Classificação de Risco de Código Orientado a Objetos: Uma Abordagem Baseada em Machine Learning

Caio Henrique F. Monteiro[1], **Cassius Marcellus do C. Figueiredo**[1], **Thiago Silva de Souza**[1]

[1] Centro Universitário IBMEC do Rio de Janeiro (IBMECRJ)
Rio de Janeiro – RJ – Brasil

caiohenriquechfm292@gmail.com

Abstract. *This study proposes an approach based on* ***class imbalance techniques*** *to enhance software risk prediction, using object-oriented design and complexity metrics. The dataset was constructed from 15 Java projects hosted on GitHub, consolidating data at the class and file levels. Correlation analyses and the Random Forest algorithm were employed for risk modeling. The results indicate that risk is primarily driven by Structural Size (NOS) and Dynamic Coupling (RFC). The optimized model demonstrated superior performance,* ***addressing class imbalance*** *and achieving an accuracy of 72.84%, with a significant gain in sensitivity. The research reinforces the value of machine learning as a supporting tool for quality-oriented software engineering and preventive code refactoring.*

Resumo. *Este estudo propõe uma abordagem baseada em* ***técnicas de balanceamento de classes*** *para aprimorar a predição de risco em software, utilizando métricas de complexidade e design orientadas a objetos. O dataset foi construído a partir de 15 projetos Java hospedados no GitHub, consolidando dados nos níveis de classe e arquivo. Foram empregadas análises de correlação e o algoritmo Random Forest para modelagem de risco. Os resultados indicam que o risco é impulsionado principalmente pelo Tamanho Estrutural (NOS) e pelo Acoplamento Dinâmico (RFC). O modelo otimizado apresentou desempenho superior,* ***corrigindo o viés de classe*** *e alcançando acurácia de 72,84%, com ganho expressivo de sensibilidade. A pesquisa reforça o valor do aprendizado de máquina como ferramenta de apoio à engenharia de software orientada à qualidade e à refatoração preventiva de código.*

1. Introdução

A predição de risco em software é um elemento essencial para apoiar decisões de gestão de manutenção e melhorar a qualidade de sistemas orientados a objetos. Métricas de código, como as propostas no conjunto CK [3] — WMC, RFC, CBO, LCOM5 e DIT bem como métricas estruturais como LLOC, NM e NOS, são amplamente utilizadas para avaliar complexidade, acoplamento, coesão e tamanho. Estes fatores estão diretamente associados ao risco de intervenção, ao custo de manutenção e ao potencial de falhas no sistema.

Com o crescimento dos sistemas, essas métricas tendem a se correlacionar fortemente, gerando multicolinearidade e dificultando a interpretação direta dos valores. Além

disso, bases reais de projetos frequentemente apresentam forte desbalanceamento entre classes de risco, o que compromete a eficácia de modelos supervisionados tradicionais [2].

Neste trabalho, utilizamos o banco de dados público desenvolvido por Tóth, Gyimesi e Ferenc [4], que reúne informações extraídas de 15 projetos Java hospedados no GitHub, incluindo métricas estruturais nos níveis de classe e arquivo, bem como registros reais de defeitos. Durante a construção da variável-alvo baseada em *WMC Média*, identificou-se um desbalanceamento extremo entre as classes, tornando essa métrica inadequada para uso direto em modelos de classificação preditiva.

Para resolver esse problema, propomos um **reescalonamento estatístico** que alinha a métrica *WMC Média* à escala da Complexidade Ciclomática de McCabe (McCC), usada como referência de risco. Isso permitiu gerar uma variável-alvo mais equilibrada e adequada a modelos supervisionados.

Com essa nova variável, utilizamos análise de correlação e o algoritmo *Random Forest* para modelar o risco de design. Os resultados indicam ganhos expressivos na detecção de riscos intermediários e destacam métricas como NOS e RFC como principais preditores.

As próximas seções apresentam a estrutura dos dados, análises exploratórias, metodologia e resultados obtidos.

2. Coleta e Estrutura Inicial dos Dados

O presente estudo baseia-se em dados provenientes de um banco de dados público de *bugs*, originário da pesquisa intitulada *"A Public Bug Database of GitHub Projects and Its Application in Bug Prediction"*. Este *dataset* foi concebido para suprir a lacuna de bases de dados abertas, essenciais para a reprodutibilidade de estudos na área de predição de defeitos em software.

2.1. Origem e Composição do Dataset

O *dataset* original é resultado da análise de 15 projetos *Java* de código aberto hospedados no *GitHub*. A metodologia empregada utilizou ferramentas específicas para correlacionar defeitos conhecidos e corrigidos aos respectivos elementos de código-fonte (classes e arquivos), além de calcular um conjunto abrangente de métricas de produto.

- **Granularidade:** Os dados foram organizados em diferentes níveis de granularidade: **Nível de Classe (Class Level)** e **Nível de Arquivo (File Level)**.
- **Agregação:** A coleta foi realizada em intervalos de aproximadamente seis meses. No total, foram gerados 210 arquivos de banco de dados a partir de 105 versões de *release*.

3. Unificação e Estrutura Analítica

Para a criação de uma base analítica coesa voltada à construção dos modelos preditivos, a estrutura fragmentada do *dataset* original foi consolidada em múltiplas etapas de integração e refinamento, conforme descrito a seguir:

1. **Unificação por Granularidade:** Todos os arquivos referentes ao nível de classe foram unificados em um único *dataset* mestre. Da mesma forma, os arquivos referentes ao nível de arquivo foram integrados em um segundo *dataset* mestre.

2. **Criação do Dataset de Análise Refinado:** Após a unificação inicial, foi gerado um terceiro *dataset*, denominado **matriz_class**, a partir da base consolidada de classes. Este conjunto foi submetido a uma seleção criteriosa de métricas essenciais para a análise de risco, incluindo especificamente as métricas de Design Orientado a Objetos (OO) e de Complexidade: **LCOM5**, **CBO**, **DIT**, **RFC**, **WMC**, **LLOC**, **NOS** e **NM**.
   O *dataset* **matriz_class** serviu como fonte exclusiva das variáveis preditoras (*features*) e também para o cálculo da variável de risco **WMC_Media**, definida como a razão entre o **WMC (Weighted Methods per Class)** e o **NM (Number of Methods)**, ou seja:
$$WMC\_Media = \frac{WMC}{NM}$$

4. Análise das Matrizes de Correlação

As Matrizes de Correlação de Pearson foram empregadas para quantificar as relações lineares entre as variáveis preditoras (métricas de complexidade e design) nos níveis de **Classe** e **Arquivo**. Essa análise é essencial para identificar redundâncias (multicolinearidade) e validar a escolha de um modelo de aprendizado de máquina não linear, como o *Random Forest (RF)*.

Foram geradas duas matrizes distintas: **(i)** a matriz de correlação de classes (*matriz-correlacao-class.png*) e **(ii)** a matriz de correlação de arquivos (*matriz-correlacao-file.png*). Ambas apresentaram padrões semelhantes, com forte relação entre métricas de tamanho, complexidade e acoplamento, refletindo dependências estruturais típicas de sistemas orientados a objetos.

4.1. Correlação entre Métricas de Tamanho, Complexidade e Acoplamento

A análise das correlações entre as métricas estruturais revelou padrões consistentes com a literatura de Engenharia de Software Orientada a Objetos, evidenciando relações fortes entre tamanho, complexidade e acoplamento.

4.1.1. Correlações Fortes: Tamanho e Complexidade Ponderada

As correlações mais fortes (valores próximos de 1) ocorrem entre métricas que capturam o **tamanho** e a **complexidade agregada** da classe, confirmando padrões de interdependência típicos em sistemas Orientados a Objetos.

- **LLOC (Linhas Lógicas de Código) vs. WMC (Complexidade Ponderada por Classe)**
  Correlação: 0.94 (Muito Forte)
  Implicação: O **volume de código** (LLOC) é o principal fator que determina a complexidade total da classe (WMC). Este achado é crucial, pois classes grandes são inerentemente complexas, validando o uso de LLOC como um forte preditor de risco.
- **LLOC vs. NM (Número de Métodos)**
  Correlação: 0.96 (Quase Perfeita)
  Implicação: Confirma a alta dependência entre o tamanho físico da classe e o número de métodos que ela contém.

- **WMC vs. NM**
  Correlação: 0.91 (Muito Forte)
  Implicação: Como WMC é a soma das complexidades dos métodos, classes com alto NM inevitavelmente apresentam alto WMC.

4.1.2. Acoplamento e Risco Estrutural (RFC e CBO)

As métricas de acoplamento evidenciam a interconexão da classe, sendo essenciais para avaliar o risco de dependência e de propagação de mudanças.

- **RFC (Fator de Resposta) vs. NM (Número de Métodos) Correlação:** 0.86 (Muito Forte). **Implicação:** Classes com mais métodos (NM alto) tendem a se comunicar mais intensamente com outras partes do sistema (RFC alto). Isso aumenta a complexidade de interconexão e reforça o RFC como um **indicador robusto de risco de dependência**.
- **CBO (Acoplamento Entre Objetos) vs. WMC Correlação:** 0.69 (Forte). **Implicação:** Classes mais complexas (WMC alto) tendem a ser mais acopladas (CBO alto), um padrão esperado, pois a complexidade funcional exige mais dependências externas.

4.1.3. Coesão e Hierarquia (LCOM5 e DIT)

- **LCOM5 (Falta de Coesão) vs. DIT (Profundidade da Árvore de Herança) Correlação:** 0.10 (Baixa). **Implicação:** Embora estatisticamente significativa, o valor numérico é baixo, sugerindo que a **coesão interna** das classes ($LCOM5$) é amplamente **independente de sua posição hierárquica** (DIT).

4.1.4. Conclusão para Modelagem

A matriz de correlação demonstra a presença de **multicolinearidade severa** (por exemplo, entre LLOC, WMC, NM e RFC). Este achado metodológico valida a escolha do *Random Forest* (RF) para a modelagem preditiva, uma vez que, como modelo baseado em árvores, ele é **tolerante à multicolinearidade** e ainda fornece uma hierarquia de importância das variáveis robusta e interpretável.

4.2. Relações no Nível de Arquivo

A matriz de correlação no nível de arquivo apresentou padrões semelhantes aos observados em nível de classe, reforçando a consistência das relações entre tamanho, complexidade e acoplamento estrutural.

- **Relação entre McCC e LLOC:** Foi identificada uma correlação positiva expressiva entre a **Complexidade Ciclomática Média (McCC)** e o **LLOC**, o que indica que arquivos com maior volume de código tendem também a apresentar fluxos de controle mais complexos. Esse achado reforça que o aumento do tamanho do código está diretamente associado ao crescimento da complexidade lógica, impactando negativamente a manutenibilidade e o risco de falhas.

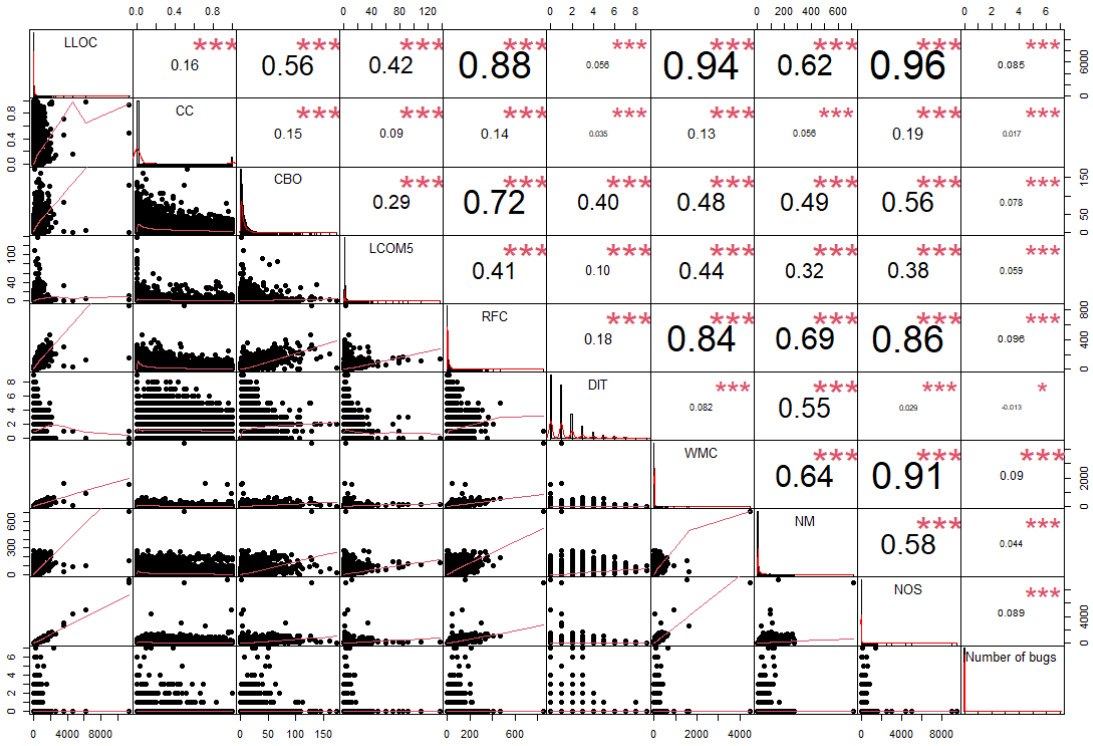


Figure 1. Matriz de correlação entre métricas no nível de Classe.

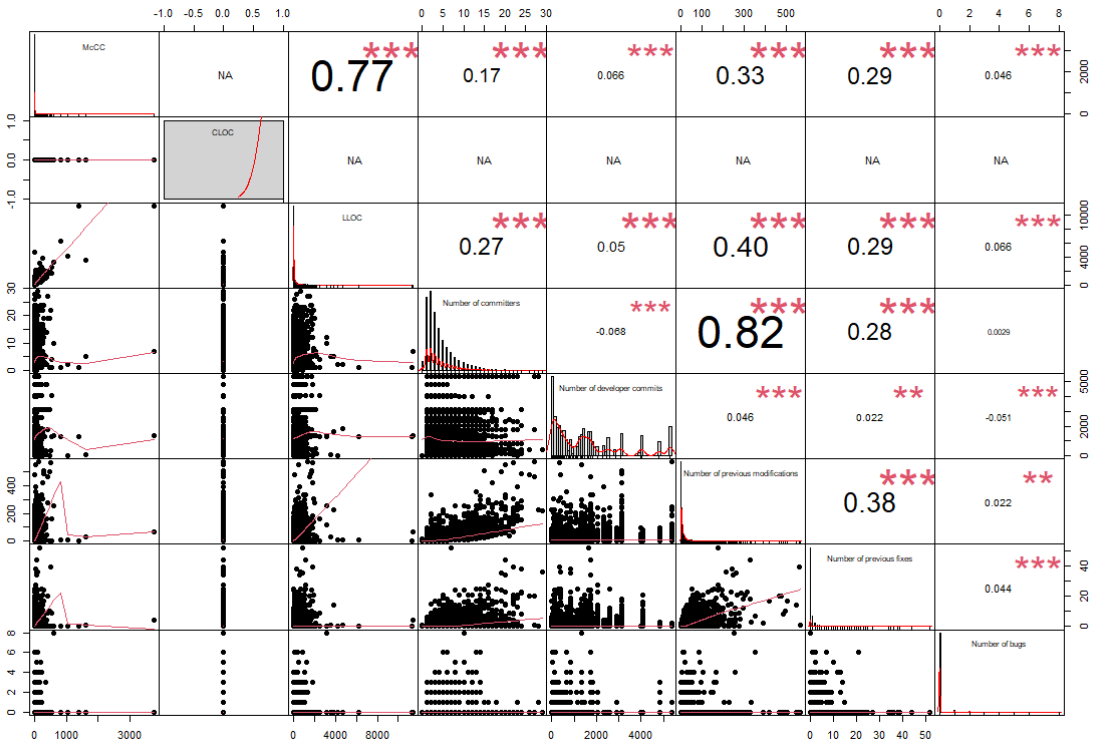


Figure 2. Matriz de correlação entre métricas no nível de Arquivo.

5. Metodologia de Análise e Solução para o Viés de Classe

5.1. Contextualização do Risco de Referência ($Risco\_McCC$)

Antes de qualquer etapa de engenharia de *features*, a **Complexidade Ciclomática de McCabe (McCC)** (tabela `dados_file`) foi adotada como **métrica de referência para a avaliação de risco** neste estudo. A classificação inicial `Risco_McCC` foi construída aplicando **limites fixos** (51, 21, 11), os quais representam as regras de negócio estabelecidas para definir as faixas de risco (*Baixo Risco*, *Risco Moderado*, *Alto Risco*, *Risco Muito Alto*) utilizadas nas análises subsequentes.

5.2. Engenharia da Variável Target Otimizada

A análise identificou o desafio central da modelagem: a categorização da métrica WMC/NM resultava em um desbalanceamento extremo, com aproximadamente 99% das instâncias classificadas como *Baixo Risco*[1]. Esse efeito ocorreu porque a escala numérica da variável WMC_Media era significativamente distinta da escala utilizada pela métrica de referência McCC, cujos limites de corte fixos (51, 21, 11) definem categorias de risco de forma consolidada na literatura[2]. Assim, a coluna original $Risco\_WMC\_Media$ tornou-se inadequada para utilização em algoritmos de *Machine Learning*, uma vez que o desbalanceamento extremo reduz drasticamente a capacidade de generalização dos classificadores supervisionados[3].

Para superar essa limitação, foi adotado um **balanceamento estatístico baseado no desvio-padrão**, cujo objetivo foi recalibrar a variável WMC_Media para uma escala compatível com a da métrica McCC. Primeiramente, calcularam-se a média (μ_{McCC}) e o desvio-padrão (σ_{McCC}) da Complexidade Ciclomática no nível de arquivo, que serviram como parâmetros de referência. Em seguida, foram obtidas as estatísticas correspondentes para WMC_Media no nível de classe (μ_{WMC} e σ_{WMC}).

Com base nessas estatísticas, aplicou-se a transformação:

$$WMC\_Media\_Escalada = (WMC\_Media - \mu_{WMC}) \cdot \left(\frac{\sigma_{McCC}}{\sigma_{WMC}}\right) + \mu_{McCC},$$

que ajusta a distribuição de WMC_Media para a mesma ordem de magnitude da McCC, preservando sua forma estatística, mas corrigindo a discrepância de escala. Essa recalibração permitiu a aplicação consistente dos limites fixos utilizados pela McCC.

A partir da variável transformada $WMC\_Media\_Escalada$, foram geradas as quatro categorias de risco (*Baixo*, *Moderado*, *Alto* e *Muito Alto*), originando a variável otimizada $Risco\_WMC\_Media\_TESTE$, apresentada na Tabela 1. Diferentemente da variável original, essa nova versão apresentou distribuição balanceada entre as classes, possibilitando sua utilização eficaz em modelos supervisionados e permitindo a detecção de riscos intermediários que eram negligenciados na estrutura inicial.

As novas colunas permitiram a criação de uma variável *target* coerente com o

[1] Observação obtida durante a inspeção da distribuição de classes.

[2] A métrica McCC foi utilizada como padrão de risco para calibração.

[3] Modelos treinados com forte desbalanceamento tendem a ignorar classes minoritárias.

Table 1. Variáveis Otimizadas Criadas por balanceamento

Coluna Gerada	Por que foi gerada	O que ela é
`WMC_Media_Escalada`	Para alinhar a escala do WMC_Media à escala da métrica de referência (McCC).	Valor numérico do WMC_Media após ser "calibrado" utilizando a média e o desvio padrão do McCC.
`Risco_WMC_Media_TESTE`	Para criar uma variável *target* balanceada e utilizável no modelo Random Forest.	Resultado da aplicação dos limites fixos (51, 21, 11, 1) na coluna $WMC\_Media\_Escalada$.

risco de referência, corrigindo o viés de classe e tornando o conjunto de dados utilizável em análises supervisionadas de aprendizado de máquina.

6. Resultados e Discussão da Classificação de Risco

6.1. Desempenho do Modelo Otimizado (RF 1)

O modelo *Random Forest* (RF 1), treinado com a variável balanceada $Risco\_WMC\_Media\_TESTE$ e pesos de custo (`classwt`)[4], apresentou os resultados detalhados na Tabela 2.

Table 2. Matriz de Confusão do Modelo *Random Forest* (RF 1) no Conjunto de Teste

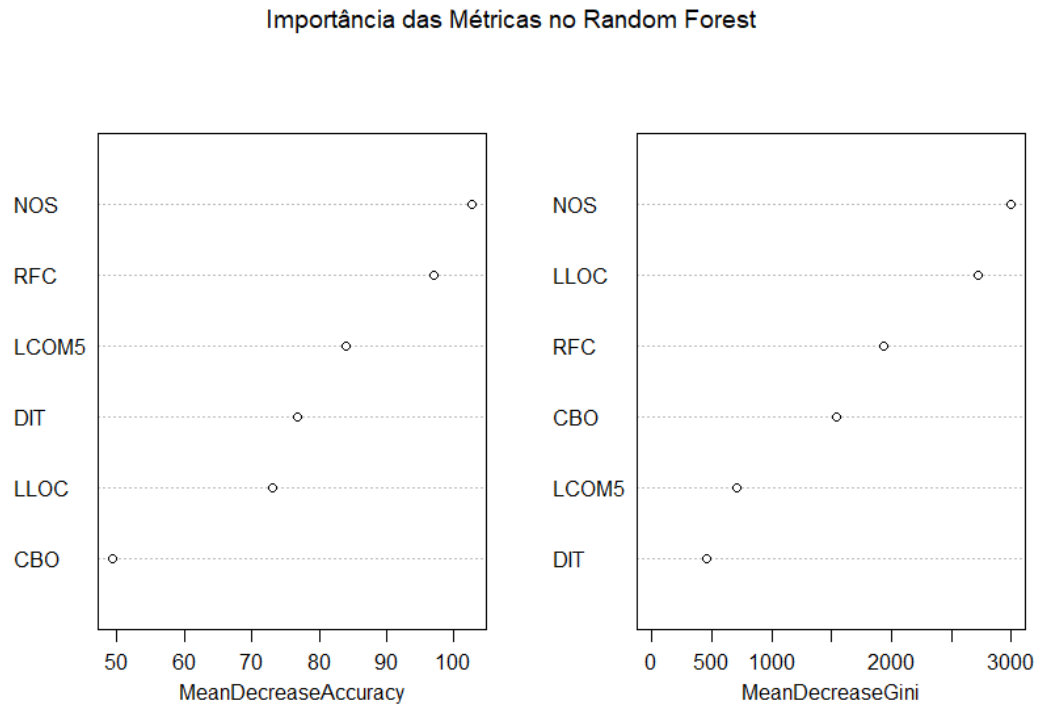
Prediction ↓ / Reference →	Baixo Risco	Risco Moderado	Alto Risco	Risco Muito Alto
Baixo Risco	4365	370	268	131
Risco Moderado	353	357	105	31
Alto Risco	171	122	259	91
Risco Muito Alto	139	65	126	309

O modelo obteve uma acurácia geral de **72,84%** e um coeficiente Kappa moderado de **0,434**[5].

Esses resultados demonstram que o RF 1 alcançou um equilíbrio adequado entre *precisão* e *generalização*, apresentando desempenho satisfatório nas quatro categorias de risco.

[4]Os pesos foram aplicados para corrigir o desbalanceamento residual entre as classes.

[5]O valor de Kappa indica concordância substancial entre as previsões do modelo e os valores reais de referência.



6.2. Superioridade Metodológica e Ganhos na Detecção

A superioridade metodológica do modelo *Random Forest* (RF 1) reside na resolução do problema de viés obtida por meio do **Reescalonamento Estatístico**[6]. O resultado mais significativo dessa abordagem é o expressivo ganho de **Sensibilidade** (*Recall*), apresentado na Tabela 3.

Table 3. Comparativo de Desempenho entre Modelos (Otimizado vs. Enviesado)

Ganho	Evidência Quantitativa	Implicação
Ganhos na Sensibilidade	A sensibilidade (*Recall*) da classe **Risco Moderado** aumentou de 0.0087 (no modelo enviesado) para aproximadamente $\mathbf{0.39}$, representando um crescimento de cerca de $45\times$ na capacidade de detecção[7].	O modelo foi capacitado a **enxergar e classificar riscos** que anteriormente eram ignorados, aumentando a utilidade prática na gestão preventiva de defeitos.
Robustez da Acurácia	A acurácia geral foi de 72.84%, enquanto o coeficiente Kappa atingiu 0.434[8].	O modelo é mais **robusto e confiável**, pois a acurácia resulta de um bom desempenho em todas as classes, e não apenas na classe majoritária.

O modelo enviesado (**Risco_MCCC**), embora tenha alcançado uma acurácia de **84,69%**[9], apresentou uma sensibilidade de apenas **0,0087** para a classe **Risco Moderado**,

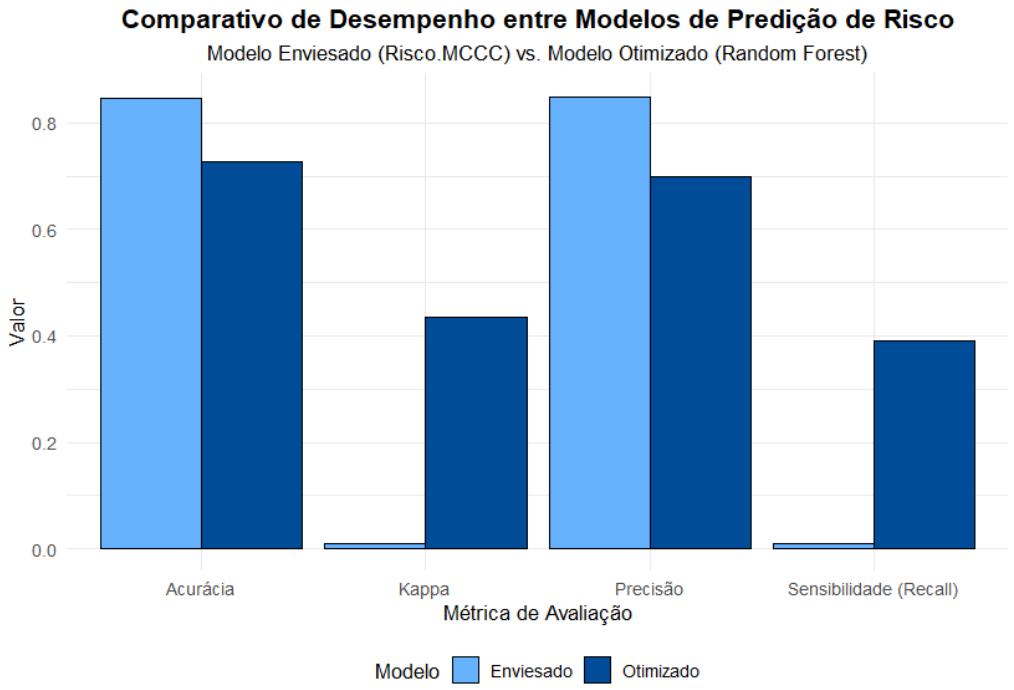
[6]Procedimento que ajusta a escala de variáveis numéricas para corrigir discrepâncias de magnitude e desbalanceamento entre as classes.

[9]O alto valor de acurácia é ilusório, refletindo o domínio da classe majoritária.

o que o torna estatisticamente ineficaz para fins de *gestão de riscos*[10].

Esses resultados reforçam que a reengenharia da variável alvo foi decisiva para obter um modelo mais **interpretável**, **balanceado** e **adequado para uso preditivo em contextos reais**.

6.3. Comparativo de Desempenho entre Modelos



6.4. Fatores de Risco (Importância das Variáveis)

A análise da Importância das Variáveis no modelo Random Forest (RF-1) identificou os atributos com maior contribuição para a capacidade preditiva do classificador, conforme apresentado na Tabela 4. As métricas **NOS** e **RFC** obtiveram os maiores valores de *MeanDecreaseAccuracy* e *MeanDecreaseGini*, indicando que medidas relacionadas ao *tamanho estrutural* do código (NOS) e ao *potencial de resposta* de uma classe (RFC) foram particularmente informativas na discriminação de módulos com maior propensão ao risco no conjunto de dados analisado.

É importante destacar que os resultados representam *importância preditiva* no contexto do modelo e **não implicam causalidade direta** entre essas métricas e a ocorrência de defeitos. Ainda assim, essa hierarquia sugere que componentes caracterizados por grande volume de código aliado a maior complexidade comportamental podem demandar inspeções mais cuidadosas por parte de equipes de engenharia de software, auxiliando decisões de priorização em atividades de manutenção e garantia de qualidade.

[10]Modelos com sensibilidade próxima de zero em classes relevantes são considerados inviáveis para aplicações práticas.

Table 4. Importância das Variáveis (Métricas) no Modelo Random Forest Otimizado

Métrica	MeanDecreaseAccuracy	MeanDecreaseGini
NOS	102.72	2999.77
RFC	97.02	1932.51
LCOM5	84.06	709.23
LLOC	73.13	2723.73
DIT	76.87	456.40
CBO	49.46	1535.07

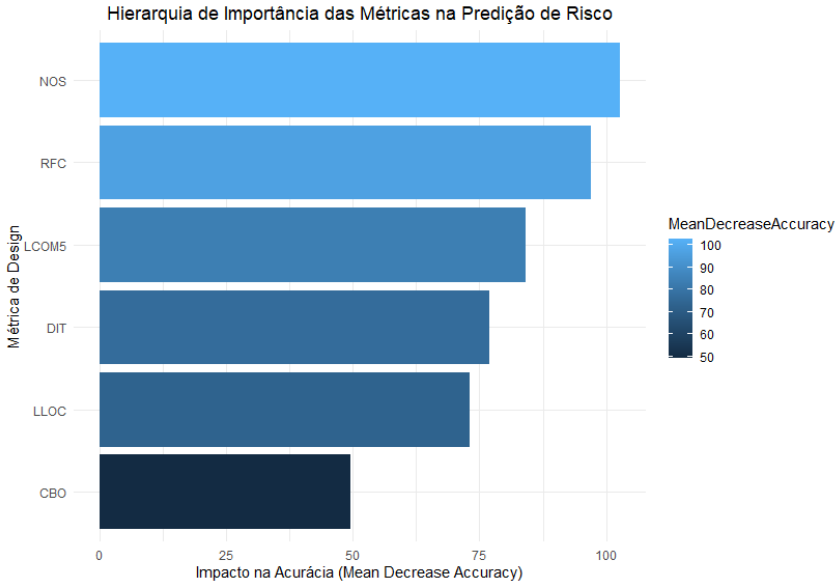


Figure 3. Hierarquia das Métricas segundo a Importância no Modelo Random Forest

7. Conclusão

Esta pesquisa evidenciou a eficácia da abordagem metodológica baseada no **balanceamento de classes** para a construção de uma variável *target* mais robusta e balanceada, contribuindo para a melhoria da capacidade preditiva dos modelos de aprendizado de máquina aplicados à predição de defeitos em software.

Essa abordagem de balanceamento foi determinante para o ganho expressivo de **Sensibilidade (*Recall*)** nas classes de risco moderado e alto. Ao eliminar o viés da classe majoritária, o modelo passou a reconhecer padrões de risco intermediário antes negligenciados, tornando-o estatisticamente viável e prático para a priorização de refatoração de código.

Os resultados obtidos com o modelo *Random Forest* indicam uma performance significativamente superior na detecção de risco crítico, evidenciando a adequação dessa técnica para cenários com variáveis de natureza mista e distribuição assimétrica.

A análise das variáveis de importância do modelo revelou que o risco de **Complexidade de Design/Qualidade** (WMC/NM) é fortemente influenciado pelos atribu-

tos **NOS** e **RFC**, os quais apresentaram maior impacto na classificação dos módulos de software com maior propensão a falhas.

Esses achados fornecem **insights práticos** relevantes para o gerenciamento da qualidade de software, permitindo que equipes de desenvolvimento priorizem componentes críticos e adotem estratégias de mitigação de risco mais direcionadas. O estudo reforça, portanto, a utilidade de técnicas estatísticas e de *machine learning* como ferramentas complementares na engenharia de software orientada à qualidade.

Referências

[1] Santos, L. V., Martinez, F., Ribeiro, J. R. *Extração da métrica WMC a partir de código Java.* In: Encontro Anual de Tecnologia da Informação – EATI, 2015, Frederico Westphalen. Disponível em: `http://2015.eati.info/assets/anais/Longos/L28.pdf`.

[2] Ghotra, B., McIntosh, S., Hassan, A. E. *Revisiting the Impact of Classification Techniques on the Performance of Defect Prediction Models.* In: *International Conference on Software Engineering (ICSE)*, 2015, Florence. IEEE. Disponível em: `https://dl.acm.org/doi/10.1109/ICSE.2015.47`.

[3] Chidamber, S. R., Kemerer, C. F. *A Metrics Suite for Object-Oriented Design.* Technical Report, M.I.T. Sloan School of Management, 1993. Disponível em: `https://www.eso.org/~tcsmgr/oowg-forum/TechMeetings/Articles/OOMetrics.pdf`.

[4] Tóth, Z., Gyimesi, P., Ferenc, R. *A Public Bug Database of GitHub Projects and Its Application in Bug Prediction.* In: Lecture Notes in Computer Science (LNCS), *International Conference on Computational Science and Its Applications (ICCSA 2016)*. Springer, Cham, 2016. Disponível em: `https://www.researchgate.net/publication/304664263`.

[5] Hastie, T., Tibshirani, R., Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Series in Statistics, 2nd ed., Springer, New York, 2009. Disponível em: `https://hastie.su.domains/ElemStatLearn/`.

[6] Fenton, N. E., Pfleeger, S. L. *Software Metrics: A Rigorous and Practical Approach.* 2ª ed., PWS Publishing Company, Boston, 1997.