MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇÃO

PROPOSTA DE DISSERTAÇÃO DE MESTRADO

FÁBIO DO ROSARIO SANTOS

# UMA ABORDAGEM PARA DETECÇÃO E CLASSIFICAÇÃO DE GRAVIDADE DE *CODE SMELL* BASEADA EM TRANSFERÊNCIA DE APRENDIZADO

**Rio de Janeiro**
**2024**

# 1 TÍTULO DA PROPOSTA DE DISSERTAÇÃO

**Título da Proposta de Dissertação:**
Uma abordagem para detecção e classificação de gravidade de *code smell* baseada em transferência de aprendizado

**Área de Concentração:**
Ciência da Computação

**Linha de Pesquisa:**
Engenharia de Sistemas e Informação

# 2 INTRODUÇÃO

*Code smell* refere-se a um sintoma de código inserido em um programa devido a falhas de *design* ou práticas de codificação inadequadas, podendo levar a problemas de qualidade de software e impedir o desenvolvimento e a manutenção de sistemas de software. O conceito de *code smell* foi inicialmente introduzido por Fowler et al. (1999), i.e., indicadores primários de dívidas técnicas no código-fonte, destacando áreas que necessitam de manutenção completa. Alguns fatores contribuem para o surgimento dessas dívidas técnicas, como prazos rigorosos e introdução de novos requisitos durante o desenvolvimento de software (FOWLER et al., 1999).

Diversas ferramentas para detecção de *code smell*, incluindo tanto ferramentas comerciais quanto protótipos de pesquisa, foram propostas. Essas ferramentas adotam diversas técnicas para identificar *code smell*: algumas baseiam-se em métricas (LANZA; MARINESCU, 2007; FONTANA et al., 2015), enquanto outras empregam uma linguagem de especificação dedicada (MOHA et al., 2010), análise de programa para identificar oportunidades de refatoração (TSANTALIS; CHATZIGEORGIOU, 2009, 2011), análise de repositórios de software (PALOMBA et al., 2015), ou técnicas de aprendizado de máquina (ARCELLI FONTANA et al., 2016). A maioria dessas abordagens utiliza heurísticas e diferencia artefatos de código afetados (ou não) por um determinado tipo de *code smell*, aplicando regras de detecção que comparam os valores de métricas relevantes extraídas do código-fonte com limites identificados empiricamente (PECORELLI et al., 2019b). No entanto, a abordagem baseada em aprendizado de máquina possui a vantagem de poder explorar qualquer característica do código que um desenvolvedor considere importante para definir, e.g., a gravidade de um *code smell*, na medida em que essa característica pode ser estimada por meio de uma ou mais métricas de software (ARCELLI FONTANA; ZANONI, 2017).

A detecção de *code smell* é um processo baseado em dados para garantia de qualidade de código que visa detectar se um determinado trecho de código apresenta violação de princípios fundamentais de *design*. O conjunto de dados rotulado manualmente de Madeyski e Lewowski (2020), *Madeyski Lewowski Code Quest* (MLCQ), é reconhecido como o mais abrangente em relação a diferentes aspectos, como tamanho e qualidade das amostras (ZAKERI-NASRABADI et al., 2023). Entretanto, alguns trechos de código foram rotulados por múltiplos engenheiros, e é fortemente alinhado com o paradigma descritivo,

encorajando a subjetividade dos anotadores. Diferentemente, o paradigma prescritivo, centrado em dados, desencoraja essa subjetividade e visa a consistência na detecção de *code smell*. Autores como Arcelli Fontana et al. (2016), Nanda e Chhabra (2022) e Slivka et al. (2023) adotaram o paradigma prescritivo, estabelecendo critérios claros para a anotação de *code smells* em conjuntos de dados. Esta área de pesquisa cresceu recentemente em popularidade dentro da comunidade de pesquisa, havendo um aumento significativo no interesse por este problema de investigação, com uma taxa de crescimento anual atingindo 34,95%, conforme mostrado na Tabela 2.1, com as principais informações da nossa Revisão Sistemática da Literatura (RSL), que é detalhada na Seção 3.1.

TAB. 2.1: Principais Informações da RSL

| | |
|---|---|
| Período | 2015:2023 |
| Fontes | 56 |
| Documentos | 69 |
| Taxa de Crescimento Anual | 34,95% |

No entanto, a detecção de *code smell* por meio do Aprendizado de Máquina (AM) enfrenta desafios inerentes à sua natureza desequilibrada e à propensão para vieses de interpretação (PECORELLI et al., 2019b). Isto requer uma análise cuidadosa e melhoria dos mecanismos internos do modelo de previsão antes de interpretar os resultados gerados. Ao abordar a questão dos *code smells*, os pesquisadores empregaram uma ampla gama de modelos de AM, e.g., baseados em árvores de decisão, aprendizes de regras, máquinas de vetores de suporte, redes bayesianas (ARCELLI FONTANA et al., 2016; DI NUCCI et al., 2018; PECORELLI et al., 2019b; DEWANGAN et al., 2021), e também baseados em regressão logística e redes neurais (DEWANGAN et al., 2021). Estes modelos visam não só detectar e identificar *code smells*, mas também avaliar a sua gravidade (ARCELLI FONTANA; ZANONI, 2017; RAO et al., 2023). Assim, a classificação da gravidade dos *code smell* representa uma área de estudo crucial, pois categoriza os problemas associados a este domínio, permitindo a priorização de custos e esforços de manutenção de software com base na gravidade enfrentada, o que contribui significativamente para o ciclo de vida do software.

Inicialmente os estudos utilizavam abordagens baseadas em aprendizado de máquina supervisionada para para detectar *code smells*. Normalmente, um método supervisionado explora um conjunto de variáveis independentes para determinar o valor de uma variável dependente (i.e, presença de um *code smell* ou de sua gravidade em um trecho de código) usando um único método de aprendizado de máquina (DI NUCCI et al., 2018).

Entretanto, com os avanços dos últimos anos das técnicas e algoritmos de aprendizado de máquina, outras abordagens passaram a ser consideradas nesta área de pesquisa, como métodos de comitê (DEWANGAN et al., 2023) e transferência de aprendizado (SHARMA et al., 2021).

Os métodos de comitê desempenham um papel crucial no aprimoramento do desempenho de modelos de aprendizado de máquina para detecção de *code smell*. A ideia central por trás desses métodos é a combinação de vários classificadores para obter resultados superiores em comparação com um único classificador. Esta abordagem está bem estabelecida e tutoriais foram fornecidos para orientar profissionais interessados na construção de sistemas de classificação baseados em comitês (POLIKAR, 2006; ROKACH, 2010). Nesse sentido, a adoção desses métodos para melhorar modelos de aprendizado de máquina para detecção de *code smell* tem sido amplamente sugerida por diversos autores (PECORELLI et al., 2019b; AZEEM et al., 2019). Os métodos de comitê podem ser categorizados como heterogêneos ou homogêneos. Por um lado, os heterogêneos utilizam diferentes tipos de classificadores. Por outro lado, os homogêneos são construídos com classificadores do mesmo tipo, treinados em diferentes subconjuntos do conjunto de dados (ALAZBA; ALJAMAAN, 2021).

A Transferência de Aprendizado (TA) é uma técnica na qual um algoritmo de aprendizado aproveita pontos em comum entre diferentes tarefas de aprendizado para facilitar a transferência de conhecimento entre elas (BENGIO et al., 2013). Esta abordagem pode ser categorizada em dois tipos com base na comparabilidade de seus domínios: transferência de aprendizado homogênea ou heterogênea (WEISS et al., 2016). O domínio de origem e o domínio de destino compartilham o mesmo conjunto de atributos na transferência de aprendizado homogênea. Já a transferência de aprendizado heterogênea, ocorre quando os domínios de origem e de destino possuem, cada um, conjuntos distintos de atributos.

## 2.1 JUSTIFICATIVA

O Escritório de Projetos e Sistemas Digitais do Corpo de Fuzileiros Navais da Marinha do Brasil (EPSD-CFN-MB) é responsável pela manutenção dos Softwares Corporativos do CFN. Entretanto, o principal objetivo do CFN não é desenvolver ou manter softwares. Por isso, o Escritório conta com um número restrito de desenvolvedores, limitando assim o efetivo disponível para essa atividade. Diante desse cenário, torna-se essencial contar com ferramentas baseadas em modelos de AM que possam auxiliar em tarefas que são desafiadoras para a equipe de manutenção, como a detecção e classificação da gravidade de

*code smell*. Isso assegura que a equipe de desenvolvimento seja alocada de forma eficiente para abordar os problemas mais relevantes na manutenção dos Softwares Corporativos do CFN. Portanto, a classificação da gravidade de *code smell* é uma área de estudo fundamental, pois permite categorizar os problemas relacionados a este domínio e facilita a priorização de recursos e esforços de manutenção com base na gravidade das questões enfrentadas, o que é essencial para o ciclo de vida do software. Além disso, é importante ressaltar que essa abordagem pode ser aplicada tanto nos softwares corporativos do CFN quanto em qualquer outro projeto de software desenvolvido na linguagem CSharp.

O estudo de Arcelli Fontana e Zanoni (2017) foi pioneiro ao buscar classificar a gravidade de *code smell* baseado em Aprendizado de Máquina, estabelecendo um conjunto de dados fundamental para pesquisas subsequentes. Nos últimos dois anos, observou-se um aumento no interesse por esse tema entre os pesquisadores, refletido na realização de três estudos em 2022 (NANDA; CHHABRA, 2022; ABDOU; DARWISH, 2022; GUPTA; CHAUHAN, 2022) e mais três em 2023 (RAO et al., 2023; HU et al., 2023; DEWANGAN et al., 2023).

Neste contexto, os dados usados para treinar um modelo de AM são o componente mais importante desse processo. Embora a maioria dos estudos tenha relatado o pré-processamento e a qualidade dos dados como questões significativas para esta área de pesquisa (AZEEM et al., 2019; ALAZBA; ALJAMAAN, 2021; ZAKERI-NASRABADI et al., 2023), estes estudos não realizaram análise aprofundada do pré-processamento de dados. Os estudos de investigação anteriormente referenciados, centrados na gravidade, foram orientados principalmente para categorizar a gravidade dos *code smells* em conjuntos de dados que compreendem instâncias de apenas um tipo específico de *code smell*. Esta orientação separa estas abordagens dos cenários do mundo real no software existente, uma vez que negligenciam a influência que diferentes tipos de *code smells* podem exercer uns sobre os outros.

## 2.2 ESTRUTURA DO TEXTO

O Capítulo 2 apresenta o tema e a justificativa da proposta. O Capítulo 3 apresenta a revisão e o estado-da-arte para a detecção e classificação de gravidade de *code smell*. No Capítulo 4 são discutidos o problema e os objetivos desta proposta. Já o Capítulo 5, descreve a proposta, as hipóteses e as contribuições esperadas. No Capítulo 6, o processo metodológico para a realização das atividades é detalhado, acompanhado do plano de ação e cronograma a ser seguido.

# 3 REVISÃO E ESTADO DA ARTE

Neste capítulo, é apresentada a Revisão Sistemática da Literatura (RSL) com foco no pré-processamento de dados para detecção de *code smell* baseado em AM (Seção 3.1). Decorrente do resultado desta revisão, as técnicas de pré-processamento de dados mais usadas em conjuntos de dados de *code smells* existentes são detalhadas na Seção 3.1.1. Além disso, também são apresentados os resultados da relação entre técnicas de AM utilizadas juntamente com técnicas de pré-processamento de dados (Seção 3.1.2). Na Seção 3.2, são apresentados os paradigmas de anotação e os conjuntos de dados que servirão de base para detecção e classificaçao da gravidade de *code smell*. E, por fim, é discutido o estado da arte quanto a detecção e classificação de gravidade de *code smell* baseado em AM (Seção 3.3).

## 3.1 REVISÃO

Para obter uma melhor compreensão das abordagens de pré-processamento de dados para esse tipo de detecção, conduzimos uma RSL, na qual foram examinados e sintetizados empiricamente as práticas e soluções atuais para pré-processamento de dados de *code smell*. Para conduzir esta RSL, seguimos a orientação metodológica fornecida por Kitchenham e Charters (2007) para ajudar a manter a integridade e a credibilidade do processo de revisão sistemática, levando a resultados mais robustos e confiáveis. Para orientar nossa análise, abordamos as duas Questões de Pesquisa (QPs) apresentadas abaixo:

- QP1 - Quais técnicas de pré-processamento de dados foram aplicadas em modelos de Aprendizado de Máquina para detecção de *code smell*?
- QP2 - Quais técnicas de AM para detecção de *code smell* foram usadas com técnicas de pré-processamento de dados?

### 3.1.1 TÉCNICAS DE PRÉ-PROCESSAMENTO DE DADOS (QP1)

O pré-processamento de dados destaca-se como um aspecto importante na construção de modelos de aprendizado de máquina de alto desempenho. Este processo envolve a limpeza e a transformação de dados brutos para facilitar a análise e a identificação das informações mais relevantes, melhorando em última análise o desempenho do modelo

(PARASHAR et al., 2023). Na última década, os pesquisadores se concentraram nisso em duas facetas principais: balanceamento de dados e técnicas de seleção de atributos. O primeiro aborda o desafio colocado pela natureza desequilibrada dos conjuntos de dados disponíveis, enquanto o último visa identificar as principais características que influenciam a detecção de *code smells*.

#### 3.1.1.1 TÉCNICAS DE BALANCEAMENTO DE DADOS

Dados desequilibrados podem ser resolvidos transformando o conjunto de treinamento ou usando metaclassificadores, como classificadores sensíveis ao custo (PECORELLI et al., 2019a). *Synthetic Minority Oversampling Technique (SMOTE)* e balanceador de classe foram as técnicas de balanceamento de dados mais utilizadas, implementadas em 20 e 9 estudos, respectivamente.

- ***Synthetic Minority Oversampling Technique* (SMOTE)** é um dos quase 100 algoritmos que integram as técnicas de Sobreamostragem. Essas técnicas buscam resolver o desequilíbrio nos conjuntos de dados, aumentando o tamanho da amostra do grupo minoritário, enquanto todo o conteúdo e qualidade das amostras permanecem inalterados. Entretanto, no caso de grandes conjuntos de dados, os custos de tempo e memória para a fase de classificação podem aumentar significativamente (CHAWLA et al., 2002). A técnica SMOTE se destaca dentre elas, por ser um método estatístico projetado para lidar com conjuntos de dados desequilibrados, gerando instâncias adicionais de maneira equilibrada. Esta técnica sintetiza novos pontos de dados com base em instâncias minoritárias existentes fornecidas como entrada (MALHOTRA et al., 2023).
- **O Balanceador de Classe** repondera as instâncias do conjunto de treinamento para que a soma dos pesos de cada classe de instância no conjunto de dados seja igual (HALL et al., 2009).

#### 3.1.1.2 TÉCNICAS DE SELEÇÃO DE ATRIBUTOS

Identificar os atributos mais impactantes em um conjunto de dados é o objetivo da seleção de atributos, e.g., melhorando o desempenho com foco nas métricas de software. Isso desempenha um papel significativo na distinção entre funções semelhantes em padrões de projeto (ROMERO; SOPENA, 2008). Quanto à seleção de atributos, destacam-se técnicas como *Autoencoder* em 5 estudos e Qui-quadrado, Taxa de Ganho, Ganho de Informação

e Análise de Componentes Principais (PCA) em 4 estudos, cada. Entretanto, algumas técnicas foram poucos empregadas, como LDA, em apenas 1 estudo.

- **Autoencoder** é uma técnica de aprendizado não supervisionado que usa redes neurais para aprendizado de representação, copiando entradas em saídas. Ele comprime a entrada em uma representação de espaço latente e então reconstrói a saída. Além disso, um *autoencoder* variacional é uma versão probabilística de um *autoencoder* típico e um modelo generativo profundo que estima a representação latente usando métodos de inferência bayesiana (HADJ-KACEM; BOUASSIDA, 2019).

- **Qui-quadrado** é uma técnica de seleção de atributos que é aplicada a atributos categóricos em um conjunto de dados. Ele calcula o qui-quadrado entre cada métrica e seleciona o número desejado de métricas com as melhores pontuações. As melhores métricas são escolhidas em cada conjunto de dados de *code smell* usando esta técnica (DEWANGAN et al., 2021). Equação 3.1 é a fórmula para o teste Qui-quadrado (DEWANGAN et al., 2023):

$$X^2 = \frac{(FrequenciaObservada - FrequenciaEsperada)^2}{FrequenciaEsperada} \tag{3.1}$$

- **Algoritmo taxa de ganho** é empregado para identificar os atributos que têm maior peso na detecção de code smell. A saída do algoritmo é uma lista classificada, com métricas com maior taxa de ganho recebendo mais peso, sendo assim classificadas em primeiro lugar na previsão (DI NUCCI et al., 2018).

- **Algoritmo de ganho de informação.** Aljamaan (2021), Kaur e Kaur (2021) e Tarwani e Chug (2022) utilizaram esta técnica, que é a quantidade esperada de informação obtida ao dividir o conjunto de dados com base em uma variável específica. É comumente aplicado no aprendizado de árvore de decisão, onde cada variável é avaliada e a variável com maior ganho de informação é selecionada.

- **Análise de Componentes Principais (PCA)** é uma técnica de aprendizado de máquina não supervisionado que determina as características mais significativas reduzindo a dimensionalidade dos dados (PATNAIK; PADHY, 2022).

- **Análise Discriminante Linear (LDA)** é um algoritmo de aprendizado de máquina supervisionado especializado em identificar características distintivas entre diferentes grupos ou classes (PATNAIK; PADHY, 2022).

#### 3.1.1.3 FILTRAGEM

A filtragem remove o ruído dos dados (PARASHAR et al., 2023). Apesar do uso desta técnica em diversos estudos, muitos pesquisadores a aplicaram sem maiores considerações, e.g., Abdou e Darwish (2022) e Madeyski e Lewowski (2020). No entanto, Liu et al. (2021) concluiu que a filtragem de dados de treinamento estranhos diminuiu o número de *code smell* identificados com sucesso, diminuindo a capacidade de generalização dos classificadores resultantes. Consequentemente, observaram que os algoritmos de filtragem, que dependem de métricas de código comuns e simples, garantem melhorias substanciais. Contudo, algoritmos de filtragem mais avançados poderiam melhorar a qualidade dos dados de treinamento gerados sem comprometer significativamente a generalização dos classificadores treinados nesses dados.

#### 3.1.1.4 ESCALONAMENTO DE DADOS

O escalonamento de dados é uma etapa de pré-processamento no aprendizado de máquina que visa padronizar o intervalo de variáveis independentes ou características do conjunto de dados. Este processo envolve a transformação dos valores dos atributos para um intervalo específico, normalmente entre 0 e 1 ou -1 e 1, para garantir que tenham escalas semelhantes.

- **Normalização de dados.** A técnica de normalização Min-Max é empregada no estágio de pré-processamento de dados para prepará-los para uso por diversas técnicas de aprendizado de máquina, como Máquinas de Vetores de Suporte e Redes Neurais (MUHAMMAD ALI; FARAJ, 2014). A equação 3.2 é utilizada para mudar e redimensionar os valores de um atributo (X) para que fiquem dentro do intervalo de 0 e 1. Os dados reformados (X') resultantes deste processo de normalização são então considerados como entrada para técnicas de aprendizado de máquina.

$$X' = \frac{X - Xmin}{Xmax - Xmin} \tag{3.2}$$

- **Padronização de dados.** De acordo com Afrin et al. (2022), a padronização é uma técnica que reduz os valores dos atributos para um intervalo específico com base na distribuição normal padrão, onde a média é 0 e o desvio padrão é 1. Além disso, a padronização transforma todos os atributos em um formato de unidade, tornando mais fácil para as máquinas aprenderem com mais precisão a partir de um conjunto de dados. A equação 3.3 descreve a padronização, onde Z é a variável padronizada,

X é a variável original, $\mu$ é a média da variável original, e $\sigma$ é o desvio padrão da variável original.

$$Z = \frac{X - \mu}{\sigma} \tag{3.3}$$

Da mesma forma que a filtragem, muitos pesquisadores aplicaram essas técnicas sem maiores discussões em seus estudos (DEWANGAN et al., 2021; ALKHARABSHEH et al., 2022; DEWANGAN et al., 2023). No entanto, vale a pena notar que Arcelli Fontana e Zanoni (2017) experimentaram diversas abordagens de normalização nos conjuntos de dados durante a fase inicial de exploração do seu estudo. Entretanto, apesar dos seus esforços, não observaram resultados significativamente diferentes, particularmente em termos de classificação dos classificadores. Isso destaca possibilidades para a melhoria dos conjuntos de entradas para modelos de aprendizado de máquina.

#### 3.1.1.5 TOKENIZAÇÃO

Tokenização é o processo de dividir uma sequência de caracteres em uma sequência de *tokens* (ALAZBA et al., 2023). De acordo com Yu et al. (2021), enfrentar os desafios de capturar informações sintáticas e semânticas incorporadas em códigos-fonte emergiu como um novo gargalo no desenvolvimento de soluções baseadas em modelos de aprendizado profundo. A maioria desses modelos opera em um paradigma baseado em *tokens*, emprestado de modelos de processamento de linguagem natural. Eles geram uma *string* digital exclusiva para um trecho de código com base na semântica do código, utilizando corpora de texto em linguagem natural. Apesar das semelhanças entre o código-fonte e a linguagem natural, existem diferenças significativas. O código-fonte abrange atributos estruturais intrincados que podem ser perdidos ao tratá-lo apenas como texto simples.

#### 3.1.1.6 IMPUTAÇÃO DE VALORES FALTANTES

A falta de dados é um processo que dificulta a extração de informações importantes presentes nos bancos de dados e limita o bom funcionamento dos algoritmos de aprendizado de máquina. Em relação à detecção de *code smell*, alguns estudos utilizaram a média para imputar valores faltantes (ALAZBA; ALJAMAAN, 2021; ALJAMAAN, 2021; ALKHARABSHEH et al., 2022; GUPTA; SINGH, 2023), enquanto outros optaram por preenchê-los com valor zero (AFRIN et al., 2022; HO et al., 2023). No entanto, os pesquisadores fizeram apenas uma menção superficial ao seu uso.

#### 3.1.1.7 ÁRVORE DE SINTAXE ABSTRATA (AST)

AST é usado para representar a estrutura sintática abstrata dos códigos-fonte. As informações lexicais e a estrutura sintática do código-fonte são representadas por nós e seus relacionamentos nesta técnica (YU et al., 2021). Em contraste com o texto serializado da linguagem natural, a semântica dos códigos-fonte é mais padronizada, embora relativamente frágil. Essa fragilidade surge de mudanças sutis na sintaxe, que podem levar a alterações drásticas no significado do código. Experimentos comparativos entre modelos baseados em árvores e baseados em *tokens* indicam uma notável melhoria de desempenho com a incorporação de informações estruturais. Entretanto, ao contrário da linguagem natural, extrair características semânticas válidas de blocos de código-fonte pode ser um desafio devido a códigos não padronizados em projetos práticos.

Nesse sentido, as rígidas regras de sintaxe das linguagens de programação oferecem uma vantagem: informações estruturais implícitas entre códigos, regidas por essas regras de sintaxe, facilitam a extração de características significativas para redes neurais. Vale ressaltar que em projetos práticos, onde prevalecem códigos não padronizados, extrair características semânticas válidas de blocos de código-fonte pode ser mais desafiador em comparação com a linguagem natural.

### 3.1.2 TÉCNICAS DE AM USADAS COM TÉCNICAS DE PRÉ-PROCESSAMENTO DE DADOS (QP2)

Após o pré-processamento dos dados, a seleção da técnica de aprendizado de máquina mais adequada é necessário para o desenvolvimento de modelos capazes de abordar o desequilíbrio inerente aos *code smells* e, ao mesmo tempo, mitigar o viés de interpretação. Para Pecorelli et al. (2019b) esses são os pontos mais relevantes deste problema de pesquisa. Além de usar apenas aprendizado supervisionado, outras técnicas de AM, i.e., métodos de comitê, aprendizado profundo e transferência de aprendizado, combinadas com técnicas de pré-processamento de dados foram utilizadas para melhorar os modelos baseados em AM para detecção de *code smell*.

#### 3.1.2.1 MÉTODOS DE COMITÊ

Os métodos de comitê são importantes no aprimoramento do desempenho de modelos de aprendizado de máquina para detecção de *code smell*. A ideia central por trás desses métodos é combinar vários classificadores para obter resultados superiores em comparação com um único classificador. Esta abordagem está bem estabelecida e tutoriais foram for-

necidos para orientar profissionais interessados na construção de sistemas de classificação baseados em comitê (POLIKAR, 2006; ROKACH, 2010).

A adoção dos métodos de comitê para melhorar modelos de aprendizado de máquina para detecção de *code smell* tem sido amplamente sugerida por vários autores, e.g., (PECORELLI et al., 2019b; AZEEM et al., 2019). Os comitês podem ser categorizados como heterogêneos ou homogêneos. Por um lado, comitês heterogêneos utilizam diferentes tipos de classificadores. Por outro lado, comitês homogêneos são construídos com classificadores do mesmo tipo, treinados em diferentes subconjuntos do conjunto de dados (ALAZBA; ALJAMAAN, 2021). O conceito dos métodos de comitê homogêneos utilizados para detecção de *code smell* estão descritos abaixo:

- *Max Voting* é uma técnica que usa um conjunto de métodos de comitê para produzir resultados (classificações) com base na classe com a maior probabilidade. Essa abordagem envolve o treinamento de múltiplos classificadores, e a previsão final é feita por um mecanismo de votação que considera as decisões coletivas dos classificadores individuais. Um classificador de votação pode contribuir para o processo de tomada de decisão na detecção de *code smell*, combinando os resultados de vários modelos, fornecendo uma previsão mais robusta e precisa para identificar *code smell* (NGUYEN THANH et al., 2022).

- *Bootstrap Aggregating (Bagging)* é um meta-algoritmo de comitê poderoso usado para melhorar a estabilidade e precisão de algoritmos de aprendizado de máquina (WANG et al., 2015). Esta técnica gera múltiplas amostras de *bootstrap* simultaneamente, começando a partir de um conjunto de dados de treinamento. Em seguida, ele treina vários classificadores binários, e a classificação final é determinada através de um mecanismo de votação (WANG et al., 2015; SAGI; ROKACH, 2018).

- *Random Forest (RF)* é uma variação do algoritmo de *bagging* que consiste em árvores de decisão individuais com parâmetros de configuração que variam aleatoriamente. Assim como *bagging*, os parâmetros podem ser inicializados com réplicas dos dados de treinamento. No entanto, esses parâmetros também podem ser subconjuntos de diferentes atributos em RF, utilizando métodos de subespaço aleatório (POLIKAR, 2006). Essa aleatoriedade na configuração ajuda a aumentar a diversidade das árvores individuais, levando a um modelo de comitê robusto e preciso.

- *Boosting* é considerado um dos algoritmos mais significativos da história do aprendizado de máquina (POLIKAR, 2006), é empregado na detecção de *code smell*.

Semelhante ao *bagging*, o *boosting* cria um comitê de classificadores por meio da reamostragem de dados combinados por meio de um mecanismo de votação por maioria. No entanto, ao contrário do *bagging*, o *boosting* concentra-se estrategicamente na reamostragem para fornecer os dados de treinamento mais informativos para cada classificador sucessivo. Esta técnica provou ser valiosa para melhorar o desempenho de modelos de aprendizado de máquina para detecção de *code smell*.

- *Adaptive Boosting (AdaBoost),* um algoritmo proeminente na família *boosting*, desempenha um papel significativo na detecção de *code smell*. Este algoritmo treina modelos sequencialmente, gerando um novo modelo treinado para cada rodada. Assim, as instâncias identificadas incorretamente recebem maior peso nas rodadas subsequentes. *AdaBoost* aproveita essa característica para selecionar algoritmos de aprendizado típicos, como árvores de decisão e SVM, como "aprendizes fracos". Essa combinação estratégica melhora o desempenho geral de modelos de aprendizado de máquina para detecção de *code smell* (FREUND; SCHAPIRE, 1995).

- *Gradient Boosting.* Na área de detecção de *code smell*, Luiz et al. (2019) utilizaram a técnica de *Gradient Boosting*, que amalgama um comitê de alunos fracos para construir um aluno mais robusto. Este método de comitê foi projetado para atingir menor viés e variância, aumentando sua eficácia.

- *Light Gradient Boosting (LightGBoost)* constrói um modelo aditivo usando árvores de decisão simples, que são então generalizadas otimizando uma função de perda definida pelo usuário para aumentar a robustez da previsão. As vantagens do *LightGBM* incluem velocidade de treinamento mais rápida, maior eficiência e melhor precisão em comparação com muitos outros algoritmos de *boosting*, juntamente com menor consumo de memória. Além disso, apresenta compatibilidade aprimorada com grandes conjuntos de dados (MCCARTY et al., 2020). Essas características tornam o *LightGBM* uma ferramenta valiosa no domínio da detecção de *code smell*.

- *XGBoost,* uma abreviatura de *Extreme Gradient Boosting*, é um importante algoritmo de aprendizado de máquina baseado em árvore, conhecido por seu desempenho e velocidade superiores. Desenvolvido inicialmente por Tianqi Chen e supervisionado pela organização *Distributed Machine-Learning Community (DMLC)*, o *XGBoost* é reconhecido por sua eficácia no tratamento de dados estruturados e tabulares. Dada a sua simplicidade e eficácia, o *XGBoost* ganhou ampla popularidade,

especialmente no contexto de detecção de *code smell* (DEWANGAN et al., 2023).

- *Category Boosting (CatBoost)* emprega o algoritmo *Sort Boosting* como um substituto para o método tradicional de estimativa de gradiente. Esta substituição reduz o desvio na estimativa do gradiente, melhorando em última análise a capacidade de generalização do modelo (WANG et al., 2022).

#### 3.1.2.2 APRENDIZADO PROFUNDO

A aplicação de técnicas de aprendizado profundo na detecção de *code smell* tem sido frequentemente associada a características semânticas relevantes extraídas do código-fonte para complementar as características estruturais. Esses atributos semânticos melhoram a compreensão do significado do código, complementando as métricas orientadas a objetos já amplamente utilizadas.

Liu et al. (2018) introduziram pela primeira vez uma abordagem de detecção baseada em aprendizado profundo. Seu método utilizou atributos textuais e métricas de código como entrada para detectar *Feature Envy*. Especificamente, para entrada textual, os autores focaram na utilização de uma fonte de informação, nomeadamente a identificação de nomes.

Ademais, de acordo com os resultados experimentais do estudo realizado por Yu et al. (2021), o número de amostras positivas foi identificado como um fator que influencia a eficiência do treinamento e também exerce um impacto no desempenho de predição dos modelos em certa medida. No entanto, o estudo destacou a natureza matizada desta relação, enfatizando que ter um conjunto de formação excessivamente grande pode não levar necessariamente a melhores resultados. Descobriu-se que muitas amostras diminuem potencialmente a velocidade de convergência dos modelos e aumentam o tempo geral de treinamento. Isso ressalta a importância de encontrar um equilíbrio no tamanho do conjunto de treinamento para otimizar a eficiência e o desempenho em modelos de aprendizado de máquina para detecção de *code smell*.

#### 3.1.2.3 TRANSFERÊNCIA DE APRENDIZADO

A Transferência de Aprendizado é uma técnica na qual um algoritmo de aprendizado aproveita pontos em comum entre diferentes tarefas de aprendizado para facilitar a transferência de conhecimento entre tarefas (BENGIO et al., 2013). Esta abordagem pode ser categorizada em dois tipos com base na comparabilidade de seus domínios: transferência

de aprendizado homogênea ou heterogênea (WEISS et al., 2016). Por um lado, o domínio de origem e o domínio de destino compartilham o mesmo conjunto de atributos na transferência de aprendizado homogênea. Por outro lado, a transferência de aprendizado heterogênea ocorre quando os domínios de origem e de destino possuem, cada um, conjuntos distintos de atributos. No entanto, poucos estudos utilizaram a transferência de aprendizado para detecção de *code smell*, e.g., (SHARMA et al., 2021; MA et al., 2023).

O estudo de Sharma et al. (2021) se destaca como um esforço pioneiro, ao investigar minuciosamente a aplicação de técnicas de aprendizado profundo para examinar características de qualidade do código-fonte sem depender de atributos derivados, como métricas, para modelos de aprendizado de máquina. Suas descobertas demonstram a viabilidade de aplicar a aprendizado por transferência na detecção de *code smell*, potencialmente levando a uma nova categoria de ferramentas para essa finalidade, especialmente para linguagens de programação que carecem de ferramentas maduras de detecção de *code smell*.

Além disso, Ma et al. (2023) estabeleceram que determinar se uma classe deve ter um método específico é influenciado pela relação semântica entre a classe e o método, aumentando assim a coesão. Nesse sentido, o estudo deles propôs a utilização de um modelo pré-treinado, *CodeT5*, para detectar *feature-envy*. Inicialmente, eles converteram pares de códigos de métodos e classes em sequências de *tokens*, alimentadas no modelo pré-treinado. Posteriormente, o modelo pré-treinado foi ajustado para detectar *feature-envy* e recomendar métodos que exibam este *code smell* para as classes apropriadas. Além disso, o estudo explorou o desempenho de vários outros modelos pré-treinados convencionais, incluindo *CodeBERT* e *CodeGPT*, na detecção de *feature-envy*.

### 3.1.3 ESTRATÉGIA DE BUSCA

Começamos com uma estratégia de busca para extrair todos os artigos de pesquisa potencialmente relevantes de bibliotecas digitais acadêmicas. Identificamos palavras-chave relevantes para o foco desta RSL e formulamos a sequência de pesquisa a partir de questões de pesquisa seguindo a abordagem de Kitchenham e Charters (2007) —PICO (*Population, Intervention, Comparison, and Outcomes*). Além disso, nos concentramos na detecção de *code smell* baseada em AM; assim, identificamos os sinônimos das técnicas de AM. Em seguida, foi realizada a busca em todas as bibliotecas digitais especificadas, utilizando como base a *string* de busca abaixo:

*Population:* (“code smell” OR “bad smell”) AND; *Intervention:* (“machine-learning” OR “machine learning” OR “structural-feature” OR “structural feature” OR “supervised”)

AND; *Comparation:* (“deep-learning” OR “deep learning” OR “semantic-feature” OR “semantic feature” OR “un- supervised” OR “semisupervised” OR “semi-supervised” OR “semi supervised”) AND; *Outcomes:* (“recall” OR “f-measure” OR “f-score” OR “F1” OR “precision” OR “mcc”).

Identificamos estudos relevantes utilizando a *string* de busca nas duas bibliotecas digitais acadêmicas mais utilizadas para engenharia de software: *IEEE Xplore* e *ACM Digital* (ZHANG et al.). Em seguida, incluímos a biblioteca *SCOPUS*, pois ela indexa diversas outras bases de dados acadêmicas menores. A busca foi realizada em novembro de 2023.

### 3.1.4 SELEÇÃO DE ESTUDOS

Para fornecer uma base para responder às nossas questões de pesquisa, foram estabelecidos critérios específicos de inclusão e exclusão de estudos primários, delineando o conjunto de fontes consideradas para este estudo. Além disso, consideramos diversos critérios de qualidade para que os estudos selecionados forneçam dados qualificados para responder às questões de pesquisa formuladas. O processo de inclusão sistemática de referências foi integrado ao processo de seleção dos estudos para complementar os resultados da estratégia de busca. E por fim, apresentamos um panorama dos estudos primários encontrados.

#### 3.1.4.1 CRITÉRIOS DE INCLUSÃO/EXCLUSÃO

Um estudo primário foi incluído somente se preenchesse todos os critérios descritos na Tabela 3.1.

TAB. 3.1: Critérios de Inclusão

| ID | Descrição do Critério de Inclusão |
|---|---|
| IC1 | Escrito em inglês |
| IC2 | Revisado por pares e publicado em Conferência ou Periódico |
| IC3 | Resultados empíricos relatados |
| IC4 | Técnicas de pré-processamento de dados, métodos de comitê, aprendizado profundo ou transferência de aprendizado relatados |

Foram excluídos os estudos que preencheram pelo menos um dos critérios detalhados na Tabela 3.2.

TAB. 3.2: Critério de Exclusão

| ID | Descrição do Critério de Exclusão |
|---|---|
| EC1 | Não escrito em inglês |
| EC2 | Publicação que não estava em fase final |
| EC3 | Estudo de revisão (e.g., RSL, *survey*, etc.) |
| EC4 | Estudo que se concentrou em outras técnicas de detecção, em vez de aprendizado de máquina |
| EC5 | Incapaz de acessar |
| EC6 | Procedimento ou índice |
| EC7 | Estudo duplicado |

#### 3.1.4.2 AVALIAÇÃO DE QUALIDADE

De acordo com Alazba et al. (2023), avaliar a qualidade dos estudos selecionados auxilia na interpretação dos principais dados da pesquisa, ao mesmo tempo que avalia a qualidade das informações extraídas. Kitchenham et al. (2009) observaram que não existem definições uniformes para as principais métricas de qualidade do estudo. Portanto, consideramos vários critérios de qualidade, conforme detalhado na Tabela 3.3.

TAB. 3.3: Avaliação de Qualidade

| ID | Questões da Avaliação de Qualidade |
|---|---|
| QA1 | Os objetivos e questões da pesquisa estão claramente definidos? |
| QA2 | As medidas de desempenho utilizadas para avaliar os modelos do estudo são especificadas? |
| QA3 | As variáveis independentes e variáveis dependentes estão claramente definidas? |
| QA4 | O método de validação foi especificado? |
| QA5 | Os conjuntos de dados estão adequadamente descritos? |
| QA6 | Os detalhes do método experimental estão descritos adequadamente? |
| QA7 | As ameaças à validade do estudo são especificadas? |
| QA8 | As técnicas de pré-processamento de dados usadas são especificadas? |
| QA9 | São usados métodos de comitê, aprendizado profundo ou transferência de aprendizado? |

Os critérios de qualidade incluem a clareza das questões de pesquisa, as medidas de desempenho, variáveis independentes/dependentes, os métodos de validação, os conjuntos de dados, os detalhes de experimentos e as ameaças à validade, conforme usado por Alazba et al. (2023). Além disso, consideramos o uso de pré-processamento de dados, métodos de comitê, aprendizado profundo e transferência de aprendizado. Foram atribuidas pontuações de acordo com o cumprimento desses critérios: “Sim” = 1, “Parcialmente” = 0,5 e ”Não” = 0.

Esta abordagem visa auxiliar na seleção de estudos primários que forneçam dados

suficientemente qualificados para abordar as questões de pesquisa formuladas. Por isso, estabelecemos um limite de 50% (ou seja, qualquer estudo que obtivesse pontuação inferior a 4,5 na avaliação da qualidade seria excluído), seguindo a abordagem utilizada por Alazba et al. (2023).

#### 3.1.4.3 *SNOWBALLING*

Com os estudos da seleção final derivados da *string* de busca, integramos o processo de seleção dos estudos adotando a inclusão sistemática de referências, ou seja, *snowballing*, conforme definido por Wohlin (2014) para busca de possíveis artigos faltantes. Em nosso contexto, aplicamos o *forward snowballing* aos artigos inicialmente selecionados, que incluiu todos os artigos que os referenciavam, e o *backward snowballing*, que incluiu todos os artigos que foram referenciados por eles.

### 3.1.5 VISÃO GERAL DOS ESTUDOS PRIMÁRIOS

A figura 3.1 mostra o número de artigos selecionados ao longo dos anos.

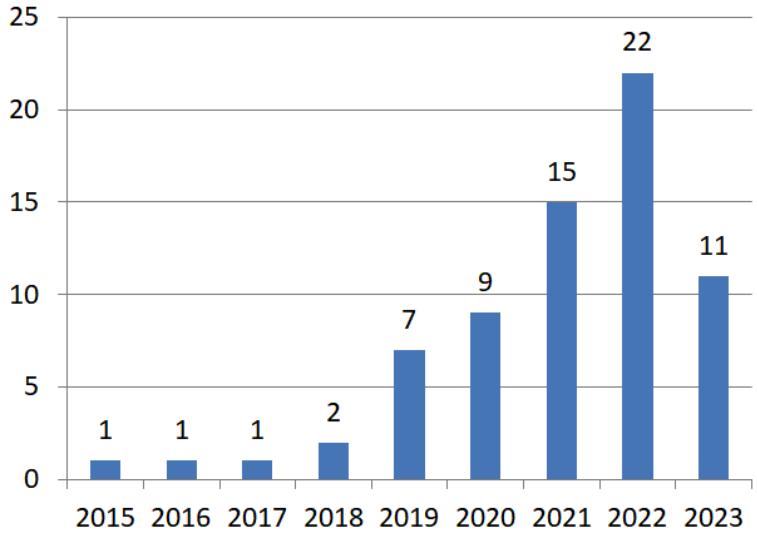


FIG. 3.1: Produção Científica Anual

Observamos que esta área de pesquisa tem ganhado importância. Percebe-se que o número de estudos publicados entre 2015 e 2017 manteve-se estável, sendo um estudo publicado por ano, enquanto em 2018 foram publicados dois estudos. Porém, em 2019, houve um aumento de três vezes nas publicações, i.e., sete estudos publicados. Além disso, tem havido um aumento no número de estudos publicados nesta área nos últimos anos (9, 15 e 22, respectivamente, em 2020, 2021 e 2022), o que indica que a atenção

da investigação nesta área está crescendo. E por fim, em 2023, já foram publicados 11 estudos.

Um total de 328 estudos potenciais foram identificados aplicando a *string* de busca detalhada na Subseção 3.1.3. Além disso, foram incluídos filtros na biblioteca digital *SCOPUS*, para limitar os estudos àqueles escritos em inglês, artigos ou conferências, e aqueles publicados em periódico ou conferências.

A figura 3.2 ilustra o processo de seleção dos estudos primários empregado.

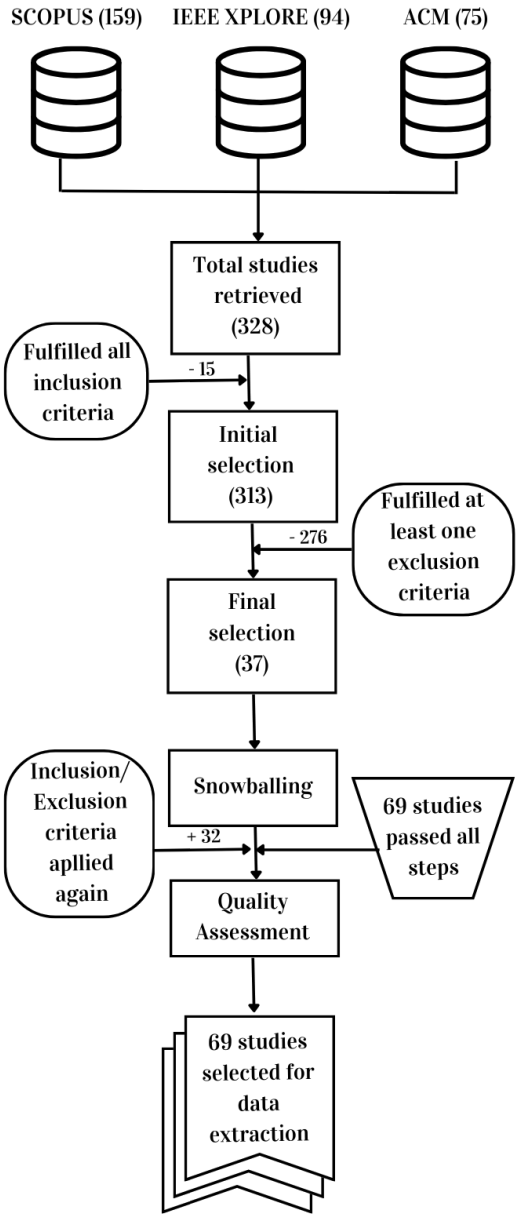


FIG. 3.2: Processo de seleção de estudos primários.

Partindo do conjunto de estudos potenciais, filtramos inicialmente os artigos que atendiam a todos os critérios de inclusão, resultando na seleção inicial de 313 estudos (15

foram excluídos por não terem sido submetidos à revisão por pares e por não terem sido publicados em congresso ou periódico). Posteriormente, foram eliminados desta seleção os estudos irrelevantes, aplicando-se os critérios de exclusão: i) 7 estudos não escritos em inglês (EC1); ii) 3 estudos que não estavam em fase final (EC2); iii) 38 estudos de revisão (EC3); iv) 185 estudos que centraram-se em outras técnicas de deteção em vez de aprendizado de máquina (EC4); v) 1 estudo sem acesso (EC5); vi) 33 anais ou índice (EC6); e vii) 9 estudos duplicados (EC7).

Isso resultou na exclusão de um total de 276 estudos, conforme detalhado na Tabela 3.4. Depois, 37 estudos foram selecionados por *snowballing* e submetidos aos mesmos critérios de exclusão/inclusão. Como resultado, foram selecionados 32 novos artigos (ver Tabela 3.5), totalizando 69 artigos a serem analisados na avaliação de qualidade. Nenhum artigo foi excluído após a avaliação.

TAB. 3.4: Detalhamento dos estudos excluídos

| BIBLIOTECA DIGITAL | EC1 | EC2 | EC3 | EC4 | EC5 | EC6 | EC7 | TOTAL |
|---|---|---|---|---|---|---|---|---|
| *ACM* | 0 | 0 | 9 | 31 | 0 | 29 | 3 | 72 |
| *IEEE Xplore* | 4 | 0 | 15 | 50 | 0 | 4 | 0 | 73 |
| *SCOPUS* | 3 | 3 | 14 | 104 | 1 | 0 | 6 | 131 |
| TOTAL | 7 | 3 | 38 | 185 | 1 | 33 | 9 | 276 |

TAB. 3.5: Fontes de dados e resultados de pesquisa

| FONTE DE DADOS | ESTUDOS POTENCIAIS | SELEÇÃO INICIAL | SELEÇÃO FINAL |
|---|---|---|---|
| ACM | 75 | 75 | 3 |
| IEEE Xplore | 94 | 94 | 21 |
| Scopus | 159 | 144 | 13 |
| Subtotal | 328 | 313 | 37 |
| *Backward Snowballing* | - | - | 7 |
| *Forward Snowballing* | - | - | 25 |
| Total Geral | 328 | 313 | 69 |

A figura 3.3 ilustra a contribuição de cada fonte de pesquisa em cada etapa do processo de seleção dos estudos primários. O *IEEE Xplore* teve uma participação elevada e constante no percentual de estudos selecionados, variando de 28,66% dos estudos potenciais a 30,43% da seleção final. O *SCOPUS* começou com estudos de maior potencial do que outras fontes de pesquisa, mas teve uma queda acentuada de quase 30% na participação, terminando com 18,84% dos estudos primários. *ACM Digital* foi a fonte de pesquisa com menor número de contribuições, chegando à seleção final com apenas 4,35% dos estudos

primários selecionados. Vale destacar que 53,62% dos estudos primários foram selecionados através da *string* de busca, e os 46,38% restantes foram encontrados através do processo de *Snowballing*.

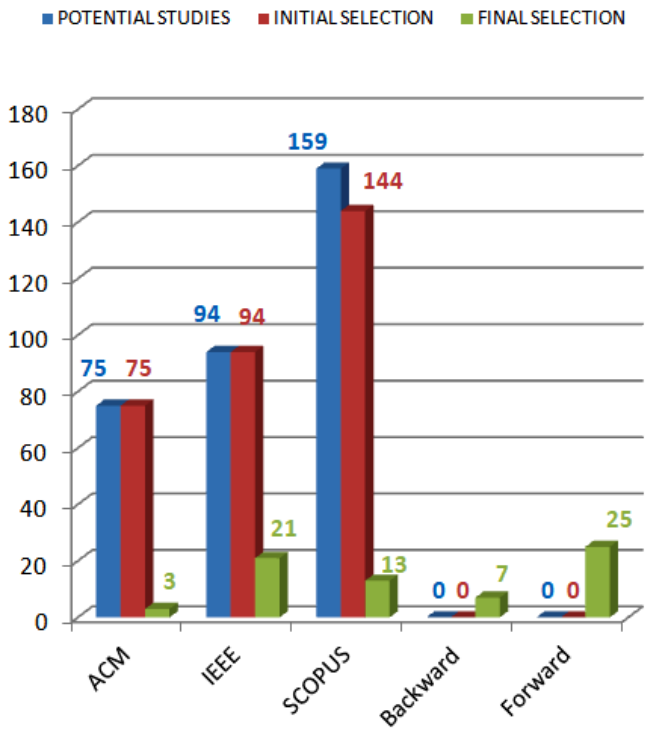


FIG. 3.3: Contribuição de cada fonte de pesquisa.

### 3.1.6 ANÁLISE DE DADOS

#### 3.1.6.1 EXTRAÇÃO DE DADOS

Após finalizar a seleção de estudos primários para nossa RSL, extraímos os dados necessários desses estudos para responder às nossas questões de pesquisa. Este processo de extração foi facilitado pela utilização do formulário de extração de dados detalhado na Tabela 3.6. Dentro deste formulário, alocamos uma seção designada para documentar possíveis limitações observadas nos estudos, com o objetivo geral de desenvolver futuras diretrizes de pesquisa.

Além disso, coletamos sistematicamente informações sobre atributos específicos em consideração. Esses atributos abrangeram uma gama de técnicas empregadas em pré-processamento de dados, métodos de comitê, aprendizado profundo e transferência de aprendizado, todos destinados a aprimorar modelos de aprendizado de máquina para detecção de *code smell*. Ao reunir metodicamente esta informação, analisamos de forma

TAB. 3.6: Formulário de Extração de Dados

| DADOS EXTRAÍDOS | DESCRIÇÃO |
|---|---|
| Metadados | ID do Estudo, Título, Autores, Ano de Publicação, Tipo de Publicação Local de Publicação e Resumo. |
| Técnicas de Pré-processamento de dados aplicadas | Quais técnicas de pré-processamento de dados foram utilizadas nos estudos? |
| Melhoria dos modelos de AM | Quais métodos de comitê, aprendizado profundo ou transferência de aprendizado foram utilizados nos estudos? |
| Limitações ou Oportunidades | Quais são as limitações e oportunidades para melhoria das abordagens aplicadas? |
| Progresso alcançado | Quais avanços foram alcançados pelos estudos na detecção de *code smell*? |
| Problema a ser resolvido | Quais foram os detalhes contextuais do problema a ser resolvido? |

abrangente as metodologias e estratégias predominantes na literatura, enriquecendo assim a profundidade da nossa revisão.

#### 3.1.6.2 SÍNTESE DE DADOS

A síntese de dados envolve a coleta e o resumo dos resultados dos estudos primários incluídos. A síntese pode ser descritiva (não quantitativa) e, apesar disso, é ocasionalmente viável aprimorar uma síntese descritiva com um resumo quantitativo (KITCHENHAM; CHARTERS, 2007).

Para sintetizar os dados extraídos relativos às questões de pesquisa, definimos a estratégia de síntese narrativa para QP1 e QP2. Nesse sentido, buscamos extrair informações sobre os estudos (e.g., técnicas de pré-processamento de dados utilizadas, contexto, limitação e qualidade do estudo). Tabulamos os dados em estilo compatível com as questões. Além disso, estruturamos as tabelas geradas para evidenciar os resultados obtidos pelo estudo.

### 3.1.7 PRINCIPAIS RESULTADOS DA RSL

A RSL procurou, principalmente, resumir e sintetizar estudos que empregavam técnicas de pré-processamento de dados e a sua relação com técnicas de AM, e.g., métodos de comitê, aprendizado profundo e transferência de aprendizado, para aprimorar modelos de detecção de *code smell*. Os principais resultados derivadas deste estudo, alinhadas com as questões de pesquisa, são apresentadas abaixo:

a) **Técnicas de Pré-processamento de Dados (QP1)**

Várias técnicas de pré-processamento de dados foram empregadas nos conjuntos de dados para treinar, validar e testar modelos de aprendizado de máquina, conforme ilustrado na Figura 3.4 Entre o total de ocorrências dessas técnicas (122 vezes nos trabalhos selecionados), técnicas de balanceamento de dados, técnicas de seleção de atributos e filtragem foram proeminentes, constituindo 33%, 26% e 14%, respectivamente.

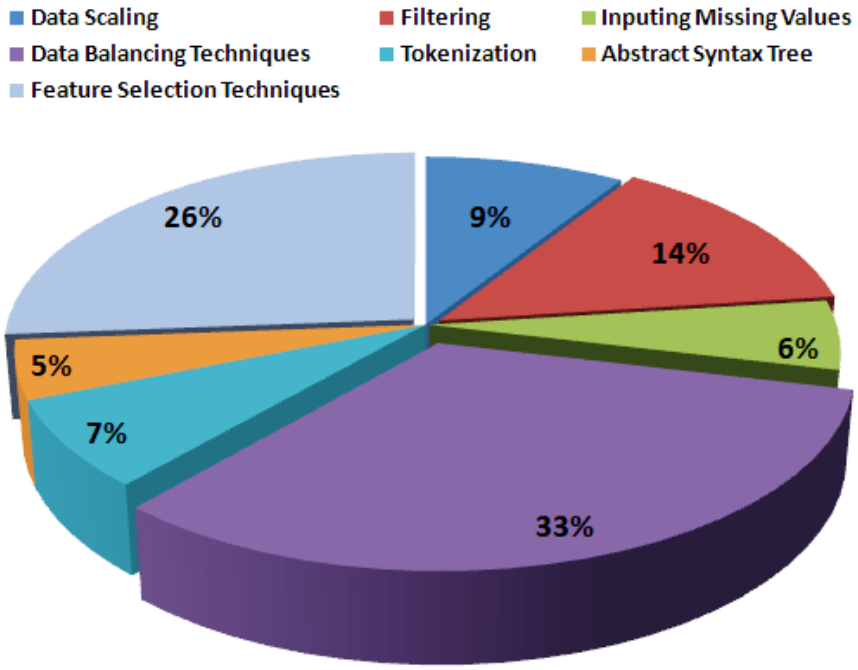


FIG. 3.4: Técnicas de pré-processamento de dados encontradas.

Abordagens adicionais de pré-processamento de dados também foram exploradas além dessas técnicas fundamentais. O escalonamento de dados (9%), tokenização (7%), imputação de valores faltantes (6%) e árvore de sintaxe abstrata (5%) foram investigados como estratégias viáveis. Esses esforços buscam refinar e otimizar os dados de entrada para modelos de aprendizado de máquina, contribuindo, em última análise, para o desempenho e eficácia geral dos sistemas de detecção de *code smells*.

b) **Técnicas de AM usadas com Técnicas de Pré-processamento de Dados (QP2)**

A Figura 3.5 mostra a proporção do relacionamento das técnicas de AM com técnicas de pré-processamento de dados em modelos baseados em AM para detecção de *code smell*. Encontramos 158 combinações diferentes para esse fim. Destas combinações, 46% das vezes, as técnicas de pré-processamento de dados foram utilizadas juntamente com pelo menos um método de comitê. O aprendizado profundo foi utilizado em 37% dos casos e 11% das combinações tiveram transferência de aprendizado como técnica de AM utilizada. É necessário destacar que em 6% dos casos foram aplicadas apenas técnicas de aprendizado supervisionado.

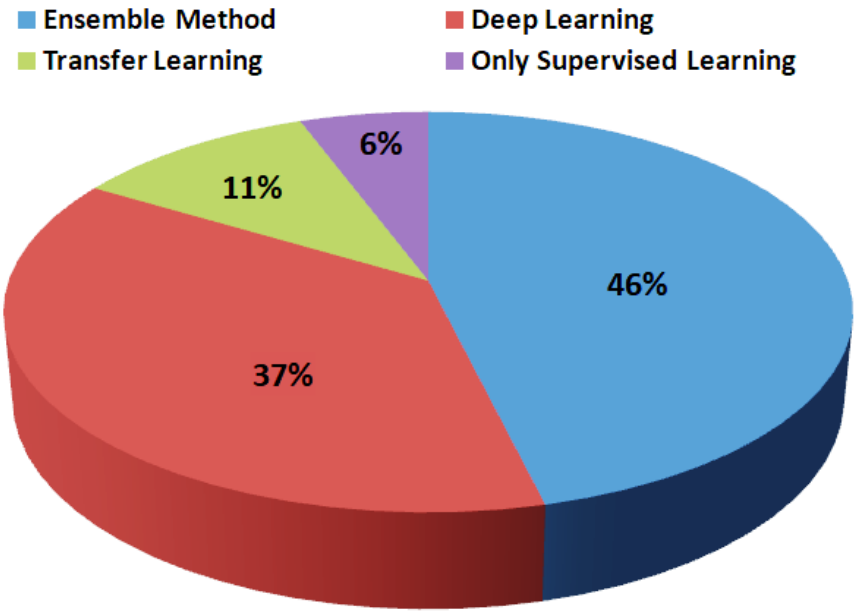


FIG. 3.5: Técnicas de AM utilizadas com técnicas de pré-processamento de dados para detecção de *code smell*.

Esses resultados ressaltam que os métodos de comitê, particularmente *Random Forest*, tornaram-se amplamente prevalentes entre as técnicas para detecção de *code smell*. Esses métodos apresentam desempenho robusto, mesmo quando o conjunto de dados carece de pré-processamento abrangente. Arcelli Fontana et al. foram pioneiros na exploração do *Random Forest* tanto para detecção de *code smell* (ARCELLI FONTANA et al., 2016) quanto para classificação de gravidade de *code smell* (ARCELLI FONTANA; ZANONI, 2017). Entretanto, até onde sabemos, apenas dois estudos (ALAZBA; ALJAMAAN, 2021; ALJAMAAN, 2021) se aventuraram a empregar métodos de comitê heterogêneos. Ademais, outro método de comitê que obteve bons resultados na detecção de *code smell* foi

o *Category Boosting* (CatBoost) –Alkharabsheh et al. (2022). No caso da classificação de gravidade do *code smell*, um problema multiclasse, *XGBoost* se destaca por seus resultados proeminentes (ABDOU; DARWISH, 2022; HU et al., 2023; DEWANGAN et al., 2023).

Essas descobertas sugerem que, além da seleção cuidadosa dos atributos, a utilização de métodos de comitê e a normalização de dados tem igual importância na detecção de *code smell*. Isso ressalta a natureza holística do desenvolvimento eficaz de modelos, onde considerações como relevância de atributos, técnicas de comitê e normalização de dados contribuem coletivamente para o desempenho geral e a precisão dos sistemas de detecção de *code smell*. Nesse sentido, é essencial para obter resultados robustos e confiáveis no domínio de detecção de *code smell* adotar uma abordagem abrangente que aborde vários aspectos da construção e otimização do modelo.

## 3.2 PARADIGMAS DE ANOTAÇÃO E O CONJUNTO DE DADOS

No processo de criação de conjuntos de dados, os construtores desses conjuntos devem decidir explicitamente seguir um dos dois paradigmas de anotação contrastantes – um paradigma prescritivo ou um paradigma descritivo (RÖTTGER et al., 2021). Entretanto, para muitas tarefas de anotação de dados, diferentes pontos de vista podem ser igualmente válidos e não existe uma verdade objetiva única (SLIVKA et al., 2023).

Por um lado, **o paradigma descritivo**, a.k.a. um procedimento personalizado ou centrado em crenças (KOCOŃ et al., 2021), tem o objetivo de capturar uma multiplicidade de crenças válidas, incentivando explicitamente a subjetividade dos anotadores. Assim, o conjunto de dados resultante podem ser usados para obter *insights* sobre as crenças dos anotadores, ajudar a identificar casos extremos, treinar modelos capazes de aprender com divergências e avaliar como um modelo treinado seria aceitável para diferentes grupos de usuários (RÖTTGER et al., 2021). Contudo, uma limitação do paradigma descritivo é que ele não pode ser facilmente usado para treinar um modelo que aplique consistentemente um comportamento pré-especificado. Múltiplas crenças precisam ser agregadas em um único rótulo para criar o rótulo de verdade básica, ocultando divergências informativas e arriscando que as crenças minoritárias sejam descartadas (RÖTTGER et al., 2021).

Por outro lado, **o paradigma prescritivo**, a.k.a. procedimento generalizado ou centrado em dados, visa criar o conjunto de dados capturando a melhor perspectiva comum (KOCOŃ et al., 2021). Este paradigma desencoraja a subjetividade do anotador, incumbindo-os de rotular os dados de acordo com uma crença específica prescrita nas diretrizes de anotação (RÖTTGER et al., 2021). Apesar da liberdade dos anotadores

para tomar decisões possa ser limitada no paradigma prescritivo (KOCOŃ et al., 2021), existem alguns benefícios na sua capacidade de desenvolver um modelo com um comportamento pré-especificado, garantir a qualidade do rótulo e fornecer visibilidalade da crença codificada que cria responsabilidade, pois as pessoas podem rever e desafiar crenças explicitamente formuladas (RÖTTGER et al., 2021).

O conjunto de dados manualmente rotulado MLCQ (MADEYSKI; LEWOWSKI, 2020) é considerado o conjunto de dados de *code smell* mais abrangente disponível de acordo com diferentes aspectos, principalmente o tamanho e a qualidade das amostras de dados (ZAKERI-NASRABADI et al., 2023). Entretanto, alguns, mas não todos, trechos de códigos foram rotulado por múltiplos engenheiros, e, também, é muito alinhado ao paradigma descritivo (SLIVKA et al., 2023). Além disso, por escolha dos autores, o MLCQ não possui preditores, i.e., variávies independentes na sua constituição. Diferentemente, Slivka et al. (2023), Arcelli Fontana e Zanoni (2017) e Nanda e Chhabra (2022) utilizaram o paradigma prescritivo, que desencouraja a subjetividade do anotador, com seu procedimento centrado em dados e possuem preditores na sua constituição. Portanto, os conjuntos de dados que serão utizados nesta proposta seguem o paradigma prescritivo e estão disponíveis, respectivamente, em Zenodo[1]–Slivka et al. (2023), em *Evolution of Software Systems and Reverse Engineering (ESSeRE Lab)*[2]–Arcelli Fontana e Zanoni (2017) e em *Google Drive*[3]–Nanda e Chhabra (2022).

Para criar seu conjunto de dados *CSharp*, Slivka et al. (2023) se basearam em quatro requisitos para o procedimento de anotação de *code smell* para um conjunto de dados com critérios de detecção especificados consistentemente, a saber:

a) Cada instância do conjunto de dados deve ser rotulada manualmente.

b) Cada instância do conjunto de dados deve ser rotulada por vários anotadores.

c) Os anotadores devem chegar a um consenso de verdade que seja consistente com as diretrizes de detecção de *code smell* amplamente aceitas.

d) A relação entre instâncias com *code smell* e sem *code smell* deve refletir o equilíbrio realista encontrado em projetos de software.

O estudo realizado por Arcelli Fontana e Zanoni (2017) modelou a gravidade do *code smell* como uma variável ordinal, expandindo os modelos propostos pelo trabalho de Ar-

https://doi.org/10.5281/zenodo.6520056
http://essere.disco.unimib.it/reverse/MLCSD.html
https://drive.google.com/drive/folders/16BqUdNlKNgdM_qrrJqGWKdQ_NfEdRPVD?usp=sharing

celli Fontana et al. (2016), que alcançaram resultados promissores ao empregar modelos de classificação binária para detecção de *code smell*. Contudo, no estudo realizado por Nanda e Chhabra (2022) uma análise detalhada e a correção dos conjuntos de dados da pesquisa de Arcelli Fontana e Zanoni (2017) foram ralizados, removendo inconsistências nos dados referentes às classes *God-class* e *Data-class*. Esta intervenção resultou em melhorias no desempenho das técnicas de aprendizado de máquina usadas para classificação de gravidade de *code smell*.

A maior diferença entre a metodologia de Slivka et al. (2023) e de Arcelli Fontana et al. (2016); Arcelli Fontana e Zanoni (2017) é o procedimento de amostragem de trechos de código que resultou na distribuição de classes irrealista (requisito d) das últimas. Esta distribuição irreal pode afetar a generalização dos modelos de AM treinados neste conjunto de dados (DI NUCCI et al., 2018). Concretamente, a distribuição das instâncias do conjunto de dados responsável pelo alto desempenho do modelo difere significativamente de um projeto de software realista (PALOMBA et al., 2018).

Para uma compreensão mais clara, apresentamos as seguintes definições para as instâncias de *code smells* presentes nos conjunto de dados:

***God-class* (GC)** refere-se a classes que centralizam a inteligência do sistema e podem tender a exibir características como complexidade, excesso de código, uso abundante de dados de outras classes e implementação de múltiplas funcionalidades (HU et al., 2023).

***Data-class* (DC)** são classes que servem principalmente como armazenamento de dados, possuem funcionalidade limitada e muitas outras classes dependem delas. Essas classes possuem múltiplos atributos, não possuem complexidade e fornecem acesso aos dados por meio de métodos acessadores (HU et al., 2023).

***Feature-envy* (FE)** refere-se a métodos que acessam e usam predominantemente dados de outras classes em vez dos seus próprios. Esses métodos são frequentemente caracterizados pelo uso excessivo de atributos de outras classes, incluindo aqueles acessados através de métodos acessadores (HU et al., 2023).

***Long-method* (LM)** é caracterizado por métodos que encapsulam uma quantidade excessiva de funcionalidade dentro de uma única classe. Esses métodos tendem a ser demorados, complexos, difíceis de entender e dependem extensivamente de dados de outras classes (HU et al., 2023).

Slivka et al. (2023) adaptaram a metodologia prescritiva de anotação de processamento de linguagem natural para detecção dos *code smell* LM e GC. Eles construiram um conjunto de dados de *code smell* de tamanho médio a partir de 8 projetos *CSharp* de código aberto. Apesar da linguagem *CSharp* dominar as discussões dos desenvolvedores

sobre *code smell* (TAHIR et al., 2020), a maioria dos conjuntos de dados anotados manualmente contêm projetos escritos em Java (ZAKERI-NASRABADI et al., 2023). Para o *code smell* LM, foram anotados 2.574 instâncias, e para o code smell GC foram anotados 920 instâncias.

Arcelli Fontana e Zanoni (2017) selecionaram 76 de 111 sistemas, que foram avaliados para diferentes tamanhos, considerando um extenso conjunto de atributos orientados a objetos. A escolha dos sistemas baseou-se no *Qualitas Corpus* versão 20120401r, coletado por Tempero et al. (2010). Para detectar a gravidade do *code smell*, i.e, DC, GC, FE e LM, eles empregaram uma variedade de ferramentas e métodos conhecidos como "conselheiros", como *iPlasma* (MARINESCU et al., 2005), *Fluid Tool* (NONGPONG, 2012), *JSpIRIT* (VIDAL et al., 2015), PMD (MARINESCU et al., 2005) e regras de detecção de Marinescu (2005). A Tabela 3.7 apresenta as ferramentas de detecção automática utilizadas no estudo.

TAB. 3.7: Ferramenta de detecção automática ("Conselheiros")

| Classe | "Conselheiros" |
|---|---|
| DC | *iPlasma* |
| GC | *iPlasma, JSpIRIT, PMD* |
| FE | *iPlasma, Fluid Tool* |
| LM | *iPlasma (Brain Method)*, *PMD*, Regras de Detecção de Marinescu (2005) |

No estudo realizado por Nanda e Chhabra (2022), durante a análise dos conjuntos de dados fornecidos por Arcelli Fontana e Zanoni (2017), foi identificado um número significativo de inconsistências entre conjuntos de dados binários e multinomiais de GC e DC. Essas discrepâncias foram adequadamente corrigidas e os demais casos foram minuciosamente avaliados quanto a quaisquer erros de classificação, exigindo ajustes na gravidade quando necessário. Vários "conselheiros" foram empregados durante esta fase de correção, notadamente *iPlasma* (MARINESCU et al., 2005), *JSpIRIT* (VIDAL et al., 2015) e PMD[4] para GC, enquanto a ferramenta *iPlasma* foi usada para DC, e a decisão final e os rótulos de gravidade foram atribuídos com base na avaliação de especialistas. A Tabela 3.8 resume as diferentes correções realizadas, indicando os motivos de cada ajuste.

Cada um desses quatro conjuntos de dados inclui 420 instâncias, representando classes ou métodos. Estas instâncias estão associadas a um vetor de características, composto por 63 valores para GC e DC, e 84 valores para LM e FE. Além disso, cada conjunto de

https://github.com/pmd/pmd

TAB. 3.8: Detalhes das instâncias corrigidas fornecidas por Nanda e Chhabra (2022)

| Motivo | *God-class* | *Data-class* |
|---|---|---|
| A. Conflito entre conjunto de dados | 128 | 129 |
| B. Classificado incorretamente como instância negativa | 5 | 12 |
| C. Classificado incorretamente como instância positiva | 6 | 6 |
| D. Reclassificação de gravidade | 18 | 20 |
| E. Mudanças não conflitantes (B+C+D) | 29 | 38 |
| Total (A + E) | 157 | 167 |

dados contém informações sobre o valor da gravidade do *code smell* (Tabela 3.9).

TAB. 3.9: Valor da gravidade do *code smell*

| Gravidade | Descrição |
|---|---|
| 0 - Sem *code smell* | A classe ou método não é afetado pelo *code smell*. |
| 1 - *Code smell* não severo | A classe ou método é parcialmente afetado pelo *code smell*. |
| 2 - *Code smell* | As características do *code smell* estão todas presentes na classe ou método. |
| 3 - *Code smell* severo | O *code smell* está presente e tem valores particularmente altos de tamanho, complexidade ou acoplamento. |

## 3.3 DETECÇÃO E CLASSIFICAÇÃO DE GRAVIDADE DE *CODE SMELL* BASEADA EM AM.

Esta seção apresenta estudos que empregaram técnicas de AM para detectar e classificar a gravidade do *code smell*. Algumas dessas investigações examinaram os impactos das técnicas de pré-processamento de dados e métodos de comitê na classificação de gravidade do *code smell*.

Hu et al. (2023) avaliaram o desempenho de 21 modelos de predição na estimativa da gravidade do *code smell*. Esses modelos incluíram 10 métodos de classificação e 11 algoritmos de regressão, onde as principais métricas de desempenho adotadas foram *Cumulative Lift Chart* (CLC) e *Severity@20%*. Este último é usado para avaliar a porcentagem da gravidade total das 20 principais instâncias de software previstas. Em contrapartida, a acurácia foi utilizada como indicador secundário.

O estudo realizado por Abdou e Darwish (2022) teve como objetivo avaliar o desempenho de modelos de classificação para a gravidade do *code smell* após aplicação da técnica de reamostragem SMOTE. Um foco adicional do estudo foi a seleção dos melho-

res modelos através de uma análise comparativa. Além disso, eles extraíram regras de detecção para examinar a eficácia do uso de métricas de software na detecção de *code smell*.

Em sua pesquisa, Rao et al. (2023) empregaram o método SMOTE para enfrentar o desafio do desequilíbrio de classe e aplicaram a técnica de seleção de atributos baseada na Análise de Componentes Principais (PCA) para melhorar a precisão. Diferentemente, Dewangan et al. (2023) usaram modelos de AM de comitê, empregando uma abordagem de seleção de atributos qui-quadrado em cada conjunto de dados, e avaliaram o impacto da otimização de hiperparâmetros nos resultados de classificação para detectar a gravidade do *code smell*.

O estudo de Arcelli Fontana e Zanoni (2017) focou na avaliação da gravidade do *code smell* aplicando técnicas de AM em diversos experimentos. Foram testados diferentes modelos, desde classificação multinomial até regressão, incluindo um método para adaptar classificações binárias à classificação ordinal. Notavelmente, eles modelaram a gravidade do *code smell* como uma variável ordinal, expandindo os modelos propostos por Arcelli Fontana et al. (2016), onde resultados promissores foram alcançados através do emprego de modelos de classificação binária para detecção de *code smell*.

No estudo de Nanda e Chhabra (2022) foi realizada a análise detalhada e a correção dos conjuntos de dados da pesquisa de Arcelli Fontana e Zanoni (2017), removendo inconsistências nos dados relacionados a *God-class* e *Data-class*. Esta intervenção melhorou o desempenho das técnicas de AM utilizadas para classificação da gravidade do *code smell*. Posteriormente, eles propuseram uma abordagem chamada SSHM (*Synthetic Minority Over-sampling Technique - SMOTE and Stacking in combination*), aplicando-a à classificação de gravidade dos quatro tipos de *code smell*, i.e., GC, DC, FE e LM.

A Tabela 3.10 apresenta uma síntese da literatura sobre os estudos que buscaram classificar a gravidade de *code smell*. Nesse sentido, é importante destacar que todos os estudos buscaram resolver a questão do desbalanceamento do conjunto de dados com a utilização de métodos de comitê da família de *Bagging* ou de *Boosting*. Além disso, a técnica de sobreamostragem SMOTE foi a única técnica de balanceamento de dados utilizada nos conjuntos de dados desbalanceados. E, também vale a pena destacar que todos os estudos empregaram pelo menos uma técnica de pré-processamento de dados para melhorar a precisão dos modelos baseados em AM para classificação da gravidade de *code smell*, exceto pelo estudo de Hu et al. (2023), que buscou enfrentar o problema apenas com métodos de comitê.

TAB. 3.10: Síntese da Literatura sobre Classificação de Gravidade de *Code Smell*

| Estudo | Balanceamento de Dados | Seleção de Atributos | Escalonamento de Dados | Método de Comitê | Medida de Desempenho |
|---|---|---|---|---|---|
| Arcelli Fontana e Zanoni (2017) | - | Filtros de baixa variância e de alta correlação linear | Normalização | Adaboost e Random Forest | Acurácia |
| Nanda e Chhabra (2022) | *SMOTE* | - | - | Adaboost e Random Forest | Acurácia |
| Abdou e Darwish (2022) | *SMOTE* | Ganho de Informação | - | Random Forest e Bosting | Acurácia |
| Hu et al. (2023) | - | - | - | Adaboost, Bagging, Random Forest e XGBoost | Acurácia |
| Rao et al. (2023) | *SMOTE* | *PCA* | Normalização | Random Forest | Acurácia |
| Dewangan et al. (2023) | - | Qui-quadrado | Normalização | Adaboost, Gradient Boost, Random Forest e XGBoost | Acurácia, Precisão, Sensibilidade e *F1-Score* |

No que diz respeito ao escalonamento de dados na fase de pré-processamento de dados, embora Arcelli Fontana e Zanoni (2017), Rao et al. (2023) e Dewangan et al. (2023) tenham utilizado a técnica de normalização com essa finalidade, esse uso não foi acompanhado nem de uma comparação com outras técnicas de escalonamento de dados, e.g., padronização de dados, nem de uma análise do impacto dessas técnicas sobre a precisão dos modelos. Além disso, quanto às medidas de desempenhos utilizadas para avaliar os modelos baseados em AM para classificação da gravidade de *code smell*, exceto pelo estudo de Dewangan et al. (2023), os estudos não consideraram outras métricas de desempenho, como precisão, sensibilidade ou *f-score*, limitando a análise dos resultados da matriz de confusão.

Nexte contexto, os estudos de pesquisa, com foco na gravidade, utilizaram o conjunto de dados de referência de Arcelli Fontana e Zanoni (2017) e tiveram como objetivo principal classificar a gravidade dos *code smells* em conjuntos de dados que continham instâncias de apenas um tipo específico de *code smell*. Esse foco estreito separa essas abordagens das situações do mundo real no desenvolvimento de software real. Ao fazer isso, eles ignoram a interação e a influência entre os diferentes tipos de *code smell*.

Ao contrário de trabalhos anteriores, para atingir o objetivo de detectar e classificar os *code smells*, combinaremos os dois conjuntos de dados de nível de método de Arcelli Fontana e Zanoni (2017) e os dois conjuntos de dados de nível de classe de Nanda e Chhabra (2022) em um único. Essa combinação visa representar com mais precisão o desafio encontrado em software real, no qual as instâncias dos *code smells*, tanto no nível do método quanto na classe, são influenciadas por formas distintas de gravidade do *code smell*.

Também propomos uma abordagem em duas etapas para detectar e classificar a gravidade do *code smell* com base em métodos de comitê, que usam balanceamento de dados, escalonamento de dados e seleção de atributos para tornar os modelos mais robustos e precisos. Esta abordagem permite a detecção de instâncias com ou sem *code smell*, seguindo uma abordagem binária, e a classificação de suas gravidades, realizada apenas com instâncias positivas, em uma abordagem multiclasse.

Finalmente, com essa abordagem de aprendizado mais robusta, espera-se transferir o conhecimento adquirido pelo detector e classificador de gravidade de *code smell* para outro conjunto de dados contendo instâncias de gravidades de *code smell* de softwares desenvolvidos em *CSharp*. Diferentemente da abordagem de Sharma et al. (2021) que abordaram a transferência de aprendizado de modelos treinados em Java para conjunto de dados em *CSharp*, utilizando aprendizado profundo sem o auxílio de métricas de software como preditores.

# 4 PROBLEMA E OBJETIVOS

O problema a ser resolvido decorre da necessidade de aprimorar o ciclo de vida dos softwares corporativos do EPSD-CFN-MB, especialmente em relação à manutenibilidade da qualidade do software. Depois, da condução de uma revisão sistemática da literatura focada no pré-processamento de dados para detecção de *code smell*, lacunas significativas e áreas carentes de melhorias foram reveladas. Dentre essas lacunas, é importante ressaltar que os conjuntos de dados disponíveis apresentam amostras desequilibradas, não oferecem suporte adequado aos níveis de gravidade e, em sua maioria, estão limitados à linguagem Java (ZAKERI-NASRABADI et al., 2023; SLIVKA et al., 2023). Ademais, a criação manual de conjuntos de dados anotados é um processo oneroso, com a maioria desses conjuntos focando em projetos Java, à exceção do conjunto de dados de Slivka et al. (2023). Isso é preocupante, considerando que a linguagem *CSharp* é predominante nas discussões dos desenvolvedores sobre *code smells* (ABUHASSAN et al., 2021), o que resulta em uma escassez de ferramentas de análise de código para detecção de *code smell* (TAHIR et al., 2020; ABUHASSAN et al., 2021).

Diante disso, à medida que a complexidade do código-fonte aumenta, a combinação entre técnicas avançadas de pré-processamento, métodos de comitê e modelos de transferência de aprendizado pode abrir novas possibilidades em termos de precisão e eficácia na detecção e classificação da gravidade de *code smell*. Assim, a urgência em resolver essas questões sublinha a necessidade de uma abordagem que supere tais desafios. Esta abordagem deve ser capaz de aproveitar os avanços obtidos na detecção de *code smells* em softwares Java, sem a necessidade de anotar manualmente um um novo conjunto de dados para projetos *CSharp*.

Neste contexto, o objetivo desta Proposta é desenvolver uma abordagem de detecção e classificação de gravidade de *code smell* baseado em transferência de aprendizado. A Proposta apresenta os seguintes objetivos específicos:

- A construção de um conjunto de dados que contenha instâncias de múltiplos tipos de gravidade de *code smells*, que seja mais realista e facilite aos algoritmos de AM aprenderem melhor as nuances dessas instâncias, com base em um conjunto de métricas, i.e., variáveis independentes, abrangendo diversos aspectos da qualidade de software, como tamanho, complexidade, coesão, acoplamento, encapsulamento e

herança.

- O desenvolvimento de uma abordagem em duas etapas para detectar e classificar a gravidade de *code smell* baseado em métodos de comitê, utilizando o balanceamento de dados, o escalonamento de dados e a seleção de atributos. Em primeiro lugar, esta abordagem visa permitir a detecção de instâncias com ou sem *code smell* seguindo uma abordagem binária. Além disso, a classificação da gravidade do *code smell* utilizando uma abordagem multiclasse, apenas sobre as instâncias positivas. Com o intuito de evitar a interferência do grande número de instâncias negativas, comuns neste tipo de problema, no desempenho dos modelos.

# 5 A PROPOSTA

O objetivo principal desta proposta é desenvolver uma abordagem em duas etapas para detecção e classificação de gravidade de *code smell* baseada em transferência de aprendizado e métodos de comitê. Esta abordagem visa aplicar os avanços obtidos nesse área em projetos de softwares Java, que são a maioria de estudos realizados, através da Transferência de Aprendizado, para projetos de softwares *CSharp* com custos menores. Com o intuito de confirmar a validade desta proposta para resolver o problema da limitação de abordagens de detecção e classificação de *code smell* em projetos em *CSharp*, as hipóteses abaixo foram formuladas:

**Hipótese 1.** *A construção de um conjunto de dados abrangente e realista, baseado em métricas de qualidade de software, melhorará a capacidade dos algoritmos de aprendizado de máquina em detectar e classificar a gravidade de code smells.*

A hipótese é que um conjunto de dados bem construído, com métricas abrangentes e representativas, fornecerá aos algoritmos de aprendizado de máquina as informações necessárias para aprender e generalizar padrões complexos de *code smells*, melhorando assim a detecção e classificação.

**Hipótese 2.** *A abordagem em duas etapas, utilizando métodos de comitê, balanceamento de dados, escalonamento de dados e seleção de atributos, resultará em uma detecção e classificação mais precisa e eficiente da gravidade de code smells em comparação com os métodos existentes.*

A hipótese de que a abordagem em duas etapas proposta combina técnicas avançadas que têm demonstrado sucesso em outros contextos de aprendizado de máquina e análise de dados. Acredita-se que essa combinação permitirá uma detecção e classificação mais robustas e eficientes de *code smells*.

**Hipótese 3.** *A implementação da abordagem proposta resultará em melhorias na qualidade e manutenibilidade dos softwares desenvolvidos não só em Java, mas também em CSharp, independentemente do contexto ou equipe de desenvolvimento.*

A hipótese é que a abordagem proposta, focada na detecção e classificação de gravidade de *code smells*, pode proporcionar *insights* valiosos sobre áreas específicas que requerem atenção e melhorias nos *softwares* desenvolvidos em CSharp. Ao identificar e abordar proativamente os *code smells*, é possível reduzir a complexidade do código, au-

mentar a coesão, diminuir o acoplamento e, consequentemente, melhorar a qualidade geral do software. Além disso, ao melhorar a manutenibilidade do código, a abordagem pode facilitar e agilizar futuras atualizações, correções e implementações, contribuindo para um ciclo de vida de software mais eficiente e sustentável.

Diante deste contexto, as seguintes questões de pesquisa são propostas para guiar o desenvolvimento e a avaliação da abordagem:

**Questão de Pesquisa 1.** *Quais métricas são mais eficazes para representar a gravidade de diferentes tipos de code smells?*

**Questão de Pesquisa 2.** *Como a transferência de aprendizado pode ser aplicada para melhorar a detecção e classificação de code smells em linguagens de programação como CSharp?*

**Questão de Pesquisa 3.** *Como a abordagem proposta pode influenciar a qualidade e a manutenibilidade dos softwares desenvolvidos em CSharp?*

## 5.1 VISÃO GERAL DA PROPOSTA

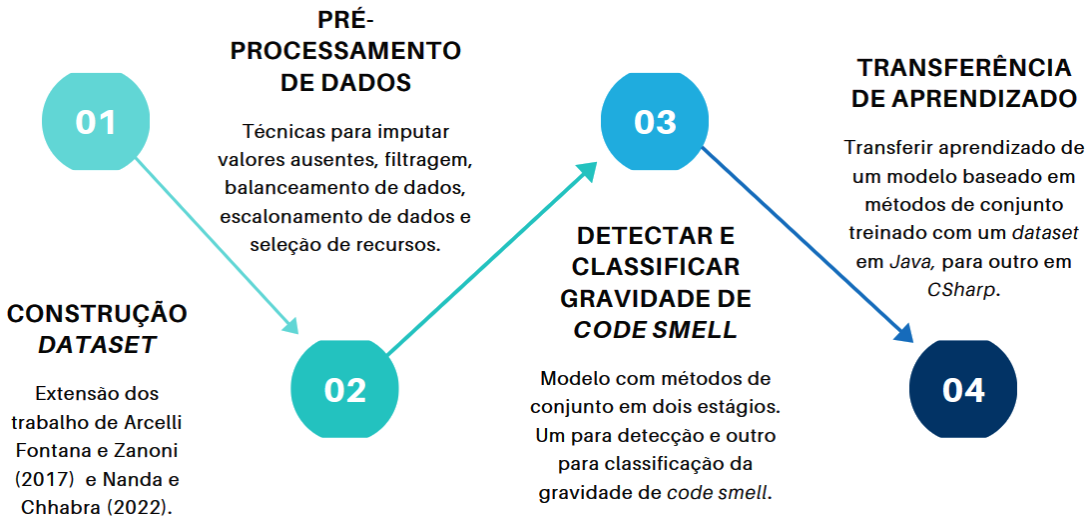


FIG. 5.1: Visão Geral da Proposta.

A Figura 5.1 ilustra os passos que essa proposta de dissertação pretende seguir para alcançar o seu objetivo. A primeira etapa da proposta (Seção 5.2) consiste na extensão dos trabalhos de Arcelli Fontana e Zanoni (2017) e Nanda e Chhabra (2022), porém com o intuito de construir um único conjunto de dados com instâncias de quatro classes de *code smell*, tanto em nível de método (FE e LM) quanto em nível de classe (GC e DC), diferentemente dos originais, em que cada tipo de *code smell* tem seu conjunto de dados

específico. O próximo passo (Seção 5.3) diz respeito às técnicas de pré-processamento de dados empregadas para lidar com valores faltantes, instâncias redudantes de *code smell*, balanceamento de dados, escalonamento dos dados e a seleção de atributos mais importantes. A terceira etapa (Seção 5.4) consiste na utilização de métodos já utilizados na classificação de gravidade, i.e., *XGBoost*, e do *CatBoost* que, salvo melhor juízo, não foi utilizado anteriormente. Com esses dois métodos de comitê, o intuito é construir um modelo de dois estágios: o primeiro para detecção de *code smell*; e o segundo para classificar a gravidade do *code smell*, apenas com as instâncias positivas (i.e., com *code smell*). Por fim, na quarta e última etapa da proposta (Seção 5.5) espera-se transferir o aprendizado do modelo, que é baseado em métodos de comitê treinados num conjunto de dados em Java para outro em CSharp,

## 5.2 CONSTRUÇÃO DO CONJUNTO DE DADOS

A distinção principal entre a abordagem metodológica adotada por Slivka et al. (2023) e aquela aplicada por Arcelli Fontana et al. (2016) e Arcelli Fontana e Zanoni (2017) reside no método de seleção de segmentos de código, resultando em uma distribuição de classes considerada irreal nas últimas. Essa distribuição distorcida pode comprometer a capacidade de generalização dos modelos de Aprendizado de Máquina treinados com esse conjunto de dados (DI NUCCI et al., 2018). Mais especificamente, a composição das instâncias no conjunto de dados, que influenciou fortemente o desempenho do modelo, difere significativamente de um ambiente de projeto de software realista (PALOMBA et al., 2018).

Pelas razões expostas acima e para atingir o objetivo desta proposta, combinaremos os dois conjuntos de dados em nível de método (FE e LM) de Arcelli Fontana e Zanoni (2017) e os dois conjuntos de dados em nível de classe (GC e DC) de Nanda e Chhabra (2022) em um único conjunto de dados. Esta abordagem visa representar com mais precisão o desafio encontrado em software real, no qual as instâncias de *code smell*, tanto no nível do método quanto no nível da classe, são influenciadas por diversas formas de gravidade de *code smell*.

Devido à complexidade desta estratégia, a tarefa de classificar a gravidade do *code smell* pode tornar-se mais desafiadora, exigindo uma solução que vá além do simples uso de técnicas de aprendizado de máquina, como otimização de hiperparâmetros dessas técnicas, padronização de dados e uso de técnicas de seleção de atributos.

## 5.3 PRÉ-PROCESSAMENTO DE DADOS

Nesta seção, são descritos os métodos de pré-processamento de dados que serão utilizadas para se alcançar os objetivos desta proposta. Dentre elas se destacam a imputação de valores faltantes, a filtragem, o escalonamento de dados, a seleção de atributos e o balanceamento de dados.

### 5.3.1 IMPUTAÇÃO DE VALORES FALTANTES

Esse pré-processamento de dados é necessário pois as variávies independentes dos conjuntos de dados de LM e FE possuem 84 métricas, em detrimento das 63 métricas dos conjuntos de dados de GC e DC. Então 21 métricas desse segundo subconjunto estão com valores faltantes, para resolver esse problema, procederemos conforme a seguir: i) para cada preditor, verificaremos a distribuição dos valores para a amostra de instâncias "*no code smell*" de *feature-envy* ou *long-method*; ii) e depois, imputaremos o valor, de acordo com cada distribuição padrão normal, para cada uma dessas amostras.

### 5.3.2 FILTRAGEM

Antes de aplicar outras técnicas de pré-processamento ao conjunto de dados combinado, que inclui rótulos e métricas como atributos, é crucial abordar questões específicas dentro desse conjunto, como a presença de dados duplicados. A exclusão desses dados será ralizada seguindo alguns princípios básicos. Por um lado, no caso de instâncias duplicadas em nível de método ou de classe com gravidades diferentes, serão removidas as instâncias de gravidade mais baixa. Por outro lado, no caso de instâncias com gravidades iguais, as instâncias das classes LM e GC serão priorizadas em detrimento das instâncias das classes FE e DC, respectivamente, uma vez que são *code smell* que têm maior impacto nos atributos de qualidade e propensão a *bugs* (LACERDA et al., 2020), sendo também considerado como os mais problemático pelos desenvolvedores (PALOMBA et al., 2014).

### 5.3.3 ESCALONAMENTO DE DADOS

Os estudos relacionados de Arcelli Fontana e Zanoni (2017), Rao et al. (2023) e Dewangan et al. (2023) que utilizaram escalonamento de dados em seus estudos para gravidade de *code smell*, utilizaram a normalização, mas não fizeram esclarecimentos mais aprofundados dos motivos e do impacto dessa técnica de pré processamento de dados nos modelos de aprendizado de máquina para classificação da gravidade do *code smell*. No caso da

padronização dos dados, Afrin et al. (2022), analisaram como esta técnica melhora o desempenho da Máquina de Vetores de Suporte (SVM) em termos de precisão, comparando a aplicação em quatro *code smell* (i.e., *Blob, Functional Decomposition, Swiss Army Knife, and Spaghetti Code*). Eles concluíram que a transformação de todas as unidades do atributo em uma unidade padronizada no conjunto de dados aumentou a taxa de aprendizado de conjunto de dados. Assim, o SVM teve um desempenho muito melhor com os dados processados (AFRIN et al., 2022). Contudo, nenhum estudo realizou a comparação da utilização das técnicas de normalização e de padronização num mesmo conjunto de dados de gravidade de *code smell*, como este estudo proposto pretende fazer.

### 5.3.4 SELEÇÃO DE ATRIBUTOS

Algumas técnicas de seleção de atributos foram aplicadas para classificação de gravidade de *code smell*, e.g., Ganho de Informação (ABDOU; DARWISH, 2022), PCA (RAO et al., 2023), Qui-quadrado (DEWANGAN et al., 2023). Destas técnicas, o resultado que mais se destacou foi a abordagem com *XGBoost* de Dewangan et al. (2023) sobre o conjunto de dados de LM de Arcelli Fontana e Zanoni (2017), obtendo 99,12% de acurácia. Nesta proposta este conjunto de dados é modificado, então, faz-se necessário aplicar qui-quadrado novamente. Além disso, faremos um comparação da aplicação das técnicas de seleção de atributos qui-quadrado e LDA. Esta último, até onde sabemos, não foi aplicado na classificação de gravidade de *code smell*.

### 5.3.5 BALANCEAMENTO DE DADOS

Ao final da primeira etapa desta proposta, espera-se que a proporção de classes com e sem *code smell* de código estejam equilibradas. Caso isso não ocorra, será aplicada a técnicas de balanceamento de dados para que a razão das classes negativa (gravidade 0) e positiva (gravidades 1, 2 e 3) seja 1:1, para facilitar o processo de detecção de *code smell* (Seção 5.4). Ademais, outra técnica de balanceamento de dados que será utilizada, especificamente para a classificação de gravidade de *code smell* (Seção 5.4), será a técnica de sobreamostragem SMOTE, que foi utilizada com resultados satisfatórios em alguns estudos relacionados, e.g., (NANDA; CHHABRA, 2022; ABDOU; DARWISH, 2022; RAO et al., 2023). Para exemplificar, Nanda e Chhabra (2022) alcançaram melhorias de acurácia máxima variando de 97% a 99%, em comparação com a faixa original (ARCELLI FONTANA; ZANONI, 2017) de 76% a 92%, em vários *code smell*. E, ainda, a abordagem de Rao et al. (2023), aplicando o SMOTE para balanceamento de classes, jun-

tamente com seleção de atributos com base na análise de componentes principais (PCA), resultou em uma pontuação de acurácia de gravidade de 0,99 usando os algoritmos *Random Forest* e Árvore de Decisão (AD) no contexto de detecção do *code smell* LM. Isto ressalta a eficácia do método SMOTE em conjunto com técnicas de seleção de atributos para aumentar a precisão do modelo.

## 5.4 DETECÇÃO DE *CODE SMELL* E CLASSIFICAÇÃO GRAVIDADE

A Figura 5.2 ilustra o processo de deteção de *code smell* e da classificação da sua gravidade, sob a qual os experimentos serão realizados

Inicialmente, procederemos com a otimização dos hiperparâmetros, utilizando o algoritmo *Grid Search*, para determinar as configurações mais adequadas para cada técnica de aprendizado de máquina (XGBoost e CatBoost). Posteriormente, validaremos os modelos por meio de validação cruzada de 5 vezes, aplicando o modelo mais adequado para detecção de *code smell*.

Depois de validar o modelo, utilizaremos uma abordagem em duas etapas para detectar e classificar a gravidade de *code smell* com base em métodos de comitê, que utilizam escalonamento de dados e seleção de atributos. Em primeiro lugar, esta abordagem permite a detecção de instâncias com ou sem *code smell* seguindo uma abordagem binária. Além disso, a classificação da gravidade do *code smell* não é restringida pelo grande número de instâncias negativas, comuns neste tipo de problema, uma vez que essas instâncias são previamente descartadas na segunda etapa da abordagem, deixando apenas as instâncias positivas para serem tratadas pelo subseqüente abordagem multiclasse, que é mais complexa que a primeira.

Diferentemente de estudos anteriores, e.g., (ABDOU; DARWISH, 2022; HU et al., 2023; DEWANGAN et al., 2023) que utilizaram os métodos de comitê *Adaboost, Random Forest e XGBoost*, com bons resultados, para classificar a gravidade de *code smell*, consta nesta proposta a utilização do *CatBoost* que, salvo melhor juízo, não foi utilizado anteriormente para este fim. *CatBoost* foi utilizado por Alkharabsheh et al. (2022) para detectar *God-class*, obtendo o mais alto desempenho dentre outros 27 classificadores utilizados. Isto reforça a importância da aplicação dos métodos de comitê em cenários reais onde os dados geralmente são desequilibrados pela natureza inerente dos *code smells*.

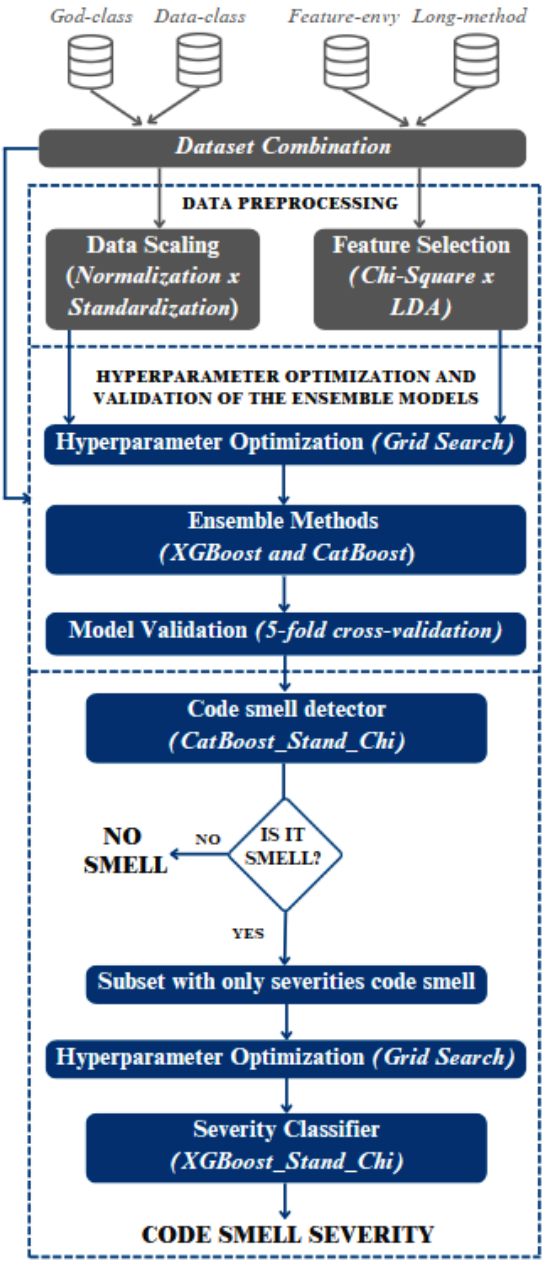


FIG. 5.2: Processo de detecção de *code smell* e da classificação da sua gravidade.

### 5.4.1 OTIMIZAÇÃO DE HIPERPARÂMETROS E VALIDAÇÃO DOS MODELOS

Após a aplicação de técnicas de pré-processamento de dados (escalonamento de dados e seleção de atributos), procederemos uma otimização dos hiperparâmetros, utilizando o algoritmo *Grid Search*, para determinar as configurações mais adequadas para os hiperparâmetros de cada método de comitê. Esses hiperparâmetros serão configurados para otimizar o modelo com base na métrica *f1-macro*, sem considerar, portanto, o desequilíbrio entre as classes alvo. Além disso, para melhorar o desempenho dos modelos, utilizaremos um método de validação cruzada de 5 vezes na técnica de otimização de hiperparâmetros.

Esta abordagem foi selecionada devido à presença de classes com instâncias limitadas.

A seguir, serão analisados os resultados de acurácia dos modelos de comitê combinados ou não com as técnicas de pré-processamento de dados para escolher o modelo mais adequado para cada fase da abordagem proposta, ou seja, um detector de *code smell* e um classificador de gravidade de *code smell*.

Por último, será utilizada uma metodologia de validação para calcular o desempenho de cada experimento para evitar o ajuste excessivo do algoritmo no conjunto de dados de teste. Os conjuntos serão calculados usando validação cruzada de 5 vezes para dividir os conjuntos de dados em cinco segmentos, cinco vezes para o treinamento do algoritmo. Em cada conjunto de dados de replicação, uma parte é avaliada como conjunto de teste e as outras são avaliadas como conjunto de treinamento.

### 5.4.2 DETECTOR E DE *CODE SMELL* E CLASSIFICADOR DE GRAVIDADE

O modelo mais adequado será escolhido para cada etapa de nossa abordagem, após configurar hiperparâmetros, validar os modelos e analisar as melhorias de desempenho de cada método de comitê, usando técnicas de escalonamento de dados e seleção de atributos.

Ao contrário de Arcelli Fontana e Zanoni (2017) e Nanda e Chhabra (2022), usaremos métodos de comitê com escalonamento de dados e seleção de atributos para detectar e classificar *code smells*. O primeiro utilizou normalização sem mostrar seu uso e não utilizou abordagens sofisticadas de seleção de atributos, enquanto o último não utilizou escalonamento de dados e seleção de atributos, baseando sua abordagem apenas no método de comitê e balanceamento de dados.

Na abordagem proposta, a primeira etapa consiste em um módulo de detecção de *code smell* composto pelo método comitê *CatBoost*, padronização e Qui-quadrado. A etapa seguinte visa classificar a gravidade dos *code smell*, utilizando padronização e qui-quadrado, mas com o método comitê *XGBoost*.

Ao final, o conjunto de experimentos serão convertidos e para avaliar a qualidade do modelo do método de conjunto serão considerados quatro parâmetros de desempenho: precisão, sensibilidade, *f1-score* e acurácia.

## 5.5 TRANSFERÊNCIA DE APRENDIZADO

Para finalizar a proposta ora apresentada, será realizado a transferência de aprendizado do modelo de detecção de classificação da gravidade de *code smell* (Seção 5.4), com o conjunto de dados pré-processados (Seção 5.3), composto por instâncias sem *code smell* e

das múltiplas instâncias de níveis de gravidades de quatro *code smells*, i.e., FE, LM, GC e DC (Seção 5.2). Essa transferência envolve o treinamento em um conjunto de dados de softwares baseados em Java, e os testes em um conjunto dados de softwares baseados em *CSharp*, com métricas de softwares orientadas a objetos como preditores. O conjunto de dados CSharp escolhido é o de Slivka et al. (2023), por se tratar um razoável conjunto de dados anotado seguindo o paradgma prescritivo e com critérios de detecção especificados com consistência. Existem outros conjunto de dados em *CSharp*, e.g., (WANG et al., 2012), porém com poucas instâncias, sem utilização de mecanismos de validação e sem foco no tratamento da gravidade de *code smell* (ZAKERI-NASRABADI et al., 2023). Sharma et al. (2021) abordaram a transferência de aprendizado de modelos treinados em Java para conjunto de dados em *CSharp*, utilizando aprendizado profundo sem o auxílio de métricas de software como preditores. Diferentemente, nossa proposta consiste em utilizar as métricas de softwares orientados a objetos, que não dependem de linguagem específica, com modelos treinados com métodos de comitê homogêneos.

### 5.5.1 CONTRIBUIÇÕES ESPERADAS

Com base nas informações apresentadas, esta Proposta de Dissertação, alinhada com a linha de pesquisa em Engenharia de Sistemas e Informação, se propõe à elaboração de uma abordagem para detectar e classificar a gravidade do *code smell* em um único conjunto de dados que cobre instâncias de quatro tipos diferentes de *code smell*, i.e., *God-class*, *Data-class*, *Feature-envy* e *Long-method* baseado em pré-processamento de dados, métodos de comitê e transferência de aprendizado. Esta abordagem surge como resposta às lacunas identificadas na literatura e às oportunidades de aprimoramento percebidas no âmbito acadêmico. As principais contribuições desta pesquisa são:

- A criação de um conjunto de dados que contenha instâncias de múltiplos tipos de gravidade de *code smell* que seja mais realista e facilite aos algoritmos de AM aprender melhor as nuances dessas instâncias, com base em um conjunto de métricas, ou seja, variáveis independentes, abrangendo diversos aspectos da qualidade de software, como tamanho, complexidade, coesão, acoplamento, encapsulamento e herança.

- A proposta de uma abordagem em duas etapas para detectar e classificar a gravidade do *code smell* com base em métodos de comitê, que utilizam escalonamento de dados, balanceamento de dados e seleção de atributos para melhorar a performance dos modelos. Em primeiro lugar, esta abordagem permite a detecção de instâncias

com ou sem *code smell* seguindo uma abordagem binária. Além disso, a classificação da gravidade do *code smell* não é restringida pelo grande número de instâncias negativas, comuns neste tipo de problema, uma vez que essas instâncias são previamente descartadas na segunda etapa da abordagem, deixando apenas as instâncias positivas para serem tratadas pelo subseqüente abordagem multiclasse, que é mais complexa que a primeira.

- Transferência de Aprendizado de um conjunto de dados de softwares baseados em *Java*, para um conjunto de dados de softwares baseados em *CSharp*, com a utilização de Métodos de Comitê já empregados anteriormente (XGBoost) para classificação de gravidade de *code smell* e, também, o CatBoost, que não foi empregado para este fim.

# 6 PLANO DE AÇÃO

Esta Proposta compreenderá um conjunto articulado de atividades, de acordo com o descrito na Tabela 6.1.

TAB. 6.1: Descrição das atividades planejadas.

| | **Atividades** |
|---|---|
| 1 | Revisão Sistemática da Literatura |
| 2 | Submissão da Revisão Sistemática da Literura a um periódico relevante da área de Engenharia de Software |
| 3 | Construção do conjunto de dados |
| 4 | Aplicação das técnicas de pré-processamento de dados |
| 5 | Desenvolvimento da abordagem de detecção e classificação de gravidade de *code smell* |
| 6 | Experimentação da abordagem de detecção e classificação de gravidade de *code smell* |
| 7 | Escrita de artigo baseado nesta proposta e dos resultados parciais obtidos |
| 8 | Submissão do artigo com os resultados parciais obtidos para o SBES38 |
| 9 | Desenvolvimento da abordagem de transferência de aprendizado |
| 10 | Experimentação da abordagem de transferência de aprendizado |
| 11 | Escrita da artigo com os resultados de todos os experimentos realizados |
| 12 | Escrita da dissertação |
| 13 | Defesa |

## 6.1 METODOLOGIA

A Proposta adotará uma abordagem metodológica de pesquisa empírica com a realização de experimentos com base na abordagem proposta, com o intuito de analisar a aplicação, na deteção e classificação de gravidade de *code smell* das técnicas e métodos abaixo:

a) Técnicas de escalonamento de dados (Normalização e Padronização).

b) Técnicas de seleção de atributos (Qui-quadro e LDA).

c) Métodos de comitê (*XGBoost* e *CatBoost*).

d) Transferência de aprendizado de um conjunto de dados combinado baseado em softwares *Java* (ARCELLI FONTANA; ZANONI, 2017; NANDA; CHHABRA, 2022), para outro conjunto de dados baseados em softwares *CSharp* (SLIVKA et al., 2023).

Na primeira etapa de experimentação, serão realizados experimentos para determinar a melhor combinação de técnicas de escalonamento de dados, seleção de atributos e métodos de comitê para deteção e classificação de gravidade de *code smell* no conjunto de dados Java construído. Esse conjunto de dados será dividido na proporção de 70% para dados de treinamento e 30% para dados de teste.

Na segunda etapa de experimentação, na transferência de aprendizado, para fazer com que os domínios de origem e de destino compartilhem o mesmo conjunto de atributos e se tornem homogêneos, i.e, com as mesmas métricas de software, utilizaremos a Calculadora de métricas de código Java de Aniche (2015), para incluir esses atributos no conjunto de dados de Slivka et al. (2023), caso seja necessário. A Calculadora de Aniche (2015) é amplamente adotado nos estudos que pesquisam detecção de *code smell* (SLIVKA et al., 2023). Nessa etapa, os dados de treinamento serão compostos por todos os dados do conjunto de dados Java, e o conjunto de teste será composto pelo conjunto de dados CSharp de Slivka et al. (2023).

Os modelos AM serão implementados no *Google Colaboratory*, utilizando a biblioteca de aprendizado de máquina em Python e scikit-learn, e incorporando as bibliotecas XGBoost[5] e CatBoost[6]. Além disso, para determinar as configurações mais adequadas para os hiperparâmetros de cada método de comitê será realizada uma otimização dos hiperparâmetros, utilizando o algoritmo *Grid Search*. Esses hiperparâmetros serão configurados para otimizar o modelo com base na métrica *f1-macro*, sem considerar, portanto, o desequilíbrio entre as classes alvo.

A fim de evitar o ajuste excessivo do algoritmo no conjunto de dados de teste será utilizada a metodologia de validação cruzada de 5 vezes, para calcular o desempenho de cada experimento. Essa metodologia consiste em dividir os conjuntos de dados em cinco segmentos, cinco vezes para o treinamento do algoritmo. Em cada conjunto de dados de replicação, uma parte é avaliada como conjunto de teste e as outras são avaliadas como conjunto de treinamento.

Por fim, para avaliar o desempenho dos algoritmos de aprendizado de máquina, serão considerados quatro dos principais parâmetros de desempenho utilizados na detecção de *code smell* (AZEEM et al., 2019; DEWANGAN et al., 2023): Acurácia, *F1-Score*, Sensibilidade e Precisão. Esses parâmetros serão calculados usando uma matriz de confusão que contém as informações reais e previstas estimadas pelas classificações de detecção de padrões de projeto (CATAL, 2012), i.e., Verdadeiro Positivo (VP), Falso Positivo (FP),

https://xgboost.readthedocs.io/en/stable/
https://catboost.ai/en/docs/

Verdadeiro Negativo (VN) e Falso Negativo (FN).

## 6.2 CRONOGRAMA

O cronograma para o desenvolvimento das atividades relacionadas a esta proposta pode ser visto na Figura 6.1.

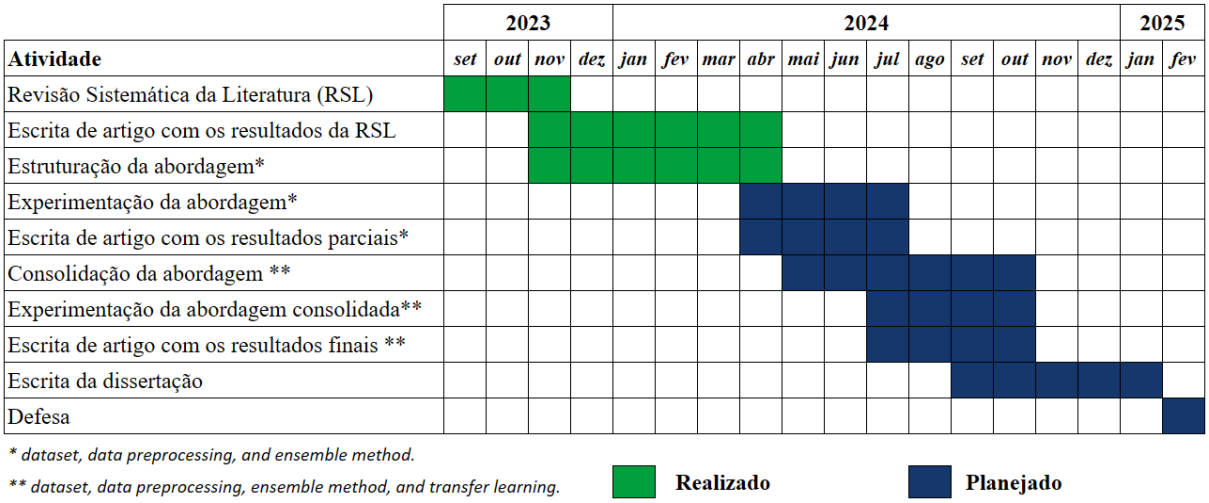


FIG. 6.1: Cronograma da Proposta de Dissertação.

# 7 REFERÊNCIAS BIBLIOGRÁFICAS


ABDOU, A.; DARWISH, N. Severity classification of software code smells using machine learning techniques: A comparative study. **Journal of Software: Evolution and Process**, v. 36, 2022. Disponível em: <https://doi.org/10.1002/smr.2454>. Acesso em: 04/03/2024.

ABUHASSAN, A.; ALSHAYEB, M. ; GHOUTI, L. Software smell detection techniques: A systematic literature review. **Journal of Software: Evolution and Process**, v. 33, n. 3, 2021. Disponível em: <https://doi.org/10.1002/smr.2320>. Acesso em: 04/03/2024.

AFRIN, M.; AKTER ASMA, S.; AKHTER, N.; HASAN RIDOY, J.; SHARMILA SAUDA, S. ; ABU TAHER, K. A hybrid approach to investigate anti-pattern from source code. In: 25TH INTERNATIONAL CONFERENCE ON COMPUTER AND INFORMATION TECHNOLOGY (ICCIT), 25., 2022. **Anais**... [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2022, p. 888–892.

ALAZBA, A.; ALJAMAAN, H. Code smell detection using feature selection and stacking ensemble: An empirical investigation. **Information and Software Technology**, v. 138, p. 106648, 2021. Disponível em: <https://doi.org/10.1016/j.infsof.2021.106648>. Acesso em: 01/03/2024.

ALAZBA, A.; ALJAMAAN, H. ; ALSHAYEB, M. Deep learning approaches for bad smell detection: a systematic literature review. **Empirical Software Engineering**, v. 28, 2023. Disponível em: <https://doi.org/10.1007/s10664-023-10312-z>. Acesso em: 01/03/2024.

ALJAMAAN, H. Voting heterogeneous ensemble for code smell detection. In: 2021 20TH IEEE INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND APPLICATIONS (ICMLA), 20., 2021. **Anais**... [S.l.: s.n.], 2021, p. 897–902.

ALKHARABSHEH, K.; ALAWADI, S.; KEBANDE, V. R.; CRESPO, Y.; FERNÁNDEZ-DELGADO, M. ; TABOADA, J. A. A comparison of machine learning algorithms on design smell detection using balanced and imbalanced dataset: A study of god

class. **Information and Software Technology**, v. 143, p. 106736, 2022. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950584921001865>. Acesso em: 01/04/2024.

ANICHE, M. Java code metrics calculator (ck). **Available in https://github.com/mauricioaniche/ck**, v. release ck-0.7.0, 2015. Disponível em: <https://github.com/mauricioaniche/ck/>. Acesso em: 05/04/2024.

ARCELLI FONTANA, F.; MÄNTYLÄ, M. V.; ZANONI, M. ; MARINO, A. Comparing and experimenting machine learning techniques for code smell detection. **Empirical Softw. Engg.**, v. 21, n. 3, p. 1143–1191, 2016.

ARCELLI FONTANA, F.; ZANONI, M. Code smell severity classification using machine learning techniques. **Knowledge-Based Systems**, v. 128, p. 43–58, 2017.

AZEEM, M. I.; PALOMBA, F.; SHI, L. ; WANG, Q. Machine learning techniques for code smell detection: A systematic literature review and meta-analysis. **Information and Software Technology**, v. 108, p. 115–138, 2019.

BENGIO, Y.; COURVILLE, A. ; VINCENT, P. Representation learning: A review and new perspectives. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 35, n. 8, p. 1798–1828, 2013.

CATAL, C. Performance evaluation metrics for software fault prediction studies. **Acta Polytechnica Hungarica**, v. 9, 2012. Disponível em: <https://www.semanticscholar.org/paper/Performance-Evaluation-Metrics-for-Software-Fault-Catal/dbd4b674fbd4bb5bc91b95a59eac9d1f3f81209b>. Acesso em: 05/04/2024.

CHAWLA, N. V.; BOWYER, K. W.; HALL, L. O. ; KEGELMEYER, W. P. Smote: Synthetic minority over-sampling technique. **J. Artif. Int. Res.**, v. 16, n. 1, p. 321–357, 2002.

DEWANGAN, S.; RAO, R. S.; CHOWDHURI, S. R. ; GUPTA, M. Severity classification of code smells using machine-learning methods. **SN Computer Science**, v. 4, n. 5, 2023. Disponível em: <https://doi.org/10.1007/s42979-023-01979-8>. Acesso em: 20/03/2024.

DEWANGAN, S.; RAO, R. S.; MISHRA, A. ; GUPTA, M. A novel approach for code smell detection: An empirical study. **IEEE Access**, v. 9, p. 162869–162883,

2021. Disponível em: <https://doi.org/10.1109/ACCESS.2021.3133810>. Acesso em: 25/03/2024.

DI NUCCI, D.; PALOMBA, F.; TAMBURRI, D. A.; SEREBRENIK, A. ; DE LUCIA, A. Detecting code smells using machine learning techniques: Are we there yet?. In: IEEE 25TH INTERNATIONAL CONFERENCE ON SOFTWARE ANALYSIS, EVOLUTION AND REENGINEERING (SANER), 25., 2018. **Anais**... [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2018, p. 612–621.

FONTANA, F. A.; FERME, V.; ZANONI, M. ; ROVEDA, R. Towards a prioritization of code debt: A code smell intensity index. In: 2015 IEEE 7TH INTERNATIONAL WORKSHOP ON MANAGING TECHNICAL DEBT (MTD), 7., 2015. **Anais**... [S.l.: s.n.], 2015, p. 16–24.

FOWLER, M.; BECK, K.; BRANT, J.; OPDYKE, W. ; ROBERTS, D. **Refactoring: Improving the Design of Existing Code**. USA: Addison-Wesley Longman Publishing Co., Inc., 1999. ISBN 0201485672.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. In: PROCEEDINGS OF THE SECOND EUROPEAN CONFERENCE ON COMPUTATIONAL LEARNING THEORY, 2., EUROCOLT '95, 2., 1995. **Anais**... Berlin, Heidelberg: Springer-Verlag, 1995, p. 23–37.

GUPTA, A.; CHAUHAN, N. K. A severity-based classification assessment of code smells in kotlin and java application. **Arabian Journal for Science and Engineering**, v. 47, n. 2, p. 1831–1848, 2022.

GUPTA, R.; SINGH, S. K. A novel transfer learning method for code smell detection on heterogeneous data: A feasibility study. **SN Computer Science**, v. 4, n. 6, 2023. Disponível em: <https://doi.org/10.1007/s42979-023-02157-6>. Acesso em: 03/03/2024.

HADJ-KACEM, M.; BOUASSIDA, N. Deep representation learning for code smells detection using variational auto-encoder. **Proceedings of the International Joint Conference on Neural Networks**, v. 2019-July, p. 1–8, 2019.

HALL, M.; FRANK, E.; HOLMES, G.; PFAHRINGER, B.; REUTEMANN, P. ; WITTEN, I. H. The weka data mining software: An update. **SIGKDD Explor. Newsl.**, v. 11, n. 1, p. 10–18, 2009.

HO, A.; BUI, A. M. T.; NGUYEN, P. T. ; DI SALLE, A. Fusion of deep convolutional and lstm recurrent neural networks for automated detection of code smells. In: PROCEEDINGS OF THE 27TH INTERNATIONAL CONFERENCE ON EVALUATION AND ASSESSMENT IN SOFTWARE ENGINEERING (EASE '23), 27., 2023. **Anais**... [S.l.]: Association for Computing Machinery, 2023, p. 229–234.

HU, W.; LIU, L.; YANG, P.; ZOU, K.; LI, J.; LIN, G. ; XIANG, J. Revisiting "code smell severity classification using machine learning techniques". In: IEEE 47TH ANNUAL COMPUTERS, SOFTWARE, AND APPLICATIONS CONFERENCE (COMPSAC), 47., 2023. **Anais**... [S.l.]: IEEE Computer Society, 2023, p. 840–849.

KAUR, I.; KAUR, A. A novel four-way approach designed with ensemble feature selection for code smell detection. **IEEE Access**, v. 9, p. 8695–8707, 2021.

KITCHENHAM, B.; PEARL BRERETON, O.; BUDGEN, D.; TURNER, M.; BAILEY, J. ; LINKMAN, S. Systematic literature reviews in software engineering – a systematic literature review. **Information and Software Technology**, v. 51, n. 1, p. 7–15, 2009. Acesso em: Special Section - Most Cited Articles in 2002 and Regular Research Papers.

KITCHENHAM, B. A.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. [S.l.: s.n.], 2007. (Relatório Técnico, EBSE 2007-001).

KOCOŃ, J.; FIGAS, A.; GRUZA, M.; PUCHALSKA, D.; KAJDANOWICZ, T. ; KAZIENKO, P. Offensive, aggressive, and hate speech analysis: From data-centric to human-centered approach. **Information Processing Management**, v. 58, n. 5, p. 102643, 2021. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0306457321001333>. Acesso em: 02/04/2024.

LACERDA, G.; PETRILLO, F.; PIMENTA, M. ; GUÉHÉNEUC, Y. G. Code smells and refactoring: A tertiary systematic review of challenges and observations. **Journal of Systems and Software**, v. 167, p. 110610, 2020. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121220300881>. Acesso em: 25/03/2024.

LANZA, M.; MARINESCU, R. **Object-oriented metrics in practice: using software metrics to characterize, evaluate, and improve the design of object-oriented systems**. [S.l.]: Springer Science & Business Media, 2007.

LIU, H.; JIN, J.; XU, Z.; ZOU, Y.; BU, Y. ; ZHANG, L. Deep learning based code smell detection. **IEEE Transactions on Software Engineering**, v. 47, n. 9, p. 1811–1837, 2021.

LIU, H.; XU, Z. ; ZOU, Y. Deep learning based feature envy detection. In: PROCEEDINGS OF THE 33RD ACM/IEEE INTERNATIONAL CONFERENCE ON AUTOMATED SOFTWARE ENGINEERING, 33., ASE '18, 33., 2018. **Anais**... New York, NY, USA: Association for Computing Machinery, 2018, p. 385–396.

LUIZ, F. C.; DE OLIVEIRA, B. R. ; PARREIRAS, F. S. Machine learning techniques for code smells detection: An empirical experiment on a highly imbalanced setup. In: PROCEEDINGS OF THE XV BRAZILIAN SYMPOSIUM ON INFORMATION SYSTEMS (SBSI '19), 15., 2019. **Anais**... [S.l.]: Association for Computing Machinery, 2019, p. 1–8.

MA, W.; YU, Y.; RUAN, X. ; CAI, B. Pre-trained model based feature envy detection. In: IEEE/ACM 20TH INTERNATIONAL CONFERENCE ON MINING SOFTWARE REPOSITORIES (MSR), 20., 2023. **Anais**... [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2023, p. 430–440.

MADEYSKI, L.; LEWOWSKI, T. Mlcq: Industry-relevant code smell data set. In: PROCEEDINGS OF THE 24TH INTERNATIONAL CONFERENCE ON EVALUATION AND ASSESSMENT IN SOFTWARE ENGINEERING, 24., EASE '20, 24., 2020. **Anais**... New York, NY, USA: Association for Computing Machinery, 2020, p. 342–347. Disponível em: <https://doi.org/10.1145/3383219.3383264>. Acesso em: 03/04/2024.

MALHOTRA, R.; JAIN, B. ; KESSENTINI, M. Examining deep learning's capability to spot code smells: A systematic literature review. **Cluster Computing**, v. 26, n. 6, p. 3473–3501, 2023.

MARINESCU, C.; MARINESCU, R.; MIHANCEA, P.; RATIU, D. ; WETTEL, R. iplasma: An integrated platform for quality assessment of object-oriented design.. In: PROCEEDINGS OF THE 21ST IEEE INTERNATIONAL CONFERENCE ON

SOFTWARE MAINTENANCE - INDUSTRIAL AND TOOL VOLUME, ICSM 2005, 25-30 SEPTEMBER 2005, BUDAPEST, HUNGARY, 21., 2005. **Anais**... [S.l.]: IEEE, 2005, p. 77–80.

MARINESCU, R. Measurement and quality in object-oriented design. In: 21ST IEEE INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE (ICSM'05), 21., 2005. **Anais**... Budapest, Hungary: IEEE, 2005, p. 701–704.

MCCARTY, D. A.; KIM, H. W. ; LEE, H. K. Evaluation of light gradient boosted machine learning technique in large scale land use and land cover classification. **Environments**, v. 7, n. 10, 2020. Disponível em: <https://doi.org/10.3390/environments710008>. Acesso em: 22/04/2024.

MOHA, N.; GUEHENEUC, Y.-G.; DUCHIEN, L. ; LE MEUR, A.-F. Decor: A method for the specification and detection of code and design smells. **IEEE Transactions on Software Engineering**, v. 36, n. 1, p. 20–36, 2010.

MUHAMMAD ALI, P.; FARAJ, R. **Data Normalization and Standardization: A Technical Report**. [S.l.]: Machine Learning Technical Reports, 2014.

NANDA, J.; CHHABRA, J. K. Sshm: Smote-stacked hybrid model for improving severity classification of code smell. **International Journal of Information Technology (Singapore)**, v. 14, n. 5, p. 2701–2707, 2022.

NGUYEN THANH, B.; NGUYEN N. H, M.; LE THI MY, H. ; NGUYEN THANH, B. Ml-codesmell: A code smell prediction dataset for machine learning approaches. In: PROCEEDINGS OF THE 11TH INTERNATIONAL SYMPOSIUM ON INFORMATION AND COMMUNICATION TECHNOLOGY (SOICT '22), 11., 2022. **Anais**... [S.l.]: Association for Computing Machinery, 2022, p. 368 – 374.

NONGPONG, K. **Integrating "code smells"detection with refactoring tool support**. 2012. Tese (Ph.D. Dissertation) – University of Wisconsin at Milwaukee, USA, 2012. Acesso em: AAI3523928.

PALOMBA, F.; BAVOTA, G.; DI PENTA, M.; OLIVETO, R. ; DE LUCIA, A. Do they really smell bad? a study on developers' perception of bad code smells. In: 2014 IEEE INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE AND EVOLUTION, 30., 2014. **Anais**... Victoria, BC, Canada: IEEE, 2014, p. 101–110.

PALOMBA, F.; BAVOTA, G.; PENTA, M. D.; FASANO, F.; OLIVETO, R. ; LUCIA, A. D. On the diffuseness and the impact on maintainability of code smells: a large scale empirical investigation. **Empirical Software Engineering**, v. 23, n. 3, p. 1188 – 1221, 2018. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85026909182doi=10.1007

PALOMBA, F.; DI NUCCI, D.; TUFANO, M.; BAVOTA, G.; OLIVETO, R.; POSHYVANYK, D. ; DE LUCIA, A. Landfill: An open dataset of code smells with public evaluation. In: 2015 IEEE/ACM 12TH WORKING CONFERENCE ON MINING SOFTWARE REPOSITORIES, 12., 2015. **Anais**... [S.l.: s.n.], 2015, p. 482–485.

PARASHAR, A.; PARASHAR, A.; DING, W.; SHABAZ, M. ; RIDA, I. Data preprocessing and feature selection techniques in gait recognition: A comparative study of machine learning and deep learning approaches. **Pattern Recognition Letters**, v. 172, p. 65–73, 2023.

PATNAIK, A.; PADHY, N. Does code complexity affect the quality of real-time projects? detection of code smell on software projects using machine learning algorithms. In: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON DATA SCIENCE, MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE, 21., DSMLAI '21', 21., 2022. **Anais**... New York, NY, USA: Association for Computing Machinery, 2022, p. 178–185.

PECORELLI, F.; DI NUCCI, D.; DE ROOVER, C. ; DE LUCIA, A. On the role of data balancing for machine learning-based code smell detection. In: PROCEEDINGS OF THE 3RD ACM SIGSOFT INTERNATIONAL WORKSHOP ON MACHINE LEARNING TECHNIQUES FOR SOFTWARE QUALITY EVALUATION (MALTESQUE 2019), 3., 2019. **Anais**... [S.l.]: Association for Computing Machinery, Inc, 2019, p. 19–24.

PECORELLI, F.; PALOMBA, F.; DI NUCCI, D. ; DE LUCIA, A. Comparing heuristic and machine learning approaches for metric-based code smell detection. In: IEEE/ACM 27TH INTERNATIONAL CONFERENCE ON PROGRAM COMPREHENSION (ICPC), 27., 2019. **Anais**... [S.l.]: IEEE Computer Society, 2019, p. 93–104.

POLIKAR, R. Ensemble based systems in decision making. **IEEE Circuits and Systems Magazine**, v. 6, n. 3, p. 21–45, 2006.

RAO, R. S.; DEWANGAN, S.; MISHRA, A. ; GUPTA, M. A study of dealing class imbalance problem with machine learning methods for code smell severity detection using pca-based feature selection technique. **Scientific Reports**, v. 13, n. 1, 2023. Disponível em: <https://doi.org/10.1038/s41598-023-43380-8>. Acesso em: 02/03/2024.

ROKACH, L. Ensemble-based classifiers. **Artificial Intelligence Review**, v. 33, p. 1–39, 2010.

ROMERO, E.; SOPENA, J. M. Performing feature selection with multilayer perceptrons. **IEEE Transactions on Neural Networks**, v. 19, n. 3, p. 431–441, 2008.

RÖTTGER, P.; VIDGEN, B.; HOVY, D. ; PIERREHUMBERT, J. B. Two contrasting data annotation paradigms for subjective nlp tasks. **arXiv preprint arXiv:2112.07475**, v. 2, 2021. Disponível em: <https://doi.org/10.48550/arXiv.2112.07475>. Acesso em: 03/03/2024.

SAGI, O.; ROKACH, L. Ensemble learning: A survey. **WIREs Data Mining and Knowledge Discovery**, v. 8, n. 4, p. e1249, 2018. Disponível em: <https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1249>. Acesso em: 22/04/2024.

SHARMA, T.; EFSTATHIOU, V.; LOURIDAS, P. ; SPINELLIS, D. Code smell detection by deep direct-learning and transfer-learning. **Journal of Systems and Software**, v. 176, 2021. Disponível em: <https://doi.org/10.1016/j.jss.2021.110936>. Acesso em: 05/03/2024.

SLIVKA, J.; LUBURIĆ, N.; PROKIĆ, S.; GRUJIĆ, K.-G.; KOVAČEVIĆ, A.; SLADIĆ, G. ; VIDAKOVIĆ, D. Towards a systematic approach to manual annotation of code smells. **Science of Computer Programming**, v. 230, p. 102999, 2023. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167642323000813>. Acesso em: 20/03/2024.

TAHIR, A.; DIETRICH, J.; COUNSELL, S.; LICORISH, S. ; YAMASHITA, A. A large scale study on how developers discuss code smells and anti-pattern in stack exchange sites. **Information and Software Technology**, v. 125, p. 106333, 2020. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950584920300926>. Acesso em: 04/04/2024.

TARWANI, S.; CHUG, A. Application of deep learning models for code smell prediction. In: 10TH INTERNATIONAL CONFERENCE ON RELIABILITY, INFOCOM TECHNOLOGIES AND OPTIMIZATION (TRENDS AND FUTURE DIRECTIONS) (ICRITO), 10., 2022. **Anais**... [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2022, p. 1–5.

TEMPERO, E.; ANSLOW, C.; DIETRICH, J.; HAN, T.; LI, J.; LUMPE, M.; MELTON, H. ; NOBLE, J. The qualitas corpus: A curated collection of java code for empirical studies. In: 2010 ASIA PACIFIC SOFTWARE ENGINEERING CONFERENCE, 17., 2010. **Anais**... Sydney, NSW, Australia: IEEE, 2010, p. 336–345.

TSANTALIS, N.; CHATZIGEORGIOU, A. Identification of move method refactoring opportunities. **IEEE Transactions on Software Engineering**, v. 35, n. 3, p. 347–367, 2009.

TSANTALIS, N.; CHATZIGEORGIOU, A. Ranking refactoring suggestions based on historical volatility. In: 2011 15TH EUROPEAN CONFERENCE ON SOFTWARE MAINTENANCE AND REENGINEERING, 15., 2011. **Anais**... [S.l.: s.n.], 2011, p. 25–34.

VIDAL, S.; VAZQUEZ, H.; DIAZ-PACE, J. A.; MARCOS, C.; GARCIA, A. ; OIZUMI, W. Jspirit: a flexible tool for the analysis of code smells. In: 2015 34TH INTERNATIONAL CONFERENCE OF THE CHILEAN COMPUTER SCIENCE SOCIETY (SCCC), 34., 2015. **Anais**... Santiago, Chile: IEEE, 2015, p. 1–6.

WANG, S.; MINKU, L. L. ; YAO, X. Resampling-based ensemble methods for online class imbalance learning. **IEEE Transactions on Knowledge and Data Engineering**, v. 27, n. 5, p. 1356–1368, 2015.

WANG, X.; DANG, Y.; ZHANG, L.; ZHANG, D.; LAN, E. ; MEI, H. Can i clone this piece of code here?. In: PROCEEDINGS OF THE 27TH IEEE/ACM INTERNATIONAL CONFERENCE ON AUTOMATED SOFTWARE ENGINEERING, 12., ASE '12, 12., 2012. **Anais**... New York, NY, USA: Association for Computing Machinery, 2012, p. 170–179. Disponível em: <https://doi.org/10.1145/2351676.2351701>. Acesso em: 03/04/2024.

WANG, Z.; REN, H.; LU, R. ; HUANG, L. Stacking based lightgbm-catboost-randomforest algorithm and its application in big data modeling. In: 4TH INTER-

NATIONAL CONFERENCE ON DATA-DRIVEN OPTIMIZATION OF COMPLEX SYSTEMS (DOCS), 4., 2022. **Anais**... Chengdu, China: IEEE, 2022, p. 1–6.

WEISS, K.; KHOSHGOFTAAR, T. M. ; WANG, D. A survey of transfer learning. **Journal of Big Data**, v. 3, 2016. Disponível em: <https://doi.org/10.1186/s40537-016-0043-6>. Acesso em: 03/03/2024.

WOHLIN, C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: PROCEEDINGS OF THE 18TH INTERNATIONAL CONFERENCE ON EVALUATION AND ASSESSMENT IN SOFTWARE ENGINEERING, 18., EASE '14, ., 2014. **Anais**... New York, NY, USA: Association for Computing Machinery, 2014, p. 10.

YU, J.; MAO, C. ; YE, X. A novel tree-based neural network for android code smells detection. In: IEEE 21ST INTERNATIONAL CONFERENCE ON SOFTWARE QUALITY, RELIABILITY AND SECURITY (QRS), 21., 2021. **Anais**... [S.l.]: Institute of Electrical and Electronics Engineers, 2021, p. 738–748.

ZAKERI-NASRABADI, M.; PARSA, S.; ESMAILI, E. ; PALOMBA, F. A systematic literature review on the code smells datasets and validation mechanisms. **ACM Comput. Surv.**, v. 55, n. 13s, 2023. Disponível em: <https://doi.org/10.1145/3596908>. Acesso em: 02/03/2024.

Documento assinado digitalmente
gov.br FABIO DO ROSARIO SANTOS
Data: 22/05/2024 17:37:26-0300
Verifique em https://validar.iti.gov.br

Capitão-Tenente Fábio do Rosario Santos (SC 23105)

Aluno

Documento assinado digitalmente
gov.br RICARDO CHOREN NOYA
Data: 13/06/2024 07:45:19-0300
Verifique em https://validar.iti.gov.br

Ricardo Choren Noya, D.Sc.

Orientador

Documento assinado digitalmente
gov.br GABRIELA MOUTINHO DE SOUZA DIAS
Data: 03/07/2024 12:39:07-0300
Verifique em https://validar.iti.gov.br

Major Gabriela Moutinho de Souza Dias, D.Sc.

Coordenador de Pós-graduação

Concordo com a presente Proposta de Dissertação e declaro que as necessidades para sua execução serão garantidas pela Seção.

IME, em 16 de Maio de 2024.

Documento assinado digitalmente
gov.br JULIO CESAR DUARTE
Data: 03/07/2024 12:44:40-0300
Verifique em https://validar.iti.gov.br

Cel Julio Cesar Duarte

CHEFE da SE/9