UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Douglas Montanha Giordano

PREDIÇÃO DO EVENTO DE TURNOVER EM EQUIPES DE SOFTWARE LIVRE E DE CÓDIGO ABERTO

Santa Maria, RS
2019

Douglas Montanha Giordano

PREDIÇÃO DO EVENTO DE TURNOVER EM EQUIPES DE SOFTWARE LIVRE E DE CÓDIGO ABERTO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação, Área de Concentração em Computação Aplicada, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Mestre em Ciência da Computação**.

ORIENTADOR: Prof. Marcelo Serrano Zanetti

Santa Maria, RS
2019

Giordano, Douglas
Predição do Evento de Turnover Em Equipes de Software Livre e de Código Aberto / Douglas Giordano.- 2019.
95 p.; 30 cm

Orientador: Marcelo Serrano Zanetti
Dissertação (mestrado) - Universidade Federal de Santa Maria, Centro de Tecnologia, Programa de Pós-Graduação em Ciência da Computação , RS, 2019

1. Sistemas Complexos 2. Engenharia de Software 3. Predição 4. Turnover I. Serrano Zanetti, Marcelo II. Título.

Sistema de geração automática de ficha catalográfica da UFSM. Dados fornecidos pelo autor(a). Sob supervisão da Direção da Divisão de Processos Técnicos da Biblioteca Central. Bibliotecária responsável Paula Schoenfeldt Patta CRB 10/1728.

Endereço: Travessa Alvorada, n. 36

End. Eletr.: douglasmontanhagiordano@gmail.com

Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

PREDIÇÃO DO EVENTO DE TURNOVER EM EQUIPES DE SOFTWARE LIVRE E DE CÓDIGO ABERTO

elaborada por
Douglas Montanha Giordano

como requisito parcial para obtenção do grau de
Mestre em Ciência da Computação

COMISSÃO EXAMINADORA:

Marcelo Serrano Zanetti, Dr.
(Presidente/Orientador)

Lisandra Manzoni Fontoura, Dr. (UFSM)

Igor Scaliante Wiese, Dr. (UTFPR)

Santa Maria, 26 de Março de 2019.

DEDICATÓRIA

Dedico este trabalho a minha família: Ana Cláudia (mãe), Rui (pai), Diego e Diogo (irmãos) e Adriana (namorada).

AGRADECIMENTOS

Primeiramente agradeço a minha mãe, Ana Cláudia, e ao meu pai, Rui, pelo apoio durante os anos de mestrado e toda vida. Agradeço também aos meus irmãos Rui Diego e Diogo pelos momentos de alegria. Agradeço a minha namorada, Adriana, por aguentar meus momentos de estresse e não terminar comigo.

Agradeço ao meu orientador Marcelo Zanetti por não desistir de mim, e me ajudar a seguir a direção certa no meio de um conjunto de ideias mirabolantes. Meus agradecimentos também ao CNPq pela bolsa de estudo no mestrado.

E por último, mas não menos importante, aos meus amigos Alex, Miguel, Ecar e Gean, pelas risadas e apoio durante todo o processo de tristeza da escrita da dissertação.

Todas as grandes coisas são simples. E muitas podem ser expressas numa só palavra: liberdade; justiça; honra; dever; piedade; esperança.

(Winston Churchill)

RESUMO

PREDIÇÃO DO EVENTO DE TURNOVER EM EQUIPES DE SOFTWARE LIVRE E DE CÓDIGO ABERTO

AUTOR: Douglas Montanha Giordano
ORIENTADOR: Marcelo Serrano Zanetti

Os projetos livres e de código aberto, em tradução literal da língua Inglesa *free and open source software* (FOSS) têm uma alta rotatividade de colaboradores. RAILS, ELIXIR e LINUX são exemplos famosos de projetos FOSS. Essa rotatividade acontece quando pessoas deixam o projeto ou novas entram. Quando um contribuidor deixa de ser ativo em um projeto, uma série de consequências negativas podem afetar o software. Dentre algumas situações estão a falta de manutenção em módulos do software. Portanto, a saída de um desenvolvedor pode ser prejudicial ao projeto FOSS. Caso exista viável de descobrir quem não realizará mais contribuições no projeto, os gestores podem executar ações para evitar a saída do contribuidor ou diminuir as possíveis consequências. Este estudo propõe um classificador que faça a predição do turnover quando o mesmo aconteça. Com base nos dados de repositórios FOSS do GITHUB, o classificador oferecerá como resposta se o contribuidor interagiu pela última vez ou continuará no projeto. As métricas de desempenho dos classificadores apresentaram bons resultados com relação aos trabalhos anteriores. A *precision* foi de 0,83 e a *recall* de 0,35 para classe de contribuidores que irão continuar no projeto. Por outro lado, a classe de contribuidores que podem ficar inativos tiveram uma *precision* de 0,81 e uma *recall* de 0,98.

Palavras-chave: Sistemas Complexos. Engenharia de Software. Predição. Turnover.

ABSTRACT

TURNOVER EVENT PREDICTION IN FREE SOFTWARE AND OPEN SOURCE TEAMS

AUTHOR: Douglas Montanha Giordano
ADVISOR: Marcelo Serrano Zanetti

FOSS projects like RAILS, ELIXIR, LINUX among others have a high employee turnover. This turnover happens when people leave the project or new ones come in. When a contributor ceases to be active in a project, a series of negative consequences can affect the software. Among some situations are the lack of maintenance in software modules. Therefore, a developer's exit can be detrimental to the FOSS project. If a way of knowing who will not make the most contributions to the project is offered, managers or core staff may have time to avoid leaving the contributor or lessen the consequences. This study propose a classifier that predicts turnover when it happens. Based on the FOSS data repositories of the GITHUB, the classifier will offer as a response whether the contributor has last interacted or will continue in the project. The results of the best classifier, offered a good performance. The precision was 0.83 and the recall of 0.35 for the class of contributors who will continue in the project. On the other hand, the class of contributors who may be inactive had a precision of 0.81 and an recall of 0.98.

Keywords: System Complex. Software Engineering. Predict. Turnover.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

$FOSS$	*Free and Open Source Software*
SVM	*Suport Vector Machine*
API	*Application Programming Interface*
$JSON$	*JavaScript Object Notation*
$HTML$	*HyperText Markup Language*
XML	*eXtensible Markup Language*
MCC	*Matthews Correlation Coefficient*

LISTA DE SÍMBOLOS

$\sum$	Somatório
σ	Sigma
$\neq$	Inequação
λ	Lambda

SUMÁRIO

1 INTRODUÇÃO

A rotatividade de pessoas, ou *turnover*, como expresso na língua Inglesa, é um tema recorrente na Engenharia de Software, especialmente no contexto de comunidades de desenvolvimento de Software Livre e de Código Aberto (FOSS) (ROBLES; GONZALEZ-BARAHONA, 2006).

O *turnover* pode representar a perda de membros ativos da comunidade, daqui em diante denominados *contribuidores*, potencialmente sendo substituídos por novos contribuidores recém ingressos no projeto, ou seja, contribuidores deixam o projeto enquanto que novos contribuidores ingressam. Intuitivamente, a perda de um contribuidor importante para o projeto pode afetar seu andamento negativamente, por exemplo, atrasos no cumprimento de prazos, aumento na quantidade de problemas de software (i.e. *bugs*) reportados, e até mesmo dificuldades na articulação entre contribuidores. Por outro lado, o ingresso de um novo contribuidor pode trazer renovação dentro do projeto, com novas ideias e habilidades afetando positivamente em seu desenvolvimento. Desta forma, a literatura descreve uma série de estudos de caso sobre as diversas consequências do *turnover* sobre o andamento de projetos de software (HALL et al., 2008; FOUCAULT et al., 2015; RIGBY et al., 2016).

Novos contribuidores são geralmente bem-vindos a uma comunidade FOSS, aos quais são atribuídas responsabilidades compatíveis com as suas experiências prévias, interesses ou habilidades. Apesar de a atração de novos contribuidores ser uma questão importante, esta não será abordada neste trabalho. Por outro lado, uma vez que um contribuidor se interessa pelo projeto, a mão de obra torna-se disponível, e é trivial aloca-la à uma demanda existente. Considerando-se que uma parcela significativa dos contribuidores de projetos FOSS são voluntários, ou seja, não possuem amarras legais ou financeiras com o projeto além de sua própria motivação, os mesmos podem abandonar o projeto espontaneamente e sem aviso prévio. Assim, ao perder subitamente um contribuidor responsável por uma tarefa muito específica e complexa, ao gestor do projeto cabe apenas aguardar que um novo contribuidor se interesse e seja capaz de assumir a tarefa vacante. A substituição pode levar certo tempo, influenciando negativamente no desenvolvimento do projeto, como exposto anteriormente. Portanto, pelo fato de que a manutenção de contribuidores ser uma questão mais imediatista para o gestor de um projeto, além de tratável internamente através de dados extraídos de dentro da comunidade, este trabalho foca em apontar os prováveis eventos de *turnover*, possibilitando ao gestor executar ações para evitar o turnover ou diminuir suas possíveis consequências.

A literatura de fato lista a motivação dos contribuidores como aspecto fundamental no *turnover*. Como aspectos relacionados com a manutenção de contribuidores nos projetos são mencionados o reconhecimento e renome do projeto, o bom gerenciamento e a

identificação pessoal com as tarefas (HALL et al., 2008). Além disso, contribuidores que tem como principal função o desenvolvimento de código tem uma maior chance de continuar contribuindo (LIN; ROBLES; SEREBRENIK, 2017). Ou seja, no contexto de FOSS, contribuidores relacionados a outras tarefas, como a documentação ou tratamento de relatórios de *bugs*, tem uma maior chance de sair do projeto.

Dada a subjetividade relacionada ao levantamento das motivações para um desenvolvedor manter-se no projeto, busca-se indicadores correlatos e quantitativos que podem ser inferidos diretamente dos diversos repositórios de código e de gerenciamento utilizados pela comunidade no desenvolvimento de um dado projeto. Os indicadores utilizados expressam características individuais dos contribuidores e a interação entre os mesmos. Esta última, denominada *análise de redes sociais*, é realizada através da construção de um grafo na qual os vértices representam os contribuidores e as arestas representam suas interações, o que possibilita o cálculo de diversas métricas utilizadas como indicadores. O teor das interações pode ser analisado através de metodologia quantitativa, conhecida por *análise de sentimentos*, que estima a qualidade da comunicação entre os contribuidores. Apesar da interação complexa entre os diferentes indicadores, que dificulta ou impossibilita o estabelecimento de relações de causa e efeito, verificou-se que contribuidores prestes a deixar o projeto apresentam um padrão em seus indicadores distinto do padrão apresentado por contribuidores que pretendem continuar no projeto. Isto possibilita a criação de um método para predizer quais contribuidores provavelmente deixarão o projeto, e permite que o gestor, ao monitorar a comunidade quantitativamente nestes termos, aja de forma a mitigar os efeitos negativos do *turnover*. Para tanto, este trabalho foca em projetos FOSS listados na plataforma de desenvolvimento e versionamento GITHUB, pela facilidade na obtenção de dados e pela diversidade de projetos, em termos de áreas e número de contribuidores.

1.1 OBJETIVO

O objetivo deste trabalho é propor um classificador para os contribuidores de um projeto FOSS, indicando quem tem indícios de um possível *turnover*.

1.1.1 Objetivos Específicos

Este trabalho tem como objetivos específicos:

- propor uma definição de quando ocorre o evento de *turnover* em um projeto FOSS.

- correlacionar um conjunto de indicadores extraídos de cada contribuidor com o *turnover*.
- criar um ou mais classificadores para categorizar contribuidores com indicio de *turnover*.

1.2 ESTRUTURA

Este trabalho está organizado da seguinte forma: na seção 2 o referencial teórico é descrito, abordando os principais conceitos e tecnologias envolvidas neste estudo; na seção 3 são explorados os trabalhos relacionados a predição do *turnover*; na seção 4 a metodologia executada para criar o classificador é abordada; na seção 5 são apresentados os resultados de desempenho dos classificadores, descrevendo os melhores; na seção 6 são feitas considerações sobre este estudo.

2 REFERENCIAL TEÓRICO

A proposta de um classificador para o evento de turnover envolve um conjunto de saberes bastante distintos. Portanto, nessa seção serão descritos os conhecimentos básicos para entender os demais capítulos do trabalho.

Primeiramente, nas Seções 2.1, 2.2 e 2.3, é abordado o conhecimento relacionado com a origem dos dados e o entendimento do turnover, descrevendo alguns estudos correlacionados. Posteriormente, na Seção 2.4, abordamos as técnicas quantitativas utilizadas para extrair características dos contribuidores.

2.1 SOFTWARE LIVRE E DE CÓDIGO ABERTO

Um software de código aberto permite que as contribuidores acessem o código fonte do software, possibilitando-os indicar melhorias. A licença livre de um software permite alterar, compartilhar e estudar o código livremente. Essas duas características denominam o que chamamos de FOSS.

Atualmente existe um grande número de projetos FOSS, mantidos por comunidades e também empresas. Exemplos famosos de sucesso como LINUX, mostram os esforços da comunidade FOSS nas últimas décadas (LERNER; TIROLE, 2001). Adicionalmente empresas famosas como por exemplo FACEBOOK [1] e MICROSOFT [2] também mantêm projetos FOSS.

Outra característica bastante marcante em projetos FOSS é o seu desenvolvimento através de equipes distribuídas. Os contribuidores comumente estão em locais diferentes e a maior parte da comunicação é feita por ferramentas *online* (GUZZI et al., 2013). Além disso, qualquer um pode tentar contribuir para um projeto FOSS, seja enviando possíveis melhorias ou mesmo relatando possíveis problemas no software.

Existe uma gama de estudos sobre a estrutura organizacional de um projeto FOSS. Por exemplo, o autor Riehle (2015) categoriza os indivíduos envolvidos como usuário, contribuidor e *commiter*. Tomando como base que qualquer pessoa pode enviar solicitações de mudança ou mesmo reportar um possível erro, o usuário é a primeira categoria descrita. O contribuidor é outra categoria, porém se torna mais restrito pois as suas solicitações de modificações devem ser aceitas. O *commiter* tem total acesso no envio de modificações ao projeto. Alguém que começa a enviar modificações ou relatar erros em um projeto FOSS acaba passando por todas essas categorias. Muitas vezes contribuidores casuais chegam ao nível de *commiter* pela qualidade de suas contribuições ao projeto FOSS.

[1] <https://github.com/facebook>
[2] <https://github.com/Microsoft>

A estrutura organizacional pode ser dividida também em núcleo e periferia. Com base em uma estrutura de rede social, toma-se como base que aqueles na borda da rede são a periferia do projeto FOSS. O núcleo são os desenvolvedores ou contribuidores mais centrais. Esses contribuidores mais centrais comumente são os criadores do projeto FOSS ou a equipe principal. Os autores Luthiger (2005), Toral, Martínez-Torres e Barrero (2010) e Crowston e Shamshurin (2017) fazem uso dessas classificações em suas pesquisas.

Além disso, um estudo de Riehle et al. (2014) apresenta indícios de que em torno de 50% dos desenvolvedores da estrutura organizacional do projeto FOSS são pagos. Portanto, um grande número de contribuidores ativos são contratados por empresas privadas, o que mostra a ascensão cada vez maior e também o esforço que empresas privadas colocam na manutenção e criação de projetos FOSS.

Outro ponto, que é importante salientar sobre a comunidade FOSS, é a parcela significativa de contribuidores que não recebe nenhum valor monetário para contribuir para um o projeto. O estudo de Bitzer, Schrettl e Schröder (2007) descreve os contribuidores voluntários, comumente caracterizados como indivíduos mais jovens, possuindo habilidades de programação mais altas. Ou seja, a diversão da programação é uma grande motivação para continuar contribuindo com o projeto. O aprendizado é outro motivador das contribuições no ambiente FOSS (YE; KISHIDA, 2003). Portanto, projetos com uma curva de aprendizado maior, podem ter uma menor quantidade de contribuidores. Além disso, o estilo de gerenciamento está relacionado com a motivação dos contribuidores (LI; TAN; TEO, 2012). Por outro lado, o estudo de Lakhani e Wolf (2003) relaciona fatores externos na motivação da contribuição, como o progresso na carreira e melhores perspectivas de empregos. Lakhani e Wolf (2003) também relacionam a motivação a desafios intelectuais, onde o objetivo é o aprimoramento das habilidades.

Atualmente existe uma grande variedade de plataformas abrigando projetos FOSS. Exemplos famosos como GITHUB, BITBUCKET e GITLAB, hospedando milhares de projetos. Este estudo foca nos projetos de software encontrados no GITHUB.

2.2 GITHUB

O GITHUB é uma plataforma de hospedagem de código-fonte com controle de versão baseado em GIT para projetos de software. Milhares de projetos de FOSS são hospedados no GITHUB, tornando-se uma ferramenta bastante conhecida entre os desenvolvedores de software. Projetos populares como RAILS[3] (*framework* para aplicações web), ELIXIR[4] (linguagem funcional e dinâmica para a construção de aplicações) e REACT[5] (bi-

[3]<https://github.com/rails/rails>
[4]<https://github.com/elixir-lang/elixir>
[5]<https://github.com/facebook/react>

blioteca JAVASCRIPT para a construção de interfaces para o usuário) tem seus repositórios de versionamento hospedados no GITHUB.

Algumas funções são mais comuns de serem utilizadas no repositório de controle de versionamento. Portanto quando é pesquisado algum repositório no GITHUB, comumente as opções utilizadas pela equipe são o controle de versionamento através de *commits* para armazenar as versões do código fonte do software, as *issues* para reportar erros e as requisições de *pull* para permitir que pessoas sem permissões de modificação no projeto realizem contribuições.

O GITHUB oferece outras possibilidades de controle do repositório. Porém, este estudo foca nas informações criadas em *commits*, *issues* e *pull requests* pois apresentam o maior número de interações entre os indivíduos. Além disso, essas funcionalidades são mais comuns de serem utilizadas em outros repositórios, possibilitando a expansão do estudo em outras ferramentas de controle de versionamento. As principais opções que o GITHUB oferece para o controle do projeto são descritas na lista abaixo:

- *Commits*: são modificações feitas em um arquivo ou ao conjunto de arquivos (GITHUB, 2018). Essas modificações são submetidas ao repositório por contribuidores, que juntamente enviam uma pequena descrição das modificações. Diariamente um projeto pode ter uma grande quantidade de modificações, consequentemente, uma grande quantidade de *commits*. Essas modificações são feitas em arquivos, comumente relacionados com a codificação do software. Cada modificação pode ter comentários, onde os usuários podem interagir.

- *Issues*: são as descrições de possíveis funcionalidades, problemas e(ou) melhorias para o software. Cada *issue* pode receber comentários, ser atribuída a outros usuários, ter status e também ter rótulos. As *issues* possuem uma seção especifica no GITHUB. Comumente além de desenvolvedores, usuários do projeto também podem criar ou comentar em *issues*, expondo problemas do software.

- *Pull Requests*: são solicitações de modificação ao projeto, ou seja, são utilizadas por usuários sem permissões diretas de modificações no repositório. Após a criação da requisição os membros internos do projeto decidem aceitar ou não as modificações recomendadas. As *pull requests* tem as mesmas possibilidades de interação que uma *issue*.

2.3 ROTATIVIDADE DE COLABORADORES

A rotatividade de pessoas, afeta várias áreas da sociedade. Administração, medicina, computação dentre outras várias áreas tem em comum em seu ambiente de trabalho

pessoas se desligando de atividades e novas pessoas entrando. O *turnover* acaba trazendo uma gama de riscos para atividades de desenvolvimento de software, no qual existe um grande número de pessoas contribuindo para a construção do software e que muitas vezes se baseiam no princípio de trabalho coletivo, como é o caso de projetos FOSS.

Quando um membro deixa uma equipe de software, existe um grande risco de componentes do software ficarem sem manutenção ou também dificuldades de encontrar pessoas com a mesma capacidade de execução de tarefas dentro do projeto de software (RIGBY et al., 2016). Quanto maior a dificuldade de uma atividade, mais problemático é a ocorrência da saída de um contribuidor que realiza a mesma. Além disso, quando existe a possibilidade de regeneração da equipe, o novo membro precisa passar por uma série de etapas até adquirir o conhecimento necessário para equiparar-se ao conhecimento técnico e organizacional da empresa que o antigo membro possuía. Portanto, o *turnover* de um membro da equipe de software pode levar ao atraso a entrega de um produto de software ou mesmo ao fracasso.

Ao longo do tempo, os projetos FOSS tem sido investigados em decorrência da sua taxa de *turnover*. Esses projetos possuem uma alta taxa de rotatividade de seus colaborares (ROBLES; GONZALEZ-BARAHONA, 2006). Dependendo da responsabilidade do colaborador no projeto de software, essa rotatividade pode acontecer isoladamente ou ocorrer em massa, ou seja, dependendo das características do colaborador, a rotatividade pode ocorrer com mais frequência. Casos de alta frequência de *turnover* ocorrem com usuários casuais, para os quais o número de interações não passa de um e estão muitas vezes ligados com a descrição de problemas no software (LIN; ROBLES; SEREBRENIK, 2017).

Além disso, analisando o *turnover* em outra perspectiva, existem também os benefícios do mesmo. O estudo de Foucault et al. (2015) apresenta indícios de que o *turnover* também pode levar projetos ao sucesso. Membros desmotivados ou que causam conflitos dentro da equipe, quando deixam a equipe abrem espaço para novos membros entrarem e também deixam de causar conflitos que fazem membros motivados deixarem o projeto.

As causas do *turnover* em projetos FOSS ainda estão sendo estudadas. O estudo de Foucault et al. (2015) demonstra indícios do *turnover* ocorrendo em situações aonde o colaborador se sente desmotivado. Outro indício de *turnover* é o conflito de interesse relacionados com as decisões do projeto (HOMSCHEID; SCHAARSCHMIDT, 2016). Por outro lado, também temos a insatisfação do contribuidor com o projeto (YU; BENLIAN; HESS, 2012) ou mesmo conflitos internos entre a equipe como motivo da saída de um contribuidor (FILIPPOVA; CHO, 2016).

Como descrito nas seções anteriores, um usuário pode colaborar de diversas formas em um projeto. Seja criando solicitações de modificação para corrigir pequenos problemas ou apenas reportar regularmente *bugs* no software através de *issues*. Portanto, projetos FOSS são uma boa fonte para estudar a rotatividade, tendo exposto várias ca-

racterísticas dos contribuidores para serem analisadas através dos dados do repositório. Além disso, existe um grande número de interação entre esses desenvolvedores o que possibilita criar métricas baseadas em seus relacionados usando análise de redes sociais.

2.4 ANÁLISE QUANTITATIVA

Este estudo aplica métodos que requerem a extração de dados quantitativos do repositório de software. Esses dados são usados como indicadores relacionados ao *turnover* de contribuidores. Dentre as técnicas utilizada para elaboração dos indicadores consideramos a análise de rede sociais e a análise de sentimentos.

A análise de redes sociais possibilitou extrair métricas ligadas ao relacionamento dos contribuidores. Cada uma das métricas é detalhada na Subseção 2.4.1. Por outro lado, a análise de sentimentos, detalhada na Subseção 2.4.2, oferece recursos para mensurar o teor emocional contido nas mensagens de texto trocadas pelos contribuidores.

2.4.1 Análise de Redes Sociais

Uma rede social comumente descrita como um grafo definido como $G = (V, E)$, onde V é um conjunto de indivíduos e E é um conjunto de relações entre os referidos indivíduos. Portanto, uma rede social oferece recursos para descrever a interação entre um grupo de indivíduos e permitir detectar padrões entre essas relações.

Exemplificando a definição apresentada, pense em um grupo de quatro estudantes discutindo a resolução de um problema de matemática. A comunicação entre os estudantes é descrita no Quadro 2.1 e os mesmos são denominados respectivamente de Maria, João, Roberto e Luiz.

Quadro 2.1 – Conversa entre um grupo de estudantes.

Estudante	**Frase**
João	Luiz sua questão de matemática está errada.
Luiz	Eu copiei essa questão do Roberto.
Maria	Roberto você quer ajuda para corrigir a questão?
Roberto	Não, obrigado Maria, já encontrei o erro.
Maria	Luiz você não deve copiar do colega.

Fonte: Autor.

Percebe-se na conversa a menção entre os alunos em cada uma das frases. Partindo da hipótese que é necessário descobrir quem mencionou quem no grupo, o Quadro

2.2 demonstra a ordem da comunicação, qual estudante falou (origem) e a qual estudante se referiu (destino).

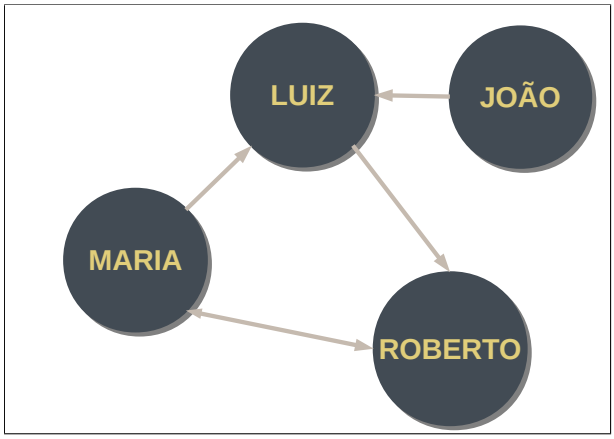
Quadro 2.2 – Descrição da origem e destino da comunicação entre os estudantes.

Ordem	Origem	Destino	Frase
1	João	Luiz	Luiz sua questão de matemática está errada.
2	Luiz	Roberto	Eu copiei essa questão do Roberto.
3	Maria	Roberto	Roberto você quer ajuda para corrigir a questão?
4	Roberto	Maria	Não obrigado Maria, já encontrei o erro.
5	Maria	Luiz	Luiz você não copiar do colega, deixa que eu te ajudo a encontrar o erro.

Fonte: Autor.

Agora com as relações devidamente separadas entre origem e destino é possível criar uma rede social. Essa rede é ilustrada na Figura 2.1. Uma série de métricas podem ser calculadas, refletindo diversos aspectos estruturais e topológicos da rede social, descrevendo o papel de cada vértice na rede.

Figura 2.1 – Rede Social do Grupo de Estudantes.



Fonte: Autor.

As métricas em uma rede social podem ser locais ou da rede como um todo. As métricas locais são baseadas na importância, ou centralidade, de cada vértice. Cada métrica de centralidade baseia-se na busca de diferentes características do vértice na rede. Essas características podem ser a distancia de um vértice para outro, a intermediação entre os vértice ou até mesmo o grau de relacionamento de um vértice com os demais (FREEMAN, 1978). As métricas de rede utilizadas nesse estudos são *betweenness*, *degree*, *closeness*, *eigenvector* e Corenness. Cada uma das centralidades é descrita nas próximas seções.

*2.4.1.1 Centralidade de Intermediação (*Betweeness*)*

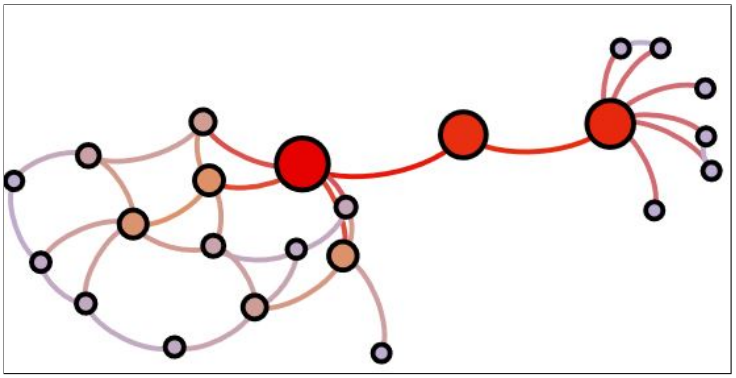
A centralidade de intermediação, em tradução literal da língua Inglesa *betweenness centrality*, é calculada com base no menor caminho entre dois vértice da rede. Portanto, o valor de centralidade da intermediação de um vértice V é a soma de todos os menores caminhos de um conjunto de vértices que se conectam através da passagem por V (FREEMAN, 1977). Em sua definição formal temos:

$$C_B(v) = \sum_{u \neq v \neq w} \frac{\sigma_{uw}(v))}{\sigma_{uw}}, \tag{2.1}$$

onde σ_{uw} representam os menores caminhos entre u e w, e $\sigma_{uw}(v)$ representam os caminhos entre u e w que passam por v.

A Figura 2.2 exemplifica uma rede social com foco na visualização dos valores de centralidade de intermediação. Quanto maior o valor de centralidade de intermediação de um vértice, maior e mais vermelho é o vértice. Logo, caso um contribuidor conecte varias sub-redes, o mesmo terá uma alta centralidade de intermediação.

Figura 2.2 – Exemplo de uma rede com enfase na centralidade de intermediação.



Fonte: Zanetti (2013).

*2.4.1.2 Centralidade de Grau (*Degree*)*

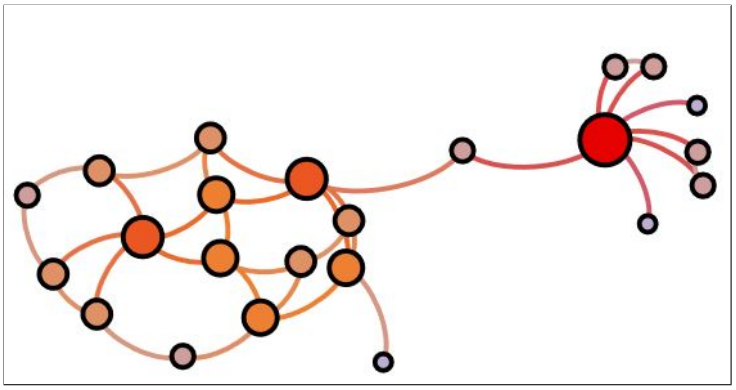
A centralidade de grau, em tradução literal da língua Inglesa *degree centrality*, é calculada com base no número de conexões de um vértice. O valor da centralidade de grau de um vértice v é a soma de todas suas relações com outros vértices (FREEMAN, 1978). Em sua definição formal temos:

$$C_D(v) = \sum_{n}^{j} x_{vj}, \tag{2.2}$$

onde v é o vértice no qual se deseja descobrir a centralidade, N o número total de vértices e X a matriz de adjacência dos vértices (OPSAHL; AGNEESSENS; SKVORETZ, 2010).

A Figura 2.3 exemplifica uma rede social com foco na visualização dos valores de centralidade de grau. Quanto maior o valor de centralidade de grau, maior e mais vermelho é o vértice. Logo, caso um contribuidor tenha um grande número de conexões, o mesmo terá uma alta centralidade de grau.

Figura 2.3 – Exemplo de uma rede com enfase na centralidade de grau.



Fonte: Zanetti (2013).

*2.4.1.3 Centralidade de Proximidade (*Closeness*)*

A centralidade de proximidade, em tradução literal da língua Inglesa *closeness centrality*, é calculada com base nos caminhos entre dois vértice e existem diferentes vertentes dessa centralidade. O calculo de centralidade mais natural baseado em proximidade segundo (FREEMAN, 1978) é o proposto por (SABIDUSSI, 1966).

O valor de centralidade de proximidade de um vértice V é calculado com base no crescimento da distância entre os vértices. Em outras palavras, é uma medida de descentralização ou centralização inversa, quanto maior o valor da centralidade, menor é a proximidade do vértice com os demais. Em sua definição formal temos:

$$C_C(v)^{-1} = \sum_{n}^{w=1} d(P_w, P_v), \tag{2.3}$$

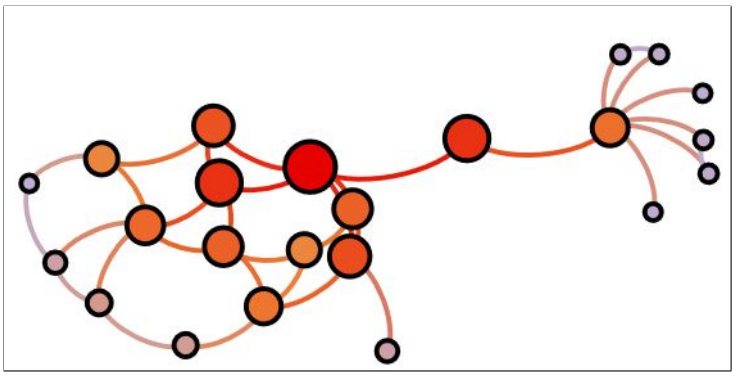
onde $d(w, v)$ é o número de arestas ligando w a v.

Os vértices centrais desta métrica apresentam um custo ou tempo mínimo para se comunicar com os demais vértices (FREEMAN, 1978). Considerando um caso onde uma informação qualquer deva ser espalhada por toda rede, o vértice central distribuirá em um

tempo menor a informação e com um menor custo de transmissão.

A Figura 2.4 exemplifica uma rede social com foco na visualização dos valores da centralidade de proximidade. Quanto maior o valor de centralidade de proximidade, maior e mais vermelho é o vértice. Logo, quanto maior a distância de um vértice para os demais, maior será sua centralidade de proximidade.

Figura 2.4 – Exemplo de uma rede com enfase na centralidade de proximidade.



Fonte: Zanetti (2013).

*2.4.1.4 Centralidade de Autovetor (*Eigenvector*)*

A centralidade de autovetor, em tradução literal da língua Inglesa *eigenvector centrality*, é calculada com base no valor de centralidade dos vértices vizinhos. Ou seja, o valor de centralidade de um vértice v é "[...] soma de suas conexões com outras, ponderada por suas centralidades"(BONACICH, 1987). Em sua definição formal temos:

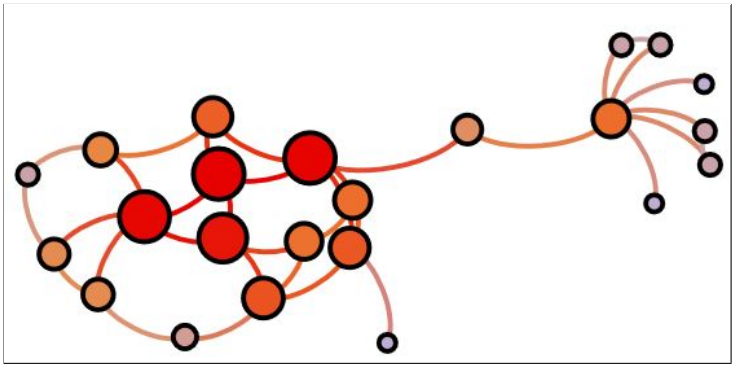
$$\lambda C_E(v) = \sum_j R_{vj} e_j, \tag{2.4}$$

onde R é a matriz de relação entre os vértices, e é a centralidade de um vértice j e λ é a necessidade de um resultado diferente de zero.

O valor de centralidade de autovetor permite descobrir o grau de influência de um vértice em uma rede social. Quanto mais conexões com alto valor de centralidade um vértice está conectado, mais influente este vértice é.

A Figura 2.5 exemplifica uma rede social com foco na visualização dos valores da centralidade de autovetor. Quanto maior o valor da centralidade, maior e mais vermelho é o vértice. Logo, quanto maior a centralidade dos vértices vizinhos, maior será sua centralidade de autovetor.

Figura 2.5 – Exemplo de uma rede com enfase na centralidade de autovetor.



Fonte: Zanetti (2013).

2.4.1.5 Centralidade CORENNESS

A centralidade de CORENNESS, sem tradução literal para o português, é calculada com base no índice K-SHELL dos vértices vizinhos (BAE; KIM, 2014). O índice K-SHELL, é a localização topológica de um vértice em uma rede. Em sua definição formal temos:

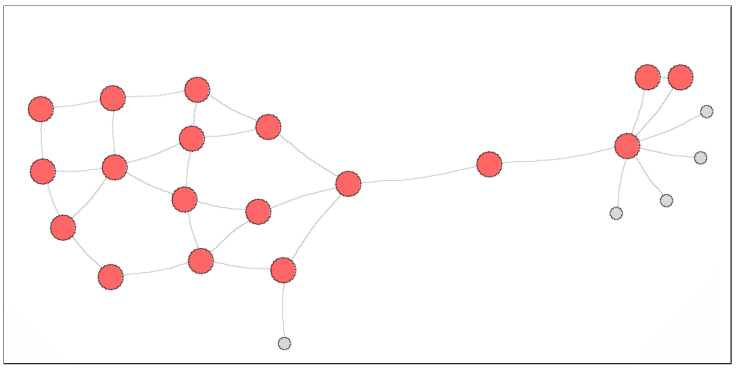
$$C_{NC}(v) = \sum_{w \epsilon N(v)} Ks(w), \tag{2.5}$$

onde $N(v)$ é o conjunto dos vizinhos adjacentes ao vértice v e $ks(w)$ é o índice k-shell do vértice vizinho w.

A centralidade CORENNESS é uma ferramenta poderosa para avaliar a capacidade de disseminação de informações através da rede.

A Figura 2.6 exemplifica uma rede social com foco na visualização dos valores da centralidade CORENNESS. Quanto maior o valor de centralidade de proximidade, maior e mais vermelho é o vértice.

Figura 2.6 – Exemplo de uma rede com enfase na centralidade CORENNESS.



Fonte: Autor.

2.4.2 Análise de Sentimentos de Textos

Existem um conjunto de ferramentas que podem ser usadas para realizar a medição dos sentimentos em textos. Atualmente existem duas ferramentas que oferecem a leitura do sentimento baseado na positividade e negatividade: SENTISTRENGTH e SENTISTRENGTH-SE. SENTISTRENGTH-SE é uma extensão de SENTISTRENGTH e oferece um dicionário específico para medir textos encontrados em engenharia de software. Porém, optamos pela versão original da ferramenta, o SENTISTRENGTH, pelo fato de ser amplamente utilizada em diversos estudos, e bastante conhecido e renomado na comunidade de análise de sentimentos.

SENTISTRENGTH analisa os sentimentos através do texto. Atualmente tem suporte para textos em inglês e outras línguas, como o português contudo a ferramenta foi extensivamente testada para língua Inglesa. O mesmo possui um conjunto de dicionários, onde cada palavra tem seu peso positivo e negativo. Quando inserido um texto na ferramenta, dois valores são dados como resultado, esses valores representam o sentimento daquele texto, frase ou palavra em uma escala de negatividade e positividade. A positividade é representada em uma escala de 1 até 5. A negatividade é representada em uma escala de -1 até -5.

O GITHUB possui vários textos de comunicação entre os contribuidores que podem ser analisados no SENTISTRENGTH. O título e comentários de uma *issue* podem ter sua positividade e negatividade mensuradas. O título e comentários de *pull requests* também. A descrição de *commits* e seus comentários também podem ser analisados.

3 TRABALHOS RELACIONADOS

Neste capítulo é apresentado alguns dos trabalhos relacionados com este estudo. A quantidade encontrada de artigos com objetivo de predizer o evento de *turnover* não é muito grande. Contudo, cada um dos estudos abordam o *turnover* em uma perspectiva diferente.

O estudo de Feeley e Barnett (1997) define modelos que representam condições no qual o *turnover* é mais provável em uma rede de comunicação. Contudo, Feeley e Barnett (1997) não estudam especificamente o *turnover* em projetos de software, mas oferece modelos que servem como motivação para a aplicação da análise de redes sociais neste estudo. Os modelos descritos por Feeley e Barnett (1997) são denominados de modelo de equivalência estrutural, influência social e erosão. Cada modelo representa características de um nó que pode ser suscetível ao *turnover* no trabalho.

O modelo de equivalência estrutural diz que os indivíduos estruturalmente equivalentes em uma rede saem juntos ou em um diferente curto período de tempo. O outro modelo, denominado de influência social descrevem que um indivíduo pode ser influenciado pela saída de outro, ou seja, caso seja informado de sua saída, pode influenciar a saída de outros indivíduos. O modelo de erosão descreve que os indivíduos que estão na periferia da rede são mais propensos a saírem da rede. Isso ocorre pois uma menor quantidade de informação chega a eles, o que o torna-os menos comprometidos. O estudo conclui que existem indícios da relação entre a rede social e o *turnover*. Consequentemente, isto motiva este estudo na utilização de métricas de rede como uma característica do contribuidor.

O *turnover* pode ser visto em uma ótica local, quando analisamos a rotatividade de uma única pessoa ou global quando analisamos um grupo de pessoas, sem considerar características especificas. Os autores Zhu et al. (2017) foca na predição considerando a quantidade de eventos de *turnover* durante o tempo, focando nos dados globais. O mesmo utiliza técnicas de *time series forecasting* (predição de série temporal) para predição da quantidade de colaboradores que deixaram a empresa. O estudo apesar de não estar relacionado especificamente com projetos FOSS, demonstra um outro lado da análise do turnover.

Zhu et al. (2017) colacionam um conjunto técnicas de *time serie forecasting*, apresentando o modelo denominado de *dynamic regression model* (modelo de regressão dinâmica) como a melhor opção para predizer o número total de *turnover*. O estudo usa dados de turnover coletados ao longo de 12 anos em uma companhia.

Os sentimentos também são usados como um meio para criar métodos com o objetivo de prever o evento de *turnover*. O estudo de Garcia, Zanetti e Schweitzer (2013) é o trabalho mais próximo que encontramos de uma predição de *turnover*. No mesmo são

descritos resultados relevantes da relação dos sentimentos com o *turnover* em projetos FOSS. Os dados utilizados pelos autores são do projeto GENTOO. Foram extraídos em torno de 10 anos de informações do projeto. Os dados estão relacionados ao *bug tracker* (Rastreadores de bug) e também ao *mainlist* (Semelhante a um fórum) do projeto.

Os autores Garcia, Zanetti e Schweitzer (2013) conseguem atingir um bom nível de precisão (0,93) para os contribuidores ativos, contudo o mesmo não acontece com os inativos, chegando em torno de 0,19 o melhor resultado. O *recall* tem em torno de 0.65, tanto para contribuidores ativos como inativos.

É importante salientar que o estudo relatado neste documento não tem como objetivo dar sequência ao método proposto por Garcia, Zanetti e Schweitzer (2013) e sim propor um conjunto de classificadores novos e treinados de maneira diferente, apenas se baseando em alguns aspectos do estudo de Garcia, Zanetti e Schweitzer (2013). Logo, com base em alguns artigos revisados sobre o *turnover*, é possível chegar a algumas conclusões sobre as lacunas na área de predição do evento. Atualmente não encontramos nenhum trabalho além do estudo de Garcia, Zanetti e Schweitzer (2013) que tenta predizer o *turnover*, especificamente em projetos FOSS. Grande parte da literatura encontrada está preocupada em encontrar os motivos da saída dos contribuidores do projeto e não predize-lo.

4 METODOLOGIA

Neste capítulo descrevemos quais os passos necessários para chegar ao resultado da nossa pesquisa. Executamos três etapas antes de chegar ao melhor classificador para predição de *turnover*. A primeira etapa é a de extração, no qual demandou a pesquisa e seleção da fonte de dados. A segunda etapa, chamada de análise e seleção, demandou a utilização de técnicas para auxiliar a seleção das métricas. A última etapa, no qual denominamos de treinamento, descrevemos as variáveis preditoras e classes que quando pré processadas, foram usadas como entrada para treinar o classificador. A Figura 4.1 descreve sucintamente cada uma dessas etapas, onde os objetos de cor preta (esquerda) são as atividades e os objetos da cor bege (direita) são os artefatos ou micro atividades produzidas em cada etapa. Após a execução de todas atividades, encontradas dentro de cada uma das etapas, conseguimos como resultado o classificador.

O classificador tem como entrada as principais características da vida de determinado contribuidor no projeto. Ou seja, as interações dentro do repositório feitas pelo contribuidor são usadas como entrada para a construção do classificador. Com o classificador treinado através das características do contribuidor, o mesmo poderá responder a seguinte pergunta: o contribuidor x vai sair do projeto? Uma resposta verdadeira, significa que existem indícios da saída do contribuidor. Porém uma resposta negativa, indica que as informações atuais não apresentam qualquer indício de um *turnover* naquele momento.

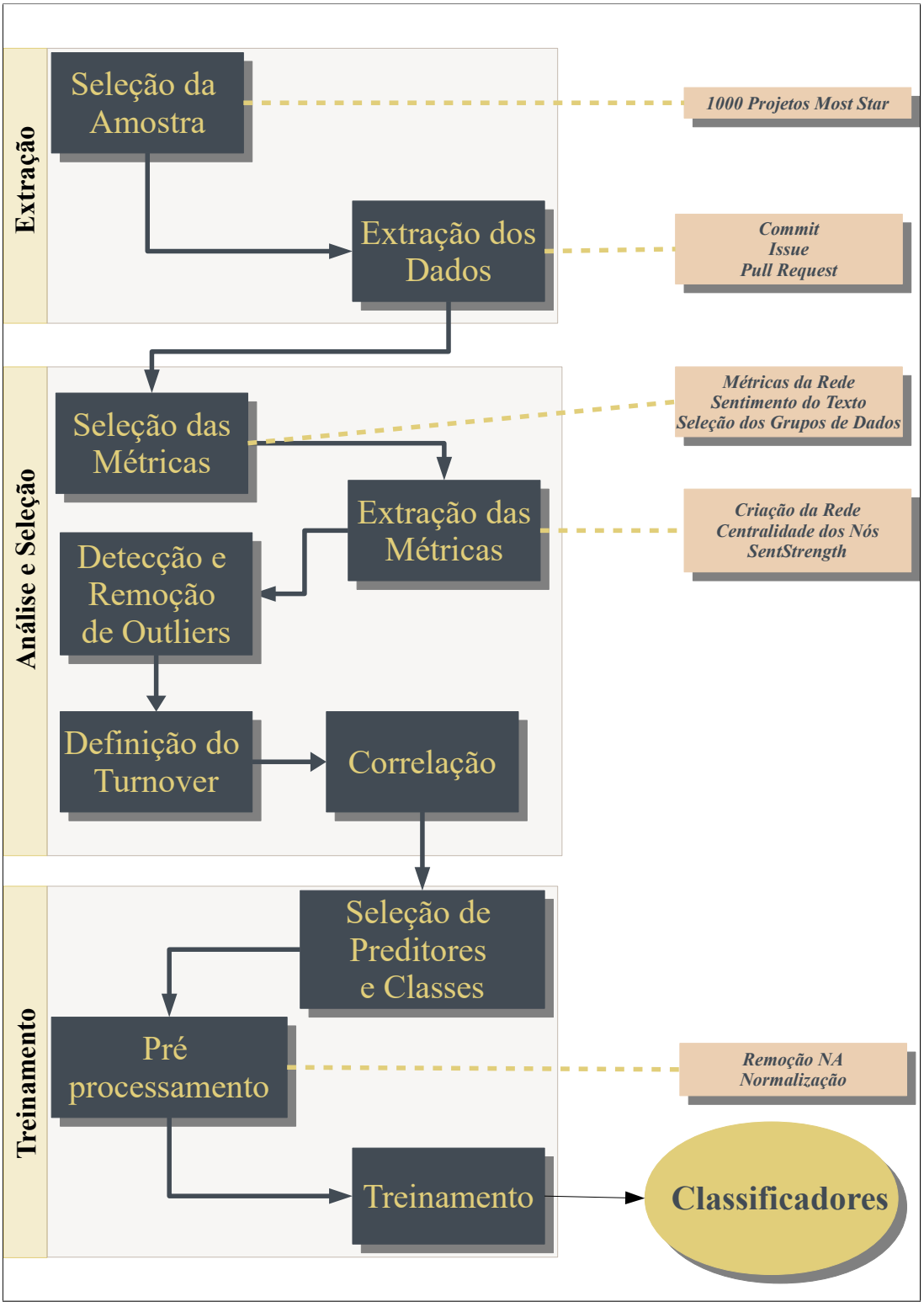
Usamos algoritmos de aprendizado de máquina para criar o classificador. Nós executamos um conjunto de algoritmos de aprendizagem para analisar qual o melhor resultado. Dentre os algoritmos executados estão: NAIVE BAYES, SVM, ADABOOST e TREE DECISION. Esse conjunto de algoritmos foi selecionado com base em sua facilidade de execução e sua precisão quando aplicados a criação de classificadores. Além disso, foi executado também um conjunto de algoritmos de UNDERSAMPLING e OVERSAMPLING para gerar a base de treinamento. Isso ocorreu pois nosso problema apresenta uma base não balanceada. A quantidade de exemplos de *turnover* é bem maior que a quantidade de exemplos de pessoas que continuaram no projeto.

Nas próximas seções, descrevemos detalhadamente cada etapa realizada em nosso trabalho, demonstrando todas atividades executadas que possibilitaram a criação do classificador de *turnover*.

4.1 SELEÇÃO DA AMOSTRA

Os repositórios de software foram selecionados com base no seu número de estrelas (*most star*). Esta técnica de seleção é baseada no estudo de (COELHO; VALENTE,

Figura 4.1 – Diagrama de processo para criação dos classificadores que realizam a predição do *turnover*.



Fonte: Autor.

2017), no qual também tem como critério a seleção da amostra considerando o número de estrelas. No GITHUB as estrelas são atribuídas aos repositórios pelos usuários. Neste es-

tudo tomamos como pressuposto que cada repositório é um projeto. Comumente, quanto mais estrelas um repositório possuir, mais popular entre os usuários o mesmo será. Os repositórios com mais estrelas podem ser encontrados na página de pesquisa do GITHUB usando o critério de ordenação como *most star*.

Utilizando a página de pesquisa do GITHUB, a seleção foi feita em ordem decrescente. O resultado da busca continha um total de 1000 repositórios. Contudo, nem todos repositórios foram selecionados, ou seja, repositórios sem características de um projeto de software foram excluídos do conjunto amostral. Dentre a lista de excluídos estão: listas de exemplos de código fonte, tutoriais, livros e repositórios com textos que não foram escritos em inglês. A lista de repositórios selecionados [1] e excluídos[2] da seleção estão disponíveis para *download*. Obtemos um número total de 758 projetos selecionados.

A Figura 4.2 demonstra a distribuição do número de registros de *pull request*, *issue* e *commit* dos projetos selecionados. Quando a extremidade inferior do gráfico está mais achatada, existe uma maior densidade de registros com valores mais próximos a zero. Notamos no gráfico uma distribuição mais homogênea entre o número de *commits* por repositório. Contudo, a extremidade inferior do número de *issues* e *pull requests* por repositório está mais achatada, ou seja, tem uma distribuição menor do número de registros em *issues* e menor ainda em *pull requests*. Isso acontece pois um certo número de repositórios tem como atividade apenas o uso de *commits*. Ou seja, os número total de *issues* ou *pull requests* estão zerados, levando a uma porção maior na parte inferior do gráfico.

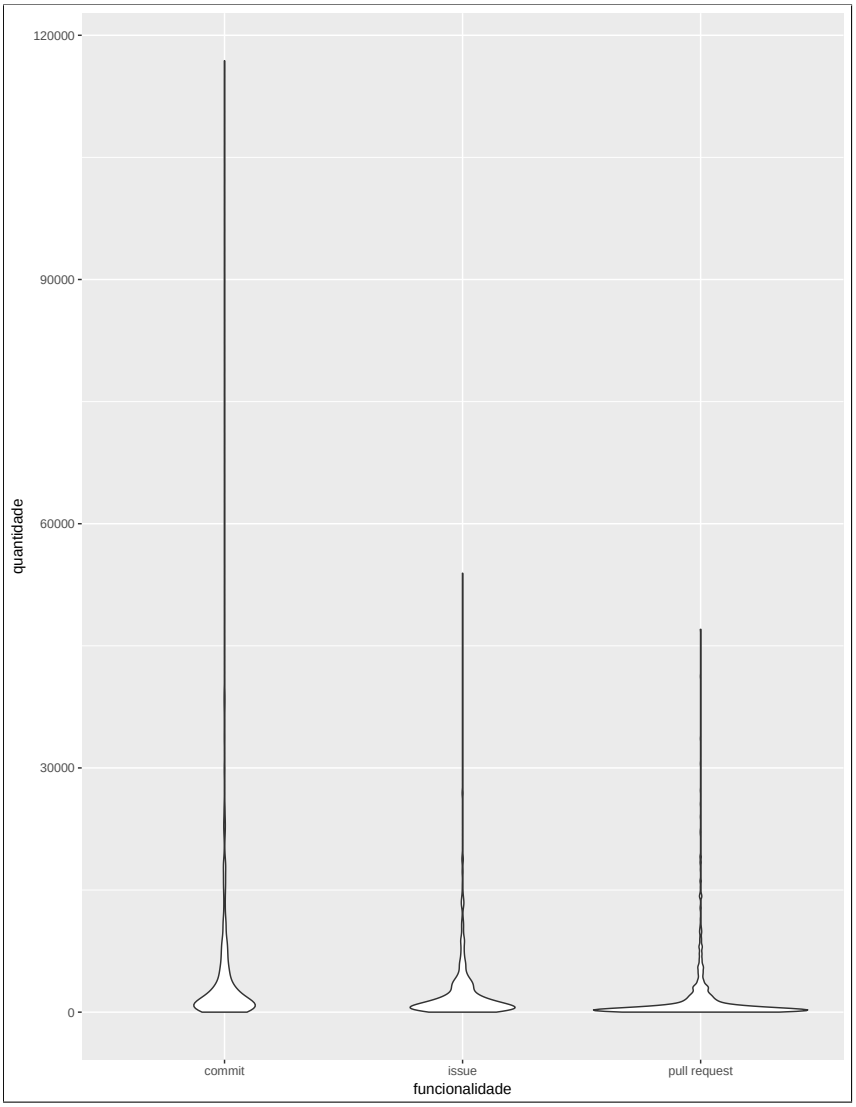
4.2 EXTRAÇÃO DOS DADOS

O GITHUB oferece uma interface, ou também chamada de *Application Programming Interface (API)*, para acessar informações de seus repositórios. Atualmente a mesma está em sua versão 4 [3].

Exemplificando como é feita a requisição para o GITHUB, imagine que na versão 3 seria necessário acessar as *issues* de um repositório. Para isso era necessário acessar um endereço específico e enviar alguns parâmetros, realizando a autenticação. Contudo, a versão 3 enviava como resposta toda a estrutura da *issue*. Logo, aumentava o tempo consideravelmente para extrair todo um repositório. Em contraponto, a versão 4 possibilitava escolher apenas as informações necessárias de uma *issue* para este estudo. Portanto, até metade desse trabalho, estávamos usando a versão 3 da API do GITHUB e posteriormente alteramos a ferramenta para usar a versão 4.

[1] Projetos Selecionados: <http://bit.ly/2SGRq8F>
[2] Projetos Excluídos: <http://bit.ly/2NJMSh1>
[3] Documentação da API V4 do GITHUB: <https://developer.github.com/v4/>

Figura 4.2 – Distribuição do número de *issue*, *pull request* e *commit* dos projetos selecionados.



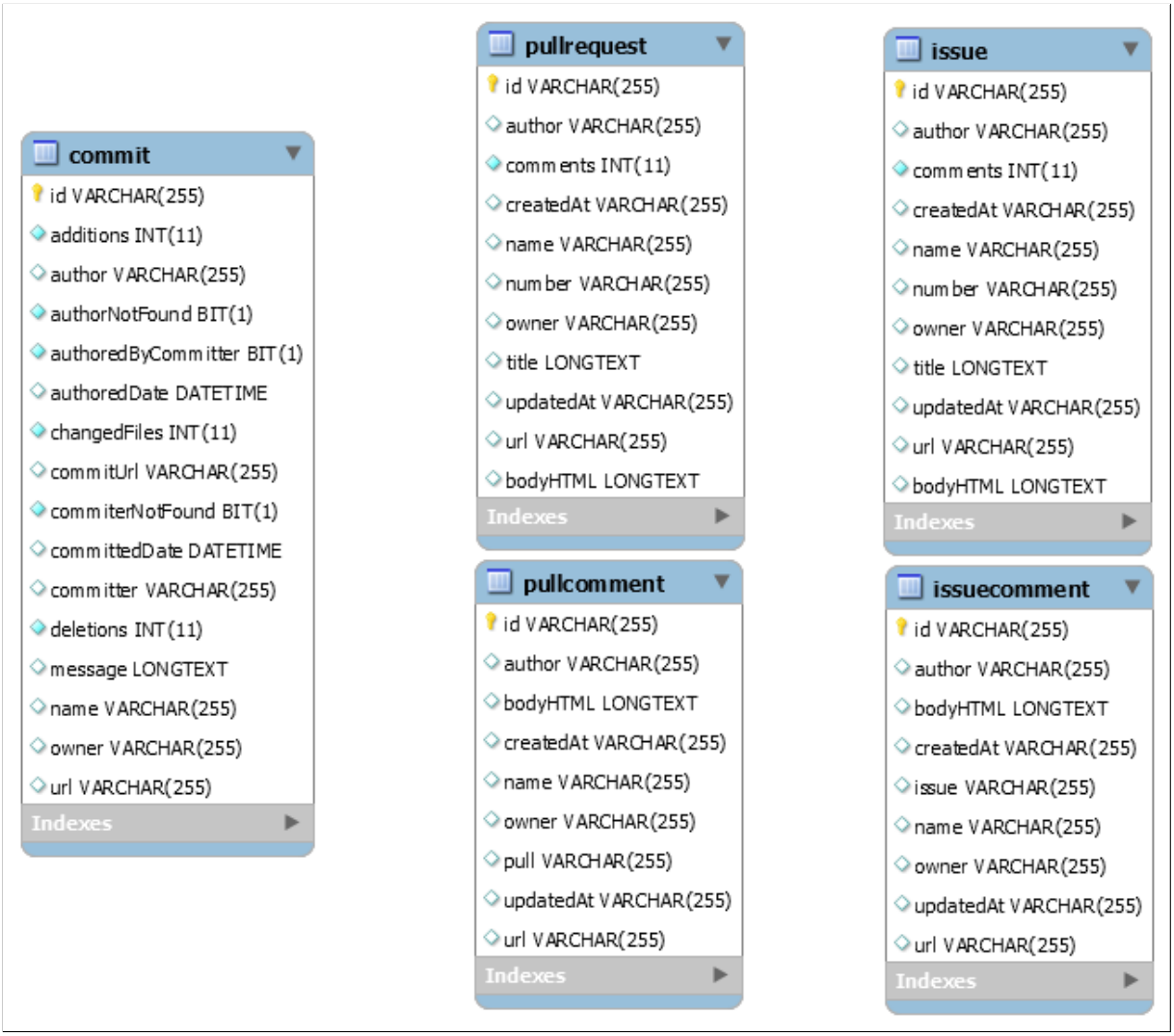
Fonte: Autor.

Nós criamos uma ferramenta[4] usando a linguagem de programação JAVA. Essa ferramenta consome os serviços oferecidos da versão 4 da API do GITHUB. Com o uso da ferramenta, extraímos apenas as informações necessárias para esse estudo. Atualmente este estudo extraiu em torno de 27 gigabytes de informações do GITHUB. Dentre as informações estão: *issues*, *commits*, comentários de *issues* e comentários de *pull requests*. O detalhe das informações extraídas são apresentadas na Figura 4.3 em forma de tabelas do banco de dados.

É importante salientar a quantidade de informações extraídas dos 758 projetos. Estamos utilizando o banco de dados MYSQL para armazenar os dados. A tabela de *commit* possui em torno de 4 milhões e 500 mil registros. A tabela de *issue* possui 1 milhão

[4]Repositório da ferramenta de extração: <https://github.com/DouglasGiordano/repository_analisys>

Figura 4.3 – Tabelas com as informações extraídas do GITHUB.



Fonte: Autor.

e 900 mil registros. E a tabela de *pull request* possui 1 milhão e 700 mil registros registros. Por fim, as tabelas de comentários de *issue* e *pull request* possuem respectivamente em torno de 9 milhões e 5 milhões e 500 mil registros. Todos esses registros foram utilizados como entrada para a geração das métricas.

4.3 SELEÇÃO DAS MÉTRICAS

Um conjunto de variáveis técnicas ou organizacionais pode motivar a saída de um contribuidor na equipe de software. Como visto na seção de *turnover* de software, a insatisfação ou mesmo o conflito pode levar a consequências como o *turnover*. Tomando como partida os indícios apresentados pelas pesquisas citadas anteriormente, nosso primeiro

objetivo foi selecionar um método para mensurar quais são os sentimentos do contribuidor em relação ao projeto de software. Contudo, temos um limite de informações disponíveis sobre os contribuidores no repositório. Isso nos levou ao estudo de (GARCIA; ZANETTI; SCHWEITZER, 2013), no qual vinculava o *turnover* de software ao sentimento encontrado nos textos escritos por contribuidores. Logo, nossa primeira métrica definida foi a emoção expressa na forma de texto. O texto escrito por um contribuidor pode ser encontrado em qualquer comentário criado dentro de uma *issue* ou *pull request* no GITHUB.

Outra variável importante dentro do projeto é a relação do contribuidor com outras pessoas. Atualmente existem ferramentas poderosas para analisar isso, como é o exemplo da análise de redes sociais. Por meio de uma rede social, podemos extrair métricas de centralidade e analisar várias perspectivas do relacionamento do contribuidor com os demais. As métricas escolhidas para este estudo são *betweenness*, *degree*, *closeness*, *eigenvector* e *corenness*.

A lista abaixo apresenta detalhadamente cada uma das métricas com a adição de algumas variações. Para as métricas de relação, a única variação é a contida na métrica *degree*. Usamos a *degree in*, *out* e total para descrever o grau de conexão do contribuidor. Para os sentimentos, além da positividade e negatividade de um texto escrito, também realizamos o cálculo para os textos recebidos. Um texto recebido é qualquer texto que menciona o contribuidor. Além das métricas de sentimentos e de relação, foram adicionados pequenas características do contribuidor no projeto, como o número de interações, os intervalos sem interagir e também o número de dias ativos. Quando falamos de dias ativos, nos referimos a soma de dias contando a partir da primeira interação até a sua última interação no projeto.

- *betweenness*
- *degree in*, *out* e *total*
- *closeness*
- *eigenvector*
- *coreness*
- positividade do texto
- negatividade do texto
- negatividade de textos recebidos
- positividade de textos recebidos
- número de interações

- intervalo médio sem interação no projeto
- dias ativo no projeto

4.4 EXTRAÇÃO DAS MÉTRICAS

A extração de cada métrica seguiu um conjunto de passos distintos. As métricas de rede apenas poderiam ser mensuradas a partir da construção de uma rede social baseada na comunicação dos membros do repositório. Por outro lado, era necessário pré processar os textos antes de mensurar os sentimentos, pois existiam ruídos dentro do texto (trechos de código-fonte) que atrapalhavam o algoritmo responsável pelos sentimentos.

Além disso, todas as métricas apresentadas nas próximas seções foram geradas conforme o período de interação. Ou seja, estudamos um grupo de métricas baseado em toda a vida do contribuidor no projeto e outro grupo baseado apenas no período de 1 mês anterior a última interação. Contudo, não existe nenhuma mudança na forma de calcular as métricas além do filtro por período entre os dois grupos de dados.

Nós apresentamos nas próximas seções detalhes da criação das métricas. Na Subseção 4.4.1 descrevemos as atividades relacionadas com a criação das métricas da rede social de contribuidores. Na Subseção 4.4.2 o pré processamento dos textos e a análise dos sentimentos é detalhada. Por fim na Subseção 4.4.3 é descrito como foram criadas as métricas mais simples de cada um dos contribuidores.

4.4.1 Extração de Métricas da Rede Social

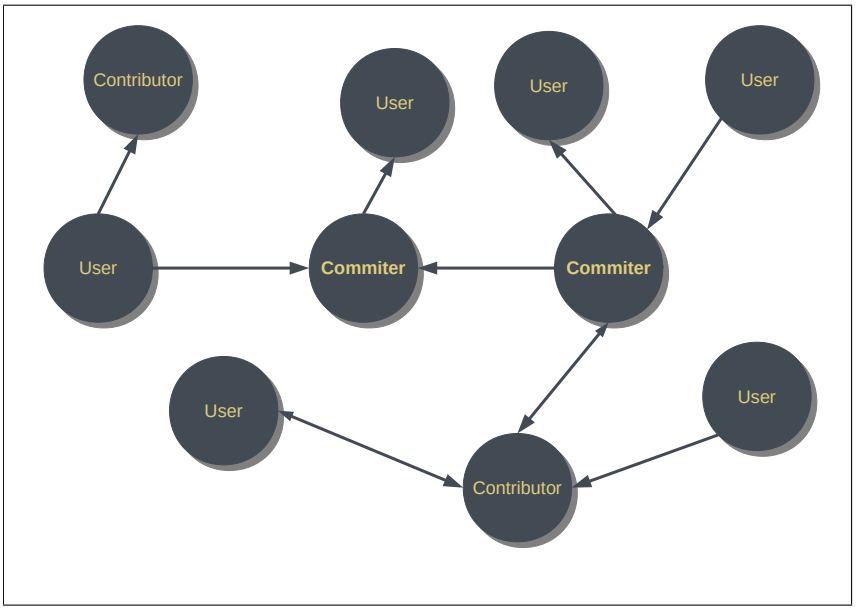
4.4.1.1 Criação da Rede Social

Antes de realizar o calculo das métricas sociais do repositório, é necessário definir como a rede social vai ser construída. O local central de comunicação entre os contribuidores é comumente *issue* e *pull request*. Nesses dois locais são discutidas novas funcionalidades, *bugs*, novas edições de código no repositório dentre outras questões, ou seja, contribuidores com diferentes responsabilidades se relacionam.

Para construir a rede social, este trabalho se baseou na troca de mensagens entre os contribuidores. Toda atividade relacionada a troca de mensagens realizada dentro de uma *issue* ou *pull request* alimenta a construção da rede social. De forma resumida as mensagens são as arestas da rede e os vértices são representadas pelos contribuidores do projeto.

A Figura 4.4 apresenta a ideia geral da rede social baseada na troca de mensagens. Os nós denominados de *user*, *commiter* e *contributer* são pessoas com diferentes responsabilidades no repositório, e as setas direcionadas são as relações. Quando uma mensagem é criada, pode existir a menção entre os esses contribuidore. A menção é a base da relação.

Figura 4.4 – Representação abstrata de uma rede social de contribuidores e suas relações.



Fonte: Autor.

Existem dois casos aonde o usuário menciona outro em um comentário. O primeiro caso ocorre diretamente, ou seja, o usuário escreve especificamente o nome de outro usuário em seu texto. O segundo caso acontece quando o contribuidor apenas responde um comentário anterior, sem mencionar especificamente o nome do outro usuário e sim o texto no qual o mesmo escreveu. Os dois casos podem ser reconhecidos a partir do processamento do texto. Para facilitar a identificação das menções este trabalho denomina o primeiro caso como **menção direta** e o segundo caso como **menção indireta**.

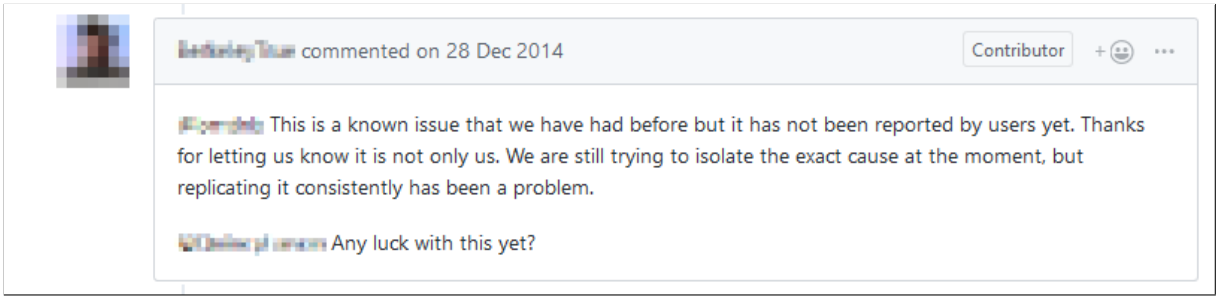
A *issue* 4 do projeto FREECODECAMP [5] é um exemplo real de dois tipos de menções entre os contribuidores. Na Figura 4.5 o usuário "@BT" menciona outro usuário denominado de "@QL", representando a menção direta entre colaboradores. Posteriormente nos demais comentários da *issue* podemos encontrar uma menção indireta onde um usuário do tipo *ghost* [6] responde outro comentário. O comentário da menção indireta é apresentado na Figura 4.6.

Alguns contribuidores foram removidos da análise de rede. Usuários desativados que são denominados *ghost* em *issues* e *pull requests* não possuem qualquer distinção.

[5] *Issue* 4:<https://github.com/freeCodeCamp/freeCodeCamp/issues/4>

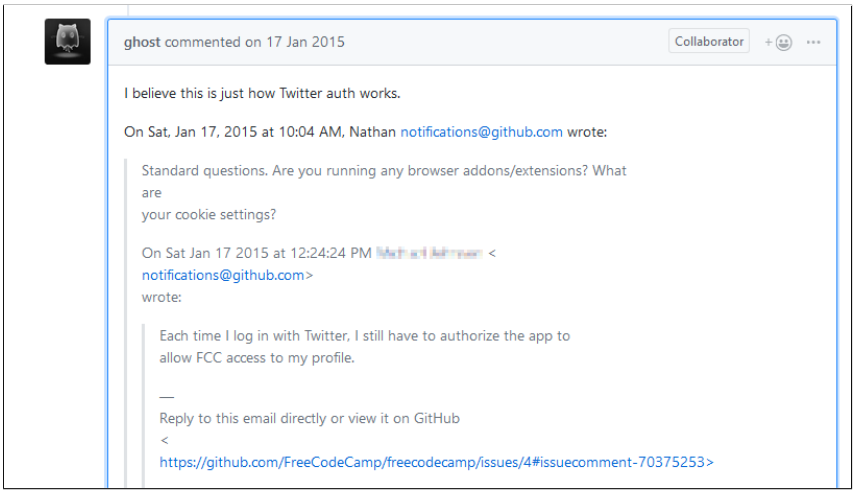
[6] Os usuários *ghost* (tradução livre: fantasma) são usuários desativados no GITHUB, e são denominados de *ghost* para arquivar as comunicações do usuário

Figura 4.5 – Menção direta feita por um usuário no GITHUB.



Fonte: Autor.

Figura 4.6 – Usuário respondendo um comentário.



Fonte: Autor.

No caso de *commits* os usuários desativados continuam com o nome em seus *commits*. Entretanto existe a impossibilidade de relacionar o mesmo usuário que criou um *commit* com o contribuidor que criou uma *issue* ou *pull request*. Portanto, não existe a possibilidade de calcular corretamente o número de interações de um usuários ghost. A Figura 4.6 apresenta um caso onde o usuário é do tipo ghost.

4.4.1.2 Centralidade dos Contribuidores

As métricas de centralidade dos contribuidores foram calculadas usando a linguagem R. Neste caso, usamos a biblioteca para redes denominada de IGRAPH (CSARDI; NEPUSZ et al., 2006). Esta biblioteca oferece um conjunto de algoritmos que auxiliam na análise da rede. Dentre esses algoritmos estão o cálculo de centralidade dos nós *betweenness*, *degree*, *closeness*, *eigenvector* e *corenness*.

Em primeiro lugar, criamos um objeto do tipo grafo na linguagem R, com o algo-

rítimo apresentado na Figura 4.7. Esse grafo serve como base para calcular cada uma das métricas. Optamos por utilizar um grafo direcionado, ou seja, quem mencionar outra pessoa será a origem da relação e quem for mencionado é o destino. Ao utilizar um grafo direcionado, conseguimos saber, por exemplo, quem mencionou mais pessoas no projeto. Caso tivéssemos optado por um grafo não direcionado, essa informação seria suprimida e as métricas que dependem da direção do grafo não estariam corretas para este estudo. As variações da centralidade *degree in*, *out* e total são um exemplo de métricas que diferem entre grafos direcionados e grafos não direcionados.

Figura 4.7 – Criação do grafo através da estrutura de origem e destino de contribuidores.

```
    graph = graph.data.frame(edges, directed = T)
```

Fonte: Autor.

Em contraste com as demais métricas de centralidade, não conseguimos executar a função de centralidade eigenvetor usando um grafo direcionado. Visto que, quando escolhida a opção para realizar o cálculo com um grafo direcionado, a biblioteca ARPACK (utilizada dentro da biblioteca IGRAPH) apresentava alguns erros. O erro ocorria com grafos considerados grandes (milhões de vértices), pois o algoritmo atingia o número máximo de interações para descobrir a centralidade de cada um dos nós. Portanto, optamos por usar o calculo com um grafo não direcionado, o que resolveu o erro.

Exemplificando como conseguimos cada uma das métricas, a Figura 4.8 demonstra o algorítimo das respectivas funções disponíveis no IGRAPH para calcular cada tipo de centralidade.

Figura 4.8 – Funções da biblioteca IGRAPH para extração de métricas de centralidade dos nós da rede social.

```
#betweness
betweenness = centr_betw(graph, directed=T, normalized=T)$res
#degreee
in.degree = centr_degree(graph, mode=c("out"))$res
out.degree = centr_degree(graph, mode=c("in"))$res
total.degree = centr_degree(graph, mode=c("total"))$res
#closeness
closeness = centr_clo(graph, mode="all", normalized=T)$res
#Eigenvector
eigenvector = centr_eigen(graph, directed=F,
normalized=T)$vector
#coreness
coreness = graph.coreness(graph, mode="all")
```

Fonte: Autor.

Cada função retorna uma lista com o conjunto de métricas de todo o grafo. Por exemplo, se passamos um grafo com 5 nós para a função, temos como retorno 5 linhas.

Cada linha tem o respectivo valor de centralidade daquele nó referente ao algoritmo utilizado. Com as métricas calculadas, armazenamos cada registro no banco de dados, relacionando cada métrica a seu respectivo contribuidor em determinado projeto. Logo, existe uma única métrica para cada contribuidor condizente ao tempo de contribuição em cada projeto. O Quadro 4.1 demonstra as métricas de 3 contribuidores e como os dados são armazenados.

Quadro 4.1 – Métricas de centralidade geradas para 3 contribuidores do projeto RECOMPOSE.

Usuário	**Betw.**[a]	**De. In**[b]	**De. Out**[c]	**De. Total**[d]	**Closeness**	**Eigen.** [e]	**Coren.**[f]
X	0.4214	18	11	29	0.0521	0.0062	16
Y	24710	322	386	708	0.0532	0.2068	366
Z	437	7	3	10	0.0523	0.003983	9

Fonte: Autor.

[a]Betweenness
[b]Degree In
[c]Degree Out
[d]Degree Total
[e]Eigenvector
[f]Coreness

4.4.2 Extração de Métricas dos Sentimentos

Os sentimentos foram extraídos usando a linguagem de programação JAVA, através da biblioteca denominada de SENTISTRENGTH. Foi realizado o pré processamento no texto antes de sua entrada no algoritmo da biblioteca.

As próximas seções detalham respectivamente o processo de pré processamento do texto e o processo de execução do algoritmo SENTISTRENGTH.

4.4.2.1 Pré Processamento do Texto

Todos os textos extraídos do GITHUB estavam no formato HTML. Logo, o processo de pré processamento foi simplificado, pois códigos fontes estavam demarcados com a *tag* “code”.

Para percorrer o texto em HTML, utilizamos a biblioteca RCURL e também eXtensible Markup Language (XML) na linguagem R. Na Figura 4.9 demonstramos o algoritmo que seleciona as *tags* “p” e “li” e ignoramos a *tag* “code”.

Figura 4.9 – Algoritmo para remoção da *tag* <code> de um documento em HTML.

```
doc = htmlParse(text)
text.html <- xpathSApply(doc, "//p[not(code)]", xmlValue)
text.html <- paste(text.html, xpathSApply(doc,
"//li[not(code)]", xmlValue), collapse = "\n")
```

Fonte: Autor.

Exemplificando o processo, podemos observar na Figura 4.10 o seguinte trecho de um comentário feito no GITHUB em seu formato HTML.

Figura 4.10 – HTML de um comentário criado por um contribuidor.

```
<p>okay, fixed here. One import in a module was wrong.</p>
<p>
<code>
openerp.addons.web.http import Controller, route, request
</code>
</p>
<p>
made it to search http class in \"/openerp/addons/web/\".
Changing it to
</p>
<p>
<code>
from openerp.http import request, route, Controller
</code>
</p>
<p>helped so it searches the class in \"/odoo/addons/web/\"</p>
```

Fonte: Autor.

Utilizando o algoritmo de pré processamento, podemos ter como resultado apenas o texto que interessa para ser analisado como mostrado na Figura 4.11. Porém, como também é possível notar, alguns limites ainda existem e o pequeno trecho com "/openerp/addons/web/" ainda continua na frase. Não encontramos uma forma de remover trechos que estão fora da *tag* "code".

Figura 4.11 – HTML com a tag <code> removida.

```
<p>okay, fixed here. One import in a module was wrong.</p>
<p>
made it to search http class in \"/openerp/addons/web/\".
Changing it to
</p>
<p>helped so it searches the class in \"/odoo/addons/web/\"</p>
```

Fonte: Autor.

4.4.2.2 *Análise de Sentimento do Texto*

SENTISTREGTH já foi usado em estudos para mensurar mensagens de textos em repositórios de software (GUZMAN; AZÓCAR; LI, 2014; RISHI, 2017; DESTEFANIS et al., 2018; GARCIA; ZANETTI; SCHWEITZER, 2013). Portanto, atualmente se mostra uma das ferramentas mais utilizadas para mensurar o que as mensagens de texto expressam em um ambiente de software. Usamos como entrada no SENTISTRENGTH o texto encontrado no título e comentário de *Pull Requests* e *Issues*.

Após pré-processar o texto, criamos um algoritmo para realizar a chamada da biblioteca SENTISTRENGTH na linguagem R. Isso foi necessário pois a biblioteca foi criada em JAVA e apenas pode ser executada na máquina virtual. Portanto, criamos no R um conjunto de instruções com o objetivo de configurar a máquina virtual do JAVA, endereçar o arquivo da biblioteca SentiStrength e inicia-lo para uso. A Figura 4.12 descreve o código usado para a inicialização da biblioteca.

Figura 4.12 – Inicialização da biblioteca SENTISTRENGTH.

```
Sys.setenv(JAVA_HOME='C:\\Program Files\\Java\\jre-10.0.1')
.jinit('.')
.jaddClassPath('input/SentiStrength.jar')
jsentistrength <- .jnew('uk/ac/wlv/sentistrength/SentiStrength')
jType <- .jnew('java/lang/String', "sentidata")
jDataLocation <- .jnew('java/lang/String', "input/data_2011/")
jparams <- .jarray(list(jType, jDataLocation),
  contents.class = "java/lang/String")
.jcall(jsentistrength, 'V', 'initialise', jparams)
```

Fonte: Autor.

O segundo passo foi executar a chamada na biblioteca, enviando como parâmetro o texto a ser analisado. A Figura 4.13 descreve a função utilizada para realizar a análise do texto. O resultado da chamada é uma lista com dois valores. O primeiro valor é um intervalo entre 1 e 5, e representa a positividade do texto. O segundo valor está entre o intervalo de -1 a -5, e representa a negatividade do texto. A análise foi repetida em todos os textos dos 758 repositórios e armazenada no banco de dados.

Figura 4.13 – Enviando texto e recebendo retorno da análise do sentimento.

```
result <- .jcall(jsentistrength, 'Ljava/lang/String;',
  'computeSentimentScores', jtext)
```

Fonte: Autor.

O Quadro 4.2 demonstra algumas frases mensuradas pela biblioteca SENTISTRENGTH. As frases são de alguns contribuidores de projetos que nós extraímos nesses estudo.

O terceiro passo foi a realização da média da positividade e negatividade dos textos criados pelo contribuidor. Os contribuidores que apenas tiveram como interação no reposi-

Quadro 4.2 – Exemplo de frases mensuradas pelo SENTISTRENGTH.

Frase	Positividade	Negatividade
Looks fine to me. Tested in Chrome 20 on Ubuntu 12.04.	3	-1
thank you @menção. that's fantastic!	4	-1
I hate the ugly cmd icon and would love all of them to be the same as cmder	2	-4

Fonte: Autor.

tório a criação de *commits* e não tenham interagido nenhuma vez com as demais funções tem a sua positividade e negatividade os valores de 1 e -1, que são valores neutros dos sentimentos. Além disso, também calculamos o valor de positividade e negatividade de todas mensagens das *issues* e *pull requests* no qual o mesmo participou. Com isso criamos duas métricas chamadas de positividade recebida e negatividade recebida. Essas métricas tentam representar os ambientes que o contribuidor participou. Dessa maneira, um contribuidor pode no mesmo projeto ter participado de apenas discussões saudáveis sem nenhum conflito, como também pode ter participado de discussões que geraram conflitos.

O Quadro 4.3 exemplifica como armazenamos os dados da média de sentimentos de cada um dos contribuidores.

Quadro 4.3 – Análise dos sentimentos gerados para 3 contribuidores do projeto RECOMPOSE FOSS.

Usuário	Média Positivi.	Média Negativi.	Posit. Recebida	Negat. Recebida
x	2.3333	-1.3333	-1.5882	1.5294
y	1.7992	-1.2125	-1.2626	1.7478
z	1.8461	-1.2307	-1.35	1.7

Fonte: Autor.

4.4.3 Extração de Métricas Simples

Além das métricas de rede e dos sentimentos, buscamos métricas relacionadas a periodicidade de interações do contribuidor com o projeto. Cada uma dessas métricas foi selecionada baseada no nosso conhecimento empírico. A periodicidade de dias sem interação, os dias ativos e o número de interações são métricas simples retiradas dos dados de cada um dos contribuidores no projeto.

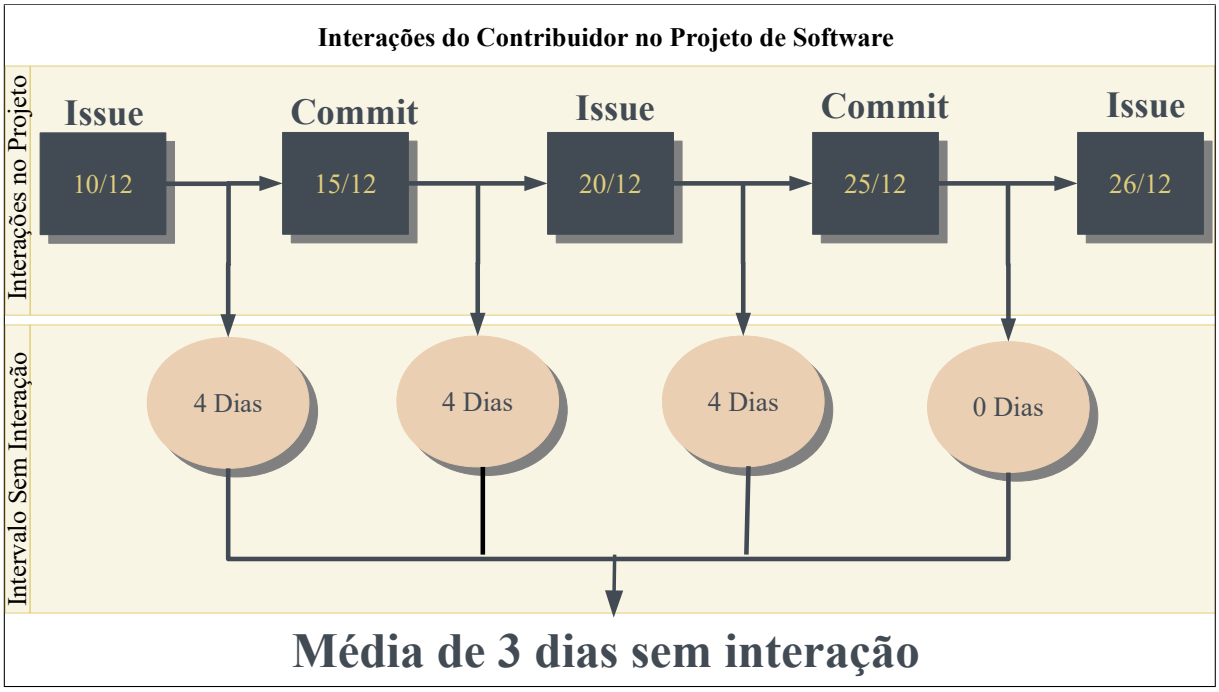
A periodicidade foi calculada com base nas datas encontradas em cada interação. Quanto maior o intervalo sem interação do contribuidor, menor é a periodicidade de interação com repositório. Portanto, através dessa métrica é possível saber se o contribuidor,

por exemplo, interage todo dia ou apenas uma vez ao mês. Para realizar o cálculo foram recuperadas todas as datas de interação do contribuidor. Obtemos através da comparação de cada uma das datas um conjunto de intervalos. O resultado da média do conjunto de intervalos é a periodicidade de dias sem interação do contribuidor.

Exemplificando o cálculo, imagine o conjunto de interações de um contribuidor x. Suas interações são mostradas na Figura 4.14. Os quadrados pretos são todas as datas de interação no repositório e o rótulo acima do quadrado é o tipo de interação. As formas arredondadas são o conjunto de intervalos obtidos do conjunto de datas. No caso da figura, ocorreu interações nos dias 10, 15, 20, 25 e 26 do mês de dezembro. Portanto, a periodicidade de dias sem interagir no repositório do contribuidor é de 3 dias.

Os dias ativos são obtidos com base na primeira e última interação do contribuidor no projeto. Com base na Figura 4.14 o contribuidor x ficou em torno de 16 dias ativos no projeto. Caso não ocorra nenhuma nova interação no projeto, o valor de dias ativos também pode ser considerado como o tempo de vida no projeto. Por fim, o número de interações do contribuidor é calculado simplesmente com o número de vezes que o mesmo interagiu com o repositório. O contribuidor x da Figura 4.14 interagiu 5 vezes em 5 dias diferentes.

Figura 4.14 – Representação gráfica do cálculo da média de dias sem interação de um contribuidor x.



Fonte: Autor.

Descrevemos no Quadro 4.4 as métricas simples de 3 contribuidores com características distintas do projeto RECOMPOSE. Podemos notar que o contribuidor z é bastante ativo, e tem um grande número de interações, possuindo uma média de 1 dia sem interação com o repositório. O contribuidor y interagiu 12 vezes com o repositório em um intervalo

de 3 dias, mostrando que a média do intervalo entre as interações não passou de 1 dia. Por último, o contribuidor x faz pequenas interações em média a cada 150 dias, tendo um total de 4 interações divididas durante 1 ano e alguns meses.

Quadro 4.4 – Métricas simples geradas para 3 contribuidores do projeto RECOMPOSE FOSS.

Usuário	**Média Dias sem Interação**	**Dias Ativo**	**Número de Interações**
x	150.333333333333	451	4
y	0.272727272727273	3	12
z	1.7673343605547	1147	650

Fonte: Autor.

4.5 DETECÇÃO E REMOÇÃO DE *OUTLIERS*

Segundo (HAWKINS, 1980) um *outlier* é "[..] uma observação que se diferencia tanto das demais observações que levanta suspeitas de que aquela observação foi gerada por um mecanismo distinto". Portanto, o primeiro passo é determinar onde está o *outlier* e o segundo passo é analisar o porquê de sua discrepância com relação aos demais dados.

Nós utilizamos um gráfico de distribuição para encontrar os *outliers*. Esse gráfico foi criado com base em variáveis de uma dimensão, ou seja, cada uma das métricas. Tomando como um guia o artigo de (AGUINIS; GOTTFREDSON; JOO, 2013), as técnicas de detecção e remoção de *outliers* devem ser replicáveis. Portanto gráficos são uma técnica simples para encontrar discrepâncias em um conjunto de dados e também são bastante simples de serem entendidas e replicadas.

Porém, é importante salientar que estamos processando um conjunto de dados não balanceado. Ou seja, o número de eventos com *turnover* é muito maior que o número de contribuidores que ficaram no projeto. Consequentemente, grande parte dos dados que podem ser considerados *outliers*, podem ser apenas parte do grupo com menor quantidade.

Para resolver esse problema, utilizamos o gráfico de dispersão apenas como uma ferramenta para encontrar indícios de *outliers*. Cada um dos gráficos utilizados nesse estudo estão descritos no Apêndice A. Nas próximas seções explicamos quais *outliers* foram detectados e as remoções feitas.

4.5.1 Outliers nas Métricas de Rede

A maioria dos *outliers* encontrados em métricas de rede são de contribuidores ativos a muito tempo no projeto. Ou seja, durante sua vida no projeto foi realizada muitas interações com outros membros, ocasionando valores altos de centralidade na rede. Esses valores são tão altos, em alguns casos, que se distanciam dos demais em valores próximos a milhões. Contudo, eliminamos esses casos e adicionalmente no momento de treinar os classificadores, separamos os grupos em sem *outliers* e com *outliers*. Portanto, caso a remoção afete parte do aprendizado, vamos ter o conhecimento.

Além disso, notamos nos gráficos que a média de grande parte das métricas estão próximas a zero. Isso ocorre pois existe uma grande porção de contribuidores que não se relacionam ou se relacionam pouco com os demais. Portanto, existe uma grande quantidade de contribuidores com poucas relações.

Em outro caso, encontramos um grupo de dados que sempre possuía todas métricas iguais a zero. Nossa primeira hipótese ligava os dados a contribuidores que apenas realizavam *commits* no repositório e nunca interagiram com outros membros. Porém, ao aprofundar a pesquisa dentro dos dados, descobrimos que esses usuários são denominados de *ghost* e apenas existem seus e-mails e nomes como referencia a um *commit*. Além disso, as *issues* criadas por usuários que deletaram suas contas apenas são ligadas a um único usuário denominado *ghost*. Com isso desconsideramos esse grupo do conjunto de métricas, pois impossibilitava o relacionamento entre *commits*, *issues* e *pull requests*.

Descrevemos no Quadro 4.5 todas as remoções feitas em cada uma das métricas relacionadas com toda vida do usuário. Por outro lado, no Quadro 4.6 apresentamos todas as remoções feitas nas métricas relacionadas com os últimos 30 dias de vida do contribuidor no projeto. Cada um está separado entre a população considerando todos contribuidores e outra população considerando apenas quem tem mais de 3 interações, o que chamamos de contribuidor casual.

4.5.2 *Outliers* nos Sentimentos

Existe um conjunto de dados que não teve seu sentimento calculado pois apenas interagiram com o repositório através dos *commits*, ou seja, não existia nenhum texto para ser analisado. Todos os contribuidores que não tinham nenhum texto como interação no repositório, tinham suas métricas de sentimento igual a zero. Portanto, todos os dados de contribuidores com o valor 0 estavam incorretos pois o sentimentos positivos estão entre 1 e 5 e os sentimentos negativos entre -1 e -5. Para corrigir este problema, alteramos os valores de 0 para o valor neutro de cada tipo de sentimento (1 positivo e -1 negativo).

Nenhuma outra discrepância foi considerada pois sentimentos altamente positivos

Quadro 4.5 – Quantidade de *outliers* removidos por população referente as métricas de rede.

População	**Métrica**	**Qt. Modificada**	**Qt. Removida**	**Motivo**
Todos	betweenness	0	8	Valor alto
Todos	degree in	0	12	Valor alto
Todos	degree out	0	13	Valor alto
Todos	degree total	0	11	Valor alto
Todos	coreness	0	10	Valor alto
Não Casual	betweenness	0	79	Valor alto
Não Casual	closeness	0	14	Valor alto
Não Casual	degree out	0	267	Valor alto
Não Casual	degree total	0	389	Valor alto
Não Casual	coreness	0	219	Valor alto

Fonte: Autor.

Quadro 4.6 – Quantidade de *outliers* removidos por população referente as métricas de rede da última interação.

População	**Métrica**	**Qt. Modificada**	**Qt. Removida**	**Motivo**
Todos	betweenness	0	27	Valor alto
Todos	degree in	0	12	Valor alto
Todos	degree out	0	12	Valor alto
Todos	degree total	0	16	Valor alto
Todos	coreness	0	26	Valor alto
Não Casual	betweenness	0	17	Valor alto
Não Casual	degree in	0	9	Valor alto
Não Casual	degree out	0	5	Valor alto
Não Casual	degree total	0	10	Valor alto
Não Casual	coreness	0	19	Valor alto

Fonte: Autor.

ou negativos são dados que devem ser consideradas pelo algoritmo preditor. Além disso, diferente das métricas de rede, a discrepância dos valores não entravam na casa dos milhões.

Apresentamos no Quadro 4.7 as modificações feitas nas métricas de sentimentos dos textos considerando toda vida do contribuidor no projeto. Por outro lado, apresentamos o Quadro 4.8 as métricas de sentimentos considerando apenas os últimos 30 dias de vida do contribuidor no projeto.

Quadro 4.7 – Quantidade de *outliers* modificados por população referente as métricas dos sentimentos de textos.

População	Métrica	Qt. Modificada	Qt. Removida	Motivo
Todos	positividade	48243	0	Valor 0
Todos	negatividade	48243	0	Valor 0
Todos	negatividade recebida	141684	0	Valor 0
Todos	positividade recebida	141684	0	Valor 0
Não Casual	positividade	3919	0	Valor 0
Não Casual	negatividade	3919	0	Valor 0
Não Casual	negatividade recebida	8028	0	Valor 0
Não Casual	positividade recebida	8028	0	Valor 0

Fonte: Autor.

Quadro 4.8 – Quantidade de *outliers* modificados por população referente as métricas dos sentimentos de textos da última interação.

População	Métrica	Qt. Modificada	Qt. Removida	Motivo
Todos	positividade	18661	0	Valor 0
Todos	negatividade	18661	0	Valor 0
Todos	negatividade recebida	127766	0	Valor 0
Todos	positividade recebida	127766	0	Valor 0
Não Casual	positividade	3969	0	Valor 0
Não Casual	negatividade	3969	0	Valor 0
Não Casual	negatividade recebida	20307	0	Valor 0
Não Casual	positividade recebida	20307	0	Valor 0

Fonte: Autor.

4.5.3 *Outliers* nas Métricas Simples

As métricas simples apresentaram mais discrepâncias de valores nas métricas que consideram toda vida do contribuidor no projeto. Os valores removidos são descritos no Quadro 4.9. Por outro lado, as métricas simples considerando apenas os últimos 30 dias dos contribuidores, tem um total de apenas duas remoções. As remoções são descritas no Quadro 4.10.

4.6 DEFINIÇÃO DO *TURNOVER* DE SOFTWARE

Para entender o processo de *turnover*, é necessário definir quando ele ocorre. Em uma empresa privada quando um empregado é demitido, fica explícito a sua saída do projeto. Isso não ocorre em um ambiente FOSS. Os contribuidores voluntários não sinalizam sua saída, até podem fazer isso por meio de mensagens para outros usuários, mas a única

Quadro 4.9 – Quantidade de *outliers* removidos por população referente as métricas simples.

População	Métrica	Qt. Modificada	Qt. Removida	Motivo
Todos	Numero de Interações	0	11	Valor Alto
Todos	Intervalo Sem Interagir	0	12	Valor Alto
Todos	Dias Ativo	0	5	Valor Alto
Não Casual	Numero de Interações	0	86	Valor Alto
Não Casual	Intervalo Sem Interagir	0	1	Valor Alto
Não Casual	Dias Ativo	0	4	Valor Alto

Fonte: Autor.

Quadro 4.10 – Quantidade de *outliers* removidos por população referente as métricas simples da última interação.

População	Métrica	Qt. Modificada	Qt. Removida	Motivo
Todos	Numero de Interações	0	2	Valor Alto
Não Casual	Numero de Interações	0	2	Valor Alto

Fonte: Autor.

informação clara que existe da sua saída, é a ausência de interações no projeto.

Para encontrar uma forma dinâmica de definir se o voluntário está ou não ainda no projeto, utilizamos duas abordagens. Cada abordagem considera os intervalos de interação do usuário para detectar o *turnover*. Na Subseção 4.6.1 descrevemos o primeiro método usado, considerando apenas os dias sem interagir e na Subseção 4.6.2 descrevemos o método detectando a mudança de comportamento através da discrepância do ultimo intervalo sem interação.

4.6.1 Número de Dias

Todo usuário que ficou mais de 180 dias sem interagir com o projeto é considerado um *turnover*. O número de dias não foi baseado em nenhum estudo, então definimos um valor de seis meses entre a última interação do contribuidor e a última atividade do projeto. Essa abordagem não consegue levar em consideração o padrão de interação do contribuidor, se tornando uma medida não muito confiável da ocorrência *turnover*. Porém, existe um grande número de indivíduos com poucas interações, que possuem igual ou menos de três interações no projeto. Esse número pequeno de interações impossibilita encontrar um padrão de tempo nas interações do contribuidor no projeto. Portanto, não existe possibilidade de usar o segundo método para analisar o *turnover*. Nesses casos, considerar um valor em torno de seis meses é a única forma de definir a ocorrência do *turnover*.

4.6.2 Mudança de Comportamento

Durante a vida de um contribuidor no projeto, ele tem um conjunto de interações. Essas interações tem intervalos de tempo entre elas. Alguns indivíduos por exemplo interagem com o projeto a cada seis meses, outros a cada três dias. Logo, cada indivíduo vai ter um intervalo padrão de interação no projeto. Contudo, é necessário um método para comparar os dias que o contribuidor está sem realizar atividades no projeto com o seu padrão de interação.

(GRUBBS, 1969) descreve uma forma simples de detectar a mudança de comportamento de um dado em cima de um conjunto de dados. No caso de contribuidores, o dado é os dias sem interação no projeto e o conjunto de dados é o seu histórico de intervalos sem interação. Quando ele deixa de seguir o padrão de contribuição, é um forte indício que o mesmo deixou de contribuir com o projeto.

O cálculo para detectar a mudança de comportamento de (GRUBBS, 1969) consiste em:

$$T = ((x - y))/s \tag{4.1}$$

O valor de x é uma amostra, ou seja, um dado único a ser comparado. No caso dos contribuidores o valor de x é o intervalo entre a sua última interação com projeto e a data atual de atividade do projeto. Exemplificando, o usuário criou um *commit* no dia 20 de dezembro porém hoje é dia 29 de dezembro, então o usuário ficou 9 dias sem interagir com o projeto.

O valor de y é a média das observações. No caso dos contribuidores o y será a média do histórico de intervalos que o mesmo não interagiu com o projeto.

O valor de s é o desvio padrão das observações. Considerando os dados de um contribuidor, seria o desvio padrão do histórico de intervalos.

O resultado do cálculo é comparado com o Anexo A.1. Optamos por utilizar um valor de significância de 5% para este caso. O valor da significância descrito por Grubbs vai depender do número de observações. Portanto, se o contribuidor tem um histórico de 20 interações, é necessário procurar na coluna observações o valor referente a 20 na tabela.

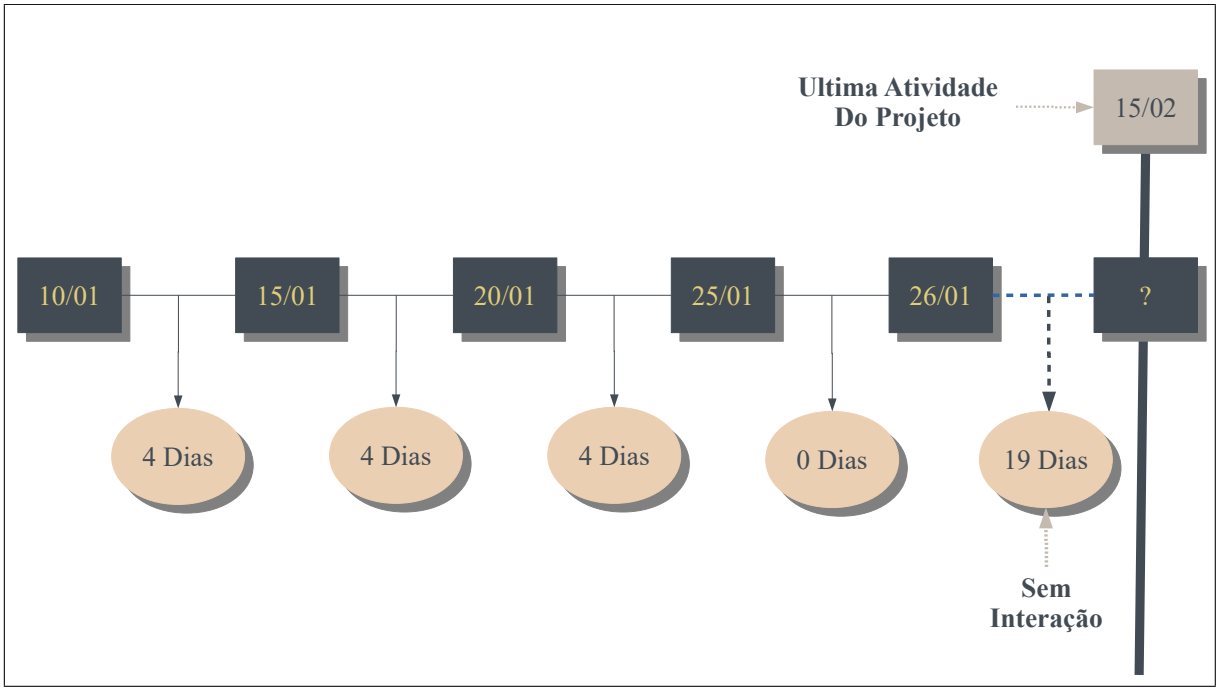
Caso o resultado do cálculo para determinado contribuidor for maior que o valor de significância encontrado na tabela, seu comportamento mudou e ocorreu o evento de *turnover*. Então quanto maior a diferença do valor resultado do cálculo com os valores na tabela, mais provável que aquele contribuidor deixou o projeto.

Algumas considerações são importantes para esta abordagem. Usuários com menos de 3 interações não tem sua significância calculada. Quando usuário tem 2 interações existe apenas um intervalo de tempo, não possibilitando calcular sua média ou desvio padrão para o cálculo de Grubbs (1969). Quando o usuário tem apenas uma interação, não existe histórico suficiente para realizar o cálculo. Quando o usuário possui interações ape-

nas no mesmo dia também não existe nenhum intervalo para ser calculado, consequentemente não tem como detectar a mudança de comportamento. Nestes casos levamos em consideração o valor de 180 dias sem interação para considerar um possível *turnover*.

Exemplificando o método, imagine o cenário onde um contribuidor x realizou interações nos dias 10/01, 15/01, 20/01, 25/01 e 26/01. Como apresentado na Figura 4.15 a ultima atividade registrada em um projeto y foi no dia 15/02. Contudo, o contribuidor não fez nenhuma interação no projeto desde o dia 26/01. O que podemos notar é que faz 19 dias que não existe nenhuma interação do mesmo no repositório.

Figura 4.15 – Exemplo de interações de um contribuidor x considerado *turnover* com base no calculo de (GRUBBS, 1969).



Fonte: Autor.

Aplicando o cálculo de GRUBBS, temos como resultado um valor de 9,2376. Portanto, ao buscar o número de observações do contribuidor no Anexo A.1 (5 interações = 5 observações), encontramos o valor de significância igual a 1,67. Ao comparar esse valor com o resultado de 9,2376, descobrimos que o contribuidor saiu fora do seu comportamento, pois o resultado ideal seria abaixo de 1,67. Consequentemente, é considerado que ocorreu um evento de *turnover* com o contribuidor x.

$$T = ((x - y))/s = ((19 - 3))/1,7320508075689 = 9,2376 \tag{4.2}$$

4.7 CORRELAÇÃO

Um conjunto de variáveis foi proposto por nós para ser utilizado no treinamento do classificador. Dentre essas variáveis estão a centralidade dos contribuidores na rede, os sentimentos dos textos escritos por eles e algumas métricas simples de sua vida no projeto. Contudo, essas variáveis foram definidas com o objetivo de formarem o comportamento do contribuidor. Portanto, é necessário avaliar qual a correlação dessas métricas com a definição do *turnover*. Adicionalmente também é necessário avaliar a relação entre as próprias métricas propostas.

Utilizamos o coeficiente de correlação de Pearson e de Spearman. Os resultados da aplicação do cálculo de coeficiente entre as métricas pode ser encontrado no Apêndice B. Cada gráfico descreve a correlação de um grupo de dados específico, usando um dos dois coeficientes de correlação.

Nenhum grupo de dados em nenhum dos coeficientes mostrou uma correlação precisa entre o *turnover* e as métricas selecionadas. Isso pode ter acontecido por causa do tamanho da amostra em cada grupo de dados. Existe uma grande variabilidade de situações das métricas de cada contribuidor. Contudo, apesar de não existir nenhuma correlação das métricas entre o *turnover*, várias métricas independentes apresentaram um alto índice de semelhança entre si.

4.8 CRIAÇÃO PREDITORES E CLASSES

Nós selecionamos os preditores para os algoritmos de treinamento com base na população e período das métricas. Foram criados quatro grupos de dados com as seguintes configurações:

1. todos contribuidores e todo período de vida no projeto;
2. contribuidores não casuais (maior de 3 interações) e todo período de vida no projeto;
3. todos os contribuidores e considerando apenas os últimos 30 dias de sua vida no projeto;
4. contribuidores casuais e considerando apenas os últimos 30 dias de sua vida no projeto.

Apresentamos no Quadro 4.11 cada um dos grupos de dados e a quantidade de contribuidores em cada um deles.

A decisão de quebrar as métricas geradas em quatro grupos facilita o entendimento do problema em diferentes perspectivas. Além disso, adiciona a possibilidade de entender qual dos grupos tem as informações necessárias para predizer o evento de *turnover*.

Quadro 4.11 – Resumo dos grupos de dados de treinamento.

Grupo de Dados	**População**	**Período de Tempo**	**Quantidade** [a]
Todos	Todos	Tudo	1.832.205
Não Casuais	> 3 Interações	Tudo	536.676
Ultima Interação	Todos	30 Dias	1.802.715
Não Casuais Ultima Interação	> 3 Interações	30 Dias	537.190

Fonte: Autor.

[a]Quantidade com os *outliers* removidos.

A classe definida para predição, é um valor lógico onde é “sim” para o *turnover* e “não” para caso o contribuidor continue no projeto. Ou seja, estamos trabalhando com duas possíveis classes. Portanto, cada arquivo de treinamento possui um conjunto de colunas com as métricas e a última coluna com o valor de uma das duas possíveis classes.

4.9 PRÉ PROCESSAMENTO

Os preditores de cada um dos quatro conjuntos de dados foram escalonados para otimizar o processo de treinamento. Para escalonar os valores utilizamos normalização dos dados. Nenhuma outra técnica foi necessária no pré processamento, pois todos os dados são números reais.

Utilizamos a biblioteca SCIKIT-LEARN [7] para realizar as operações de processamento. Ao utilizar a classe STANDARDSCALER conseguimos normalizar os dados e seguir para a fase de treinamento. A Figura 4.16 descreve parte do código fonte criado para normalizar os dados.

Figura 4.16 – Algoritmo para normalização dos dados.

```
from sklearn.preprocessing import StandardScaler
forecasts = base.loc[:,
 ['betweenness', 'degree_in',
 'degree_out', 'degree_total',
 'closeness', 'eigenvector', 'coreness',
 'mean_positive', 'mean_negative',
 'received_negative', 'received_positive',
 'num_interaction','mean_interval',
 'num_active_days']].values
scaler = StandardScaler()
forecasts = scaler.fit_transform(forecasts)
```

Fonte: Autor.

[7]Documentação da biblioteca SCIKIT-LEARN: <https://scikit-learn.org/stable/documentation.html>

4.10 TREINAMENTO DOS CLASSIFICADORES

O treinamento dos classificadores seguiu a configuração mostrada no Quadro 4.12. Escolhemos inicialmente os algoritmos NAIVE BAYES e TREE DECISION para treinar os modelos preditivos. Posteriormente adicionamos as variações do algoritmo SVM com os seguintes parâmetros de *kernel*: LINEAR e RBF.

Adicionalmente, como estávamos com um conjunto de treinamento não balanceado, escolhemos um algoritmo para aumentar a performance do aprendizado. O algoritmo escolhido foi o ADABOOST e foi executado juntamente com o algoritmo TREE DECISION.

As variações de grupos de dados também foram consideradas, ou seja, alguns classificadores utilizam determinada população e determinado período de vida dos contribuidores no projeto.

Os algoritmos foram executados utilizando a biblioteca SCIKIT-LEARN. Além disso, a separação da amostra em teste e treinamento e as técnicas de OVERSAMPLING e UNDERSAMPLING utilizaram os algoritmos oferecidos pela biblioteca.

4.10.1 Dados de Teste e Treinamento

O SCIKIT-LEARN oferece uma função denominada de "TRAIN TEST SPLIT" [8] que tem como objetivo separar o conjunto de dados em teste e treinamento. A divisão é feita aleatoriamente em cima dos dados ao ser informado a porcentagem do grupo de testes.

Escolhemos 10% dos dados para o grupo de testes. Visto que existe uma grande quantidade de exemplos da classe majoritária, utilizar uma quantidade maior para teste poderia afetar o aprendizado do modelo da classe minoritária. Em contra ponto, poderia ser usado a validação cruzada para testar todos os dados, porém demoraria um tempo muito maior em relação a separação simples, visto a grande quantidade de dados usada como entrada. Na Figura 4.17 é apresentando a função de separação dos dados em treinamento e teste.

Figura 4.17 – Função de separação dos dados em treinamento e teste.

```
self.forecasts_train, self.forecasts_test,
self.class_train, self.class_test =
train_test_split(forecasts_all,classes_all,test_size=0.10)
```

Fonte: Autor.

[8] Documentação da função TRAINT TEST SPLIT: <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html>

Quadro 4.12 – Configuração usada em cada um dos classificadores.

Algoritmo							População		Período	
ID	Adaboost Tree	SVM		Naive Bayes	Tree	Rand. Forest	Todos	Não Casual	Tudo	30 Dias
		linear	rbf							
1	x							x	x	
2	x						x		x	
3	x							x		x
4	x						x			x
5		x						x	x	
6		x						x		x
7			x					x	x	
8			x					x		x
9				x				x	x	
10				x			x		x	
11				x				x		x
12				x			x			x
13					x			x	x	
14					x		x		x	
15					x			x		x
16					x		x			x
17						x		x	x	
18						x	x		x	
19						x		x		x
20						x	x			x

Fonte: Autor.

4.10.2 OVERSAMPLING e UNDERSAMPLING

Existe um conjunto de algoritmos utilizados para equilibrar os dados de treinamento. Esses algoritmos estão categorizados em OVERSAMPLING e UNDERSAMPLING e tem como objetivo respectivamente aumentar o número de amostra da classe minoritária ou diminuir o número de amostras da classe majoritária.

Neste estudo usamos um conjunto de algoritmos de OVERSAMPLING e escolhemos um algoritmo para UNDERSAMPLING. Obtivemos melhores resultados nas técnicas de OVERSAMPLING o que explica a utilização de apenas uma técnica de UNDERSAMPLING. Os algoritmos de OVERSAMPLING utilizados são: BORDERLINE SMOTE, RANDOM OVER SAMPLER e ADASYN. O único algoritmo de UNDERSAMPLING usado foi RANDON UNDER SAMPLER.

Cada um dos algoritmos foi utilizado nas variações descritas no Quadro 4.12.

4.10.3 Medidas de Desempenho

A avaliação dos classificadores foi realizada usando *accuracy* , matriz de confusão, *recall*, *precision* e Matthews Correlation Coefficient (MCC). Tomando como hipótese duas possíveis condições, uma positiva (P) e outra negativa (N). A matriz de confusão tem como resultado o número de positivos verdadeiros (TP), falsos positivos(FP), negativos verdadeiros (TN) e falsos negativos (FN).

A *accuracy* é calculada com base na soma dos positivos e negativos verdadeiros divididos pela soma das duas condições.

$$(TP + TN)/(P + N) \tag{4.3}$$

O *recall* é calculado com base na divisão dos positivos verdadeiros com a soma dos falsos negativos e os positivos verdadeiros.

$$TP/(FN + TP) \tag{4.4}$$

A *precision* é calculada com base na divisão dos positivos verdadeiros com a soma dos falsos positivos e positivos verdadeiros.

$$TP/(FP + TP) \tag{4.5}$$

O MCC é uma medida de qualidade para classificadores binários. O resultado igual a $+1$ representa uma predição sem erros, 0 é considerada uma predição randômica e -1 indica erros em toda predição. Abaixo segue a formula do MCC:

$$MCC = \frac{TP \times TN - FP \times FN}{(TP + FP)(TP + FN)(TN + FP)(TN + FN))} \tag{4.6}$$

5 RESULTADOS

Obtemos resultados satisfatórios tanto na predição de quem fica no repositório (classe 1), quanto relacionada com a predição do evento de turnover (classe 2). Os 3 melhores resultados com o grupo de dados com *outliers* é apresentado na tabela 5.1 e sem *outliers* na tabela 5.2.

O melhor resultado foi encontrado usando o algoritmo SVM com o *kernel* LINEAR, configuração 5, descrita na tabela 4.12. Existe um equilíbrio no acerto de positivos verdadeiros da classe 1 e 2 com respectivos resultados de 83,23% e 81,04 %. A *accuracy* do modelo foi de 78%. A *precision* da classe 1 tem o valor de 0,83, porém a *recall* se mostra baixa com 0,35. Em relação a classe 2, o *precision* e *recall* tem um ótimo desempenho, chegando respectivamente a 0,81 e 0,98.

O segundo melhor resultado foi encontrado usando o algoritmo ADABOOST baseado em TREE DECISION, no qual 89,08 % de acerto com a classe 1 e 76,87 da classe 2. O algoritmo foi executado juntamente com o uso do BORDERLINE SMOTE. A classe 1 mostra novamente uma boa *precision* com o valor de 0,89, porém a baixa *recall* se repete com 0,32. A *precision* da classe 2 chega a 0,77 e a *recall* a 0,98. A remoção dos *outliers* mostrou uma melhora de quase 2 % no acerto da classe 1 no algoritmo ADABOOST.

O algoritmo SVM com *kernel* LINEAR e ADABOOST se mostraram bons algoritmos para a predição do evento de turnover. Contudo ainda é necessário melhorias da medida de *recall* da classe 1, que se mostrou baixa nos dois algoritmos. O uso de OVERSAMPLING foi essencial, pois permitiu ajustar os dados de treinamento para a classe minoritária, ou seja, a classe 1.

O apêndice C detalha os resultados dos classificadores. O Quadro C.1 do apêndice C descreve os resultados com *outliers* e os resultados sem *outliers* são descritos no Quadro C.3. Para detalhes de mais métricas de desempenho como *recall* e *precision* acesse o arquivo de resultados [1].

As próximas seções descrevem detalhes do treinamento dos classificadores com melhor desempenho.

5.1 GRUPO DE DADOS COM *OUTLIERS*

O grupo de dados com *outliers* teve um melhor desempenho usando técnicas de OVERSAMPLING. Situações onde o classificador tinha um péssimo desempenho para a classe 1 foram melhorados usando a técnica BORDERLINE SMOTE. As técnicas de UN-

[1] Resultados completos: <http://bit.ly/2XNTNtY>

DERSAMPLING não proporcionaram nenhuma melhoria no desempenho e em alguns casos deixavam o desempenho do classificador abaixo do valor sem nenhuma técnica adicional. O Quadro 5.1 demonstra as configurações usadas em cada classificador.

Quadro 5.1 – Configuração usada no treinamento dos classificadores com melhores resultados considerando os *outliers*.

Algoritmo	Parâmetros	OVERSAMPLING	Grupo de Dados
13 - TREE DECISION		BORDERLINE SMOTE	População: Não Casual Período: Tudo
1 - ADABOOST	Taxa de Aprendizagem: 0,8 Estimadores: 25	BORDERLINE SMOTE	População: Não Casual Período: Tudo
2 - ADABOOST	Taxa de Aprendizagem: 0,8 Estimadores: 25	BORDERLINE SMOTE	População: Todos Período: Tudo

Fonte: Autor.

Os parâmetros dos classificadores treinados com o algoritmo ADABOOST foram definidos com base em um conjunto de treinamentos. A cada execução foram testadas diferentes taxas de aprendizagem e estimadores, que por fim nos levaram à melhor combinação. A Tabela 5.1 apresenta os melhores desempenhos encontrados nos classificadores treinados com base na medida de *precision* e *recall*.

Tabela 5.1 – Os 3 melhores resultados usando os grupos de dados com *outliers*.

Algoritmo	Classe 1[a]		Classe 2[b]	
	Precision	***Recall***	***Precision***	***Recall***
13 - TREE DECISION	0,92	0,29	0,72	0,99
1 - ADABOOST	0,88	0,31	0,77	0,88
2 - ADABOOST	0,65	0,28	0,76	0,94

Fonte: Autor.

[a]Nenhum Indicio de Turnover
[b]Indício de Turnover

5.2 GRUPO DE DADOS SEM *OUTLIERS*

O grupo de dados após ter seus *outliers* removidos, apresentou melhores resultados em relação ao grupo de dados com *outliers*. Por exemplo, o número de acertos da classe 1

melhorou em torno de 2% sem *outliers* usando o algoritmo ADABOOST. A nossa hipótese é que com a remoção de contribuidores com um longo tempo de vida (valores na casa de milhões em alguamas métricas), possibilitou o algoritmo ajustar os pesos em valores plausíveis a quem fica no projeto, consequentemente aumentando o acerto da classe 1.

As técnicas de balanceamento deste grupo de dados foram as mesmas utilizadas no grupo de dados com *outliers*. Os resultados de um melhor desempenho usando BORDERLINE SMOTE repetiram-se. No Quadro 5.2 é apresentada a configuração usada em cada um dos algorítimos que criaram os melhores classificadores.

Quadro 5.2 – Configuração usada no treinamento dos classificadores com os melhores resultados desconsiderando os *outliers*.

Algoritmo	**Parâmetros**	**OVERSAMPLING**	**Grupo de Dados**
5- SVM	KERNEL: Linear	BORDERLINE SMOTE	População: Não Casual Período: Tudo
1 - ADABOOST	Taxa de Aprendizagem: 0.8 Estimadores: 25	BORDERLINE SMOTE	População: Não Casual Período: Tudo
2 - ADABOOST	Taxa de Aprendizagem: 0.8 Estimadores: 25	BORDERLINE SMOTE	População: Todos Período: Tudo

Fonte: Autor.

A configuração dos parâmetros do algoritmo ADABOOST seguiu a mesma lógica descrita no grupo de dados com *outliers*, foram executados vários treinamentos com diferentes taxas de aprendizado e estimadores e foram selecionadas as melhores combinações. Na Tabela 5.2 é apresentado os classificadores com melhores desempenhos.

Tabela 5.2 – Os 3 melhores resultados usando os grupos de dados sem *outliers*.

Algoritmo	**Classe 1**[a]		**Classe 2**[b]	
	Precision	***Recall***	***Precision***	***Recall***
5 - SVM	0,83	0,35	0,81	0,98
1 - ADABOOST	0,89	0,32	0,77	0,98
2 - ADABOOST	0,63	0,30	0,79	0,94

Fonte: Autor.

[a]Nenhum Indício de Turnover
[b]Indício de Turnover

5.3 VALIDAÇÃO CRUZADA DOS MELHORES RESULTADOS

A validação cruzada possibilita testar todas as partes do nosso conjunto de dados, para que seja possível analisar possíveis grupos de dados que o classificador não tenha um bom desempenho, consequentemente mostrando a capacidade de generalização do nosso modelo. Para executarmos a validação cruzada, usamos o método K-FOLD, que consiste em dividir os nossos dados em conjunto K de partes, onde cada parte em algum momento será usada como dados de testes, e as demais partes como dados de treinamento.

Nas próximas seções serão descritos os resultados encontrados a partir da execução do método K-FOLD nos melhores resultados encontrados nesse estudo, focando no uso de dados sem *outliers* e usando um número de 10 FOLDS.

5.3.1 O Melhor resultado: SVM

O algoritmo SVM, usando o KERNEL Linear, apresentou o melhor resultado com base na amostra de teste aleatória de 10% dos dados. Após usar a validação cruzada, os resultados encontrados na amostra aleatória se mantiveram. O desvio padrão do resultado de *precision* e *recall* teve valores baixos, mostrando que não ocorreu resultados muito distintos entre as partes K-FOLD. Logo, nosso classificador usando SVM obteve bons resultados de generalização. A Tabela 5.3 demostra os resultados obtidos em cada FOLD, adicionalmente mostrando a medida MCC.

Tabela 5.3 – Resultados de desempenho obtidos usando validação cruzada através do método K-FOLD para o classificador gerado com o algoritmo SVM.

K-FOLD	**MCC**	**Classe 1**		**Classe 2**	
		Precision	***Recall***	***Precision***	***Recall***
1-Fold	0,4511	0,8250	0,3466	0,8092	0,9742
2-Fold	0,4575	0,8353	0,3500	0,8082	0,9754
3-Fold	0,4591	0,8352	0,3525	0,8082	0,9751
4-Fold	0,4548	0,8341	0,3466	0,8080	0,9755
Desvio Padrão	0,0035	0,0049	0,0029	0,0006	0,0006
Média	0,4556	0,8324	0,3489	0,8084	0,9751

Fonte: Autor.

5.3.2 O Segundo Melhor Resultado: ADABOOST

O algoritmo ADABOOST com 25 estimadores e 0,8 de taxa de aprendizado, usando como grupo de dados os contribuidores não casuais, manteve seus resultados de desem-

penho obtidos com uma amostra de dados aleatória. O desvio padrão entre as métricas de desempenho foram baixas considerando todos os 10 FOLDS. A maior variação encontrada entre os FOLDS foi na métrica *precision* da classe 1, tendo como maior valor 0,8914 e menor valor 0,8736. Logo, o classificador criado usando o algoritmo AdaBoost com as configurações citadas anteriormente também tem um bom desempenho em sua generalização. A Tabela 5.4 detalha os dados de desempenho obtidos na validação cruzada.

Tabela 5.4 – Resultados de desempenho obtidos usando validação cruzada através do método K-FOLD para o classificador gerado com o algoritmo ADABOOST e o grupo de dados com população apenas não casual.

K-FOLD	**MCC**	**Classe 1**		**Classe 2**	
		Precision	***Recall***	***Precision***	***Recall***
1-Fold	0,4395	0,8819	0,3165	0,7664	0,8762
2-Fold	0,4427	0,8914	0,3170	0,7625	0,8736
3-Fold	0,4318	0,8848	0,3105	0,7543	0,8720
4-Fold	0,4389	0,8847	0,3146	0,7647	0,8762
5-Fold	0,4424	0,8932	0,3162	0,7610	0,8732
6-Fold	0,4347	0,8819	0,3108	0,7641	0,8778
7-Fold	0,4356	0,8736	0,3162	0,7668	0,8767
8-Fold	0,4390	0,8838	0,3150	0,7655	0,8765
9-Fold	0,4358	0,8884	0,3104	0,7600	0,8755
10-Fold	0,4410	0,8868	0,3170	0,7637	0,8744
Desvio Padrão	0,0035	0,0055	0,0027	0,0037	0,0018
Média	0,4381	0,8850	0,3144	0,7629	0,8752

Fonte: Autor.

5.3.3 O Terceiro Melhor Resultado: ADABOOST

O algoritmo ADABOOST, considerando todos os contribuidores na população dos dados, manteve o comportamento dos algoritmos anteriores nos dados de validação cruzada. Logo, o algoritmo ADABOOST com todos os contribuidores também teve um bom desempenho de generalização. Contudo, o resultado da métrica MCC se mostrou abaixo dos demais algoritmos, mostrando que o terceiro melhor resultado está mais próximo de resultados aleatórios. A Tabela 5.5 descreve os dados obtidos na validação cruzada.

Tabela 5.5 – Resultados de desempenho obtidos usando validação cruzada através do método K-FOLD para o classificador gerado com o algoritmo ADABOOST e o grupo de dados com toda a população.

K-FOLD	**MCC**	**Classe 1**		**Classe 2**	
		Precision	***Recall***	***Precision***	***Recall***
1-Fold	0,3117	0,6349	0,2967	0,7816	0,9365
2-Fold	0,2946	0,6823	0,2702	0,7302	0,9402
3-Fold	0,3130	0,6190	0,3021	0,7937	0,9352
4-Fold	0,3205	0,6449	0,2997	0,7849	0,9394
5-Fold	0,3259	0,5988	0,3215	0,8173	0,9337
6-Fold	0,3173	0,6210	0,3061	0,7959	0,9354
7-Fold	0,3189	0,5839	0,3209	0,8197	0,9310
8-Fold	0,3156	0,6062	0,3107	0,8028	0,9329
9-Fold	0,3123	0,6455	0,2940	0,7753	0,9379
10-Fold	0,3112	0,6625	0,2884	0,7623	0,9395
Devio Padrão	0,0083	0,0301	0,0153	0,0266	0,0031
Média	0,3141	0,6299	0,3010	0,7864	0,9362

Fonte: Autor.

6 CONCLUSÃO

Este estudo selecionou algumas características de contribuidores, que podem ser mensuradas através dos dados dos repositórios. Logo, a saída do membro da equipe foi explorada através de quatro perspectivas: relacionamento entre os membros, os sentimentos expressos em mensagens de texto, a frequência de contribuições e o intervalo entre elas. O GITHUB foi a fonte de dados para a extração de cada perspectiva, no qual foram extraídos *commits*, *issues* e *pull requests* de 758 repositórios de software.

O relacionamento entre os membros de cada um dos repositórios foi mensurado através do uso da análise de redes sociais, o que permitiu a extração de um conjunto de métricas de centralidade de cada um dos contribuidores. Os sentimentos dos textos foi analisado usando uma ferramenta denominada de SENTISTRENGTH, o que permitiu extrair a positividade e negatividade dos textos. Além disso, o comportamento de interações possibilitou definir quando ocorria o evento de turnover.

Após obter as métricas, foram criados quatro grupos de dados que consideravam o tempo e a população de contribuidores. Ao usar o conjunto de métricas de cada um dos quatro grupos como entrada nos algoritmos SVM, NAIVE BAYES, TREE DECISION e ADABOOST este estudo encontrou resultados satisfatórios da predição. Adicionalmente, este estudo tem como diferencial a quantidade de contribuidores usados como treinamento, o que mostra uma grande variedade de exemplos como entrada para os classificadores.

Nosso classificador em comparação com o estudo de (GARCIA; ZANETTI; SCHWEITZER, 2013) tem um desempenho melhor na medida *precision*. A diferença é em torno de 0,18 (Este estudo: 0,83; (GARCIA; ZANETTI; SCHWEITZER, 2013): 0,65) com base na predição dos contribuidores que continuam ativo no projeto. No caso da predição dos possíveis contribuidores inativos, o valor tem um desempenho melhor, chegando a 0,16 de diferença (Este estudo: 0,81; (GARCIA; ZANETTI; SCHWEITZER, 2013): 0,65).

Considerando a medida *recall* existe uma inversão no melhor desempenho. (GARCIA; ZANETTI; SCHWEITZER, 2013) tem uma melhor medida *recall* de quem fica no projeto, com uma diferença de 0,58 (Este estudo: 0,35; (GARCIA; ZANETTI; SCHWEITZER, 2013): 0,93). Por outro lado, este estudo tem uma melhor *recall* na predição de quem sai do projeto, com uma diferença em torno de 0,79 (Este estudo: 0,93; (GARCIA; ZANETTI; SCHWEITZER, 2013): 0,19).

Adicionalmente também salientamos que o estudo de (GARCIA; ZANETTI; SCHWEITZER, 2013) aborda apenas um projeto FOSS. Em contra ponto, este trabalho utiliza uma quantidade significativa de dados, que totalizam 758 projetos. Logo, os classificadores propostos em nosso estudo são generalistas, podendo ser aplicados em diferentes repositórios de software.

Outro ponto que é importante salientar, é que o desempenho dos classificadores

diminui para a classe 1 quando o grupo de dados considera usuários casuais. Ou seja, os classificadores propostos nesse estudo não conseguem predizer com uma boa *precision* e *recall* novos usuários que terão uma longa vida no projeto.

Contudo, alguns pontos ficam em aberto para estudos futuros, como a melhoria da métrica *precision* da classe 1, a criação de uma ferramenta que use o melhor classificador para analisar em tempo real a mudança de classificação de um contribuidor entre a classe 1 e classe 2, a utilização de validação cruzada no treinamento e teste das melhores classes. E por fim, procurar novos indicadores que possam ser extraídos do repositório e aumentem o desempenho das medidas de cada classificador.

6.1 CONTRIBUIÇÕES

Este estudo oferece algumas contribuições para a área de Engenharia de Software no estudo do evento de *turnover* em equipe de software livre e de código aberto. Entre essas contribuições estão a predição do evento de *turnover* e a análise da relação do evento com um conjunto de indicadores.

Adicionalmente concede contribuições para a resolução de problemas encontrados em engenharia de software através do uso de aprendizado de máquina, em tradução literal da língua Inglesa *machine learning*. Dentre essas contribuições estão o uso de métodos quantitativos no tratamento de dados empíricos encontrados no repositório de software.

6.2 LIMITES

Este trabalho analisa apenas dados de projetos FOSS do GITHUB. Portanto, para outros casos o comportamento dos classificadores pode ser diferente. Além disso, esse trabalho se baseia em um conjunto de repositórios mais populares do GITHUB (i.e. com mais estrelas), o que também pode modificar o resultado para casos onde os projetos não são populares.

Outro ponto importante é a escolha dos indicadores quantitativos, ou seja, as métricas extraídas dos projetos. As características selecionadas dos contribuidores está limitada a 14 dados quantitativos. Além disso, os dados de número de interações, intervalo sem interagir no projeto e dias ativo no projeto foram propostos de forma empírica. Contudo, ao executar a correlação dos indicadores quantitativos não foi encontrada nenhuma relação direta com o *turnover*. Nossa hipótese é que existe um conjunto de grupos distintos nos dados, fazendo com que o coeficiente de correlação entre as métricas e o *turnover* seja baixo. Adicionalmente, os dados quantitativos podem não ter uma grande relação di-

reta com o *turnover* quando vistos separadamente. O que explica a baixa correlação entre os indicadores individuais e o *turnover*.

Além disso, é importante salientar que o classificador se limita a predizer se existe um indicio da saída do contribuidor com base em todas suas interações no projeto ou apenas considerando os últimos 30 dias. Isto não permite que o gestor saiba da saída do contribuidor com uma grande antecedência. Portanto, pode ocorrer de o gestor conhecer o indicio da saída do contribuidor apenas quando ocorrer a ultima interação do mesmo.

6.3 AMEAÇAS A VALIDADE

Abaixo segue uma lista de possíveis ameaças a validade deste estudo:

- Não foi utilizada nenhuma técnica para eliminação de contribuidores com perfil falso, no qual tinham como objetivo espalhar algum spam dentre os dados do repositório.
- Não foi utilizada validação cruzada em todas execuções e sim apenas nos melhores resultados.
- A definição de turnover, usando como base o método proposto por (GRUBBS, 1969) para detecção de *outliers*, não foi validada estatisticamente.

REFERÊNCIAS BIBLIOGRÁFICAS

AGUINIS, H.; GOTTFREDSON, R. K.; JOO, H. Best-practice recommendations for defining, identifying, and handling outliers. **Organizational Research Methods**, Sage Publications Sage CA: Los Angeles, CA, v. 16, n. 2, p. 270–301, 2013.

BAE, J.; KIM, S. Identifying and ranking influential spreaders in complex networks by neighborhood coreness. **Physica A: Statistical Mechanics and its Applications**, Elsevier, v. 395, p. 549–559, 2014.

BITZER, J.; SCHRETTL, W.; SCHRÖDER, P. J. Intrinsic motivation in open source software development. **Journal of comparative economics**, Elsevier, v. 35, n. 1, p. 160–169, 2007.

BONACICH, P. Power and centrality: A family of measures. **American journal of sociology**, University of Chicago Press, v. 92, n. 5, p. 1170–1182, 1987.

COELHO, J.; VALENTE, M. T. Why modern open source projects fail. In: ACM. **Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering**. [S.l.], 2017. p. 186–196.

CROWSTON, K.; SHAMSHURIN, I. Core-periphery communication and the success of free/libre open source software projects. **Journal of Internet Services and Applications**, Springer, v. 8, n. 1, p. 10, 2017.

CSARDI, G.; NEPUSZ, T. et al. The igraph software package for complex network research. **InterJournal, Complex Systems**, v. 1695, n. 5, p. 1–9, 2006.

DESTEFANIS, G. et al. On measuring affects of github issues' commenters. In: IEEE. **2018 IEEE/ACM 3rd International Workshop on Emotion Awareness in Software Engineering (SEmotion)**. [S.l.], 2018. p. 14–19.

FEELEY, T. H.; BARNETT, G. A. Predicting employee turnover from communication networks. **Human Communication Research**, Oxford University Press, v. 23, n. 3, p. 370–387, 1997.

FILIPPOVA, A.; CHO, H. The effects and antecedents of conflict in free and open source software development. In: ACM. **Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing**. [S.l.], 2016. p. 705–716.

FOUCAULT, M. et al. Impact of developer turnover on quality in open-source software. In: ACM. **Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering**. [S.l.], 2015. p. 829–841.

FREEMAN, L. C. A set of measures of centrality based on betweenness. **Sociometry**, JSTOR, p. 35–41, 1977.

____. Centrality in social networks conceptual clarification. **Social networks**, North-Holland, v. 1, n. 3, p. 215–239, 1978.

GARCIA, D.; ZANETTI, M. S.; SCHWEITZER, F. The role of emotions in contributors activity: A case study on the gentoo community. In: IEEE. **Cloud and green computing (CGC), 2013 third international conference on**. [S.l.], 2013. p. 410–417.

GITHUB. **GitHub Glossary**. 2018. <https://help.github.com/articles/github-glossary/>. Accessed: 2018-06-02.

GRUBBS, F. E. Procedures for detecting outlying observations in samples. **Technometrics**, Taylor & Francis, v. 11, n. 1, p. 1–21, 1969.

GUZMAN, E.; AZÓCAR, D.; LI, Y. Sentiment analysis of commit comments in github: an empirical study. In: ACM. **Proceedings of the 11th Working Conference on Mining Software Repositories**. [S.l.], 2014. p. 352–355.

GUZZI, A. et al. Communication in open source software development mailing lists. In: IEEE PRESS. **Proceedings of the 10th Working Conference on Mining Software Repositories**. [S.l.], 2013. p. 277–286.

HALL, T. et al. The impact of staff turnover on software projects: the importance of understanding what makes software practitioners tick. In: ACM. **Proceedings of the 2008 ACM SIGMIS CPR conference on Computer personnel doctoral consortium and research**. [S.l.], 2008. p. 30–39.

HAWKINS, D. M. **Identification of outliers**. [S.l.]: Springer, 1980. v. 11.

HOMSCHEID, D.; SCHAARSCHMIDT, M. Between organization and community: investigating turnover intention factors of firm-sponsored open source software developers. In: ACM. **Proceedings of the 8th ACM Conference on Web Science**. [S.l.], 2016. p. 336–337.

LAKHANI, K. R.; WOLF, R. G. Why hackers do what they do: Understanding motivation and effort in free/open source software projects. 2003.

LERNER, J.; TIROLE, J. The open source movement: Key research questions. **European economic review**, Elsevier, v. 45, n. 4-6, p. 819–826, 2001.

LI, Y.; TAN, C.-H.; TEO, H.-H. Leadership characteristics and developers' motivation in open source software development. **Information & Management**, Elsevier, v. 49, n. 5, p. 257–267, 2012.

LIN, B.; ROBLES, G.; SEREBRENIK, A. Developer turnover in global, industrial open source projects: Insights from applying survival analysis. In: IEEE. **2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)**. [S.l.], 2017. p. 66–75.

LUTHIGER, B. Fun and software development. In: **Proceedings of the first international conference on open source systems, genova**. [S.l.: s.n.], 2005. p. 11–15.

OPSAHL, T.; AGNEESSENS, F.; SKVORETZ, J. Node centrality in weighted networks: Generalizing degree and shortest paths. **Social networks**, Elsevier, v. 32, n. 3, p. 245–251, 2010.

RIEHLE, D. How open source is changing the software developer's career. **IEEE Computer**, v. 48, n. 5, p. 51–57, 2015.

RIEHLE, D. et al. Paid vs. volunteer work in open source. In: IEEE. **2014 47th Hawaii International Conference on System Sciences (HICSS)**. [S.l.], 2014. p. 3286–3295.

RIGBY, P. C. et al. Quantifying and mitigating turnover-induced knowledge loss: case studies of chrome and a project at avaya. In: IEEE. **Software Engineering (ICSE), 2016 IEEE/ACM 38th International Conference on**. [S.l.], 2016. p. 1006–1016.

RISHI, D. **Affective sentiment and emotional analysis of pull request comments on GitHub**. 2017. Dissertação (Mestrado) — University of Waterloo, 2017.

ROBLES, G.; GONZALEZ-BARAHONA, J. M. Contributor turnover in libre software projects. In: SPRINGER. **IFIP International Conference on Open Source Systems**. [S.l.], 2006. p. 273–286.

SABIDUSSI, G. The centrality index of a graph. **Psychometrika**, Springer, v. 31, n. 4, p. 581–603, 1966.

TORAL, S. L.; MARTÍNEZ-TORRES, M. d. R.; BARRERO, F. Analysis of virtual communities supporting oss projects using social network analysis. **Information and Software Technology**, Elsevier, v. 52, n. 3, p. 296–303, 2010.

YE, Y.; KISHIDA, K. Toward an understanding of the motivation open source software developers. In: IEEE COMPUTER SOCIETY. **Proceedings of the 25th international conference on software engineering**. [S.l.], 2003. p. 419–429.

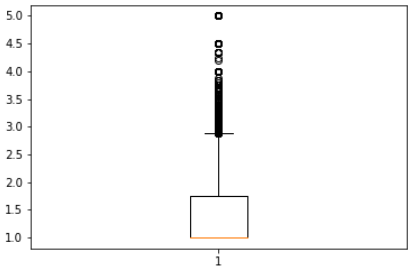
YU, Y.; BENLIAN, A.; HESS, T. An empirical study of volunteer members' perceived turnover in open source software projects. In: IEEE. **System Science (HICSS), 2012 45th Hawaii International Conference on**. [S.l.], 2012. p. 3396–3405.

ZANETTI, M. S. **A complex systems approach to software engineering**. 2013. Tese (Doutorado) — ETH Zurich, 2013.

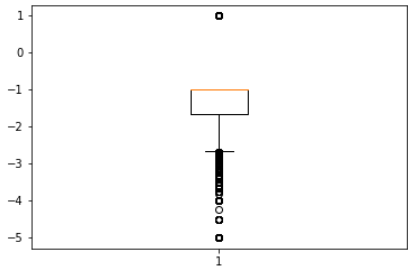
ZHU, X. et al. Employee turnover forecasting for human resource management based on time series analysis. **Journal of Applied Statistics**, Taylor & Francis, v. 44, n. 8, p. 1421–1440, 2017.

APÊNDICE A – OUTLIERS

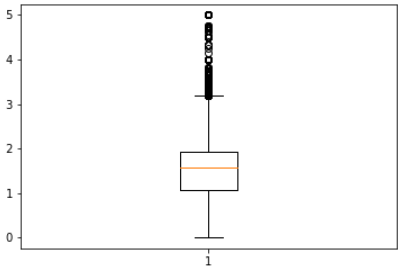
Figura A.1 – *Outliers* das Métricas do Sentimento de Textos: Todos Contribuidores



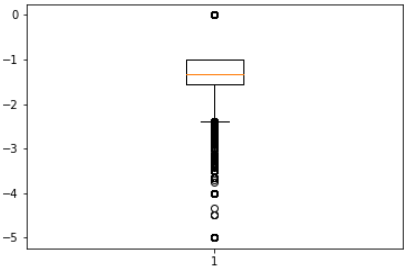
(a) Positividade



(b) Negatividade



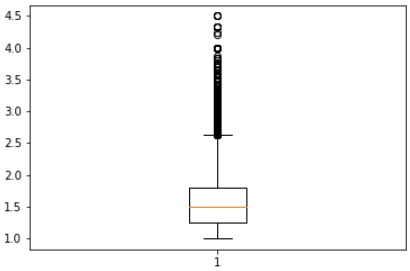
(c) Negatividade Recebida



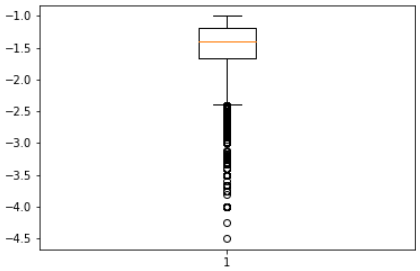
(d) Positividade Recebida

Fonte: Autor.

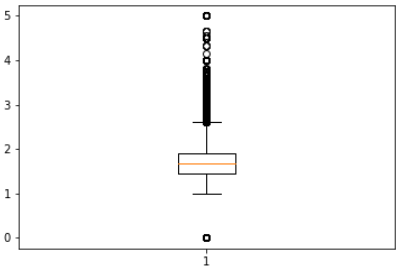
Figura A.2 – *Outliers* das Métricas do Sentimento de Textos: Contribuidores Não Casuais



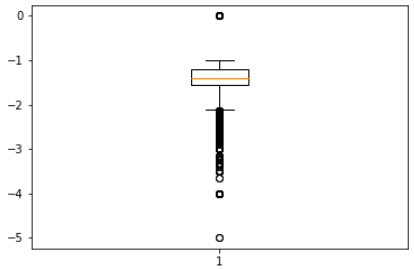
(a) Positividade



(b) Negatividade



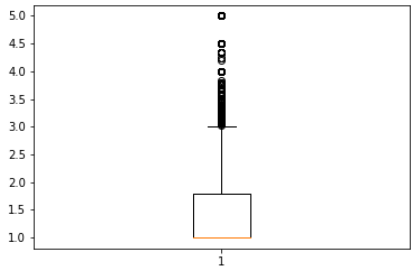
(c) Negatividade Recebida



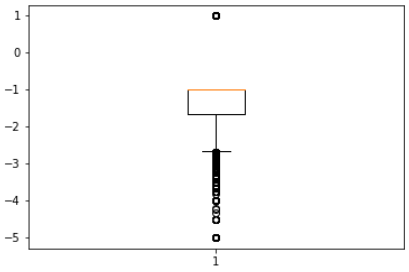
(d) Positividade Recebida

Fonte: Autor.

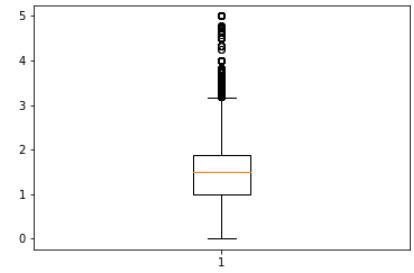
Figura A.3 – *Outliers* das Métricas do Sentimento de Textos da Ultima Interação: Todos Contribuidores



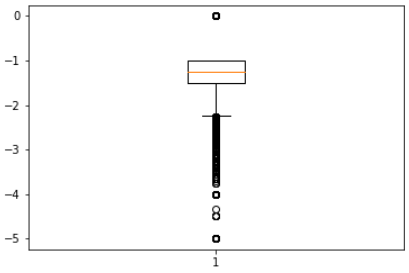
(a) Positividade



(b) Negatividade



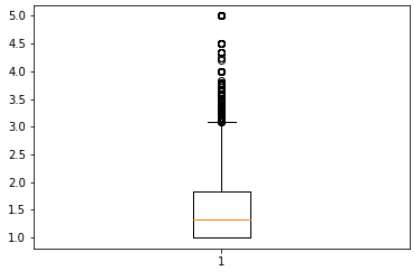
(c) Negatividade Recebida



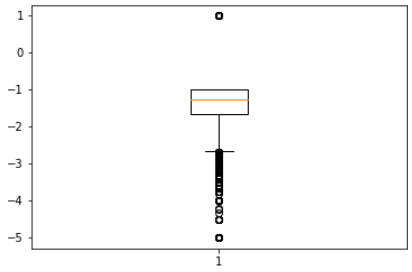
(d) Positividade Recebida

Fonte: Autor.

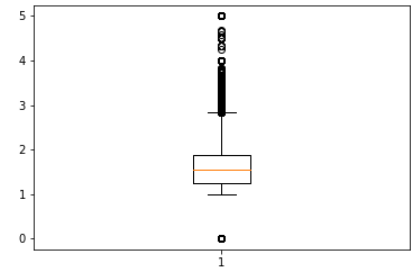
Figura A.4 – *Outliers* das Métricas do Sentimento de Textos da Ultima Interação: Contribuidores Não Casuais



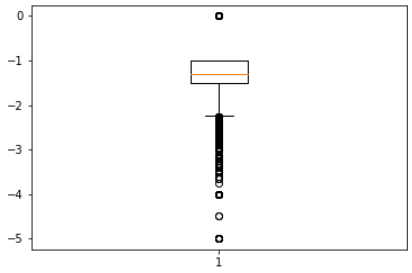
(a) Positividade



(b) Negatividade



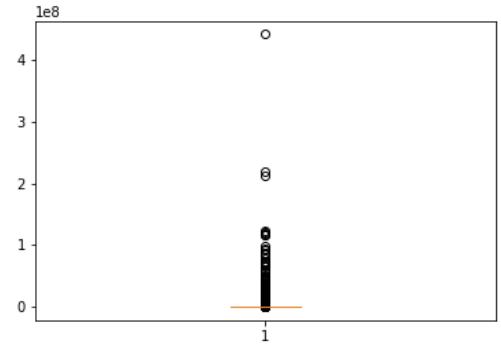
(c) Negatividade Recebida



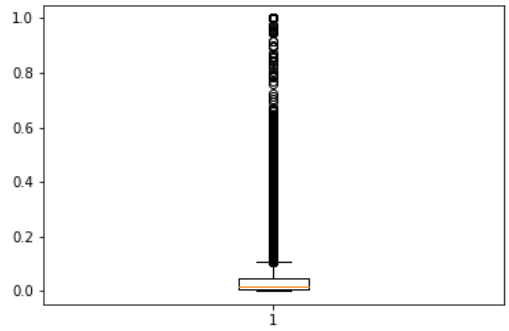
(d) Positividade Recebida

Fonte: Autor.

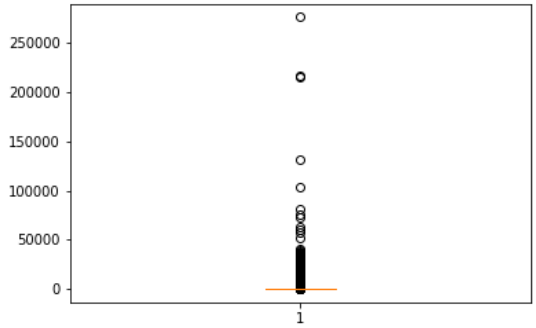
Figura A.5 – *Outliers* das Métricas de Rede: Todos Contribuidores



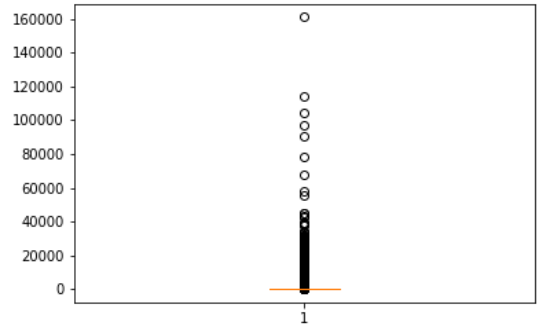
(a) Betweenness



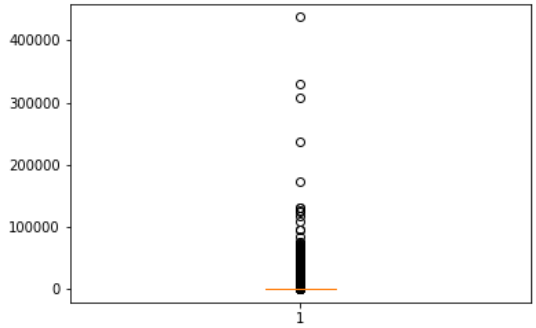
(b) Closeness



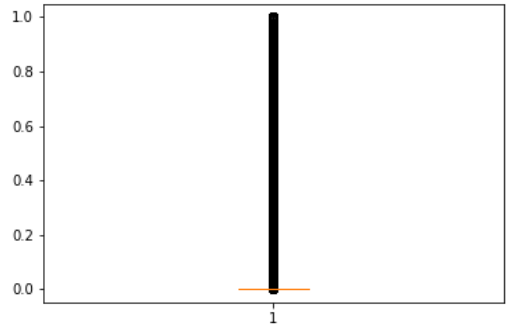
(c) Degree In



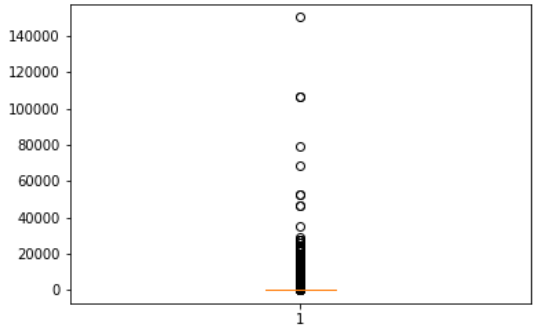
(d) Degree Out



(e) Degree Total



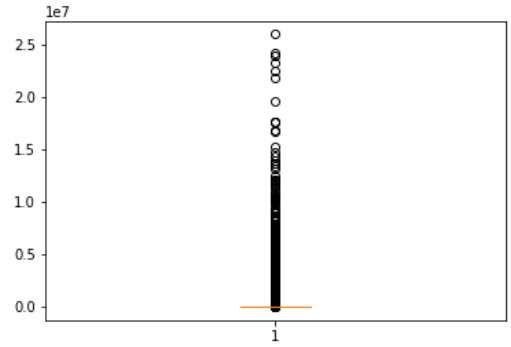
(f) Eigenvector



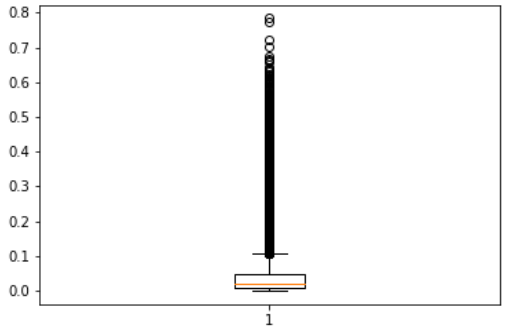
(g) Coreness

Fonte: Autor.

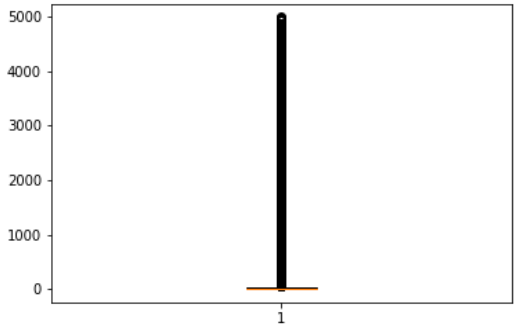
Figura A.6 – *Outliers* das Métricas de Rede: Contribuidores Não Casuais



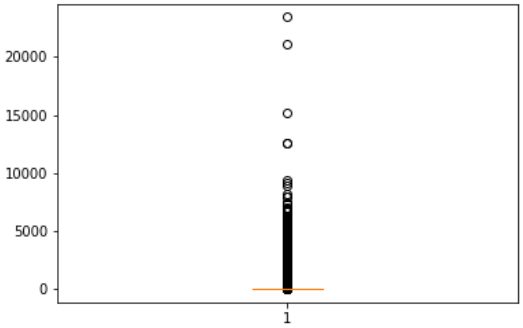
(a) Betweenness



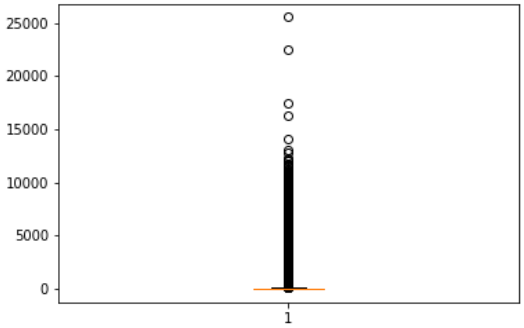
(b) Closeness



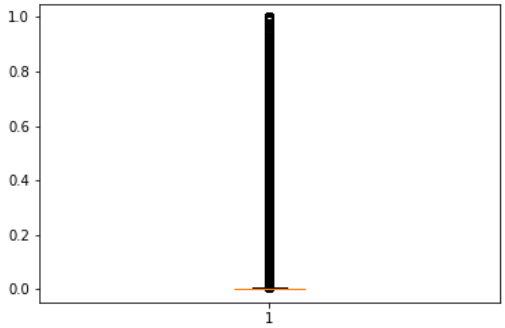
(c) Degree In



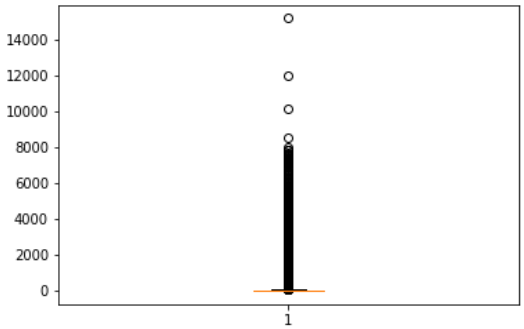
(d) Degree Out



(e) Degree Total



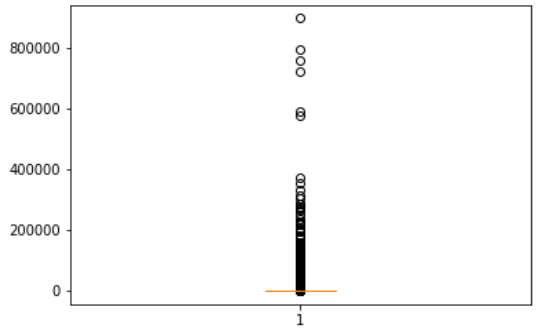
(f) Eigenvector



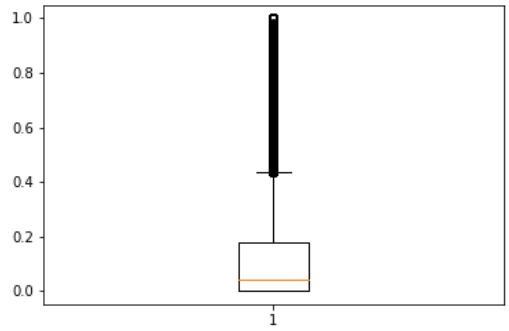
(g) Coreness

Fonte: Autor.

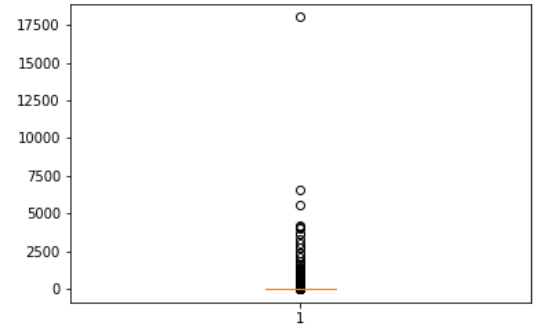
Figura A.7 – *Outliers* das Métricas de Rede da Ultima Interação: Todos Contribuidores



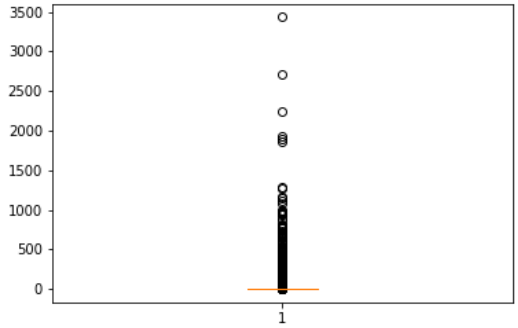
(a) Betweenness



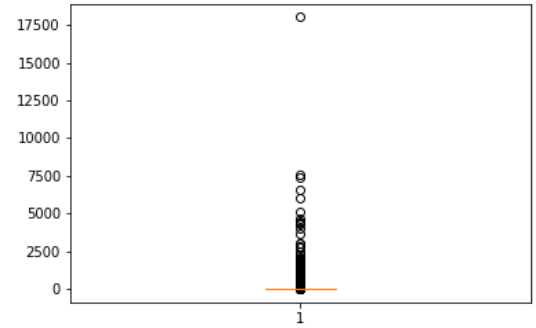
(b) Closeness



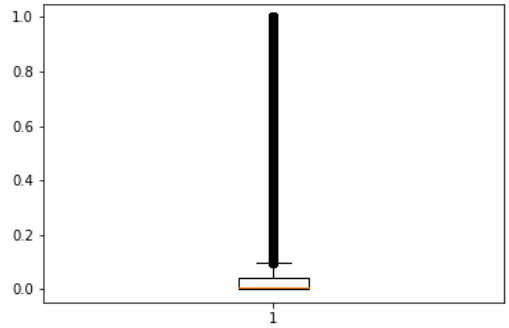
(c) Degree In



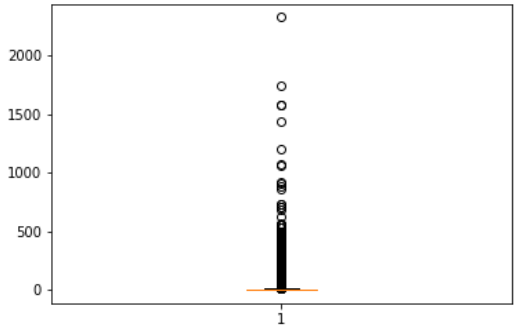
(d) Degree Out



(e) Degree Total



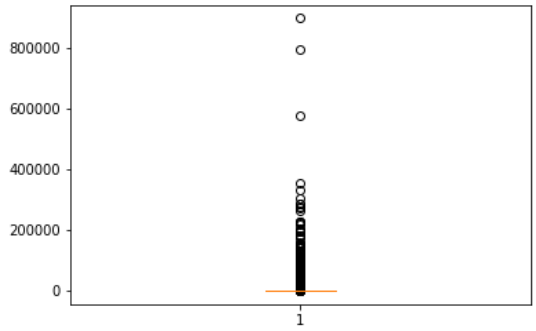
(f) Eigenvector



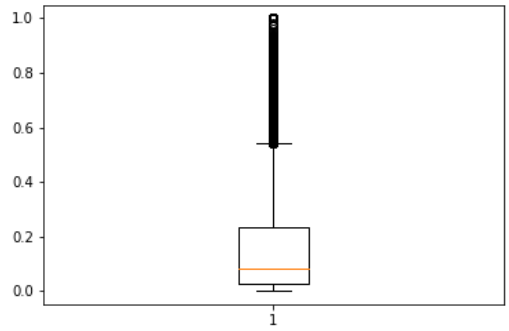
(g) Coreness

Fonte: Autor.

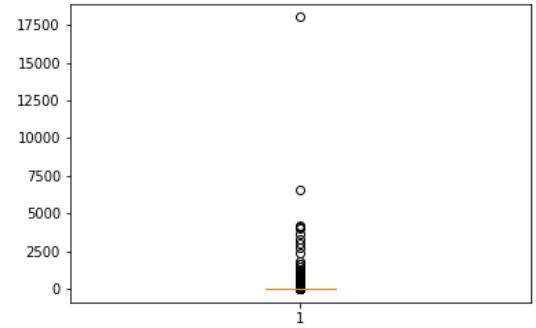
Figura A.8 – *Outliers* das Métricas de Rede da Ultima Interação: Não Casuais



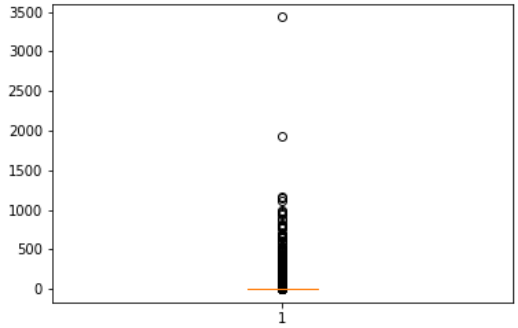
(a) Betweenness



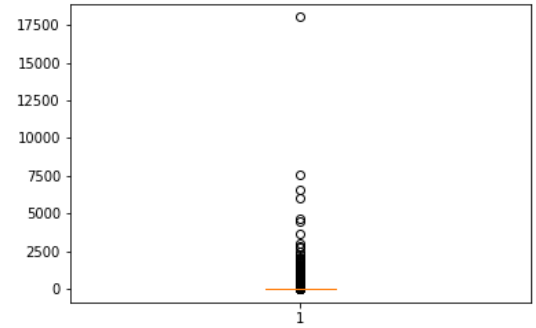
(b) Closeness



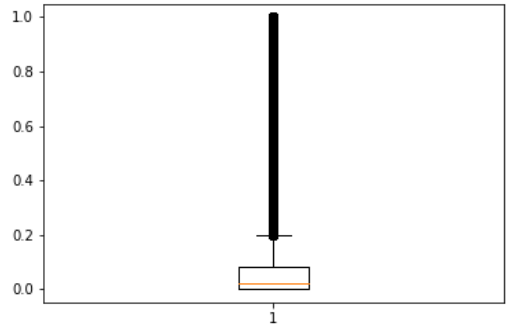
(c) Degree In



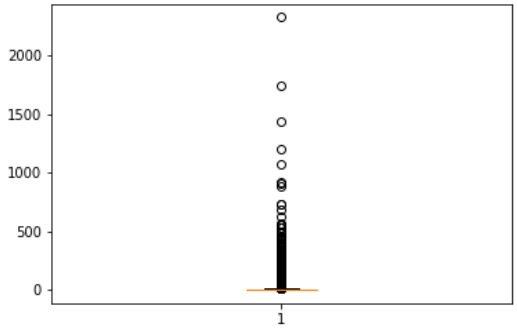
(d) Degree Out



(e) Degree Total



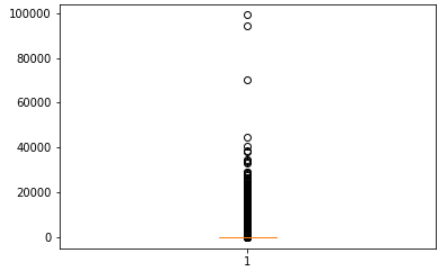
(f) Eigenvector



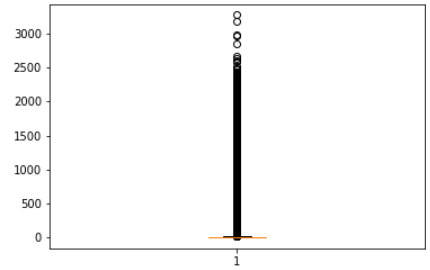
(g) Coreness

Fonte: Autor.

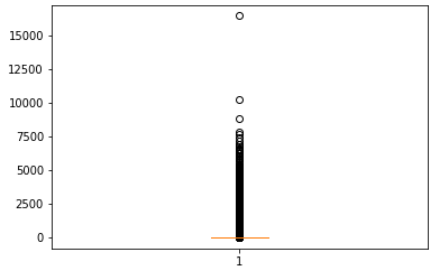
Figura A.9 – *Outliers* das Métricas Simples: Todos Contribuidores



(a) Número de Interações



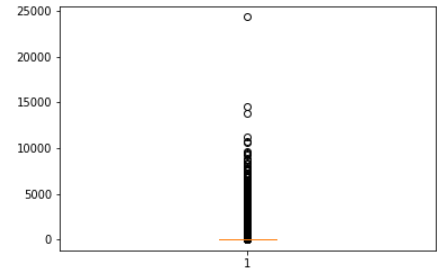
(b) Intervalo Sem Interagir



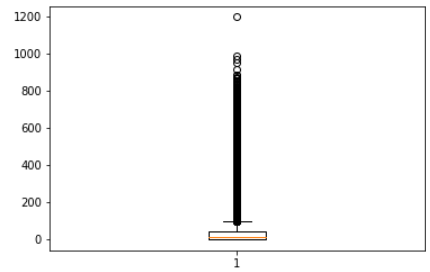
(c) Número de Dias Ativo

Fonte: Autor.

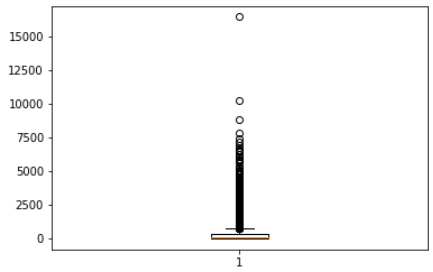
Figura A.10 – *Outliers* das Métricas Simples: Contribuidores Não Casuais



(a) Número de Interações



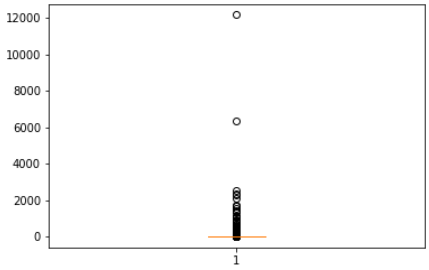
(b) Intervalo Sem Interagir



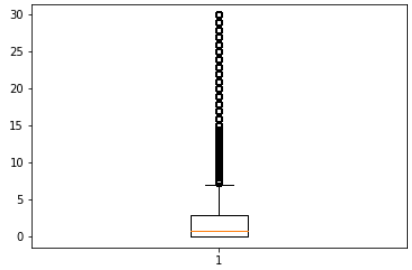
(c) Número de Dias Ativo

Fonte: Autor.

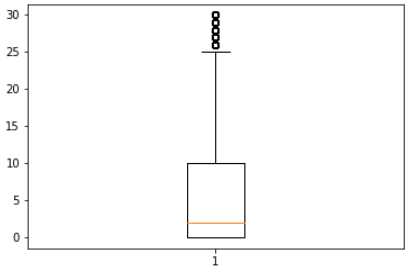
Figura A.11 – *Outliers* das Métricas Simples da Ultima Interação: Todos Contribuidores



(a) Número de Interações



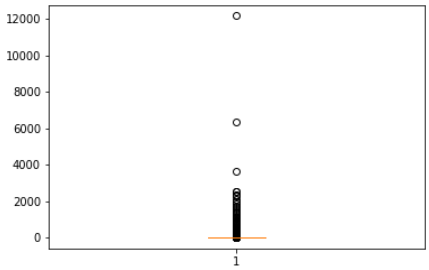
(b) Intervalo Sem Interagir



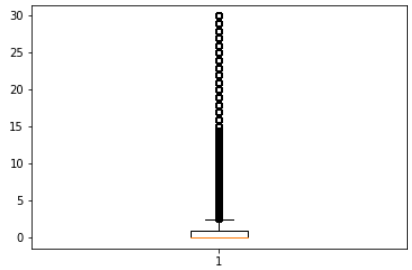
(c) Número de Dias Ativo

Fonte: Autor.

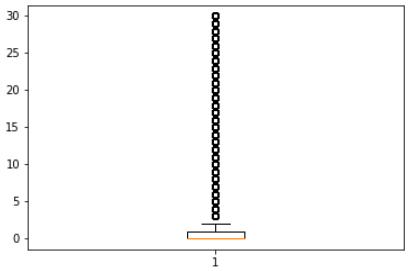
Figura A.12 – *Outliers* das Métricas Simples da Ultima Interação: Contribuidores Não Casuais



(a) Número de Interações



(b) Intervalo Sem Interagir

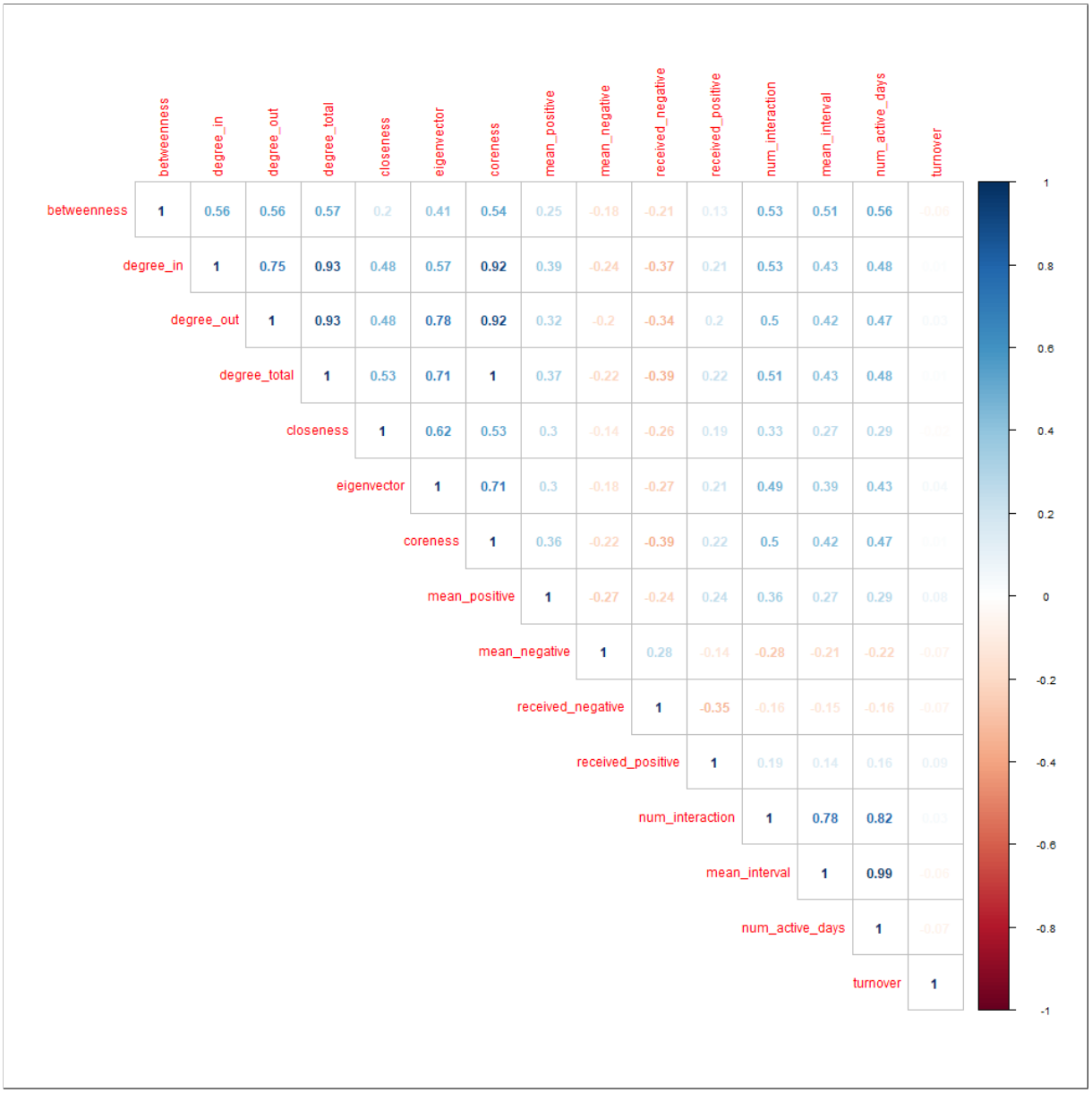


(c) Número de Dias Ativo

Fonte: Autor.

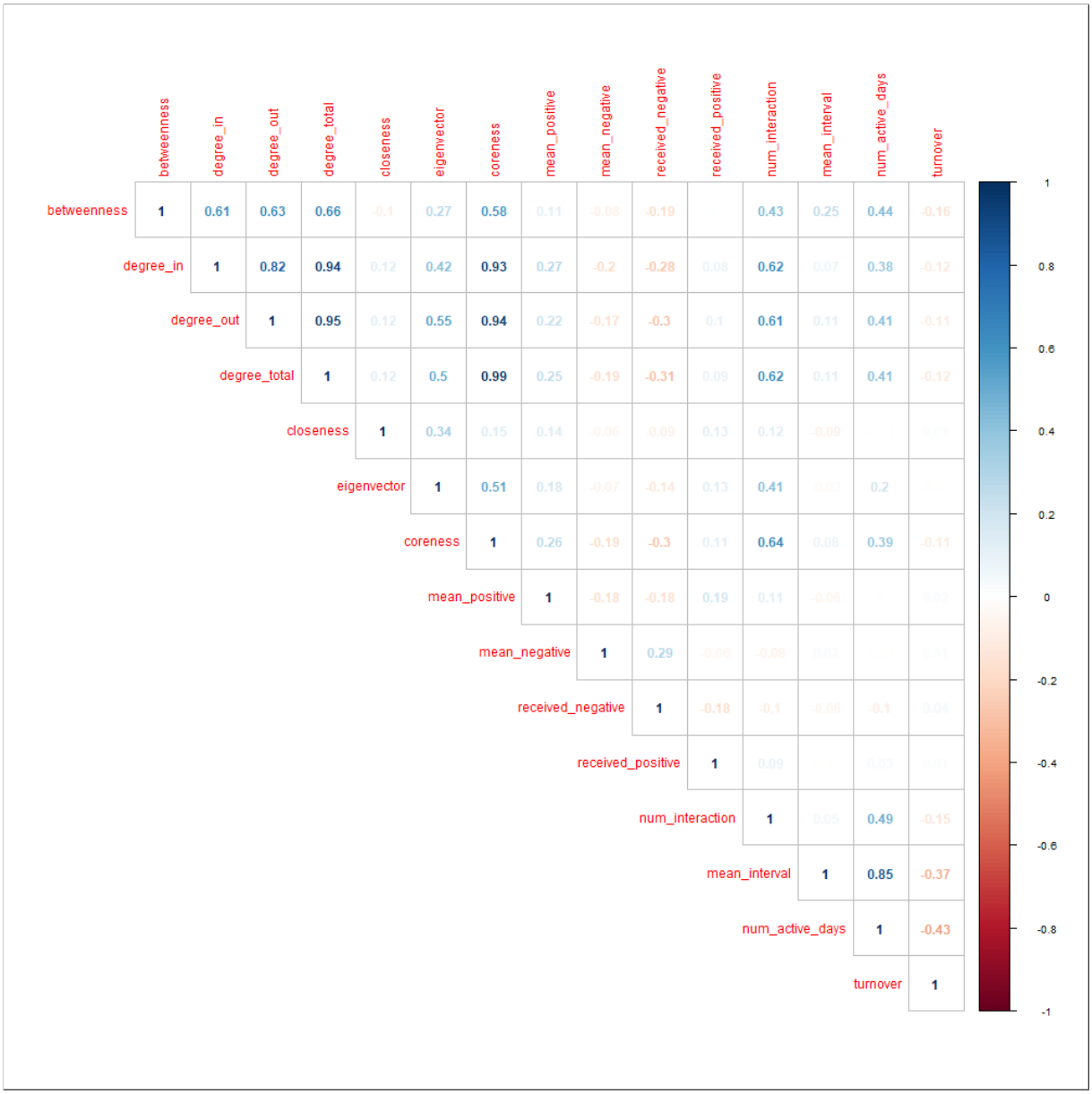
APÊNDICE B – CORRELAÇÃO

Figura B.1 – Correlação SPEARMAN das Métricas de Todos os Contribuidores.



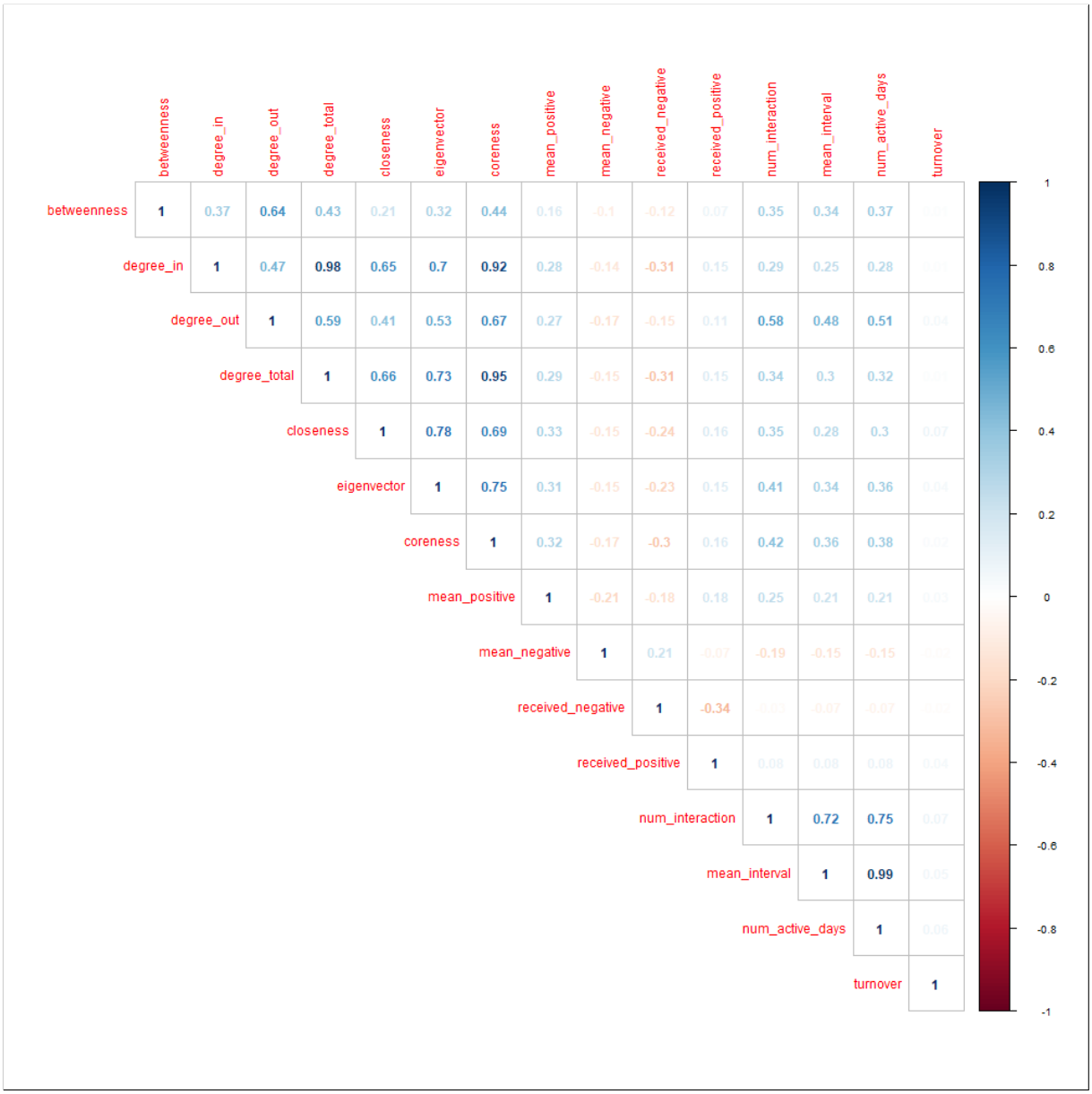
Fonte: Autor.

Figura B.2 – Correlação SPEARMAN das Métricas dos Contribuidores Não Casuais.



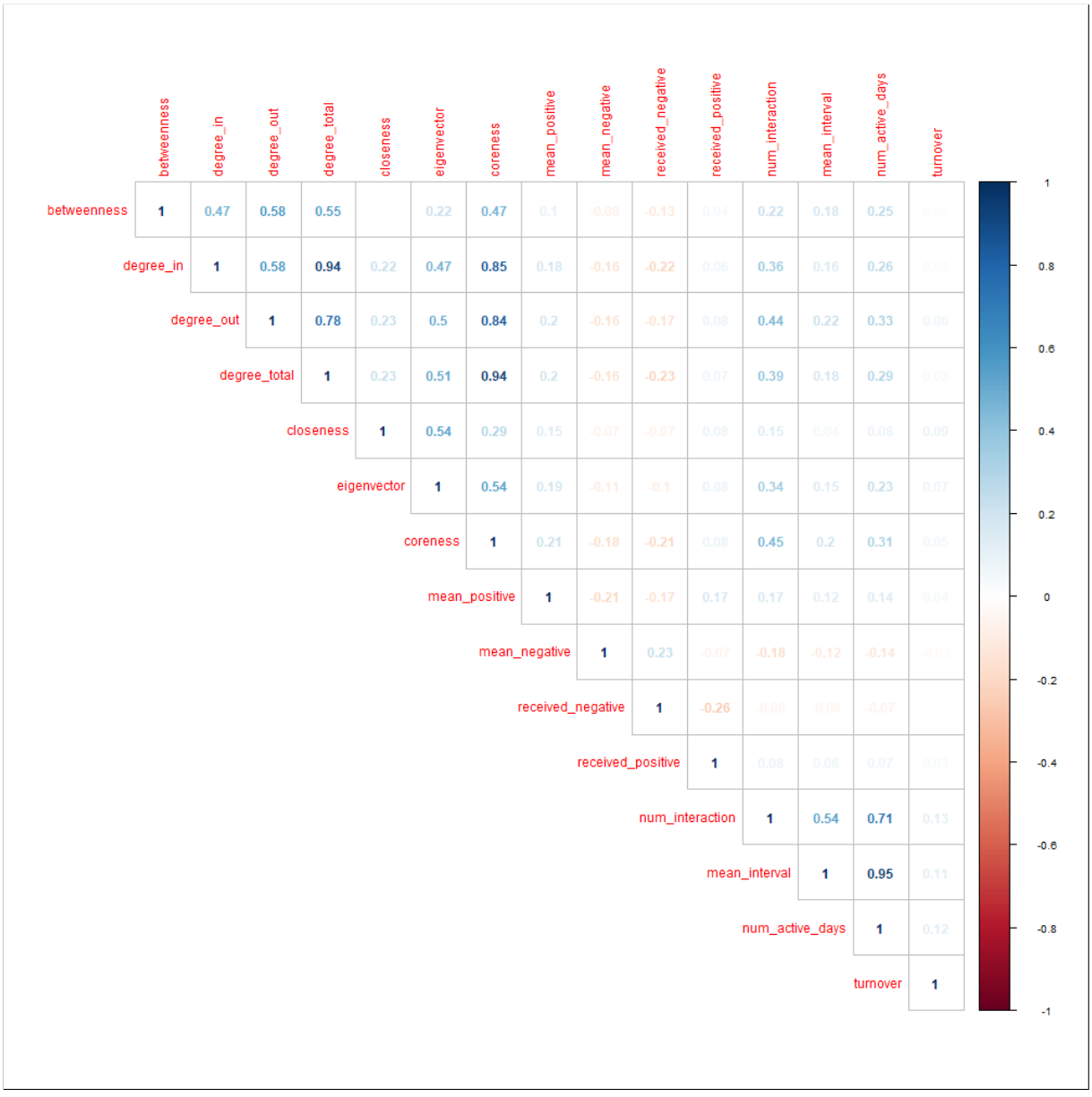
Fonte: Autor.

Figura B.3 – Correlação SPEARMAN das Métricas de Todos os Contribuidores Considerando a Ultima Interação.



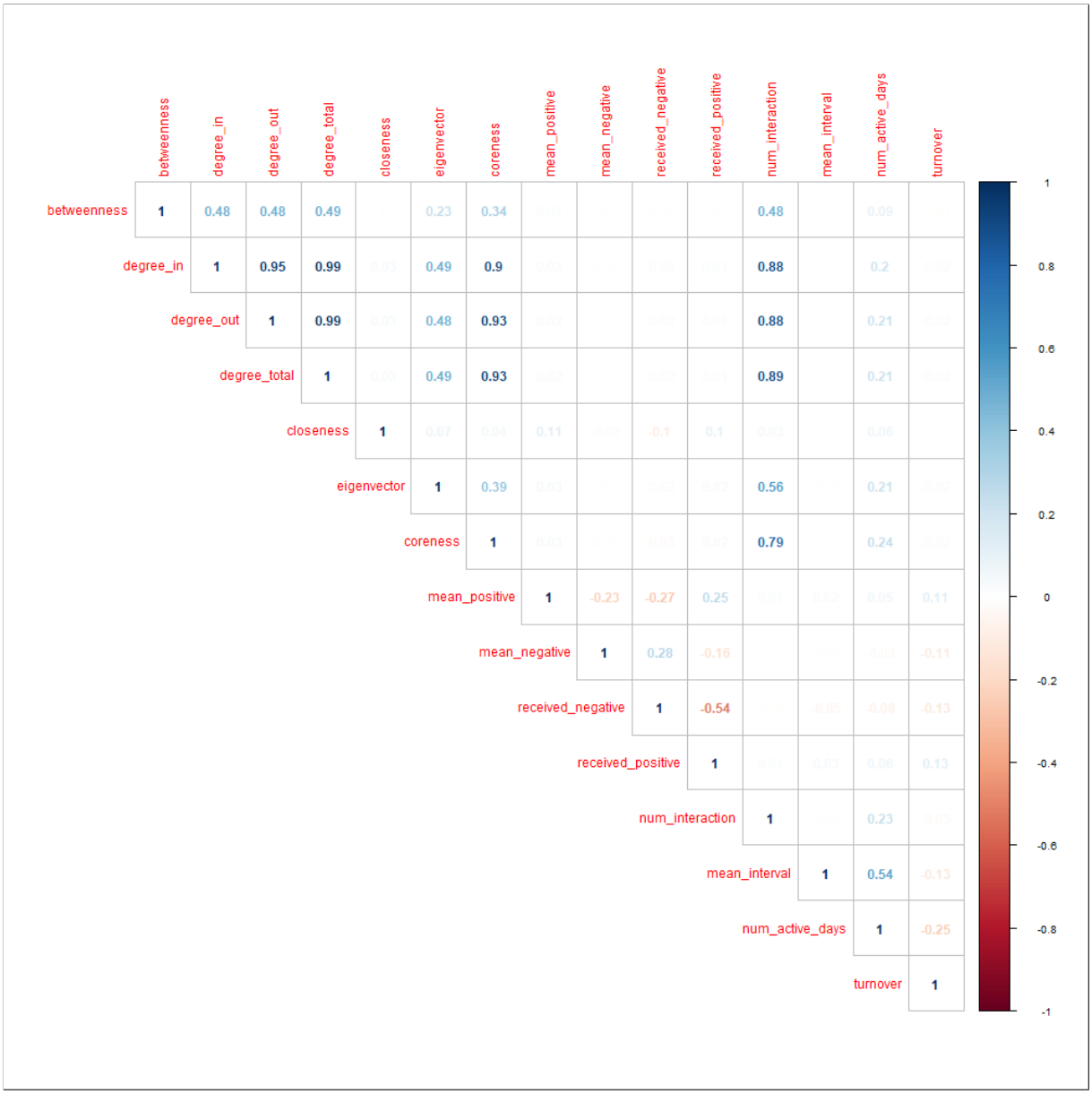
Fonte: Autor.

Figura B.4 – Correlação SPEARMAN das Métricas dos Contribuidores Não Casuais Considerando a Ultima Interação.



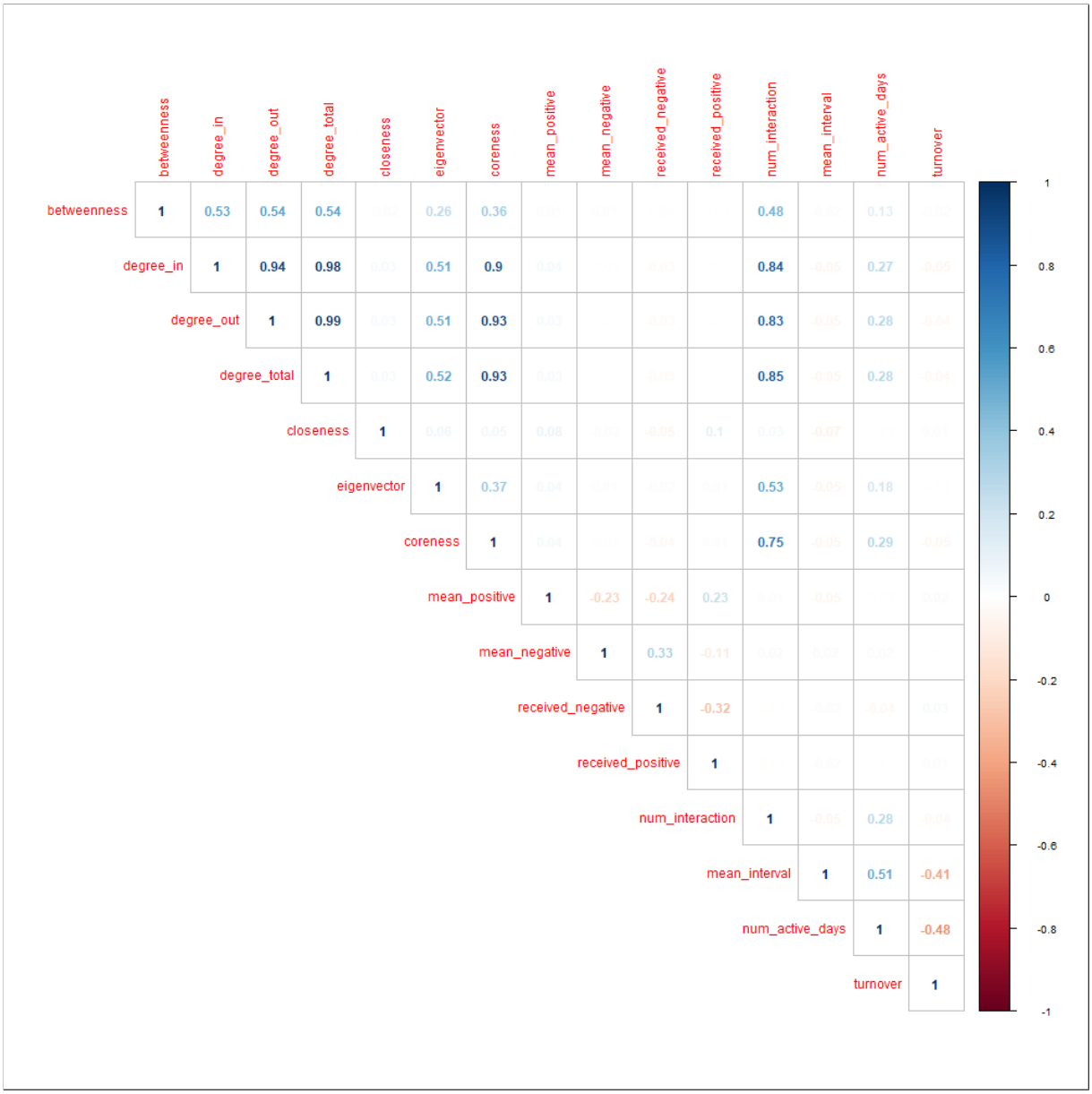
Fonte: Autor.

Figura B.5 – Correlação PEARSON das Métricas de Todos os Contribuidores.



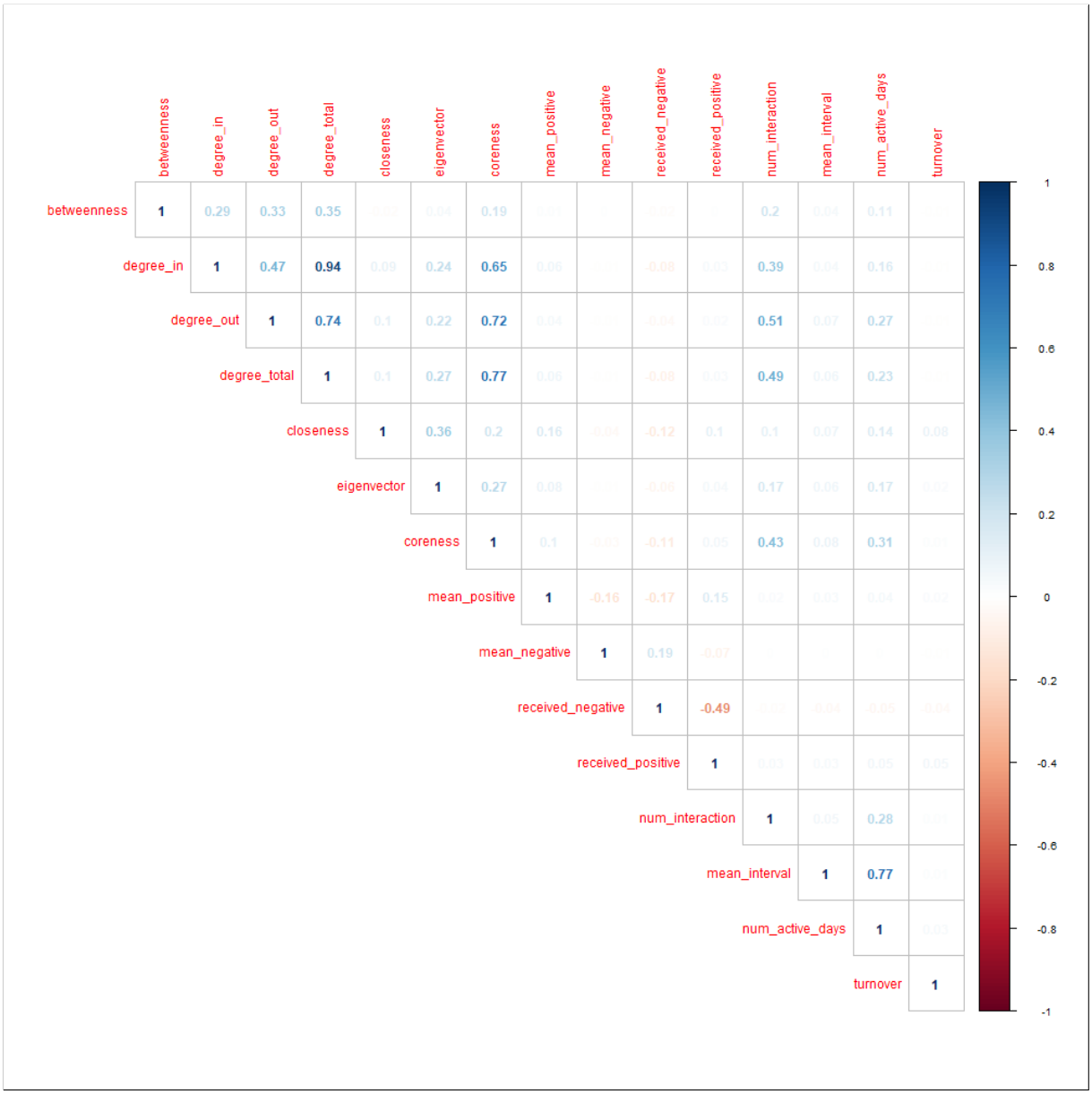
Fonte: Autor.

Figura B.6 – Correlação PEARSON das Métricas dos Contribuidores Não Casuais.



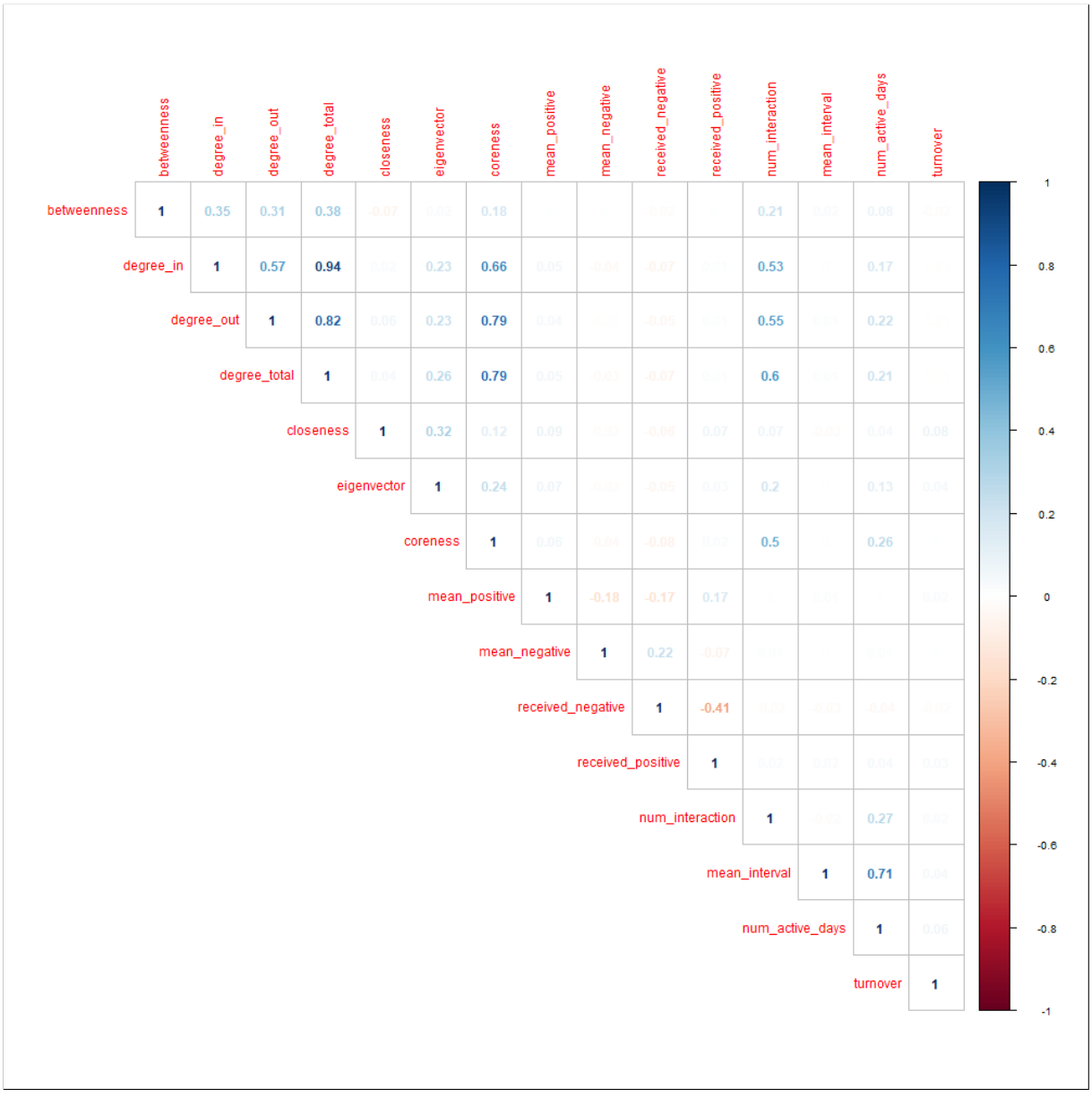
Fonte: Autor.

Figura B.7 – Correlação PEARSON das Métricas de Todos os Contribuidores Considerando a Ultima Interação.



Fonte: Autor.

Figura B.8 – Correlação PEARSON das Métricas dos Contribuidores Não Casuais Considerando a Ultima Interação.



Fonte: Autor.

APÊNDICE C – RESULTADOS

Quadro C.1 – Resultados com *outliers*.

(continua)

ID	Parâmetros	OverSampling	Classe 1 (%)		Classe 2 (%)	
			TP	FP	TN	FN
1	M=50 L= 1		30,25	69,75	2,24	97,76
1	M= 10 L = 1		26,26	73,74	1,92	98,08
1	M = 100 L = 2		100,00	0,00	100,00	0,00
1	M = 25 L=0.8	Borderline S.	87,57	12,43	23,25	76,75
1	M = 25 L=1	Borderline S.	88,48	11,52	24,43	75,57
2	M=50 L= 1		25,11	74,89	1,32	98,68
2	M= 10 L = 1		18,20	81,80	0,84	99,16
2	M = 100 L = 2		13,49	86,51	0,00	100,00
2	M = 25 L = 0.8	Borderline S.	65,33	34,67	24,14	75,86
2	M = 25 L=1	Borderline S.	60,01	39,99	19,28	80,72
3	M=50 L= 1		0,02	99,98	0,00	100,00
3	M= 10 L = 1		0,00	100,00	0,00	100,00
3	M = 100 L = 2		32,38	67,62	62,43	37,57
3	M = 25 L=0.8	Borderline S.	55,08	44,92	27,63	72,37
3	M = 25 L=1	Borderline S.	50,00	50,00	29,33	70,67
4	M=50 L= 1		0,00	100,00	0,00	100,00
4	M= 10 L = 1		0,00	100,00	0,00	100,00
4	M = 100 L = 2		20,47	79,53	32,73	67,27
4	M = 25 L = 0.8	Borderline S.	55,48	44,52	38,85	61,15
4	M = 25 L=1	Borderline S.	65,56	34,44	51,37	48,63
7			19,20	80,80	0,89	99,11
8			0,05	99,95	0,00	100,00
9		Borderline S.	48,71	51,29	6,55	93,45
9		R.OverSam.	52,64	47,36	7,60	92,40
9			36,84	63,16	4,34	95,66
9		ADASYN	51,15	48,85	7,20	92,80
9			48,20	51,80	6,43	93,57
9			59,36	40,64	9,80	90,20
9		Borderline S.	58,74	41,26	9,66	90,34
10		Borderline S.	13,52	86,48	2,37	97,63
10		R.OverSam.	14,43	85,57	2,56	97,44
10			11,49	88,51	1,98	98,02
10		ADASYN	15,17	84,83	2,73	97,27
10		R.OverSam.	12,07	87,93	2,11	97,89
10			13,94	86,06	2,44	97,56
10		Borderline S.	13,70	86,30	2,30	97,70
11		Borderline S.	74,59	25,41	59,25	40,75
11		R.OverSam.	86,64	13,36	76,54	23,46
11			8,40	91,60	5,45	94,55
11		ADASYN	91,47	8,53	84,83	15,17
11		R.OverSam.	11,68	88,32	8,04	91,96

Quadro C.2 – Resultados com *outliers*.

(conclusão)

ID	Parâmetros	OverSampling	Classe 1 (%)		Classe 2 (%)	
			TP	FP	TN	FN
11			73,41	26,59	57,46	42,54
11		Borderline S.	70,92	29,08	52,22	47,78
12		Borderline S.	1,56	98,44	1,24	98,76
12		R.OverSam.	2,17	97,83	1,91	98,09
12			0,00	100,00	0,00	100,00
12		ADASYN	14,06	85,94	9,51	90,49
12		R.OverSam.	2,15	97,85	1,85	98,15
12			11,53	88,47	9,99	90,01
12		Borderline S.	11,68	88,32	7,71	92,29
13		Borderline S.	91,77	8,23	28,03	71,97
13		R.OverSam.	92,79	7,21	29,46	70,54
13			0,00	100,00	0,00	100,00
13		ADASYN	95,91	4,09	34,40	65,60
13		R.OverSam.	92,79	7,21	29,46	70,54
13			48,79	51,21	10,59	89,41
13		Borderline S.	47,42	52,58	10,32	89,68
14		Borderline S.	79,51	20,49	52,82	47,18
14		R.OverSam.	41,09	58,91	7,04	92,96
14			13,49	86,51	0,00	100,00
14		ADASYN	86,10	13,90	61,06	38,94
14			40,66	59,34	6,78	93,22
14			44,35	55,65	17,43	82,57
14		Borderline S.	45,92	54,08	16,05	83,95
15		Borderline S.	67,62	32,38	37,57	62,43
15		R.OverSam.	72,53	27,47	40,35	59,65
15			0,00	100,00	0,00	100,00
15		ADASYN	67,62	32,38	37,57	62,43
15		R.OverSam.	72,53	27,47	40,35	59,65
15			19,90	80,10	10,91	89,09
15		Borderline S.	20,47	79,53	11,26	88,74
16		Borderline S.	75,85	24,15	60,48	39,52
16		R.OverSam.	74,03	25,97	58,58	41,42
16			0,00	100,00	0,00	100,00
16		ADASYN	87,38	12,62	76,72	23,28
16		R.OverSam.	73,73	26,27	58,29	41,71
16			31,93	68,07	23,13	76,87
16		Borderline S.	29,24	70,76	19,62	80,38

Fonte: Autor.

Quadro C.3 – Resultados sem *outliers*.

(continua)

ID	Parâmetros	OverSampling	Classe 1 (%)		Classe 2 (%)	
			TP	FP	TN	FN
1	M=50 L= 1		30,25	69,75	2,24	97,76
1	M=25 L= 1.5	Borderline S.	86,58	13,42	22,97	77,03
1	M=25 L= 0.8	Borderline S.	89,08	10,92	23,13	76,87
1	M=20 L= 0.7	Borderline S.	89,18	10,82	24,81	75,19
1	M=20 L=2	Adasyn	4,20	95,80	66,32	33,68
1	M=25 L=0.7	Adasyn	93,45	6,55	29,96	70,04
1	M=30 L=0.7	Adasyn	93,36	6,64	29,29	70,71
1	M=50 L=0.7	Adasyn	92,25	7,75	27,42	72,58
1	M=50 L=1	Adasyn	89,19	10,81	25,04	74,96
2	M=25 L= 1.5	Borderline S.	60,44	39,56	20,00	80,00
2	M=25 L= 0.8	Borderline S.	61,39	38,61	20,26	79,74
2	M=20 L= 0.7	Borderline S.	58,55	41,45	18,76	81,24
2	M=25 L=0.7	Adasyn	62,93	37,07	21,19	78,81
2	M=30 L=0.7	Adasyn	61,46	38,54	20,04	79,96
3	M=25 L= 1.5	Borderline S.	40,31	59,69	19,12	80,88
3	M=25 L= 0.8	Borderline S.	58,22	41,78	29,04	70,96
3	M=20 L= 0.7	Borderline S.	59,05	40,95	29,49	70,51
3	M=20 L=2	Adasyn	22,78	77,22	39,36	60,64
3	M=25 L=0.7	Adasyn	50,77	49,23	26,05	73,95
3	M=30 L=0.7	Adasyn	52,10	47,90	25,75	74,25
4	M=25 L= 1.5	Borderline S.	53,64	46,36	34,62	65,38
4	M=20 L= 0.7	Borderline S.	55,92	44,08	37,76	62,24
4	M=25 L=0.7	Adasyn	62,35	37,65	47,71	52,29
4	M=30 L=0.7	Adasyn	62,35	37,65	47,71	52,29
4	M=25 L= 0.8	Borderline S.	49,78	50,22	30,72	69,28
5			6,75	93,25	0,26	99,74
5		Borderline S.	83,23	16,77	18,96	81,04
6		Borderline S.	71,41	28,59	53,67	46,33
7		Borderline S.	88,72	11,28	23,08	76,92
8		Borderline S.	65,60	34,40	34,12	65,88
9		Borderline S.	52,15	47,85	7,88	92,12
9		ADASYN	58,14	41,86	9,36	90,64
9		Borderline S.	42,70	57,30	5,70	94,30
10		Borderline S.	13,29	86,71	2,36	97,64
10		ADASYN	16,11	83,89	3,02	96,98
10		Borderline S.	14,07	85,93	2,57	97,43
11		Borderline S.	72,40	27,60	55,92	44,08

Quadro C.4 – Resultados sem *outliers*.

(conclusão)

ID	Parâmetros	OverSampling	Classe 1 (%)		Classe 2 (%)	
			TP	FP	TN	FN
11		ADASYN	89,42	10,58	80,14	19,86
11		Borderline S.	12,28	87,72	7,40	92,60
12		Borderline S.	6,36	93,64	3,82	96,18
12		ADASYN	3,34	96,66	2,89	97,11
12		Borderline S.	16,88	83,12	11,15	88,85
13		Borderline S.	47,54	52,46	10,35	89,65
14		Borderline S.	49,00	51,00	19,65	80,35
15		Borderline S.	19,93	80,07	11,10	88,90
16		Borderline S.	32,17	67,83	23,52	76,48
17		Borderline S.	64,89	35,11	12,18	87,82
18		Borderline S.	50,46	49,54	14,80	85,20
19		Borderline S.	14,08	85,92	5,50	94,50
20		Borderline S.	23,48	76,52	13,24	86,76

Fonte: Autor.

ANEXO A – TABELA DE SIGNIFICÂNCIA DE GRUBBS

Quadro A.1 – Tabela de Significância criada por Grubbs para detectar *outliers* em listas de valores.

Número de Observações	Significância (5%)
3	1.15
4	1.46
5	1.67
6	1.82
7	1.94
8	2.03
9	2.11
10	2.18
11	2.23
12	2.29
13	2.33
14	2.37
15	2.41
16	2.44
17	2.47
18	2.50
19	2.53
20	2.56
21	2.58
22	2.60
23	2.62
24	2.64
25	2.66
30	2.75
35	2.82
40	2.87
45	2.92
50	2.96
60	3.03
70	3.09
80	3.14
90	3.18
100	3.21

Fonte: Grubbs (1969).