Entendendo o Comportamento de Pull Requests em Python: Um estudo de cenário de tempo de vida e revisores

Álvaro Lopes Rios[1]

[1]Centro de Ciências Exatas e Tecnológicas - Universidade Federal do Acre (UFAC)
Rio Branco – AC – Brasil

alvaro.rios@sou.ufac.br

Resumo. *O estudo de referência de [Soares 2017] analisou fatores de pull requests por regras de associação, e um dos vários cenários investigados foram os cenários de lifetime e revisor. Este trabalho replica essa abordagem no recorte de projetos open-source em Python, preservando a mesma taxonomia de rótulos e comparando o lift por célula (valor do antecedente, consequente) entre a nossa base ("Bases Python") e a referência.*

1. Introdução

O modelo *pull-based* impulsiona a colaboração em software, porém impõe desafios relevantes no tratamento de *pull requests* (PRs), especialmente quanto ao tempo até a decisão (*lifetime*) e à dinâmica de revisão. [Soares 2017] investigou essas relações por meio de regras de associação, cobrindo múltiplos cenários para entender o comportamento de PRs e destacando, entre os atributos analisados, *lifetime* e *revisor*.

Neste trabalho, a abordagem da referência foi replicada no recorte de projetos *open-source* da linguagem Python, mantendo-se a mesma taxonomia de rótulos para permitir comparação direta do *lift*. Os projetos Python escolhidos foram os 5 maiores com relação aos numeros de Pull Requests: commcare-hq (6,981 PRs), rosdistro (5,739), django (3,730), ipython (3,231) e pandas (2,232), totalizando 21,913 pull requests. Para a análise foram adotados dois cenários complementares:

- **Cenário *Lifetime*** — o consequente das regras é a *classe de lifetime* (*very short*, *short*, *medium*, *lengthy*). Como antecedentes, foram utilizados atributos estruturais do PR e fatores de processo/perfil já harmonizados à referência:
  - `total_lines_D` (discretizado em {*1 line*, *some lines*, *many lines*});
  - `changedFiles_D` ({*1 file*, *some files*, *many files*});
  - `commitsPull_D` ({*1 commit*, *some commits*, *many commits*});
  - `typeDeveloper` ({*external*, *core*});
  - `requester_experience_project` ({*no contributions*, *some contributions*, *many contributions*});
  - `coreTeamFollowsRequester` ({*True*, *False*});
  - `first_Pull` ({*True*, *False*});
  - `followers_boolean` ({*has followers*, *no followers*}).

  O atributo *status* (*accepted*/*rejected*) foi considerado apenas como contexto, não como objetivo do estudo.
- **Cenário *Reviewer* (Close By)** — o consequente das regras é *Close By* (*quem revisa o PR*). Como principais antecedentes, foram testados atributos ligados a tamanho/complexidade da mudança e a vínculo com o *core team*:

- commitsPull_D ({*1 commit*, *some commits*, *many commits*});
- changedFiles_D ({*1 file*, *some files*, *many files*});
- total_lines_D ({*1 line*, *some lines*, *many lines*});
- typeDeveloper ({*external*, *core*});
- coreTeamFollowsRequester ({*True*, *False*}).

O objetivo é verificar em que medida os padrões observados na referência se mantêm no ecossistema Python e onde mudam de intensidade, comparando o *lift* obtido na nossa base ("Bases Python") com o *lift* reportado na referência, por célula (par *valor do antecedente*, *consequente*). As contribuições incluem: (i) um *pipeline* de harmonização de rótulos alinhado à referência; (ii) um conjunto de visualizações comparativas (Bases Python $\times$ Referência) por valor de antecedente, com *lifetime* ou *Close By* como consequente; (iii) uma análise das diferenças de intensidade dos padrões no contexto de projetos Python; e (iv) implicações práticas para triagem, dimensionamento de PRs e alocação de revisores.

2. Metodologia

Esta seção descreve os dois cenários analisados (Lifetime e Reviewer/Close By), o conjunto de dados, o pré-processamento aplicado (incluindo filtros do WEKA), a extração de regras e métricas no formato $X \Rightarrow Y$, bem como a estratégia de comparação e visualização adotada.

2.1. Cenários e Conjunto de Dados

Foram considerados *pull requests* (PRs) de projetos *open-source* da linguagem Python. Manteve-se a mesma taxonomia de rótulos do estudo de referência para permitir comparação direta do *lift*.

O consequente das regras é a *classe de lifetime* em {*very short*, *short*, *medium*, *lengthy*}. Como antecedentes, foram utilizados os seguintes atributos:

- total_lines_D $\rightarrow$ {*1 line*, *some lines*, *many lines*};
- changedFiles_D $\rightarrow$ {*1 file*, *some files*, *many files*};
- commitsPull_D $\rightarrow$ {*1 commit*, *some commits*, *many commits*};
- typeDeveloper $\rightarrow$ {*external*, *core*};
- requester_experience_project $\rightarrow$ {*no contributions*, *some contributions*, *many contributions*};
- coreTeamFollowsRequester $\rightarrow$ {*True*, *False*};
- first_Pull $\rightarrow$ {*True*, *False*};
- followers_boolean $\rightarrow$ {*has followers*, *no followers*}.

O atributo *status* (*accepted/rejected*) tambem foi utilizado, porém sendo antecedente para verificar o resultado como consequente o tempo de vida dos PRs.

O consequente das regras é *Close By* (*quem revisa o PR*). Como principais antecedentes, foram testados:

- commitsPull_D $\rightarrow$ {*1 commit*, *some commits*, *many commits*};
- changedFiles_D $\rightarrow$ {*1 file*, *some files*, *many files*};
- total_lines_D $\rightarrow$ {*1 line*, *some lines*, *many lines*};
- typeDeveloper $\rightarrow$ {*external*, *core*};
- coreTeamFollowsRequester $\rightarrow$ {*True*, *False*}.

2.2. Pré-processamento

O pré-processamento teve dois objetivos: (i) alinhar a nomenclatura dos rótulos com a referência e (ii) preparar os dados conforme o cenário (base unificada no *Lifetime* e segmentada por projeto no *Reviewer/Close By*).

Para garantir comparabilidade, empregou-se o filtro *unsupervised* `MergeManyValues` do WEKA para consolidar valores nominais em categorias canônicas:

- ***lifetime*** $\rightarrow$ {*very short*, *short*, *medium*, *lengthy*};
- ***status*** $\rightarrow$ {*accepted*, *rejected*}.

Essa etapa assegura que regras e métricas (em particular, o *lift*) sejam calculadas sobre a mesma taxonomia da referência.

Após harmonização, os PRs de todos os projetos Python foram analisados em conjunto. As variáveis de contagem foram discretizadas em {*1*, *some*, *many*} (linhas, arquivos, *commits*); os categóricos de processo/perfil foram padronizados conforme os domínios listados na subseção de cenários. O *status* foi usado apenas como contexto.

Diferentemente do cenário *Lifetime*, as regras foram extraídas separadamente para cada projeto. Utilizou-se o filtro WEKA `RemoveWithValues` sobre o identificador do repositório, retendo-se um projeto por vez para mineração. A segmentação permite observar associações específicas de *quem revisa o PR* (*Close By*) em cada repositório.

Antes da mineração, foram removidos nulos e inconsistências, e verificados os domínios após harmonização. Em células repetidas (mesma combinação *antecedente=valor, consequente*), as medidas foram agregadas por média, reduzindo ruído e facilitando a comparação direta de *lift* com a referência.

2.3. Extração de Regras e Métricas

As regras de associação são expressas no formato $X \Rightarrow Y$, em que X é o *antecedente* (uma ou mais condições sobre atributos do PR) e Y é o *consequente* (o desfecho/classe de interesse). Neste estudo, adotou-se:

- **Cenário *Lifetime***: Y é uma classe de *lifetime* {*very short*, *short*, *medium*, *lengthy*}. Exemplos de X: `total_lines_D` $=$ *many lines*, `changedFiles_D` $=$ *some files*, `commitsPull_D` $=$ *1 commit*, `typeDeveloper` $=$ *core*.
- **Cenário *Reviewer* (Close By)**: Y é *Close By (quem revisa o PR)*[1]. Exemplos de X: `commitsPull_D` $=$ *many commits*, `total_lines_D` $=$ *1 line*, `typeDeveloper` $=$ *external*, `coreTeamFollowsRequester` $=$ *True*.

Para extração do maior número possiveis de regras, o suporte e confiança foram diminuidos em 0,1% para a maior obtenção possíveis de regras para uma análise mais abrangente.

2.4. Estratégia de comparação e visualização

Após a extração, os resultados foram organizados em planilhas separadas por cenário (Lifetime e Reviewer/Close By) com as colunas: Antecedente, Consequente, Support, Confidence, Lift, Leverage, Conviction, Lift_{ref} e Diff = $\text{Lift}_{Python} - \text{Lift}_{ref}$.

[1] Categorias conforme disponibilidade na base.

Foram aplicadas três verificações simples:

1. **Variação percentual**: classificação de $|Diff|/Lift_{ref}$ nas faixas $< 25\%$, 25–50%, 50–75%, 75–100% e $\geq 100\%$.
2. **Mudança de dependência**: indica "sim" quando há cruzamento do limiar $1{,}0$ (isto é, $(Lift_{Python} > 1)$ XOR $(Lift_{ref} > 1)$).
3. **Mudança na regra**: para um mesmo antecedente, marca "sim" quando o consequente de maior lift difere entre Python e referência.

A Figura 1 representa de tela que mostra o inicio da planilha elaborada.

Antecendente	Consequente	Suporte	Confianca	Lift	Leverage	Conviction	Daricelio	diff	Mudou a dep	Check 1	Check 2
life_time=very short	status=accepeted	0,6	0,89	1,13	0,07	1,9	1,11	-0,02	Não	Variação abaixo de 25%	Apenas Variação
life_time=short	status=accepeted	0,1	0,73	0,92	-0,01	0,77	1,04	0,12	Sim	Variação abaixo de 25%	Apenas Variação
life_time=medium	status=accepeted	0,04	0,66	0,83	-0,01	0,61	0,96	0,13	Não	Variação abaixo de 25%	Apenas Variação
life_time=lengthy	status=rejected	0,08	0,57	2,74	0,05	1,85	2,14	-0,6	Não	Variação entre 50% a 75%	Apenas Variação
life_time=lengthy	status=accepeted	0,06	0,43	0,54	-0,05	0,36	0,66	0,12	Não	Variação abaixo de 25%	Apenas Variação
life_time=medium	status=rejected	0,02	0,34	1,64	0,01	1,2	1,13	-0,51	Não	Variação entre 50% a 75%	Apenas Variação
life_time=short	status=rejected	0,04	0,27	1,3	0,01	1,09	0,87	-0,43	Sim	Variação entre 25% a 50%	Mudança na Regra
life_time=very short	status=rejected	0,07	0,11	0,53	-0,07	0,89	0,65	0,12	Não	Variação abaixo de 25%	Apenas Variação
commitsPull_D=1 commit	life_time=very short	0,55	0,76	1,14	0,07	1,4	1,14	0	Não	Variação abaixo de 25%	Apenas Variação
commitsPull_D=some commits	life_time=very short	0,1	0,5	0,75	-0,03	0,67	0,75	0	Não	Variação abaixo de 25%	Apenas Variação
commitsPull_D=many commits	life_time=lengthy	0,03	0,39	2,84	0,02	1,4	1,92	-0,92	Não	Variação de 75% a 100%	Apenas Variação
commitsPull_D=many commits	life_time=very short	0,02	0,25	0,37	-0,03	0,44	0,51	0,14	Não	Variação abaixo de 25%	Apenas Variação
commitsPull_D=many commits	life_time=short	0,02	0,21	1,62	0,01	1,1	1,14	-0,48	Não	Variação entre 25% a 50%	Apenas Variação
commitsPull_D=some commits	life_time=short	0,04	0,21	1,6	0,02	1,1	1,24	-0,36	Não	Variação entre 25% a 50%	Apenas Variação
commitsPull_D=some commits	life_time=lengthy	0,04	0,18	1,36	0,01	1,06	1,28	-0,08	Não	Variação abaixo de 25%	Apenas Variação
commitsPull_D=many commits	life_time=medium	0,01	0,16	2,41	0,01	1,11	1,54	-0,87	Não	Variação de 75% a 100%	Apenas Variação
commitsPull_D=some commits	life_time=medium	0,02	0,1	1,59	0,01	1,04	1,29	-0,3	Não	Variação entre 25% a 50%	Apenas Variação
commitsPull_D=1 commit	life_time=short	0,07	0,1	0,76	-0,02	0,97	0,91	0,15	Não	Variação abaixo de 25%	Apenas Variação
commitsPull_D=1 commit	life_time=lengthy	0,07	0,09	0,69	-0,03	0,95	0,8	0,11	Não	Variação abaixo de 25%	Apenas Variação
commitsPull_D=1 commit	life_time=medium	0,03	0,04	0,68	-0,02	0,98	0,84	0,16	Não	Variação abaixo de 25%	Apenas Variação
changedFiles_D=1 file	life_time=very short	0,47	0,83	1,25	0,09	1,98	1,21	-0,04	Não	Variação abaixo de 25%	Apenas Variação
changedFiles_D=some files	life_time=very short	0,15	0,51	0,77	-0,04	0,68	0,87	0,1	Não	Variação abaixo de 25%	Apenas Variação
changedFiles_D=many files	life_time=lengthy	0,05	0,34	2,49	0,03	1,3	1,54	-0,95	Não	Variação de 75% a 100%	Apenas Variação
changedFiles_D=many files	life_time=very short	0,05	0,34	0,5	-0,05	0,5	0,68	0,18	Não	Variação abaixo de 25%	Apenas Variação
changedFiles_D=some files	life_time=lengthy	0,06	0,2	1,46	0,02	1,08	1,19	-0,27	Não	Variação entre 25% a 50%	Apenas Variação
changedFiles_D=many files	life_time=short	0,03	0,2	1,5	0,01	1,08	1,17	-0,33	Não	Variação entre 25% a 50%	Apenas Variação
changedFiles_D=some files	life_time=short	0,05	0,19	1,45	0,02	1,07	1,1	-0,35	Não	Variação entre 25% a 50%	Apenas Variação
changedFiles_D=many files	life_time=medium	0,02	0,13	2,01	0,01	1,07	1,36	-0,65	Não	Variação entre 50% a 75%	Apenas Variação

Figura 1. Tabela para o cenário de Lifetime

Para cada cenário de LifeTime foi gerado um grafico representado como exemplo na Figura 2, que compara os resultados de referencia obtidos por [Soares 2017] e os obtidos nas bases de projeto em Python.

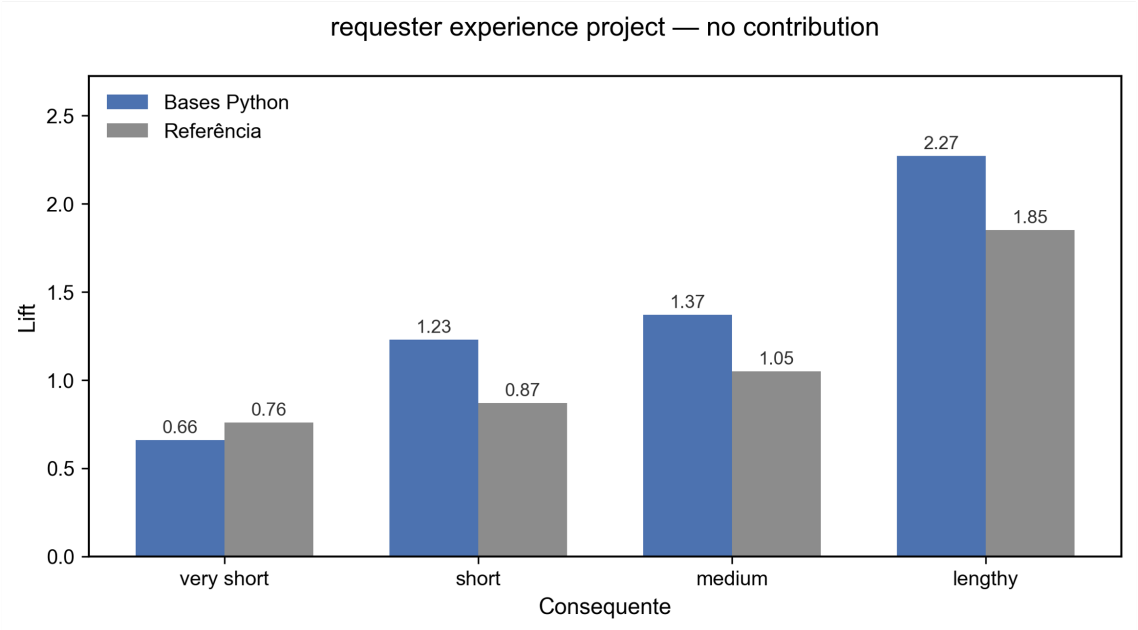

Figura 2. Exemplo de grafico gerado

Para o cenário Reviewer, os resultados foram apresentados em tabelas por projeto. Além de Confianca e Lift da regra $X \Rightarrow Y$ (onde Y é Close By), foi calculada também a confiança inversa $Conf(Y \Rightarrow X)$ para checar especialização do revisor (por exemplo, se um revisor costuma atuar em um padrão específico X). As tabelas listam: regra, Suporte, $Conf(X \Rightarrow Y)$, $Conf(Y \Rightarrow X)$ e $Lift(X \Rightarrow Y)$. A Figura 3 representa um exemplo de resultado da tabela de revisor, da qual compara uma regra para cada projeto separadamente, porém centralizando os resultados.

Base	Antecendente	Consequente	Suporte	Confianca x -> y	Confianca y -> x	Lift
Commcare	changedFiles_D=1 file	closedBy=yedi	1,79%	3,72%	66,14%	1,38
Commcare	changedFiles_D=1 file	closedBy=dmyung	1,26%	2,62%	55,00%	1,14
Django	changedFiles_D=1 file	closedBy=ubernostrum	0,16%	0,39%	85,71%	2,08
Django	changedFiles_D=1 file	closedBy=kmtracey	0,13%	0,33%	71,43%	1,73
Ipython	changedFiles_D=1 file	closedBy=epatters	0,25%	0,51%	88,89%	1,84
Ipython	changedFiles_D=1 file	closedBy=bfroehle	1,30%	2,70%	66,67%	1,38
Pandas	changedFiles_D=1 file	closedBy=takluyver	0,09%	0,32%	100,00%	3,54
Pandas	changedFiles_D=1 file	closedBy=adamklein	0,40%	1,43%	69,23%	2,45
Rosdistro	changedFiles_D=1 file	closedBy=130s	0,12%	0,13%	100,00%	1,08
Rosdistro	changedFiles_D=1 file	closedBy=dirk-thomas	27,67%	29,88%	96,65%	1,04

Figura 3. Tabela Revisor

3. Resultados

3.1. Lifetime

No cenario de LifeTime então status (Figura 4), percebe-se que tanto o estudo de referencia de Soares, quanto nos projetos em Python seguem o mesmo padrão de tendencia, exceto na regra lifetime=short → status = rejected, que mostrou uma alteração do lift de 0,87 da referencia, para 1,38 nas bases de Python, mostrando então que nos casos das bases de Python o tempo de vida do PRs ser curto aumenta as chances de ser rejeitado, diferente do resultado de Soares.

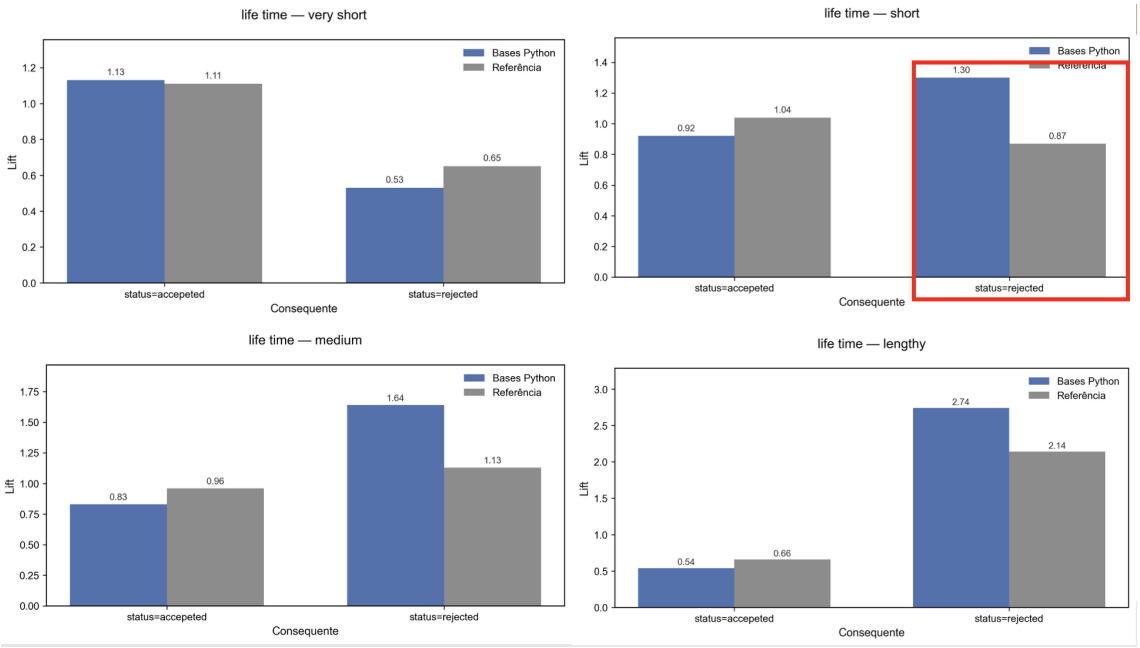

Figura 4. LifeTime → Status

No cenário changedFiles$_D \rightarrow$ lifetime, referência e bases Python seguem o mesmo padrão geral: quanto mais arquivos alterados, maior o lifetime. Com 1 file, há associação positiva para para very short (1,25 vs 1,21) e queda para prazos maiores (short 0,64 vs 0,87; medium 0,48 vs 0,78; lengthy 0,38 vs 0,67). Com some files, o efeito se desloca para prazos mais longos: short 1,45 vs 1,10; medium 1,51 vs 1,13; lengthy 1,46 vs 1,19 (very short permanece ¡1: 0,77 vs 0,87). Com many files, o crescimento é mais intenso: medium 2,01 vs 1,36 e lengthy 2,49 vs 1,54 (very short 0,50 vs 0,68). Em resumo, o número de arquivos é um bom indício de aumento do lifetime, e o efeito é mais forte nas bases Python.

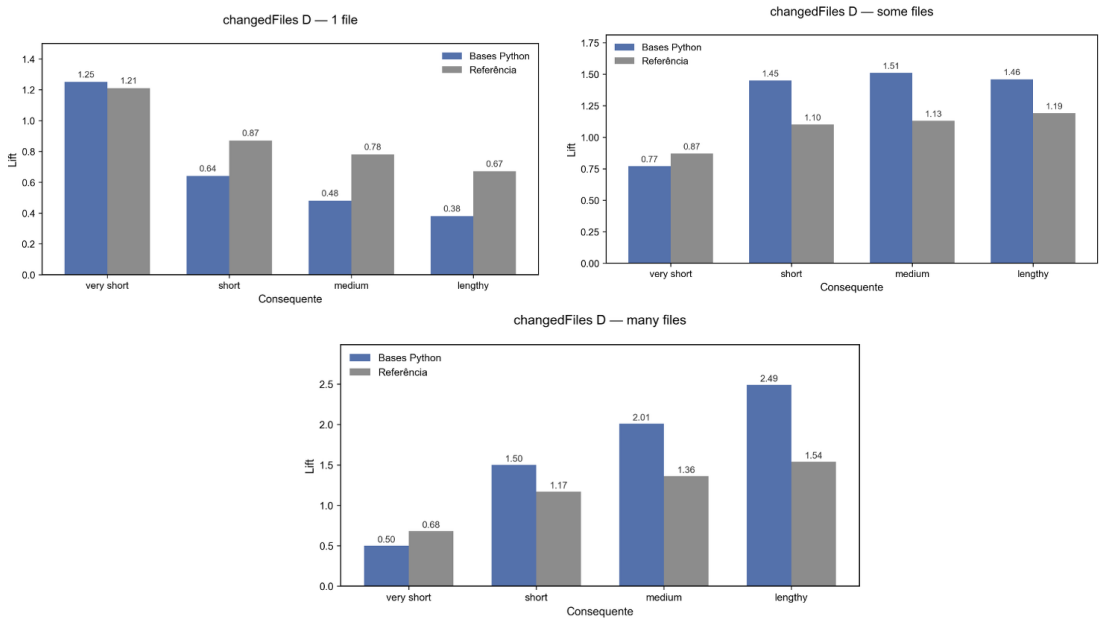

Figura 5. Changed Files → Lifetime

No cenário commitsPull → lifetime, o padrão é claro: quanto mais commits, maior o lifetime, e o efeito é mais forte nas bases Python. Com 1 commit, há associação positiva com very short e negativa com as demais classes — valores praticamente iguais à referência (very short 1,14 vs 1,14; short 0,76 vs 0,91; medium 0,68 vs 0,84; lengthy 0,69 vs 0,80). Com some commits, a associação migra para prazos maiores, com aumentos de lift em Python em relação à referência (short 1,60 vs 1,24, medium 1,59 vs 1,29, lengthy 1,36 vs 1,28; very short permanece ¡1 em 0,75 vs 0,75). Com many commits, o crescimento é acentuado: very short 0,37 vs 0,51 (negativa), short 1,62 vs 1,14, medium 2,41 vs 1,54 e lengthy 2,84 vs 1,92. Em síntese, o número de commits é um bom indicativo de aumento do tempo até decisão, replicando a tendência da referência e amplificando-a no ecossistema Python, sobretudo em medium e lengthy.

No cenário total lines → lifetime, com 1 line o efeito é praticamente plano: os lifts ficam próximos e maiores do que 1 em todas as classes (1,27 em Python vs 1,24 na referência), sem discriminar o lifetime. Com some lines, há associação positiva apenas com very short (1,23 vs 1,24) e associação negativa para short/medium/lengthy (0,70 vs 0,86; 0,53 vs 0,73; 0,40 vs 0,61), sendo a queda mais forte em Python. Com many lines, ocorre a inversão esperada: very short fica ¡ 1 (0,63 vs 0,72), enquanto short/medium/lengthy

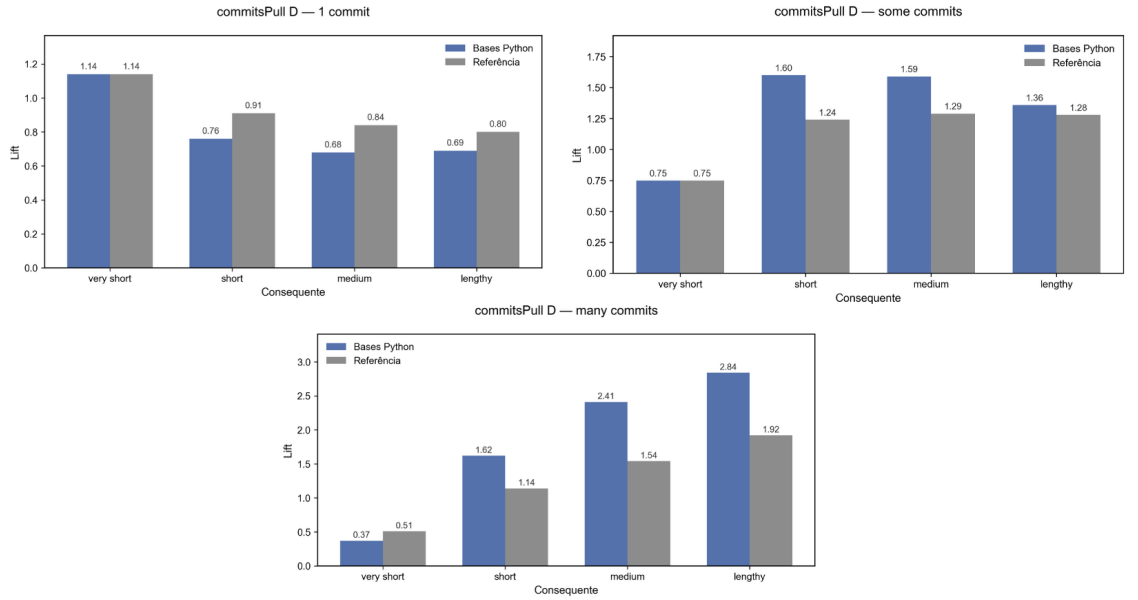

Figura 6. Commits → Lifetime

passam a positivas e mais intensas nas bases Python (short 1,50 vs 1,16, medium 1,77 vs 1,31, lengthy 1,98 vs 1,43). Resumo: quanto mais linhas modificadas, maior o lifetime; o gradiente é mais forte em Python, e o caso de 1 line é quase neutro (efeito plano).

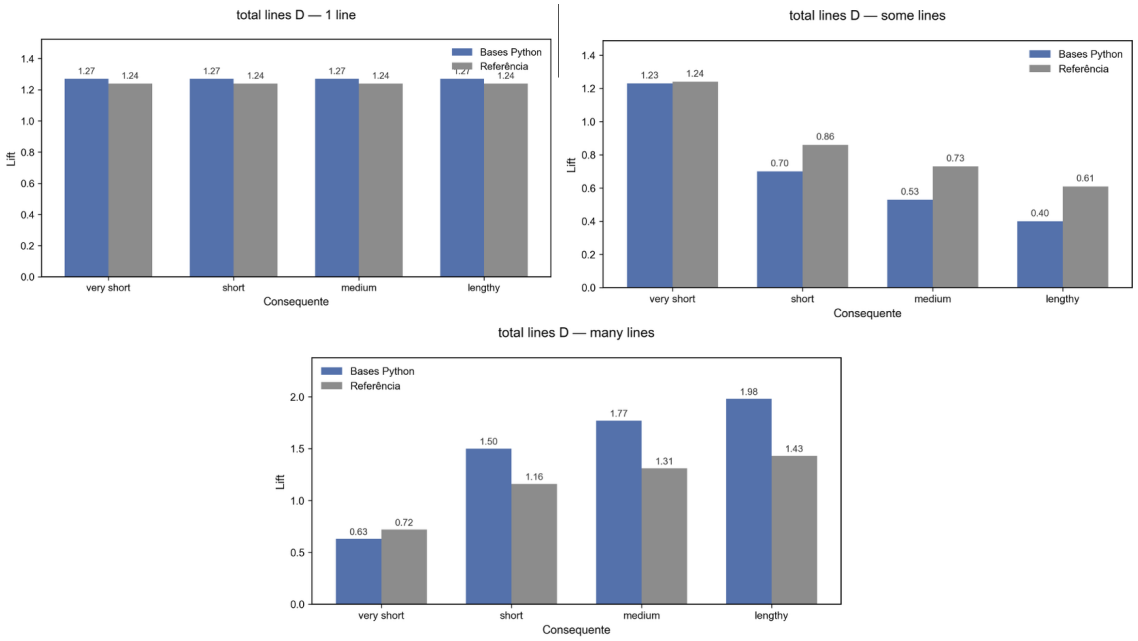

Figura 7. total Lines → Lifetime

No cenário requester experience → lifetime, três faixas foram analisadas. Com no contribution, há mudança de tendência em → short: na referência o lift é 0,87 (negativo) e nas bases Python é 1,23 (positivo). Em → very short permanece < 1 (0,66 vs 0,76) e em → medium/lengthy segue positivo, mais forte em Python (1,37/2,27 vs 1,05/1,85).

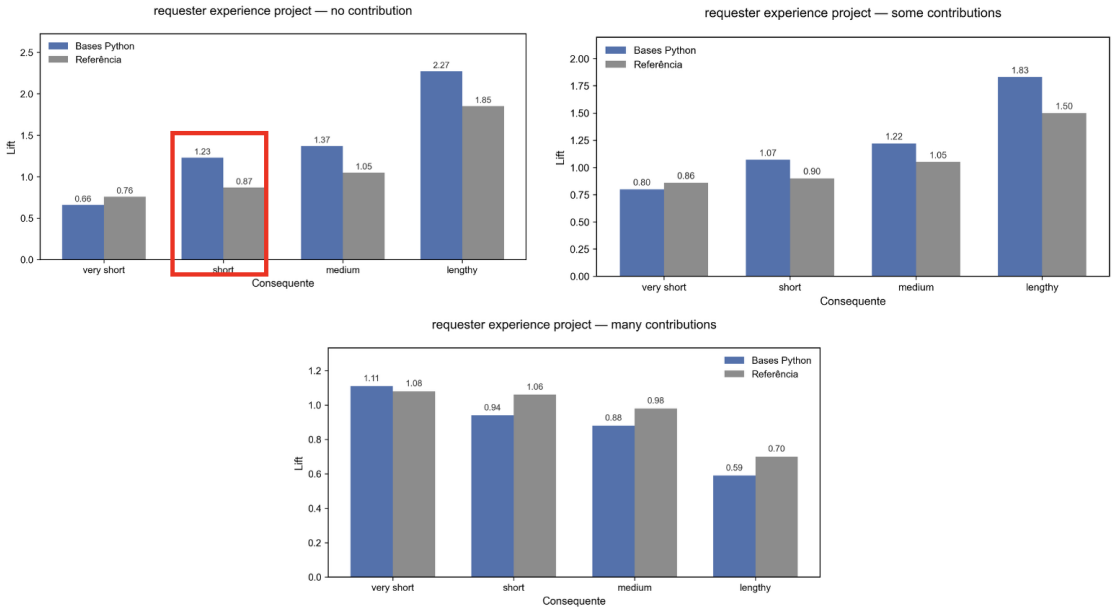

Figura 8. Requester Experience → lifetime

Com some contributions, o desenho repete a referência: → short/medium/lengthy > 1 e → very short < 1, porém com intensidades maiores em Python (1,07/1,22/1,83 vs 0,90/1,05/1,50). Com many contributions ocorre o inverso: → very short > 1 (1,11 vs 1,08) e → short/medium/lengthy caem para ≤ 1 em Python (0,94/0,88/0,59), abaixo da referência. Resumo: sem experiência, → short ganha força nas bases Python; com alguma experiência, o lifetime tende a alongar; com muita experiência, encurta para → very short.

No cenário de first pull = true, representado pela 9o padrão nas bases Python é mais inclinado para tempos de decisão maiores, e há uma mudança clara em *short*: na referência *short* tinha associação negativa (lift $\approx 0{,}87$), enquanto nas bases Python a associação é positiva (lift $\approx 1{,}26$). Além disso, *medium* e *lengthy* ficam mais fortes ($1{,}43$ vs. $1{,}05$; $2{,}37$ vs. $1{,}85$), e *very short* reduz levemente ($0{,}63$ vs. $0{,}76$).

No cenário de first pull = false, o comportamento é estável e mais neutro: *very short* levemente acima de 1 ($1{,}09$ vs. $1{,}07$), *short* próximo da neutralidade com pequena inversão ($0{,}94$ vs. $1{,}03$), e *medium/lengthy* seguem abaixo de 1 em ambos os casos ($0{,}90$ vs. $0{,}98$; $0{,}67$ vs. $0{,}77$).

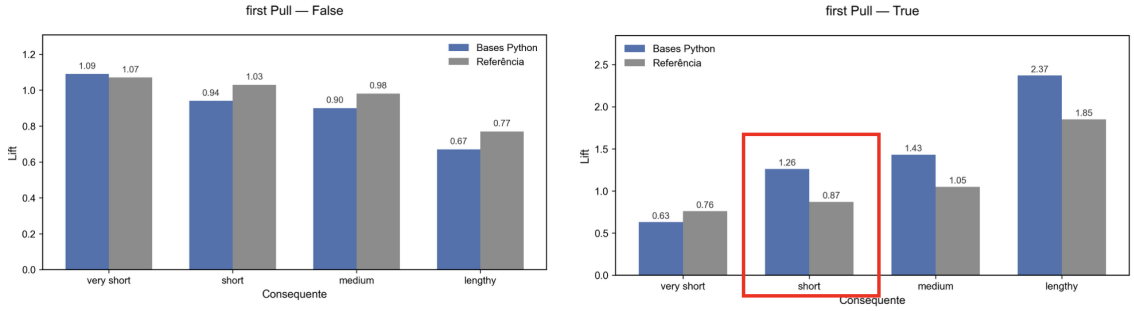

Figura 9. First Pull → lifetime

No cenário coreTeamFollowsRequester = False, não há associação relevante com o *lifetime*: os lifts ficam muito próximos de 1 em todas as classes (very short 1.00; short 0.99; medium 0.99; lengthy 1.02), reproduzindo o padrão da referência (valores praticamente idênticos).

No cenário coreTeamFollowsRequester = True, o padrão coincide com o de Soares, porém com intensidades um pouco maiores nas bases Python: há associação positiva com short (1,28 vs. 1,14) e leve associação positiva com medium (1,12 vs. 1,10), enquanto lengthy apresenta associação negativa (0,59 vs. 0,66). Para very short ambos ficam 1 (1,02 vs. 1,03). Em resumo, quando alguém do core segue o autor do PR, cresce a chance de tempos de vida mais curtos (short/medium) e cai a chance de tempos longos (lengthy); quando não segue, o efeito é praticamente neutro.

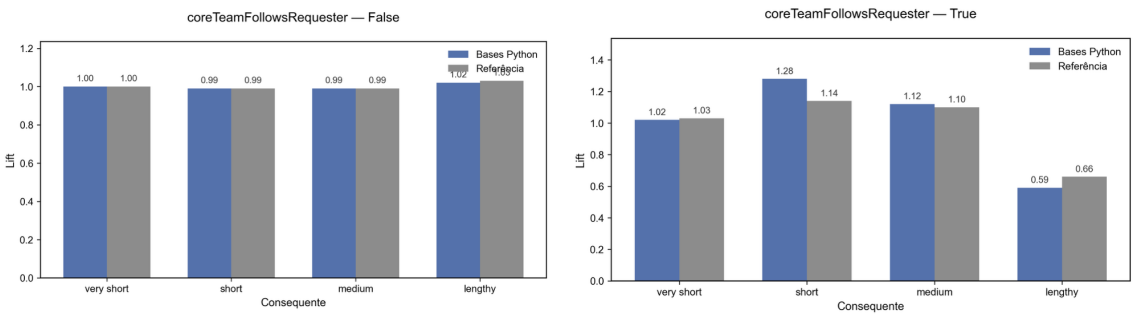

Figura 10. Core Team Follows Requester → lifetime

No cenário followers = no followers, observa-se associação positiva com lifetimes maiores: short (1,25 vs 1,01), medium (1,17 vs 0,84) e lengthy (1,29 vs 1,25). Em medium, há mudança de dependência em relação à referência, pois a regra no followers → medium passa de negativa (lift ¡ 1) no estudo de Soares para positiva (lift maior que 1) nas bases Python. Para very short, ambos os lifts ficam abaixo de 1 (0,88 vs 0,96), indicando menor probabilidade de lifetime muito curto quando não há seguidores.

Já no cenário followers = has followers, os valores ficam muito próximos de 1 em todas as classes (Python ≈ 0,97–1,01; referência ≈ 0,98–1,01), sem variações relevantes. Em outras palavras, ter seguidores não mostra associação sistemática com o lifetime em nenhum dos conjuntos.

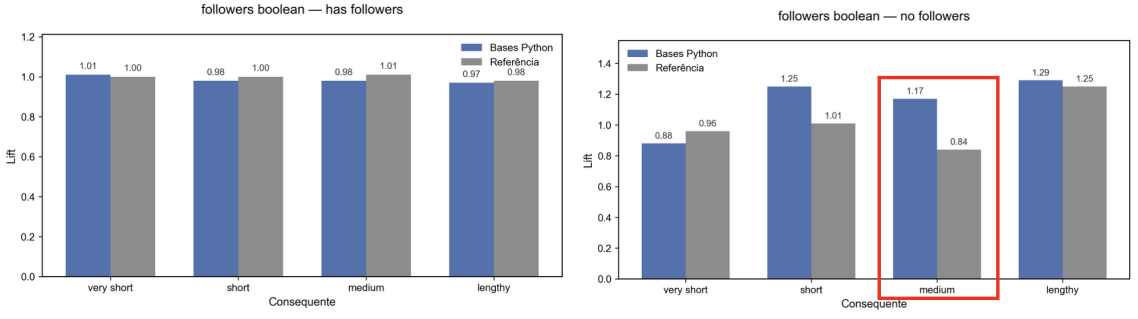

Figura 11. Followers → lifetime

No cenário typeDeveloper = external, os valores ficam muito próximos de 1 em todas as classes de lifetime, indicando efeito fraco/próximo da neutralidade. Há leve associação positiva apenas para very short (1,04 vs 1,03), e associações ligeiramente abaixo de 1 para short/medium/lengthy (0,97/0,94/0,88 vs 1,00/0,98/0,93). Em síntese, para external não há mudança de tendência relevante em relação à referência, e quanto maior o lifetime menor o lift (leve queda até lengthy).

No cenário typeDeveloper = core, o padrão é diferente e mais forte: há associação positiva crescente para short, medium e lengthy, com lifts maiores nas bases Python do que na referência. Em particular, core $\rightarrow$ short apresenta mudança de dependência (1,56 nas bases Python vs 0,97 na referência, passando de $\approx$ neutro/levemente < 1 para > 1). As associações core $\rightarrow$ medium e core $\rightarrow$ lengthy são bem fortes (2,06 e 3,01 vs 1,26 e 2,18), reforçando que PRs de desenvolvedores core tendem a permanecer mais tempo até decisão. Para very short, a associação permanece negativa e até mais pronunciada (0,38 vs 0,56).

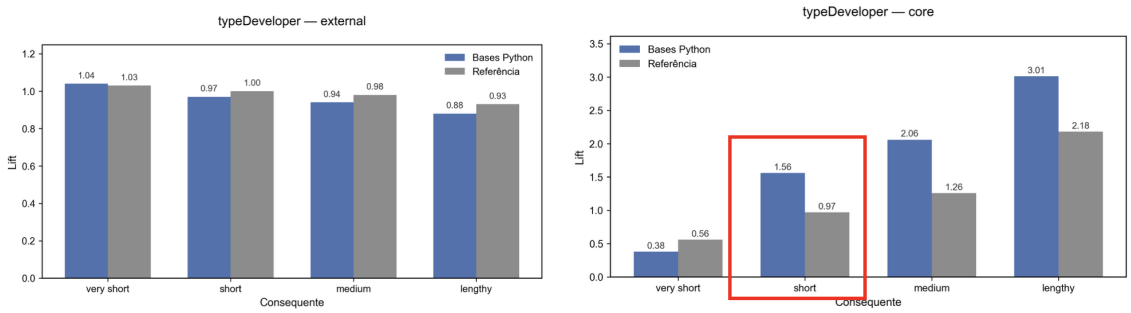

Figura 12. typedeveloper $\rightarrow$ lifetime

Das 85 regras analisadas, 55 (64,7%) tiveram variação ¡ 25% em relação ao lift da referência, indicando alta aderência ao padrão original. Apenas 5 regras (5,9%) apresentaram mudança de tendência (cruzaram o limiar de neutralidade, de <!1 para >!1 ou vice-versa). O restante distribuiu-se nas faixas de variação 25–50%, 50–75%, 75–100% e 100%, representando mudanças de intensidade, mas sem alterar o sentido da associação na maioria dos casos.

3.2. Reviwer

Na Figura 3 é possivel analisar algumas tendencias como no cenário de Reviewer (Close By), considerando changedFiles = 1 file, aparece um padrão consistente de especialização por revisor. No projeto Commcare, há associação positiva para yedi (lift = 1,38; Conf(Y $\rightarrow$ X) = 66,14%) e para dmyung (lift = 1,14; Conf(Y $\rightarrow$ X) = 55,00%). Em Django, o efeito é mais forte: ubernostrum (lift = 2,08; Conf(Y $\rightarrow$ X) = 85,71%) e kmtracey (lift = 1,73; Conf(Y $\rightarrow$ X) = 71,43%). Em IPython, epatters (lift = 1,84; Conf(Y $\rightarrow$ X) = 88,89%) e bfroehle (lift = 1,38; Conf(Y $\rightarrow$ X) = 66,67%) também se destacam. Em Pandas, os sinais são muito fortes: takluyver (lift = 3,54; Conf(Y $\rightarrow$ X) = 100%) e adamklein (lift = 2,45; Conf(Y $\rightarrow$ X) = 69,23%). Em rosdistro, há alta especialização com intensidade menor de associação: 130s (lift = 1,08; Conf(Y $\rightarrow$ X) = 100%) e dirk-thomas (lift = 1,04; Conf(Y $\rightarrow$ X) = 96,65%). Em síntese, PRs com um único arquivo alterado tendem a ser encaminhados a revisores específicos (lift > 1), e a confiança inversa elevada

indica especialização do revisor nesse padrão; já Conf(X $\rightarrow$ Y) costuma ser menor, pois vários revisores podem atuar sobre PRs desse tipo.

No caso changedFiles = some files, apresentado na 13 observa-se a recorrência de revisores específicos por projeto, com lifts de moderados a altos indicando aumento de probabilidade desses revisores aparecerem como Close By quando o PR altera some files. Em vários projetos há sinais claros de especialização representado pela $\mathrm{Conf}(Y \rightarrow X)$ elevado (tipicamente $\geq 80\%$) sugerindo que, quando esses revisores aparecem, com frequência isso ocorre em PRs com some files.

Base	Antecendente	Consequente	Suporte	Confianca x -> y	Confianca y -> x	Lift
Django	changedFiles_D=some files	closedBy=jezdez	0,54%	1,44%	74,07%	1,98
Django	changedFiles_D=some files	closedBy=andrewgodwin	0,80%	2,15%	65,22%	1,75
Ipython	changedFiles_D=some files	closedBy=rgbkrk	0,28%	0,82%	39,13%	1,15
Ipython	changedFiles_D=some files	closedBy=ellisonbg	3,19%	9,34%	38,01%	1,11
Pandas	changedFiles_D=some files	closedBy=TomAugspurger	1,48%	2,88%	75,00%	1,46
Pandas	changedFiles_D=some files	closedBy=hayd	1,08%	2,09%	70,59%	1,37
Rosdistro	changedFiles_D=some files	closedBy=tfoote	2,13%	32,11%	10,02%	1,51
Rosdistro	changedFiles_D=some files	closedBy=isucan	0,14%	2,11%	8,89%	1,34

Figura 13. somefiles $\rightarrow$ reviwer

Para changedFiles_D = many files (Figura 14), observa-se recorrência de revisores por projeto com *lift* acima de 1 (moderado a alto), indicando aumento de probabilidade desses revisores aparecerem como Close By quando o PR altera muitos arquivos. A especialização é sugerida pela $\mathrm{Conf}(Y \rightarrow X)$ elevada (tipicamente $\geq 80\%$, com casos próximos de 100%).

Base	Antecendente	Consequente	Suporte	Confianca x -> y	Confianca y -> x	Lift
Commcare	changedFiles_D=many files	closedBy=nickpell	1,27%	6,84%	30,38%	1,63
Commcare	changedFiles_D=many files	closedBy=snopoke	1,50%	8,07%	21,97%	1,18
Django	changedFiles_D=many files	closedBy=freakboy3742	0,19%	0,87%	33,33%	1,55
Django	changedFiles_D=many files	closedBy=loic	0,21%	1,00%	30,77%	1,43
Ipython	changedFiles_D=many files	closedBy=ellisonbg	2,23%	12,61%	26,57%	1,5
Ipython	changedFiles_D=many files	closedBy=fperez	3,25%	18,39%	25,93%	1,47
Pandas	changedFiles_D=many files	closedBy=jreback	14,02%	68,79%	24,43%	1,2
Pandas	changedFiles_D=many files	closedBy=wesm	4,30%	21,10%	20,08%	0,99
Rosdistro	changedFiles_D=many files	closedBy=tfoote	0,38%	48,89%	1,81%	2,3
Rosdistro	changedFiles_D=many files	closedBy=wjwwood	0,14%	17,78%	0,69%	0,87

Figura 14. manyfiles $\rightarrow$ reviwer

Para quando tem apenas 1 commit enviado (Figura 15), observa-se recorrência de revisores por projeto com *lift* > 1, indicando maior probabilidade de aparecerem como *Close By* quando a condição do antecedente ocorre. Os casos mais fortes combinam *lift* moderado/alto e $\mathrm{Conf}(Y \rightarrow X)$ elevada (tipicamente $\geq 80\%$), sugerindo especialização do revisor para esse tipo de PR. Em geral, as maiores confianças $Y \rightarrow X$ concentram-se em poucos revisores por projeto; já as regras com suporte muito baixo devem ser interpretadas com cautela, mesmo quando o *lift* é > 1.

No caso *commitsPull_D = some commits* (Figura 16), observam-se revisores recorrentes por projeto com *lift* > 1, indicando maior probabilidade de esses revisores aparecerem como *Close By* quando o PR traz *some commits*. Os destaques combinam $\mathrm{Conf}(X \rightarrow Y)$ e $\mathrm{Conf}(Y \rightarrow X)$ altas (tipicamente $\geq 80\%$), sugerindo especialização do revisor nesse tipo de PR. Regras com suporte muito baixo, contudo, devem ser interpretadas com cautela, mesmo quando o *lift* é > 1.

Base	Antecendente	Consequente	Suporte	Confianca x -> y	Confianca y -> x	Lift
Commcare	commitsPull_D=1 commit	closedBy=yedi	2,11%	3,40%	77,78%	1,26
Commcare	commitsPull_D=1 commit	closedBy=dmyung	1,63%	2,64%	71,25%	1,15
Django	commitsPull_D=1 commit	closedBy=erikr	0,24%	0,32%	100,00%	1,31
Django	commitsPull_D=1 commit	closedBy=evildmp	0,19%	0,25%	100,00%	1,31
Ipython	commitsPull_D=1 commit	closedBy=epatters	0,25%	0,46%	88,89%	1,66
Ipython	commitsPull_D=1 commit	closedBy=jdmarch	0,15%	0,29%	71,43%	1,33
Pandas	commitsPull_D=1 commit	closedBy=takluyver	0,09%	0,12%	100,00%	1,39
Pandas	commitsPull_D=1 commit	closedBy=jorisvandenbossche	5,56%	7,71%	91,18%	1,27
Rosdistro	commitsPull_D=1 commit	closedBy=130s	0,12%	0,13%	100,00%	1,1
Rosdistro	commitsPull_D=1 commit	closedBy=dirk-thomas	27,18%	29,85%	94,95%	1,04

Figura 15. 1 commit → reviwer

Base	Antecendente	Consequente	Suporte	Confianca x -> y	Confianca y -> x	Lift
Django	commitsPull_D=some commits	closedBy=bmispelon	0,70%	4,04%	23,85%	1,38
Django	commitsPull_D=some commits	closedBy=freakboy3742	0,13%	0,78%	23,81%	1,38
Ipython	commitsPull_D=some commits	closedBy=rgbkrk	0,34%	1,09%	47,83%	1,53
Ipython	commitsPull_D=some commits	closedBy=fperez	4,49%	14,40%	35,80%	1,15
Pandas	commitsPull_D=some commits	closedBy=hayd	0,54%	2,55%	35,29%	1,67
Pandas	commitsPull_D=some commits	closedBy=wesm	7,17%	33,97%	33,47%	1,59
Rosdistro	commitsPull_D=some commits	closedBy=kwc	0,14%	1,72%	34,78%	4,28
Rosdistro	commitsPull_D=some commits	closedBy=isucan	0,21%	2,58%	13,33%	1,64

Figura 16. some commits → reviwer

No caso *total_lines_D = 1 line* (Figura 17), observam-se revisores recorrentes por projeto com *lift* > 1, indicando maior probabilidade de esses revisores aparecerem como *Close By* quando o PR altera apenas uma linha. Os destaques combinam $\mathrm{Conf}(X \to Y)$ e $\mathrm{Conf}(Y \to X)$ elevadas (tipicamente $\geq 80\%$), sugerindo especialização do revisor nesse tipo de PR. Regras com suporte muito baixo devem ser interpretadas com cautela, mesmo quando o *lift* é > 1.

Base	Antecendente	Consequente	Suporte	Confianca x -> y	Confianca y -> x	Lift
Django	total_lines_D=1 line	closedBy=adrianholovaty	0,11%	5,48%	10,00%	5,11
Django	total_lines_D=1 line	closedBy=charettes	0,16%	8,22%	6,67%	3,41
Ipython	total_lines_D=1 line	closedBy=Carreau	0,46%	18,29%	3,03%	1,19
Ipython	total_lines_D=1 line	closedBy=minrk	0,90%	35,37%	2,91%	1,14
Pandas	total_lines_D=1 line	closedBy=cpcloud	0,09%	7,14%	5,41%	4,31
Pandas	total_lines_D=1 line	closedBy=y-p	0,18%	14,29%	3,77%	3,01
Rosdistro	total_lines_D=1 line	closedBy=wjwwood	0,58%	30,28%	2,83%	1,49
Rosdistro	total_lines_D=1 line	closedBy=tfoote	0,58%	30,28%	2,71%	1,43

Figura 17. 1 line → reviwer

Na figura 18, representa quando se tem algumas linhas alteradas as regras com *maior confiança* concentram-se em poucos revisores por projeto, com *lift* > 1 indicando aumento de probabilidade de esses revisores aparecerem como *Close By* quando a condição do antecedente é satisfeita. Nos destaques, observa-se $\mathrm{Conf}(X \to Y)$ elevada (regra útil para predição do revisor) e, em vários casos, $\mathrm{Conf}(Y \to X)$ também alta ($\geq 80\%$), sugerindo *especialização* do revisor naquele tipo de PR. Regras com suporte muito baixo devem ser interpretadas com cautela, mesmo quando o *lift* é > 1.

Na Figura 19, a regra analisada é total_lines_D = many lines; observa-se recorrência de revisores específicos por projeto com *lift* > 1, indicando aumento de probabilidade de esses revisores aparecerem como *Close By* quando o PR altera apenas uma

Base	Antecendente	Consequente	Suporte	Confianca x -> y	Confianca y -> x	Lift
Django	total_lines_D=some lines	closedBy=erikr	0,19%	0,40%	77,78%	1,66
Django	total_lines_D=some lines	closedBy=kmtracey	0,13%	0,29%	71,43%	1,52
Ipython	total_lines_D=some lines	closedBy=epatters	0,22%	0,44%	77,78%	1,56
Ipython	total_lines_D=some lines	closedBy=jdmarch	0,15%	0,31%	71,43%	1,43
Pandas	total_lines_D=some lines	closedBy=takluyver	0,09%	0,25%	100,00%	2,8
Pandas	total_lines_D=some lines	closedBy=changhiskhan	0,72%	2,01%	69,57%	1,95
Rosdistro	total_lines_D=some lines	closedBy=130s	0,12%	0,13%	100,00%	1,07
Rosdistro	total_lines_D=some lines	closedBy=dirk-thomas	27,27%	29,25%	95,25%	1,02

Figura 18. some lines $\rightarrow$ reviwer

linha. Em vários casos, a especialização do revisor é sugerida por $\mathrm{Conf}(Y \rightarrow X)$ elevada (tipicamente $\geq 80\%$), ou seja, quando o revisor aparece, com frequência está associada a PRs de 1 linha. Regras com suporte muito baixo, ainda que com *lift* > 1, devem ser interpretadas com cautela.

Base	Antecendente	Consequente	Suporte	Confianca x -> y	Confianca y -> x	Lift
Django	total_lines_D=many lines	closedBy=jphalip	0,51%	1,00%	82,61%	1,61
Django	total_lines_D=many lines	closedBy=jezdez	0,54%	1,05%	74,07%	1,45
Ipython	total_lines_D=many lines	closedBy=ellisonbg	4,98%	10,45%	59,41%	1,25
Ipython	total_lines_D=many lines	closedBy=fperez	7,43%	15,58%	59,26%	1,24
Pandas	total_lines_D=many lines	closedBy=TomAugspurger	1,43%	2,27%	72,73%	1,15
Pandas	total_lines_D=many lines	closedBy=jtratner	1,16%	1,85%	72,22%	1,14
Rosdistro	total_lines_D=many lines	closedBy=kwc	0,10%	2,14%	26,09%	5,35
Rosdistro	total_lines_D=many lines	closedBy=tfoote	1,46%	30,00%	6,90%	1,41

Figura 19. many lines $\rightarrow$ reviwer

Na Figura 20, a regra analisada é *typeDeveloper* $= core \rightarrow CloseBy$, onde $core$ indica que o autor do PR é do time principal do projeto. Observa-se, em vários repositórios, associação positiva de $core$ com revisores específicos (*lift* > 1), sugerindo que certos mantenedores tendem a aparecer como *Close By* quando o PR é aberto por alguém do time principal. Em diversos casos, a especialização do revisor é reforçada por $\mathrm{Conf}(Y \rightarrow X)$ elevada (tipicamente $\geq 80\%$), indicando que, quando esse revisor aparece, com frequência é em PRs de desenvolvedores $core$.

Base	Antecendente	Consequente	Suporte	Confianca x -> y	Confianca y -> x	Lift
Commcare	typeDeveloper=core	closedBy=czue	0,13%	45,00%	0,36%	1,25
Django	typeDeveloper=core	closedBy=MarkusH	0,48%	2,35%	38,30%	1,86
Django	typeDeveloper=core	closedBy=ramiro	0,24%	1,17%	30,00%	1,46
Ipython	typeDeveloper=core	closedBy=rgbkrk	0,09%	1,70%	13,04%	2,39
Ipython	typeDeveloper=core	closedBy=bfroehle	0,15%	2,84%	7,94%	1,46
Pandas	typeDeveloper=core	closedBy=shoyer	0,31%	2,97%	17,95%	1,7
Pandas	typeDeveloper=core	closedBy=jorisvandenbossche	0,90%	8,47%	14,71%	1,39
Rosdistro	typeDeveloper=core	closedBy=tfoote	0,24%	29,17%	1,15%	1,37
Rosdistro	typeDeveloper=core	closedBy=wjwwood	0,16%	18,75%	0,77%	0,92

Figura 20. coreteam $\rightarrow$ reviwer

Na Figura 21, a regra analisada é $\mathrm{typeDeveloper} = external \rightarrow Close\ By$. Observa-se associação positiva em diferentes repositórios ($\mathrm{lift} > 1$), com intensidade variável por projeto. Em vários casos há forte especialização do revisor: $\mathrm{Conf}(Y \rightarrow X) = 100\%$ para alguns pares — por exemplo, quando $Close\ By = jacobian$, sempre que esse revisor aparece o solicitante é externo. Em outros projetos, $\mathrm{Conf}(Y \rightarrow X)$ também é alta ($\geq 80\%$), ainda que com suporte baixo a moderado. Em síntese, quando o

autor é externo, emergem revisores característicos por projeto, com especialização marcada em casos como o de jacobian.

Base	Antecendente	Consequente	Suporte	Confianca x -> y	Confianca y -> x	Lift
Commcare	typeDeveloper=external	closedBy=czue	35,80%	35,90%	99,64%	1
Commcare	typeDeveloper=external	closedBy=dannyroberts	14,37%	14,41%	99,70%	1
Django	typeDeveloper=external	closedBy=jacobian	0,24%	0,30%	100,00%	1,26
Django	typeDeveloper=external	closedBy=kmtracey	0,19%	0,24%	100,00%	1,26
Ipython	typeDeveloper=external	closedBy=epatters	0,28%	0,29%	100,00%	1,06
Ipython	typeDeveloper=external	closedBy=jdmarch	0,22%	0,23%	100,00%	1,06
Pandas	typeDeveloper=external	closedBy=takluyver	0,09%	0,10%	100,00%	1,12
Pandas	typeDeveloper=external	closedBy=TomAugspurger	1,88%	2,10%	95,45%	1,07
Rosdistro	typeDeveloper=external	closedBy=130s	0,12%	0,12%	100,00%	1,01
Rosdistro	typeDeveloper=external	closedBy=dirk-thomas	28,45%	28,69%	99,39%	1

Figura 21. external → reviwer

Na Figura 22, a regra analisada é $\text{coreTeamFollowsRequester} = True \rightarrow Close\ By$. Em vários repositórios observa-se associação positiva ($\text{lift} > 1$), chegando muitas as vezes ser redundante de tão alto o lift gerado entre o fato de o time principal seguir o autor do PR e a recorrência de revisores específicos como $Close\ By$. Em diversos pares há indícios de especialização do revisor, com $\text{Conf}(Y \rightarrow X)$ elevada (tipicamente $\geq 80\%$, com casos próximos de 100%), sugerindo que, quando esses revisores aparecem, frequentemente é em PRs de autores seguidos pelo core team. O suporte tende a ser baixo a moderado, variando por projeto, mas os padrões são consistentes: relações de seguimento no grafo social do projeto aumentam a probabilidade de certos mantenedores revisarem o PR.

Base	Antecendente	Consequente	Suporte	Confianca x -> y	Confianca y -> x	Lift
Commcare	coreTeamFollowsRequester=true	closedBy=kennknowles	0,23%	2,44%	41,03%	4,37
Commcare	coreTeamFollowsRequester=true	closedBy=czue	8,74%	93,13%	24,32%	2,59
Django	coreTeamFollowsRequester=true	closedBy=jezdez	0,16%	46,15%	22,22%	63,76
Ipython	coreTeamFollowsRequester=true	closedBy=ivanov	2,07%	32,52%	82,72%	12,97
Ipython	coreTeamFollowsRequester=true	closedBy=rgbkrk	0,50%	7,77%	69,57%	10,91
Pandas	coreTeamFollowsRequester=true	closedBy=wesm	6,27%	93,96%	29,29%	4,39
Pandas	coreTeamFollowsRequester=true	closedBy=cpcloud	0,18%	2,68%	10,81%	1,62
Rosdistro	coreTeamFollowsRequester=true	closedBy=130s	0,12%	12,73%	100,00%	104,35
Rosdistro	coreTeamFollowsRequester=true	closedBy=wjwwood	0,77%	80,00%	3,77%	3,93

Figura 22. core team follows requester true → reviwer

Na Figura 23, a regra analisada é $\text{coreTeamFollowsRequester} = False \rightarrow Close\ By$. Em geral, os *lifts* ficam próximos de 1, com alguns casos > 1 indicando aumento modesto da probabilidade de revisores específicos aparecerem como $Close\ By$ quando o autor do PR não é seguido pelo *core team*. A especialização do revisor ($\text{Conf}(Y \rightarrow X)$ elevada) aparece com menor frequência do que no caso $\text{coreTeamFollowsRequester} = True$, mas, quando ocorre, pode atingir patamares altos (tipicamente $\geq 80\%$). O suporte tende a ser baixo, variando por projeto. Em síntese, a ausência de vínculo de "seguir" não elimina padrões de revisão, porém eles são mais fracos e menos consistentes do que quando o autor é seguido pelo time principal.

4. Conclusão

Os resultados indicam que, no cenário Lifetime, o comportamento geral dos projetos Python acompanha o observado por [Soares 2017]. Ainda assim, 5 de 85 regras apresentaram mudança de tendência, sinalizando que, ao ampliar o conjunto de atributos

Base	Antecendente	Consequente	Suporte	Confianca x -> y	Confianca y -> x	Lift
Commcare	coreTeamFollowsRequester=false	closedBy=dannyroberts	14,41%	15,90%	100,00%	1,1
Commcare	coreTeamFollowsRequester=false	closedBy=snopoke	6,85%	7,56%	100,00%	1,1
Django	coreTeamFollowsRequester=false	closedBy=timgraham	62,90%	63,12%	100,00%	1
Django	coreTeamFollowsRequester=false	closedBy=aaugustin	5,23%	5,25%	100,00%	1
Ipython	coreTeamFollowsRequester=false	closedBy=takluyver	23,46%	25,06%	99,87%	1,07
Ipython	coreTeamFollowsRequester=false	closedBy=fperez	12,53%	13,39%	100,00%	1,07
Pandas	coreTeamFollowsRequester=false	closedBy=jreback	57,39%	61,50%	100,00%	1,07
Pandas	coreTeamFollowsRequester=false	closedBy=y-p	4,75%	5,09%	100,00%	1,07
Rosdistro	coreTeamFollowsRequester=false	closedBy=dirk-thomas	28,63%	28,91%	100,00%	1,01
Rosdistro	coreTeamFollowsRequester=false	closedBy=vrabaud	27,55%	27,81%	100,00%	1,01

Figura 23. core team follows requester false → reviwer

e variáveis de processo, é plausível identificar padrões distintos dos reportados no estudo de referência. Em especial, vimos deslocamentos pontuais de associação para classes específicas de lifetime quando certas características do PR (por exemplo, volume de mudanças ou experiência do proponente) mudam.

No cenário Reviewer (Close By), as evidências reforçam a tese de que, mesmo em ecossistemas open-source, há especialização e concentração de revisão: determinados mantenedores aparecem com frequência como revisores sob certos contextos (por projeto/atributo), e em vários casos a confiança inversa sugere domínio daquele revisor em padrões específicos de PR. Esse achado é consistente com a literatura e destaca efeitos organizacionais e de divisão de trabalho nas comunidades.

Como implicações práticas, os resultados podem apoiar triagem (estimando tempo até decisão), dimensionamento de mudanças (evitando perfis que alongam o ciclo) e alocação de revisores (valorizando especialização onde ela melhora fluxo e qualidade). Como limitações, destacam-se o suporte baixo de algumas regras, a discretização de variáveis e a harmonização de rótulos (que podem suavizar nuances). Como trabalhos futuros, propõe-se: (i) ampliar atributos e projetos, (ii) analisar janelas temporais para captar evolução de papéis, (iii) contrastar diferentes linguagens/stack e (iv) combinar regras com modelos preditivos para priorização de PRs.

Referências

Soares (2017). *On The Nature Of Pull Requests*. PhD thesis, Universidade Federal Fluminense.