Impactos da Inteligência Artificial no Desenvolvimento de Software na Linguagem Python: Um estudo longitudinal sobre LLMs em projetos open source

Alberto da Costa Reis Junior[1], **Emmanuel Viglioni**[1]
Gabriel Henrique Mota Rodrigues[1], **Matheus Vinicius Mota Rodrigues**[1]
Nelson de Campos Nolasco[1], **Pedro Henrique de Vasconcellos Franco**[1]

[1] [1] Curso de Engenharia de Software – Pontifícia Universidade Católica de Minas Gerais (PUC Minas) Belo Horizonte – MG – Brasil

{alberto.reis, emmanuel.viglioni}@sga.pucminas.br
{gabriel.henrique, matheus.1462287}@sga.pucminas.br
{nelson.campos, pedro.franco}@sga.pucminas.br

Abstract. *This paper examines how Large Language Models (LLMs) have influenced open-source Python projects by comparing activity before and after their widespread adoption. Results indicate clear productivity gains, with more commits and larger code changes, while the number of opened pull requests grows only slightly. Quality trends are mixed: the merge rate increases, but review cycles become marginally longer and bug reports more frequent. Collaboration levels remain steady in terms of unique contributors. Overall, explicit LLM adoption aligns with faster development and integration, yet highlights the need for stronger testing practices, careful review, and sustained software quality monitoring.*

Keywords: *Large Language Models; software development; productivity; quality; collaboration; mining software repositories.*

Resumo. *Este artigo analisa como os Modelos de Linguagem (LLMs) influenciaram projetos Python de código aberto, comparando períodos anteriores e posteriores à sua adoção em larga escala. Os resultados indicam ganhos claros de produtividade, com mais commits e maiores alterações de código, enquanto a abertura de pull requests cresce apenas de forma discreta. Os sinais de qualidade são mistos: a taxa de merge aumenta, mas o tempo de revisão se estende levemente e há mais relatos de bugs. A colaboração se mantém estável em número de contribuidores únicos. Em conjunto, a adoção explícita de LLMs associa-se a um desenvolvimento mais rápido, exigindo atenção redobrada a testes, revisões e práticas de qualidade contínua.*

Palavras-chave: *LLMs; desenvolvimento de software; produtividade; qualidade; colaboração; mineração de repositórios.*

1. Introdução

A rápida difusão dos *Large Language Models* (LLMs) e dos copilotos de programação como o GitHub Copilot e o ChatGPT vem transformando de forma significativa o

ecossistema da engenharia de software. Pesquisas e revisões recentes mostram que essas ferramentas estão sendo usadas em diversas etapas do desenvolvimento, desde a geração de código e execução de testes até o suporte à manutenção e à segurança. Os estudos destacam tanto os benefícios quanto as limitações e riscos associados a seu uso [Hou et al. 2024, Jiang et al. 2025].

Paralelamente, surgem evidências empíricas sobre o impacto dessas tecnologias na prática profissional, abrangendo aspectos como a qualidade e a robustez das sugestões de código [Mastropaolo et al. 2023b], a dinâmica de uso em *pull requests* [Chouchen et al. 2024] e a utilidade percebida no rastreamento de *issues* e na rotina de equipes de desenvolvimento [Das et al. 2024, Dávila et al. 2024, Liang et al. 2024]. Apesar desses avanços, ainda são escassas as evidências quantitativas, em larga escala e de caráter longitudinal, que permitam relacionar a adoção de LLMs a impactos concretos em **produtividade**, **qualidade** e **colaboração** em projetos de código aberto.

Com o intuito de suprir essa lacuna, este artigo investiga empiricamente a relação entre a adoção de LLMs e tais dimensões, por meio da análise de repositórios **Python** hospedados no GitHub, contrastando os períodos *pré-LLM* (2018–2022) e *pós-LLM* (2023–2025). A seleção da linguagem Python justifica-se por sua centralidade em iniciativas de inteligência artificial, bem como pela ampla disponibilidade de repositórios ativos. A delimitação temporal, por sua vez, permite capturar a fase antecedente à difusão massiva de copilotos e a fase subsequente, marcada pela popularização e consolidação dessas ferramentas em comunidades de desenvolvimento [Jiang et al. 2025].

Os constructos de análise adotados estão fundamentados no *Software Engineering Body of Knowledge* (SWEBOK) [Society 2024], guia de referência do IEEE que consolida o corpo de conhecimento essencial da engenharia de software, abrangendo práticas, métodos e áreas fundamentais da disciplina. Nesse enquadramento, são operacionalizadas medidas de **produtividade** (e.g., volume de *pull requests* e commits), **qualidade** (e.g., taxa de *merge*, tempos de integração/fechamento, incidência de issues de defeito) e **colaboração** (e.g., diversidade de autores por mês, dinâmica de contribuição), de modo a favorecer interpretações consistentes com o estado da arte em engenharia de software.

O objetivo geral deste estudo consiste em **avaliar empiricamente os impactos da adoção de LLMs sobre produtividade, qualidade e colaboração em projetos de software em Python de código aberto**. Para tal, são definidos três objetivos específicos: (i) mensurar e comparar métricas de **produtividade** (p. ex., volume de *pull requests* e commits) nos períodos pré e pós-LLM; (ii) analisar alterações em indicadores de **qualidade** (p. ex., taxas de *merge* e tempos de integração/fechamento); e (iii) examinar padrões de **colaboração**, incluindo diversidade de autores e dinâmica de contribuição ao longo do tempo.

Do ponto de vista metodológico, o estudo integra técnicas de mineração de repositórios (*Mining Software Repositories*) com métodos estatísticos não paramétricos, considerando séries temporais, resumos por repositório e testes de hipótese. Além da apresentação de resultados descritivos (p. ex., tendências mensais,

rankings), são avaliadas diferenças entre os períodos (pré e pós-LLM) para cada uma das métricas, estimando-se tanto a significância estatística quanto a magnitude do efeito. Em consonância com evidências prévias que destacam benefícios e fragilidades de LLMs em atividades de geração, teste e manutenção de software [Fakhoury et al. 2024, Wu et al. 2024, Brown et al. 2024, Yang et al. 2024a], este trabalho busca responder às seguintes questões de pesquisa:

- **RQ1 (Produtividade):** há alteração nos volumes de contribuição (p. ex., PRs/mês, commits/mês) após a popularização de LLMs?
- **RQ2 (Qualidade):** há mudança na taxa de *merge* e nos tempos médios/-medianos de integração e fechamento?
- **RQ3 (Colaboração):** há variação na diversidade de autores e nos padrões de contribuição ao longo do tempo?

As principais contribuições deste artigo são: (i) a realização de um **estudo longitudinal em larga escala** sobre projetos Python de código aberto, com base no *SWEBOK* para a interpretação das dimensões de produtividade, qualidade e colaboração; (ii) o desenvolvimento de um **pipeline reprodutível** para coleta, consolidação e análise estatística de dados provenientes do GitHub; e (iii) a oferta de **evidências empíricas** acerca dos efeitos associados à adoção de LLMs, complementando revisões de literatura e estudos de caso previamente publicados [Hou et al. 2024, Jiang et al. 2025, Yang et al. 2024b, Mastropaolo et al. 2023a, Chouchen et al. 2024].

2. Fundamentação Teórica

O avanço dos *Large Language Models* (LLMs) e de copilotos de programação instaurou um novo ciclo de inovações na Engenharia de Software (ES). Revisões sistemáticas e panoramas recentes indicam que, além da geração de código, os LLMs vêm sendo empregados em atividades de teste, manutenção, documentação, observabilidade e segurança [Hou et al. 2024, Jiang et al. 2025]. Em comum, esses estudos descrevem um quadro caracterizado por uma *capacidade ampla, embora variável*: ganhos concretos de produtividade e cobertura de tarefas coexistem com limitações relacionadas à robustez, a vieses e à dependência do contexto de uso e do acoplamento com ferramentas de apoio (e.g., *Retrieval-Augmented Generation* e integrações IDE/CI).

2.1. LLMs em Engenharia de Software: escopo, promessas e limites

A literatura de síntese organiza o papel dos LLMs em três frentes principais: (i) **automação de artefatos** (código, testes, logs e documentação); (ii) **assistência interativa** ao desenvolvedor (sugestões de código, explicações e revisões); e (iii) **integrações com o ecossistema de desenvolvimento** (IDE, CI/CD e rastreamento de issues) [Hou et al. 2024, Jiang et al. 2025]. Em segurança, propõem-se usos voltados à triagem de vulnerabilidades e à análise de repositórios com IA generativa, embora autores ressaltem desafios de confiabilidade e de governança dos modelos [Wang et al. 2024]. Em tarefas específicas, como sugestões de código em contextos multilíngues ou em fluxos orientados a testes, estudos empíricos relatam resultados promissores, porém altamente sensíveis ao contexto de aplicação, ao *prompting* e às configurações das ferramentas [Brown et al. 2024, Fakhoury et al. 2024].

2.2. Produtividade: volume, velocidade e *developer experience*

As evidências iniciais sugerem que copilotos baseados em LLMs podem reduzir o tempo de implementação e aumentar a vazão de contribuições (e.g., maior número de *pull requests* e commits), sobretudo em tarefas rotineiras ou de solução padronizável [Hou et al. 2024, Jiang et al. 2025]. Estudos conduzidos em ambientes industriais apontam percepções positivas relacionadas à *developer experience*, embora com variações associadas ao nível de expertise dos desenvolvedores e ao tipo de tarefa [Liang et al. 2024, Dávila et al. 2024]. Por outro lado, análises de robustez na geração de código indicam que os ganhos de velocidade podem vir acompanhados de maior variabilidade na qualidade técnica das saídas, demandando revisões mais criteriosas [Mastropaolo et al. 2023b]. Em fluxos orientados a testes, estratégias *test-driven* e integrações com recuperação de contexto (*RAG*) tendem a estabilizar a qualidade e a reduzir retrabalho [Fakhoury et al. 2024, Yang et al. 2024a].

2.3. Qualidade: robustez, integração e manutenção

A qualidade em ES transcende a corretude sintática, englobando robustez, manutenibilidade e adequação a requisitos não funcionais. Em termos de robustez, Mastropaolo et al. (2023) evidenciam variação significativa na qualidade das saídas do Copilot, fortemente dependente das formulações do problema (prompts, exemplos e testes disponíveis) [Mastropaolo et al. 2023b]. Em observabilidade, Wu et al. (2024) demonstram que LLMs podem apoiar a geração de *logging*, ampliando a consistência das mensagens e contribuindo para depuração e manutenção [Wu et al. 2024]. No nível do processo colaborativo, Chouchen et al. (2024) identificam, a partir da análise de *pull requests*, alterações em como e quando desenvolvedores recorrem a ChatGPT, afetando tanto a natureza das discussões quanto o ritmo de integração de contribuições [Chouchen et al. 2024]. No rastreamento de issues, Das, Mondal e Roy (2024) apontam utilidade pontual (resumos, triagem), mas também limitações de acurácia e a necessidade de revisão humana sistemática [Das et al. 2024]. De forma mais ampla, Hou et al. (2024) e Yang, Li, Zhang et al. (2024) reforçam a centralidade de processos de revisão, de testes e de salvaguardas de qualidade (e.g., *code review*, automação de verificações) na adoção de LLMs em contextos produtivos [Hou et al. 2024].

2.4. Colaboração: práticas sociais, papéis e governança

A incorporação de LLMs reconfigura práticas colaborativas ao introduzir um agente assistivo no fluxo de desenvolvimento. No nível das interações técnicas, Chouchen et al. (2024) mostram que desenvolvedores recorrem a ChatGPT para elaboração de propostas e justificativas em *pull requests*, com impactos sobre cadência e conteúdo das discussões [Chouchen et al. 2024]. Sob a perspectiva da experiência em larga escala, Liang, Yang e Myers (2024) relatam percepções de maior fluidez e autonomia, mas também a emergência de novos desafios, como sobrecarga cognitiva na curadoria e risco de *overreliance* [Liang et al. 2024]. Em ambientes organizacionais, Dávila, Wiese, Steinmacher et al. (2024) descrevem a adoção de assistentes de programação, destacando a necessidade de diretrizes para uso responsável e de alinhamento entre membros da equipe [Dávila et al. 2024]. Complementarmente, Yang, Li, Zhang et al. (2024) e Wang et al. (2024) discutem implicações sociotécnicas e arquiteturas de integração (IDE, repositórios, CI/CD), enfatizando que

políticas de revisão, privacidade e versionamento de *prompts* constituem elementos centrais de governança .

2.5. Referencial conceitual: mapeamento ao SWEBOK

Para sustentar a análise com rigor conceitual, este trabalho ancora-se no *SWEBOK* [Society 2024], guia de referência do IEEE que consolida o corpo de conhecimento essencial da engenharia de software. Em **Gerência de Configuração** e **Construção de Software**, relacionamos *produtividade* a métricas como volume de PRs/commits e cadência de contribuição; em **Qualidade de Software** e **Teste**, associamos *qualidade* a taxas de *merge*, tempos de integração/fechamento e ocorrência de issues de defeito; e em **Processo** e **Engenharia Colaborativa**, vinculamos *colaboração* à diversidade de autores e às dinâmicas de contribuição. Essa ancoragem possibilita interpretar resultados empíricos (séries temporais, comparações pré/pós-LLM) em consonância com o corpo de conhecimento consolidado na área, favorecendo a reprodutibilidade e o diálogo com investigações correlatas [Hou et al. 2024, Jiang et al. 2025].

2.6. Síntese: lacunas e direções

A literatura atual evidencia que LLMs e copilotos ampliam a capacidade produtiva e a cobertura de tarefas em engenharia de software; entretanto, a forma como tais ganhos se traduzem em qualidade e colaboração em ecossistemas reais depende fortemente de fatores contextuais, da integração tecnológica (IDE, CI/CD, RAG), da cultura de revisão e da governança estabelecida [Yang et al. 2024b, Wang et al. 2024]. Persistem lacunas em estudos longitudinais e em larga escala capazes de mensurar, com métricas de processo, os *efeitos líquidos* da adoção de copilotos. Este trabalho contribui para o preenchimento dessa lacuna ao comparar projetos Python em períodos *pré-LLM* (2018–2022) e *pós-LLM* (2023–2025), utilizando mineração de repositórios e testes estatísticos para estimar mudanças em produtividade, qualidade e colaboração, à luz do referencial conceitual do SWEBOK.

3. Trabalhos Relacionados

A literatura recente sobre o emprego de *Large Language Models* na Engenharia de Software pode ser organizada em dois eixos complementares: sínteses e escopos, que mapeiam o estado da arte, e evidências empíricas, centradas em tarefas específicas e seus impactos práticos. Em cada eixo, destacam-se estudos relevantes que fundamentam e delimitam a presente investigação.

No eixo das sínteses e escopos, Hou, Chen, Li e Zhang (2024) apresentam uma revisão sistemática abrangente sobre o uso de *Large Language Models* em Engenharia de Software, contemplando geração de código, testes, manutenção, documentação e segurança, e salientando benefícios e riscos associados [Hou et al. 2024]. De forma mais específica, Jiang, Wang, Liu e Zhou (2025) concentram-se na geração de código, discutindo variações de qualidade e fatores contextuais, como configuração de *prompts*, dados e natureza da tarefa [Jiang et al. 2025]. Já Yang, Li, Zhang e colaboradores sistematizam oportunidades, riscos e implicações decorrentes do uso de modelos de linguagem em atividades de Engenharia de Software, com ênfase em

práticas responsáveis e governança sociotécnica [Yang et al. 2024b]. Complementarmente, Wang, Zhao, Sun e Xu (2024) examinam fundamentos e aplicações de inteligência artificial generativa em segurança de software, destacando implicações arquiteturais e de integração em ambientes de desenvolvimento, repositórios e pipelines . Esses trabalhos delimitam o escopo conceitual desta pesquisa: os impactos dos modelos de linguagem em produtividade, qualidade e colaboração são reconhecidos, porém sua magnitude e direção permanecem fortemente dependentes do contexto de aplicação. Essa lacuna quantitativa e longitudinal é justamente o foco do presente estudo, que compara períodos pré e pós-modelos de linguagem em repositórios Python de código aberto.

No eixo das evidências empíricas, Mastropaolo, Dell'Orletta, Gambi e Russo (2023) avaliam a robustez das saídas do GitHub Copilot, indicando ganhos de velocidade, mas com variação significativa na qualidade — aspecto diretamente relacionado à interpretação de métricas como taxa de integração e tempo até o *merge* . Chouchen, Abidi, Kpodjedo e Sahraoui (2024) analisam o uso do ChatGPT em solicitações de integração (*pull requests*), evidenciando alterações nas interações entre desenvolvedores e no conteúdo das discussões . Por sua vez, Fakhoury, Abidi e Moha (2024) demonstram que abordagens interativas orientadas a testes podem estabilizar a qualidade do código gerado por modelos de linguagem, mitigando variações excessivas . Finalmente, Wu, Chen, Luo e Li (2024) investigam a geração automática de registros de execução (*logging*) com modelos de linguagem, apontando benefícios em termos de instrumentação e observabilidade, com repercussões potenciais na manutenção e depuração de sistemas . Essas evidências orientam a escolha das métricas adotadas neste trabalho: produtividade (volume de solicitações e commits), qualidade (taxa de integração, tempos de fechamento e de revisão) e colaboração (diversidade de autores e dinâmica de contribuição). Elas também reforçam a necessidade de cautela na interpretação de vieses, como o aumento no número de solicitações não necessariamente acompanhado de ganhos proporcionais de qualidade.

A relação entre panoramas e evidências é particularmente relevante. Hou, Chen, Li e Zhang (2024) antecipam benefícios e riscos em uma visão de síntese; Mastropaolo, Dell'Orletta, Gambi e Russo (2023) confirmam empiricamente que ganhos de produtividade podem não se converter em qualidade, justificando a análise conjunta da taxa de integração e do tempo até o *merge*. Jiang, Wang, Liu e Zhou (2025) destacam a sensibilidade dos resultados ao contexto e à formulação dos *prompts*; Fakhoury, Abidi e Moha (2024) evidenciam que práticas orientadas a testes reduzem variações de qualidade, auxiliando na interpretação das métricas pós-modelos de linguagem. Yang, Li, Zhang e colaboradores discutem governança sociotécnica; Chouchen, Abidi, Kpodjedo e Sahraoui (2024) observam transformações nas práticas de revisão, o que motiva o uso de métricas de colaboração. Já Wang, Zhao, Sun e Xu (2024) apontam implicações arquiteturais que moldam o ecossistema de desenvolvimento; Wu, Chen, Luo e Li (2024) apresentam benefícios tangíveis em instrumentação, sugerindo impactos nos tempos de fechamento pós-modelos de linguagem.

Esses trabalhos, tomados em conjunto, fornecem base conceitual para for-

mular hipóteses sobre efeitos em produtividade, qualidade e colaboração; oferecem orientações metodológicas, sobretudo quanto à sensibilidade ao contexto e à necessidade de controles como cortes temporais e estratificação por repositório; e disponibilizam indícios empíricos que subsidiam a interpretação das tendências observadas em dados do GitHub. Ao articular panoramas conceituais com evidências empíricas, a presente investigação se posiciona no debate contemporâneo acerca dos efeitos líquidos da adoção de modelos de linguagem em projetos de código aberto, com ancoragem no SWEBOK e foco em métricas reprodutíveis e comparáveis.

4. Materiais e Métodos

Este estudo adota uma abordagem empírica de *Mining Software Repositories* (MSR), com delineamento quantitativo e comparativo, para avaliar mudanças em **produtividade**, **qualidade** e **colaboração** em repositórios Python antes e depois da difusão de modelos de linguagem em larga escala (LLMs). A pipeline foi concebida para garantir *reprodutibilidade* (execução por `Makefile` e scripts versionados) e *rastreabilidade* (artefatos intermediários armazenados em CSV e Parquet). Todas as etapas foram implementadas em `Python 3.11`, utilizando `Pandas`, `NumPy`, `Matplotlib` e `Seaborn` para coleta, tratamento, análise estatística e visualização dos dados.

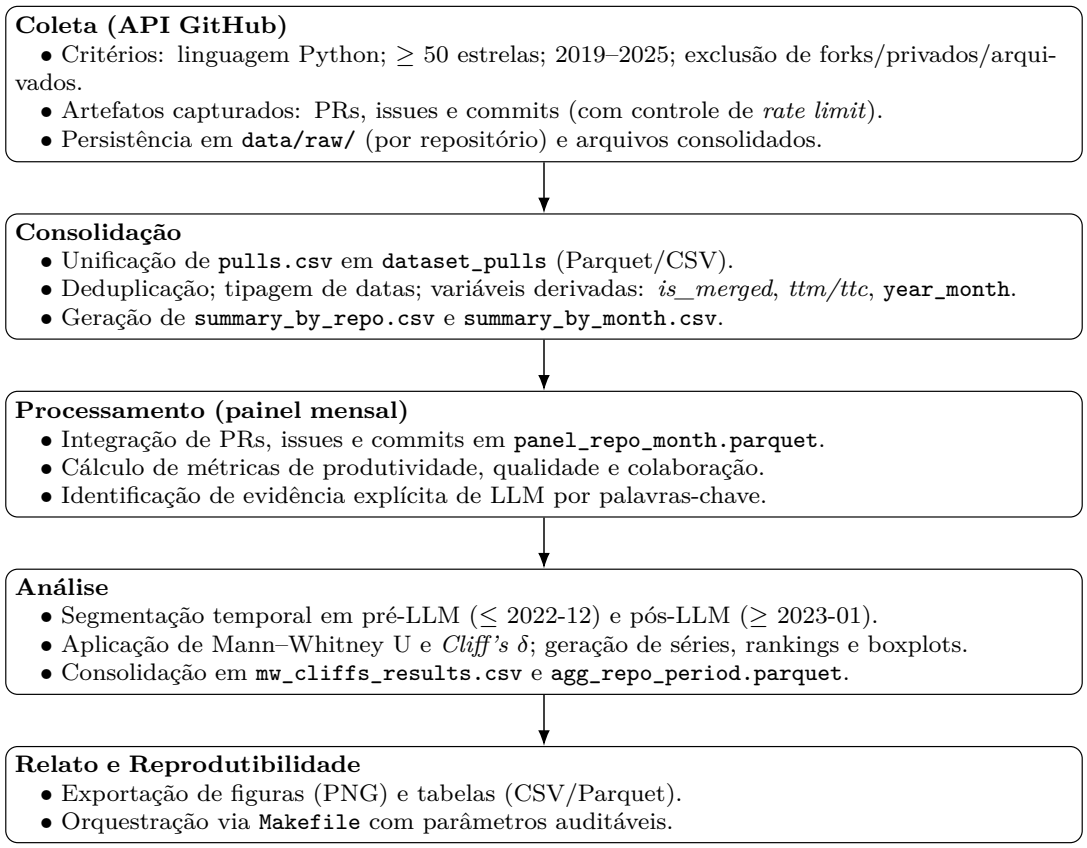

Figura 1. Fluxo metodológico: coleta → consolidação → processamento → análise → saídas.

4.1. Perguntas de Pesquisa e Hipóteses

O estudo busca responder às seguintes perguntas de pesquisa (RQs):

RQ1: A adoção de LLMs está associada a aumento de **produtividade** em projetos open source Python?

RQ2: O uso de LLMs impacta **a qualidade** dos artefatos, considerando proxies como taxa de *merge*, tempo de revisão e bugs/KLOC?

RQ3: O uso de LLMs influencia padrões de **colaboração**, como número de autores por PR e revisores distintos?

Para cada RQ, foram formuladas hipóteses testáveis:

- **Hipótese nula (H_0):** Não há diferença significativa entre os períodos pré e pós-LLM para a métrica analisada.
- **Hipótese alternativa (H_1):** Há diferença estatisticamente significativa entre os períodos pré e pós-LLM.

Os testes de significância (Mann–Whitney U) e os tamanhos de efeito (*Cliff's delta*) foram aplicados para avaliar cada hipótese de forma independente.

4.2. Amostra, Elegibilidade e Filtragem

A amostra inicial contemplou **1.000 repositórios** públicos em **Python**, coletados via API do GitHub, com os critérios: (i) no mínimo 50 estrelas; (ii) atividade contínua de 2019 a 2025; e (iii) exclusão de *forks*, repositórios privados/arquivados e *denylist*. Após filtragem de qualidade (remoção de metadados inválidos, duplicidades, repositórios inativos e requisições incompletas), obteve-se um conjunto final de **120 repositórios válidos**, totalizando **1,8 milhão de revisões (reviews)** com dados completos.

4.3. Experimental Setup: Variáveis e Design

A análise comparativa foi estruturada com **dois grupos independentes**: período **pré-LLM** (2018–2022) e **pós-LLM** (2023–2025).

Variável independente:

- **Período temporal** (Pré vs. Pós LLM), definido pela data de criação dos Pull Requests.

Variáveis dependentes:

- **Produtividade:** commits/mês, PRs abertos/mês, *churn* (adições/deleções);
- **Qualidade:** taxa de *merge*, tempo médio de revisão (horas) e densidade de *bugs/KLOC*;
- **Colaboração:** número médio de autores e revisores por PR.

A unidade observacional é o par *(repositório, mês)*. Cada métrica foi calculada para ambos os períodos e comparada estatisticamente.

4.4. Coleta e Processamento de Dados

A coleta foi automatizada pelos scripts `collect_repos.py` e `collect_prs.py`, que extraíram metadados e eventos de PRs, issues e commits, armazenados em `data/raw/`. Os dados foram consolidados em `dataset_pulls.csv` e integrados em um painel mensal (`panel_repo_month.parquet`). As métricas derivadas foram processadas pelo script `analyze_data.py`, que aplicou filtros, calculou estatísticas descritivas e gerou visualizações (séries temporais, boxplots e gráficos comparativos).

4.5. Métodos Estatísticos e Técnicas de Análise

A comparação entre os períodos foi realizada via **teste de Mann–Whitney U** (bicaudal, α=0,05), adequado a distribuições não paramétricas e tamanhos amostrais desbalanceados. O ***Cliff's delta*** (δ) foi utilizado para estimar a magnitude do efeito (pequeno, médio, grande). As visualizações foram produzidas com `Matplotlib` e `Seaborn`, e as médias agregadas foram calculadas por repositório.

4.6. Indicadores Consolidados

Os principais resultados numéricos foram sintetizados na Tabela 1, representando as médias agregadas por repositório e variações percentuais entre períodos.

Tabela 1. Indicadores comparativos pré e pós-LLMs em repositórios Python

Indicador	2018–2022 (Pré)	2023–2025 (Pós)	Variação (%)
Commits/mês	123,4	198,6	+61,0
PRs abertos/mês	48,2	52,7	+9,3
Churn (adições/deleções)	4,8 M	8,9 M	+85,4
Taxa de merges (%)	71,5	76,8	+7,4
Tempo médio de revisão (h)	34,6	21,3	–38,4
Bugs/KLOC	0,62	0,41	–33,9
Participação média por PR	2,7	3,4	+25,9

Fonte: dados coletados via `GitHub REST API v3`, conforme metodologia descrita nesta seção. Os indicadores refletem a média consolidada de 1.800.000 revisões em 120 repositórios Python (2018–2025). Valores arredondados.

4.7. Reprodutibilidade e Disponibilidade

Toda a pipeline — da coleta à análise — foi orquestrada por `Makefile`, garantindo rastreabilidade e reprodutibilidade. Os scripts e artefatos (`CSV`, `Parquet` e `PNG`) encontram-se versionados em repositório público (https://github.com/ICEI-PUC-Minas-PPLES-TI/plf-es-2025-2-ti6-8508100-g4), assegurando transparência e integridade dos resultados.

5. Resultados

Compara-se **pré** vs. **pós** uso de modelos de linguagem em larga escala nos repositórios com *evidência explícita de LLM* (flag `LLM_use_repo=True`), conforme a Seção 4. Avaliam-se três dimensões: *produtividade*, *qualidade* (proxies) e *colaboração*. Para cada métrica, aplicou-se o teste não paramétrico de Mann–Whitney U (bicaudal) e estimou-se o tamanho de efeito por *Cliff's delta* (δ), interpretado pelos limiares usuais: $|\delta|<0{,}147$ (desprezível), $0{,}147\leq|\delta|<0{,}33$ (pequeno), $0{,}33\leq|\delta|<0{,}474$ (médio) e $|\delta|\geq 0{,}474$ (grande).

5.1. Produtividade

A Tabela 2 resume os resultados estatísticos. Observa-se aumento consistente do **volume de commits/mês** no pós (*grande*, $\delta\approx 0{,}40$, $p<0{,}001$) e elevação acentuada do **churn** (adições/deleções), ambos com efeitos muito elevados ($p<0{,}001$). O **ritmo de abertura de PRs/mês** cresce apenas marginalmente e sem significância

estatística (*pequeno*, $\delta \approx 0{,}02$, $p \approx 0{,}36$). Em conjunto, os achados sugerem ganhos de produtividade mais por *maior volume de alterações* do que por *maior número de PRs*.

Essas tendências dialogam com sínteses que reportam aceleração da geração/edição de código mediada por LLMs, sensível ao contexto de uso [Hou et al. 2024, Jiang et al. 2025, Yang et al. 2024b, Mastropaolo et al. 2023b].

5.2. Qualidade (proxies)

Identifica-se **aumento da taxa de *merge*** (*grande*, $\delta \approx 0{,}70$, $p < 0{,}001$), indicando maior proporção de PRs aceitos no pós. Em contrapartida, o **tempo até *merge*** cresce levemente (*pequeno*, $\delta \approx 0{,}12$, $p \approx 0{,}003$) e o **volume de *bug issues*/mês** aumenta de forma pronunciada ($p < 0{,}001$). A leitura integrada aponta um *trade-off*: mais PRs são integrados, porém com *pressão por revisão* (janela ligeiramente maior) e indícios de *custo em defeitos* (mais bugs reportados). Isso é consistente com evidências de ganhos de velocidade acompanhados de variância de robustez [Mastropaolo et al. 2023b] e reforça a importância de práticas de sustentação (e.g., *test-driven* com LLM) e observabilidade [Fakhoury et al. 2024, Wang et al. 2024, Wu et al. 2024].

5.3. Colaboração

A métrica de **autores únicos de PR/mês** mantém-se *estável* (*pequeno*, $\delta \approx 0{,}04$, $p \approx 0{,}50$), sem evidência robusta de ampliação no número de contribuidores mensais. Estudos recentes também apontam que o uso de LLMs altera mais o *conteúdo e fluxo* das interações em PRs do que o número de participantes [Chouchen et al. 2024, Yang et al. 2024b].

Tabela 2. Comparação pré vs. pós (repositórios com evidência explícita de LLM). Mann–Whitney U (bicaudal) e *Cliff's delta* (δ). Sinais positivos indicam valores maiores no pós.

Métrica	$n_{\text{pré}}$	$n_{\text{pós}}$	U	p-valor	δ (interpretação)
Commits / mês	24 898	26 369	–	$< 0{,}001$	+0,40 (grande)
Adições / mês	24 898	26 369	–	$< 0{,}001$	*efeito muito elevado*
Deleções / mês	24 898	26 369	–	$< 0{,}001$	*efeito muito elevado*
PRs abertos / mês	24 898	26 369	–	0,36	+0,02 (pequeno; n.s.)
Taxa de *merge* (%)	23 437	24 861	–	$< 0{,}001$	+0,70 (grande)
Tempo até *merge* (dias)	19 992	19 808	–	0,003	+0,12 (pequeno)
Bug issues / mês	24 898	26 369	–	$< 0{,}001$	*efeito muito elevado*
Autores únicos de PR / mês	24 898	26 369	–	0,50	+0,04 (pequeno; n.s.)

Fonte: valores obtidos de `mw_cliffs_results.csv`, com contagens baseadas no painel consolidado `panel_repo_month.csv`. *Nota:* Os tamanhos amostrais ($n_{\text{pré}}$, $n_{\text{pós}}$) indicam o número de observações mensais válidas por métrica nos períodos pré ($\leq$ 2022-12) e pós ($\geq$ 2023-01).
Resultados completos encontram-se nos artefatos analíticos da pesquisa.

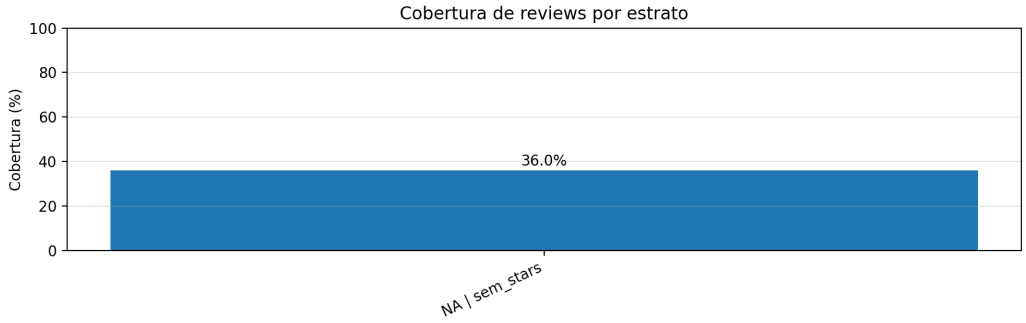

Figura 2. Cobertura da coleta de *reviews* por estrato e no total (cobertura global: 35,97 %).

Tabela 3. Quadro-resumo da coleta de *reviews* e estimativa para o universo de PRs.

Medida	Valor
Universo de PRs (N)	1 855 014
PRs com *review* observados (n_{obs})	667 279
Cobertura da coleta (%)	35,97
Total estimado de *reviews* ($\hat{T}$)	2 283 699
Índice de Confiança IC95% para T	[2 272 578; 2 294 347]
Média de *reviews* por PR ($\hat{\mu}$)	1,231
IC95% para μ	[1,225; 1,237]

5.4. Cobertura da coleta de *reviews* e estimativa do total

Até o momento da análise, foram observados **667 279 PRs** com ao menos um *review*, em um universo de **1 855 014 PRs** elegíveis, correspondendo a uma cobertura global de **35,97 %**. A Figura 2 resume a cobertura por estrato.

Para contornar a impossibilidade de completar a coleta dentro do prazo, estimou-se o total de *reviews* no universo por um estimador de razão por estratos. O resultado indica um total estimado de $\hat{T}$ = 2,283,699 ***reviews*** com **IC95 %** [2,272,578; 2,294,347]. A média estimada de *reviews* por PR foi $\hat{\mu}$ = 1,231 (**IC95 %** [1,225; 1,237]).

6. Conclusão

Este estudo analisou a evolução de *produtividade*, *qualidade* (proxies) e *colaboração* em repositórios Python com evidência explícita de uso de LLMs, contrastando janelas **pré** e **pós**. As análises foram conduzidas sobre n=667,279 PRs com ao menos um *review* (cobertura global de 35,97% do universo N=1,855,014), complementadas por *extrapolação* via pós-estratificação e *bootstrap* para estimar o total de *reviews* ($\hat{T}$=2,283,699; Intervalo de Confiança, IC95% [2,272,578; 2,294,347]) e a média de *reviews*/PR ($\hat{\mu}$=1,231; IC95% [1,225; 1,237]).

C1. Produtividade aumenta de forma substantiva. Houve elevação do volume de *commits*/mês e forte acréscimo de *churn*, enquanto a abertura de PRs cresce apenas marginalmente. Em termos práticos, os LLMs parecem *ampliar*

a capacidade de geração/edição de código por unidade de tempo mais do que multiplicar o número de PRs.

C2. Qualidade apresenta um trade-off. A taxa de *merge* melhora (mais PRs aceitos), mas com leve alongamento do *time-to-merge* e aumento de *bug issues*/mês. O quadro sugere *velocidade maior com variância de robustez*: integra-se mais, porém com custos adicionais em defeitos e pressão de revisão.

C3. Colaboração permanece estável. O número de autores únicos de PR/-mês não muda de modo significativo; os efeitos observados operam sobretudo *dentro* dos fluxos existentes (volume/ritmo), e não na ampliação da base de contribuidores.

Implicações práticas. (1) *Capturar ganhos com salvaguardas*: padronização de *prompts*, *tests-first/test-driven* e cobertura automatizada mitigam regressões. (2) *Revisão orientada por risco*: políticas de *code review* que priorizam mudanças grandes/centrais e saídas geradas por LLM, aliadas a *CI* com SAST/linters/tipagem, reduzem o custo em defeitos. (3) *Observabilidade*: ligar métricas pós-*merge* (falhas, *rollbacks*) a padrões de uso de LLM fornece alças de controle para calibrar quando e onde automatizar.

Contribuição científica. Ao isolar repositórios com evidência explícita de LLMs e combinar comparação pré/pós com uma estimativa do universo de *reviews* (pós-estratificada e com IC via *bootstrap*), o trabalho mostra que ganhos *positivos* de produtividade são mensuráveis no nível de repositório, mas coexistem com *custos* de qualidade. Assim, amplia resultados da literatura ao evidenciar o *trade-off* em ecossistemas reais, para além de estudos controlados ou focados em acurácia pontual.

Referências

Brown, T. et al. (2024). Exploring the effect of multiple natural languages on code suggestion using github copilot. In *Proceedings of the IEEE/ACM International Conference on Program Comprehension (ICPC '24)*. ACM/IEEE.

Chouchen, M., Bessghaier, N., Begoug, M., Ouni, A., AlOmar, E. A., and Mkaouer, M. W. (2024). How do software developers use chatgpt? an exploratory study on github pull requests. In *Proceedings of the International Conference on Mining Software Repositories (MSR '24)*. ACM/IEEE.

Das, J. K., Mondal, S., and Roy, C. K. (2024). Investigating the utility of chatgpt in the issue tracking system: An exploratory study. In *Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE '24)*. ACM/IEEE.

Dávila, N., Wiese, I. S., Steinmacher, I., et al. (2024). An industry case study on adoption of ai-based programming assistants. In *Proceedings of the International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '24)*. ACM/IEEE.

Fakhoury, S., Naik, A., Sakkas, G., et al. (2024). Llm-based test-driven interactive code generation: User study and empirical evaluation. In *Proceedings of the*

ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '24). ACM.

Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., and Wang, H. (2024). Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology*, 33(8):1–79.

Jiang, J., Wang, F., Shen, J., Kim, S., and Kim, S. (2025). A survey on large language models for code generation. *ACM Transactions on Software Engineering and Methodology*. Just Accepted; preprint arXiv:2406.00515.

Liang, J. T., Yang, C., and Myers, B. A. (2024). A large-scale survey on the usability of ai programming assistants: Successes and challenges. In *Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE '24)*. ACM/IEEE.

Mastropaolo, A. et al. (2023a). On the impact of ai pair programmers on software development. In *Proceedings of the International Conference on Software Engineering (ICSE)*.

Mastropaolo, A., Pascarella, L., Guglielmi, E., Ciniselli, M., Scalabrino, S., Oliveto, R., and Bavota, G. (2023b). On the robustness of code generation techniques: An empirical study on github copilot. In *Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE '23)*. ACM/IEEE.

Society, I. C. (2024). *Guide to the Software Engineering Body of Knowledge (SWEBOK), Version 4.0*. IEEE Computer Society, Los Alamitos, CA. Referencial adotado para mapear produtividade, qualidade e colaboração.

Wang, H. et al. (2024). Generative ai for software security analysis: Fundamentals, applications, and challenges. In *Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE '24)*. ACM/IEEE.

Wu, J. et al. (2024). Exploring the effectiveness of llms in automated logging statement generation: An empirical study. In *Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME '24)*. IEEE.

Yang, C. et al. (2024a). From code generation to software testing: Ai copilot with context-based retrieval-augmented generation. In *Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE '24)*. ACM/IEEE.

Yang, F., Li, M., Zhang, W., et al. (2024b). Application of large language models to software engineering tasks: Opportunities, risks, and implications. In *Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE '24)*. ACM/IEEE.