屋出租价格的预测，应从大量的数据出发，然而 Hisashi, Tsuyoshi 等人[14] 在研究中提到，直接使用核方法，其计算的时间复杂度是训练样本数量的立方级的，如此在处理实际问题时会面临极大的挑战。这一问题将在2.3节中具体阐述。而在2.4节中将阐述对于核化监督学习的求解方法的相关研究。最后在2.5节中将叙述用于大规模核机器学习的随机特征。

2.1　核函数理论基础

核函数源自于积分算子理论[15]，算子 T_k 定义为：

$$\left(T_k f\right)(\mathrm{x}) = \int_{\mathcal{X}} k\left(\mathrm{x}, \mathrm{x}'\right) f\left(\mathrm{x}'\right) d\mu\left(\mathrm{x}'\right) \tag{2-1}$$

称二元函数 $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ 为核函数。而 Shawe, John 和 Nello 在合著的书[16] 中进一步将核函数定义为 $k\left(\mathrm{x}, \mathrm{x}'\right) = \langle \phi(\mathrm{x}), \phi(\mathrm{x}') \rangle$，其中 ϕ 是从 $\mathcal{X}$ 到 (内积) 特征空间 $\mathcal{X}'$ 的映射[①]。这一内积形式也是现在所广泛使用的。

2.1.1　相关的性质和定义

称一个实值的核函数是对称的，若 $k\left(\mathrm{x}, \mathrm{x}'\right) = k\left(\mathrm{x}', \mathrm{x}\right)$。

从核函数出发，我们可以定义核矩阵 K：给定输入点集 $\{\mathrm{x}_i | i = 1, ..., n\}$，核矩阵 K 的元素可以定义为 $K_{ij} = k\left(\mathrm{x}_i, \mathrm{x}_j\right)$，即核函数的对应取值。

一个实值 $n \times n$ 对称矩阵 K，若对所有向量 $\mathrm{v} \in \mathbb{R}^n$ 均有 $Q(\mathrm{v}) = \mathrm{v}^T K \mathrm{v} \geq 0$，则称 K 是半正定的。若仅当 $\mathrm{v} = 0$ 时 $Q(\mathrm{v}) = 0$ 成立，则称 K 是正定的，$Q(\mathrm{v})$ 相应称为二次型。由一般的核函数所定义的核矩阵不一定是半正定的，但是由对称核函数定义的核矩阵一定是半正定的。

[①]注意到 k 是 x 和 x′ 的函数，通过选取适当的 k，可以避免高维空间中 ϕ 的计算。

称核函数 $k(\mathrm{x},\mathrm{x}')$ 为对称半正定的[17]，若 $k(\mathrm{x},\mathrm{x}')=k(\mathrm{x}',\mathrm{x})$ 且

$$\int k(\mathrm{x},\mathrm{x}')\,f(\mathrm{x})f(\mathrm{x}')\,d\mu(\mathrm{x})d\mu(\mathrm{x}')\geq 0 \tag{2-2}$$

对任意 $f\in L_2(\mathcal{X},\mu)$ 均成立。

2.1.2 核函数的特征函数

称满足积分方程

$$\int k(\mathrm{x},\mathrm{x}')\,\phi(\mathrm{x})d\mu(\mathrm{x})=\lambda\phi(\mathrm{x}') \tag{2-3}$$

的函数 $\phi(\cdot)$ 为核 k 的对应于特征值 λ 和测度 μ 的特征函数。其中，我们感兴趣的测度 μ 主要有两类：(i) $\mathbb{R}^n$ 的一个紧子集 $\mathcal{C}$ 上的勒贝格测度；(ii) 当 x 服从某一分布时，相应于概率密度函数 $p(\mathrm{x})$ 可将 $d\mu(\mathrm{x})$ 写为 $p(\mathrm{x})d\mathrm{x}$。

通常，特征函数的数量是无限的：$\phi_1(\mathrm{x}),\phi_2(\mathrm{x}),\ldots$，我们假定特征值按降序排列：$\lambda_1\geq\lambda_2\geq\ldots$。特征函数相对于 μ 是正交的并且可以被规范化以使得 $\int\phi_i(\mathrm{x})\phi_j(\mathrm{x})d\mu(\mathrm{x})=\delta_{ij}$，其中当 $i=j$ 时，δ_{ij} 为 1，否则为 0。

Mercer 定理[18] 让我们能够将核 k 表示成特征值和特征函数的形式：

定理 2.1 (Mercer 定理) 令 $(\mathcal{X},\mu)$ 为有限测度空间，$k\in L_\infty(\mathcal{X}^2,\mu^2)$ 为使得 $T_k: L_2(\mathcal{X},\mu)\to L_2(\mathcal{X},\mu)$ 正定的 (见方程2-2) 核。令 $\phi_i\in L_2(\mathcal{X},\mu)$ 为相应于特征值 $\lambda_i>0$ 的 T_k 的规范化的特征函数。则：

1. 特征函数 $\lambda_{i\,i=1}^{\infty}$ 是绝对可和的；

2. $$k(\mathrm{x},\mathrm{x}')=\sum_{i=1}^{\infty}\lambda_i\phi_i(\mathrm{x})\phi_i^*(\mathrm{x}') \tag{2-4}$$

   μ^2 几乎处处成立，即右端序列 μ^2 几乎处处绝对一致收敛，

2.2 再生核希尔伯特空间

再生核希尔伯特空间 (RKHS) 在统计学习理论中有着举足轻重的地位，这是由 Representer 定理[19] 所保证的。该定理指出，RKHS 中的每个能最小化经验风险函数的元素能被表示成在训练样本处的核函数的线性组合，这一结果将无穷维的经验风险最小化问题转化为了有限维的最优化问题。对再生核希尔伯特空间的正式定义如下：

定义 2.1 [20] (再生核希尔伯特空间) 令 $\mathcal{H}$ 为索引集 $\mathcal{X}$ 上的实值函数的希尔伯特空间，称 $\mathcal{H}$ 为带有内积 $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ ($\|f\|_{\mathcal{H}}=\sqrt{\langle f,f\rangle_{\mathcal{H}}}$) 的再生核希尔伯特空间，若存在一个函数 $k:\mathcal{X}\times\mathcal{X}\to\mathbb{R}$ 满足如下条件：

1. 对于所有 x，$k(\mathrm{x},\mathrm{x}')$ 作为 x' 的函数属于 $\mathcal{H}$;

2. k 具有再生性质，即 $\langle f(\cdot),k(\cdot,\mathrm{x})\rangle_{\mathcal{H}}=f(\mathrm{x})$.

注意到 $k(\mathrm{x},\cdot)$ 和 $k(\mathrm{x}',\cdot)$ 在 $\mathcal{H}$ 中，我们有 $\langle k(\mathrm{x},\cdot),k(\mathrm{x}',\cdot)\rangle_{\mathcal{H}}=k(\mathrm{x},\mathrm{x}')$

Aronszajn 于 1950 年证明了再生核希尔伯特空间与核函数 k 之间是相互唯一确定的：

定理 2.2 [21] (Moore-Aronszajn 定理) 令 $\mathcal{X}$ 为索引集，则对 $\mathcal{X}\times\mathcal{X}$ 上的每一正定函数 $k(\cdot,\cdot)$，都有唯一的再生核希尔伯特空间，反之亦然。

从 Mercer 定理的角度，RKHS 的抽象形式的好处是在不同测度下特征基底会改变，但是 RKHS 范数会保持不变，它仅仅是核 k 的一个性质。

从再生核映射构造的角度，考虑关于函数 f 的空间，其中 f 定义为：

$$\left\{f(\mathrm{x})=\sum_{i=1}^{n}\alpha_i k(\mathrm{x},\mathrm{x}_i):n\in\mathbb{N},\mathrm{x}_i\in\mathcal{X},\alpha_i\in\mathbb{R}\right\}$$

令 $g(\mathrm{x})=\sum_{j=1}^{n'}\alpha'_j k\left(\mathrm{x},\mathrm{x}'_j\right)$，并定义内积 $\langle f,g\rangle_{\mathcal{H}}=\sum_{i=1}^{n}\sum_{j=1}^{n'}\alpha_i\alpha'_j k\left(\mathrm{x}_i,\mathrm{x}'_j\right)$。显然，定义2.1的条件 1 在再生核映射构造下是成立的。而条件 2 也成立：$\langle k(\cdot,\mathrm{x}),f(\cdot)\rangle_{\mathcal{H}}=\sum_{i=1}^{n}\alpha_i k(\mathrm{x},\mathrm{x}_i)=f(\mathrm{x})$。

2.3 核方法的计算瓶颈

考虑线性参数模型：

$$f(\boldsymbol{x};\boldsymbol{w})\equiv\langle\boldsymbol{w},\boldsymbol{x}\rangle \tag{2-5}$$

其中，$\mathrm{x}\in\mathbb{R}^d$ 是输入变量，$\mathrm{w}\in\mathbb{R}^d$ 是参数向量，$\langle\cdot,\cdot\rangle$ 表示内积。

给定训练数据 $\{(\mathrm{x}_i,y_i)|\mathrm{x_i}\in\mathbb{R}^d,y_i\in\mathbb{R}\}_{i=1}^{n}$，为了确定参数 w，需要最小化经验风险项和正则项的线性和：$J(\mathrm{w})\equiv R_{emp}(\mathrm{w})+\lambda\Omega(\mathrm{w})$，其中 $\lambda>0$ 为正则化参数。

以岭回归[22] 为例，使用平方损失函数来度量经验风险，l_2 范数作为正则项：

$$R_{emp}(\mathrm{w})\equiv\sum_{i=1}^{n}(y_i-f(\mathrm{x_i};\mathrm{w}))^2 \tag{2-6}$$

$$\Omega(\mathrm{w}) \equiv ||\mathrm{w}||_2^2 \tag{2-7}$$

输入样本构成的数据矩阵记作：$\mathrm{X} = (\mathrm{x}_1|\mathrm{x}_2|...|\mathrm{x}_n)$，则岭回归的解析解为：

$$\widehat{\mathrm{w}} = (\mathrm{X}^T\mathrm{X} + \lambda\mathrm{I})^{-1}\mathrm{X}^T\mathrm{y} \tag{2-8}$$

等式 (2-8) 表明岭回归求解的计算复杂度为 $O(d^3)$，依赖于输入空间的维数。

根据 Representer 定理[19]，假设参数 w 可以写成训练样本的线性组合：

$$\mathrm{w} \equiv \sum_{i=1}^{n} \alpha_i \mathrm{x_i} = \mathrm{X}^T\alpha \tag{2-9}$$

则等式 (2-5) 可以表示成：

$$f(\mathrm{x};\alpha) \equiv \sum_{i=1}^{n} \alpha_i \langle \mathrm{x}_i, \mathrm{x}\rangle \tag{2-10}$$

从而定义核函数为：$k(\mathrm{x},\mathrm{x}') \equiv \langle \mathrm{x},\mathrm{x}'\rangle$，而参数模型可以表示为：$f(\mathrm{x}) = \sum_{i=1}^{n} \alpha_i k(\mathrm{x}_i,\mathrm{x})$.

现在考虑核岭回归 (KRR)[23]，基于核化的模型，岭回归的目标函数可以重写为：

$$J(\alpha) = ||\mathrm{y} - \mathrm{K}\alpha||^2 + \lambda\alpha^T\mathrm{K}\alpha \tag{2-11}$$

其中 K 是 $n \times n$ 的核矩阵。可解得：

$$\widehat{\alpha} = (\mathrm{K} + \lambda\mathrm{I})^{-1}\mathrm{y} \tag{2-12}$$

等式 (2-12) 表明核岭回归的计算复杂度为 $O(dn^3)$，其中 d 来自于计算核函数的值，n^3 来自于计算核矩阵的逆。事实上，有很多核函数的计算复杂度是独立于 d 的，如多项式核：$k(\mathrm{x},\mathrm{x}') = (<\mathrm{x},\mathrm{x}'>+1)^c$；高斯核：$k(\mathrm{x},\mathrm{x}') = e^{-||\mathrm{x}-\mathrm{x}'||^2/\sigma^2}$ 等。因此，核岭回归的计算复杂度主要由训练样本的数量决定。

2.4 在监督学习方法中的改进

2.4.1 核岭回归

核岭回归 (见2.3中所述) 是一种广泛使用的核化的回归技巧。从等式 (2-12) 知道，解 KRR 即相当于求解线性方程：

$$(\mathrm{K} + \lambda\mathrm{I}_n)\alpha = \mathrm{y} \tag{2-13}$$

通常的方法是使用 Cholesky 分解，计算复杂度为 $O(n^3)$，而若使用共轭梯度法 (CG 方法)，计算复杂度会变为为 $O(rn^2)$ ，这里 r 是 CG 迭代次数。进一步，若核函数是高斯核，CG 方法的计算复杂度会降至 $O(n)$。

2.4.2 核偏最小二乘

偏最小二乘 (PLS)[24] 是一类启发式的迭代算法，Roman, Leonard 在此基础上提出了核偏最小二乘 (KPLS)[25]。尽管 PLS 通常没有显式的目标函数，Kramer 等人[26] 证明了 KPLS 和普通最小二乘一样最小化相同的目标函数，只是解被限制在 Krylov 子空间：由 $Ky, K^2y, ..., K^ry$，张成的子空间，其中 r 是给定的整数，表示 PLS 的分量个数。并且其解可以通过求解 $K^2\alpha = Ky$ 获得。

在大多已知的 KPLS 算法中，计算复杂度被矩阵向量乘积所主导。尽管原始的实现方式要求 $O(n^2)$ 的计算复杂度，但当选择高斯核时，快速高斯变换 (FGT)[27] 能将计算复杂度减少至线性时间。

2.4.3 核 LASSO

LASSO[28] 是一种对线性回归进行压缩和选择的方法。与岭回归类似，不同之处在于它选择用 l_1 正则化来最小化目标函数：

$$J_{\text{lasso}}(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \langle \boldsymbol{x}_i, \boldsymbol{w}\rangle)^2 + \lambda\|\boldsymbol{w}\|_1 \tag{2-14}$$

其中 $||\boldsymbol{w}||_1 = \sum_{i=1}^{d}|w_i|$。$l_1$ 正则化使得 LASSO 的解是稀疏的。

Volker[29] [30] 基于稀疏的核回归器提出了核化的 LASSO 回归 (KLASSO)，其使用 $D = \{k_\gamma(\mathrm{x}, \mathrm{x}_1), ..., k_\gamma(\mathrm{x}, \mathrm{x}_n)\}$ 作为基函数来把 LASSO 扩展到非线性回归，其中 $k_\gamma(\mathrm{x}, \mathrm{x}_i)$ 如2.2节中再生核映射构造里的带有超参数 γ 的核函数。从而要最小化的目标函数相应为：

$$J_{\text{klasso}}(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \langle k_\gamma(\mathrm{x}, \mathrm{x}_i), \boldsymbol{w}\rangle)^2 + \lambda\|\boldsymbol{w}\|_1 \tag{2-15}$$

Gang Wang，Dit-Yan Yeung，Frederick[31] 提出了核路径算法用于求解上述核化的 LASSO。

2.4.4 支持向量回归

支持向量回归 (SVR)[32] 从使用核技巧的支持向量机扩展而来，可以形式化为最小化如下目标函数：

$$J_{\text{SVR}}(\boldsymbol{w}) = \frac{C}{n}\sum_{i=1}^{n} l_\epsilon(y_i - \langle \boldsymbol{x}_i, \boldsymbol{w}\rangle) + \frac{1}{2}\|\boldsymbol{w}\|_2^2 \tag{2-16}$$

其中，l_ϵ 是 SVR 的关键所在，称为 ϵ-不灵敏损失函数：

$$l_\epsilon(\eta) \equiv \begin{cases} 0 & 若|\eta| < \epsilon \\ |\eta| - \epsilon & 其他 \end{cases} \tag{2-17}$$

一个获取 SVR 解的标准方法是求解从 (2-16) 的对偶问题得到的二次规划问题。Teo，Choon 和 Smola 等人提出的切平面算法[33] 可以更为有效地求解该问题。

2.5 用于大规模核机器学习的随机特征

至此通过前面的分析，我们知道对于基于核方法的相关学习方法，当数据量很大时所需的训练时间会很大。因此，计算上的改进至关重要。在2.4节中提到了在部分监督学习方法中的改进，而 Rahimi 和 Benjamin[34] 基于平移不变核，提出来直接分解核函数本身，这样的分解不依赖于数据，并且通过将数据映射到相对低维的随机特征空间中使得对核机器的训练和评估转换为对线性机器的相应操作。并通过实验表明，随机特征与非常简单的线性学习技术相结合，可以有力地与基于核方法的最新分类和回归算法在速度和精确性上竞争，包括那些分解核矩阵的方法。

具体而言，他们提出了两类随机特征的构造。一类是随机傅立叶特征，另一类是随机分箱特征，都使得其内积一致的近似于许多常用的核。前者适合于插值问题，后者适合于用来估计依赖于点对间 L_1 距离的核。

第3章 数据

3.1 数据获取

正如1.2.2节末尾所述，要想作出良好的预测，就需要从大量的数据出发。因此本文基于 Python 的 Scrapy 网络爬虫框架，从链家租房网站爬取了深圳市的房屋出租信息。接下来将在3.1.1节中简单介绍 Scrapy 框架，并在3.1.2节中说明爬取数据的流程及遇到的困难，最后在3.1.3节中给出爬取到的数据概览。

3.1.1 Scrapy 框架简介

Scrapy 是一种用于大型网页爬取的 Python 框架。它提供了从网页有效爬取数据的基础工具，同时我们还可以按照我们偏好的结构和格式来存储爬取到的数据。图3-1为 Scrapy 的工作流程，摘自官方文档[①]。其中各个序号分别表示：1. 爬虫向主程序获取初始请求；2. 主程序请求调度程序并预备下次的爬取；3. 调度程序返回下一个请求给主程序；4. 主程序发送请求到下载器获取网络数据；5. 下载器返回下载结果给主程序；6. 主程序将该结果通过中间件返还给爬虫；7. 爬虫开始处理响应，通过中间件返回处理后的 items 与新的请求给主程序；8. 主程序发送处理后的 items 给项目管道，并将处理结果返回给调度器，调度器开始计划处理下一个请求的抓取。在进行完这一流程之后，不断重复该过程，直到爬取完所有的 url 请求。

3.1.2 爬取流程简述

首先进入链家租房网的首页，如图2(a)可以看到有约 4 万多套房源的信息。然而，如图2(b)所示，可获取的总共只有 100 个页面的索引，而每个页面只有 32 个房源，也就是说，若从首页直接开始爬取，我们最多能够获得 3200 个房源信息，这是远小于 4 万的。解决该问题的方法是，分区域进行爬取，如图2(c)。但是这里显示的房源数仍然大于可获取的房源数，故应进一步细分，以商圈为基础单位来爬取每一个房源的 url 并进行解析，如图2(d)。

按照这样的思路，我们以每个商圈的起始页面作为初始 url，从第一个初始 url 开始获取每一页的 url，再根据每一页的 url，解析获得相应每个房源的 url，最后解析并提取房源信息，将其存储在预先设定好的 Mysql 数据库中。

[①]https://docs.scrapy.org/en/latest/topics/architecture.html

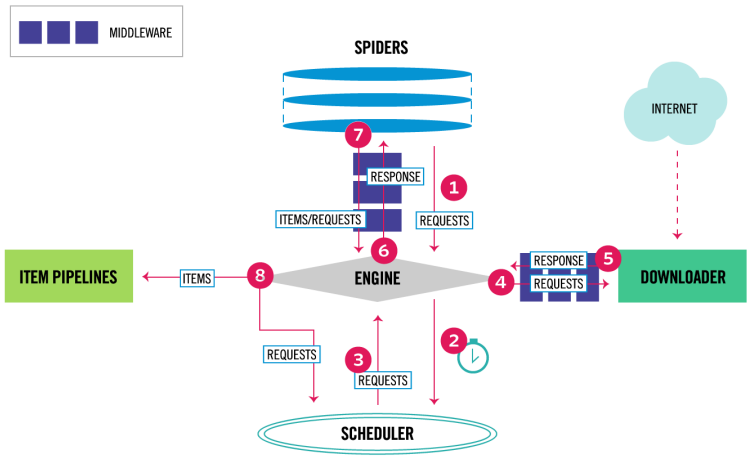
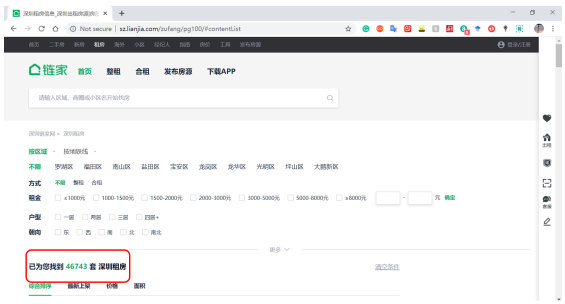
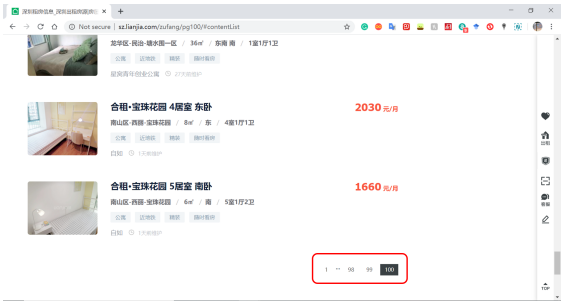


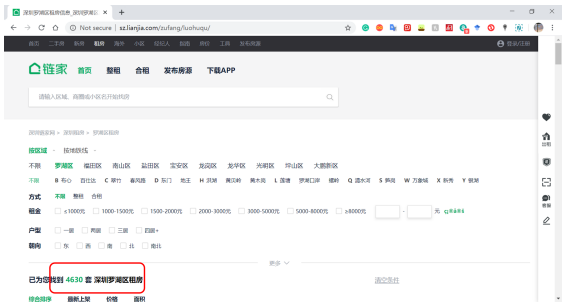
图 3-1　Scrapy 工作流程。



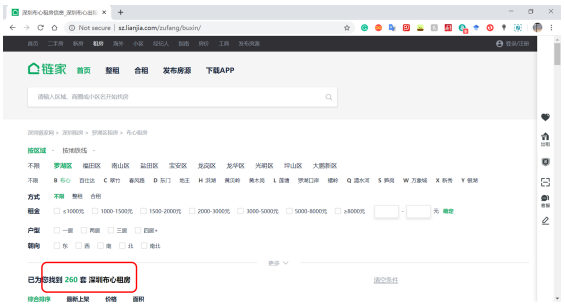
(a) 房源总数



(b) 可获取页面数



(c) 分区域信息

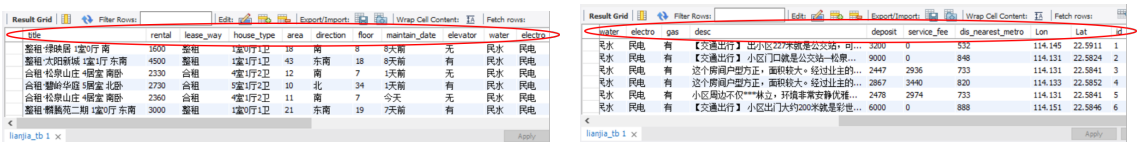


(d) 分商圈信息

图 3-2　房源信息爬取分析

3.1.3 数据概览

在前述爬取数据的过程中，存在部分网页访问失败的现象，以及因多次暂停后进行断点重爬，导致有部分房源信息重复等问题出现。因此，在删除重复数据之后，我们爬取到的有效数据量为 23643 个。图3-3给出了爬取后保存在 Mysql 数据库中的数据概览。



(a) 第一部分　　(b) 第二部分

图 3-3　爬取后的数据概览

其中，各数据项含义如表3-1所示：

3.2 数据预处理

3.2.1 分类变量预处理

因为在后续建模时，分类变量难以直接代入模型当中，故我们应先对其进行相应的预处理。而需要处理的分类变量有 lease_way(房源出租类型)、house_type(户型)、direction(朝向)、maintain_date(最近一次维护时间)、elevator(有无电梯)、gas(有无燃气)、water/electro(用水/电类型)。

对于 lease_way(房源出租类型)，它的取值仅有‘整租’和‘合租’两个，如图3-4所示，故采取二进制编码将‘整租’编码为 1，‘合租’编码为 0，并记新得到的变量为 lease_way_cat.

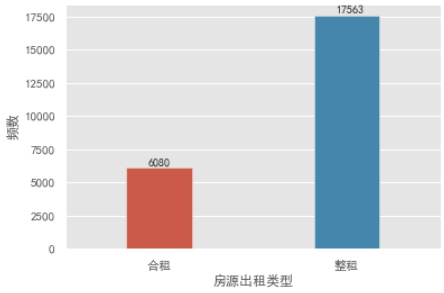


图 3-4　房源出租类型

表 3-1　数据解释说明

变量名	解释
title	房源名称
desc	房源描述
rental	房源月租价格 (元/月)，数值型变量
lease_way	房源出租类型 (整租、合租)，分类变量
house_type	户型 (X 室 X 卫 X 厅)，分类变量
direction	房源朝向，分类变量
area	房源面积 (平方米)，数值型变量
floor	房源所在楼层，数值型变量
maintain_date	最近一次维护时间
elevator/gas	有无电梯/燃气，分类变量
water/electro	用水/电类型 (如：民水/民电)，分类变量
deposit/service_fee	押金/服务费，数值型变量
dis_nearest_metro	据最近地铁站距离，数值型变量
Lon, Lat	经度，纬度，数值型变量

对于 house_type(户型)，有 12 个含有“未知”字样，因其数量相对于样本总体而言可以忽略不记，故删去这十二个对应的房源数据，我们的样本总数还有 23631 个。通过探索数据知道，它有 139 种取值。通过将所有频数小于 1000 的类别归为同一类（‘其他’），得到户型的频数如图3-5所示。可以看到，‘2 室 1 厅 1 卫’和‘1 室 1 厅 1 卫’这两种房型远远多于其他的房型。这里对于该变量 (X 室 X 卫 X 厅) 的处理思路是：将其拆分为三个变量 room、toilet、parlor，它们分别对应于卧室，卫生间和客厅的数量。

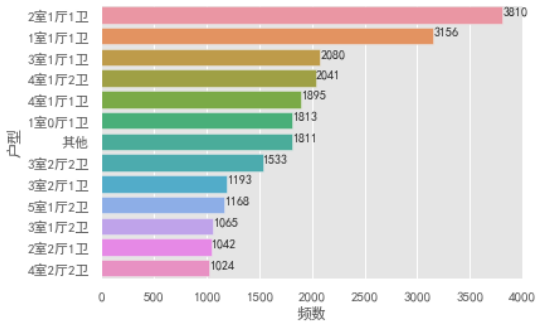


图 3-5 房型频数统计

对于 direction(朝向)，往往在同一小区其对房租价格会稍有影响，而整体来说，在不同区域其对于房租价格往往不起决定性因素。鉴于我们在爬取数据时并未爬取房源的小区信息，同时为了避免后续模型过于复杂，我们选择舍去该变量。而 maintain_date(最近一次维护时间) 同理也舍去。

对于 elevator(有无电梯)、gas(有无燃气)、water(用水类型) 和 electro(用电类型)，在探索数据时发现它们均有三种取值，这是不正常的，因为按定义应当只有两种取值 (前两者为‘有’和‘无’，后两者为‘民水/电’和‘商水/电’)。进一步探索如表3-2所示。鉴于此处缺失的数量偏多，故将它们编码为 NaN(缺失值)，以待后续做缺失值的处理。同时采取二进制编码将‘有’、‘民水’、‘民电’编码为 1，‘无’、‘商水’、‘商电’编码为 0。

至此，这一小节中我们对部分分类变量做了编码处理，对部分分类变量做了拆分处理，也删去了部分分类变量。就前两者而言，在处理后进一步可视作数值型变量，以用于后续建模。而本节 (3.2节) 后续部分则均为数值型变量的预处理。

表 3-2 取值频数统计

	elevator	gas	water	electro
有/有/民水/民电	18667	21570	20907	20983
无/无/商水/商电	4656	1516	1824	1757
暂无数据	308	545	900	891

3.2.2 样本数据标准化

经过之前的分类变量数据预处理，我们所有的数值型变量分别为：rental(房租价格)、area(房源面积)、floor(楼层)、deposit(押金)、service_fee(服务费)、dis_nearest_metro(距最近地铁站距离)、Lon(经度)、Lat(纬度)、lease_way_cat(出租类型)、room(房间数)、parlor(客厅数)、toilet(卫生间数)、elevator(有无电梯)、gas(有无燃气)、water(用水类型)、electro(用电类型)。这里，rental 为因变量，即我们的预测目标，其余则为自变量，总个数为 $d = 15$。而除了后四者有不同数量的缺失值之外，其他所有变量的样本数均为 $n = 23631$。

由于各变量的量纲不尽相同，取值范围大小不一，同时很多机器学习算法都要求使用标准化的数据 (例如使用高斯核函数的相关算法、使用 l_1，l_2 正则化的算法等)，因此我们需要将上述数据进行标准化处理。

而样本数据标准化有多种选择，如 0-1 标准化 ($\frac{x-x_{min}}{x_{max}-x_{min}}$) 或者 z-score 标准化 ($\frac{x-u}{s}$，其中 u s 分别为样本均值和标准差)。这里我们选择后者，即 z-score 标准化。而具体在实施时，我们是针对训练数据集和测试数据集分别进行的。

3.2.3 缺失值处理延申的分类子问题

如表3-2所示，我们有四个变量需要处理缺失值问题：elevator(308 个缺失值)、gas(545 个缺失值)、water(900 个缺失值)、electro(891 个缺失值)。而基于这四个变量都是分类变量且仅有两个取值，一种可行的思路是延伸出二分类子问题：考虑将这四个变量分别作为因变量，用上述其他数值型变量作为自变量，并使用未缺失的数据来训练分类模型，最后用训练好的分类模型来预测缺失的取值。

这里以 elevator(有无电梯) 为例进行阐述。我们选择训练带正则项的 Logistic Regression 分类器，以 0-1 损失得分来作为我们的评估标准[①]。

[①]0-1 损失得分定义为：$1-\frac{1}{n}\sum_{i=1}^{n}\mathcal{I}(\hat{y}_i \neq y_i)$，其中 $\mathcal{I}$ 为指示函数 (满足条件为 1，否则为 0)

表 3-3 缺失值处理的模型参数

	训练时间	正则化选项	C	0-1 损失得分
elevator	10.648s	l_1	0.00215	0.940
gas	9.034s	l_2	0.00001	0.933
water	9.103s	l_2	0.00001	0.920
electro	9.462s	l_2	0.00001	0.918

从最优化问题的视角，二分类正则化 Logistic Regression 最小化如下目标函数：

$$\min_{w,C} \sum_{i=1}^{n} \log\left(\exp\left(-y_i\left(X_i^T w\right)\right)+1\right)+\frac{1}{C}||w||_p \tag{3-1}$$

其中，w 为待估参数，$\frac{1}{C}$ 为正则化参数，当 C 很大时可近似看作没有正则化。此外，当 $p=1$ 时为 l_1 正则化；当 $p=2$ 时为 l_2 正则化。值得注意的是，在该二分类问题中，要求 $y_i=\{1,-1\}$，因此就我们的数据而言，需要先将 0 转换成-1，最后分类完毕再转换成 0。

具体来说，我们考虑 elevator(有无电梯) 所有未缺失的数据，随机将其中 70% 作为训练数据集，剩余的 30% 作为测试数据集。选取 0-1 损失得分作为评估标准，给定 p 和 C 的一组初始值 (其中 p 仅为 l_1 和 l_2 两个取值，而 C 的初始取值范围可适当调整)，在训练集上以 10 折交叉检验进行网格搜索选择最优参数 p 和 C，从而得到最优模型。最后再在测试集上进行评估，以检验所得分类器的性能。最后利用该模型来对缺失的部分进行分类，从而填补缺失值。如表3-3所示为 elevator(有无电梯) 和其他三个变量所对应的模型参数。

需要注意的是，elevator(有无电梯) 对应的模型中，由于为 l_1 正则化，得到的参数估计 $w=[0,0,0.98174608,0,0,0,0,0,0,0,0,0]$ 是稀疏的，其中仅第三项 (floor, 楼层数) 不为零，其余皆为零。而这恰好也对应了楼层和电梯有无之间的相关关系，因为往往楼层越高越有可能有电梯。

而其他三个变量，无论参数 C 的初始取值范围如何改变，在模型训练完之后总是以最小的那一个为最优 (如表3-3中均取到 0.00001，是因为通过调整的 C 的初始化最小取值为该值)。而 C 越小，正则化参数越大，对模型的压缩也越大，最后所得到的参数估计 ω 越接近于 0。因此，即使 0-1 损失得分很高，用其他变量来分别对这三个变量

分类也是不恰当的，从而我们只能选择删去这三者的缺失值所对应的房源数据。在删去后，我们的样本数变为 $n = 22593$。

第4章　描述性统计分析

4.1　词云分析

随着文本形式的非结构化数据与日俱增，对其进行分析的需求也越来越大。而一个用来帮助理解文本且快速识别出文本中重要信息的工具便是词云。所谓词云，其实就是文本数据中词频的可视化表示。通常来说，当一个词语在文本中出现的频率越高，其在生成的词云图片中的字体大小也越大。在本节中，我们基于3.2节中处理好的数据所对应的 desc(房源描述) 变量，利用 Python 来生成词云，以可视化分析所有样本的房源描述中哪些词语出现的频率最高。总的来说，词云分析主要可概括为三个步骤，即文本分词、词频统计、以及词云生成。以下将进行详细说明。

4.1.1　分词与词频统计

首先通过检索 desc(房源描述) 所对应的数据项信息，发现有 7521 个数据对应取值为‘NAN’，故在去除缺失值后以空格作为连接符将所有 desc(房源描述) 信息记录在单独的一个变量‘text’中。接下来在分词阶段，基于 Python 的中文分词组件‘jieba’，采用精确模式将‘text’进行分词，共得到 1600164 个词语。而其中包含了大量的无意义的虚词，因此需要进一步处理。

这里利用 Python 的字典类型基础数据结构，进行词频统计。如表4-1所示，首先统计出词频前二十的词语，发现存在大量虚词，如‘;’，‘。’，‘的’等。将它们删除后再次统计出词频前二十的词语，删除新出现的虚词。如此反复统计并删除多次直到词频前二十均无虚词为止。

4.1.2　词云生成

接下来基于 Python 的 wordcloud，通过设定相关参数，利用处理好的文本信息，生成词云如图4-1所示。从中我们可以直观的看出，房源信息中出现频率最高的词包括‘周边’，‘配套’，‘交通’，‘出行’，‘户型’等。而这也从侧面反映了人们的租房需求中所关心的信息，这些分词所对应的相关信息也很可能反映在最后房租的高低中。

表 4-1 房源描述的词频统计

统计次数	词频前二十词语及其频数
1	[(‘,’, 170245), (‘、’, 49236), (‘。’, 42094), (‘的’, 31523), (‘小区’, 29317), (‘】’, 25504), (‘【’, 25497), (‘:’, 16726), (‘有’, 15488), (‘配套’, 13110), (‘是’, 12313), (‘交通’, 11703), (‘好’, 11370), (‘路’, 10623), (‘介绍’, 9354), (‘出行’, 9219), (‘周边’, 8681), (‘米’, 8019), (‘等’, 7469), (‘和’, 7363)]
2	[(‘小区’, 29317), (‘配套’, 13110), (‘交通’, 11703), (‘介绍’, 9354), (‘出行’, 9219), (‘周边’, 8681), (‘号线’, 7094), (‘生活’, 7029), (‘方便’, 6457), (‘公园’, 6434), (‘装修’, 6212), (‘;’, 6179), (‘!’, 5995), (‘户型’, 5987), (‘都’, 5974), (‘地铁’, 5879), (‘采光’, 5846), (‘站’, 5556), (‘1’, 5519), (‘适合’, 5395)]
3	[(‘小区’, 29317), (‘配套’, 13110), (‘交通’, 11703), (‘介绍’, 9354), (‘出行’, 9219), (‘周边’, 8681), (‘号线’, 7094), (‘生活’, 7029), (‘方便’, 6457), (‘公园’, 6434), (‘装修’, 6212), (‘户型’, 5987), (‘地铁’, 5879), (‘采光’, 5846), (‘居住’, 5274), (‘便利’, 5227), (‘为’, 5116), (‘2’, 5088), (‘医院’, 5065), (‘3’, 5060)]
4	[(‘小区’, 29317), (‘配套’, 13110), (‘交通’, 11703), (‘出行’, 9219), (‘周边’, 8681), (‘号线’, 7094), (‘生活’, 7029), (‘方便’, 6457), (‘公园’, 6434), (‘装修’, 6212), (‘户型’, 5987), (‘地铁’, 5879), (‘采光’, 5846), (‘居住’, 5274), (‘便利’, 5227), (‘医院’, 5065), (‘房源’, 4901), (‘)’, 4883), (‘距离’, 4632), (‘您’, 4625)]
5	[(‘小区’, 29317), (‘配套’, 13110), (‘交通’, 11703), (‘出行’, 9219), (‘周边’, 8681), (‘号线’, 7094), (‘生活’, 7029), (‘方便’, 6457), (‘公园’, 6434), (‘装修’, 6212), (‘户型’, 5987), (‘地铁’, 5879), (‘采光’, 5846), (‘居住’, 5274), (‘便利’, 5227), (‘医院’, 5065), (‘距离’, 4632), (‘地铁站’, 4552), (‘舒适’, 4482), (‘超市’, 4283)]

图 4-1　房源信息词云可视化

4.2　**地理信息可视化**

4.2.1　房租总览

我们在第3.1节中爬取信息时，获取了每个房源的经纬度信息‘Lon’和‘Lat’。基于此，我们尝试将每个房源对应的房租价格可视化到地图上，以直观地感受房租相对于地理区位的分布。如图4-2所示，直接从地图上看并不能明显区价格分布的差异，但还是能看到左下角偏紫色的一些房源集中在一起，属于价格偏高的一类。另一方面，从色温条可以看出，大多数房源的价格都是不超过 50000 的，而存在房源的价格超过了 250000。可能也是这之间的差异导致了区分的不明显。

4.2.2　核密度估计

为了进一步探索房租价格的分布趋势，我们采用核密度估计来近似房租价格的概率密度函数。在统计学中，核密度估计是估计密度函数的非参数方法，其详细定义可参见Wikipedia[①]，这里以应用为主。

基于 Python 的 Seaborn 可视化组件，绘制出 rent(房租) 的概率密度曲线如图4-3所示。从图中可以直观地看出，房租价格主要集中在约 20000 以下，最高的甚至超过了250000。

[①]https://en.wikipedia.org/wiki/Kernel_density_estimation

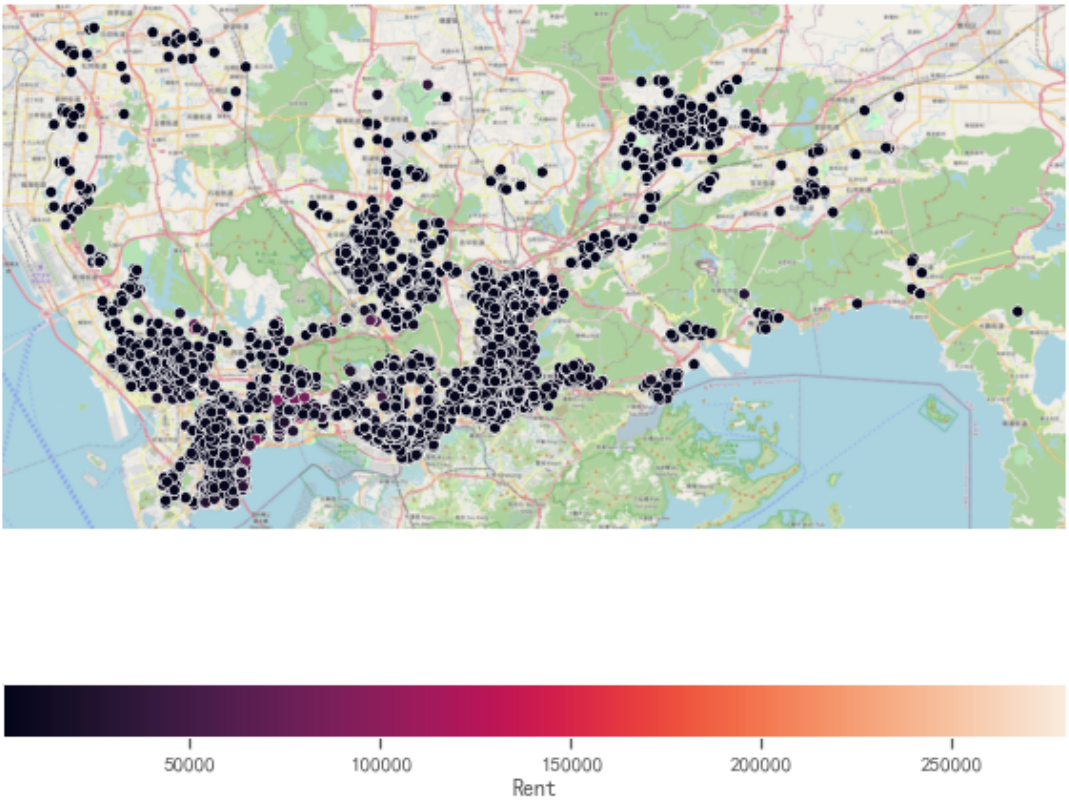


图 4-2　房租价格的地理可视化

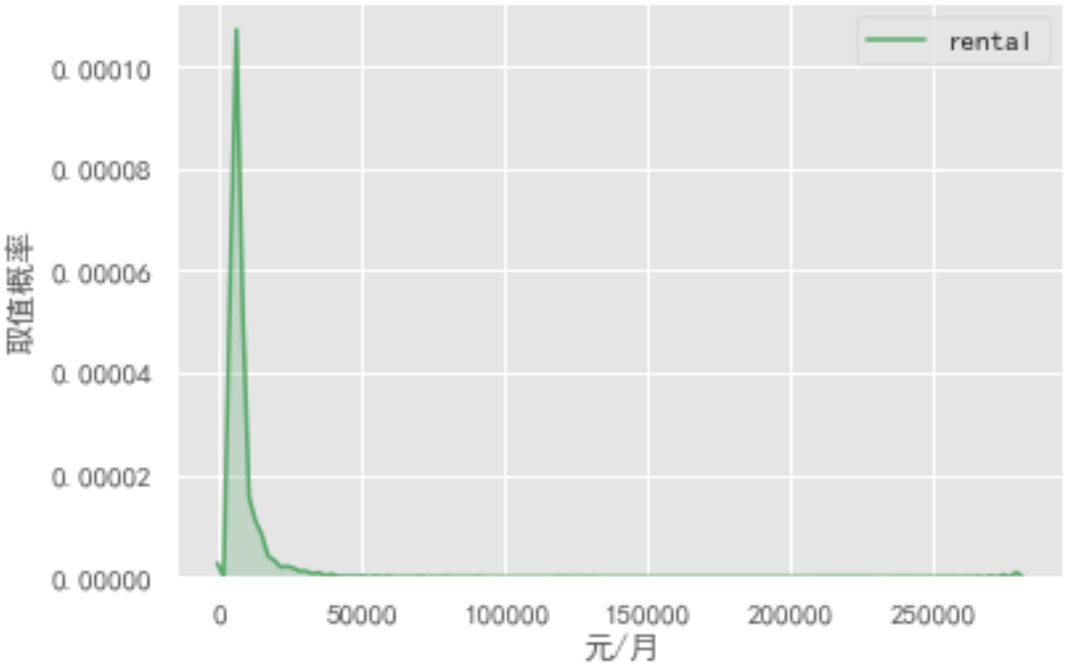


图 4-3　房租密度函数的核密度估计

根据上述分析，进一步缩小范围绘至最大值为 20000，再次将房租价格进行地理可视化，如图4-4。通过调整，可以清晰地看见不同房租的房源的分布趋势，即位于南山区、福田区的靠近香港一侧的高价格房源比较集中。这也说明了地理信息对于预测房租价格的重要性。

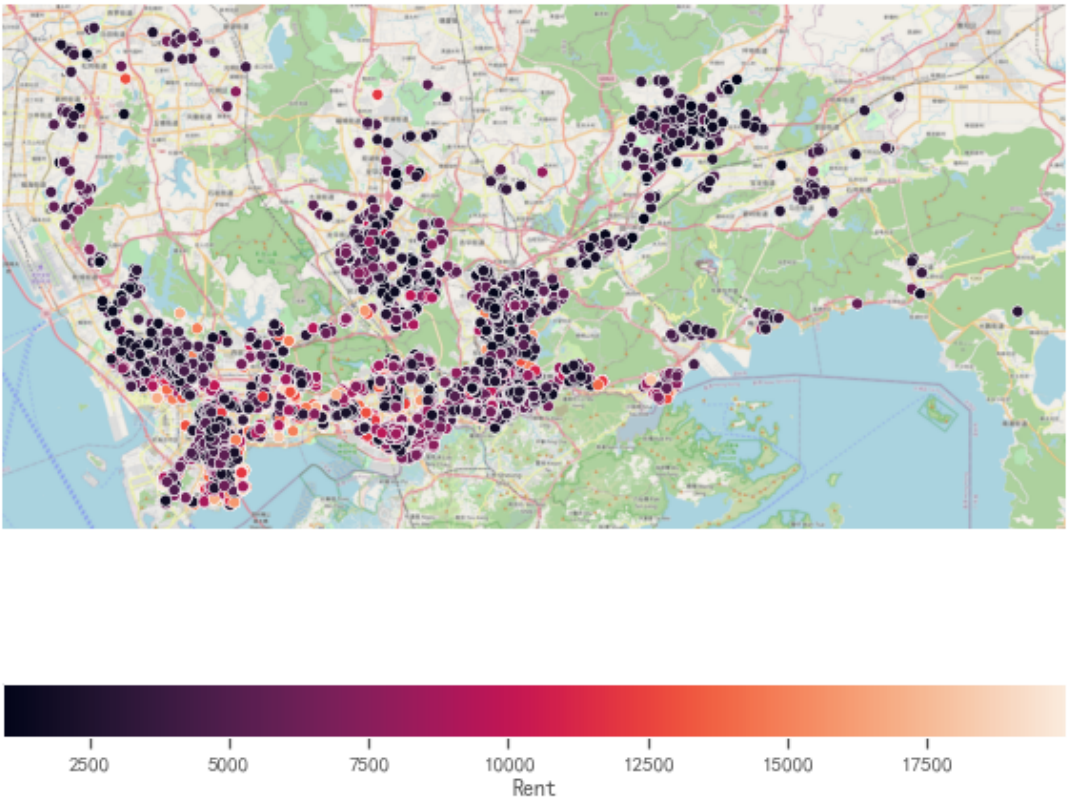


图 4-4　房租不超过 20000 的地理可视化

第5章　建模与实证分析

经过前面的数据处理与分析，我们接下来用于构建模型的变量有 15 个：area(面积), floor(楼层数), elevator(有无电梯), water(用水类型), electro(用电类型), gas(有无燃气), deposit(押金), service_fee(服务费), dis_nearest_metro(距最近地铁站距离), Lon(经度), Lat(纬度), lease_way_cat(租赁方式), room(房间数量), parlor(客厅数量), toilet(卫生间数量)，即变量个数 $d = 15$。此外，我们的总样本数为 $n = 22593$，预测目标为 rental(房租)。

基于此，我们首先利用 Python 的 train_test_split() 函数将数据的 70% 划分为训练数据集，剩下的 30% 划分为测试数据集，并设置 random_state(随机数种子) 为 7。然后分别将训练数据集和测试数据集进行 z-score 标准化。最后，我们将在5.1节简单介绍 K 折交叉检验的原理，在5.2节尝试核岭回归以及在5.3节尝试支持向量回归。

5.1　K 折交叉检验

K 折交叉检验是一种用来选择模型参数，以避免过拟合的常用方法。以 5 折交叉检验为例，其原理可概括为图5-1。

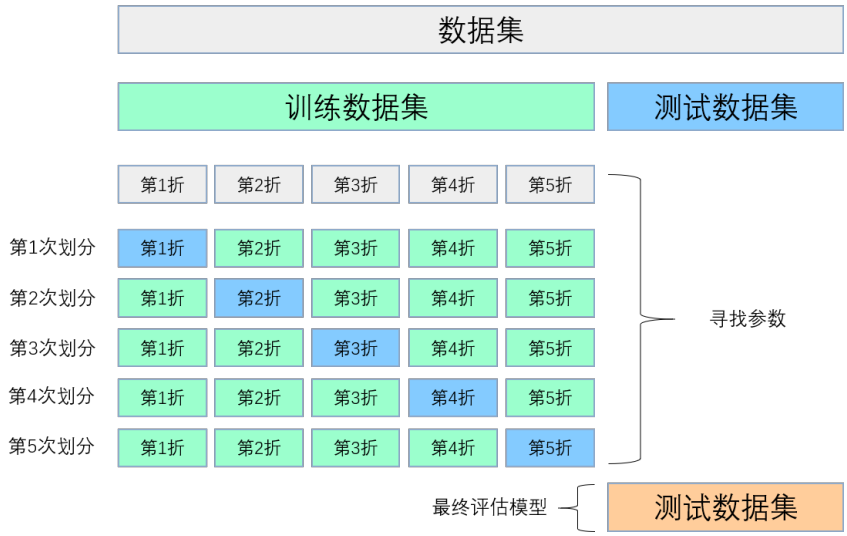


图 5-1　5 折交叉检验示意图

我们先将训练数据集划分为规模大小相近的 5 个部分 (即 5 折)，并给定需要调节的参数 α 的一组初始值 $\{\alpha_i : i = 1, 2, ..., m\}$。对于其中每一个值 α_i，基于 5 折交叉检

验。第一次划分时用第 1 折作为验证集，其余 4 折作为一个整体来训练模型，训练好后在该验证集上进行评估，记得分为 $score_1$。如此进行 5 次，取平均得分 $\bar{score}_{\alpha_i}$。待所有参数取值均计算得到相应得分后，比较选择得分最高的那个取值为最终模型参数的取值，并利用该参数取值，在完整的训练集上重新建立模型，再在测试集上做最终的评估。

5.2 核岭回归

回顾第2.3节中的式 (2-11)：$J(\alpha) = ||\mathrm{y} - \mathrm{K}\alpha||^2 + \lambda\alpha^T\mathrm{K}\alpha$，即核岭回归的目标函数，它有闭形式的解 (2-12)：$\widehat{\alpha} = (\mathrm{K} + \lambda\mathrm{I})^{-1}\mathrm{y}$。这里求解的思路是：首先指定核函数，计算出核矩阵 K 并选择适当的正则项参数 λ，然后由式2-12直接得到解。这里存在两个问题，一是核函数中的参数 (如高斯核中的 σ) 以及正则项参数 λ 如何才能恰当地选择，二是在直接套用式2-12时，如何恰当地求逆。

对于前一个问题，我们在上一节提到的 K 折交叉检验是一种可行的方法。而对于后一个问题，正如在2.4节中提到的，通常可以使用 Cholesky 分解，其计算复杂度是样本数量的立方级的，而使用共轭梯度法会使其变为 $O(rn^2)$，其中 r 为该方法的迭代次数、n 为样本数量，并且若使用的核函数为高斯核函数，其计算复杂度能降至 $O(n)$。

对于 RBF 核函数 $k(\mathrm{x}, \mathrm{x}') = e^{-\gamma||\mathrm{x}-\mathrm{x}'||^2}$，当 $\gamma = \sigma^{-2}$ 时即为高斯核函数，因此它们其实是等价的。

首先分别给定 γ 和 λ 的一组初始值 [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01]，基于 Python 的 KernelRidge() 函数和 GridSearchCV() 函数，设定 $cv = 10$(即 10 折交叉检验)，进行网格搜索选取最优参数组合[①]，并记录下所用的时间 t_1。经过 $t_1 = 20893.073s$ 后，整个过程的网格搜索得分如图5 - 2所示。其中，在 $\lambda = 0.0001$ 和 $\gamma = 0.0005$ 时取得最高得分。

因此接下来根据这两个参数取值，在训练集上重新训练基于 RBF 核函数的核岭回归模型。在训练完成后，我们可以得到模型在核空间的系数 α，其分量个数与训练样本的数量相同。从而根据式 (2-9)：$\mathrm{w} \equiv \sum_{i=1}^{n}\alpha_i\mathrm{x_i} = \mathrm{X}^T\alpha$ 计算得到各个变量的系数如表5 - 1所示。

最后我们在测试集上进行评估，预测结果与真实值之间的比较见图5 - 3。进一步可得到均方根误差为 $rmse = 0.0516$ 以及模型的决定系数 $R^2 = 0.9969$，这意味着我们要

[①]需要指出的是，KernelRidge() 函数的源代码中是直接使用 Cholesky 分解来进行求解的。

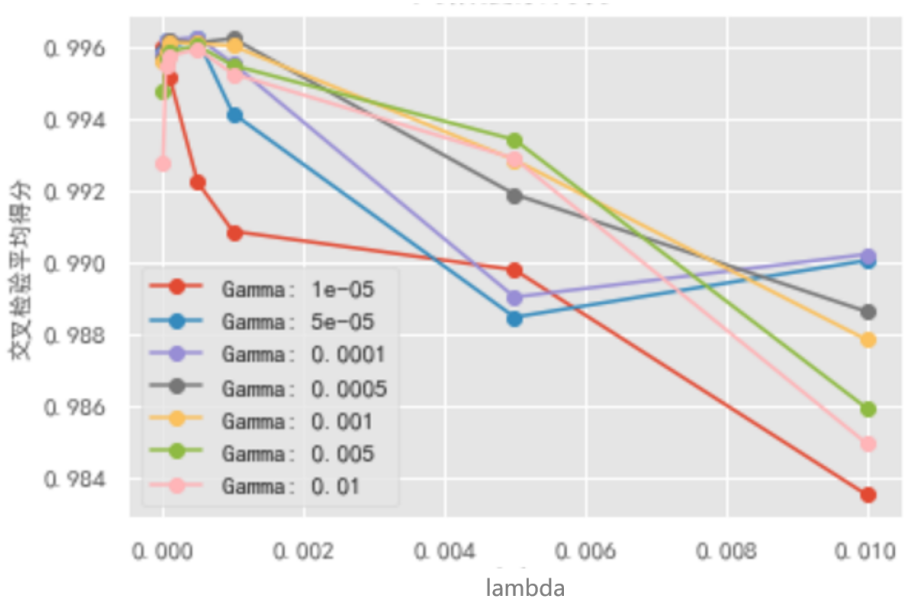


图 5-2　核岭回归的网格搜索得分

表 5-1　核岭回归模型变量系数

变量名	系数	变量名	系数	变量名	系数
area	284.1418	gas	29.1579	Lat	739.1848
floor	-158.1296	deposit	3023.6047	lease_way_cat	-352.7489
elevator	-180.7509	service_fee	-73.4232	room	77.6226
water	-104.3059	dis_nearest_metro	295.3294	parlor	61.0756
electro	100.1215	Lon	627.3274	toilet	-428.8066

预测的变量 rental(房租) 的方差中有 99.69% 能被这 15 个变量构成的核岭回归模型所解释。

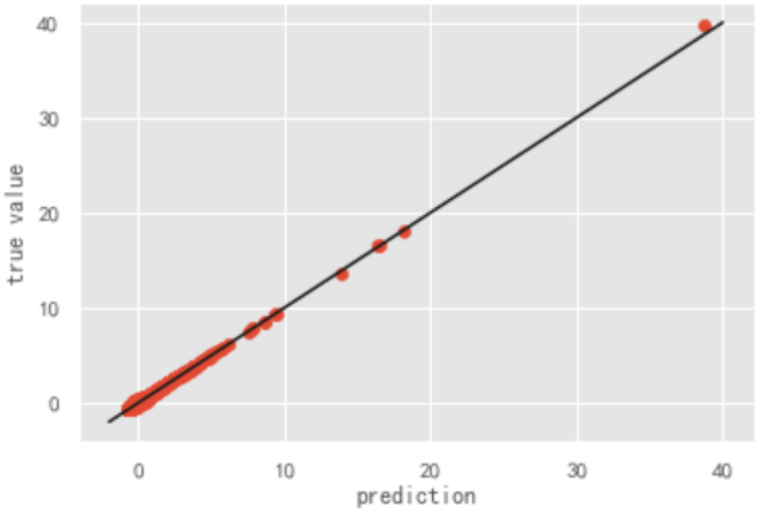


图 5-3　核岭回归模型预测值与真实值的比较

5.3　支持向量回归

在第2.4.4节中提到,SVR 的目标函数如式 (2-16):$J_{\mathrm{SVR}}(\boldsymbol{w}) = \frac{C}{n}\sum_{i=1}^{n} l_{\epsilon}\left(y_i - \langle \boldsymbol{x}_i, \boldsymbol{w}\rangle\right) + \frac{1}{2}\|\boldsymbol{w}\|_2^2$ 所示，求解该问题的标准方法是考虑由其对偶问题得到的二次规划问题[35]:

$$\begin{aligned} \max_{\varepsilon,\alpha_i^*} & P\left(\alpha_i, \alpha_i^{\star}\right) = \sum_{i=1}^{n} y_i\left(\alpha_i - \alpha_i^*\right) - \varepsilon\sum_{i=1}^{n}\left(\alpha_i + \alpha_i^*\right) \\ & -\frac{1}{2}\sum_{i,j=1}^{n}\left(\alpha_i - \alpha_i^*\right)\left(\alpha_j - \alpha_j^*\right) K\left(x_i, x_j\right) \\ \text{s.t.} & \sum_{i=1}^{N}\alpha_i = \sum_{i=1}^{N}\alpha_i^*, \quad 0 \le \alpha_i, \alpha_i^* \le C, i = 1, \ldots, n, \end{aligned} \tag{5-1}$$

其中，α_i, α_i^* 为拉格朗日乘子，$K\left(x_i, x_j\right)$ 为核函数相应取值。

我们在这里同样选择 RBF 核函数，首先分别给定 γ，C 和 ϵ 的一组初始值 [0.001, 0.01, 0.1, 1]，基于 Python 的 SVR() 函数和 GridSearchCV() 函数，选取 10 折交叉检验，在训练数据集上通过网格搜索选取最优参数组合，并记录下所用的时间 t_2。经过 $t_2 = 8327.3555s$ 后，通过调用参数 ‘best_params_’，得到最优参数组合为 { ‘C’ : 1, ‘ϵ’ : 0.01, ‘γ’ : 0.001}。

进一步利用训练好的模型在测试数据集上进行评估，预测结果与真实值之间的比较见图5-4。同时可计算得预测的均方根误差为 $rmse = 0.3241$ 以及模型的决定系数 $R^2 = 0.8949$，这意味着预测的 rental(房租) 的方差中仅有 89.49% 能被该模型所解释。

而从图5-4不难看出，预测效果差的部分主要集中在房租价格高的那些点上，尤其以最高的那个房源预测得效果最差。这说明就我们的数据而言，支持向量回归对于处理房租价格高的样本存在一定的问题。

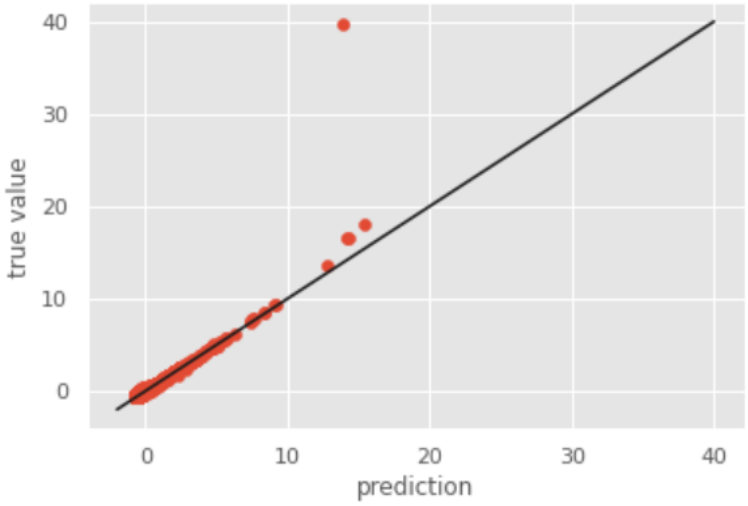


图 5-4　支持向量回归模型预测值与真实值的比较

出现这样的问题的原因可能是多样的：如给定的初始参数的取值范围太小，通过交叉检验所选出的最优参数组合只是局部范围内的最优；或是预测效果差的那些点可能为异常值点，其微小的扰动即可对模型有较大的影响等。

第6章　总结与展望

6.1　总结

随着更多政策的支持和置业成本的增加，我国的住房租赁市场正在迅猛发展。而其快速扩张的同时带来了租赁行为的不规范，同时租客和房东之间信息不对称的现象严重。因此，从合理的切入点着手对房租价格合理预测有着重要的意义。

本文首先总结了探讨影响房租因素相关的文献，为后文爬取数据作出铺垫。同时也总结了现存的预测房租的文献，并适当参考了对房价预测的文献。最后站在前人的肩膀上看问题，基于核方法能够在将低维非线性数据转换为高维线性数据的同时避免高维计算的特点，提出以核方法作为切入点进行分析。

接下来本文叙述了核方法的理论背景，存在的挑战以及相关解决方法。在理论背景部分，介绍了一些核函数相关的重要定义、性质和定理，给出了再生核希尔伯特空间的相关背景；在核方法的计算瓶颈部分，以核岭回归为例说明了其计算复杂度主要由训练样本的数量决定，能达到 $O(n^3)$ 的量级；在相关解决方法部分，针对具体的模型，如核岭回归、支持向量回归等，简单总结了相关文献中提到的改进计算的方法，如 Rahimi[34] 等人基于平移不变核提出的用于大规模核机器学习的随机特征等。

在数据部分，本文主要从数据获取和数据预处理两个模块进行展开。在数据获取模块，基于 Python 的网络爬虫框架 Scrapy，从链家租房网爬取了深圳市的房源出租数据。该过程中遇到信息获取不全的问题，通过设置筛选条件得到了解决，最后爬取到 23643 个样本，每个样本有 18 个变量。在数据预处理模块，针对分类变量做了编码、拆分、删除、缺失值填补等工作，其中缺失值填补时还延伸出了使用带正则项的 Logistic Regression 的分类子问题。在将相应分类变量转换为数值型变量后，进行了 z-score 标准化。数据处理完之后的样本总数为 $n = 22593$，变量个数为 16，包含要预测的房租变量。随后本文对处理好的数据进行了部分探索，如文本词云分析以及地理信息可视化。从前者可直观地看到在房源描述中哪些词更加的突出，而从后者也是可以直观地从地图上看到房租价格的分布趋势。

最后在建模与实证分析部分，本文以 5 折交叉检验为例说明了交叉检验的原理。并尝试核岭回归和支持向量回归，均选择 RBF 核函数，以 10 折交叉检验进行网格搜索选

表 6-1 核岭回归与支持向量回归的比较

	最佳参数组合	$rmse$	R^2	网格搜索时间
核岭回归	$[\lambda = 0.0001,\ \gamma = 0.0005]$	0.0516	0.9969	348.22 min
支持向量回归	$[C = 1,\ \gamma = 0.001,\ \epsilon = 0.01]$	0.3241	0.8949	138.79 min

取最优参数组合。选取完毕后利用该最优参数组合在训练数据集上训练模型，在测试数据集上作出预测，同时计算得模型在测试数据集上的预测均方根误差以及模型的决定系数。具体结果如表6-1所示。

6.2 需要进一步完善的工作

尽管核岭回归的预测效果较好，但是由于本文以应用为主，涉及到的东西太广泛，很多细节还不够完善，因此在以后的工作中还有以下这些需要不断完善的内容：

1. 在获取数据时，尝试优化爬虫的效率与结构，尽可能多地获取完整的数据。同时考虑增加获取房源的诸如所在商圈、区域等信息。

2. 基于在文献综述中提到的相关改进计算的算法，尝试自己编写代码，而不是借助于已有的框架，实现对核方法相关模型更高效的求解。

3. 在描述性统计分析部分对数据的挖掘还不够深入，之后的工作应当不局限于表面的可视化，而是深入分析数据之间的相关关系等。

4. 在通过交叉检验网格搜索选取最优参数组合时，囿于所需时间太长，本文仅在一个有限的范围内反复调试选取了参数。以后的工作中希望能够尝试更大的参数调节范围，以获得更好的模型。

5. 本文主要围绕基于核方法的相关模型，但是只尝试了核岭回归和支持向量回归，且支持向量回归部分还不够完善，故往后还应当完善这一部分并尝试更多的模型以及进行模型之间的比较。

致谢

随着论文的工作接近尾声，大学四年也即将成为过去。回首往事，历历在目，太多的人和事值得我道一声感谢。谨向给予我指导、帮助和关心的老师、朋友、亲人表示最衷心的感谢。

大学的前三年，不忘初心，砥砺前行。感谢曾经身边所有帮助我的人，使我扬帆远航，乘风破浪；也感谢所有给过我压力的人，使我披荆斩棘，逃离舒适区。

大学的最后一年，坐标奥兰多，美国。首先感谢学院给予的这个机会，也感谢在 UCF 交换这一年遇到的老师、同学们。这一年经历了太多，感谢所有帮助过我的人。

要特别感谢刘源远老师，由于疫情的缘故，每次通过网络汇报论文进度时，刘老师总是不断地给我信心和鼓励，会耐心地指出存在的问题，应该如何去修正。刘老师严谨求实的学术作风、渊博的学科知识以及平易近人的态度令我耳濡目染，受益终身。

最后要深深地感谢我的父母，他们无微不至的关心是我最坚实的后盾，他们也是每次我在面临困难时前进的动力。感谢我的父母对我的关怀、鼓励以及为我所做的一切。

参考文献

[1] 刘洪玉. 什么因素阻碍了租房市场健康发展[J]. 人民论坛, 2017, 24.

[2] 曹文聪. 深圳市住宅租金影响因素的实证研究[Z]. 广州 暨南大学, 2008.

[3] 汪业辉. 住房租赁市场租金的影响因素分析[Z]. 武汉 华中科技大学, 2014.

[4] ADAIR A S, BERRY J N, MCGREAL W S. Hedonic modelling, housing submarkets and residential valuation[J]. Journal of property Research, 1996, 13(1):67-83.

[5] 张蕾, 幸华. 基于特征价格模型的城中村出租屋租金影响因素研究: 以深圳市福田区为例[J]. 价格月刊, 2018(4):7.

[6] 王洪强, 李小雪, 张英婕. 上海市住宅租金价格空间分异格局及其影响因素分析[J]. 管理现代化, 2019(5):23.

[7] 张倩. 基于随机森林回归模型的住房租金预测模型的研究[Z]. 长春 东北师范大学, 2019.

[8] 谢勇, 项薇, 季孟忠, 等. 基于 Xgboost 和 LightGBM 算法预测住房月租金的应用分析[J]. 计算机应用与软件, 2019, 36:9.

[9] 马涛, 刘宁宁, 等. 基于集成学习的房租预测研究 Research of Prediction on House Rent Based on Intergration Learning[J]. Finance, 2019, 9(06):586.

[10] BASU S, THIBODEAU T G. Analysis of spatial autocorrelation in house prices[J]. The Journal of Real Estate Finance and Economics, 1998, 17(1):61-85.

[11] LIMSOMBUNCHAI V, et al. House price prediction: Hedonic price model vs. artificial neural network[C] 2004 Conference, June 25-26, 2004, Blenheim, New Zealand number 97781. New Zealand Agricultural and Resource Economics Society, 2004.

[12] WU C, YE X, REN F, et al. Spatial and social media data analytics of housing prices in shenzhen, china[J]. PloS one, 2016, 11(10).

[13] COVER T M. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition[J]. IEEE transactions on electronic computers, 1965(3): 326-334.

[14] KASHIMA H, IDÉ T, KATO T, et al. Recent advances and trends in large-scale kernel methods[J]. IEICE Transactions on Information and systems, 2009, 92(7):1338-1353.

[15] MERCER J. Xvi. functions of positive and negative type, and their connection the theory of integral equations[J]. Philosophical transactions of the royal society of London. Series

A, containing papers of a mathematical or physical character, 1909, 209(441-458):415-446.

[16] SHAWE-TAYLOR J, CRISTIANINI N, et al. Kernel methods for pattern analysis[M]. Cambridge university press, 2004.

[17] WILLIAMS C K, RASMUSSEN C E. Gaussian processes for machine learning volume 2 [M]. MIT press Cambridge, MA, 2006.

[18] KÖNIG H. Eigenvalue distribution of compact operators[J]. Operator theory, 1986, 16.

[19] SCHÖLKOPF B, HERBRICH R, SMOLA A J. A generalized representer theorem[C] International conference on computational learning theory. Springer, 2001 416-426.

[20] SCHÖLKOPF B, SMOLA A J, BACH F, et al. Learning with kernels: support vector machines, regularization, optimization, and beyond[M]. MIT press, 2002.

[21] ARONSZAJN N. Theory of reproducing kernels[J]. Transactions of the American mathematical society, 1950, 68(3):337-404.

[22] HOERL A E, KENNARD R W. Ridge regression: Biased estimation for nonorthogonal problems[J]. Technometrics, 1970, 12(1):55-67.

[23] VOVK V. Kernel ridge regression[M] Empirical inference. Springer, 2013 105-116.

[24] GELADI P, KOWALSKI B R. Partial least-squares regression: a tutorial[J]. Analytica chimica acta, 1986, 185:1-17.

[25] ROSIPAL R, TREJO L J. Kernel partial least squares regression in reproducing kernel hilbert space[J]. Journal of machine learning research, 2001, 2(Dec):97-123.

[26] KRÄMER N, BRAUN M L. Kernelizing pls, degrees of freedom, and efficient model selection[C] Proceedings of the 24th international conference on Machine learning. 2007 441-448.

[27] FREITAS N D, WANG Y, MAHDAVIANI M, et al. Fast krylov methods for n-body learning[C] Advances in neural information processing systems. 2006 251-258.

[28] TIBSHIRANI R. Regression shrinkage and selection via the lasso[J]. Journal of the Royal Statistical Society: Series B (Methodological), 1996, 58(1):267-288.

[29] ROTH V. Sparse kernel regressors[C] International Conference on Artificial Neural Networks. Springer, 2001 339-346.

[30] ROTH V. The generalized lasso[J]. IEEE transactions on neural networks, 2004, 15(1): 16-28.

[31] WANG G, YEUNG D Y, LOCHOVSKY F H. The kernel path in kernelized lasso[C] Artificial Intelligence and Statistics. 2007 580-587.

[32] DRUCKER H, BURGES C J, KAUFMAN L, et al. Support vector regression machines[C] Advances in neural information processing systems. 1997 155-161.

[33] TEO C H, SMOLA A, VISHWANATHAN S, et al. A scalable modular convex solver for regularized risk minimization[C] Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. 2007 727-736.

[34] RAHIMI A, RECHT B. Random features for large-scale kernel machines[C] Advances in neural information processing systems. 2008 1177-1184.

[35] ZHANG H, CHEN L, QU Y, et al. Support vector regression based on grid-search method for short-term wind power forecasting[J]. Journal of Applied Mathematics, 2014, 2014.

附录

Scrapy 爬虫代码主体

```
# -*- coding: utf-8 -*-
import scrapy
from scrapy.http import Request
from lianjia.items import LianjiaItem
import re

class RentalInfoSpider(scrapy.Spider):
    name = 'rental_info'
    allowed_domains = ['https://sz.lianjia.com']
    start_urls =
    ['https://sz.lianjia.com/zufang/dapengbandao/pg1/'] # all the urls, skip here.

    def parse(self, response):
        max_page = response.css('div.content___pg::attr(data-totalpage)').extract_first()
        bashurl = str(response.url)[:-2]
        for num in range(1, int(max_page) + 1):
            url = bashurl + str(num) + '/'
            yield Request(url, callback=self.parse1, dont_filter=True)

    def parse1(self, response):
        link3 = ['https://sz.lianjia.com{}'.format(i) for i in
            response.css('p.content___list--item-- title.twoline>a::attr(href)'
        ).extract() if '/zufang/SZ' in i]
        for url in link3:
            yield Request(url, callback=self.parse2, dont_filter=True)
```

```
def parse2(self, response):
    item = LianjiaItem()
    item['title'] = response.css('p.content___title::text').extract_first()
    try:
        item['rental'] =
            eval(response.css('div.content___aside--title>span::text').extract_first())
    except:
        item['rental'] = -1
    item['lease_way'] = response.css('ul.content___aside___list>li::text').extract()[0]
    item['house_type'] =
        response.css('ul.content___aside___list>li::text').extract()[1].split()[0]
    item['area'] = eval(re.findall('\d+',
        response.css('ul.content___aside___list>li::text').
    extract()[1].split('')[1])[0])
    item['direction'] =
        response.css('ul.content___aside___list>li::text').extract()[2].split()[0]
    try:
        item['floor'] = eval(re.findall('\d+',
            response.css('ul.content___aside___list>li::text').extract()
        [2].split()[1])[0])
    except:
        item['floor'] = -1
    item['maintain_date'] = \
        [re.findall('维护：(.+)', i)[0] for i in
            response.css('li.fl.oneline::text').extract() if '维护' in i][0]
    item['elevator'] = \
        [re.findall('电梯：(.+)', i)[0] for i in
            response.css('li.fl.oneline::text').extract() if '电梯' in i][0]
    item['water'] = \
        [re.findall('用水：(.+)', i)[0] for i in
```

```
            response.css('li.fl.oneline::text').extract() if '用水' in i][0]
        item['electro'] = \
            [re.findall('用电：(.+)', i)[0] for i in
                response.css('li.fl.oneline::text').extract() if '用电' in i][0]
        item['gas'] = [re.findall('燃气：(.+)', i)[0] for i in
            response.css('li.fl.oneline::text').extract() if '燃气' in i][0]
        try:
            item['desc'] =
                response.css('div#desc>p::attr(data-desc)').extract_first().replace('<br
                />', ' ').replace('\n', ' ').strip()
        except:
            item['desc'] = 'NAN'
        item['deposit'] = eval(response.css('ul.table_row>li::text').extract()[2])
        item['service_fee'] = eval(response.css('ul.table_row>li::text').extract()[3])
        try:
            item['dis_nearest_metro'] = eval(re.findall('\d+',
                response.css('div#around>ul>li>span::text').extract()[1])[0])
        except:
            item['dis_nearest_metro'] = -1
        item['Lon'] = eval(re.findall('[0-9.]+',
            response.css('script::text').extract()[3].split(';')[6])[1])
        item['Lat'] = eval(re.findall('[0-9.]+',
            response.css('script::text').extract()[3].split(';')[6])[2])
        yield item
```

数据预处理代码

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
%matplotlib inline
import seaborn as sns
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
sns.set(font='SimHei')
plt.style.use('ggplot')

#Connect to DataBase
import mysql.connector
mydb = mysql.connector.connect(
  host="127.0.0.1",
  user="root",
  passwd="970828Lyh@$",
  database="lianjia"
)
#get dataframe
df = pd.read_sql('SELECT * FROM lianjia_tb', con=mydb)
#close database
mydb.close()
df.drop('id', 1, inplace=True)
# drop duplicate values !
df.drop_duplicates(keep=False,inplace=True)
#reset index
df.reset_index(drop = True,inplace=True)
#bianry encode lease_way
df["lease_way"] = df["lease_way"].astype('category')
df["lease_way_cat"] = df["lease_way"].cat.codes
ax = sns.countplot(x="lease_way", data=df, order=['合租', '整租'])
plt.xlabel('房源出租类型')
plt.ylabel('频数')
```

```
for p in ax.patches:
    current_width = p.get_width()
    diff = current_width - .35
    # we change the bar width
    p.set_width(.35)
    # we recenter the bar
    p.set_x(p.get_x() + diff * .5)
    ax.annotate('{}'.format(p.get_height()), (p.get_x()+p.get_width()/3,
        p.get_height()+150))
plt.savefig('lease_way.png')

#Note:delete the row contain "未知"
df = df[~df.house_type.str.contains("未知")]

# divide into 3
df[['room', 'parlor', 'toilet']] = df['house_type'].str.extract('(.+)室(\d+)厅(\d+)卫',
    expand=False).astype('int64')

# treat as category variable
idxs = df.groupby('house_type').filter(lambda s:len(s)<1000).index
df.loc[idxs,'house_type'] = '其他'
ax = sns.countplot(y='house_type', data=df, order=df.house_type.value_counts().index)
plt.ylabel('户型')
plt.xlabel('频数')
for p in ax.patches:
        ax.annotate('{}'.format(p.get_width()), (p.get_x() + p.get_width() + 0.02,
            p.get_y() + p.get_height()/2))
plt.savefig('house_type.png', bbox_inches = "tight")

def convert_water(x):
```

```
    if x == '民水':
        x = 1
    elif x == '商水':
        x = 0
    else:
        x = np.nan
    return x
df['water'] = df['water'].map(convert_water)
df[df['water'].isnull()].water

def convert_ele(x):
    if x == '民电':
        x = 1
    elif x == '商电':
        x = 0
    else:
        x = np.nan
    return x
df['electro'] = df['electro'].map(convert_ele)
df[df['electro'].isnull()].electro

def convert(x):
    if x == '有':
        x = 1
    elif x == '无':
        x = 0
    else:
        x = np.nan
    return x
df['elevator'] = df['elevator'].map(convert)
```

```
df[df['elevator'].isnull()].elevator
df['gas'] = df['gas'].map(convert)
df[df['gas'].isnull()].gas

df.reset_index(drop = True,inplace=True)
#elevator  gas   water  electro
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
penalty = ['l1', 'l2']
C = np.logspace(-5, -2, 10)
parameters = dict(C=C, penalty=penalty)
import time
from sklearn.metrics import zero_one_loss, make_scorer
def my_custom_zero_one_scorer(y_true, y_pred):
    return 1 - zero_one_loss(y_true, y_pred)
zero_one_scorer = make_scorer(my_custom_zero_one_scorer)
def zero_to_negone(x):
    if x == 0:
        return -1
    return 1
Data1 = df.iloc[df[df['elevator'].notnull()].index]
X = Data1[[i for i in Data1.describe().columns if i not in ['elevator', 'gas', 'water',
    'electro']]]
y = Data1['elevator'].map(zero_to_negone)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=7)
scaler = StandardScaler()
X_train = scaler.fit(X_train).transform(X_train)
X_test = scaler.fit(X_test).transform(X_test)
```

```
clf1 = GridSearchCV(LogisticRegression(random_state=7), parameters, cv=10,
    scoring=zero_one_scorer)
s = time.time()
best_model = clf1.fit(X_train, y_train.astype('int'))
e = time.time()
t = e - s
best_model.best_params_
model = LogisticRegression(random_state=7, C= 0.00215, penalty='l1')
model.fit(X_test, y_test)
best_model.score(X_test, y_test.astype('int'))
NAN_Data = df.iloc[df[df['elevator'].isnull()].index]
NAN_X = NAN_Data[[i for i in Data1.describe().columns if i not in ['elevator', 'gas',
    'water', 'electro']]]
NAN_X = scaler.fit(NAN_X).transform(NAN_X)
best_model.predict(NAN_X)
def negone_to_zero(x):
    if x == -1:
        return 0
    return 1
df['elevator'].iloc[df[df['elevator'].isnull()].index] = list(map(negone_to_zero,
    best_model.predict(NAN_X)))
df = df[~df.gas.isnull()]
df = df[~df.water.isnull()]
df = df[~df.electro.isnull()]
df.reset_index(drop = True,inplace=True)
df.to_csv('Processed_Data.csv', index = False, header=True)
```

词云分析及地理可视化代码

```
processed_data = pd.read_csv('Processed_Data.csv')
text = processed_data.desc.values
all_text = ''
for i in text:
    if i != 'NAN':
        all_text = all_text + str(i) + ''
import jieba
cut_text = " ".join(jieba.cut(all_text))
lis_text = cut_text.split()
def word_fre_count(lis_text):
    dic_text = dict()
    for i in lis_text:
        dic_text[i] = dic_text.get(i, 0) + 1
    fre_descen = sorted(dic_text.items(),key=lambda x:x[1],reverse=True)
    print(fre_descen[:20])
word_fre_count(lis_text)
lis_text = [e for e in lis_text if e not in (',', '、','。', '的', '】', '【', ': ', '有', '是', '好',
'路', '米', '等', '和')]
#note:skip some
cut_text = ' '.join(lis_text)
from os import path
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
font = r'C:\Windows\Fonts\simfang.ttf'
bg_img = np.array(Image.open('house.jpg'))
wordcloud = WordCloud(
    font_path=font,
    mask=bg_img,
    width=3000,
```

```
    height=2000,
    background_color='white',
    random_state=7,
    stopwords=STOPWORDS).generate(cut_text)
image_colors = ImageColorGenerator(bg_img)
fig = plt.figure(
    figsize = (40, 30),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.savefig('wordcloud.png', bbox_inches = "tight")

import geopandas as gpd
gdf = gpd.GeoDataFrame(processed_data, geometry =
gpd.points_from_xy(processed_data['Lon'],
processed_data['Lat']))
import contextily as ctx
gdf.crs = {'init': 'epsg:4326', 'no_defs': True}
gdf = gdf.to_crs(epsg=3857)
ax = gdf.plot( figsize =(10, 10), alpha=1, edgecolor='w',
column = "rental", legend = True,
                legend_kwds={'label':
                "Rent",'orientation': "horizontal"})
ctx.add_basemap(ax, url=ctx.sources.OSM_A)#ST_TERRAIN
ax.set_axis_off()
plt.savefig('geo1.png', bbox_inches = "tight")
sns.kdeplot(gdf['rental'],shade=True,color="g")
plt.xlabel('元/月')
```

```
plt.ylabel('取值概率')
plt.savefig('kde.png', bbox_inches = "tight")
ax = gdf[gdf['rental']<20000].plot( figsize =(10, 10), alpha=1, edgecolor='w', column =
    "rental", legend = True,
                legend_kwds={'label': "Rent",'orientation': "horizontal"})
ctx.add_basemap(ax, url=ctx.sources.OSM_A)#ST_TERRAIN
ax.set_axis_off()
plt.savefig('geo2.png', bbox_inches = "tight")
```

KRR 与 SVR

```
import time
from sklearn.kernel_ridge import KernelRidge
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = processed_data[[i for i in
processed_data.describe().columns if i not in
['rental']]]
y = processed_data['rental']
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.3, random_state=7)
X_train = scaler.fit(X_train).transform(X_train)
X_test = scaler.fit(X_test).transform(X_test)
y_train = scaler.fit(np.array(y_train).reshape(-1,1)).tr
ansform(np.array(y_train).reshape(-1,1))
y_test = scaler.fit(np.array(y_test).reshape(-1,1)).tran
sform(np.array(y_test).reshape(-1,1))
```

```
model2_gs = GridSearchCV(KernelRidge(kernel='rbf'),
cv=10, param_grid={"alpha": [0.00001, 0.00005, 0.0001,
0.0005, 0.001, 0.005, 0.01], "gamma": [0.00001, 0.00005,
0.0001, 0.0005, 0.001, 0.005, 0.01]})
s2 = time.time()
model2_gs.fit(X_train, y_train)
e2 = time.time()
t2 = e2-s2
model2_gs.score(X_test, y_test)
pre2 = model2_gs.predict(X_test)
rmse2 = np.sqrt(mean_squared_error(y_test,pre2))
rmse2
plt.scatter(pre2, y_test)
plt.plot(np.linspace(-2, 40), np.linspace(-2, 40), color='k')
plt.xlabel('prediction')
plt.ylabel('true value')
model2_gs.best_params_
model2 = KernelRidge(kernel ='rbf', alpha=0.0001,
gamma=0.0005 )
s = time.time()
model2.fit(X_train, y_train)
e = time.time()
t = e - s
model2.score(X_test, y_test)
#corresponding coef_
np.dot(X_train.transpose(), model2.dual_coef_)
df_cv_results = pd.DataFrame(model2_gs.cv_results_)
scores_mean = df_cv_results['mean_test_score']
scores_mean = np.array(scores_mean).reshape(7, 7)
```

```
scores_sd = df_cv_results['std_test_score']
scores_sd = np.array(scores_sd).reshape(7, 7)
# Plot Grid search scores
_, ax = plt.subplots(1,1)
# Param1 is the X-axis, Param 2 is represented as a
 different  curve (color  line)
for idx, val in enumerate([0.00001, 0.00005, 0.0001,
0.0005, 0.001, 0.005, 0.01]):
    ax.plot([0.00001, 0.00005, 0.0001, 0.0005, 0.001,
    0.005, 0.01], scores_mean[idx, :], '-o', label=
    'Gamma' + ': ' + str(val))
ax.set_title("网格搜索得分", fontsize=15, fontweight='bold')
ax.set_xlabel('alpha', fontsize=12)
ax.set_ylabel('交叉检验平均得分', fontsize=10)
ax.legend(loc="best")
ax.grid('on')

from sklearn.svm import SVR
svr_gs1 = GridSearchCV(SVR(kernel='rbf'), cv=10, param_grid={"C": [0.001, 0.01, 0.1, 1],
    "gamma": [0.001, 0.01, 0.1, 1],'epsilon' :[0.001,  0.01,  0.1,  1]})
s = time.time()
svr_gs1.fit(X_train, y_train.reshape(len(y_train), ))
e = time.time()
t = e - s
svr_gs1.best_params_
svr_gs1.score(X_test, y_test.reshape(len(y_test), ))
model = SVR(kernel='rbf', gamma=0.001, C=1, epsilon= 0.01)
model.fit(X_train, y_train)
model.score(X_test, y_test)
rmse = np.sqrt(mean_squared_error(y_test,model.predict(X_test)))
```