|  |
| --- |
| 南农校徽  南农大  本科生毕业论文（设计）  题 目: 基于机器学习模型的蔬菜订单分析与  需求预测  姓 名: 陈冰  学 号: 23321211  学 院: 理学院  专 业: 统计学  指导教师: 温阳俊 职称 副教授  20 年 月 日 |

南京农业大学本科生毕业论文（设计）原创性声明

本人郑重声明：所呈交的毕业论文（设计），是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。



论文作者签名： 日期： 年 月 日

南京农业大学本科生毕业论文（设计）使用授权声明

本学位论文作者完全了解学校有关保留、使用毕业论文（设计）的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权南京农业大学教务处可以将本毕业论文（设计）的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编毕业论文（设计）。



论文作者签名： 导师签名：

日期： 年 月 日 日期： 年 月 日

基于机器学习模型的蔬菜产品订单分析与需求预测

# 摘 要

近年来，随着人们对高品质生活的追求，消费者对蔬菜类产品的品质要求也越来越高。对于供货商而言，如何最大限度地确保每日售卖蔬菜的新鲜度，是亟待解决的关键问题。而深入了解消费者对于各蔬菜产品的需求量是有效解决这一难题的重要途径。

本文以某商超消费者对各蔬菜产品的需求量为研究对象，构建不同维度下影响蔬菜产品需求量变化的特征，对不同蔬菜种类和每个蔬菜单品的日需求量分别进行预测。在不同蔬菜大类日需求量预测研究部分，分别建立随机森林模型、XGBoost模型和GRU模型，并采用五折交叉验证和网格搜索调整超参数进行模型训练。最后，通过MSE、MAE、RSE等评价指标对比分析不同模型的拟合预测效果，并选择最优模型预测未来7天的日需求量。根据研究结果，XGBoost模型的泛化能力最好，为最优模型。以辣椒类产品为例，其未来七天日需求量（单位：斤）的预测值分别为82.0688、82.8005、81.7094、78.8456、79.8265、71.3576、65.1901。在每个蔬菜单品日需求量的预测研究部分，构建了随机森林模型和XGBoost模型。结果发现，XGBoost模型在交叉验证集和测试集的评估指标值最优，可认定为最佳模型。基于该模型，对最近一个月频繁出现的蔬菜单品在未来七天的日需求量进行了预测。

关键词：随机森林模型；XGBoost模型；GRU模型；蔬菜需求量预测

**ANALYSIS OF VEGETABLE PRODUCT ORDERS AND DEMAND FORECASTING UTILIZING MACHINE LEARNING MODELS**

# **ABSTRACT**

In recent years, the growing focus on a high-quality lifestyle has resulted in elevated consumer expectations regarding the quality of vegetable products. Consequently, suppliers are confronted with the urgent challenge of guaranteeing the utmost freshness of vegetables sold on a daily basis, necessitating immediate attention. A comprehensive understanding of consumer demand for a variety of vegetable products constitutes an effective strategy to address this challenge.

This study examines consumer demand for various vegetable products within a specific supermarket context and identifies the factors influencing demand fluctuations across multiple dimensions. It offers distinct forecasts for daily demand concerning different vegetable categories and individual vegetable items. In the section focused on predicting daily demand for major vegetable categories, this research employs Random Forest (RF), XGBoost, and Gated Recurrent Unit (GRU) models, utilizing five-fold cross-validation and grid search techniques to optimize hyperparameters during the model training process. Ultimately, through a comparative analysis employing evaluation metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and Relative Standard Error (RSE), the study assesses the predictive performance of the various models and identifies the optimal model for forecasting daily demand over the subsequent seven days. The findings indicate that the XGBoost model demonstrates superior generalization capabilities, establishing it as the most effective model. For instance, the predicted daily demands for chili pepper products over the upcoming week are 41.0344 kg, 41.4003 kg, 40.8547 kg, 39.4228 kg, 39.9133 kg, 35.6788 kg, and 32.5950 kg, respectively.

This study develops Random Forest and XGBoost models to predict daily demand for individual vegetable items within each category. The results demonstrate that XGBoost consistently outperforms RF across various evaluation metrics in both cross-validation and test sets, thereby establishing it as the superior model based on these findings. Based on this model, this study predicts the daily demand for frequently appearing vegetable dishes over the past month for the next seven days.

**KEY WORDS：**Random Forest Model; XGBoost Model; GRU Model; Vegetable Demand Forecasting

目录

[摘 要 I](#_Toc197178156)

[**ABSTRACT** II](#_Toc197178157)

[第一章 绪论 1](#_Toc197178158)

[1.1 研究背景和意义 1](#_Toc197178159)

[1.1.1 选题背景 1](#_Toc197178160)

[1.1.2 研究意义 1](#_Toc197178161)

[1.2 国内外研究现状 1](#_Toc197178162)

[1.3 研究内容与技术路线 2](#_Toc197178163)

[1.3.1 研究内容 2](#_Toc197178164)

[1.3.2 技术路线 3](#_Toc197178165)

[第二章 相关理论基础 4](#_Toc197178166)

[2.1 蔬菜需求量影响因素理论分析 4](#_Toc197178167)

[2.2 RF模型概述 4](#_Toc197178168)

[2.3 XGBoost模型概述 6](#_Toc197178169)

[2.4 GRU模型概述 7](#_Toc197178170)

[第三章 数据处理和描述性分析 9](#_Toc197178171)

[3.1 数据预处理 9](#_Toc197178172)

[3.1.1 数据检查与处理 9](#_Toc197178173)

[3.1.2 特征构建 9](#_Toc197178174)

[3.1.3 数据整合 9](#_Toc197178175)

[3.2 描述性统计 10](#_Toc197178176)

[3.2.1 蔬菜类产品需求量的时间变化规律 10](#_Toc197178177)

[3.2.2 不同蔬菜种类需求量描述分析 10](#_Toc197178178)

[第四章 蔬菜类产品需求量预测研究 17](#_Toc197178179)

[4.1 特征选择 17](#_Toc197178180)

[4.1.1 影响不同种类蔬菜日需求量的特征选择 17](#_Toc197178181)

[4.1.2 影响不同蔬菜单品需求量的特征选择 18](#_Toc197178182)

[4.2 不同种类蔬菜日需求量预测 18](#_Toc197178183)

[4.2.1 构建RF预测模型 18](#_Toc197178184)

[4.2.2 构建XGBoost预测模型 20](#_Toc197178185)

[4.2.3 构建GRU预测模型 21](#_Toc197178186)

[4.2.4 预测结果 22](#_Toc197178187)

[4.3 不同蔬菜单品日需求量预测 23](#_Toc197178188)

[4.3.1 构建RF预测模型 23](#_Toc197178189)

[4.3.2 构建XGBoost预测模型 24](#_Toc197178190)

[4.3.3 预测结果 24](#_Toc197178191)

[第五章 结论与展望 23](#_Toc197178192)

[5.1 结论 26](#_Toc197178193)

[5.2 创新点 26](#_Toc197178194)

[5.3 建议与展望 27](#_Toc197178195)

[参考文献 28](#_Toc197178196)

[附 录 29](#_Toc197178197)

[致 谢 44](#_Toc197178198)

# 第一章 绪论

## 1.1 研究背景和意义

### 1.1.1 选题背景

随着全球人口的不断增长和城市化进程加快，食品需求呈现出快速增长的趋势。居民对新鲜蔬菜的需求日益上升这一现状不仅影响了农产品的生产和分配，也对食品安全和营养健康提出了更高的要求。超市作为消费者直接购买蔬菜的重要场所，必须重视如何最大化确保每日蔬菜的新鲜度。这一任务不仅关乎提升消费者的购物体验，还直接影响顾客的购买决策及超市的销售业绩。为了满足消费者对高质量产品的需求，增强顾客忠诚度，并促进超市的可持续发展，超市需采取有效、精准的需求预测、优化进货流程，以确保每日蔬菜能够全部售罄且保持新鲜，从而更有效地管理库存、减少损耗，为顾客提供最新鲜的蔬菜，提升整体竞争力。

需求预测是基于历史数据，对未来一段时间内市场需求进行估计和推测的过程，其特点是持续性和实时动态性。精准的需求预测能够为公司管理层在销售及运营计划、目标设定和资金预算方面提供重要的决策参考[1]。然而，由于时间、季节、气候、产品种类和促销活动等多种复杂因素的不确定性，蔬菜产品的需求预测面临诸多挑战。因此，超市及相关企业必须借助更先进的算法和模型，提高需求预测的精度。这样，不仅可以确保新鲜蔬菜的供应，还能更好地应对市场的快速变化，在复杂多变的环境中保持竞争优势，实现可持续发展。

### 1.1.2 研究意义

本研究以系统科学理论为指导，探索影响蔬菜需求的关键因素，并结合多维度时间数据进行分析，采用机器学习模型和深度学习模型两种思路，旨在提高预测精度并比较不同模型在实际应用中的表现。这将有助于揭示现代机器学习算法与传统统计模型在需求预测中的优缺点，为未来的预测技术提供新的参考和应用场景。

传统的蔬菜需求预测多依赖于线性回归等经典统计模型，但这些方法难以有效处理复杂的非线性关系，因此在面对多变的市场数据时，预测精度较低。与之相比，基于机器学习或深度学习预测方法能够处理高维、复杂的数据，提供更为精确的预测结果。精准的需求预测对于蔬菜生产和供应链管理至关重要，它能帮助企业制定科学的生产计划、优化库存管理，确保市场供需平衡并提升消费者满意度。此外，准确的预测还为企业在应对政策变化时提供了数据支持，帮助其做出经济最优的决策，实现利润最大化，推动行业的可持续发展。

## 1.2 国内外研究现状

需求预测技术有助于通过减少缺货情况来提高客户满意度，并通过改善库存计划和降低库存积压来减少成本，对企业意义重大。现阶段需求预测的研究方法大致可分为传统的统计模型、机器学习和深度学习模型。

对于基于传统统计模型的需求预测方法，如指数平滑(Exponential Smoothing, ES)[2]、自回归移动平均(Auto Regression Integrated Moving Average, ARIMA)[3]等模型，其使用历史需求数据的线性函数来预测未来的需求。Ramanathan&Usha[4]通过结合产品特定需求因素，使用多元线性回归方法提高了需求量的预测精度。Kourentzes & Petropoulos[5]提出改进的多重聚合预测算法(Multiple Aggregation Prediction Algorithm, MAPA)，使用多重时间聚合来改进已建立的指数平滑方法，将指数平滑的简单性和可靠性与 MAPA 鲁棒性相结合，较好的提高了预测的准确性。还有学者采用自回归移动平均模型，基于加权马尔科夫模型修正残差状态，构建加权马尔科夫-ARIMA模型，准确预测区域物流需求[6]。

在需求预测方面，常用的机器学习模型包括决策树(Decision Tree, DT)[7]、随机森林(Random Forest, RF)[8]、梯度决策树(Gradient Boosting Decision Tree, GBDT)[9]等模型。此类模型能够通过对训练数据的“自我学习”来积累经验，从而进行预测。目前，Huber & Stuckenschmidt[10]已发现机器学习方法比传统预测方法更准确，也更适用于大规模需求预测。庆豪[11]基于灰色—神经网络对民机需求进行预测，预测误差仅为1.61%。此外，机器学习算法现已经十分丰富，不少学者采用多种机器预测模型对同一问题进行研究，并比较不同模型之间的优劣，例如范华鹏[12]应用随机森林回归、ExtraTrees 回归、LightGBM 回归和支持向量机回归(Support Vector Machine, SVM)四种机器学习方法，选择最优碳储量估测模型对滇西北森林地上碳储量进行估测。

深度学习模型如循环神经网络(Recurrent Neural Networks, RNN)、深度强化学习(Deep Reinforcement Learning, DRL)、生成对抗网络(Generative Adversarial Networks, GANs)等是机器学习的一种拓展。此类模型能够自动学习并提取数据中的高级特征，从而有效处理复杂且非线性的庞大数据集。周雅夫等[13]提出了一种融合注意力机制的卷积神经网络(Convolutional Neural Networks, CNN)和门控循环单元(Gated Recurrent Unit, GRU)的组合模型(CNN-GRU)预测燃料电池老化趋势。徐慧智和杨冰冰[14]采用CNN与长短时记忆神经网络(Long Short-Term Memory, LSTM)的组合预测模型(CNN-LSTM)，针对铁路枢纽站出租车需求量进行短时预测研究，该方法有效提升了预测精度。Abbasimehr, Shabani& Yousefi[15]提出了一种基于多层 LSTM 的方法来解决单变量需求预测问题。黄欣等[16]利用GRU深度学习天然气价格数据集中的有用信息来预测价格，该方法和其他方法相比，具有较高鲁棒性。

## 1.3 研究内容与技术路线

### 1.3.1 研究内容

蔬菜作为日常必需品，其需求的精准预测不仅有利于帮助零售商优化库存管理，降低损耗，还保证了消费者的饮食安全。本文以某商超的蔬菜的需求量数据为研究对象，旨在通过构建多种预测模型对未来蔬菜需求量进行更准确的预估。

考虑到蔬菜品类繁多、受季节影响显著，且消费者偏好各异，本文将对收集到的六大种类蔬菜分别进行预测，并探讨不同种类蔬菜日需求量之间的关联。通过构建机器学习和深度学习模型，分别对不同种类蔬菜的日需求量以及各个蔬菜单品的日需求量进行预测。分析比较两者差别，以实现更精确的日需求量预测。

基于不同蔬菜品类的特性，本文将深入探究蔬菜需求量变化的规律，对不同种类蔬菜产品和每个蔬菜单品的日需求量分别进行预测。全文共分为五章：

第一章是绪论，将简单介绍蔬菜产品需求量预测的研究背景和意义，回顾国内外需求预测模型的发展和应用现状，并阐述本文的研究内容，研究路线、整体架构和创新点；

第二章是相关理论基础，将详细介绍研究过程中涉及到模型的算法原理和构建流程；

第三章是数据处理和描述性分析，将阐明了蔬菜类产品需求量数据的预处理步骤，通过对各项变量指标的描述性统计分析，探究影响蔬菜需求量的重要因素；

第四章是蔬菜类产品需求量预测研究，将利用随机森林对数据进行特征选择，针对筛选出的重要特征构建RF和XGBoost模型，分别对不同种类蔬菜产品和每个蔬菜单品的日需求量进行预测。同时利用深度学习模型GRU，探究原始数据中对需求量有影响的特征，并预测未来需求量。

第五章是结论与展望，将对全文的研究成果进行总结归纳，并结合当前消费者对蔬菜品质的需求，为推动蔬菜供应链发展提出相关建议，同时指出本研究的局限性。

### 1.3.2 技术路线

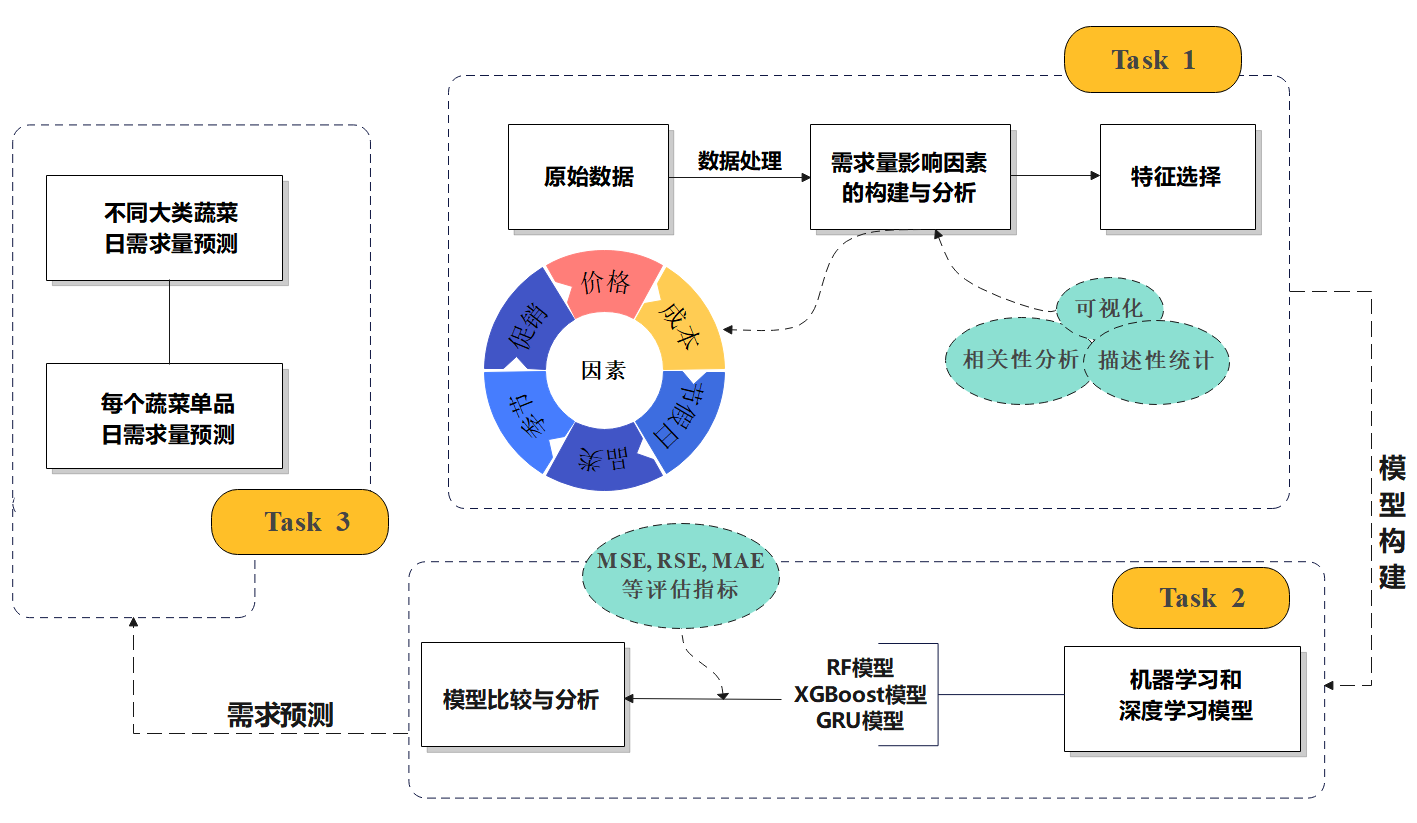
首先进行数据清洗和结构化处理，构建与需求量相关的因素，并通过可视化手段进行统计描述，展示各特征对蔬菜产品需求量的影响。在模型构建阶段，采用多种机器学习和深度学习方法，使用均方误差（MSE）、相对标准误（RSE）和平均绝对误差（MAE）等多个指标作为评估标准，选择最优模型进行需求预测，以确保预测准确性。在需求预测阶段，对不同类别的蔬菜及每个蔬菜单品进行日需求量的预测，通过模型输出的结果，帮助商超进行决策。

图 1技术路线图

Figure 1 Technical Roadmap

# 第二章 相关理论基础

## 2.1 蔬菜需求量影响因素理论分析

本文以某商超的蔬菜需求量为研究对象，在需求量预测部分，考虑了多重因素的影响。通过文献分析[17]并结合数据特征，本文初步提取出13个可能影响蔬菜需求量的因素进行深入分析，并构建预测模型。

(1) year：代表年份特征。社会的经济状况在不同年份可能处于不同的趋势，这与消费者的购买力息息相关。社会经济上行时，蔬菜产品的需求量增加，反之下降。另外不同年份发布一些新的政策、完成蔬菜种植的某些技术突破，这些都会影响蔬菜类产品的需求量。

(2) month, quarter：分别代表月份与季度特征。大部分蔬菜具有明显的时令特性，不同月份的气温和降雨等天气条件不同，主产蔬菜种类也不同，如夏季黄瓜、豆角、茄子等蔬菜纷纷上市，而冬天则以白菜为主。不同月份的蔬菜的供应结构有显著差异，其价格也会不同，从而影响消费者对部分种类蔬菜的需求量。

(3) phase：代表月阶段特征，包括月初、月中、月末三个阶段。考虑到消费者的发薪日大部分在月初和月中，这可能导致月中和月初成为消费高峰期，从而增加该阶段蔬菜产品的需求量。

(4) date：代表日特征。这是本研究中颗粒度最小的时间特征，旨在更深入的探究每日蔬菜产品需求量的变化规律。

(5) code：代表产品名称。该特征旨在探究消费者对不同蔬菜的需求量。

(6) classify\_code：代表产品所属大类。消费者对于不同种类的蔬菜有不同的需求，如花菜类是日常餐桌上的常客，根茎类则多用于炖煮和煲汤。因此蔬菜所属种类也会影响其需求量。

(7) price：代表产品价格。蔬菜的价格是消费者购买时最重要的考量因素之一，不同消费者对蔬菜价格的敏感度不同，从而会直接影响对蔬菜类产品的需求量。

(8) cost：代表产品成本。蔬菜的总成本直接影响其市场价格。

(9) on\_sale：代表是否促销。打折、满减、赠品等促销活动能够降低消费者购买的门槛，刺激消费者的购买欲望，从而提高需求量。

(10) holiday\_or\_weekend, is\_holiday, is\_weekend：holiday\_or\_weekend代表是否为周末或节假日。周末和节假日一般而言是休闲时间，消费者一般会选择在这个阶段集中购物，商家也会利用此时间开展一些促销宣传活动，因此可能对蔬菜需求量有一定的影响。为衡定单独的节假日和周末特征对需求量的影响，本文另设定了特征is\_holiday和is\_weekend。

## 2.2 RF模型概述

RF是一种基于Bagging (Bootstrap aggregating)[18]算法的集成模型，旨在聚合多个弱学习器，令其形成一个性能更好的模型，每个基模型之间相互独立。RF模型的基模型为决策树模型。决策树模型是一种基本的分类与回归方法，包括ID3算法(Iterative Dichotomiser 3)、C4.5算法和CART算法(Classification And Regression Tree)。前两者只能进行分类预测，只有CART算法可以进行回归预测。本文中蔬菜需求量数据为连续型数据，采用基于CART算法的决策树构成的随机森林模型。以下为算法原理：

决策树在节点处将特征空间划分为不同的单元，划分的起点为根节点，最终划分单元为叶节点，划分过程形成的中间节点为内部节点。在训练过程中，模型遍历可用于划分样本空间的每一个特征以及对应特征的所有取值，根据均方误差最小化原则，计算划分过程中的目标函数值增益，选取最优划分点。如此递归下去，直到所有的训练数据子集被基本正确分类，或无法再继续划分。[19]假设数据集由个样本和个特征构成，第个特征的取值分别为，label值分别为，输入空间最终被划分为个单元 ，则具体构建步骤如下：

(1) 假设存在切分特征与切分点。将输入空间划分为和两部分。

(2) 计算划分前后的SSE增益值。

①划分前数据的SSE：

②划分后两个子集的SSE：

③划分后整体加权SSE：

(3) 计算划分前后SSE的最大增益，判断最优划分点()

(4) 继续对两个子区域调用步骤(1)-(3)，直至满足停止条件,生成决策树，如式(2.2.7)

(5) 基于随机选取样本或特征的方式，从原始训练数据集中有放回抽取多个子集，针对每个子集重复步骤(1)-(4)构建决策树，这若干个决策树则构成了随机森林模型。

(6) 通过取加权平均值的方法组合每棵决策树的预测结果，即为RF模型最终的预测结果。

本文使用scikit-learn库中的RandomForestRegressor函数建立随机森林回归模型、GridSearchCV函数进行网格搜索寻找最优超参数，并使用joblib包中的dump函数保存最优模型。

## 2.3 XGBoost模型概述

XGBoost模型(eXtreme Gradient Boosting Model)是一种基于Boosting算法[18]的集成模型。旨在将多个弱学习器通过加法模型组装成一个强学习器。该模型采用向前分步算法进行迭代，每次迭代中的弱学习器都将以目前的预测误差作为拟合目标，进行目标函数的优化。为增强型的泛化能力，减小模型的方差和偏差，XGBoost模型的目标函数包括损失函数和正则项两部分。[20]

假设数据有个样本分别为，XGBoost模型由个弱学习器组成,分别为,则其总目标函数为式(2.3.1)，第步迭代的目标函数为式(2.3.2)，去掉常数项，并由泰勒二阶展开进行近似，可得式(2.3.3)。

令, ,则：

由于常数对优化无意义，舍掉。可得最终的目标函数为式(2.3.4)。

本文选定每一次迭代的弱学习器均为回归决策树模型。定义DT模型的正则项为式(2.3.5)。[20]其中为控制惩罚强度的超参数，为第棵决策树模型的叶节点数量，为这棵树第个叶节点的输出值。

将式(2.3.5)代入式(2.3.4)，定义函数，并令, , ，则可得XGBoost模型的具体目标函数为式(2.3.6)。

为损失函数的二阶导数，损失函数为严格的凸函数，所以和始终非负，故为一个开口向上的二次函数，进而可得叶节点输出为函数对称轴的横坐标时，目标函数有最优值。

假设数据有个特征，分别为，第个特征的候选分裂点为遍历每个特征以及该特征对应的分裂点，根据将样本划分为子集，计算当前分裂的目标函数增益为式(2.3.7)。若，则不进行分裂，反之，则取为最大值时对应的分裂点为最佳划分节点并进行划分，然后对划分得到的两个子区域继续重复上述步骤，直至满足停止条件，生成回归树。

本文使用xgboost库中的XGBRegressor函数建立XGBoost回归模型，scikit-learn库中的GridSearchCV函数进行网格搜索寻找最优超参数，使用joblib包中的dump函数保存最优模型。

## 2.4 GRU模型概述

GRU是一种具有门控机制的神经网络单元，旨在缓解RNN模型存在的梯度消失问题。GRU模型记忆信息的门控单元包括重置门和隐藏门两部分。重置门决定了对历史信息的保留程度，有助于捕获序列中的短期依赖关系，更新门决定使用多少历史信息和当前信息来更新当前隐藏状态，有助于捕获序列中的长期依赖关系。隐藏状态是信息在模型中被存储和传递时的内部状态，包括前时刻的历史信息和时刻输入的序列信息，分为候选隐藏状态和更新隐藏状态两种。前者是指当前输入和前一隐藏状态计算得到的一个潜在的新隐藏状态。后者是通过对前一隐藏状态和候选隐藏状态进行加权平均值得到的每个时间步产生的最终隐藏状态。每个单元的计算公式如式(2.4.3)-式(2.4.6)。[21]

假设为当前时刻的输入信息，维度为；为上一时刻的隐藏状态，维度为，为传递到下一时刻的隐藏状态；为候选隐藏状态；为更新门输出；为重置门输出；为重置门权重矩阵；为更新门权重矩阵；为候选隐藏状态计算矩阵。定义为Sigmoid函数，为双曲正切函数，其函数表达式分别为式(2.4.1)和(2.4.2)。

重置门和更新门部分将历史信息和当前输入信息进行合并，并与各自的权重矩阵相乘，选择Sigmoid函数为激活函数将值映射至之间，分别得到和。

将重置门得到的和做Hadamard乘积，对历史信息进行遗忘，同时并入当前输入信息，利用作映射再使用函数激活，可得候选隐藏状态。

将更新门得到的和做Hadamard乘积，决定使用多少当前信息。将与作Hadamard乘积，决定使用多少历史信息。二者之和即为该时间步向下一层网络的输出和传入同层网络下一时间步的隐藏状态。

本文采用pytorch库构建GRU模型并对其进行训练。

# 第三章 数据处理和描述性分析

## 3.1 数据预处理

本文数据为某商超2020年7月1日至2023年6月30日各蔬菜产品的销售明细。原始数据的处理分析部分主要利用R语言编写代码，具体可见附录二（2.1）。

### 3.1.1 数据检查与处理

原始数据集的有878042个样本，包含251种蔬菜产品，分属6大类，分别为：辣椒类、食用菌类、茄类、水生根茎类、花叶类和花菜类。经检查发现，数据中不存在缺失值、NAN值以及重复值。在异常值处理中，发现部分产品的价格低于其成本。考虑到产品在正常非打折销售情况下，其价格通常不会低于成本，本文将这部分数据认定为异常数据，将其删除。另外，本文发现2022-06-09的鲜粽叶(袋)(1)产品的需求量高达160斤，而该产品在其他时刻的需求量仅为1斤。因此，可将这一数据认定为异常值并删除。针对同一天中同一蔬菜产品价格多次变换的情况，仅保留出现次数最多价格作为该商品在当日的最终价格。经上述处理后，最终剩余844258条数据。

### 3.1.2 特征构建

在特征工程阶段，为能更全面地反映销售数据的时间属性，从而为后续模型的建立和预测提供有力的支持。本文对数据中的销售日期进行了细致的挖掘，以捕捉时间因素对销售的影响。具体而言，首先利用tidyverse包[22]中的year()、quarter()、month()和day()函数，从销售日期中分别提取出“年”、“季度”、“月”和“日”这些时间粒度的特征。在此基础上，为了进一步刻画销售日期在时间轴上的特殊性，引入了节假日信息。通过调用chinese\_calendar函数，判断每个销售日期是否为中国的法定节假日，并将此判断结果作为一个二元特征。同时，还探究了销售日期是处于工作日还是周末，以及是否为月初、月中或月末等具有潜在销售规律的特殊时间节点，并相应地构建了这些时间节点特征。最后，将分类特征的取值全部转化为0、1、2等对应的整数编码。

### 3.1.3 数据整合

为了满足后续需求预测的多样性，本文不仅着眼于预测该商超不同种类蔬菜产品的日需求量，还进一步细化预测颗粒度，对每一个蔬菜单品的日需求量进行预测。比较分析这两种产品粒度下预测结果的差异。基于上述需求，对数据进行整合处理。首先，依次提取6个大类产品的销售数据，对于“需求量(qty)”、“成本(cost)”、“价格(price)”、“是否促销(on\_sale)”四个变量分别以日为单位进行聚和处理。例如，以“水生根茎类”产品为例，本文取“on\_sale”变量值在该类下当日的均值做为该大类产品当日的促销状态。以各个单品的需求量为权重，对单品的价格进行加权平均得到当日该类蔬菜产品的销售价格，对单品的成本进行加权平均得到当日该类蔬菜产品的成本。

## 3.2 描述性统计

### 3.2.1 蔬菜类产品需求量的时间变化规律

三年间该商超每月蔬菜类产品的总需求量如图2所示。整体来看，需求量呈先下降再回升的波动趋势。在2021年1月至2022年5月，蔬菜需求量多次明显下滑，并持续处于低迷状态，直至2022年7月才逐步上升，恢复至较高水平。结合当时的社会背景分析，这可能与新冠疫情爆发有关。2021至2022年，疫情对国民经济造成了严重冲击，居民消费能力受到抑制，从而影响了蔬菜等生活必需品的市场需求，导致需求量显著下滑。

|  |
| --- |
| Rplot04  图 2蔬菜类产品每月总需求量时序图  Figure 2 Time series chart of total monthly demand for vegetable products |
|  |

### 3.2.2 不同蔬菜种类需求量描述分析

(1) 基于数字特征进行描述

为深入了解不同种类蔬菜需求量数据的分布特征，分别计算6大种类蔬菜日需求量的最大值、最小值、平均值、标准差、偏度系数和峰度系数，以进行简单描述和分析，如表1所示。通过观察表中最小值和最大值两列，可以发现蔬菜类产品的在不同时刻的需求量差异较大，由均值和标准差这两列可以发现不同蔬菜种类日需求量数据的中心趋势和离散程度具有显著差异。六大蔬菜种类的所有偏度值都大于0可以说明数据分布相对于均值向右倾斜，其中，辣椒类数据的偏度最大，表明其需求量分布相对于其他品类更加右倾。六大蔬菜种类的峰度系数都大于3说明数据分布相对于标准正态分布更陡峭。另外，还可发现花叶类蔬菜产品需求量的标准差最大，说明其日需求量的波动比较大，日需求量不稳定，同时，该类产品需求量的最大值和均值也都最大。

表 1六大蔬菜种类日需求量的数字特征

Table 1 The numerical characteristics of daily demand for six major types of vegetables

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 种类名称 | 最大值(斤) | 最小值(斤) | 均值(斤) | 标准差 | 偏度系数 | 峰度系数 |
| 水生根茎类 | 296.79 | 0.926 | 35.94 | 30.535 | 2.64 | 13.44 |
| 花叶类 | 1209.47 | 31.298 | 172.41 | 79.911 | 2.92 | 27.96 |
| 花菜类 | 186.16 | 0.632 | 37.29 | 22.290 | 1.61 | 4.73 |
| 茄类 | 118.93 | 0.252 | 21.14 | 12.852 | 1.73 | 6.11 |
| 辣椒类 | 600.51 | 6.066 | 79.77 | 51.117 | 3.53 | 22.32 |
| 食用菌 | 511.14 | 2.241 | 66.12 | 47.344 | 3.21 | 18.66 |

(2) 不同蔬菜种类每月需求量分布情况

六大蔬菜种类每月的需求量的分布情况如图3所示。可以发现，消费者对于花叶类蔬菜商品的需求量始终高于其他种类蔬菜的需求量，且随时间波动较大，最低需求量和最高需求量相差约7000斤。茄类产品的需求量始终在600斤左右上下波动，且随时间变化没有显著的变化趋势。

|  |
| --- |
| Rplot03  图 3不同种类蔬菜每月总需求量时序图 |

Figure 3 Time series chart of total monthly demand for different types of vegetables

(3) 不同蔬菜种类的需求量分布占比

六大蔬菜种类总需求量在蔬菜类产品中占比情况如图4所示。水生根茎类蔬菜和花菜类蔬菜在整体蔬菜产品需求中所占比例相近。消费者需求量最大的蔬菜类别是花叶类产品，需求量最小的则是茄类产品。这一差异可能与消费者的饮食偏好以及各类蔬菜中所包含的品种数量有关。

|  |
| --- |
| 图 4蔬菜总需求量分布情况 |

Figure 4 Distribution of total demand for vegetables

(4) 不同蔬菜种类的相关性分析

本文利用各大类蔬菜产品每月的总需求量数据，计算了六大蔬菜种类日需求量之间的相关系数，以热力图形式呈现，如图5。由图发现，花叶类和花菜类之间、水生根茎类和食用菌之间具有较强的正相关性，说明一方产品的需求增加会带动另一方产品的需求增加，即两类产品为互补品。同时可发现，茄类与食用菌和水生根茎类产品之间呈负相关性，茄类的需求量增加会导致水生根茎类产品和食用菌的需求量减少，即茄类与食用菌和水生根茎类产品之间为替代品。

|  |
| --- |
| 图 5不同蔬菜种类相关系数热力图 |

Figure 5 Heatmap of correlation coefficients among different vegetable types

(5) 不同大类中蔬菜单品分布情况

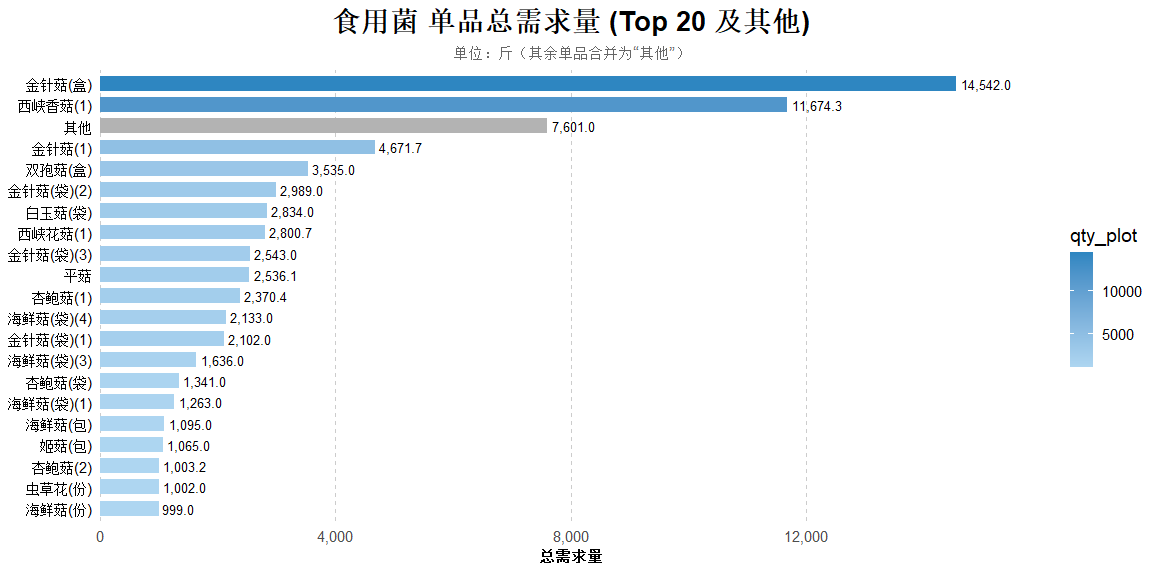
 本文提取不同大类蔬菜产品的数据，分别统计每一大类中蔬菜单品的总需求量分布情况。如图6至11所示，花菜类包含的蔬菜单品最少，仅有五种，其中需求量最高的西兰花，三年来消费者对其总需求量超过25000斤，消费者对紫白菜(1)和紫白菜(2)的需求量是最低的。在茄类产品中，消费者最偏好紫茄子(2)这一蔬菜单品，三年总需求量超过13000斤。在水生根茎类商品中，单品净藕(1)的需求量最高，而单品洪湖莲藕的需求量排名虽仅次于净藕(1)，但三年内总需求量数值远远低于净藕(1)。花叶类、辣椒类和食用菌三类产品包含了众多蔬菜单品。其中花叶类中包含99种蔬菜单品，需求量最高的为大白菜；辣椒类中包含43种蔬菜单品，需求量最高的为芜湖青椒(1)；食用菌中包含70种蔬菜单品，需求量最高的为金针菇(盒)。

图 6食用菌类单品总需求量

Figure 6 Total demand for edible mushroom varieties

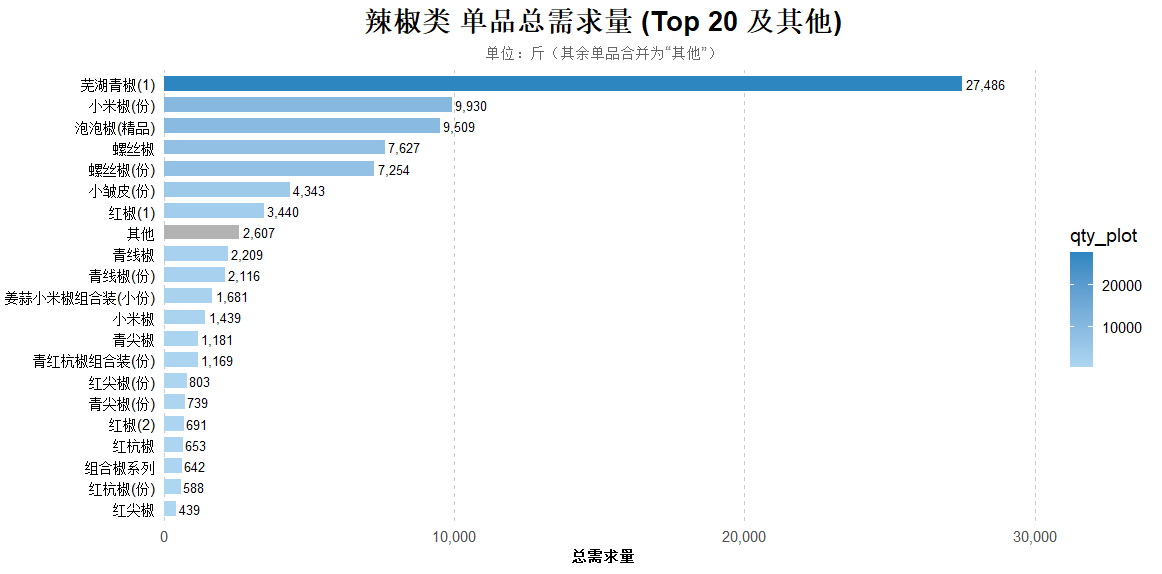


图 7辣椒类单品总需求量

Figure 7 Total demand for chili products

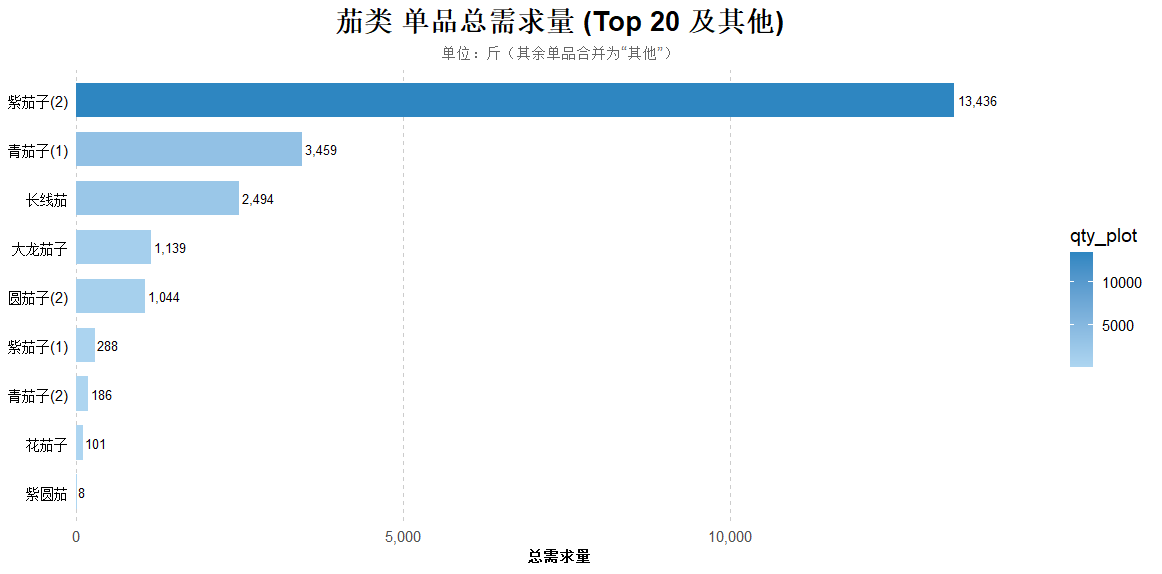


图 8茄类单品总需求量

Figure 8 Total demand for eggplant products

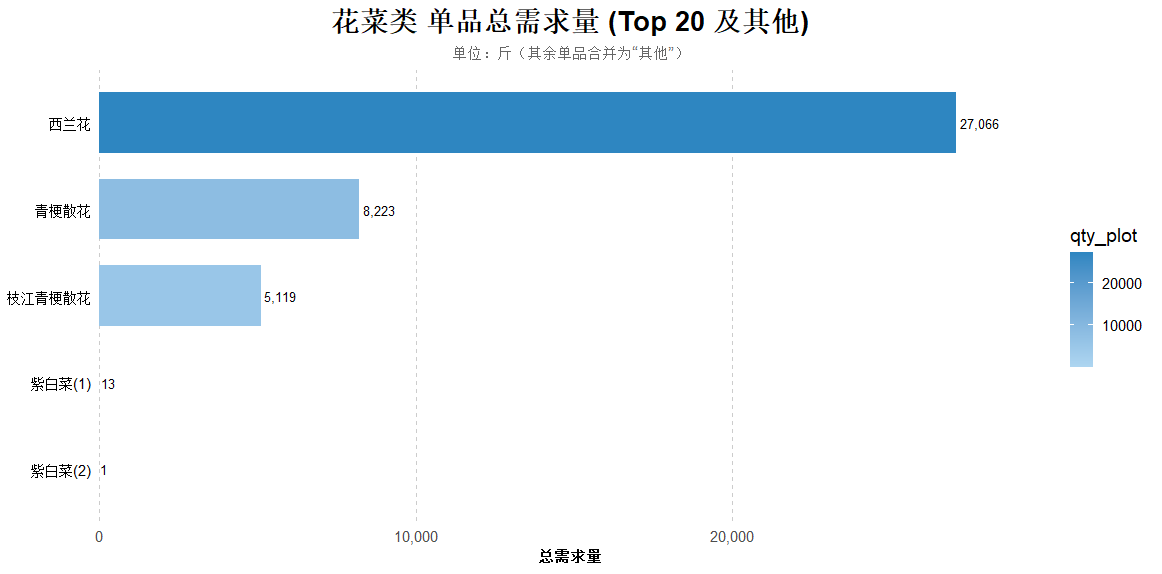


图 9花菜类单品总需求量

Figure 9 Total demand for cauliflower products

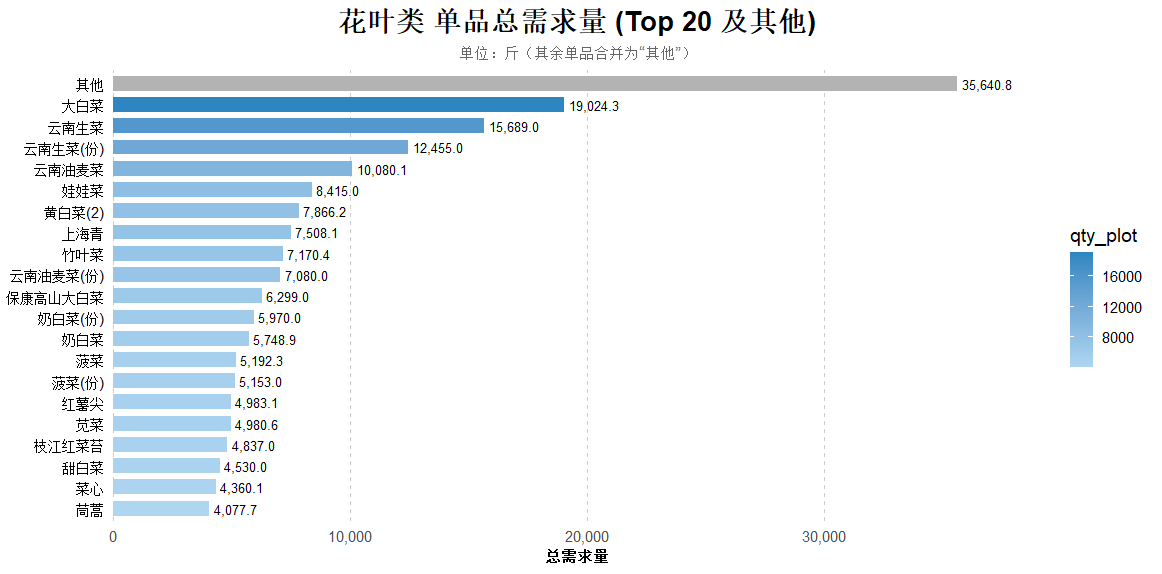


图 10花叶类单品总需求量

Figure 10 Total demand for cauliflower products

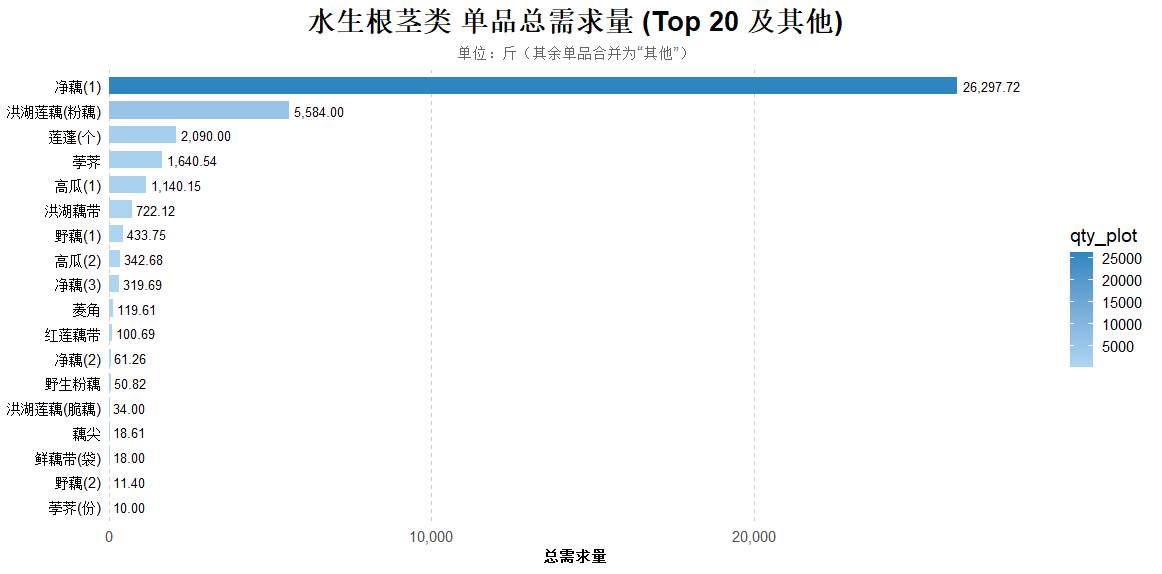


图 11水生根茎类单品总需求量

Figure 11 Total demand for aquatic rhizome products

本章采用R语言编写代码所涉及的函数及其功能见表2。具体代码见附录二(2.1)和(2.2)。

表 2本章所涉及的R语言函数及其功能

Table 2 The R language functions and their functionalities discussed in this chapter

|  |  |
| --- | --- |
| 函数名 | 功能 |
| ggplot | 画图 |
| filter | 返回满足条件的行 |
| select | 筛选出指定的列 |
| mutate | 创建新的变量 |
| pivot\_longer | 将宽数据转化为长数据 |
| summarise | 对原数据框每一行进行统计量的计算 |
| across | 对数据框的多个列执行相同的操作 |
| distinct | 去除一个数据框中指定变量组合值的重复数据 |
| group\_by | 接收列名作为参数，将数据按指定列值的组合进行分组。 |
| lapply | 接收一个列表和一个参数量为1的函数，将列表中每一个元素都传递给这个函数进行一次运算，将所有计算结果拼接成一个列表返回。 |
| inner\_join | 接收两个 data.frame 和指定列名作为参数，逐行判断第二个表指定列的值组合是否出现在第一个表中，若存在，则将第二个表该行的剩余列添加在第一个表对应行的后面，若不存在，则不作任何修改。该函数只返回匹配结束后第一个表被扩充列数的那些行。 |
| left\_join | 接收两个 data.frame 和指定列名作为参数，逐行检查第一个表指定列的值组合是否在第二个表中，若存在，则将第二个表中该行的其他列追加到第一个表对应行后，若不存在，则添加值为NA。 |

# 第四章 蔬菜类产品需求量预测研究

## 4.1 特征选择

针对数据构建了较多特征，为了筛选出真正对蔬菜类产品需求量影响显著的变量，本文采用随机森林模型评估特征的重要性，并在模型拟合后指定阈值选择特征。其核心思想为：在随机森林算法中，决策树节点的划分效果用纯度变化衡量，纯度变化越高，表明当前节点利用该特征进行划分的效果更好，反之效果越差。每棵决策树都利用特征计算样本切分后的纯度变化，计算一个特征在所有决策树上的纯度变化量，若该特征下纯度变化越高，则说明该特征具有较显著的重要性，反之则说明该特征不太重要。因此，可根据纯度变化来衡量特征的重要性，保留重要特征或删除不重要的特征[19]。

本文利用Python编写代码，使用scikit-learn库[23]中的RandomForestRegressor函数训练RF模型，分别对不同种类蔬菜日需求量数据和每个蔬菜单品日需求量数据进行模型拟合，从而判断其特征的显著性，具体代码见附录二(2.3)。

### 4.1.1 影响不同种类蔬菜日需求量的特征选择

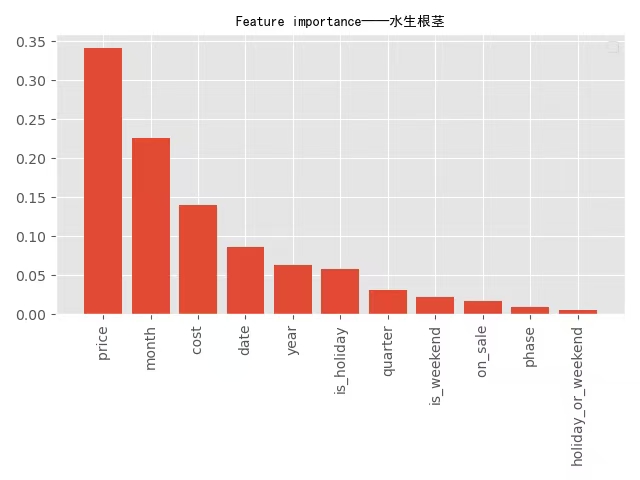
本文共构建了13个可能影响产品需求量的特征，具体见2.1节。显然，不同种类蔬菜日需求量数据中不存在code和classify\_code这两个特征，因此，可能影响每一种类蔬菜日需求量的特征只有11个。对其各特征进行显著性计算，根据计算结果，选定0.05为阈值进行特征筛选。以水生根茎类产品为例，各特征显著性程度如图12所示。可得影响该类产品日需求量的重要特征price, month, cost, date, year, is\_holiday。

图 12影响水生根茎类产品日需求量的特征重要性

Figure 12 The importance of characteristics affecting the daily demand for aquatic rhizome products

对于其余蔬菜种类，针对筛选阈值0.05，影响各种类需求量的重要特征及如表3所示。

表 3影响不同蔬菜种类日需求量的重要特征

Table 3 Important characteristics that affect the daily demand for different types of vegetables

|  |  |
| --- | --- |
| 种类 | 影响需求量的重要特征 |
| 辣椒类 | price, cost, year, month, date, is\_holiday |
| 花叶类 | price, cost, month, is\_holiday, year, date |
| 水生根茎类 | price, month, cost, date, year, is\_holiday |
| 食用菌 | month, price, cost, year, date, is\_holiday, quarter |
| 花菜类 | cost, year, price, month, date, is\_holiday |
| 茄类 | price, month, cost, date, year, is\_holiday, quarter |

### 4.1.2 影响不同蔬菜单品需求量的特征选择

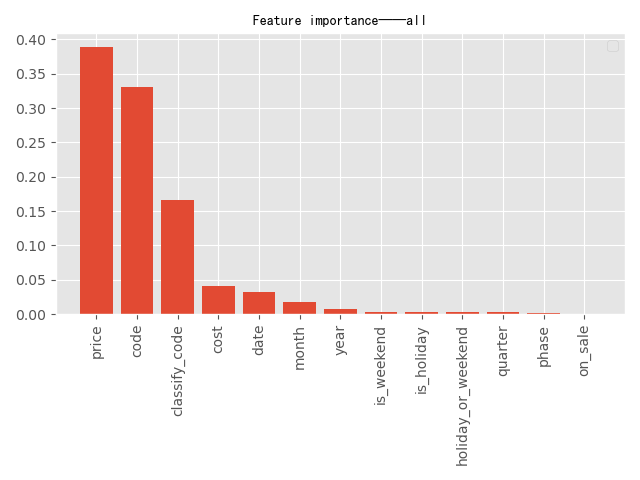
本文共构建了13个可能影响蔬菜单品日需求量的特征，为了评估这些特征的重要性，本文计算了它们的显著性程度，结果如图13所示。选取0.01为阈值进行特征筛选，可得影响蔬菜单品的需求量的重要特征为price, code, classify\_code, cost, date, month。

图 13影响蔬菜单品日需求量的特征重要性

Figure 13 The importance of characteristics affecting the daily demand for vegetable menu items.

## 4.2 不同种类蔬菜日需求量预测

### 4.2.1 构建RF预测模型

随机森林模型是一种灵活的集成算法，能够有效减少决策树过拟合的问题，精确度较高，被广泛应用于各种分类、回归问题中。本文利用RF方法进行不同种类蔬菜日需求量预测的建模分析，针对每一种类蔬菜日需求量数据，建立由4.1节筛选得到的重要特征与需求量之间的随机森林回归模型。

在建立RF回归模型时，本文以4：1的比例将数据集分别划分为训练集和测试集。另外，由于样本集较小，仅有1085组数据，为防止过拟合，增强模型的泛化能力，本文在模型训练过程采用了五折交叉验证(5-fold cross-validation)，将训练集数据平均分为五份，每次取其中四份数据进行训练，剩余一份数据作为验证集，重复五次，最后平均五次训练的结果，作为误差评估的结果。

在RF回归模型中，主要包括框架参数和决策树参数两种，框架参数主要设置随机森林模型的整体架构，如决策树个数(n\_estimators)、是否在构建每棵树时对样本进行随机采样(bootstrap)等，决策树参数设置每一棵树的结构，如决策树最大深度(max\_depth)、最大叶子节点数(max\_leaf\_nodes)、节点划分标准(criterion)、剪枝参数(ccp\_alpha)等。在构建树结构时，较大的剪枝参数意味着更容易生成一个简单的模型，以防止过拟合。本文在模型训练时，分别采用均方误差 “squared\_error”、 Friedman均方误差 “friedman\_mse” 和泊松偏差 “poisson” 作为划分节点标准，设置 “n\_ estimators” 取值分别为100、200、400和500，设置 “ccp\_alpha” 取值分别为0、0.1、0.2、0.3、0.4和0.5，其他参数默认。采用网格搜索法(Grid Search)进行超参数(Hyperparameter)寻优，分别计算平均绝对误差(Mean Absolute Error, MAE)、相对绝对误差(Relative Absolute Error, RAE)、均方误差( Mean Square Error, MSE)、相对平方误差(Relative Squared Error, RSE)和平方绝对百分比误差(Mean Absolute Percentage Error, MAPE)以评估模型。具体代码见附录二(2.4)和(2.7.1)。假设共有个样本，其真实label值为，其预测值为。其计算公式分别为式(4.2.1)-式(4.2.5)。最终得到每一蔬菜种类的最优RF模型的超参数（从左至右分别为 “ccp\_alpha”, “criterion”, “n\_ estimators”）和对应模型在训练过程的评价指标值，如表4所示。

表 4不同蔬菜种类日需求量的最优拟合RF模型参数及交叉验证评价指标值

Table 4 Optimal RF hyperparameters and cross-validation metrics for daily demand of different vegetable types

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 种类 | 最优RF模型参数 | MSE | MAE | RAE | RSE | MAPE |
| 辣椒类 | {0, 'poisson', 200} | 1517.2086 | 18.2874 | 0.5647 | 0.0032 | 0.2473 |
| 花叶类 | {0, 'poisson', 500} | 2391.8336 | 31.1300 | 0.5544 | 0.0025 | 0.1964 |
| 花菜类 | {0, 'poisson', 500} | 240.3084 | 10.6037 | 0.6486 | 0.0030 | 0.5271 |
| 茄类 | {0, 'poisson', 400} | 78.6275 | 5.7992 | 0.6144 | 0.0028 | 0.5251 |
| 水生根茎类 | {0, 'poisson', 100} | 447.1002 | 11.7678 | 0.5367 | 0.0026 | 0.5346 |
| 食用菌类 | {0.5, 'friedman\_mse', 100} | 1156.5641 | 18.9106 | 0.5951 | 0.0031 | 0.3926 |

基于上述最优模型，本文进一步进行测试集测试，得到最优RF模型在测试集上的表现状况，如表5，由表可得，花叶类日需求量最优RF模型在测试集上的评价指标均大于其交叉验证的评价指标值，说明该模型的泛化能力较差，出现了过拟合的问题，这可能与花叶类产品的需求量有较大的波动有关。而其他蔬菜种类日需求量的最优RF模型在测试集的评价指标值与其交叉验证的评价指标值相比，均更小或相差不大，由此可见，这些模型的泛化能力较好。

表 5不同蔬菜种类日需求量的最优拟合RF模型测试集评价指标值

Table 5 Evaluation metrics of the optimal RF model for daily demand of different vegetable types in the test set

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 种类 | MSE | MAE | RAE | RSE | MAPE |
| 辣椒类 | 849.6876 | 16.3427 | 0.4764 | 0.0017 | 0.2645 |
| 花叶类 | 5588.0801 | 34.3904 | 0.578 | 0.0027 | 0.2394 |
| 花菜类 | 269.7408 | 10.102 | 0.5878 | 0.0021 | 0.3807 |
| 茄类 | 82.7716 | 5.3712 | 0.563 | 0.0024 | 0.3056 |
| 水生根茎类 | 269.102 | 10.6509 | 0.5363 | 0.0019 | 0.5489 |
| 食用菌类 | 1115.0322 | 17.5864 | 0.5459 | 0.0022 | 0.4234 |

### 4.2.2 构建XGBoost预测模型

XGBoost模型使用并行计算、Weighted Quantile Sketch算法和缓存技术，具有较强的灵活性和可拓展性，即使面对复杂任务，也能有较高的准确性。本文利用Python中的xgboost库[20]对不同种类蔬菜日需求量进行建模分析。

在建立XGBoost回归模型时，数据的划分和RF模型相同。在设置模型参数时，主要包括通用参数(General parameters)、提升器参数(Booster parameters)和学习目标参数(Learning task parameters)。通用参数主要控制弱学习器模型的选择，如参数“booster”决定选择树模型或线性模型，提升器参数主要控制具体的弱学习器如何设置，如树提升器的学习率参数 “eta” 控制模型更新的步长，“max\_depth” 控制树的最大深度。学习目标参数指定相应的学习任务和学习目标，如参数 “objective” 可决定目标函数为带平方损失的回归 “reg:squarederror” 或逻辑回归的输出概率 “reg:logistic” 或其他。

本文选定提升器采用树模型 “gbtree” ，目标函数为 “reg:squarederror”，设置学习率“learning\_rate” 取值为0.01、0.05、0.1，弱学习器个数 “n\_ estimators” 取值分别为100、500、1000，训练样本的行抽样比例 “subsample” 取值分别为0.7、0.8、0.9、1.0，训练样本的特征抽样比例 “colsample\_bytree” 取值分别为0.8、0.9、1.0，设置用于离散化连续特征的最大分箱数 “max\_bin” 值分别为256、384、512，为防止过拟合，设置L2正则项系数 “reg\_lambda” 的取值分别为0、0.1、1、10，并依旧采用Grid Search寻找最优超参数，运用5-fold cross-validation训练模型，分别计算评价指标MAE、RAE、MSE、RSE、MAPE，选择最优模型。具体代码见附录二(2.5)和(2.7.2)。最终得到每一蔬菜种类的最优XGBoost模型的超参数(从左至右分别为 “colsample\_bytree”, “learning\_rate”, “max\_bin”, “n\_ estimators”, “reg\_lambda”, “subsample”)及其对应的评价指标值，如表6所示。

表 6不同蔬菜种类日需求量的最优拟合XGBoost模型参数及交叉验证评价指标值

Table 6 Optimal XGBoost hyperparameters and cross-validation metrics for daily demand of different vegetable types

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 种类 | 最优XGBoost模型参数 | MSE | MAE | RAE | RSE | MAPE |
| 辣椒类 | {1.0,0.05,256,1000,10,0.7} | 1491.3281 | 19.2232 | 0.5945 | 0.0031 | 0.2539 |
| 花叶类 | {0.9,0.1,256,100,10,0.8} | 2132.2606 | 29.4777 | 0.5240 | 0.0022 | 0.1888 |
| 花菜类 | {0.8,0.01,256,100,0,0.7} | 298.7549 | 12.3668 | 0.7559 | 0.0037 | 0.6553 |
| 茄类 | {0.8,0.05,256,500,0,08} | 75.4979 | 5.6428 | 0.5987 | 0.0027 | 0.5122 |
| 水生根茎类 | {0.8,0.01,256,1000,10,0.7} | 427.0606 | 11.3646 | 0.5182 | 0.0024 | 0.5181 |
| 食用菌类 | {0.8,0.05,384,100,1,0.7} | 1074.0414 | 18.2890 | 0.5755 | 0.0028 | 0.3471 |

基于上述最优模型，本文进一步进行测试集测试，得到最优XGBoost模型在测试集上的表现状况，如表7。对比表6和表7可得，除花叶类对应的最优模型的泛化能力较差，其他种类对应的最优模型的泛化能力均较好。

表 7不同蔬菜种类日需求量的最优拟合XGBoost模型测试集评价指标值

Table 7 Metrics of the optimal XGBoost model for daily demand of different vegetable types on the test set

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 种类 | MSE | MAE | RAE | RSE | MAPE |
| 辣椒类 | 755.0508 | 16.6538 | 0.4854 | 0.0015 | 0.2439 |
| 花叶类 | 5619.6669 | 33.4939 | 0.5629 | 0.0027 | 0.2355 |
| 花菜类 | 257.1898 | 9.7829 | 0.5692 | 0.002 | 0.3585 |
| 茄类 | 78.545 | 5.4592 | 0.5723 | 0.0023 | 0.3159 |
| 水生根茎类 | 264.1463 | 10.5303 | 0.5302 | 0.0018 | 0.5364 |
| 食用菌类 | 1043.7749 | 16.8578 | 0.5233 | 0.0021 | 0.3866 |

### 4.2.3 构建GRU预测模型

GRU模型能够较好的处理时间序列数据并学习数据中的历史信息，相比其它基于神经网络的深度学习模型，GRU模型具有架构简单、训练速度快、占用内存少等优点。本文利用Pytorch中的torch.nn模块构造GRU模型。构建步骤主要分为以下几步：

(1) 模型定义。构造继承父类nn.Moudle的GRUNet类，并定义属性，如输入维度 “input\_size”、隐藏层维度 “hidden\_size”、模型层数 “num\_layers”、随机失活率 “droupout” 等。由于数据特征既有分类特征也有连续特征且量纲不一致，为充分提取特征并增强模型的表达能力，本文利用torch.nn.linear函数创建全连接层(Fully connected layer) ，记作fc1，将输入数据进行映射，并采用torch.nn.BatchNorm1d函数进行批归一化，降低模型对参数初始化的要求。最后，再构建全连接层，记作fc2，将GRU 的隐藏状态映射为最终的预测结果。

(2) 模型初始化。本文采用 Xavier均匀初始化来初始化fc1和fc2的权重，即用正交化矩阵填充矩阵参数，并将偏置项初始化为0，以避免初始偏置项对模型学习的影响，从而提高模型的稳定性和收敛速度。

(3) 模型训练与评估。本文将样本集的60%划分为训练集，20%划分为验证集，20%划分为测试集。设置损失函数为MSE，并通过Adamax优化器进行反向传播更新权重矩阵参数以获得最优参数，同时设置早停机制，在模型无法被更最优后停止训练，防止模型过拟合。

本文对不同蔬菜种类的日需求量数据分别采用GRU模型进行拟合，得到不同种类蔬菜需求量对应的最优模型在验证集上和测试集上的评价指标值，如表8和表9。由表可得，构建的GRU模型出现了过拟合现象，其泛化能力较差。具体代码见附录二(2.6)和(2.7.3)。

表 8不同蔬菜种类日需求量的最优拟合GRU模型验证集评价指标值

Table 8 Metrics of the optimal GRU model for daily demand of different vegetable types on the validation set

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 种类 | MSE | MAE | RAE | RSE | MAPE |
| 辣椒类 | 1871.8431 | 30.2450 | 1.0001 | 0.0046 | 0.5360 |
| 花叶类 | 5444.0151 | 54.6080 | 0.9999 | 0.0046 | 0.4090 |
| 花菜类 | 397.7312 | 15.1626 | 1.0000 | 0.0047 | 0.9054 |
| 茄类 | 133.5575 | 8.9571 | 0.9973 | 0.0048 | 0.6689 |
| 水生根茎类 | 652.9751 | 20.3238 | 1.0000 | 0.0046 | 1.1468 |
| 食用菌类 | 1575.8275 | 30.3003 | 1.0002 | 0.0046 | 0.7537 |

表 9不同蔬菜种类日需求量的最优拟合GRU模型测试集评价指标值

Table 9 Evaluation metrics of the optimal fitted GRU model for daily demand of different vegetable types on the test set

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 种类 | MSE | MAE | RAE | RSE | MAPE |
| 辣椒类 | 2292.8066 | 30.9031 | 0.9756 | 0.0047 | 0.4878 |
| 花叶类 | 5365.0913 | 55.9133 | 1.0378 | 0.0047 | 0.4595 |
| 花菜类 | 417.0596 | 15.6393 | 1.0297 | 0.0047 | 0.9225 |
| 茄类 | 177.7924 | 9.7485 | 0.9819 | 0.0048 | 0.7103 |
| 水生根茎类 | 925.5318 | 20.9999 | 1.0113 | 0.0046 | 1.5331 |
| 食用菌类 | 2207.4001 | 30.7714 | 0.9926 | 0.0046 | 0.644 |

4.2.4 预测结果

经过综合分析，XGBoost模型在模型训练和测试阶段均表现出最优的性能，以六类蔬菜对应的最优模型MSE均值为例，XGBoost、RF和GRU的均值分别为1336.396、1362.402和1897.614。表明该模型和RF模型及GRU模型相比，具有更强的泛化能力。

据此，本文以XGBoost模型对未来7日不同种类蔬菜的日需求量进行预测，针对未来7日产品的成本与价格，采用滑动平均法，始终以前7天的均值作为当天特征值的估计值，最终得到预测结果，如表10。具体代码见附录二(2.7.2)。未来七天内，花叶类产品日均需求量最高，约163斤，茄类最低，约20斤。辣椒类、花菜类和食用菌蔬菜产品需求量整体呈现出下降趋势，其中辣椒类产品的日需求量下降幅度最大，约20%，花菜类和食用菌降幅分别为13%和17%。而花叶类和水生根茎类产品日需求量呈逐步、小幅度上升的趋势，涨幅分别约为3%和5%。茄类蔬菜产品的日需求量整体始终在20斤附近波动，并在未来第三天的需求量达到最高为22.9098斤。

表 10不同蔬菜种类未来7日每天需求量的预测结果（单位：斤）

Table 10 Forecast results for the daily demand of different types of vegetables over the next 7 days.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 种类 | pred\_1 | pred\_2 | pred\_3 | pred\_4 | pred\_5 | pred\_6 | pred\_7 |
| 辣椒类 | 82.0688 | 82.8005 | 81.7094 | 78.8456 | 79.8265 | 71.3576 | 65.1901 |
| 花叶类 | 158.0879 | 158.0879 | 163.6384 | 164.9210 | 164.6979 | 167.2533 | 163.2909 |
| 花菜类 | 39.3039 | 37.1201 | 37.2368 | 36.0309 | 36.0491 | 34.3880 | 34.1917 |
| 茄类 | 17.3719 | 21.1942 | 20.7460 | 22.5077 | 18.6201 | 18.6899 | 19.2346 |
| 水生根茎类 | 21.6683 | 22.0248 | 22.9098 | 22.4518 | 22.6603 | 22.9634 | 23.1220 |
| 食用菌 | 53.1569 | 49.2614 | 49.2434 | 47.9720 | 47.4274 | 51.6411 | 44.0558 |

## 4.3 不同蔬菜单品日需求量预测

4.3.1 构建RF预测模型

本文针对251个蔬菜单品采用RF模型构建其日需求量预测模型。通过五折交叉验证和网格搜索，最终得到最优RF模型的超参数为{'ccp\_alpha': 0, 'criterion': 'poisson', 'n\_estimators': 1000}，其在交叉验证过程中和在测试集上的评价指标值如表11所示。由表可得，该模型在测试集上的评价指标值和交叉验证的评价指标值相比，均较小或相差不大，因此可得该模型的泛化能力较好，稳定性较强。具体代码见附录二(2.4)。

表 11 每个蔬菜单品日需求量的最优拟合RF模型交叉验证和测试集评价指标值

Table 11 Metrics for the optimal model for daily demand of each vegetable dish on cross-validation and test set

|  |  |  |
| --- | --- | --- |
| 评估指标 | 交叉验证结果 | 测试集结果 |
| MSE | 0.0458 | 0.0539 |
| MAE | 0.1215 | 0.1210 |
| RAE | 0.4606 | 0.4580 |
| RSE | 0.0000 | 0.0000 |
| MAPE | 0.3415 | 0.3381 |

### 4.3.2 构建XGBoost预测模型

针对每个蔬菜单品，本文也采用XGBoost模型构建其日需求量预测模型，得到最优XGBoost模型的超参数为{'colsample\_bytree': 0.8, 'learning\_rate': 0.1, 'max\_bin': 384, 'n\_estimators': 1500, 'reg\_lambda': 10, 'subsample': 1.0}，其在交叉验证过程中和在测试集上的评价指标值，如表12所示。由表可得，模型在测试集上的评估指标值均比其在交叉验证过程中的指标值更小或相差不大，说明该模型的泛化能力也较好。具体代码见附录二(2.5)。

表 12 每个蔬菜单品日需求量的最优拟合XGBoost模型交叉验证和测试集评价指标值

Table 12 Metrics for the optimal model for daily demand of each vegetable dish on cross-validation and test set

|  |  |  |
| --- | --- | --- |
| 评价指标 | 交叉验证结果 | 测试集结果 |
| MSE | 0.0443 | 0.0529 |
| MAE | 0.1211 | 0.1213 |
| RAE | 0.4589 | 0.4592 |
| RSE | 0.0000 | 0.0000 |
| MAPE | 0.3408 | 0.3389 |

### 4.3.3 预测结果

综上所述，XGBoost模型在各项评估指标上均优于或与RF模型相差不大。鉴于XGBoost模型在部署上的优势，本文最终选择该模型去预测未来7天各蔬菜单品的日需求量。在预测的过程中，考虑到蔬菜产品具有时令性的特点，若预测每一蔬菜单品未来7天的日需求量，可能会出现对非应季产品进行预测的情况，这在实际生活中是不需要的；若仅依赖前七天的销售数据进行预测，许多单品可能仅出现几次，这将不利于稳定数值的滑动平均计算，从而对需求量的预测产生影响。为了克服这些问题，本文首先对最近一个月消费者对蔬菜单品的需求情况进行了筛选，接着选取出现次数超过7次的蔬菜单品，最后利用这些单品最后7次出现时的日平均价格和成本，通过滑动平均依次估计出其未来7天的价格和成本。基于XGBoost的最优参数模型得到46种蔬菜单品未来7天的预测结果。表13仅展示了其中11种蔬菜单品的需求量，全部预测结果可见附录一。

表 13 未来7天每个蔬菜单品日需求量的预测结果（单位：斤）

Table 13 Forecast results for the daily demand of each vegetable dish over the next 7 days

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| classify\_name | code\_name | pred\_1 | pred\_2 | pred\_3 | pred\_4 | pred\_5 | pred\_6 | pred\_7 |
| 花叶类 | 菜心 | 0.4328 | 0.5175 | 0.4699 | 0.4830 | 0.4117 | 0.3977 | 0.4677 |
| 花叶类 | 苋菜 | 0.5071 | 0.4693 | 0.4390 | 0.5130 | 0.5292 | 0.5278 | 0.5128 |
| 茄类 | 青茄子(1) | 0.6252 | 0.6155 | 0.6276 | 0.6188 | 0.6147 | 0.6011 | 0.6065 |
| 茄类 | 紫茄子(2) | 0.5360 | 0.5170 | 0.5203 | 0.5099 | 0.5287 | 0.5026 | 0.5071 |
| 水生根茎类 | 净藕(1) | 0.4189 | 0.4395 | 0.4334 | 0.4337 | 0.4283 | 0.4101 | 0.4160 |
| 水生根茎类 | 红莲藕带 | 0.3100 | 0.3223 | 0.3271 | 0.3105 | 0.3118 | 0.3072 | 0.3084 |
| 食用菌 | 白玉菇(袋) | 0.8570 | 0.9686 | 0.8699 | 0.9936 | 0.8904 | 0.9260 | 0.8921 |
| 食用菌 | 西峡花菇(1) | 0.1613 | 0.1778 | 0.1789 | 0.1687 | 0.1768 | 0.1696 | 0.1744 |
| 辣椒类 | 小皱皮(份) | 0.9697 | 0.9709 | 0.9795 | 0.9704 | 0.9730 | 0.9819 | 0.9783 |
| 辣椒类 | 青线椒(份) | 0.9698 | 0.9851 | 0.9727 | 0.9990 | 1.0013 | 0.9879 | 1.0092 |
| 花菜类 | 西兰花 | 0.4256 | 0.4164 | 0.4267 | 0.4123 | 0.3944 | 0.3805 | 0.4200 |

本章使用Python编写代码，所涉及的函数及功能见表14。具体代码见附录二(2.3)至(2.7)。

表 14本章所涉及的函数及功能

Table 14 The Python functions and corresponding functionalities designed in this chapter

|  |  |
| --- | --- |
| 函数名 | 功能 |
| matplotlib.pyplot | 绘图 |
| xgboost.XGBRegressor | 构建xgboost回归模型 |
| sklearn.SelectFromModel | 删除重要性小于阈值的特征 |
| sklearn.train\_test\_split | 划分数据集 |
| sklearn.ensemble.RandomForestRegressor | 构建随机森林回归模型 |
| sklearn.model\_selection.GridSearchCV | 网格搜索寻找最优超参数 |
| sklearn.metrics.mean\_squared\_error | 计算MSE |
| sklearn.metrics.mean\_absolute\_error | 计算MAE |
| sklearn.metrics.mean\_absolute\_percentage\_error | 计算MAPE |
| torch.optim.Adamax | 实现Adamax优化方法 |
| torch.nn.linear | 设置神经网络中的线性层 |

# 第五章 结论与展望

## 5.1 结论

本文通过对比分析不同粒度下蔬菜类产品需求量的变化特征，研究某商超蔬菜类产品在大类和单品两个层面的日需求量预测问题。将分析粒度从大类细化至单品，有助于更深入理解商品特性，提升预测精度。基于2020年7月1日至2023年6月30日的需求数据，本文构建了13个特征指标，并对其进行了描述性统计分析，在此基础上，筛选出关键影响因子，进一步采用多种机器学习和深度学习模型进行建模与评估，具体结论如下。

在2021-2022年，蔬菜类产品可能因受到疫情的影响，从而出现整体需求量大幅下滑的现象，直至2022年7月，才逐步回升至正常水平。同时，不同种类蔬菜的需求量分布情况也大不相同，花叶类蔬菜产品的需求量始终高于其他产品的需求量。不同种类的蔬菜之间也具有一定的联系，如花叶类和花菜类蔬菜产品之间互为互补品，食用菌和水生根茎类产品之间互为替代品等。

在不同种类蔬菜日需求量的建模与预测中，分别采用RF、XGBoost和GRU三种模型，对相关数据进行拟合训练，并通过超参数调优获取各自的最优模型。基于MSE、MAE、RAE、RSE和MAPE五种评价指标的比较结果显示，XGBoost模型在拟合效果和泛化能力方面表现最佳，其次为随机森林模型，而GRU模型出现过拟合，预测效果和泛化能力相对较差。以六类蔬菜对应的最优模型MSE均值为例，XGBoost、RF和GRU的均值分别为1336.396、1362.402和1897.614。据此，本文选用XGBoost模型对不同蔬菜种类未来7天的日需求量进行预测。结果显示，花叶类产品日均需求量最高，约163斤，茄类最低，约20斤。辣椒类、花菜类和食用菌呈下降趋势，其中辣椒类降幅最大，约21%，其未来七天日需求量（单位：斤）预测分别为82.07、82.80、81.71、78.85、79.83、71.36、65.19；花菜类和食用菌降幅分别为13%和17%。相比之下，花叶类和水生根茎类需求呈上升趋势，涨幅分别约为3%和5%；茄类产品则整体稳定在20斤左右。

在单品层面，采用RF和XGBoost两种模型进行建模，调整超参数并通过MAE、MSE、RSE等指标进行评价，最终确定XGBoost为最优模型。选取近一个月内出现频次超过7次的蔬菜单品，基于该模型预测其未来7天的日需求量。

## 5.2 创新点

(1) 大多数研究在进行产品影响分析时未能充分考虑产品本身性质以及时间等因素，而本文在需求影响分析时将引入产品所属大类、节假日、成本、价格和季节等因素，能够较精确的探究影响产品需求量的重要因素。

(2) 拟基于机器学习算法和深度学习算法构建多种模型进行产品预测，更有利于选择出适合蔬菜产品需求量预测的最优模型。

## 5.3 建议与展望

根据本研究的结论，考虑到当前消费者对蔬菜类产品的高品质需求以及市场竞争的激烈程度，本文建议通过优化商品陈列和区域布局来促进消费者对蔬菜产品的消费需求。具体而言，应将新鲜蔬菜摆放在入口或显眼的位置，并注意不同蔬菜产品的分区，按照蔬菜单品所属的大类进行分类摆放，以便顾客能够快速选购。同时，应丰富产品种类结构，强化品质管理，针对消费者的偏好引入更多蔬菜品类，例如增加花菜类产品的供应。此外，建议引入本地时令蔬菜以及绿色、有机等高附加值产品，以满足多层次的消费需求。

虽然基于不同种类蔬菜产品和每一蔬菜单品的日需求量数据，运用模型表现出较优的预测效果，但本文研究仍存在较多不足，未来可考虑从以下两个方面进行改进。

(1) 构建更加丰富的特征指标集。蔬菜为生鲜类产品，较容易收到损耗，且随着“轻食”观念的普及，蔬菜产品的热量也成为影响消费者对其需求量的一个重要因素。在未来的研究中需要尽可能量化更多的影响因素指标，将其加入模型，增加模型预测的准确性。

(2) 进一步探索和应用多元化的模型方法，积极引入和评估其他前沿的预测模型，同时深入研究模型组合方法，通过融合不同模型的优势，构建具有更高鲁棒性和预测能力的模型，为蔬菜产品需求量预测提供更先进的模型方法。

# 参考文献

1. 张亚敏.基于酒店收益管理的需求预测研究综述[J].科技创新与生产力,2019(07):7-12.
2. ALI M, Boylan E J. On the effect of non-optimal forecasting methods on supply chain downstream demand[J]. IMA journal of management mathematics, 2012, 23(1): 81-98.
3. E. C V, Matheus Z, Fidellis B.G.L et al.Analysis of time series models for Brazilian electricity demand forecasting[J]. Energy, 2022, 247: 123483.
4. Ramanathan, Usha. Supply chain collaboration for improved forecast accuracy of promotional sales[J]. International Journal of Operations & Production Management, 2012, 32(6): 676-695.
5. Kourentzes N, Petropoulos F. Forecasting with multivariate temporal aggregation: The case of promotional modelling[J]. International Journal of Production Economics, 2016, 181(A): 145-153.
6. 程元栋, 喻可欣, 李先洋. 基于加权马尔科夫-ARIMA修正模型的区域物流需求预测[J]. 山东交通学院学报, 2023, 31(03): 22-28.
7. Quinlan, J. R. Induction of Decision Trees[J]. Mach Learn 1986, 1 (1): 81–106.
8. 刘艳丽. 随机森林综述[D]. 天津: 南开大学, 2008.
9. Jerome F, Trevor H, Robert T. Additive Logistic Regression: A Statistical View of Boosting[J]. The Annals of Statistics, 2000, 28(2): 337-374.
10. Huber J, Stuckenschmidt H. Daily retail demand forecasting using machine learning with emphasis on calendric special days[J]. International Journal of Forecasting, 2020, 36(4): 1420-1438.
11. 庆豪, 方志耕, 王育红 等. 基于灰色-神经网络的民机需求组合预测[J]. 系统工程与电子技术, 2024, 46(05): 1665-1672.

范华鹏, 刘畅, 程锋云. 基于不同机器学习分类算法的滇西北森林碳储量估测[J/OL]. 西南林业大学学报(自然科学), 2025, 4: 1-9.

周雅夫, 李瑞洁, 侯代峥. 融合注意力机制的CNN-GRU燃料电池老化趋势预测[J]. 重庆理工大学学报(自然科学), 2024, 38(09): 106-112.

1. 徐慧智, 杨冰冰. 基于CNN-LSTM组合模型的铁路枢纽站出租车需求量短时预测研究[J]. 大连交通大学学报, 2023, 44(01): 58-63.
2. Abbasimehr H, Shabani M, Yousefi M. An optimized model using LSTM network for demand forecasting[J]. Computers & Industrial Engineering, 2020, 143(C): 106435.

黄欣, 赵敏彤, 郇嘉嘉 等. 基于EEMD-GRU神经网络的天然气价格预测模型构建[J]. 微型电脑应用, 2024, 40(09): 13-17.

1. 孙光彩.云南省蔬菜价格波动问题研究[D].云南农业大学, 2017.
2. Dietterich T G. Ensemble Methods in Machine Learning[M/OL]. Multiple Classifier Systems. Springer B.H.:Lecture Notes in Computer Science., 2000, 1857: 1-15.
3. Panda B, Herbach J S, Basu S et al. PLANET: massively parallel learning of tree ensembles with MapReduce[J]. Proc VLDB Endow, 2009, 2(2): 1426-1437.
4. Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System[C/OL]. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016: 785-794.
5. Cho K, Van Merrienboer B, Gulcehre C et al. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation[C/OL]. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, 2014:1724-1734.
6. Wickham H, Averick M, Bryan J et al. “Welcome to the tidyverse”[J]. Journal of Open Source Software, 2019, 4(43): 1686.
7. Fabian P, Gaël V, Alexandre G et al*.* Scikit-learn: Machine Learning in Python[J]. JMLR, 2011, 12: 2825-2830.

# 附 录

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 一、蔬菜单品未来7天日需求量预测   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 所属种类名称 | 单品名称 | pred\_1 | pred\_2 | pred\_3 | pred\_4 | pred\_5 | pred\_6 | pred\_7 | | 花叶类 | 上海青 | 0.4354 | 0.4325 | 0.4466 | 0.4404 | 0.4401 | 0.4397 | 0.4443 | | 花叶类 | 菜心 | 0.4328 | 0.5175 | 0.4699 | 0.4830 | 0.4117 | 0.3977 | 0.4677 | | 花叶类 | 云南生菜 | 0.4164 | 0.3803 | 0.3795 | 0.3918 | 0.3790 | 0.3718 | 0.3709 | | 水生根茎类 | 高瓜(1) | 0.3515 | 0.3395 | 0.3226 | 0.3456 | 0.3350 | 0.3150 | 0.2961 | | 花叶类 | 苋菜 | 0.5071 | 0.4693 | 0.4390 | 0.5130 | 0.5292 | 0.5278 | 0.5128 | | 花菜类 | 西兰花 | 0.4256 | 0.4164 | 0.4267 | 0.4123 | 0.3944 | 0.3805 | 0.4200 | | 水生根茎类 | 洪湖藕带 | 0.3591 | 0.3759 | 0.3769 | 0.3628 | 0.3717 | 0.3659 | 0.3705 | | 花叶类 | 竹叶菜 | 0.4304 | 0.4039 | 0.3688 | 0.4438 | 0.4592 | 0.4527 | 0.4418 | | 茄类 | 青茄子(1) | 0.6252 | 0.6155 | 0.6276 | 0.6188 | 0.6147 | 0.6011 | 0.6065 | | 花叶类 | 木耳菜 | 0.4139 | 0.4202 | 0.4412 | 0.4277 | 0.5067 | 0.5059 | 0.3975 | | 花叶类 | 娃娃菜 | 1.0036 | 0.9902 | 0.9997 | 0.9897 | 0.9903 | 0.9865 | 0.9940 | | 花叶类 | 红薯尖 | 0.4242 | 0.3860 | 0.3823 | 0.3797 | 0.4098 | 0.4009 | 0.3707 | | 辣椒类 | 螺丝椒 | 0.2749 | 0.2767 | 0.3229 | 0.3124 | 0.2768 | 0.2517 | 0.2589 | | 茄类 | 紫茄子(2) | 0.5360 | 0.5170 | 0.5203 | 0.5099 | 0.5287 | 0.5026 | 0.5071 | | 水生根茎类 | 净藕(1) | 0.4189 | 0.4395 | 0.4334 | 0.4337 | 0.4283 | 0.4101 | 0.4160 | | 水生根茎类 | 红莲藕带 | 0.3100 | 0.3223 | 0.3271 | 0.3105 | 0.3118 | 0.3072 | 0.3084 | | 食用菌 | 白玉菇(袋) | 0.8570 | 0.9686 | 0.8699 | 0.9936 | 0.8904 | 0.9260 | 0.8921 | | 水生根茎类 | 菱角 | 0.4194 | 0.4344 | 0.4372 | 0.4192 | 0.4201 | 0.4034 | 0.4035 | | 花叶类 | 奶白菜 | 0.4220 | 0.4207 | 0.4603 | 0.4455 | 0.4383 | 0.4653 | 0.4293 | | 茄类 | 圆茄子(2) | 0.7351 | 0.5902 | 0.5896 | 0.6230 | 0.7475 | 0.5910 | 0.6094 | | 辣椒类 | 芜湖青椒(1) | 0.3934 | 0.3923 | 0.3850 | 0.3863 | 0.3699 | 0.3508 | 0.3519 | | 食用菌 | 西峡花菇(1) | 0.1613 | 0.1778 | 0.1789 | 0.1687 | 0.1768 | 0.1696 | 0.1744 | | 茄类 | 长线茄 | 0.4052 | 0.4131 | 0.3874 | 0.4082 | 0.4127 | 0.4037 | 0.3868 | | 花叶类 | 小青菜(1) | 0.3919 | 0.3916 | 0.3918 | 0.3894 | 0.3964 | 0.3886 | 0.3977 | | 食用菌 | 海鲜菇(包) | 1.0013 | 1.0036 | 0.9750 | 0.9947 | 0.9946 | 0.9453 | 0.9400 | | 花叶类 | 菠菜(份) | 0.9557 | 0.9627 | 0.9527 | 0.9213 | 0.9193 | 0.9108 | 0.9263 | | 花叶类 | 云南油麦菜(份) | 0.9790 | 0.9824 | 0.9819 | 0.9771 | 0.9755 | 0.9746 | 0.9760 | | 花叶类 | 云南生菜(份) | 0.9918 | 1.0007 | 0.9927 | 0.9591 | 0.9533 | 0.9524 | 0.9539 | | 辣椒类 | 小米椒(份) | 0.9821 | 0.9813 | 0.9740 | 0.9768 | 0.9740 | 0.9753 | 0.9802 | | 食用菌 | 金针菇(盒) | 1.0277 | 1.0437 | 1.1445 | 1.0558 | 1.0260 | 1.0145 | 1.0225 | | 食用菌 | 虫草花(份) | 1.0204 | 0.9921 | 0.9743 | 0.9713 | 0.9770 | 0.9746 | 0.9705 | | 辣椒类 | 小皱皮(份) | 0.9697 | 0.9709 | 0.9795 | 0.9704 | 0.9730 | 0.9819 | 0.9783 | | 辣椒类 | 青线椒(份) | 0.9698 | 0.9851 | 0.9727 | 0.9990 | 1.0013 | 0.9879 | 1.0092 | | 辣椒类 | 螺丝椒(份) | 0.9648 | 0.9557 | 0.9675 | 0.9991 | 0.9613 | 0.9897 | 0.9986 | | 辣椒类 | 七彩椒(2) | 0.3805 | 0.3915 | 0.3420 | 0.3270 | 0.3689 | 0.3698 | 0.3585 | | 水生根茎类 | 高瓜(2) | 0.2574 | 0.2688 | 0.2159 | 0.2190 | 0.2240 | 0.2150 | 0.2430 | | 辣椒类 | 姜蒜小米椒组合装(小份) | 0.9450 | 0.9573 | 0.9212 | 0.9374 | 0.9348 | 0.9337 | 0.9561 | | 花叶类 | 黄心菜(2) | 0.5863 | 0.5749 | 0.5619 | 0.5717 | 0.5656 | 0.5646 | 0.5753 | | 食用菌 | 双孢菇(盒) | 1.0000 | 0.9998 | 0.9832 | 0.9846 | 0.9792 | 0.9940 | 1.0064 | | 花菜类 | 枝江青梗散花 | 0.5880 | 0.6349 | 0.5996 | 0.5840 | 0.6034 | 0.5835 | 0.5783 | | 辣椒类 | 青红杭椒组合装(份) | 0.9426 | 0.9681 | 0.9389 | 0.9717 | 0.9818 | 1.0126 | 0.9905 | | 茄类 | 紫茄子(1) | 0.6067 | 0.5690 | 0.6295 | 0.6361 | 0.6500 | 0.6283 | 0.6653 | | 辣椒类 | 红椒(2) | 0.3044 | 0.3044 | 0.3002 | 0.2923 | 0.2914 | 0.2841 | 0.2768 | | 食用菌 | 蟹味菇与白玉菇双拼(盒) | 0.9359 | 1.0509 | 0.9222 | 0.9149 | 0.9019 | 0.8777 | 1.0117 | | 花叶类 | 鲜粽叶(袋)(3) | 1.1064 | 1.1156 | 1.1005 | 1.0984 | 1.1057 | 1.0892 | 1.0950 | | 花叶类 | 木耳菜(份) | 1.0293 | 0.9194 | 1.0698 | 0.9815 | 0.9114 | 0.9292 | 0.9770 |   二、代码  2.1 数据处理  library(tidyverse)  library(readxl)  library(proxy)  library(e1071)  library(ggthemes)  library(scales)  single\_info <- read\_excel("附件1.xlsx")  data <- read\_excel("附件2.xlsx")  single\_cost <- read\_excel("附件3.xlsx")  ## 1) 整合数据  data <- data %>%  rename("日期" = "销售日期") %>%  left\_join(single\_info %>%  select(单品编码, 分类编码, 单品名称, 分类名称),  by = "单品编码") %>%  left\_join(single\_cost %>%  select(单品编码, 日期, `批发价格(元/千克)`),  by = c("单品编码", "日期")) %>%  filter(销售类型 != "退货") %>%  rename(all\_date = 日期, code = 单品编码, qty = `需求量(千克)`,  price = `销售单价(元/千克)`, on\_sale = 是否打折销售,  code\_name = 单品名称, classify\_code = 分类编码,  classify\_name = 分类名称, cost = `批发价格(元/千克)`) %>%  mutate(all\_date = ymd(all\_date), year = year(all\_date),  month = month(all\_date), date = day(all\_date)) %>%  select(-销售类型)  ## 2) 数据检查与处理  ### 1 缺失值检查(无缺失值)  missing\_summary <- data %>%  summarise(across(everything(), ~sum(is.na(.)))) %>%  pivot\_longer(everything(), names\_to = "column", values\_to = "missing\_count") %>%  filter(missing\_count > 0)  ### 2 NAN检查与处理（无缺失值NAN）  nan\_summary <- data %>%  summarise(across(everything(), ~sum(is.nan(.)))) %>%  pivot\_longer(everything(), names\_to = "column", values\_to = "missing\_count") %>%  filter(missing\_count > 0)  ### 3 重复值检查与处理(无重复值)  data <- data %>%  distinct(.keep\_all = TRUE)  ### 4 异常值处理  data <- data %>%  filter(!(on\_sale == "否" & price < cost))  data <- data %>%  filter(!(qty==160))  ### 5 查看在同一天中同一产品的成本和售价是否一样  n\_price\_cost <- data %>%  group\_by(code, all\_date) %>%  summarise(  unique\_price\_count = n\_distinct(price),  unique\_cost\_count = n\_distinct(cost),  .drop = TRUE  )  #### a 成本是不变的，但是部分产品的售价在一天中是变换的  #### 针对同一天中产品price变换的情况下，只保留price出现次数最多的数据  price\_change\_delete <- n\_price\_cost %>%  filter(unique\_price\_count != 1) %>%  inner\_join(data, by = c("code" = "code", "all\_date" = "all\_date")) %>%  group\_by(code, all\_date, price) %>%  summarise(count = n(), .groups = 'drop') %>%  arrange(code, all\_date, desc(count)) %>%  group\_by(code, all\_date) %>%  slice(-1) %>%  ungroup()  data <- data %>%  anti\_join(price\_change\_delete, by = c("code", "all\_date", "price"))  ### 6 将分类变量转换为因子类型，并将 on\_sale 转换为 0/1 (是==1，否==0)  data <- data %>%  mutate(  on\_sale = ifelse(on\_sale == "是", 1, 0),  across(c(classify\_name, classify\_code, code\_name, code,  on\_sale, year, month, date), as\_factor),  across(c(classify\_code, code), as.numeric)  )  ### 7 去除不必要的列  data <- data %>%  select(-c(扫码销售时间))## 3) 提取特征  data <- data %>%  mutate(  # 季度特征  quarter = case\_when(  month %in% c(1, 2, 3) ~ 0,  month %in% c(4, 5, 6) ~ 1,  month %in% c(7, 8, 9) ~ 2,  month %in% c(10, 11, 12) ~ 3  ),  # 周末  is\_weekend = ifelse(wday(all\_date) %in% c(6, 7), 0, 1),  # 旬  phase = case\_when(  date %in% c(1:10) ~ 0,  date %in% c(11:20) ~ 1,  date %in% c(21:31) ~ 2  )  )  write.csv(data, "data.csv", row.names = FALSE)  data <- read\_csv("data.csv")  # 改变顺序  data <- data %>%  mutate(holiday\_or\_weekend = ifelse(is\_holiday == 0 | is\_weekend == 0, 0, 1)) %>%  relocate(all\_date, year, month, date, classify\_name, classify\_code,  code\_name, code, cost, price, qty, on\_sale, quarter,  phase, is\_weekend, is\_holiday, holiday\_or\_weekend)  save\_classify <- function(data, classify) {  filtered\_data <- data %>%  filter(classify\_code == classify)  # 日  sum\_day <- filtered\_data %>%  group\_by(all\_date) %>%  summarise(  total\_qty = sum(qty, na.rm = TRUE),  total\_cost = weighted.mean(cost, qty),  total\_price = weighted.mean(price, qty),  total\_on\_sale = mean(on\_sale),  .groups = "drop")  result\_day <- left\_join(  filtered\_data %>%  select(-c(classify\_name, classify\_code, code\_name, code, cost, price, qty, on\_sale)) %>%  distinct(),  sum\_day, by = c("all\_date")) %>%  select(-all\_date) %>%  relocate(total\_qty, .after = last\_col())  write.csv(result\_day,  file = paste0("classify\_", classify, "\_dataset\_day.csv"),  row.names = FALSE)  }  lapply(1:6, function(x) save\_classify(data, x))  data <- read.csv("data.csv")  2.2 描述统计——可视化  #每月所有蔬菜的总销售量  month\_total\_qty <- data %>%  mutate(year\_month = as.Date(paste(year, month, '01', sep = "-")))%>%  select(-c(year,month)) %>%  group\_by(year\_month) %>%  summarise(month\_qty = sum(qty),  .groups = 'drop')  ggplot(month\_total\_qty, aes(x = year\_month, y = month\_qty)) +  geom\_area(  fill = "#AED6F1",  alpha = 0.4  ) +  geom\_line(  color = "#1B4F72",  linewidth = 2  ) +  geom\_point(  color = "#1B4F72",  fill = "#F1948A",  size = 3.5,  shape = 21,  stroke = 1.2  ) +  scale\_x\_date(  date\_labels = "%Y-%m",  date\_breaks = "2 months",  expand = expansion(mult = c(0.02, 0.02))  ) +  scale\_y\_continuous(  expand = expansion(mult = c(0, 0.05))  ) +  labs(  title = "每月蔬菜总需求量变化趋势",  subtitle = "数据周期：2020年7月 - 2023年6月",  x = "时间（年月）",  y = "需求量总和（单位：斤）"  ) +  theme\_minimal(base\_size = 16) +  theme(  plot.title = element\_text(  hjust = 0.5, face = "bold", size = 22, color = "#2C3E50"  ),  plot.subtitle = element\_text(  hjust = 0.5, size = 16, color = "gray40"  ),  plot.caption = element\_text(  hjust = 1, size = 10, color = "gray50"  ),  axis.title.x = element\_text(  margin = margin(t = 10), face = "bold"  ),  axis.title.y = element\_text(  margin = margin(r = 10), face = "bold"  ),  axis.text.x = element\_text(  angle = 45, hjust = 1, vjust = 1, size = 12, color = "#34495E"  ),  axis.text.y = element\_text(  size = 12, color = "#34495E"  ),  panel.grid.major.x = element\_blank(),  panel.grid.minor = element\_blank(),  panel.grid.major.y = element\_line(  color = "gray85", linetype = "dashed"  ),  plot.background = element\_rect(  fill = "white", color = NA  ),  panel.background = element\_rect(  fill = "white", color = NA  )  )  # 六大总类的月总需求量随时间变换的图像  month\_qty <- data %>%  mutate(year\_month = as.Date(paste(year, month, '01', sep = "-")))%>%  select(-c(year,month)) %>%  group\_by(classify\_name, year\_month) %>%  summarise(month\_qty = sum(qty),  .groups = 'drop')  ggplot(month\_qty, aes(x = year\_month, y = month\_qty, color = classify\_name)) +  geom\_line(linewidth = 1.5) +  geom\_point(size = 3, shape = 21, stroke = 1.2, fill = "white") +  scale\_color\_brewer(palette = "Set1") +  scale\_x\_date(  limits = as.Date(c("2020-07-01", "2023-07-01")),  date\_labels = "%Y-%m",  date\_breaks = "3 months",  expand = expansion(mult = c(0.02, 0.02))  ) +  scale\_y\_continuous(  expand = expansion(mult = c(0, 0.05))  ) +  labs(  title = "六大品类的月总需求量变化趋势",  subtitle = "数据周期：2020年7月 - 2023年6月",  x = "时间（年月）",  y = "月需求量（单位：斤）",  color = "产品大类"  ) +  theme\_minimal(base\_size = 16) +  theme(  plot.title = element\_text(  hjust = 0.5, face = "bold", size = 22, color = "#2C3E50"  ),  plot.subtitle = element\_text(  hjust = 0.5, size = 16, color = "gray40"  ),  plot.caption = element\_text(  hjust = 1, size = 10, color = "gray50"  ),  axis.title.x = element\_text(  margin = margin(t = 10), face = "bold"  ),  axis.title.y = element\_text(  margin = margin(r = 10), face = "bold"  ),  axis.text.x = element\_text(  angle = 45, hjust = 1, vjust = 1, size = 12, color = "#34495E"  ),  axis.text.y = element\_text(  size = 12, color = "#34495E"  ),  legend.position = "right",  legend.title = element\_text(face = "bold"),  legend.text = element\_text(size = 12),  panel.grid.major.x = element\_blank(),  panel.grid.minor = element\_blank(),  panel.grid.major.y = element\_line(color = "gray85", linetype = "dashed"),  plot.background = element\_rect(fill = "white", color = NA),  panel.background = element\_rect(fill = "white", color = NA)  )  # 计算六大类日需求量的数字特征  statistics <- data %>%  group\_by(all\_date, classify\_name) %>%  summarise(total\_qty= sum(qty), .groups = "drop") %>%  group\_by(classify\_name) %>%  summarise(  total = sum(total\_qty),  max\_qty = max(total\_qty),  min\_qty = min(total\_qty),  mean\_qty = mean(total\_qty),  median\_qty = median(total\_qty),  sd\_qty = sd(total\_qty),  skewness\_qty = skewness(total\_qty),  kurtosis\_qty = kurtosis(total\_qty))  ## 三年总需求量图  # ggplot(statistics, aes(x = reorder(classify\_name, total), y = total)) +  my\_colors <- c("#3498DB", "#1ABC9C", "#9B59B6", "#F1C40F", "#E67E22", "#E74C3C")  ggplot(statistics, aes(x = reorder(classify\_name, total), y = total, fill = classify\_name)) +  geom\_col(width = 0.6, alpha = 0.85) +  geom\_text(  aes(label = comma(total)),  vjust = -0.6, size = 4, color = "black", fontface = "bold"  ) +  scale\_fill\_manual(values = my\_colors) +  labs(  title = "六大品类三年总需求量直方图",  subtitle = "数据周期：2020年7月 - 2023年6月",  x = "产品类别",  y = "总需求量（斤）",  fill = "品类名称"  ) +  theme\_minimal(base\_size = 16) +  theme(  plot.title = element\_text(hjust = 0.5, face = "bold", size = 24, color = "#2C3E50"),  plot.subtitle = element\_text(hjust = 0.5, size = 16, color = "gray40", margin = margin(b = 15)),  axis.title.x = element\_text(face = "bold", size = 14, margin = margin(t = 10)),  axis.title.y = element\_text(face = "bold", size = 14, margin = margin(r = 10)),  axis.text.x = element\_text(angle = 30, hjust = 1, vjust = 1, size = 12, color = "#34495E"),  axis.text.y = element\_text(size = 12, color = "#34495E"),  legend.position = "none",  panel.grid.major.y = element\_line(color = "gray90", linetype = "dashed"),  panel.grid.major.x = element\_blank(),  panel.grid.minor = element\_blank(),  plot.background = element\_rect(fill = "white", color = NA),  panel.background = element\_rect(fill = "white", color = NA)  ) +  scale\_y\_continuous(labels = comma, expand = expansion(mult = c(0, 0.1)))  2.3 特征选择  import pandas as pd import numpy as np import matplotlib.pyplot as plt from matplotlib import font\_manager from sklearn.ensemble import RandomForestRegressor from sklearn.model\_selection import train\_test\_split from sklearn.feature\_selection import SelectFromModel #from matplotlib.font\_manager import font\_manager.FontProperties # 导入FontProperties data\_path = r'D:/桌面/毕设/2023国赛C/数据/all/all\_data.csv' data = pd.read\_csv(data\_path) # 去除第一列时间数据 #data = data .drop(data.columns[[0, 1, 7]], axis=1) # 将因变量与自变量分开 X = data.drop(columns=['total\_qty']) y = data['total\_qty'] # 划分数据集 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=1234) X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_train, y\_train, test\_size=0.25, random\_state=1234) # 训练模型 feature\_selector = RandomForestRegressor(n\_estimators=300, random\_state=42, n\_jobs=-1) feature\_selector.fit(X\_train, y\_train) # 输出特征重要性并将其从大到小排序 importances = feature\_selector.feature\_importances\_ indices=np.argsort(importances)[::-1] feat\_labels=X.columns for f in range(X\_train.shape[1]):  print ("%2d) %-\*s %f" %(f+1,30,feat\_labels[indices[f]],importances[indices[f]])) plt.style.use('ggplot') font = font\_manager.FontProperties(family='SimHei') plt.title("Feature importance——all", fontproperties=font) plt.bar(range(X\_train.shape[1]),importances[indices],align='center') plt.xticks(range(X\_train.shape[1]),feat\_labels[indices],rotation=90) plt.xlim([-1,X\_train.shape[1]]) plt.tight\_layout() plt.legend(prop=font) plt.show() plt.savefig("数据/大类/1.辣椒.png") #选择特征/评估模型 sfm = SelectFromModel(feature\_selector, threshold=0.01, prefit=True) # 将数据转换为重要性大于0.01的特征，删除小于0.01的特征 # 一步完成转换和拼接 train\_data = pd.concat([  pd.DataFrame(sfm.transform(X\_train)).reset\_index(drop=True),  pd.DataFrame(y\_train).reset\_index(drop=True) ], axis=1) val\_data = pd.concat([  pd.DataFrame(sfm.transform(X\_val)).reset\_index(drop=True),  pd.DataFrame(y\_val).reset\_index(drop=True) ], axis=1) test\_data = pd.concat([  pd.DataFrame(sfm.transform(X\_test)).reset\_index(drop=True),  pd.DataFrame(y\_test).reset\_index(drop=True) ], axis=1) train\_data.to\_csv("数据/all/train\_selected\_ml\_day.csv", index=False) val\_data.to\_csv("数据/all/val\_selected\_ml\_day.csv", index=False) test\_data.to\_csv("数据/all/test\_selected\_ml\_day.csv", index=False)  2.4 RF模型  import numpy as np import pandas as pd from sklearn.ensemble import RandomForestRegressor from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, mean\_absolute\_percentage\_error, make\_scorer from sklearn.model\_selection import GridSearchCV def rse\_scorer(true, pred):  mse = mean\_squared\_error(true, pred)  return mse / np.sum(np.square(true - np.mean(true))) def rae\_scorer(true, pred):  true\_mean = np.mean(true)  squared\_error\_num = np.sum(np.abs(true - pred))  squared\_error\_den = np.sum(np.abs(true - true\_mean))  rae\_loss = squared\_error\_num / squared\_error\_den  return rae\_loss  # 读数据 train\_data = pd.read\_csv("../../数据/all/train\_selected\_ml\_day.csv") val\_data = pd.read\_csv("../../数据/all/val\_selected\_ml\_day.csv") X\_train = pd.concat([train\_data.drop(columns=['total\_qty']), val\_data.drop(columns=['total\_qty'])]) y\_train = pd.concat([train\_data['total\_qty'], val\_data['total\_qty']]) # 定义超参数搜索空间 param\_grid = {  "n\_estimators": [500, 1000, 1500, 2000, 2500],  "criterion": ["squared\_error", "friedman\_mse", "poisson"],  "ccp\_alpha":[0, 0.1, 0.2, 0.3, 0.4, 0.5, 1] }  # 定义模型 rf = RandomForestRegressor(random\_state=42, n\_jobs=-1) # 进行网格搜索 grid\_search = GridSearchCV(  estimator=rf,  param\_grid=param\_grid,  scoring={  'MAE': make\_scorer(mean\_absolute\_error, greater\_is\_better=False),  'RAE': make\_scorer(rae\_scorer, greater\_is\_better=False),  'MSE': make\_scorer(mean\_squared\_error, greater\_is\_better=False),  'RSE': make\_scorer(rse\_scorer, greater\_is\_better=False),  'MAPE': make\_scorer(mean\_absolute\_percentage\_error, greater\_is\_better=False)  },  refit='MSE', # 以 MSE 作为最终模型选择标准  cv=5,   n\_jobs=1,  verbose=2 # 显示搜索进度 ) # 训练模型 grid\_search.fit(X\_train, y\_train) # 获取最优超参数 best\_params = grid\_search.best\_params\_ print("最佳超参数:", best\_params) # 将搜索结果转换为 DataFrame 并保存 df\_results = pd.DataFrame(grid\_search.cv\_results\_) df\_results.to\_csv("hyperparameter\_results\_day\_0.csv", index=False)  2.5 XGBoost模型  import xgboost as xgb import pandas as pd import numpy as np import joblib from sklearn.model\_selection import GridSearchCV from sklearn.metrics import (mean\_squared\_error, mean\_absolute\_error,  mean\_absolute\_percentage\_error, make\_scorer)  def rse\_scorer(true, pred):  mse = mean\_squared\_error(true, pred)  return mse / np.sum(np.square(true - np.mean(true))) def rae\_scorer(true, pred):  true\_mean = np.mean(true)  squared\_error\_num = np.sum(np.abs(true - pred))  squared\_error\_den = np.sum(np.abs(true - true\_mean))  rae\_loss = squared\_error\_num / squared\_error\_den  return rae\_loss  # 读取数据 train\_data = pd.read\_csv("../../数据/all/train\_selected\_ml\_day.csv") val\_data = pd.read\_csv("../../数据/all/val\_selected\_ml\_day.csv") X\_train = pd.concat([train\_data.drop(columns=['total\_qty']), val\_data.drop(columns=['total\_qty'])]) y\_train = pd.concat([train\_data['total\_qty'], val\_data['total\_qty']])  # 定义超参数搜索空间 param\_grid = {  'learning\_rate': [0.01, 0.05, 0.1],  'n\_estimators': [500,1000,1500],  'subsample': [0.7, 0.8, 0.9, 1.0],   'colsample\_bytree': [0.8, 0.9, 1.0],  'reg\_lambda':[0,0.1,1,10],  'max\_bin': [256,384,512] }  # 定义模型 xgb\_model = xgb.XGBRegressor(  booster='gbtree',  device='cpu',  n\_jobs=10,  validate\_parameters=True,  objective='reg:squarederror',  random\_state=1234,  tree\_method='hist',  sampling\_method='uniform',   grow\_policy='lossguide',  enable\_categorical = True ) # 使用网格搜索 grid\_search = GridSearchCV(  estimator = xgb\_model,  param\_grid = param\_grid,  cv=5, # 5折交叉验证  scoring={  'MAE': make\_scorer(mean\_absolute\_error, greater\_is\_better=False),  'RAE': make\_scorer(rae\_scorer, greater\_is\_better=False),  'MSE': make\_scorer(mean\_squared\_error, greater\_is\_better=False),  'RSE': make\_scorer(rse\_scorer, greater\_is\_better=False),  'MAPE': make\_scorer(mean\_absolute\_percentage\_error, greater\_is\_better=False)  },  refit='MSE', # 以 MSE 作为最终模型选择标准  verbose=2 # 显示搜索进度 ) # 训练模型 grid\_search.fit(X\_train, y\_train) # 获取最优超参数 best\_params = grid\_search.best\_params\_ print("最佳超参数:", best\_params)  # 保存模型 best\_model = grid\_search.best\_estimator\_ joblib.dump(best\_model, 'xgboost\_best\_model0.pkl') print("模型已保存为 xgboost\_best\_model0.pkl") # 将搜索结果转换为 DataFrame 并保存 df\_results = pd.DataFrame(grid\_search.cv\_results\_) df\_results.to\_csv("hyperparameter\_results\_day\_0.csv", index=False)  2.6 GRU模型  import logging import os import time import sys import atexit import pandas as pd from sklearn.model\_selection import train\_test\_split import torch import torch.nn as nn import torch.optim as optim from torch.optim.adamax import Adamax from torch.utils.data import DataLoader, TensorDataset from torch.utils.tensorboard import SummaryWriter  class GRUNet(nn.Module):def \_\_init\_\_(self, input\_size, xx\_size, hidden\_size, num\_layers, output\_size, bias=True, output\_type='last', dropout=0.2, seed=1234):  super(GRUNet, self).\_\_init\_\_()   # Initialize：Linear+batch\_norm  self.fc1 = nn.Linear(input\_size, xx\_size)  self.batch\_norm = nn.BatchNorm1d(xx\_size)   # Initialize the GRU layer  self.gru = nn.GRU(  input\_size = xx\_size,  hidden\_size=hidden\_size,  num\_layers=num\_layers,  bias=bias,  batch\_first=True,  dropout=dropout,  )   # Fully connected layer (adjusted for bidirectional GRU)  self.fc = nn.Linear(hidden\_size, output\_size)  # Output type: 'last' for last timestep or 'mean' for the mean of all timesteps  self.output\_type = output\_type   # Set the random seed for reproducibility  torch.manual\_seed(seed)  self.initialize\_weights()   def initialize\_weights(self):  # Apply orthogonal initialization for fc1  nn.init.xavier\_uniform\_(self.fc1.weight)  if self.fc1.bias is not None:  nn.init.constant\_(self.fc1.bias, 0) # Initialize bias to zero   # Apply orthogonal initialization for GRU's weights  for name, param in self.gru.named\_parameters():  if 'weight' in name:  nn.init.orthogonal\_(param)  elif 'bias' in name:  nn.init.constant\_(param, 0) # Initialize bias to zero   # Apply orthogonal initialization for fc  nn.init.xavier\_uniform\_(self.fc.weight)  if self.fc.bias is not None:  nn.init.constant\_(self.fc.bias, 0) # Initialize bias to zero   def forward(self, x, h\_0=None):# Pass input through the linear+bach\_norm  x = self.fc1(x)  x = x.view(x.size(0), x.size(2), x.size(1))  x = self.batch\_norm(x)  x = x.view(x.size(0), x.size(2), x.size(1))  # Pass input through the GRU layer  out, \_ = self.gru(x, h\_0)   # Process the output based on the specified output type  if self.output\_type == 'last':  out = out[:, -1, :] # Use the output from the last timestep  elif self.output\_type == 'mean':  out = out.mean(dim=1) # Compute the mean over all timesteps   # Pass the processed output through the fully connected layer  out = self.fc(out)  return out  def setup\_logger(file\_name):  logger = logging.getLogger("time\_series\_model")  logger.setLevel(logging.DEBUG)  if not logger.handlers:  # Console 输出：兼容 notebook 的 stdout  ch = logging.StreamHandler(stream=sys.stdout)  ch.setLevel(logging.INFO)  # File 输出  fh = logging.FileHandler(file\_name, mode='a', encoding='utf-8')  fh.setLevel(logging.INFO)  formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')  ch.setFormatter(formatter)  fh.setFormatter(formatter)  logger.addHandler(ch)  logger.addHandler(fh)  atexit.register(logging.shutdown)  return logger  def load\_data(file\_name, seq\_len, test\_ratio, val\_ratio, seed, logger, data\_file="processed\_data.pt", force\_reload=False):# If cached file exists and force\_reload is False, load from cache  if os.path.exists(data\_file) and not force\_reload:  logger.info(f"Loading cached data from {data\_file}")  data = torch.load(data\_file)  return data['X\_train'], data['y\_train'], data['X\_val'], data['y\_val'], data['X\_test'], data['y\_test']   # Read raw CSV data  logger.info(f"Processing data from {file\_name}")  data = pd.read\_csv(file\_name)  print(data)  X\_raw = data.iloc[:, :-1].values # All columns except the last one are features  y\_raw = data.iloc[:, -1].values # Last column is the target   # Create sliding windows  X\_windowed, y\_windowed = [], []  for i in range(0, len(X\_raw) - seq\_len, 1):  X\_windowed.append(X\_raw[i:i + seq\_len])  y\_windowed.append(y\_raw[i + seq\_len]) # Predict next step after the window   # Convert to PyTorch tensors  X = torch.tensor(X\_windowed, dtype=torch.float32)  y = torch.tensor(y\_windowed, dtype=torch.float32)   # First split: Train vs (Validation + Test)  X\_train, X\_temp, y\_train, y\_temp = train\_test\_split(  X, y, test\_size=(test\_ratio + val\_ratio), random\_state=seed, shuffle=True  )   # Second split: Validation vs Test  val\_size = val\_ratio / (test\_ratio + val\_ratio)  X\_val, X\_test, y\_val, y\_test = train\_test\_split(  X\_temp, y\_temp, test\_size=(1 - val\_size), random\_state=seed, shuffle=True  )   # Save the processed data to cache  torch.save({  'X\_train': X\_train, 'y\_train': y\_train,  'X\_val': X\_val, 'y\_val': y\_val,  'X\_test': X\_test, 'y\_test': y\_test  }, data\_file)  logger.info(f"Saved processed dataset to {data\_file}")   return X\_train, y\_train, X\_val, y\_val, X\_test, y\_test  def initialize\_model(config, device):# Initialize the GRU model  model = GRUNet(  input\_size=config['input\_size'],  xx\_size=config['xx\_size'],  hidden\_size=config['hidden\_size'],  num\_layers=config['num\_layers'],  output\_size=config['output\_size'],  bias=config["bias"],  output\_type=config['output\_type'],  dropout=config['dropout'],  seed=config['model\_seed']  ).to(device)   # Loss function and optimizer  criterion = nn.MSELoss()  optimizer = Adamax(model.parameters(), lr=config['learning\_rate'])  return model, criterion, optimizer  def evaluate\_fn(model, criterion, val\_loader, device):  model.eval()  total\_loss = 0  with torch.no\_grad():  for inputs, targets in val\_loader:  inputs, targets = inputs.to(device), targets.to(device)  outputs = model(inputs)  loss = criterion(outputs, targets)  total\_loss += loss.item()  avg\_loss = total\_loss / len(val\_loader)  return -avg\_loss  def train(model, train\_loader, val\_loader, criterion, optimizer, num\_epochs, device, writer,   model\_filename, eval\_interval, evaluate\_fn, logger, patients):  global\_step = 0  best\_score = float('-inf')   for epoch in range(num\_epochs):  epoch\_loss = 0  start\_time = time.time()  for \_, (inputs, targets) in enumerate(train\_loader):  inputs, targets = inputs.to(device), targets.to(device)  # Forward pass  outputs = model(inputs)  loss = criterion(outputs, targets)  # Backward pass  optimizer.zero\_grad()  loss.backward()  optimizer.step()  # Accumulate loss for the epoch  epoch\_loss += loss.item()  # Log the loss for the current step  global\_step += 1  writer.add\_scalar(f'Loss/train', loss.item(), global\_step)  # Log the average loss for the epoch  avg\_epoch\_loss = epoch\_loss / len(train\_loader)  writer.add\_scalar(f'Loss/epoch', avg\_epoch\_loss, epoch)  # Print results for the epoch  logger.info(f"Epoch [{epoch + 1}/{num\_epochs}], Loss: {avg\_epoch\_loss:.4f}, Time: {time.time() - start\_time:.2f}s")   # Log the gradients of parameters after the epoch  for name, param in model.named\_parameters():  if param.grad is not None:  writer.add\_histogram(f'Gradients/{name}\_epoch', param.grad, epoch)    # Evaluate the current model  if (epoch + 1) % eval\_interval == 0 and evaluate\_fn is not None:  model.eval()  with torch.no\_grad():  score = evaluate\_fn(model, criterion, val\_loader, device)  writer.add\_scalar('Score/val', score, epoch)   # If current model is better, then save it  if score > best\_score:  best\_score = score  counter = 0  torch.save(model.state\_dict(), model\_filename)  logger.info(f"New best model saved to {model\_filename} (score: {score:.4f})")  else:  logger.debug(f"No improvement in validation score (current: {score:.4f}, best: {best\_score:.4f})")  counter += 1  if counter >= patients:  logger.info(f"Stopping training early after {counter} epochs without improvement.")  break  model.train()  def main():# Setup logger  logger = setup\_logger(config['log\_file\_name'])   # Set device (GPU or CPU)  device = torch.device(config['device'] if torch.cuda.is\_available() else "cpu")  logger.info(f"Using device: {device}")   # Load data  X\_train, y\_train, X\_val, y\_val, \_, \_ = load\_data(config['data\_file'], config['time\_step'], config['test\_ratio'],  config['val\_ratio'], config['split\_seed'], logger, config['data\_save\_filename'], config['force\_reload'])   # Create DataLoaders  train\_dataset = TensorDataset(X\_train, y\_train)  val\_dataset = TensorDataset(X\_val, y\_val)  train\_loader = DataLoader(train\_dataset, batch\_size=config['batch\_size'], shuffle=False)  val\_loader = DataLoader(val\_dataset, batch\_size=config['batch\_size'], shuffle=False)   # Initialize model, loss, optimizer  model, criterion, optimizer = initialize\_model(config, device)   # TensorBoard writer  writer = SummaryWriter(config['log\_dir'])   # Train the model with validation evaluation  train(  model=model,  train\_loader=train\_loader,  val\_loader=val\_loader,  criterion=criterion,  optimizer=optimizer,  num\_epochs=config['num\_epochs'],  device=device,  writer=writer,  model\_filename=config['model\_filename'],  eval\_interval=config['eval\_interval'],  patients=config['patients'],  evaluate\_fn=evaluate\_fn,  logger=logger  )   # Close the TensorBoard writer  writer.close()  if \_\_name\_\_ == "\_\_main\_\_":  # Configuration file for training the GRU model  config = {  # Data parameters  'data\_file': 'D:/桌面/毕设/2023国赛C/数据/大类/1.辣椒/classify\_1\_dataset\_day.csv', # Path to the CSV file containing input features and targets  'time\_step': 7, # Length of the sliding window (sequence length). Defines the number of time steps to consider for each input sequence.  'val\_ratio': 0.2, # Sample ratio for validation dataset  'test\_ratio': 0.2, # Sample ratio for test dataset  'split\_seed': 42, # Random seed to ensure reproducibility of data splits  'force\_reload': True, # If True, the data will be reprocessed and resaved even if it has been divided before and the division results have been saved.   # Model parameters  'input\_size': 11, # Number of input features (dimension of the input data)  'xx\_size': 11, # Number of linearly transformed features  'hidden\_size': 16, # Number of features in the hidden state of the GRU  'num\_layers': 3, # Number of stacked GRU layers  'output\_size': 1, # Number of output features (dimension of the final output)  'bias': False, # Whether to use bias in GRU layers  'output\_type': 'mean', # 'last' or 'mean', how to process GRU outputs  'dropout': 0.1, # Dropout rate applied to GRU layers  'model\_seed':1234, # Random seed for initializing the GRUNet Model   # Training parameters  'learning\_rate': 0.005, # Learning rate for the optimizer  'batch\_size': 1078, # Batch size for training  'num\_epochs': 3000, # Number of epochs to train the model  'eval\_interval': 5, # Evaluate model on validation set every `eval\_interval` epochs  'patients':7, # Number of rounds of waiting for ealy stopping  # TensorBoard parameters  'log\_dir': 'runs/gru\_experiment', # Directory to save TensorBoard logs  'log\_file\_name': 'log1.log',   # Saving parameters  'model\_filename': 'gru\_model1.pth', # Path to save the trained model  'data\_save\_filename': 'data1.pt', # path to save the divided dataset   # Device configuration (GPU or CPU)  'device': 'cuda', # Device to run the model on ('cuda' or 'cpu')  }   main()  2.7 测试与预测  2.7.1 RF  import numpy as np import pandas as pd from sklearn.ensemble import RandomForestRegressor from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, mean\_absolute\_percentage\_error, make\_scorer import joblib import matplotlib.pyplot as plt from matplotlib import font\_manager  def rse\_scorer(true, pred):  mse = mean\_squared\_error(true, pred)  return mse / np.sum(np.square(true - np.mean(true)))  def rae\_scorer(true, pred):  true\_mean = np.mean(true)  squared\_error\_num = np.sum(np.abs(true - pred))  squared\_error\_den = np.sum(np.abs(true - true\_mean))  rae\_loss = squared\_error\_num / squared\_error\_den  return rae\_loss  # 读取数据 train\_data = pd.read\_csv("D:/桌面/毕设/2023国赛C/数据/all/train\_selected\_ml\_day.csv") val\_data = pd.read\_csv("D:/桌面/毕设/2023国赛C/数据/all/val\_selected\_ml\_day.csv") test\_data = pd.read\_csv("D:/桌面/毕设/2023国赛C/数据/all/test\_selected\_ml\_day.csv") X\_train = pd.concat([train\_data.drop(columns=['total\_qty']), val\_data.drop(columns=['total\_qty'])]) y\_train = pd.concat([train\_data['total\_qty'], val\_data['total\_qty']]) X\_test, y\_test = test\_data.drop(columns=['total\_qty']), test\_data['total\_qty'] # 定义模型 rf = RandomForestRegressor(n\_estimators=1000, criterion='poisson', ccp\_alpha=0, random\_state=42, n\_jobs=-1) rf.fit(X\_train, y\_train)  # 保存模型 joblib.dump(rf, "rf\_month\_1.pkl")#pickle这个包可以做序列化对象的保存 # 模型预测 y\_pred = rf.predict(X\_test) # 计算并打印评估指标 mae = mean\_absolute\_error(y\_test, y\_pred) mse = mean\_squared\_error(y\_test, y\_pred) mape = mean\_absolute\_percentage\_error(y\_test, y\_pred) rse = rse\_scorer(y\_test, y\_pred) rae = rae\_scorer(y\_test, y\_pred) print(f"MAE: {mae:.4f}, MSE: {mse:.4f}, MAPE: {mape:.4f}, RSE: {rse:.4f}, RAE: {rae:.4f}")  2.7.2XGBoost  import pandas as pd import numpy as np import joblib from sklearn.metrics import (mean\_squared\_error, mean\_absolute\_error,  mean\_absolute\_percentage\_error)  def rse\_scorer(true, pred):  mse = mean\_squared\_error(true, pred)  return mse / np.sum(np.square(true - np.mean(true)))  def rae\_scorer(true, pred):  true\_mean = np.mean(true)  squared\_error\_num = np.sum(np.abs(true - pred))  squared\_error\_den = np.sum(np.abs(true - true\_mean))  rae\_loss = squared\_error\_num / squared\_error\_den  return rae\_loss  # 读取数据 train\_data = pd.read\_csv("D:/桌面/毕设/2023国赛C/数据/all/train\_selected\_ml\_day.csv") val\_data = pd.read\_csv("D:/桌面/毕设/2023国赛C/数据/all/val\_selected\_ml\_day.csv") test\_data = pd.read\_csv("D:/桌面/毕设/2023国赛C/数据/all/test\_selected\_ml\_day.csv") X\_train = pd.concat([train\_data.drop(columns=['total\_qty']), val\_data.drop(columns=['total\_qty'])]) y\_train = pd.concat([train\_data['total\_qty'], val\_data['total\_qty']]) X\_test, y\_test = test\_data.drop(columns=['total\_qty']), test\_data['total\_qty']  # 载入模型 xgb\_model = joblib.load("D:/桌面/毕设/2023国赛C/model/xgboost/xgboost\_best\_model0.pkl") # 模型预测 y\_pred = xgb\_model.predict(X\_test) # 计算并打印评估指标 mae = mean\_absolute\_error(y\_test, y\_pred) mse = mean\_squared\_error(y\_test, y\_pred) mape = mean\_absolute\_percentage\_error(y\_test, y\_pred) rse = rse\_scorer(y\_test, y\_pred) rae = rae\_scorer(y\_test, y\_pred) print(f"MAE: {mae:.4f}, MSE: {mse:.4f}, MAPE: {mape:.4f}, RSE: {rse:.4f}, RAE: {rae:.4f}")  2.7.3 GRU  import numpy as np from gru import GRUNet import torch from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, mean\_absolute\_percentage\_error from torch.utils.data import TensorDataset, DataLoader import matplotlib.pyplot as plt from matplotlib import font\_manager #自定义RSE def rse\_scorer(true, pred):  mse = mean\_squared\_error(true, pred)  return mse / np.sum(np.square(true - np.mean(true)))  #自定义RAE def rae\_scorer(true, pred):  true\_mean = np.mean(true)  squared\_error\_num = np.sum(np.abs(true - pred))  squared\_error\_den = np.sum(np.abs(true - true\_mean))  rae\_loss = squared\_error\_num / squared\_error\_den  return rae\_loss  config = {  # Data parameters  'data\_file': 'D:/桌面/毕设/2023国赛C/数据/大类/1.辣椒/classify\_1\_dataset\_day.csv', # Path to the CSV file containing input features and targets  'time\_step': 7, # Length of the sliding window (sequence length). Defines the number of time steps to consider for each input sequence.  'val\_ratio': 0.2, # Sample ratio for validation dataset  'test\_ratio': 0.2, # Sample ratio for test dataset  'split\_seed': 42, # Random seed to ensure reproducibility of data splits  'force\_reload': True, # If True, the data will be reprocessed and resaved even if it has been divided before and the division results have been saved.   # Model parameters  'input\_size': 11, # Number of input features (dimension of the input data)  'xx\_size': 11, # Number of linearly transformed features  'hidden\_size': 16, # Number of features in the hidden state of the GRU  'num\_layers': 3, # Number of stacked GRU layers  'output\_size': 1, # Number of output features (dimension of the final output)  'bias': False, # Whether to use bias in GRU layers  'output\_type': 'mean', # 'last' or 'mean', how to process GRU outputs  'dropout': 0.1, # Dropout rate applied to GRU layers  'model\_seed':1234, # Random seed for initializing the GRUNet Model   # Training parameters  'learning\_rate': 0.005, # Learning rate for the optimizer  'batch\_size': 1078, # Batch size for training  'num\_epochs': 3000, # Number of epochs to train the model  'eval\_interval': 5, # Evaluate model on validation set every `eval\_interval` epochs  'patients':7, # Number of rounds of waiting for ealy stopping  # TensorBoard parameters  'log\_dir': 'runs/gru\_experiment', # Directory to save TensorBoard logs  'log\_file\_name': 'log1.log',   # Saving parameters  'model\_filename': 'gru\_model1.pth', # Path to save the trained model  'data\_save\_filename': 'data1.pt', # path to save the divided dataset   # Device configuration (GPU or CPU)  'device': 'cuda', # Device to run the model on ('cuda' or 'cpu')  }  device = 'cuda' if torch.cuda.is\_available() else 'cpu'  #加载test\_loader data\_dict = torch.load('D:/桌面/毕设/2023国赛C/model/gru/data1.pt') #print(data\_dict.keys()) X\_test = data\_dict['X\_test'] y\_test = data\_dict['y\_test'] test\_dataset = TensorDataset(X\_test, y\_test) test\_loader = DataLoader(test\_dataset, batch\_size=config['batch\_size'], shuffle=False) #print(type(test\_loader))  #模型导入 model = GRUNet(  input\_size=config['input\_size'],  xx\_size=config['xx\_size'],  hidden\_size=config['hidden\_size'],  num\_layers=config['num\_layers'],  output\_size=config['output\_size'],  bias=config["bias"],  output\_type=config['output\_type'],  dropout=config['dropout'],  seed=config['model\_seed'] ).to(device) #加载模型参数 model.load\_state\_dict(torch.load('D:/桌面/毕设/2023国赛C/model/gru/gru\_model1.pth')) model.eval()  #定义损失函数 criterion ={  'MAE': mean\_absolute\_error,  'RAE': rae\_scorer,  'MSE': mean\_squared\_error,  'RSE': rse\_scorer,  'MAPE': mean\_absolute\_percentage\_error,  } all\_preds = [] all\_targets = []  #测试模型 total\_loss = 0 with torch.no\_grad():  for inputs, targets in test\_loader:  inputs, targets = inputs.to(device), targets.to(device)  outputs = model(inputs)  #保存每一批次的输出和目标，  all\_preds.append(outputs.cpu().numpy())  all\_targets.append(targets.cpu().numpy()) all\_preds = np.concatenate(all\_preds, axis=0).squeeze() all\_targets = np.concatenate(all\_targets, axis=0).squeeze() results = {} for name, func in criterion.items():  results[name] = func(all\_targets, all\_preds)  # 打印所有指标 print("\n=== test Metrics ===") for metric, testue in results.items():  print(f"{metric}: {testue:.4f}") |

# 致 谢

文末搁笔，忽觉岁短。二十余年求学之路渐进尾声，一路跌跌撞撞走到现在，时光里有少年的不羁，有青春的迷茫，也有成熟之后的坦然与温暖。

在本文完成之际，首先要感谢我的论文指导老师温阳俊老师，从论文的选题到最终成文，感谢您陪我字斟句酌，倾尽所能的点播和指导我。不论是专业知识，亦或是行为观念，温老师都教会我很多。人生之幸，得遇良师。

再者，我要感谢我的父母陈先生和吴女士。二十载求学路，你们给予了我最大的支持，给了我选择的底气和可能，也给了我成长的机会。感谢这么多年来你们无怨无悔的付出，永不停歇的照顾我，教我人生道理，也给了我追逐自由的勇气。也感谢你们包容我的任性，理解我的固执，尊重我的理想主义，永远支持我的决定。

其次，感谢所有帮助过我的朋友。同频的人就像是礼物，感谢你们的陪伴，让我的青春更加精彩。感谢你们愿意时刻同我分享快乐，倾听我的不安与焦虑，给予我毫无理由的支持。感谢小倪同学，从决策树到深度学习，从理论学习到代码实践，感谢你把知识点揉碎了一点一点的教我。

最后，感谢这一路以来，走得很慢但一直向前的自己，愿春痕吹向四野，铸就磅礴的绿。

感谢所有为我亮起的灯，在我丧气的时候，总是仗义的过来按我的门铃。