



華東師範大學
EAST CHINA NORMAL UNIVERSITY




基于时间序列预测模型的 Github仓库活跃度计算

游明东



1. 绪论
2. 基于Github活动时间序列数据的研究现状
3. 基于Github活动日志的数据集
4. 基于PatchTST的时间序列预测
5. 仓库活跃分数的设计与验证
6. 基于活跃分数的Github OSS仓库分析

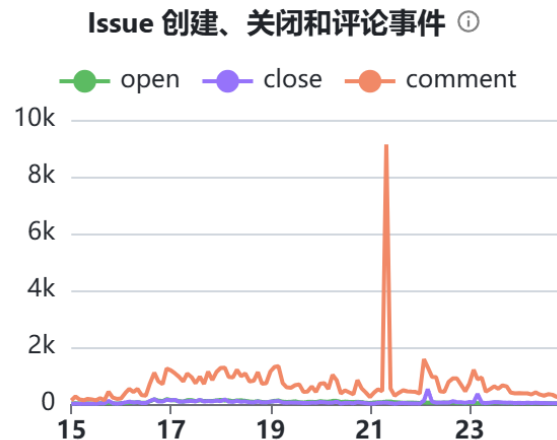
假设一名开发者在2024年初正在寻找一个开源的JavaScript test framework，他/她在一番寻找后列出了如下选择，其中cypress-io/cypress的star数最多，jestjs/jest的fork数最多，这些开源仓库也都是很优秀的仓库，例如jestjs/jest仓库也一度达到接近10k的issue comment数量。该选择哪个去使用呢？

Metric	Star	Fork	Watch
 Jasmine/jasmine	15800+	2200+	438
 jestjs/jest	44300+	6500+	561
 cypress-io/cypress	47600+	3200+	606

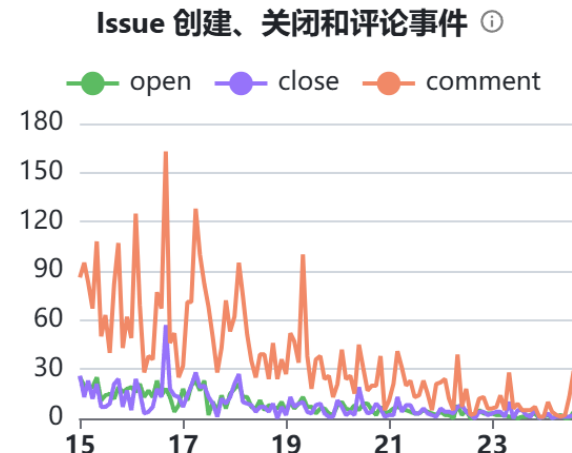
cypress-io/cypress



jestjs/jest



jasmine/jasmine



Github作为世界上最大的开源软件平台，其中丰富的开源资源如软件的关键依赖库 (eg. react/react, flutter/flutter)、科研领域的经典模型 (eg. HuggingFace/Transformer)以及诸多可供学习的资源(eg. awesome-list)为计算机领域和我们的日常生活提供了非常大的帮助。

随着开源软件的优势逐渐被广泛认可，开源软件的数量迎来了快速增长。根据GitHub Octoverse Report近五年的数据（截止2024年），2020年有60 million的仓库在Github上创建，2021年有61 million，2022年有86 million，2023年有98 million，2024年有108 million。

但是其中大部分的项目都遇到了保持长期发展的困难。当我们在Github平台搜索“is this dead”时，将会得到超过375k的issues结果，这表明了解一个项目未来是否还能继续发展良好发展是人们十分感兴趣的话题。

同时大量开源软件不明朗的未来容易使得用户们在选择时产生困难，因为他们不知道这个开源软件仓库能否继续保持良好的发展从而长期满足自己的使用需求。因此预测开源软件仓库未来的发展是有必要的。

之前的相关工作所提出的方法往往不能充分考虑到开源软件(OSS)仓库的活动数据，例如Khondhu et al.和Wenxin Xiao et al.仅考虑了commit来判断项目是否保持活跃。而Jailton Coelho et al.的工则没有充分考虑到OSS仓库中的Issue comments和Pull request review comments等文本活动数据。同时Wenxin Xiao et al.和Jailton Coelho et al.的对时间序列数据的利用是输出一个二元分类结果，本质上是分类而不是预测，没有利用到未来的数据。

在本文中，为了解决人作为OSS仓库活动数据生产的核心天然具有的易受外界干扰以及OSS活动数据非线性、周期性和偏态分布的特征，同时为了能够给OSS仓库发展做出更细致的评价，本文做出了如下两点贡献：

- 一个可供时间序列预测模型使用的Github OSS仓库活动指标数据集。其中包括Open pull requests 和 Merge pull requests等代码协作相关的活动指标以及 Issue comments 和 Pull request review comments等文本活动指标。同时数据集中每个OSS 仓库约有700个时间步，可供进行多种时间配置的预测任务。
- 一个能够表示OSS仓库未来活跃度的分数。该分数在计算过程可以极大减小OSS仓库刷指标对分数的影响，充分考虑OSS仓库的主要活动指标。并且能够不依赖OSS仓库活动指标绝对值，对大型和小型仓库都进行合理的打分。



2. 基于Github活动时间序列数据的研究现状

Trend Prediction of GitHub using Time Series Analysis

该文章的**数据集**是从 GH Archive 上收集，包含的属性有：Fork events, Push events, Issue events, Pull request events。

该文章的**主要方法**是利用LSTM对时间序列数据进行预测，在预测出的数据中利用预测时间步里发生的相应events的绝对值计算一个分数，分数计算方式为简单的线性多项式，其中权重按照作者自己的经验选取，针对每个不同的仓库会采取不同的权重，但各events权重**不公开**。

$$score_t = \sum_i W_i X_i$$

该文章的**主要问题**是权重不公开，分数计算没有说服力，**各仓库的分数计算不在同一个标准**，无法对各个仓库进行公平的打分。同时该文章在验证时序预测有效性上的方法是有问题，他利用预测后的指标计算的分数和实际值计算的分数进行对比来计算误差，作者可以通过调整权重使得预测不好的指标在分数的贡献减小，从而达到和实际分数贴合的效果。

How Early Participation Determines Long-Term Sustained Activity in GitHub Projects?

该文章的**数据集**是从 GHTorrent dump(version 2021-03-06) 上收集，包含的变量很丰富，commit, issue, pull request, contributors等等相关events都有所包含。

该文章的**主要方法**是利用commit事件对项目的是否活跃打标签，该文为仓库在 t years是否活跃打标签的方法是判断其在 t years内每个月的commit的中位数大于等于 k 。如果满足则认为其保持活跃否则为不活跃。再利用数据集中的完整数据对Xgboost模型建模，该模型会输出一个二元分类结果，用来标示项目是否活跃，从而得到实验结果的准确率指标等等。同时由于该项目的数据集利用的是项目早期的数据（1/3/5个月），所以可以探索项目早期因素对项目长期活跃度的影响。

该文章的**主要问题**是其打标签的方法，该方法对仓库是否保持活跃的判断仅仅依赖于commit事件，没有对仓库的其他属性综合考虑。

Is this GitHub project maintained? Measuring the level of maintenance activity of open-source projects

该文章的数据集是来自GitHub根据stat排名的 top-10000的仓库，具体变量如下。但是其数据集中涵盖的领域不够全面，例如缺少topic为books和awesome-lists的仓库。

Features used to identify unmaintained projects.

Dimension	Feature	Description
Project	Forks	Number of forks created by developers
	Open issues	Number of issues opened by developers
	Closed issues	Number of issues closed by developers
	Open pull requests	Number of pull requests opened by the developers
	Closed pull requests	Number of pull requests closed by the developers
	Merged pull requests	Number of pull requests merged by the developers
	Commits	Number of commits performed by developers
	Max days without commits	Maximum number of consecutive days without commits
	Max contributions by developer	Number of commits of the developer with most commits
Contributor	New contributors	Number of contributors who made their first commit
	Distinct contributors	Number of distinct contributors that committed
Owner	Projects created by the owner	Number of projects created by a given owner
	Number of commits of the owner	Number of commits performed by a given owner

Is this GitHub project maintained? Measuring the level of maintenance activity of open-source projects

该文章的主要方法是利用数据收集截止日期最近一个月是否有release的提交来作为是否活跃的门槛并给仓库打标签，接着利用随机森林模型对多个特征的时间序列进行建模并输出分类结果，也是一个分类任务

该文章在活跃度上的评分是利用基于仓库活动指标训练的随机森林模型针对Github仓库给出一个是否活跃的概率 p 。 p 值的范围是 $[0.5, 1]$ 。 p 越靠近0.5，则认为该仓库越接近“未维护状态”，反之越靠近1，则认为该仓库越靠近“活跃”状态。其活跃度计算公式如右下所示，文章根据 p 值范围将概率映射到了一个百分制的分数。

该文章的主要问题是其打标签的方法，与上一篇论文相似，该方法对仓库是否保持活跃的判断仅仅依赖于release事件，没有对仓库的其他属性综合考虑。

Prediction results (mean of 100 iterations, using 5-cross validation); best results are in bold.

Metrics	0.5 Year		1 Year			1.5 Years		2 Years		
	3 mos	6 mos	3 mos	6 mos	12 mos	3 mos	6 mos	3 mos	6 mos	12 mos
Accuracy	0.90	0.91	0.91	0.90	0.89	0.91	0.90	0.92	0.91	0.90
Precision	0.83	0.87	0.87	0.84	0.82	0.86	0.83	0.86	0.85	0.83
Recall	0.78	0.74	0.77	0.75	0.72	0.78	0.76	0.81	0.79	0.73
F-measure	0.80	0.79	0.81	0.79	0.77	0.82	0.79	0.83	0.82	0.78
Kappa	0.74	0.74	0.76	0.73	0.70	0.76	0.73	0.78	0.76	0.71
AUC	0.86	0.85	0.86	0.85	0.83	0.87	0.85	0.88	0.87	0.84

$$LMA = 2 * (p - 0.5) * 100$$

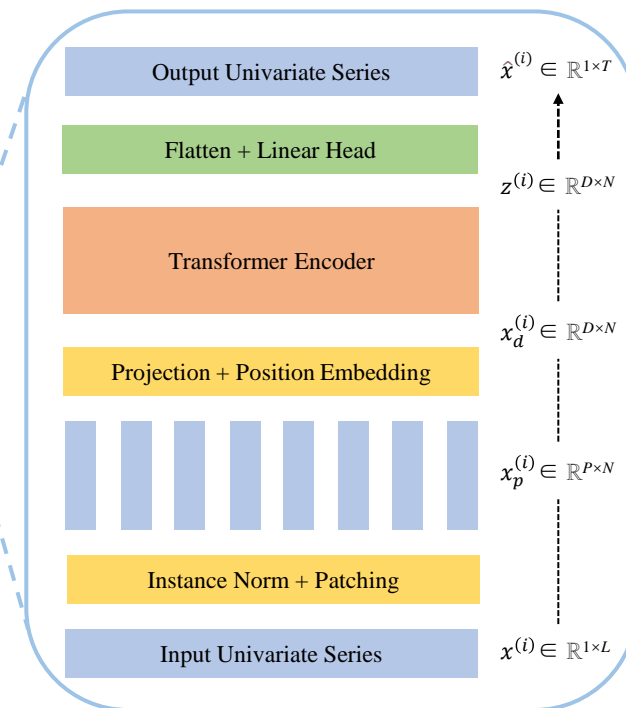
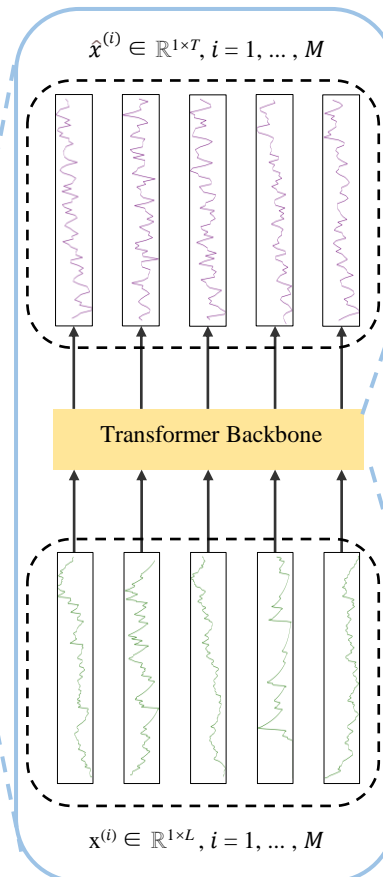
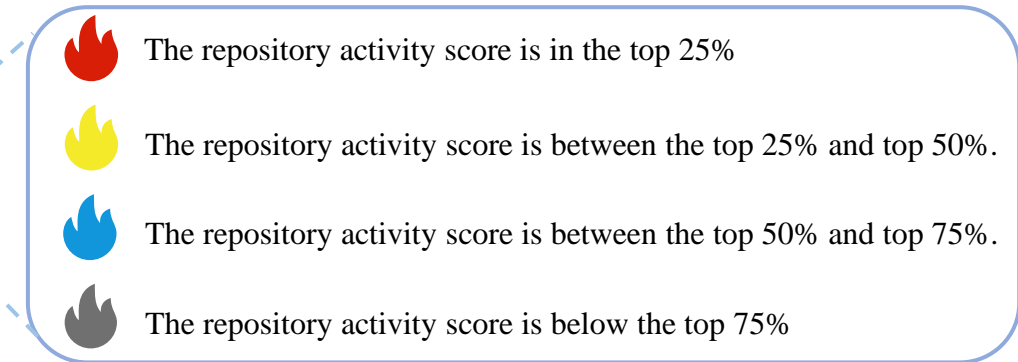
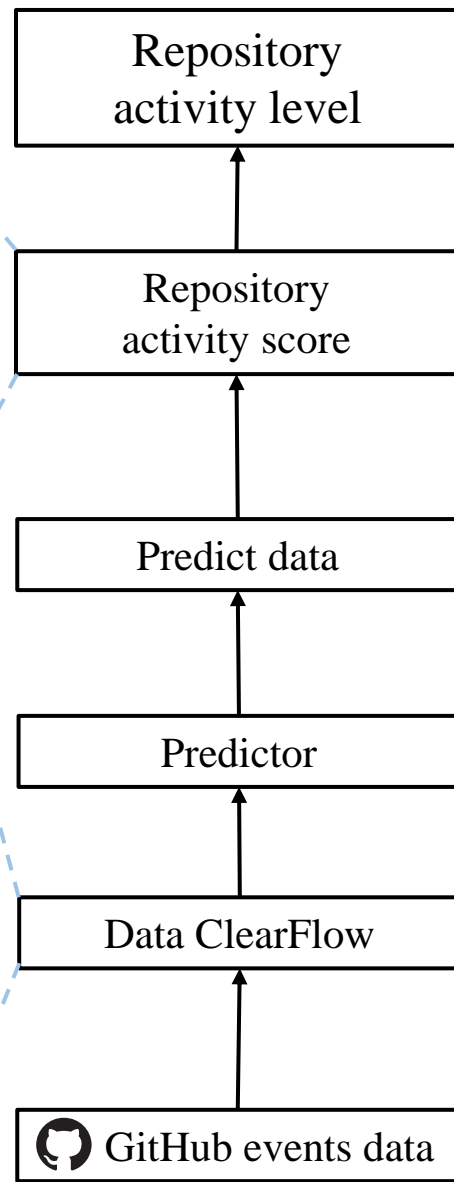
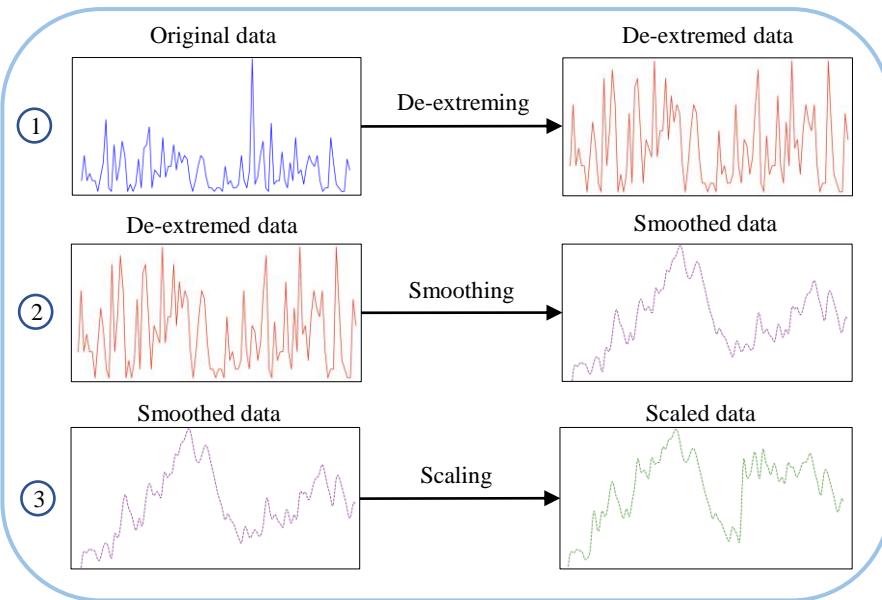
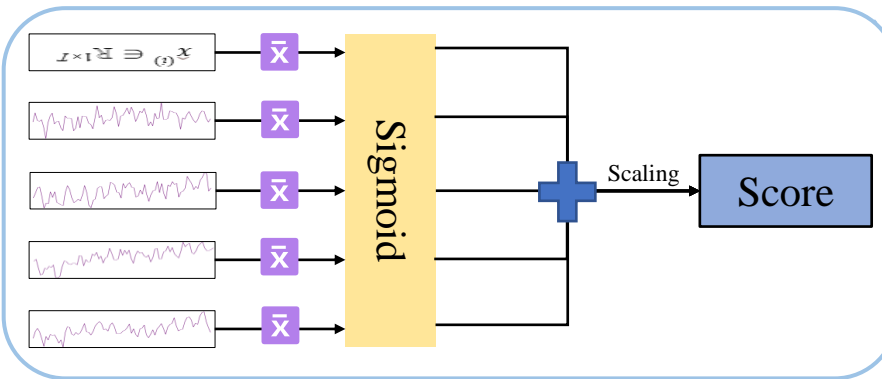
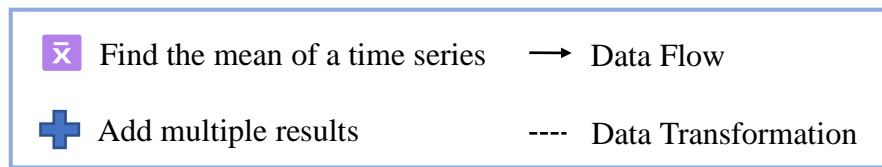
通过上述三个文献对目前研究**现状的问题**做一个总的描述：

1. **基于分类模型的训练数据标签问题**：两篇基于时间序列数据进行分类的文章在思路类似的，二者给训练数据打标签时都仅仅只考虑仓库活动数据的某个单一属性，没有对仓库活跃数据进行充分的利用，同时**仓库是否活跃的标签缺乏统一标准**。
2. **分数计算方式**：目前的分数计算方式一个是基于时间序列内事件数量的线性多项式，权重的拟定不透明且无法对仓库进行统一标准的活跃度评定，同时由于使用的是指标绝对值，即使权重公开也无法将大型和小型仓库放入同一个尺度比较。另外是基于分类模型给出的分类概率放缩到一个分数，这样分数会与训练数据的标签有关，而目前研究工作中的训练数据打标签的方式存在不合理之处。

因此在这里我提出本文的两个贡献点

1. **一个无需活跃标签的OSS仓库活动时间序列数据集**: 在没有统一的仓库活跃标准的情况下, 将目光放到活动指标本身是比拟定一个活跃标签更合理的选择。同时该数据集的时间粒度更细, 能够同时用于多种时间粒度的科研任务。
2. **一个公平的仓库活跃度评分**: 针对之前提到的不公开权重以及指标绝对值的情况, 在数据集中我们将大型和小型的仓库的活动数据范围放缩到同一个标准, 并且对采用的活动数据在结果计算的时的权重公开且统一标准。另外也规避了分类模型预测时对标签的依赖问题。

主框图





3. 基于Github活动日志的数据集

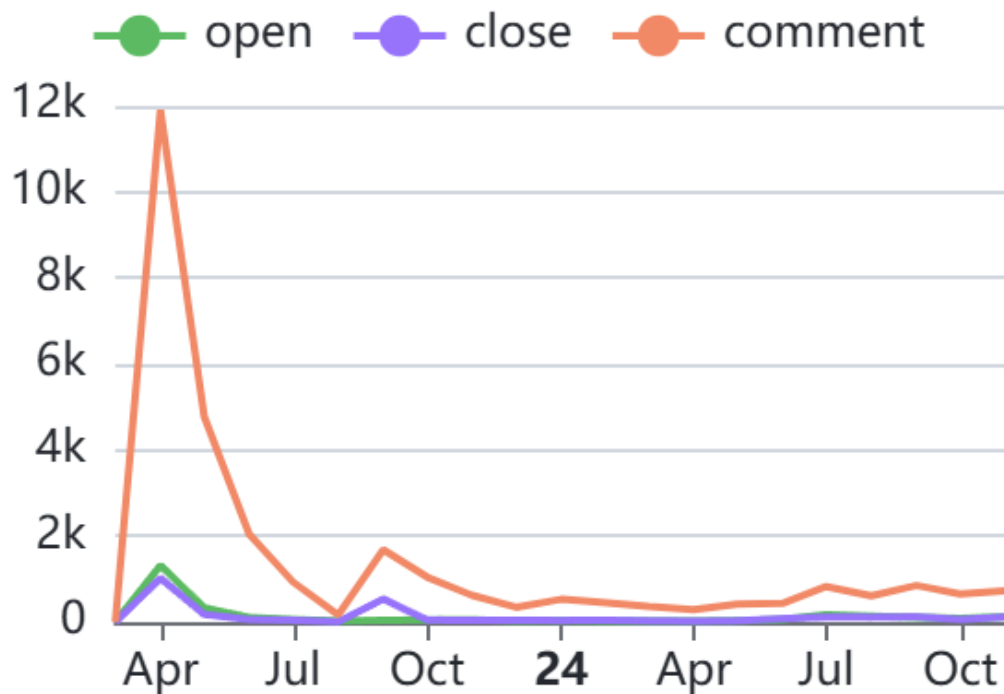
数据集制作：该数据集从是利用华东师范大学X-lab实验室的opendigger工具进行数据采集。仓库选自GitHub上根据star数排名前一千的仓库，时间段为2021年1月1日00:00整至2023年1月1日00:00整。每个数据集的范围不完全一致，具体为上述时间范围(2021年1月1日00:00整至2023年1月1日00:00整)内有活动记录的第一天到有活动记录的最后一天。

制作数据集面临的挑战：

- 由于数据集中events的日期并不连续，不同活动数据产生的记录时间不一致，所以需要对各活动数据在日期上范围上进行统一，保证在连续的时间步内都有数据。
- 在实际科研中OSS仓库的活动数据由于天然具有的易受外界干扰、活动数据非线性、周期性和偏态分布的特征，所以必须要活动数据进行适当的处理同时尽可能保有原始数据的特征。

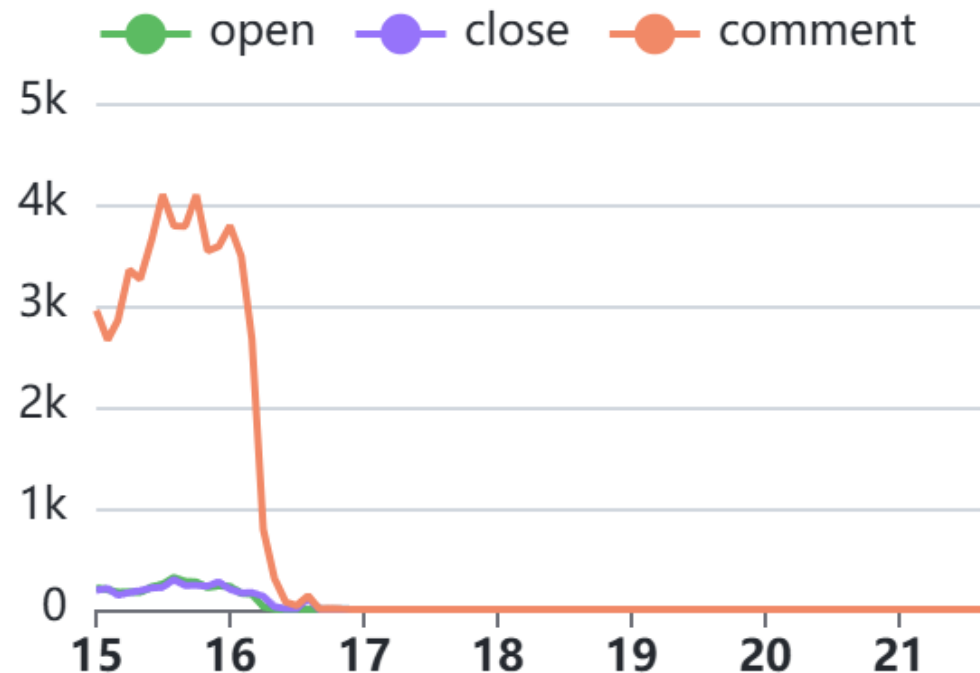
偏态分布实例: GitHub仓库
Significant-Gravitas/AutoGPT

Issue 创建、关闭和评论事件 ⓘ



人为因素实例: GitHub仓库
Homebrew/legacy-homebrew

Issue 创建、关闭和评论事件 ⓘ



数据集清洗方式:

- 通过z-score分数剔除异常值。z-score主要的应用是测量原始数据与数据总体均值相差多少个标准差。在本文中，使用 $|z\text{-score}|=2$ 作为异常处理的阈值，一般来说，该阈值内大概有95%的数据，因此超出阈值时认为其是异常值。异常值使用滚动窗口均值代替。
- 通过指数移动加权平均法对数据集进行平滑，避免人为因素或其他因素导致的噪声。加权移动平均法，是对观察值分别给予不同的权数，按不同权数求得移动平均值，并以最后的移动平均值为基础，确定预测值的方法。采用加权移动平均法，是因为观察期的近期观察值对预测值有较大影响，它更能反映近期变化的趋势。具体计算公式如下：

$$S_t = \alpha \cdot X_t + (1 - \alpha) \cdot S_{t-1}$$

- 通过最大最小值对活动数据集放缩到同一范围，是的大型和小型仓库能够放到统一标准进行评分

数据集总览： 本文制作的数据集涵盖多种类别，这里从language/user_type/license三个角度出发介绍。

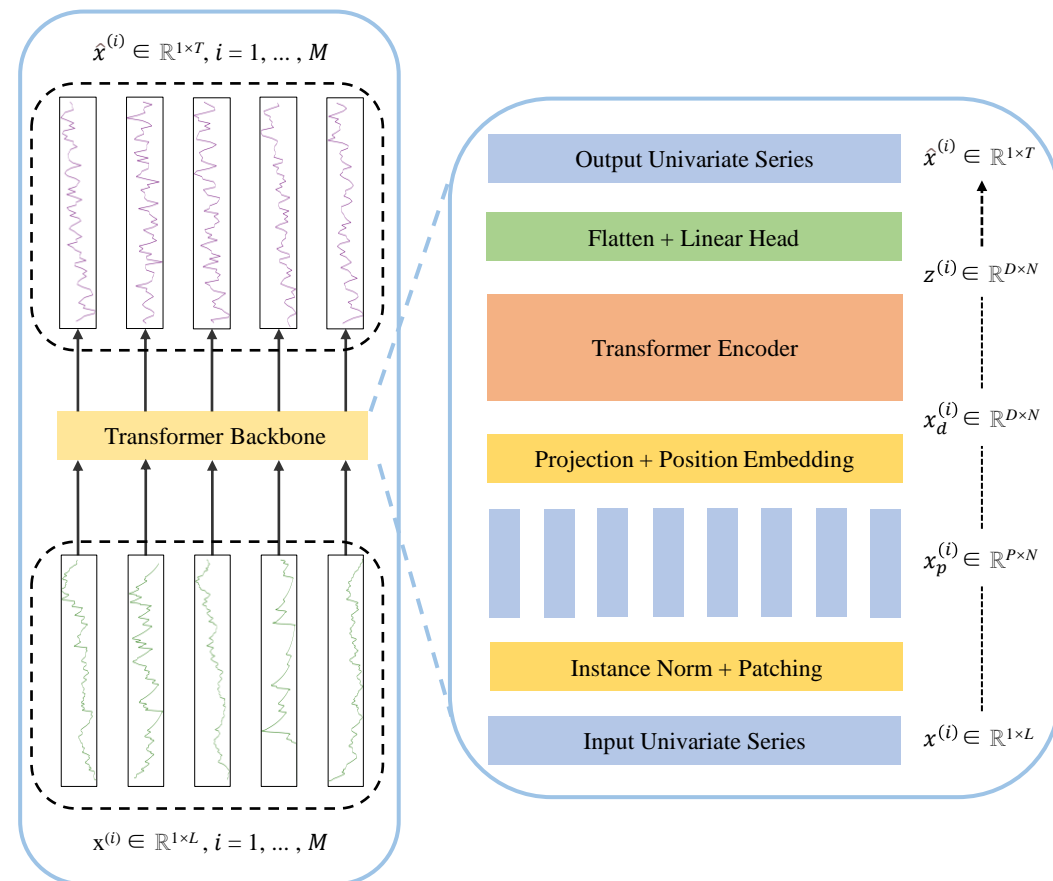
- **Language:** 涵盖JavaScript, Python, Java, C, C++, Go等等语言类型，总的语言类型数目超过40，能够涵盖彼时大多数类型的GitHub仓库语言类型。
- **License:** 涵盖MIT, Apache 2.0, BSD 3-Clause, Creative Commons Zero v1.0 Universal等许可证类型，总许可证类别共19种。
- **User_type:** 本文将user_type分为两种类型，分别是‘组织’用户和‘个人’用户，在本数据集种，二者的占比分为62.6%，37.4%。



4. 基于PatchTST的时间序列预测

PatchTST模型是一种针对时间序列数据的深度学习模型，借用了Transformer架构的思想，通过切分时间序列数据为小块（patches）来提取更高效的特征表示。其目的是提升对长时间序列的建模能力，同时避免传统方法中的计算瓶颈。其主要优点如下：

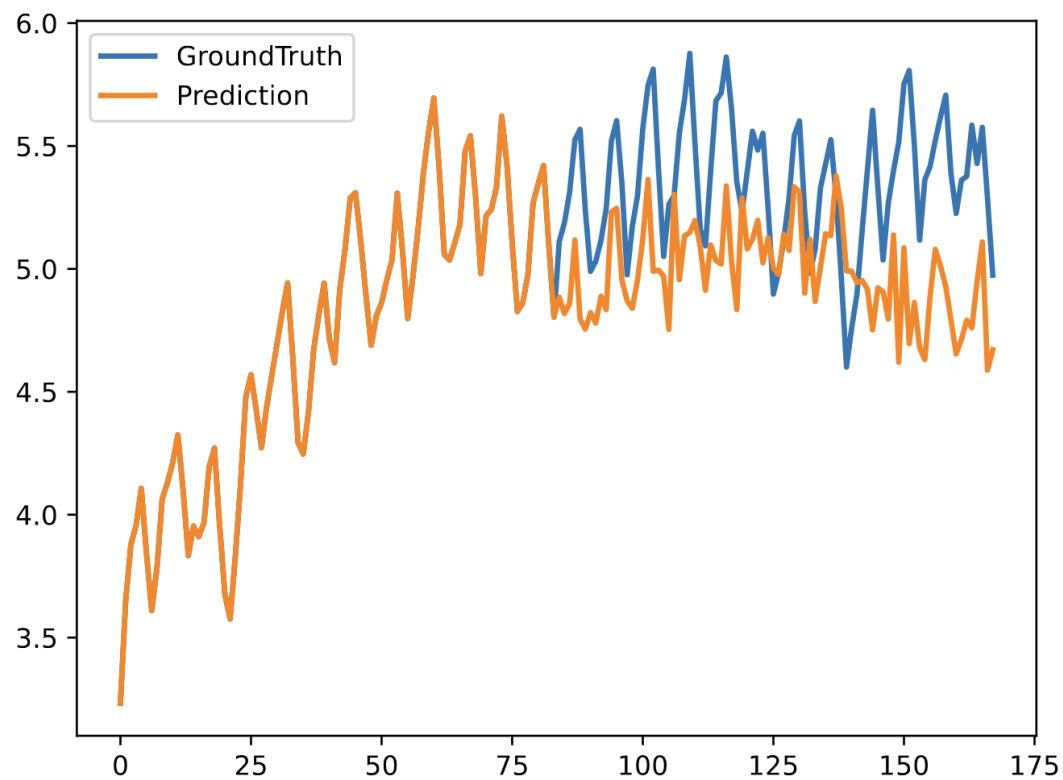
- 相比于之前基于transformer的时间序列预测方法，patch能够保留更多局部语义信息并且可以减少计算量
- 由于通过patch代替时间点进行预测，模型可以关注更长的历史信息。



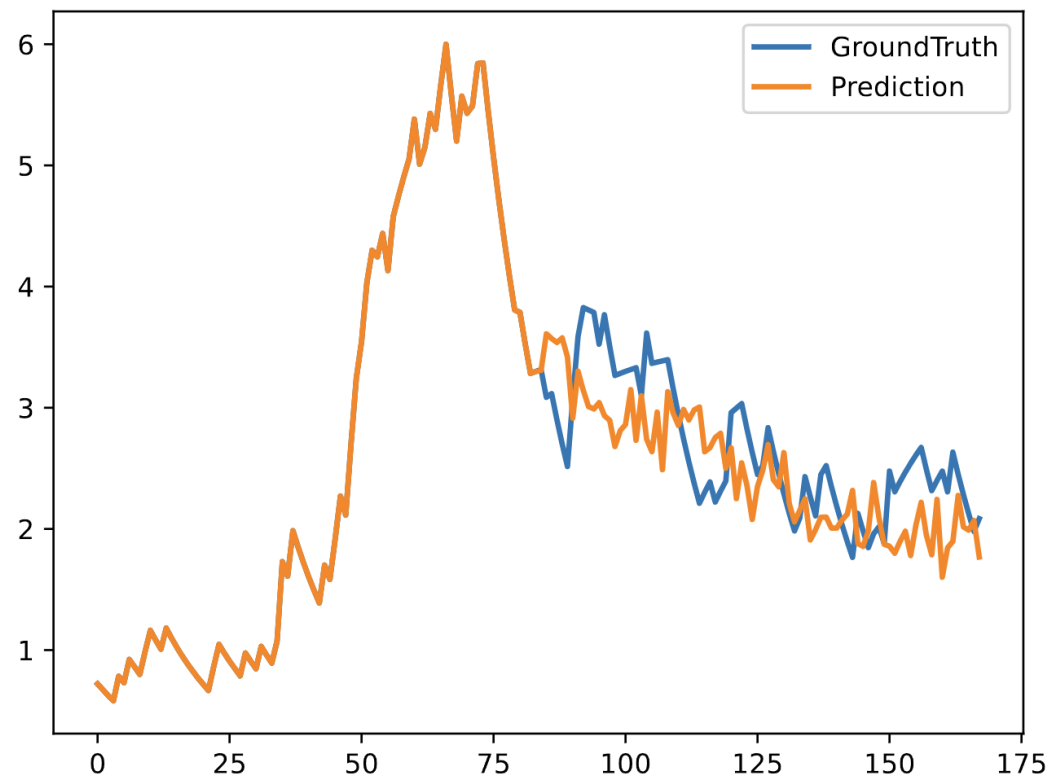
得益于*Exploring Activity and Contributors on GitHub: Who, What, When, and Where*这篇文章在开发者活跃时间上的研究，本文得以确认开发者们的主要活动周期以工作周为单位，在本文的时间步上体现为周期为7。

因此，PatchTST模型在训练和预测的过程中，本文将输入序列长度和预测序列长度定位12个周期，也就是近似三个月的预测长度。即使模型在更短的周期长度的预测任务上能取得误差更小的预测结果，但是更短的预测周期对仓库未来发展的描述说服力是不够的。

OpenPR预测实例: GitHub仓库
microsoft/vscode



OpenIssue预测实例: GitHub仓库
vuejs/vue



PatchTST在预测上的表现结果如下表所示，下表的预测结果分别表示PatchTST对原始数据和经过了cleanflow的数据进行预测后统计的平均误差。误差统计指标为均方误差(MSE)。MSE计算公式如下图所示，均方误差是评价点估计的最一般的标准，自然，我们希望估计的均方误差越小越好。

通过随机抽取数据，可以发现经过cleanflow后的数据普遍预测误差普遍小于原始数据。

Random_seed	1	2	3	4	5
Random100_results_CleanFlow_data	102.3	14.9	6.0	22.9	7.3
Random100_results_Original data	7.8	5.0	5.5	9.6	6.4

$$MSE\left(\hat{\theta}\right)=E\left(\hat{\theta}-\theta\right)^2.$$



5. 仓库活跃分数的设计与验证

公式设计理念：通过数据集处理最后一步的放缩操作，大型仓库和小型仓库的活动指标在同一个范围。

预测模型输入和输出的也都是放缩后的值，其中score公式中 \bar{x} 是未来12个周期的指标均值。 W_i 表示该指标的权重， n 表示变量数量。目前的权重全设为1，之后可能会有修改。该公式有如下优势：

公平统一：得益于放缩，此时的指标值体现的是该指标相较于其历史最高值和最低值的关系，而不是指标数量绝对值。越接近放缩上限表明越接近历史极大值。当然指标的预测值也可以高于历史最高值或低于历史最低值，表明当前的发展水平超越历史最高或者低于历史最低。

边际效应递减：由sigmoid函数的一阶导数可以看出，由于 $\text{Sigmoid}(x)$ 的值始终在0和1之间， $\text{Sigmoid}'(x)$ 会随着 x 增大而减小，意味着边际效益递减。因此可以极大减小仓库中大量制造活动指标对分数的异常影响。

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad \text{Sigmoid}'(x) = \text{Sigmoid}(x) \cdot (1 - \text{Sigmoid}(x))$$

$$\text{Score} = (\sum_{i=1}^n W_i \text{Sigmoid}(\bar{x})) - 2.5 / 2.5 * 100\%$$

公式结果验证：当一个仓库获得高分时，那么理所应当表明该仓库的发展迅猛，其相应的活动指标数量应该也要有迅猛的增长。因此本文选择平均斜率作为衡量活跃度分数是否有效的指标。平均斜率的计算方式如下，其中n为序列长度。

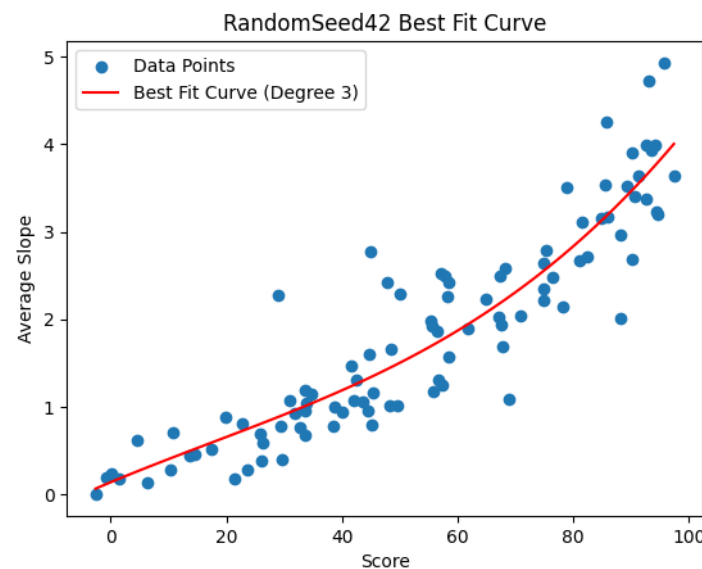
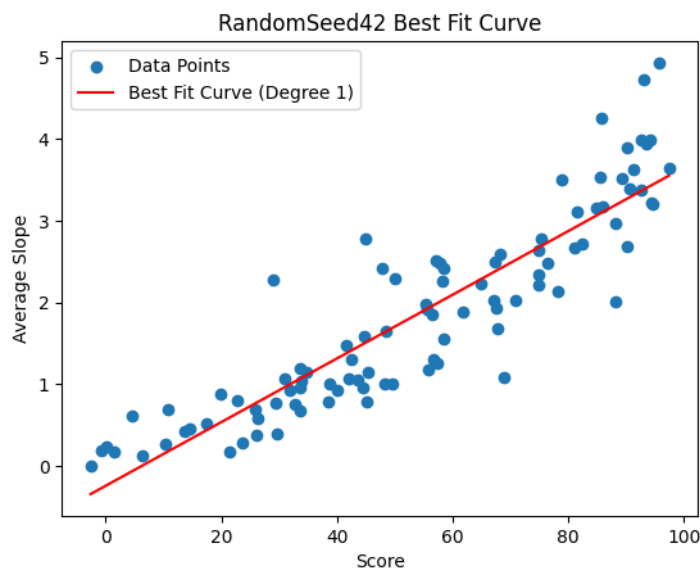
$$\text{平均斜率} = \frac{Y_n - Y_1}{n - 1}$$

假设原始的预测值序列为 $\Delta y = [\Delta y_1, \Delta y_2, \dots, \Delta y_n]$ ，可以将其累积为总值序列 $Y = [Y_1, Y_2, \dots, Y_n]$ 其中：

$$Y_t = \sum_{i=1}^t \Delta y_i$$

公式结果验证：基于计算的分数和平均斜率绘制散点图并使用多项式回归模型拟合一条最佳曲线表示数据点随x轴增长的走势。这里我按照不同类型的language从数据集中进行分层随机抽样，总共抽取一百个仓库，绘制其散点图和拟合曲线，其中x轴表示分数，y轴表示平均斜率。

可以看到无论是一次多项式还是三次多项式拟合。图中除少数离群点以外，大部分数据都贴合最佳拟合曲线，同时**直线/曲线走向单调递增**，表明分数与平均斜率为正相关关系。这表明分数越高，仓库活跃指标增长越多，仓库更活跃



公式结果验证：为进一步验证分数与平均斜率的关系，我通过Spearman相关系数和Person相关系数验证分数与各属性平均斜率和总平均斜率的关系。Spearman相关系数表明X（自变量）和Y（因变量）的相关方向。如果当X增加时，Y趋向于增加，则斯皮尔曼相关系数为正。如果当X增加时，Y趋向于减少，则斯皮尔曼相关系数为负。皮尔逊系数同理。二者的结果域都为[-1, 1]。

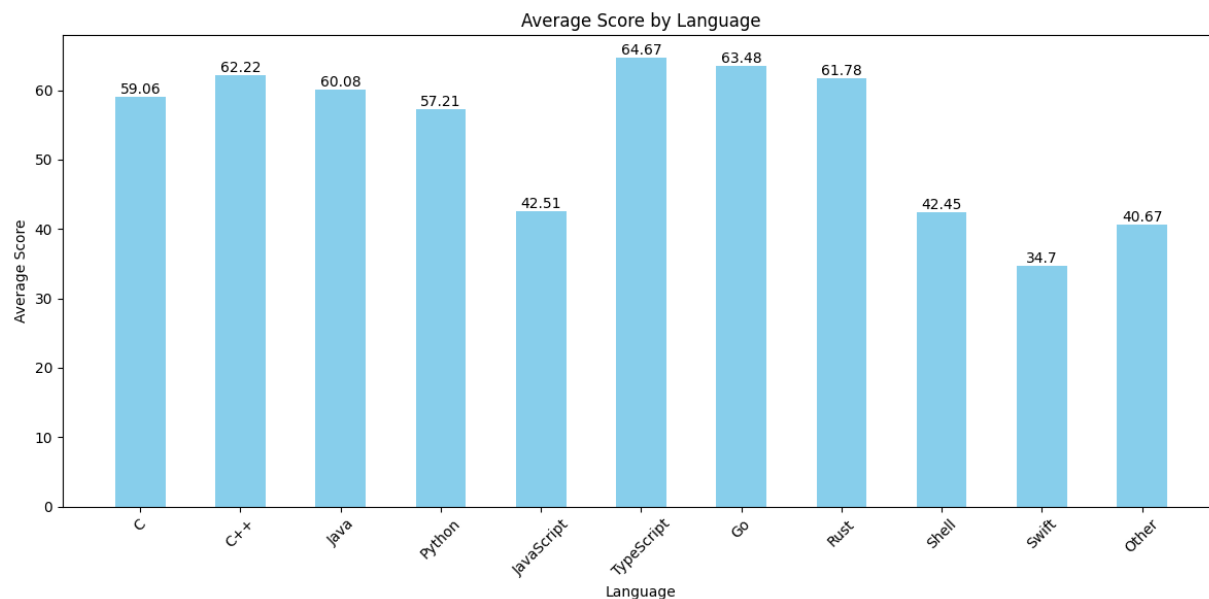
下表结果同样来自分层随机抽样的一百个仓库，其中slope行中的Average表示Average_slope，IssueComment表示IssueComment_slope，其他属性同理，可以看到分数与各属性的平均斜率以及总的平均斜率均有极高的Spearman相关系数和Person相关系数，表明分数与平均斜率的正相关性。

Slope	Average	IssueComment	OpenIssue	OpenPR	ReviewComment	MergePR
Spearman	0.95	0.84	0.73	0.91	0.80	0.88
Person	0.92	0.83	0.73	0.90	0.76	0.86



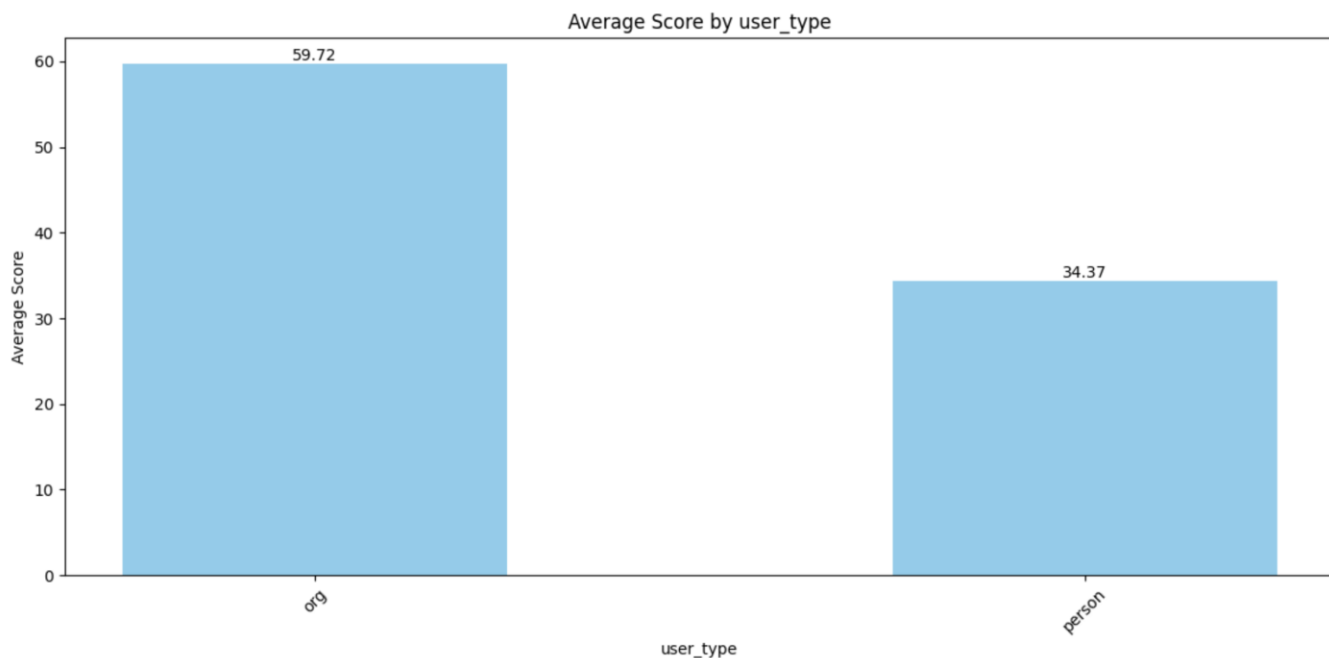
6. 基于活跃分数的Github OSS仓库分析

按照语言分类可以看到在Top1000的项目中，语言类型为TypeScript的仓库平均分数最高，该语言类型的代表仓库为microsoft/vscode，其分数为97.69。均分第二高的类型为Go语言类型，其代表仓库为pingcap/tidb，分数为97.38分，另外该仓库的issue events活动指标如右下图所示，该仓库在23年之后也保持着高歌猛进的势头继续迅猛发展。

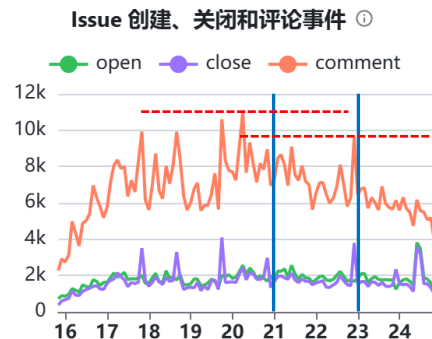


按照用户分类可以看到在Top1000的项目中，组织用户的平均分数显著高于个人用户的平均得分。组织用户中发展迅猛的仓库有著名的microsoft/vscode和python/cpython，分别为97.69，99.09。两个仓库的issue相关活动指标可以右下角中有所展示。

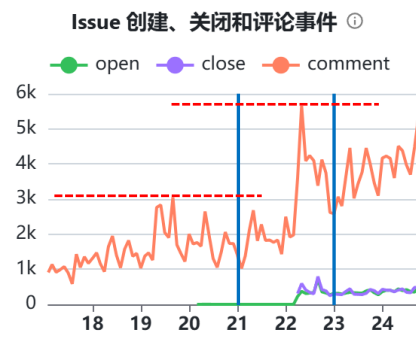
这里可以看到microsoft/vscode虽然是Github排名前几的仓库，但由于2021年至2023年间其活动指标水平相较于历史最高水平较低，其分数则不如在2021年至2023年间高于历史最高水平的python/cpython仓库。这表明本文设计的分数能够体现平稳以及超越自己的发展，同时也表明本文设计的分数能够反映仓库发展趋势。



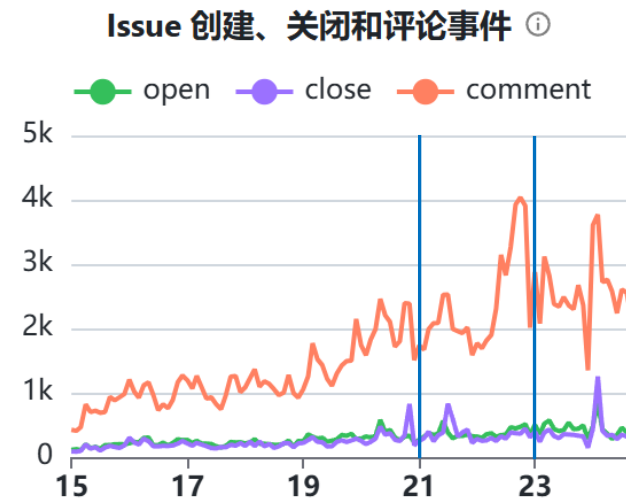
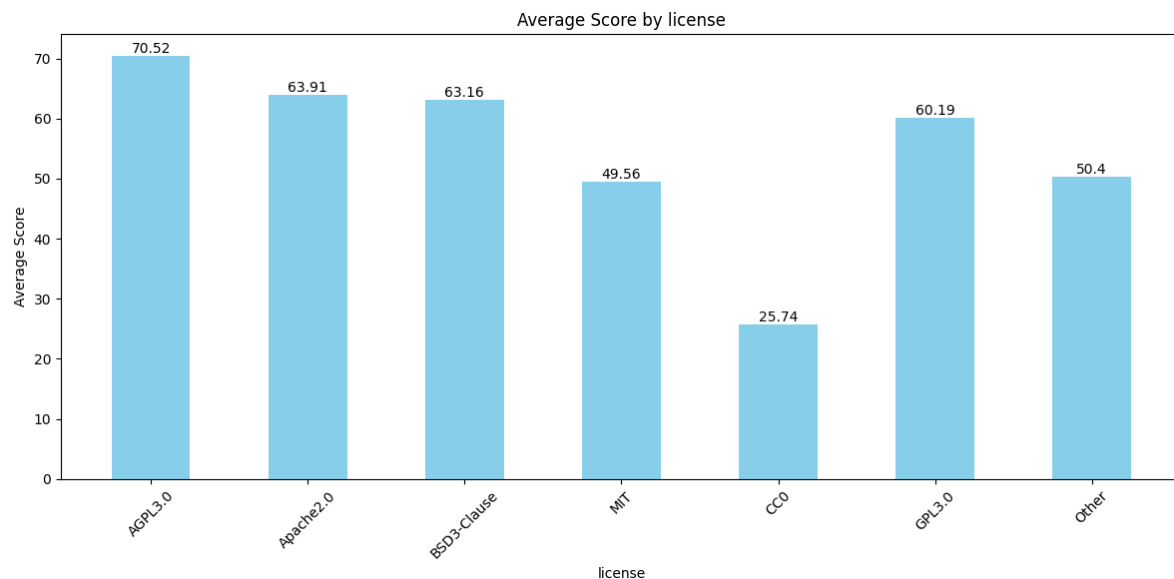
microsoft/vscode



python/cpython

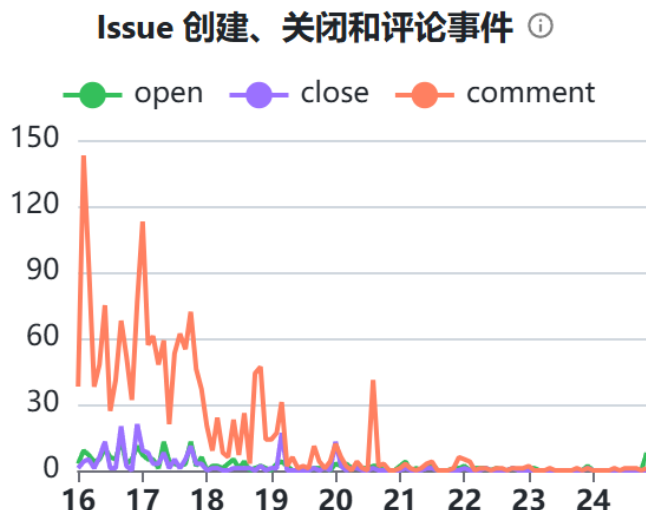


按照许可证分类可以看到在Top1000的项目中，许可证类型为AGPL3.0(GNU Affero General Public License v3.0)平均分最高，其中的代表项目为grafana/grafana，其issue相关活动指标变化如右下图所示。该仓库同时也是前文提到的在语言类别中发展最好的TypeScript类型。排名第二的Apache2.0许可证类型中的代表仓库有pingcap/tidb，同样也是前文提到的发展迅猛的仓库

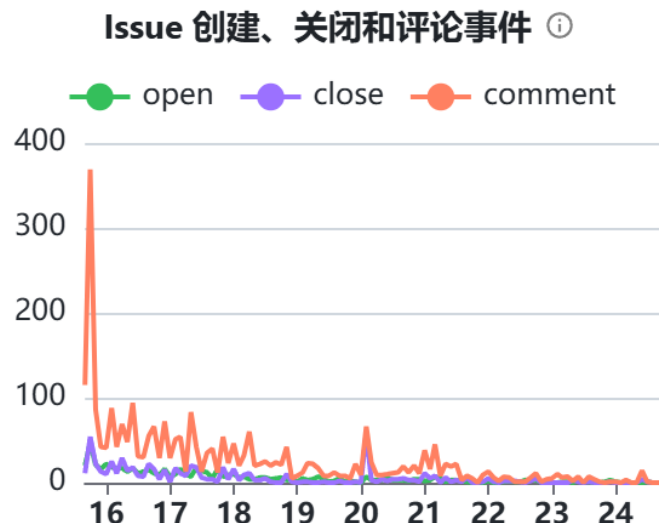


调查三个分类中平均分最低的类别中得分最低的仓库：语言类别中的仓库为swift语言中的kdecocodes/swift-algorithm-club，用户类别中的仓库为个人用户中的zenorocha/clipboard.js，许可证类别中的仓库为CC0（Creative Commons Zero v1.0 Universal）中的Chalarangelo/30-seconds-of-code，三者的得分分别为3.07， -0.72， -2.60。调查发现三个仓库的主要特点为其在仓库成立早期便完善了其主要内容，无需长期维护，因此如今的活跃度评分很低。入选数据集是因为其本身的高质量容易在初期获得大量star。同时调查发现文档性质为主的仓库如swift-algorithm-club和30-seconds-of-code也相比于软件仓库更难获得活动指标。

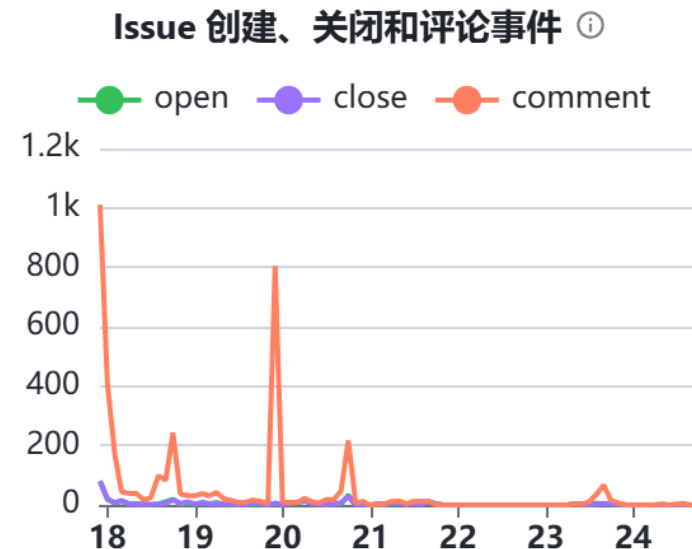
kdecocodes/swift-algorithm-club



zenorocha/clipboard.js

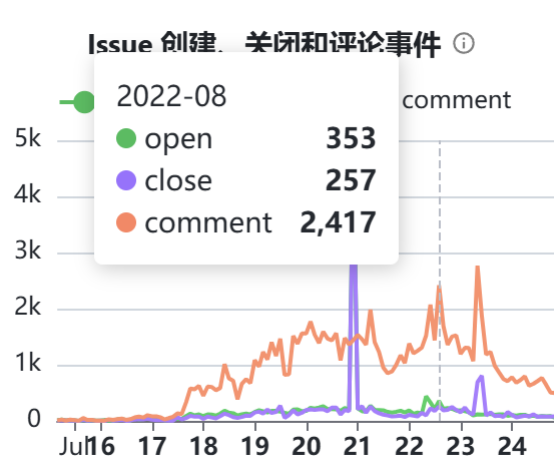


Chalarangelo/30-seconds-of-code

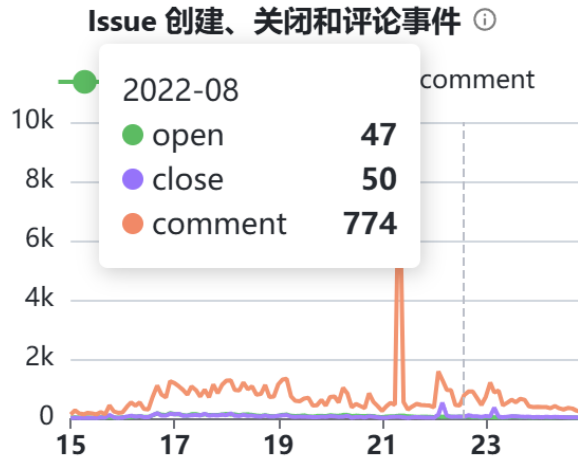


回到最开始的问题，开发者面对三个优秀的JavaScript test framework该如何选择呢？通过我们的分数计算，三个仓库的分数如下图所示。结合三个仓库如今(截止2024年12月9日)的star，fork等指标，即使jest仓库曾经收获极多的comment活动指标并且fork指标也更多，但其如今日常维护活动指标水平却低于cypress仓库。（数据集中所有训练都只利用了21年1月至22年3月的数据）

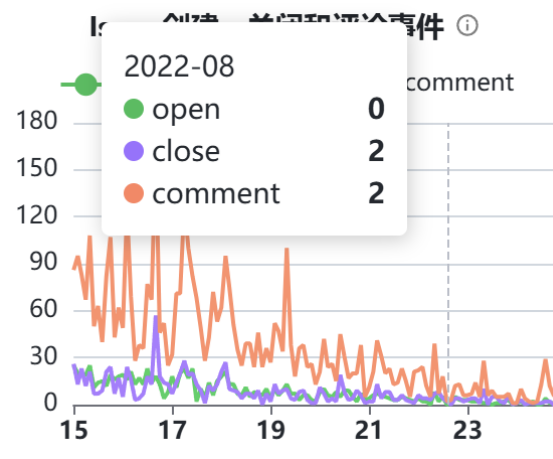
1. cypress-io/cypress: 95.12



2. jestjs/jest: 51.3



3. jasmine/jasmine: 18.90



Metric	Star	Fork	Watch
Jasmine/jasmine	15800+	2200+	438
jestjs/jest	44300+	6500+	561
cypress-io/cypress	47600+	3200+	606

同时除分数以外，本文利用数据集以外的真实世界指标变化实际数量说明分数对未来预估有效性，下面两张表分别代表三个仓库在2023年-2024年(截止24年12月9日)的活动指标总增量和每日平均增量。可以看到分数最高cypress在各项指标上，无论是总增量还是每日平均增量都是三个仓库中的最高，因此本文认为可以推荐cypress-io/cypress给这名开发者进行JavaScript test framework相关工作。

23-24年(截止24年12月9日)活动总增量

Metric	IssueComment	OpenIssue	OpenPR	ReviewComment	MergePR
jasmine	185	30	10	24	5
jest	11606	708	866	1035	552
cypress	<u>22691</u>	<u>2316</u>	<u>1788</u>	<u>4127</u>	<u>1435</u>

23-24年(截止24年12月9日)活动指标每日平均增量

Metric	IssueComment	OpenIssue	OpenPR	ReviewComment	MergePR
jasmine	0.26	0.04	0.01	0.03	0.00
jest	16.37	1.00	1.22	1.46	0.78
cypress	<u>32.00</u>	<u>3.27</u>	<u>2.52</u>	<u>5.82</u>	<u>2.02</u>



華東師範大學
EAST CHINA NORMAL UNIVERSITY

THANKS

求实创造 为人师表