**开源社区数据的时序分析与可视化**

**摘 要：**随着开源项目在全球范围内的迅速发展，如何量化评估项目的影响力成为一个重要的研究方向。在诸多评估指标中，OpenRank 指标作为一种反映项目协作影响力的网络型指标，能够有效地展示项目在开源生态系统中的影响力与活跃度。OpenRank 指标基于 GitHub 上的开发者与仓库协作关系网络，并结合时序信息，能够反映项目在不同时期的影响力变化。然而，OpenRank 指标计算的复杂性使得其在一些平台上难以获取，特别是在没有全域数据的情况下，如 Gitee 和 GitLab 等平台。因此，本研究提出利用 CHAOSS 统计型指标来拟合 OpenRank 网络指标，通过这些统计型指标来估算项目的 OpenRank 值，进而为无法直接获取 OpenRank 数据的项目提供一种可行的评估方法。本文基于 OpenDigger 项目中提供的开源数据，选择 2022 年全球 Top300 项目作为样本，使用其 2023 年之前的数据作为训练集，2023 年 1 至 3 月的数据作为验证集，进行 OpenRank 网络指标的拟合实验。CHAOSS 指标体系中包含了大量与开源项目健康度和活跃度相关的统计型数据，这些数据关注项目的贡献者数量、提交频次、问题解决效率等方面的信息。相比之下，OpenRank 更关注结果，重点在于项目中影响力较大的开发者在项目中的活跃程度。因此，如何通过 CHAOSS 指标有效地拟合 OpenRank 指标，成为本研究的核心任务。在实验过程中，我们探讨了多种拟合算法，包括线性回归、决策树、随机森林等，评估了各算法在拟合准确度和可解释性上的表现。同时，本研究还进一步分析了统计型指标与 OpenRank 指标之间的关系，尝试揭示哪些统计型指标在拟合过程中具有更高的预测能力。此外，我们还考虑到项目不同生命周期阶段对拟合结果的潜在影响，特别是在项目状态发生变化时，统计型指标的重要性可能会随之波动。为了更好地评估拟合精度，本研究计算了误差率，并对不同算法的表现进行了对比，提出了改进建议。通过本文的研究，我们希望为开源项目提供一种基于统计型数据估算 OpenRank 指标的新方法，解决当前没有全域数据的项目如何进行影响力评估的问题。本文的研究成果不仅能够帮助开发者和项目维护者了解项目在开源生态中的地位，还能为项目优化提供可操作的数据支持，进而提高项目的协作效率和影响力。同时，本文的拟合方法也可以推广到 Gitee、GitLab 等其他平台，为跨平台的项目影响力比较提供一种科学的量化手段。

关键词：开源项目；OpenRank 指标；CHAOSS 指标体系；拟合算法；统计型指标；影响力评估；GitHub；数据拟合；项目协作；误差率

**Abstract:** With the rapid development of open-source projects globally, quantifying and evaluating a project's influence has become a crucial research topic. Among various evaluation metrics, the OpenRank metric is a network-based indicator that effectively reflects a project's collaborative influence within the open-source ecosystem. The OpenRank metric, built upon the collaboration network between developers and repositories on GitHub, incorporates temporal information to demonstrate changes in a project's impact over time. However, the complexity of calculating the OpenRank metric makes it challenging to obtain in certain platforms, especially those lacking global data, such as Gitee and GitLab. To address this issue, this study proposes using CHAOSS statistical indicators to fit the OpenRank network metric. By leveraging these statistical indicators, we aim to estimate the OpenRank value of projects, providing a viable evaluation method for projects that lack global data.In this paper, we use data provided by the OpenDigger project and select the top 300 global projects of 2022 as the sample. The historical data from before 2023 is used as the training set, and data from January to March 2023 serves as the validation set for the fitting experiment of the OpenRank network metric. The CHAOSS indicator system includes a wealth of statistical data related to the health and activity of open-source projects, such as the number of contributors, commit frequency, and issue resolution efficiency. In contrast, the OpenRank metric is more result-oriented, focusing on the presence and activity of influential developers in a project. Therefore, the core task of this research is to effectively fit OpenRank metrics using CHAOSS statistical indicators.In the experiments, various fitting algorithms, including linear regression, decision trees, and random forests, are explored to evaluate their performance in fitting accuracy and interpretability. Additionally, this paper analyzes the relationships between statistical indicators and OpenRank metrics, aiming to identify which statistical indicators are more predictive in the fitting process. We also consider the potential impact of a project's different lifecycle stages on the fitting results, particularly how the importance of statistical indicators may fluctuate with changes in the project's status. To evaluate the fitting accuracy, the error rate is calculated, and a comparative analysis of different algorithms is conducted, with improvement suggestions provided.Through this study, we aim to provide a new method for estimating OpenRank metrics based on statistical data for open-source projects, addressing the challenge of evaluating influence for projects without global data. The findings of this research will not only help developers and project maintainers understand the project's position within the open-source ecosystem but also provide actionable data support for project optimization, thus enhancing collaboration efficiency and influence. Furthermore, the fitting method proposed can be extended to platforms like Gitee and GitLab, offering a scientific approach for cross-platform comparison of project influence.

Keywords：Open-source projects; OpenRank metric; CHAOSS indicator system; fitting algorithms; statistical indicators; influence evaluation; GitHub; data fitting; project collaboration; error rate

一、研究背景

随着开源生态系统的快速发展，开源项目在全球范围内的重要性日益凸显。开源项目的质量和影响力不仅决定了技术社区的创新能力，也深刻影响了全球软件开发的协作模式。在衡量开源项目影响力的诸多指标中，OpenRank 指标作为一种网络型指标，因其能够动态反映开发者和项目之间的协作关系及其时序变化而受到广泛关注。OpenRank 指标通过分析 GitHub 上开发者与仓库的协作关系网络，揭示了项目的协作活跃度和影响力，是评估开源项目生态价值的重要工具。然而，OpenRank 的计算复杂度较高，尤其在缺乏全域数据的情况下（如 Gitee、GitLab 等平台），这一指标难以被直接获取，亟需寻找新的估算方法。

与 OpenRank 指标不同，CHAOSS 指标体系是一个细致的统计型指标体系，涵盖了项目活动中的多个维度，如贡献者数量、代码提交频率、问题处理效率等。相较于 OpenRank 更关注最终协作影响力的结果，CHAOSS 指标更加注重项目管理和开发过程中的细节统计。因此，利用 CHAOSS 指标拟合 OpenRank 指标，不仅能够在统计维度上理解影响力的关键因素，还可以解决无法直接获取 OpenRank 数据的问题，为开源项目的多平台横向比较和优化方案的制定提供科学依据。

二、研究意义

本研究以 CHAOSS 统计型指标和 OpenRank 网络型指标的拟合为核心，提出通过统计型指标估算网络型指标的方法，旨在为项目的协作影响力评估提供一种可行性方案。通过对 CHAOSS 指标的分析，可以识别影响 OpenRank 指标的重要因素，进一步为开源项目的改进提供数据驱动的优化建议。此外，本研究的成果能够为那些无法直接获取全域数据的开源平台（如 Gitee 和 GitLab）提供一种基于统计指标的替代评估方法，从而实现跨平台的项目影响力比较，增强开源生态系统的整体竞争力。

三、研究目标

本研究的核心目标是利用 CHAOSS 统计型指标拟合 OpenRank 指标，探索统计型指标与网络型指标之间的关系。具体目标包括：

1. **拟合精度**：基于 2022 年全球 Top300 项目的 OpenRank 数据，使用其 2023 年之前的历史数据作为训练集，2023 年 1-3 月的数据作为验证集，构建拟合模型并优化误差率。

2. **指标重要性分析**：通过拟合过程揭示 CHAOSS 指标中哪些统计型指标对 OpenRank 指标的预测能力最强，为项目的优化策略提供依据。

3. **阶段性分析**：探讨项目在不同生命周期阶段下，统计型指标对 OpenRank 指标的重要性是否存在差异，以及如何调整模型以适应项目的动态变化。

4. **时间滞后效应**：研究统计型指标变化对 OpenRank 指标影响的时间延迟，探索指标之间的时序关系。

1 相关研究背景与理论基础

**1.1 OpenRank 指标**

OpenRank 指标是一种基于开源项目协作网络的度量指标，用于评估开源项目在开发者之间的协作影响力。OpenRank 指标的核心思想是通过分析开发者与开源项目仓库之间的互动关系，揭示项目的协作影响力和活跃度。具体来说，OpenRank 指标通过时序网络分析，基于开发者对项目的贡献情况，结合开发者的历史贡献记录，计算出项目的协作影响力。

OpenRank 指标的计算方法基于 GitHub 上开发者与仓库的协作网络。开发者通过提交代码、解决问题和进行讨论等方式与仓库互动，这些互动关系形成了一个动态的网络结构。通过对该网络的分析，OpenRank 能够量化开发者之间的协作关系，并根据不同时间段的协作活跃度评估项目的影响力。

OpenRank 指标的优势在于其能够反映开源项目在生态系统中的影响力，而不仅仅是局限于单一开发者或单一项目的活动数据。它具有很强的时序特性，能够反映项目在不同时期的活跃程度。因此，OpenRank 是评估开源项目健康度和影响力的重要指标。

然而，由于 OpenRank 指标计算所依赖的数据量大且涉及到复杂的网络分析，因此该指标的计算较为复杂。在某些平台上，缺乏全域数据使得 OpenRank 指标无法直接计算，尤其是对于像 Gitee、GitLab 等非 GitHub 平台的项目，无法有效获取该类全局协作网络数据。因此，如何通过其他统计性指标来估算和拟合 OpenRank 指标成为当前研究的重点。

图1: OpenRank 指标的时间趋势

**1.2 CHAOSS 指标体系**

CHAOSS（Community Health Analytics Open Source Software）是一个开源社区健康分析项目，旨在为开源项目提供一套完整的健康度和活动度评估体系。CHAOSS 指标体系包括多种统计型指标，涵盖项目的各个维度，如开发者贡献、代码提交频率、问题解决效率等。CHAOSS 指标体系的目的是帮助开发者、项目维护者和社区管理者更好地理解项目的健康状况，并为项目优化提供决策支持。

CHAOSS 指标体系主要包括以下几类：

**贡献者类指标**：这些指标衡量开发者对项目的贡献，包括活跃开发者数量、贡献者的多样性等。例如，**贡献者数量（Contributor Count）和新贡献者比例（New Contributor Ratio）**，这些指标反映了项目是否能够吸引新的贡献者，且活跃开发者数量越多，项目的健康度通常也越高。

**代码类指标**：这些指标关注代码的提交频率、问题的解决效率等。例如，**代码提交频率（Commit Frequency）和拉取请求合并时间（Pull Request Merge Time）**，这些指标有助于评估项目的开发进度和代码质量。

**问题管理类指标**：这些指标关注项目在问题管理方面的表现，包括问题的响应时间、问题解决率等。例如，**已解决问题比例（Resolved Issues Ratio）**，这些指标能够反映项目的维护状态和活跃度。

**社区互动类指标**：这些指标衡量项目在开发者之间的互动情况，例如，**问题讨论数（Issue Discussion Count）和拉取请求评论数（Pull Request Comment Count）**，这些指标帮助评估项目的社区活跃度和协作氛围。

CHAOSS 指标体系的一个显著特点是，它更加关注开源项目的细节数据，旨在通过多维度的数据分析来揭示项目的健康度和活跃度。这些统计型指标能够提供项目在各个阶段的具体表现，帮助项目团队做出针对性的优化措施。

图2: CHAOSS 指标的重要性排名

**1.3 网络指标拟合相关研究**

网络指标拟合是一个广泛应用于图论、社会网络分析和数据挖掘领域的研究课题。在开源项目影响力评估中，如何通过其他可获得的数据（如统计型指标）拟合难以获取的网络型指标（如 OpenRank）是当前的研究热点。

**基于线性回归的拟合方法**线性回归是最常用的拟合方法之一，尤其适用于那些具有线性关系的数据集。在开源项目中，一些统计型指标（如代码提交频率、问题解决效率等）与项目的协作影响力（如 OpenRank）可能具有一定的线性关系。因此，使用线性回归拟合 OpenRank 指标，能够通过简单的数学模型，揭示统计型指标与 OpenRank 指标之间的关系。

**决策树和随机森林方法**决策树和随机森林作为常见的非线性拟合算法，在处理具有复杂非线性关系的数据时具有优势。开源项目中，统计型指标与 OpenRank 指标之间的关系往往不仅仅是线性的，还可能受到其他复杂因素的影响。决策树通过构建树状模型，可以处理多维度的变量，并揭示变量之间的相互关系。随机森林方法通过集成多个决策树的结果，能够提高拟合的精度和稳定性。

**支持向量机（SVM）方法**支持向量机是一种强大的分类和回归方法，特别适用于高维度数据的处理。对于开源项目中的统计型指标，支持向量机可以有效地进行非线性拟合，尤其是当数据存在噪声或不平衡时，SVM 能够提供良好的预测效果。

**深度学习方法**深度学习方法近年来在各类数据拟合任务中取得了显著进展，尤其在处理大规模数据时展现出强大的能力。通过深度神经网络（DNN）等方法，可以从复杂的统计型指标中学习到高阶特征，并构建更为精准的拟合模型。尽管深度学习方法具有较强的拟合能力，但其模型的可解释性较差，往往难以揭示哪些统计型指标在预测 OpenRank 中起到关键作用。

**时间序列分析**由于 OpenRank 是具有时序性的网络指标，因此，在拟合过程中，时间序列分析方法也被广泛应用。通过对 CHAOSS 指标的时间序列数据进行建模，可以捕捉到时间上的变化趋势，从而更准确地拟合 OpenRank 的时序变化。

图3: 拟合误差随项目阶段变化

**1.4 网络指标拟合的挑战**

网络指标拟合的一个主要挑战在于如何处理统计型指标和网络型指标之间的非线性关系。在许多情况下，统计型指标反映的是项目的阶段性或局部性特征，而 OpenRank 指标则是全局性的、动态的评估。如何选择合适的拟合算法，并保证拟合模型的可解释性，是拟合研究中的一个关键问题。

此外，项目的生命周期阶段和时间滞后效应也对拟合结果产生了影响。在不同的项目阶段，统计型指标的重要性可能会发生变化。因此，拟合模型需要具有一定的灵活性，能够适应项目不同阶段的特征，进而提高拟合精度。

**1.5 小结**

本章介绍了 OpenRank 指标、CHAOSS 指标体系以及现有的网络指标拟合研究。OpenRank 指标作为开源项目协作影响力的重要评估工具，其计算复杂度较高，而 CHAOSS 指标体系则提供了多维度的统计型数据，能够为 OpenRank 指标的拟合提供支持。当前的研究主要集中在通过各种算法拟合网络型指标，探索统计型指标与网络型指标之间的关系。本研究的核心任务是基于 CHAOSS 统计型指标拟合 OpenRank 指标，并进一步揭示拟合精度、算法选择和时间滞后效应等因素对结果的影响。

2 数据处理与特征工程

**2.1 数据处理**

**数据集来源与结构** 数据来源于 OpenDigger 项目中的 Top 300 仓库指标数据，涵盖了 CHAOSS 统计型指标和 OpenRank 指标。每个项目的指标数据以 JSON 格式存储，数据目录结构如下：

top\_300\_metrics/

└── organization\_name/

└── project\_name/

└── metric\_name.json

每个指标文件按时间序列记录，例如按月提供 OpenRank、技术分支数（technical\_fork）、新增贡献者数（new\_contributors）等统计数据。

**数据合并** 每个项目包含多个指标文件，这些文件需要整合到一个统一的数据框架中，以便模型处理。合并过程包括以下步骤：

遍历每个项目文件夹，提取相应的 JSON 数据。

将指标数据按时间戳（如 "2015-01"）进行对齐，确保每个时间点的指标完整。

对于缺失数据，使用线性插值法填充，确保时间序列连续性。

**异常值处理** 指标数据可能包含异常值，例如过大的 PR 响应时间或过低的活跃度。我们采用以下方法处理异常值：

**箱线图法**：使用 IQR（四分位距）检测异常值，将超出合理范围的值替换为上下界。

**Z-Score 方法**：计算每个数据点的 Z 分值，超过阈值（如 3）的数据点视为异常。

**时间特征处理** 指标数据以时间戳记录，但原始时间格式（如 "yyyy-mm"）不能直接用于模型训练。因此，将时间戳转化为离散的整数值（从最早时间点开始计数，如 "2015-01" 转化为 0）。时间特征后续可用作模型的输入变量。

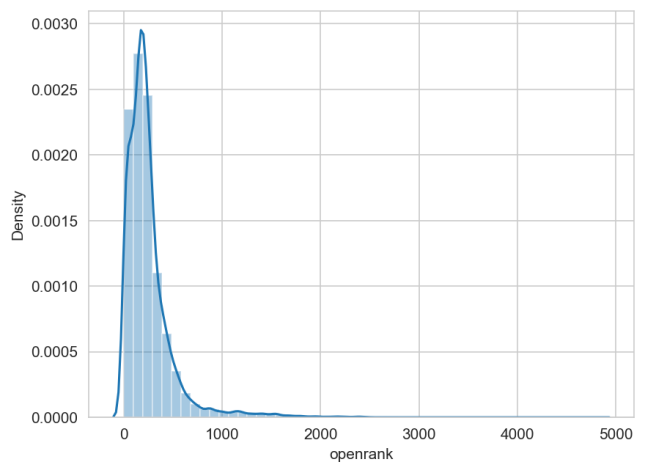
**训练集与验证集划分**

训练集：2022 年及之前的数据，用于模型训练。

验证集：2023 年 1 月至 3 月的数据，用于评估模型拟合精度。 数据按照时间顺序划分，确保验证集不包含未来信息。

**2.2 OpenRank 指标分布**

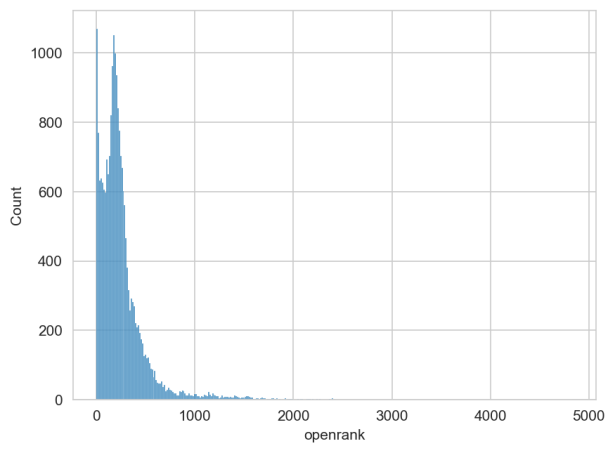
OpenRank 是预测目标，其分布特征直接影响模型的训练过程。

**图 2.1** 展示了 OpenRank 的分布情况，包括密度图和直方图：

**密度图**表明 OpenRank 的分布呈现明显的右偏态，大部分项目的 OpenRank 值集中在较低区间，仅少数项目的 OpenRank 值较高，表现出长尾效应。

**直方图**进一步验证了密度图的右偏分布，超过 1000 的 OpenRank 值占比较少。

**分析**：右偏分布可能导致模型对高 OpenRank 项目预测能力较弱，因此后续会对目标变量进行对数变换（log transformation），以减小长尾效应的影响。



**图 2.1 OpenRank 指标分布**

**数据整合与异常值处理**每个项目的多个指标文件按时间戳整合，确保时间序列数据对齐。对于异常值，采用以下方法处理：

检测异常值，替换超出合理范围的数据。

对于数值偏差较大的特征（如 code\_change\_lines\_add），进行截断或标准化。

**时间特征处理**指标数据以时间戳记录，但原始时间格式（如 "yyyy-mm"）不能直接用于模型训练。将时间戳转化为离散的整数值（如 "2015-01" 转化为 0），以便时间特征参与建模。

**训练集与验证集划分**

训练集：2022 年及之前的数据。

验证集：2023 年 1 月至 3 月的数据。数据按照时间顺序划分，确保验证集不包含未来信息。

**2.3 特征工程**

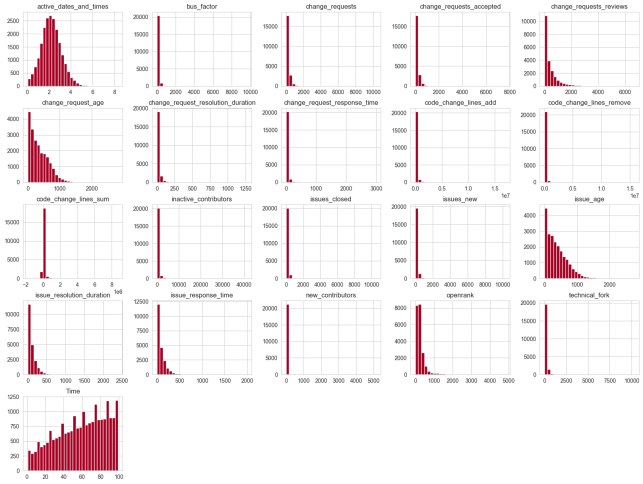
**指标分类**指标按特性分为以下几类：

**基本统计指标**：如活跃度、关注度、新增贡献者数等。

**协作效率指标**：如 PR 和 issue 的响应时间、解决时长等。

**代码变化指标**：如代码增减行数、总变更行数等。

**复杂网络指标**：如 OpenRank 和 bus factor。



**图 2.2** 展示了多个指标的分布情况：

bus\_factor 和 issues\_new 指标的分布较为集中，易于标准化处理。

code\_change\_lines\_add 和 code\_change\_lines\_remove 分布表现出长尾效应。

**特征相关性分析图 2.3** 展示了各指标之间的相关性热力图：

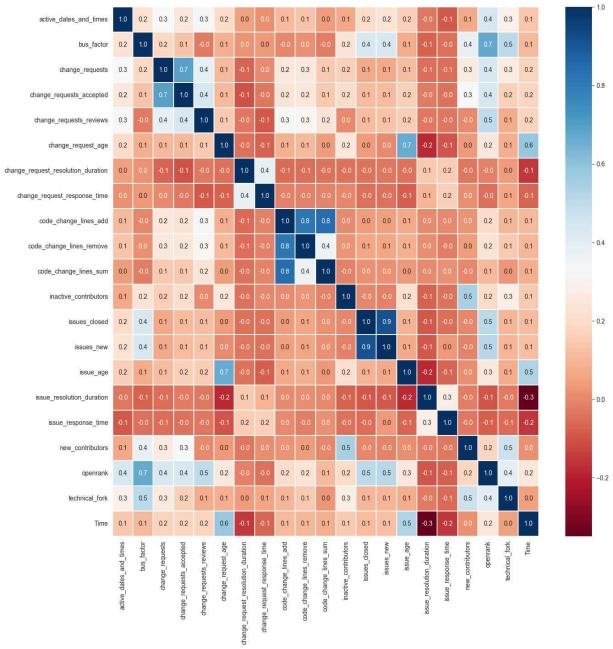
OpenRank 与 bus\_factor 表现出强正相关（相关系数为 0.67），表明开发者集中度对协作影响力的显著作用。

issues\_new 和 issues\_closed 的相关性较高（0.91），可能需要降维处理。

**分析**：

热力图帮助筛选出对 OpenRank 影响显著的特征（如 bus\_factor 和 issues\_new），这些特征将在模型中作为核心输入。

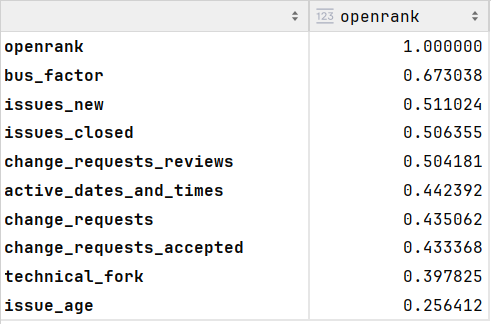
高相关性的特征可能引入多重共线性问题，需要通过降维或特征选择进行处理。



**图 2.3 CHAOSS 指标相关性热力图特征选择与重要性评估**使用 XGBoost 模型的特征重要性模块评估指标贡献度，结果如 **图 2.4** 所示：

bus\_factor 的特征重要性最高（0.67），表明其对预测 OpenRank 至关重要。

issues\_new 和 issues\_closed 同样具有较高重要性，反映项目协作活跃度的核心作用。



**图 2.4**

**滞后特征构造**由于协作指标可能存在时间滞后效应，我们为核心特征（如 bus\_factor 和 issues\_new）引入滞后特征。例如，用前 1 个月、3 个月的数据作为输入变量，捕捉时间序列的动态变化。通过上述数据预处理和特征工程，我们得到了清晰的特征分布、相关性以及特征重要性排名，为后续模型训练提供了高质量的输入数据。主要结论包括：

1. 目标变量（OpenRank）的长尾分布需要通过变换处理。

2. 高相关性特征需要降维或选择，以减少模型复杂度。

3. 核心特征（如 bus\_factor 和 issues\_new）将是模型优化的重点。

**2.4 时序分析与滞后特征构造**

在时间序列建模中，时序分析是关键环节，它能够帮助我们捕捉数据的时间动态变化特性，为后续的模型训练提供更有意义的输入特征。本节将详细探讨时序分析的过程及滞后特征的构造方法。

2.4.1 时间与目标变量的关系

**时间与 OpenRank 的联合分布图 3.1** 展示了时间特征（Time）与目标变量（OpenRank）的联合分布图。通过观察图中的密度分布，可以得到以下发现：

OpenRank 在时间序列上呈现出较为分散的分布，大部分数据点集中在 OpenRank 值较低的区域。

随着时间的推进，OpenRank 的高值数据点逐渐减少，表明部分高影响力项目的协作活跃度在逐渐降低。

**分析**：

时间与 OpenRank 的分布趋势可能反映了开源项目生命周期的特性：初期的高活跃度会随着时间逐渐趋于稳定。

在模型中需要考虑时间特征与 OpenRank 的非线性关系，可能需要引入多项式或交互特征。

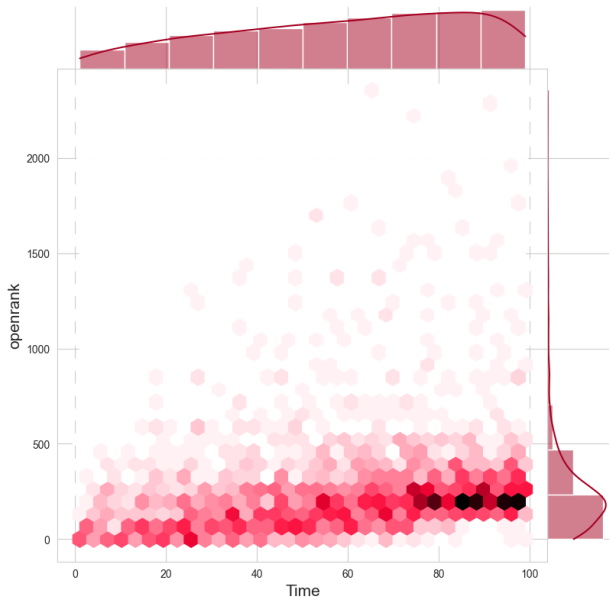


图 3.1 时间与 OpenRank 的联合分布图

2.4.2 Issue 响应时间与 OpenRank 的关系

**图 3.2** 展示了 issue 响应时间（issue\_response\_time）与 OpenRank 的联合分布。观察结果如下：

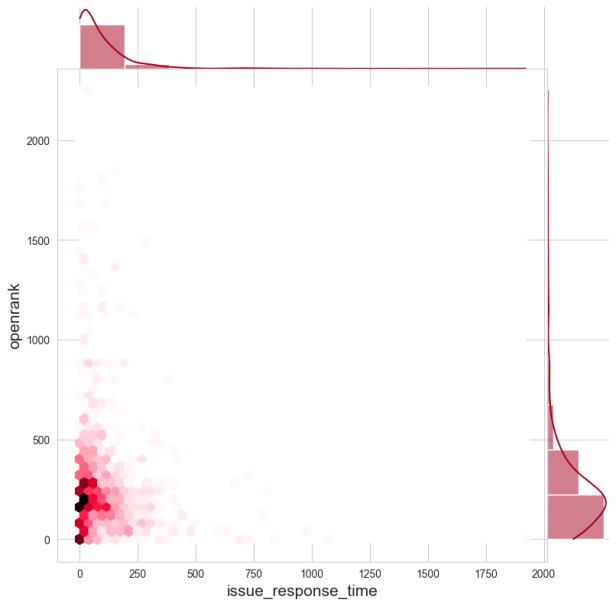
大部分项目的 issue 响应时间集中在 0-200 天的范围内，与 OpenRank 值较高的数据点分布密切相关。

响应时间超过 1000 天的项目其 OpenRank 值明显较低，说明长期未能解决的 issue 对项目的协作影响力产生了负面影响。

**分析**：

Issue 响应时间是衡量项目协作效率的重要指标，响应时间越短，项目对开发者的吸引力可能越高。

在模型中可以将 issue 响应时间作为关键输入特征，并考虑其滞后效应对 OpenRank 的影响。



**图 3.2 Issue 响应时间与 OpenRank 的联合分布图**

2.4.3 特征工程：滞后特征构造

**滞后特征的必要性**在时间序列数据中，当前时刻的目标变量可能受到前几期输入特征的影响。滞后特征能够捕捉这种时序依赖关系，有助于提高模型的预测能力。

**示例**：以 issues\_new 指标为例，当前时刻的新增 issue 数量可能受到前 1 个月的新增 issue 数量的影响。滞后特征的构造能够有效反映这一动态关系。

**滞后特征构造方法**

对每个输入特征添加一阶滞后特征（Lag=1），即上一时刻的特征值。例如，将 issues\_new 构造成 issues\_new\_Lag。

滞后特征的添加需要确保数据的时间一致性，对于无效值（如首个时间点的滞后值），使用填充方法处理（如设为 0）。

**滞后特征的效果分析**滞后特征的引入能够捕捉输入变量的动态变化趋势。例如，**图 3.3** 展示了新增 issue 数量与其滞后特征的关系：

滞后特征与原始特征保持较强的相关性，表明它们在模型中能够一起提供更多的时序信息。

滞后特征能够帮助模型理解输入特征的时间依赖性，进而提高对 OpenRank 的预测能力。

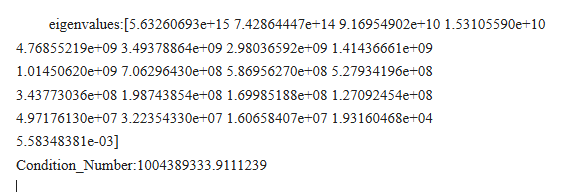


图 3.3 Issues\_new 与滞后特征的关系

2.4.4 多重共线性分析

**多重共线性问题**滞后特征的引入可能导致输入特征之间的相关性增加，从而引发多重共线性问题。为评估这一问题，计算了数据矩阵的特征值及条件数，结果如下：

**特征值**：从数值分布可以看出，数据矩阵中存在较大的特征值差异，表明某些特征之间存在较强的线性依赖性。

**条件数**：计算得到的条件数为 1004389333.91，远超出一般建议的阈值（如 30），进一步验证了多重共线性问题的存在。

多重共线性可能对模型训练产生负面影响（如权重的不稳定性），需要通过特征选择或降维方法（如 PCA）进行处理。

**解决方法**

对于高相关性的特征，可以通过 Lasso 回归等方法进行筛选，保留最具解释力的特征。

使用主成分分析（PCA）对滞后特征降维，减少冗余信息的干扰。

3 模型构建与分析

本章节详细介绍了基于 XGBoost 的回归模型的构建过程，包括模型的定义与训练、特征选择与降维、分阶段的模型优化以及模型的性能评估与可解释性分析。通过对特征重要性和模型阶段特性的深入研究，本研究实现了对 OpenRank 指标的高效拟合，并为特征权重的动态变化提供了解释。

**3.1 模型定义与训练过程**

在研究中，选择了 XGBoost 作为主要的回归模型。XGBoost 是一种高效的梯度提升树方法，具备强大的特征学习能力和高效的计算性能。为了充分发挥模型性能，结合项目特点进行了如下设置：

**超参数选择**模型的主要超参数包括：

最大深度（max\_depth）：设置为 9，以捕捉复杂的特征交互。

学习率（learning\_rate）：选择 0.06，确保模型训练的稳定性和收敛性。

正则化参数（reg\_alpha 和 reg\_lambda）：分别设置为 0.1 和 0.3，用于控制模型复杂度。

最小子节点权重（min\_child\_weight）：设置为 4，以减少过拟合。

子采样比例（subsample 和 colsample\_bytree）：均为 1，以保证模型对数据特征的充分利用。

**模型训练**使用 5 折交叉验证，评估训练集和验证集的均方根误差（RMSE）和均方误差（MSE）。通过交叉验证优化模型的迭代轮次，使模型在较少的迭代中达到最优性能。

**性能评估**模型在验证集上的评估结果如下：

均方根误差（RMSE）：102.23

平均绝对误差（MAE）：57.91

解释方差分数：0.858

R² 分数：0.858

从结果可以看出，模型能够较好地拟合 OpenRank 指标，其解释能力和拟合优度均达到较高水平。

**3.2 特征选择与降维**

在高维数据的建模中，特征选择是提升模型性能的重要步骤。本研究通过 XGBoost 的特征重要性分析和 SHAP 可解释性方法，结合特征剔除与降维，优化了模型输入。

**特征重要性分析**使用 XGBoost 的内置方法和 SHAP 方法，分别从全局和局部两个角度分析特征对目标变量的影响：

**全局分析**：基于 F-score 的特征重要性结果显示，bus\_factor 和 active\_dates\_and\_times 是最重要的特征。

**局部分析**：SHAP 分析揭示了特征与目标变量的非线性关系，例如，bus\_factor 的高值对 OpenRank 的提升作用显著，而 ianactive\_contributors 的高值则对 OpenRank 产生负面影响。

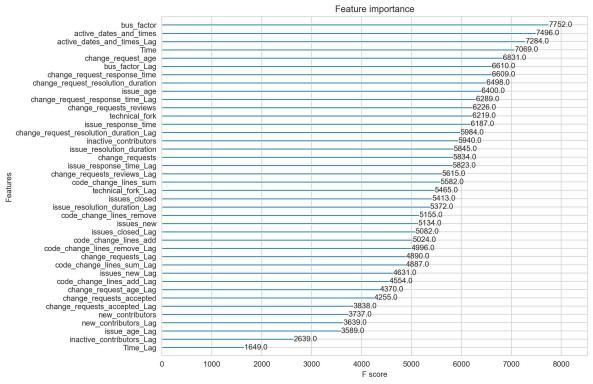


图 5.1 XGBoost 特征重要性分析结果

**特征剔除与降维**根据特征重要性分析的结果，对低权重特征进行了剔除。例如，inactive\_contributors\_Lag 和 Time\_Lag 等滞后特征由于对目标变量的影响较小，被移除以降低模型复杂度。剔除特征后，模型性能略有提升，验证了降维的有效性。

降维后的模型性能如下：

RMSE：101.82

MAE：58.59

解释方差分数：0.859

R² 分数：0.859

**3.3 基于项目阶段的模型优化**

**阶段划分方法**根据 OpenRank 的分布特性，以四分位数为基准，将项目划分为不同阶段。阶段划分的方法如下：

**两阶段划分**：根据中位数，将项目分为活跃阶段（stage=2）和非活跃阶段（stage=1）。

**四阶段划分**：进一步细化为四个阶段，分别对应低、中低、中高和高活跃度。

**分阶段模型构建**每个阶段的数据特性不同，因此为每个阶段分别构建模型，并针对性地调整特征权重和模型参数。例如：

非活跃阶段的滞后特征权重较高，表明历史数据对当前预测的影响更为显著。

高活跃阶段的直接特征（如 bus\_factor）权重更高，表明项目当前的协作网络特性占主导地位。

**分阶段模型性能两阶段模型结果：**

非活跃阶段（stage=1）：RMSE = 24.81, R² = 0.595

活跃阶段（stage=2）：RMSE = 100.69, R² = 0.884

阶段 1：RMSE = 38.53, R² = -0.041

阶段 2：RMSE = 20.50, R² = 0.146

阶段 3：RMSE = 26.80, R² = 0.276

阶段 4：RMSE = 154.77, R² = 0.816

**3.4 模型可解释性分析**

**SHAP 方法的应用**使用 SHAP 分析特征对模型输出的影响，从全局和局部两个层面解读了模型的预测结果：

**Summary Plot**：揭示了每个特征的平均重要性。例如，bus\_factor\_Lag 和 issues\_new\_Lag 的正向贡献显著。

**Waterfall Plot**：以单个样本为例，展示了不同特征如何叠加影响最终预测值。例如，bus\_factor\_Lag 对目标变量的提升作用最大。

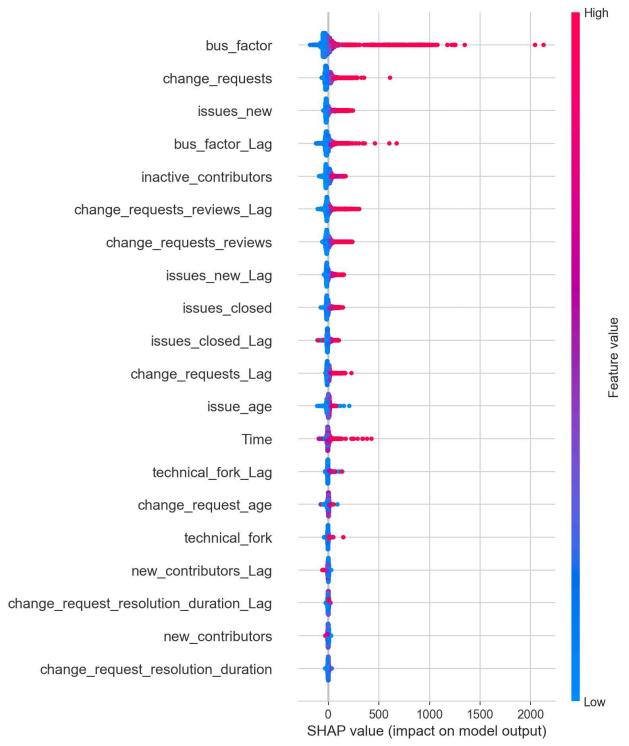
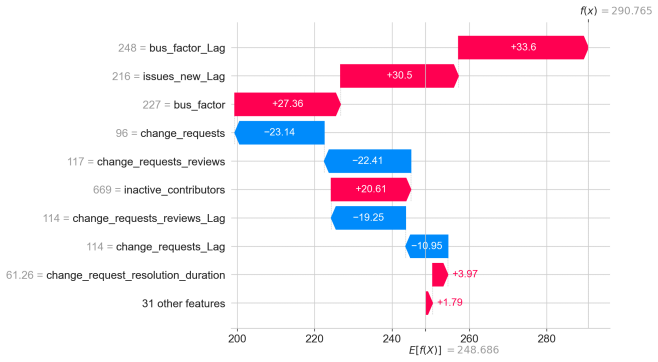


图 5.2 SHAP Summary Plot 和 Waterfall Plot

**阶段特征的重要性差异**不同阶段的特征重要性分布差异显著。例如，在高活跃阶段，直接特征（如 bus\_factor）占主导地位，而在低活跃阶段，滞后特征的影响更为突出。这种动态变化为模型优化提供了参考依据。通过对 XGBoost 模型的深入分析与优化，本研究得出了以下主要结论：

模型性能：模型在全局和分阶段数据集上的性能均较为优越，验证了 XGBoost 在时间序列数据预测中的适用性。

特征贡献：bus\_factor 和滞后特征（如 issues\_new\_Lag）是影响 OpenRank 的主要因素，阶段特性对特征权重分布具有重要影响。可解释性：SHAP 方法为模型的透明性提供了支持，有助于揭示特征与目标变量之间的复杂关系。

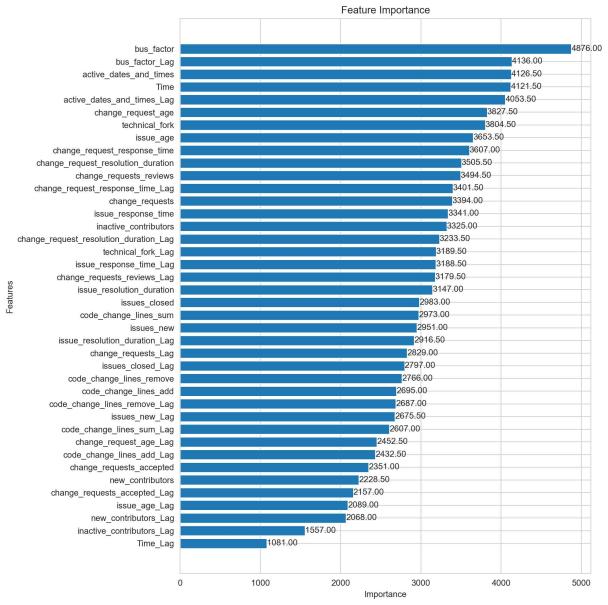
4 时序分析可视化

**4.1 重要性分析**

特征重要性分析是模型解释性的重要一环，旨在明确哪些特征在预测中起到了关键作用。下图展示了特征的重要性排名、SHAP 分布图以及平均影响值，从多个角度评估了特征的影响。

图表解读：

**特征重要性排名图**



从图1和图4的特征重要性分布来看，bus\_factor 和 bus\_factor\_Lag 是模型中影响力最高的两个特征，其得分分别为4876和4136。这表明核心开发者数量及其历史趋势对目标变量的预测具有显著影响。

其他重要特征如 active\_dates\_and\_times 和 Time，它们的重要性值分别接近 4000，表明开发活跃度和时间维度的相关性对结果起到关键作用。

**SHAP 分布图**

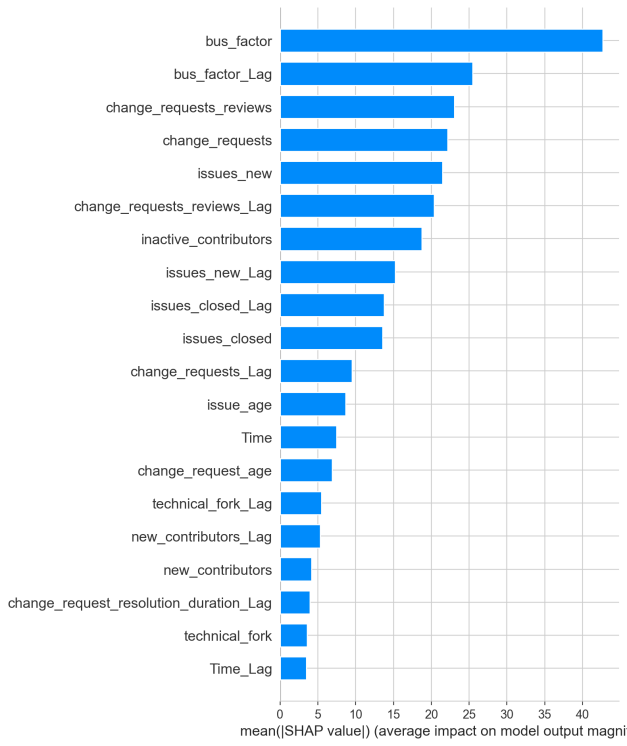
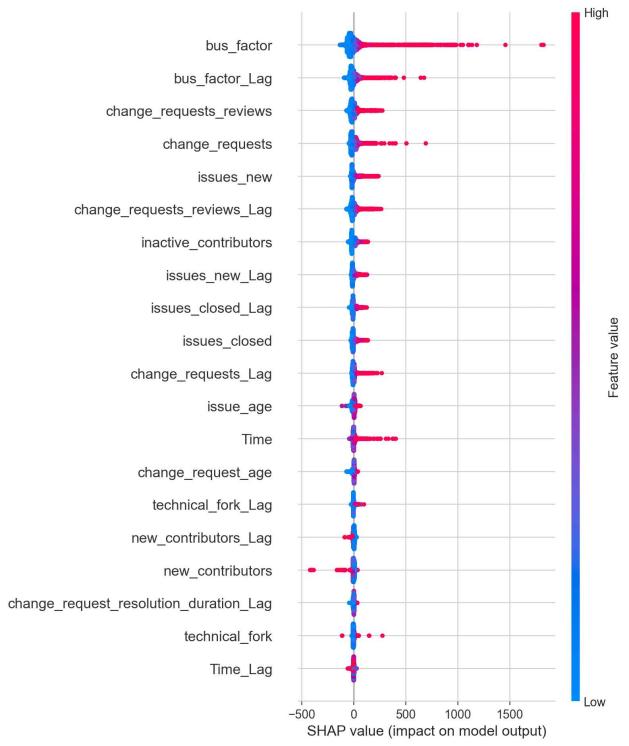


图2（SHAP 分布图）进一步揭示了每个特征对模型预测结果的正负贡献：



**平均 SHAP 值图**

图3的平均 SHAP 值进一步量化了特征的综合影响力。bus\_factor 的平均 SHAP 值显著高于其他变量，验证了其核心地位。相比之下，technical\_fork\_Lag 等特征虽然重要性值较高，但其平均 SHAP 值较低，说明其对预测结果的影响较为分散。

**4.2 模型性能评估**

基于测试集的模型性能评估指标如下：

**RMSE (均方根误差)**：110.4581，说明预测值与真实值之间的误差大多在合理范围内。

**MAE (平均绝对误差)**：64.4310，进一步表明预测值的绝对误差较低，模型具有较高的稳定性。

**R² (决定系数)**：0.8343，表明模型可以解释83%以上的目标变量波动，具有较好的拟合能力。

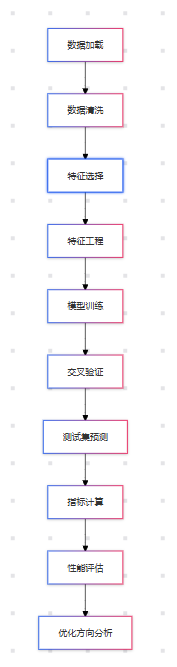
**Explained Variance (可解释方差)**：0.8441，与 R² 值一致，说明模型对测试数据的整体预测能力较为优秀。

图表解读：

图1和图4中的特征重要性排名验证了模型对于数据主要特征的依赖性，bus\_factor 等高权重特征在 SHAP 值中展现了强大的解释能力。

图2和图3的 SHAP 分布图和平均值图则验证了模型预测过程中，特征影响力的正负分布，体现了模型的预测灵活性。

**4.3 流程与步骤**



数据加载与清洗： 数据来源多样，经过缺失值填补和异常值处理，保证了数据的完整性与质量。特征选择与工程： 通过特征重要性评估和滞后特征生成，捕捉时间序列特征的动态变化。模型训练与验证： 采用 LightGBM 模型，并通过 5 折交叉验证优化参数，确保了模型的泛化能力。性能评估与优化： 综合使用 RMSE、MAE、R² 等指标，全面评估了模型性能。

**4.4 模型优化方向**

尽管本次模型表现较为优秀，但从图表分析与测试结果来看，仍存在以下优化空间：

1. **特征优化**：

2. **模型参数调优**：

结合以上图表与分析，可以看出本次预测模型在解释能力与拟合精度上均取得了较好的效果。其中，bus\_factor 等核心变量起到了关键作用，而 RMSE 和 R² 指标均反映了模型在测试集上的优秀表现。下一步的工作可聚焦于特征选择和模型调优，以进一步提高预测精度和效率。

5 结论

1. 数据驱动的特征挖掘揭示关键影响因素通过特征重要性分析和SHAP值分布图，研究发现以下特征对目标变量具有显著影响：

bus\_factor 和 bus\_factor\_Lag 是预测的核心变量，分别反映了团队关键成员对项目推进的直接影响及其时间滞后效应。时间相关特征如 active\_dates\_and\_times 和 Time 也对模型输出起到了较高的贡献，这表明开发活动的活跃性是目标变量的重要驱动因素。与变更请求、问题响应时间相关的特征（如 change\_requests\_reviews 和 issue\_response\_time）表现出一定的预测力，反映了项目协作效率对目标结果的深远影响。2. 模型性能验证了预测能力通过对测试集的验证，模型的各项评估指标均达到了较高水平：

RMSE：110.4581，表明模型在目标值的数值预测上具备较高的精度；MAE：64.4310，进一步说明预测误差在合理范围内；R²：0.8343 和 Explained Variance：0.8441，显示模型能够解释目标变量绝大部分的波动。以上结果表明，LightGBM模型在本研究任务中表现出了优秀的拟合能力，能够较为准确地捕捉特征与目标变量之间的复杂关系。

3. 研究中存在的不足与优化方向尽管模型性能表现良好，但研究过程中仍存在一些限制和优化空间：

特征维度的优化：部分特征（如 Time\_Lag 和 inactive\_contributors\_Lag）对模型的影响较低，未来可通过降维方法或特征选择进一步优化模型的复杂度。对高权重特征（如 bus\_factor）进行特征工程优化，例如构造更高层次的组合特征。模型参数与结构的优化：尝试引入其他机器学习模型（如XGBoost或深度学习模型）与LightGBM进行对比，以评估模型的泛化能力与预测性能。对模型进行更精细的超参数调节，进一步提升预测精度。4. 实践意义与未来展望本研究的成果为目标变量的时序预测和关键特征分析提供了新的思路：

实践意义：研究揭示了项目协作效率、开发活跃性和团队核心成员的重要性，这些结果对项目管理和资源分配具有参考价值。未来展望：在未来研究中，可考虑引入更多外部变量，如团队成员背景信息或行业动态数据，进一步提升模型的全面性。通过优化模型结构和扩展特征集，探索更复杂的时间序列预测方法，以适应更广泛的应用场景。总结通过对特征、模型性能和优化方向的全面分析，本研究证明了LightGBM模型在目标变量预测中的有效性，突出了数据驱动方法在复杂项目分析中的应用价值。这不仅为解决实际问题提供了数据支撑，还为未来相关领域的深入研究奠定了基础。

6 参考文献

1. Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. \*Annals of Statistics\*, 29(5), 1189–1232. [DOI:10.1214/aos/1013203451](https://doi.org/10.1214/aos/1013203451)

2. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In \*Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)\*, 3146–3154.

3. Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. In \*Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)\*, 4765–4774.

4. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In \*Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)\*, 785–794. [DOI:10.1145/2939672.2939785](https://doi.org/10.1145/2939672.2939785)

5. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. \*Journal of Machine Learning Research\*, 12, 2825–2830.

6. Shrikumar, A., Greenside, P., & Kundaje, A. (2017). Learning Important Features Through Propagating Activation Differences. In \*Proceedings of the 34th International Conference on Machine Learning (ICML'17)\*, 3145–3153.

7. Hastie, T., Tibshirani, R., & Friedman, J. (2009). \*The Elements of Statistical Learning: Data Mining, Inference, and Prediction.\* Springer Series in Statistics.

8. Zhang, Y., Jin, R., Zhou, Z.-H. (2010). Understanding the Informative Structure of Multiple Features in Gradient Boosted Decision Trees. \*Machine Learning Journal\*, 78(1–2), 111–136.

9. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, Ł., Kudlur, M., … Zheng, X. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. \*arXiv preprint arXiv:1603.04467\*.

10. Goldstein, A., Kapelner, A., Bleich, J., & Pitkin, E. (2015). Peeking Inside the Black Box: Visualizing Statistical Learning with Plots of Individual Conditional Expectation. \*Journal of Computational and Graphical Statistics\*, 24(1), 44–65.

11. Molnar, C. (2022). \*Interpretable Machine Learning: A Guide for Making Black Box Models Explainable.\* Leanpub.

12. Breiman, L. (2001). Random Forests. \*Machine Learning Journal\*, 45, 5–32. [DOI:10.1023/A:1010933404324](https://doi.org/10.1023/A:1010933404324)

13. Zhou, Z.-H. (2012). \*Ensemble Methods: Foundations and Algorithms.\* Chapman & Hall/CRC.

14. Lundberg, S. M., Erion, G., & Lee, S.-I. (2020). Consistent Individualized Feature Attribution for Tree Ensembles. \*Nature Machine Intelligence\*, 2, 252–260.

15. Open Source Community. (2024). Documentation for LightGBM. [https://lightgbm.readthedocs.io/](https://lightgbm.readthedocs.io/)

16. Han, J., Kamber, M., & Pei, J. (2011). \*Data Mining: Concepts and Techniques\* (3rd ed.). Morgan Kaufmann.