Avaliação de Técnicas para a Identificação de Mantenedores de Código Fonte

Otávio Cury da Costa Castro

Orientador: Pedro de Alcântara dos Santos Neto

Coorientador: Guilherme Amaral Avelino

INTRODUÇÃO:

- Contexto e Motivação
- Definição do Problema
- Visão Geral da Proposta
- Objetivos

Contexto e Motivação

Contexto e Motivação

- Alterações no código fonte

Contexto e Motivação

- Necessidade de uma gestão eficiente

Contexto e Motivação

- Gerenciamento complexo de projetos grandes

Contexto e Motivação

- Informações relevantes sobre o desenvolvimento

Contexto e Motivação

- Quais desenvolvedores possuem conhecimento em quais arquivos de código fonte?

Contexto e Motivação

- Informação útil em diferentes situações:
  - **Resolução de *bugs***

Contexto e Motivação

- Informação útil em diferentes situações:
  - **Orientação de novos membros**

Contexto e Motivação

- Informação útil em diferentes situações:
  - **Implementação de novas funcionalidades**

Contexto e Motivação

- Informação útil em diferentes situações
  - **Revisão de código**

Contexto e Motivação

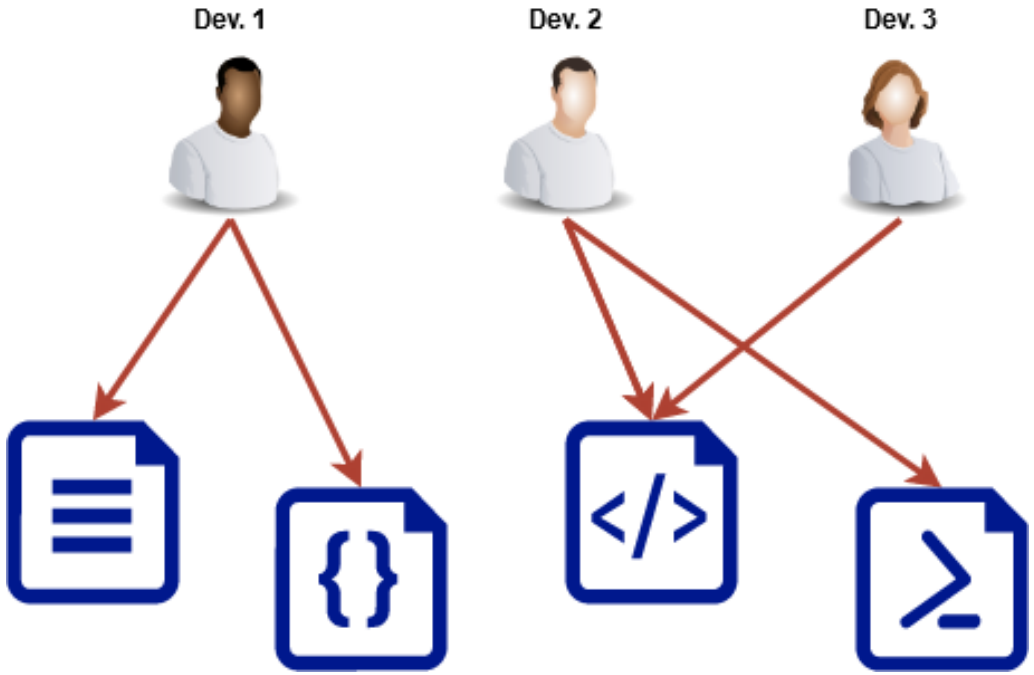
- Desafios da gerência: grande quantidade de informações.

Contexto e Motivação

- Solução: recorrer à informações presentes em Sistemas de Controle de Versão (SCV).

Contexto e Motivação

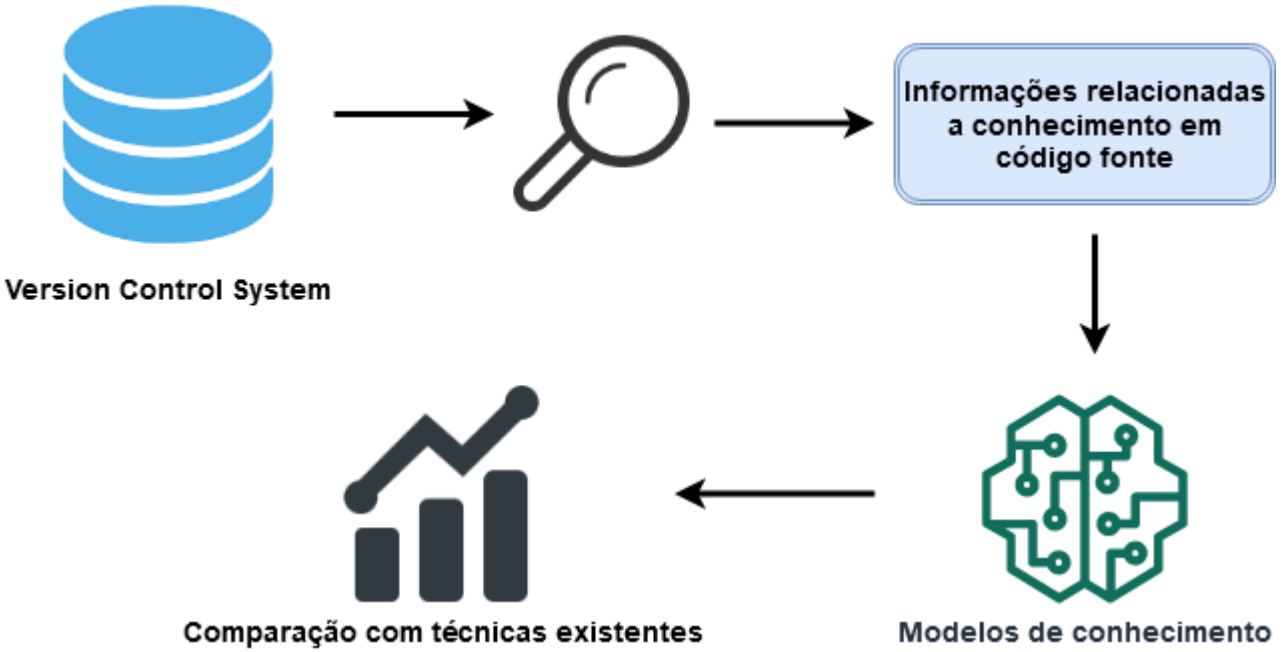
- Técnicas automatizar a identificação de mantenedores de arquivos de código fonte.



Contexto e Motivação

- Em um trabalho relacionado foi feita a comparação da performance de três técnicas na identificação de mantenedores de arquivo (AVELINO et al., 2018).
- *Tamanho do arquivo* e *recência* influenciam na performances das técnicas.
- Esses resultados sugerem que técnicas que utilizam essas variáveis podem ser mais precisas.

Visão Geral da Proposta



Objetivos

- **Objetivos Principais:**
  1. Propor modelos para inferência de conhecimento em código fonte.
  2. Propor a utilização de classificadores de aprendizagem de máquina na identificação de mantenedores de arquivos.
  3. Comparar os desempenhos dos modelos propostos com os de outros presentes na literatura.
- **Objetivos secundários:**
  1. Construir um *dataset* de projetos para o estudo.
  2. Construir um *dataset* com o conhecimento dos desenvolvedores em arquivos de código fonte.
  3. Identificar quais variáveis extraídas de SCVs são relevantes para inferir o conhecimento de desenvolvedores em arquivos de código fonte.

TRABALHOS RELACIONADOS:

- Propostas de Novas Técnicas
- Comparação de Técnicas

Trabalhos Relacionados
Pesquisa Realizada

- Bases de dados bibliográficos utilizadas:

Trabalhos Relacionados
Pesquisa Realizada

- *String* de pesquisa: *(developer* **OR** programmer*) **AND** (familiarity **OR** knowledge **OR** expertise **OR** ownership **OR** authorship **OR** maintenance) **AND** (code **OR** file **OR** class **OR** package **OR** module) **AND** (version control **OR** revision control **OR** repositor*)*
- Resultados das buscas:

Base de Dados	Resultados
Scopus	475
Web Of Science	369
IEE Xplore	269
Total	1113
Total sem duplicação	663

Tabela 1 – Resultados das buscas nas bases bibliográficas.

Trabalhos Relacionados
Pesquisa Realizada

Critérios de Inclusão

1. Utiliza alguma técnica para a inferência de conhecimento em código fonte a partir de informações SCV; ***AND***
2. Publicado em revistas, jornais, ou trilhas de conferências; ***AND***
3. É um estudo completo.

- Foram selecionados 31 trabalhos relacionados após duas etapas de filtragem.

Trabalhos Relacionados
Novas Técnicas

- Muitos trabalhos baseiam-se na quantidade de alterações feitas para a inferência de conhecimento: (MCDONALD; ACKERMAN, 2000; MOCKUS; HERBSLEB, 2002; GIRBA, et al., 2005; HATTORI; LANZA, 2009; HATTORI; LANZA, 2010; RAHMAN; DEVANBU, 2011; HOSSEN; KAGDI; POSHYVANYK, 2014; SÜLÜN; TÜZÜN; DOGRUSÖZ, 2019).
- *Degree of Knowledge* (DOK): (FRITZ, et. al, 2014).
- Outras abordagem levam em consideração a recência de modificação no cálculo de expertise: (KAGDI; POSHYVANYK, 2009; KAGDI et al., 2012; SILVA, et. al, 2015; SUN et al., 2017; SÜLÜN; TÜZÜN; DOGRUSÖZ, 2020).

Trabalhos Relacionados
Novas Técnicas – Aprendizagem de máquina

- O trabalho (MONTANDON; SILVA; VALENTE, 2019) investigou a performance de classificadores supervisionados na identificação de expertise em bibliotecas e frameworks.
- Não foram identificados estudos que utilizam classificadores supervisionados para identificar mantenedores com base em informações de SCVs.

Trabalhos Relacionados
Comparação de técnicas

- Comparação de técnicas:
  - Identificação de experts (AVELINO et al., 2018; ANVIK; MURPHY, 2007)
  - Familiaridade com código fonte (KRÜGER et al., 2018)
  - Recomendação de revisores de código (HANNEBAUER et al., 2016)
- Há diferenças em propósito, técnicas comparadas e abrangência.

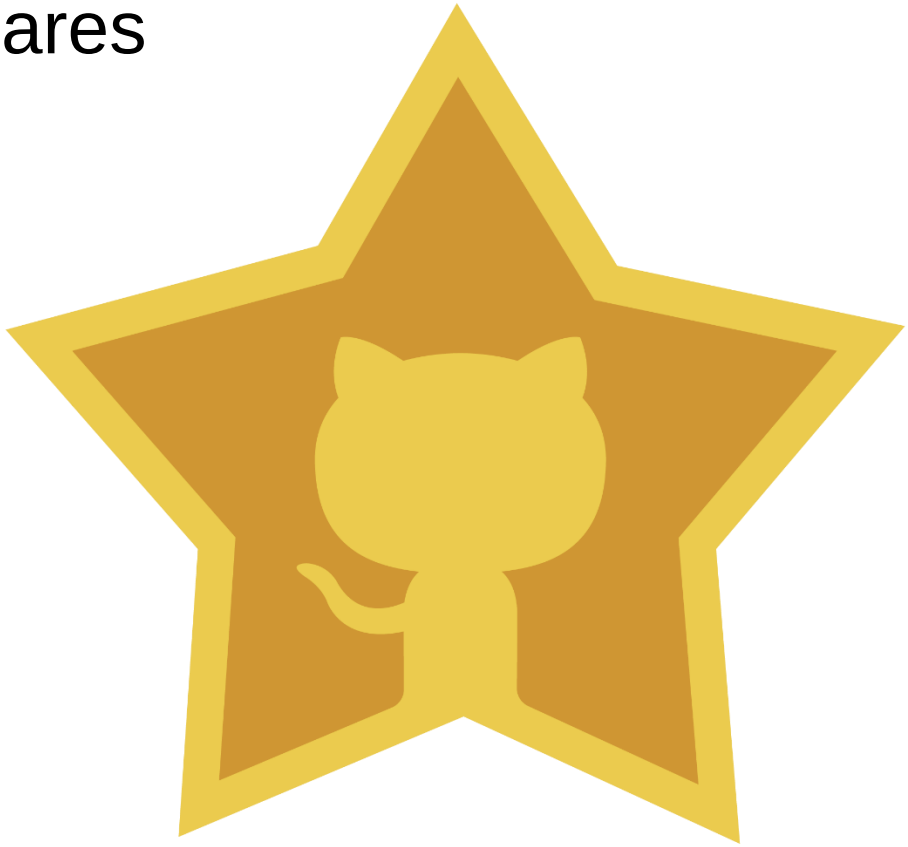
- **PROPOSTA:**
  - Coleta de Dados
  - Extração dos Históricos de Desenvolvimento
  - Aplicação de um Survey
  - Processamento das Respostas
  - Analise de Variáveis de Desenvolvimento
  - Técnicas Comparadas
  - Avaliação dos Técnicas Lineares
  - Classificadores de Aprendizagem de Máquina

Coleta de Dados
Projetos *open-source*

- Foram escolhidas 6 linguagens de programação populares no GitHub.

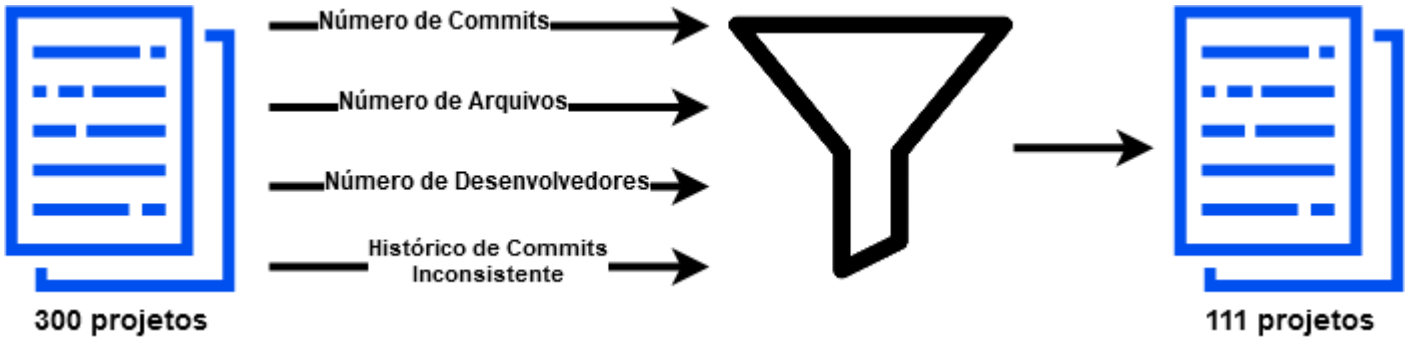
Coleta de Dados
Seleção dos projetos

- 50 projetos mais populares de cada linguagem.



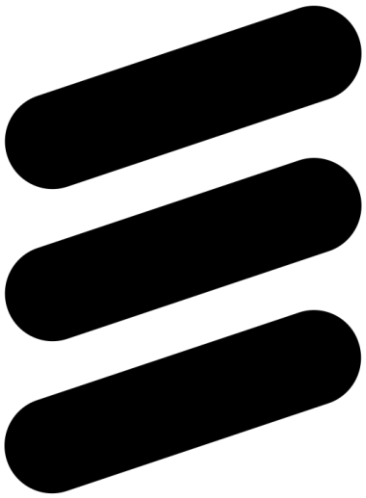
Coleta de Dados
Seleção dos projetos

- Filtragem dos projetos selecionados.



Coleta de Dados
Projetos privados

- Foram utilizados dados do desenvolvimento de dois projetos privados.



ERICSSON

Extraindo Histórico de Desenvolvimento

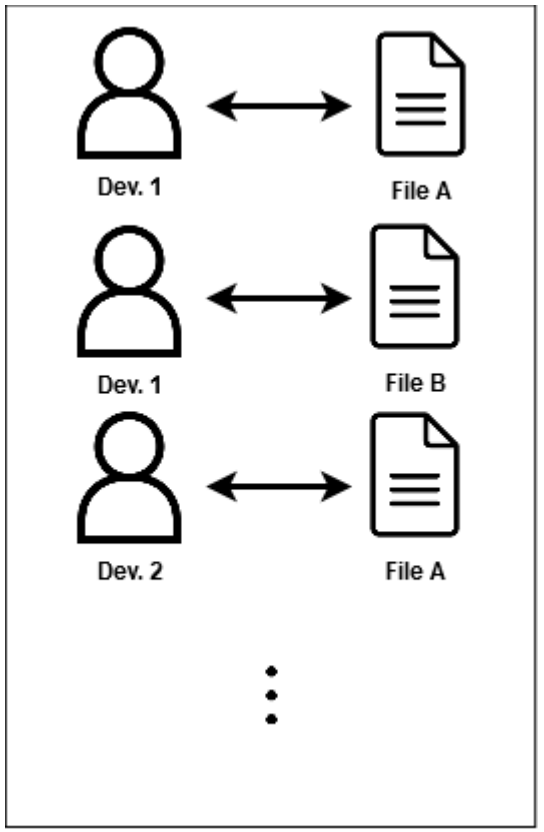
- Foram extraídos dados do histórico de desenvolvimento do *branch master/default-branch* de cada projeto.
- Foram extraídos os históricos de *commits* dos arquivos do projeto.
- De cada *commit* foram extraídos: Caminho dos arquivos, nome e email do autor do *commit*, o tipo de alteração realizada em cada arquivo.

Extraindo Histórico de Desenvolvimento

- Foram removidos arquivos que não contém código fonte.
- Foi unificado o histórico de desenvolvedores e seus *aliases* por meio de duas etapas:
  - Unificação de desenvolvedores que compartilham o mesmo email, com *usernames* distintos.
  - Unificação de desenvolvedores com emails diferentes, porém com *usernames* semelhantes.

Amostras para o *Survey*

- Pares *(dev, arquivo)* de cada projeto.
- Limite máximo de 5 arquivos por desenvolvedor.



Projetos	Pares	Desenvolvedores
Públicos	20.564	7.803
Privados	394	92

Aplicando o *Survey*

- Foi enviado um email para cada desenvolvedor das amostras geradas.
- Especificar o conhecimento de 1 a 5 em cada arquivo da sua lista.

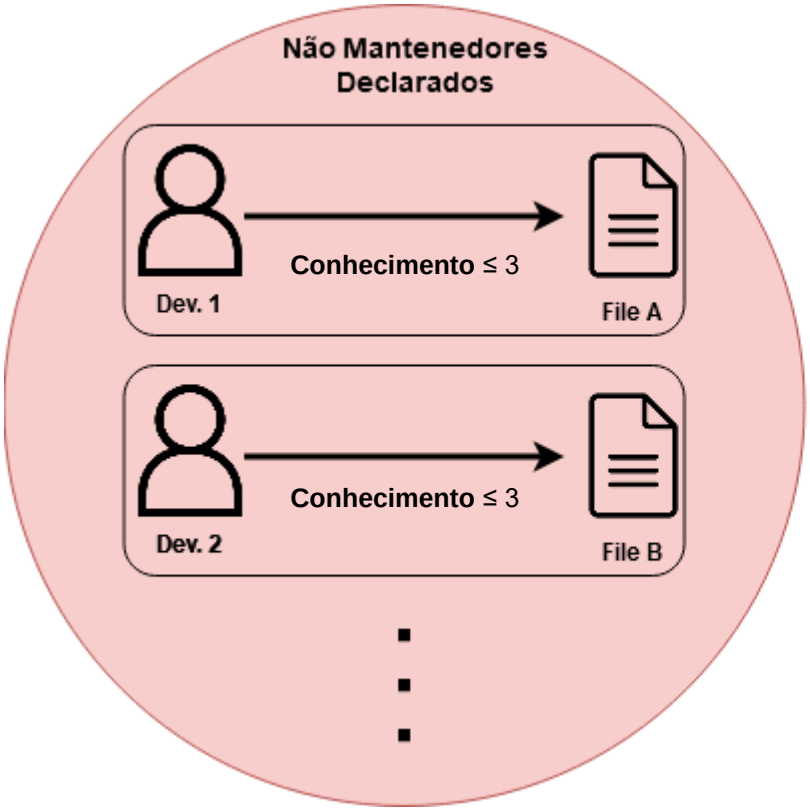
	Enviados	Respostas	Porcentagem
Públicos	7.803	501	7%
Privados	92	38	41%

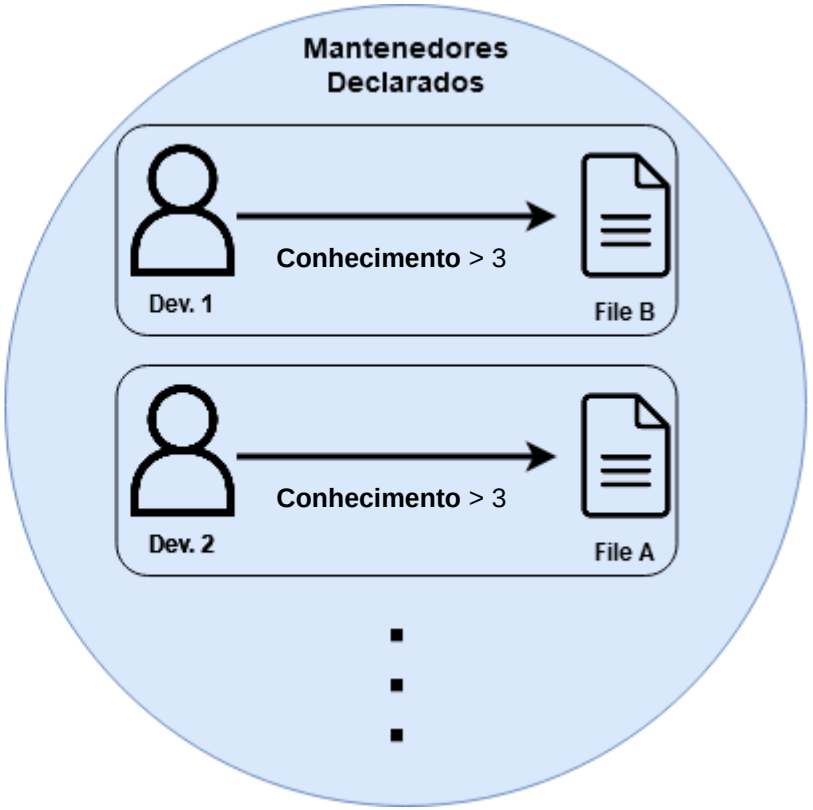
Tabela 1 – Emails enviados e repostas recebidas.

- Dois datasets foram criados:
  - Público: 1024 pares (desenvolvedor, arquivo)
  - Privado: 163 pares (desenvolvedor, arquivo)

Aplicando o *Survey*

- Classificação das respostas do *survey*:





Processando as Respostas

- Classificação das respostas do *survey*:

	Mantenedores	Não Mantenedores
Público	54%	46%
Privado	47%	53%

Extraindo e Analisando Variáveis de Desenvolvimento

Variável	Significado
Adds	Número de linhas adicionadas por um desenvolvedor *d* até a versão *v* de um arquivo *f*
Dels	Número de linhas removidas por um desenvolvedor *d* até a versão *v* de um arquivo *f*
Mods	Número de linhas modificadas por um desenvolvedor *d* até a versão *v* de um arquivo *f*
Conds	Número de condicionais adicionadas por um desenvolvedor *d* até a versão *v* de um arquivo *f*
Amount	Soma do número de linhas adicionadas e removidas por um desenvolvedor *d* até a versão *v* de um arquivo *f*
FA	Valor binário que indica se um desenvolvedor *d* adicionou o arquivo *f* ao projeto
Blame	Número de linhas adicionadas por um desenvolvedor *d* que estão na versão v de um arquivo f
NumCommits	Número de *commits* feitos por um desenvolvedor *d* no arquivo *v* até a versão *v*
NumDays	Número de dias desde o último *commit* feito pelo desenvolvedor *d* no arquivo *f* até a versão *v*
NumModDevs	Número de *commits* feitos por outros desenvolvedores no arquivo *f* desde o último *commit* de um desenvolvedor *d* no arquivo *f*
Size	Número de linhas de código (LOC) do arquivo *f*
AvgDaysCommits	Média do número de dias entre os *commits* de um desenvolvedor *d* em um arquivo *f* até a versão *v*

Tabela 3 – Variáveis extraídas dos históricos de desenvolvimento.

Extraindo e Analisando Variáveis de Desenvolvimento

QP1 – Qual das variáveis extraídas do histórico de desenvolvimento possui a maior correlação positiva com Conhecimento?

QP2 – Qual das variáveis extraídas do histórico de desenvolvimento possui a maior correlação negativa com Conhecimento?

QP3 – Quais os níveis de correlações existentes entre as variáveis extraídas?

Extraindo e Analisando Variáveis de Desenvolvimento

Variável	Direção	Corr. com Conhecimento	p-value
FA	Positiva	0,33	0
Adds	Positiva	0,31	0
Blame	Positiva	0,29	0
Amount	Positiva	0,28	0
NumDays	Negativa	0,24	0
Size	Negativa	0,21	6,313e-13
NumModDevs	Negativa	0,21	1,588e-13
AvgDaysCommits	Positiva	0,21	4,894e-13
NumCommits	Positiva	0,20	1,279e-11
Cond	Positiva	0,19	8,221e-11
Dels	Positiva	0,02	0,369
Mods	Positiva	0,01	0,515

Tabela 4 – Correlação das variáveis extraídas com *Conhecimento* utilizando o rho de *Spearman.*

Extraindo e Analisando Variáveis de Desenvolvimento

- **Correlação entre as variáveis extraídas**
  - Alta correlação positiva entre: *Adds*, *Dels*, *Mods*, e *Amount.*
  - Correlação positiva moderada entre *Adds* e *NumCommits.*
  - Correlação positiva moderada entre *NumModDevs* e *NumDays.*

Técnicas Comparadas

- **Commits**
  - Número de commits (*NumCommits*) como medida de conhecimento.
  - Quanto maior o número de commits em um arquivo, maior o conhecimento do desenvolvedor.
  - Utilizada tanto de forma atômica, quanto na composição de técnicas.
- **Blame**
  - Conhecimento inferido pela quantidade de linhas adicionadas presentes na última versão do arquivo.

Técnicas Comparadas

- ***Degree of Authorship* (DOA)**
  - O conhecimento é inferido por fatores como: autoria do arquivo, número de contribuições, número de mudanças feitas por outros desenvolvedores (FRITZ et al., 2014).

$$\textbf{DOA(d, f(v))} = 3{,}293 + 1{,}098 * FA + 0{,}164 * DL - 0{,}321 * ln(1 + AC)$$

- *FA*: 1 se o desenvolvedor *d* é o criador do arquivo *f*, 0 caso contrário
- *DL*: número de mudanças feitas por um desenvolvedor *d* até a versão *v* de um arquivo *f*
- *AC*: número de mudanças feitas por outros desenvolvedores no arquivo *f*

Técnicas Comparadas

- **Nível de Expertise**
  1. As variáveis *Size* e *NumDays* foram incluídas no modelo.
  2. Entre *Adds*, *Amount, Blame, NumCommits* foi escolhida a variável *Adds.*
  3. Foi incluída a variável *FA* no modelo.
- O modelo final foi formando pelas variáveis: *Adds*, *FA*, *NumDays*, *Size.*

Técnicas Comparadas

- O **Conhecimento** de um desenvolvedor *d* na versão *v* de um arquivo *f* é dado por:

$$\textbf{Conhecimento(d, f(v))} = C + b_0 * ln(1 + Adds^{d,\, f(v)}) + b_1 * (FA^f) - b_2 * ln(1 + NumDays^{d,\, f(v)}) - b_3 * ln(Size^{f(v)})$$

- Utilizamos regressão linear múltipla nas avaliações.
- Na apresentação dos resultados esse modelo é referenciado por **NE.**

Técnicas Comparadas

- **CoDiVision**
- Modelo linear para inferência da familiaridade de um desenvolvedor com um arquivo (LIRA, 2016).
- Quantidade total de alterações:

$$\mathbf{W(d, f(v))} = (b_0 * Adds^{d,\, f(v)}) + (b_1 * Mods^{d,\, f(v)}) + (b_2 * Dels^{d,\, f(v)}) + (b_3 * Conds^{d,\, f(v)})$$

- Degradação do conhecimento por tempo decorrido desde a última modificação:

$$\mathbf{T(d, f(v))} = 1 - (b_4 * NumDays)$$

Técnicas Comparadas

- Degradação do conhecimento por mudanças realizadas por outros desenvolvedores desde a última modificação:

$$\textbf{N(d, f(v))} = 1 - (b_5 * \mathit{NumModDevs})$$

- Familiaridade de um desenvolvedor *d* na versão *v* de um arquivo *f*:

$$\textbf{Familiaridade(d, f(v))} = W(d, f(v)) * T(d, f(v)) * N(d, f(v))$$

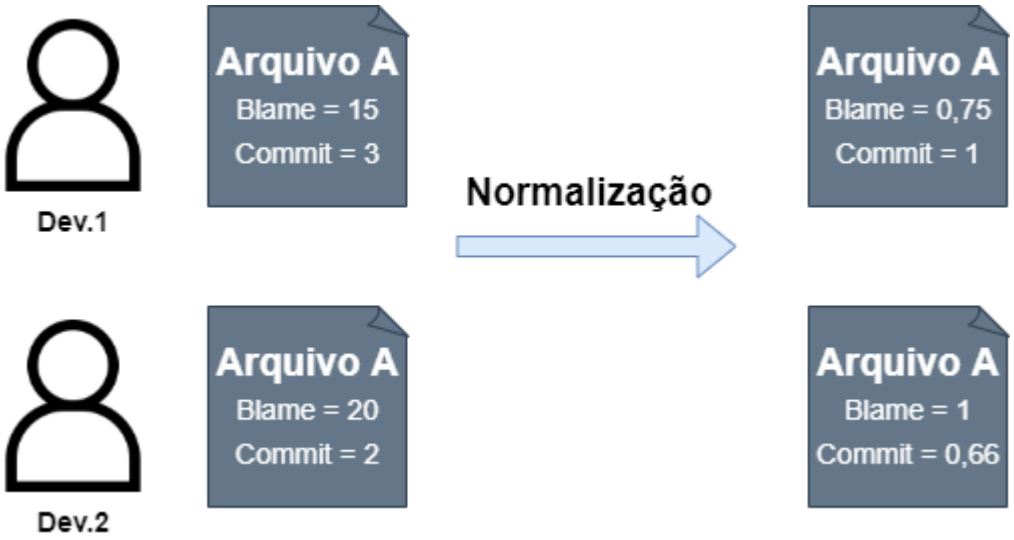
- Utilizamos regressão linear múltipla nas avaliações.

Técnicas Comparadas

- **Classificadores de Aprendizagem de Máquina**
  - Random Forest
  - Support Vector Machines (**SVM**)
  - K-Nearest Neighbors (**KNN**)
  - Gradient Boosting Machine (**GBM**)
  - Logistic Regression.

Identificando Mantenedores - Técnicas Lineares

- Os valores de cada métrica foram normalizadas para cada par *(desenvolvedor, arquivo)* nos dois *datasets*.



Identificando Mantenedores - Técnicas Lineares

- Foi utilizado um *threshold k* para classificação em mantenedores

- No exemplo, para um *k* = 0,7:
  - Dev. 1 é um mantenedor do *Arquivo A* de acordo com *Blame* e *Commit*
  - Dev. 2 é um mantenedor do *Arquivo A* de acordo com *Blame*

Avaliando Performance

- **Técnicas Lineares:**
  - *Threshold K* foi variado de 0,1 a 1, usando passos de 0,1. A cada passo foi computado o *F-Measure*.
  - Para os dois modelos de regressão linear foi executado *10-fold Cross Validation* em cada *threshold k.*
- **Classificadores de Aprendizagem de Máquina:**
  - *10-fold Cross Validation.*
  - Busca em grade para achar melhores configurações.

Avaliando Performance

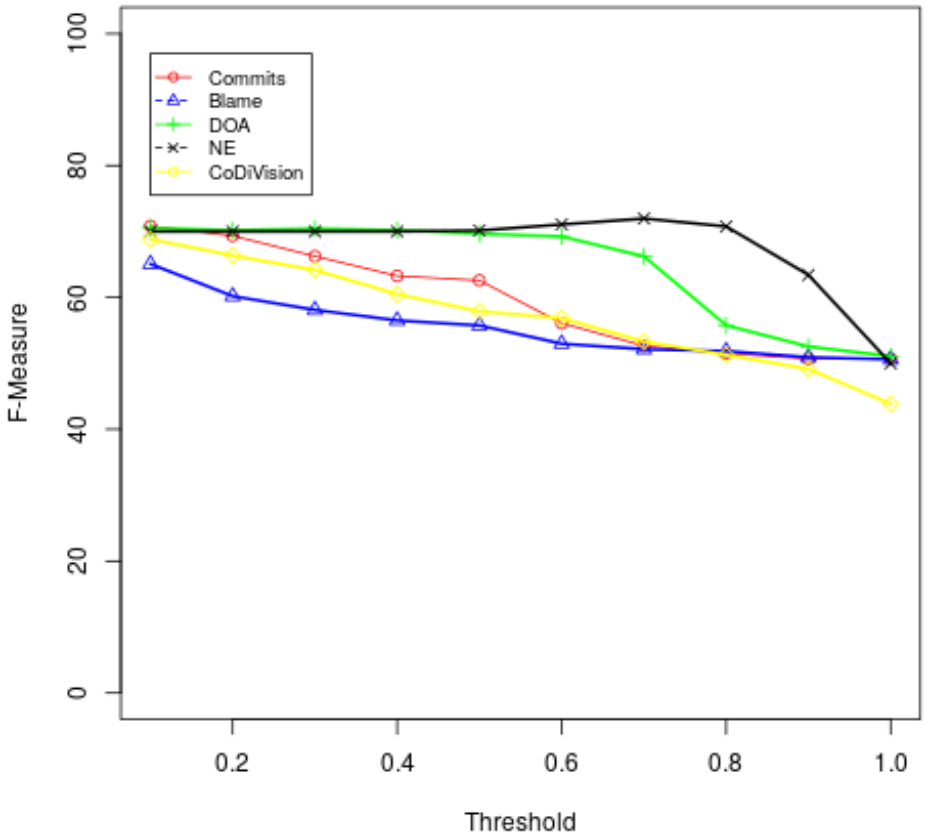
- **Métricas Utilizadas:**
  - *Precision* = proporção de desenvolvedores classificados como mantenedores que são efetivamente mantenedores (*Conhecimento* > *3*).
  - *Recall* = proporção de mantenedores identificados corretamente.
  - *F-Measure* = média harmônica entre *precision* e *recall*.

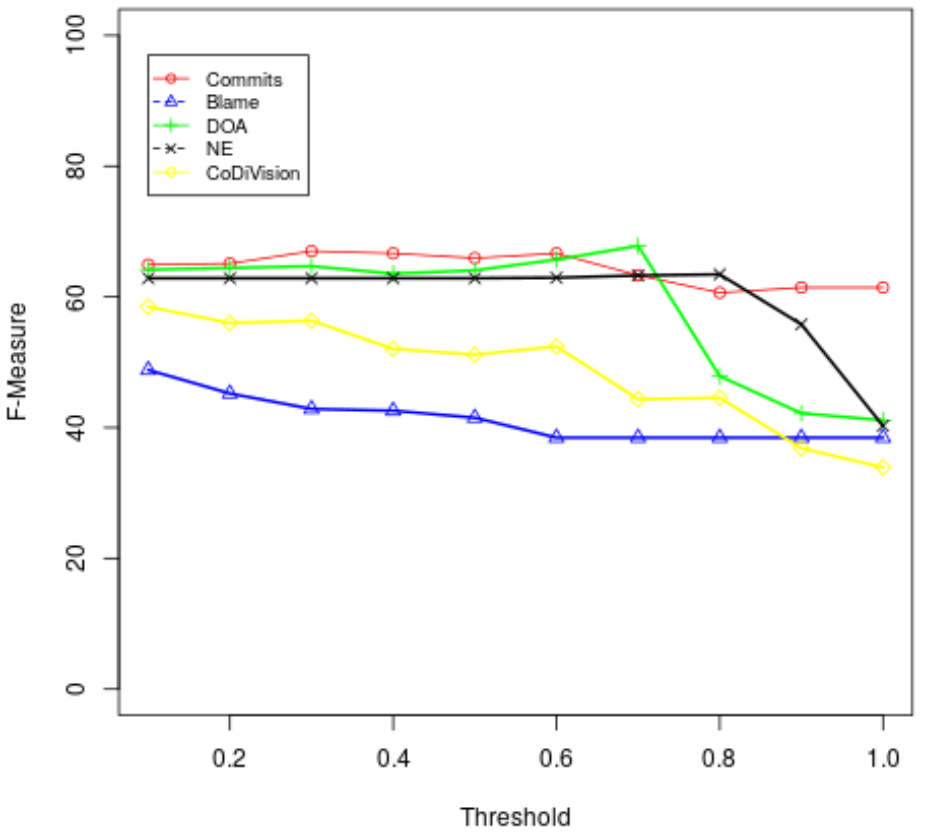
- **AVALIAÇÃO:**
  - **Resultados**
  - **Discussão**
  - **Ameaças à Validade**

Resultados e Discussão

- Análise em dois cenários:
  - *Dataset* público
  - *Dataset* privado
- Técnicas comparadas nas suas melhores configurações.

Melhores Configurações das Técnicas Lineares





Melhores Configurações das Técnicas Lineares

Melhores *Thresholds*		
	Dados Públicos	**Dados Privados**
NE	0,7	0,8
CoDiVision	0,1	0,1
DOA	0,1	0,7
NumCommits	0,1	0,3
Blame	0,1	0,1

Resultados e Discussão

Questões de pesquisa:

- **QP4** – Qual técnica linear apresenta a melhor performance na identificação correta de mantenedores de arquivos?
- **QP5** – Qual classificador de aprendizagem de máquina apresenta a melhor performance na identificação correta de mantenedores de arquivos?
- **QP6** – As técnicas propostas neste trabalho trouxeram melhorias na identificação de mantenedores de arquivo?

Resultados e Discussão

- Performances das técnicas lineares:

	Público			Privado		
	Precision	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**
NE	0,60	0,91	0,72	0,50	0,95	0,64
DOA	0,55	0,97	0,70	0,61	0,77	0,68
NumCommits	0,60	0,87	0,70	0,55	0,86	0,67
CoDiVision	0,55	0,91	0,69	0,45	0,84	0,58
Blame	0,69	0,62	0,65	0,62	0,40	0,49

Resultados e Discussão

- *Nível de Expertise* não trouxe uma melhora significativa na identificação de mantenedores.
- A técnica *CoDiVision* também não obteve um desempenho satisfatório.
- *Precision* baixo e *Recall* alto foi obtido por todas as técnicas lineares nos três cenários, com exceção de *Blame*.

Resultados e Discussão

- Performances dos classificadores de aprendizagem de máquina :

	Público			Privado		
	Precision	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**
Random Forest	0,73	0,74	0,73	0,59	0,62	0,59
SVM	0,71	0,75	0,73	0,63	0,60	0,61
KNN	0,71	0,71	0,71	0,70	0,69	0,67
GBM	0,73	0,73	0,73	0,65	0,58	0,60
Logistic Regression	0,69	0,75	0,72	0,70	0,51	0,58

Resultados e Discussão

- Os classificadores de aprendizagem de máquina obtiveram um melhor balanceamento entre *Precision* e *Recall*.
- Os classificadores alcançaram os melhores *Precision* em todos os cenários analisados.

Resultados e Discussão

- **QP4** – Qual técnica linear apresenta a melhor performance na identificação correta de mantenedores de arquivos?
  - **Resposta –** Dados Públicos: *Nível de Expertise*; Dados Privados: *DOA.*
- **QP5** – Qual classificador de aprendizagem de máquina apresenta a melhor performance na identificação correta de mantenedores de arquivos?
  - **Resposta –** Dados Públicos: *Random Forest*, *Support Vector Machines* e *Gradient Boosting Machines*; Dados Privados: *K Nearest Neighbors*.
- **QP6** – As técnicas propostas neste trabalho trouxeram melhorias na identificação de mantenedores de arquivo?
  - **Resposta:** As técnicas propostas alcançaram os melhores *F-Measures* nos cenários analisados.

Resultados e Discussão

- Qual a melhor técnica para a identificação de mantenedores de arquivos?

Ameaças à Validade

- Auto avaliação de conhecimento inflada.
- Risco de má interpretação da escala de conhecimento (1 a 5) utilizada no *survey*.
- Definição de *expert* no *survey*.
- Definição de mantenedor utilizada nas avaliações.
- Outros fatores influenciam o conhecimento de um desenvolvedor em um arquivo de código fonte.
- Outras variáveis podem ser extraídas de SCV.

CONCLUSÃO:

- Principais Achados
- Desafios e Limitações
- Trabalhos Futuros

Principais Achados

- Sobre correlações entre as variáveis estudadas:
  - *Primeira autoria* demonstrou a maior correlação positiva com conhecimento em arquivos de código fonte.
  - Das variáveis que representam as mudanças, a variável *Número de linhas adicionadas* apresentou a maior correlação positiva com conhecimento.
  - *Recência da Modificação* apresentou a maior correlação negativa com conhecimento em arquivos de código fonte.

Principais Achados

- Sobre a performance das técnicas comparadas:
  - *Nível de Expertise* obteve o maior *F-Measure* entre as técnicas lineares.
  - Os classificadores de aprendizagem de máquina tiveram os melhores *Precisions* em todos os três cenários analisados, e alcançaram os melhores *F-Measures*.

Desafios e Limitações

- Aplicabilidade das técnicas propostas:

Revisão de código

Resolução de bugs

Desafios e Limitações

- Implementações em ferramentas:

Trabalhos Futuros

- Extração de outras informações presentes em SCV:
  - Arquivos relacionados
  - Caminho de arquivos no projeto
  - Complexidade do código
- Outras fontes de dados:
  - Repositório de *bugs*
  - Fóruns de programação
- Aplicação em contextos reais.

Avaliação de Técnicas para a Identificação de Mantenedores de Código Fonte

Otávio Cury da Costa Castro

Orientador: Pedro de Alcântara dos Santos Neto

Coorientador: Guilherme Amaral Avelino