1.03.99 - Ciência da Computação

# ANÁLISE DE PREDIÇÃO DE TENDÊNCIA E SEVERIDADE DE FALHAS EM MÓDULOS DE SOFTWARE UTILIZANDO PROGRAMAÇÃO GENÉTICA

Vitor O. Franco[1], Arielson A. de Souza[1], Davi J. de Souza[1], Marco A. P. Araújo[1]
1. Estudante do curso Bacharelado em Sistemas de informação no Instituto Federal do Sudeste de Minas Gerais - Campus Juiz de Fora
2. Professor do Instituto Federal do Sudeste de Minas Gerais - Campus Juiz de Fora - Departamento de Informática/Orientador

## Resumo
Com a crescente demanda por softwares maiores, mais seguros e mais complexos a qualidade de software nunca se fez tão presente na vida de um programador. Um dos campos mais importantes da qualidade de software são os testes, que buscam encontrar falhas em módulos de um software. Ser capaz de identificar previamente a severidade de falhas de um módulo de software, permite melhor gerenciamento dos recursos de um projeto. Diversas técnicas têm sido utilizadas para automatizar esta tarefa, uma vez que tal procedimento pode ser muito custoso, porém algumas técnicas ainda não foram exploradas em todo seu potencial, como o caso da programação genética (PG). Esta pesquisa busca então demonstrar a eficácia da PG na tarefa predição de tendencia e severidade de falhas em módulos de software. Os resultados mostram que o modelo de PG possui eficiência afim dos modelos estado-da-arte, obtendo uma precisão de 88,5417%, uma sensibilidade de 86,7357% e uma acurácia de 77,9817%.

**Palavras-chave:** engenharia de software; machine learning; qualidade de software

**Apoio financeiro:** IF Sudeste MG.

**Trabalho selecionado para a JNIC:** IF Sudeste MG.

## Introdução
A qualidade de software é uma área extremamente importante da engenharia de software que envolve desde o início de um projeto de software até a manutenção do mesmo. Com a crescente demanda por softwares maiores, mais confiáveis e mais seguros a qualidade de software nunca se fez tão presente na vida de um programador. Tais demandas por sistemas seguros, confiáveis e robustos acabam inevitavelmente aumentando a complexidade do sistema. A alta complexidade de softwares é um dos principais fatores causadores de falhas, erros e atrasos em projetos de software [Basili e Barry 1984]. As fases de testes são algumas das fases mais críticas de um projeto, pois são nestas fases que os erros e inconsistências são encontrados e corrigidos, além de serem algumas das fases mais custosas de um projeto de software, podendo chegar a custar 50% do tempo e orçamento do projeto [Isong e Obeten 2013]. Assim a utilização de uma estratégia eficiente, eficaz e inteligente para as fases de testes, pode ajudar bastante a minimizar os custos dessas etapas, uma das melhores resoluções para este problema seria conseguir automatizar o máximo possível do processo de identificação de severidade de falhas em módulos de software. A não existência de técnicas universais de identificação de falhas [Gondra 2008] e torna a utilização de métodos especializados em identificação de padrões, como as técnicas de Machine Learnig (ML), extremamente promissora. Diversos modelos de ML juntamente com diversas metodologias foram aplicados ao tema, porém devido a rápida evolução dos resultados e popularidade das técnicas utilizadas, alguns modelos de ML acabam sendo pouco aplicados. Como é o caso da programação genética (PG), que é justamente uma técnica de ML que não exige conhecimento prévio de funções otimizadoras. Dessa forma utilização de algoritmos evolutivos, como a programação genética, possuem um potencial pouco explorado. O objetivo do trabalho então é buscar esclarecer e determinar se a programação genética pode ser aplicada para classificar em classes de severidades módulos de software, utilizando métricas de software como base de treinamento, e com isso produzir resultados afins de técnicas de estado-da-arte da literatura.

## Metodologia
Existem diversas formas de se implementar uma técnica de ML, tanto no âmbito da implementação mais baixo nível quanto nas de mais alto nível. E interpostos à essas duas categorias existem os ambientes de programação, como a linguagem Python, que proporcionam auxilio na construção de modelos de ML, através de bibliotecas e estruturas já pré-implementadas, contudo devido à alta capacidade de configuração e a possibilidade de combinar diversas técnicas diferentes na mesma implementação, não comprometem a liberdade e adaptabilidade de modelos a serem implementados. Dessa forma, utilizando a linguagem Python, neste trabalho foram construídos dois modelos preditores de tendência e severidade de falhas em módulos de software, o primeiro seguindo métodos de estado da arte da

problemática, especificamente o modelo de Random Forest (RF) [Isong 2013], e o segundo utilizando técnicas de PG. Ambos foram implementados na linguagem Python, com auxílio das bibliotecas GPLearn [Stephens 2015] e ScikitLearn [Pedregosa 2011]. Os modelos foram treinados com a base de dados PROMISSE, que dispunha de diversos módulos de software acompanhados de suas métricas de software e classificação de severidade de falha. Para realizar a análise de eficiência dos resultados produzidos foram utilizados os parâmetros: Acurácia, Precisão, Sensibilidade e F-Measure. Por definição os parâmetros acurácia, precisão e sensibilidade são dados em porcentagem e, logicamente, quanto mais próximos de 100% melhores serão os resultados de predição de classificação do método utilizado. Contudo, dificilmente um método de classificação resulta em proporções sem erros para acurácia, precisão e sensibilidade simultaneamente, portanto identificar quais parâmetros extraem uma medição mais efetiva e descritiva da predição é de extrema importância. Uma alta precisão identifica que a predição teve uma taxa alta de acerto ao classificar cada módulo de software de acordo com sua severidade de falha, uma alta sensibilidade identifica que a predição teve uma taxa alta de acerto em classificar módulos de acordo com sua severidade de falha e uma alta acurácia identifica uma alta taxa de acerto classificando cada módulo. A F-Measure, por ser a média harmônica entre a precisão e sensibilidade, também é dada em porcentagem, sua utilização é mais útil em comparações de modelos, uma vez que facilita a comparação entre modelos em que se obteve resultados muito diferentes para a precisão e a sensibilidade.

## Resultados e Discussão

Esta pesquisa buscou demonstrar a eficiência da PG em classificar módulos de software de acordo com sua severidade de falha, através de métricas de software, para isso o modelo de PG foi comparado com o modelo de RF, modelo da literatura que apresenta os melhores resultados para esta problemática. Após a construção dos modelos foram obtidos os resultados: O método utilizando PG possuindo uma precisão de 88,5417%, uma sensibilidade de 86,7357% e uma acurácia de 77,9817%. O método utilizando o Randon Forest (RF) uma precisão de 94,1748%, uma sensibilidade de 94,1748% e uma acurácia de 88,9908%. Para afirmar que os resultados possuem diferença estatisticamente significantes é necessário a utilização de testes de hipótese. Para tal será utilizado um nível de significância ($\alpha$) igual a 5%. Para todos os testes de hipóteses foi utilizado o valor dado pelo Teste exato de Fisher. A diferença de acurácias entre os modelos apresentou valor do p-value menor do que 0,05 (0,044=p; p<0,05) então há significância estatística, concluindo que o resultado do método Random Forest teve estatisticamente uma acurácia pouco superior ao método utilizando programação genética. A diferença entre as precisões dos modelos apresenta valor do p-value para o teste exato de Fisher maior do que 0,05 (0,206=p; p>0,05) então não há significância estatística, concluindo que o resultado do método programação genética teve estatisticamente uma precisão semelhante ao método Random Forest. A diferença entre as sensibilidades dos modelos apresenta valor do p-value maior do que 0,05 (0,092=p; p>0,05) então não há significância estatística, concluindo que o resultado do método programação genética teve estatisticamente uma sensibilidade semelhante ao método Random Forest. A F-Measure de uma classificação é definida pela média harmônica entre a precisão e a sensibilidade, dessa forma a diferença entre as F-measures entre os modelos apresenta valor do p-value maior do que 0,05 (0,073=p; p>0,05) então não há significância estatística, concluindo que o resultado do método programação genética teve estatisticamente um F-Measure semelhante ao método Random Forest

## Conclusões

Durante a construção deste trabalho, foi possível constatar a ampla variedade de empregabilidade de técnicas de ML em identificação de módulos de software defeituosos de acordo com sua severidade. Esta é uma área que expande e evolui a passos largos somando isso ao impacto positivo que os resultados podem trazer à projetos de software em geral, evidenciam que é uma área com muito futuro. A base de dados PROMISE é uma base extremamente utilizada na literatura, e é disponibilizada abertamente, o que a torna uma excelente base para a reprodutibilidade de publicações. Ademais é uma base com uma quantidade razoável de total de amostras, além de possuir um tratamento prévio, evitando dados faltantes e dados irrelevantes. Apesar disto a base apresenta uma questão substancialmente prejudicial, que é apenas reunir dados de módulos de software trabalhados nas linguagens JAVA e C++. Uma maior diversidade de linguagens traria uma maior confiança para extrapolações para casos mais diversos além de possivelmente diminuir o impacto que a arquitetura de uma linguagem tem sobre as métricas produzidas durante o desenvolvimento. Neste trabalho foi proposto uma implementação de algoritmos evolutivos, focando na PG, para predizer tendência e severidade de falha em módulos de software, utilizando métricas de software como base de treino. Diversos trabalhos da literatura apontam para a deficiência de trabalhos envolvendo algoritmos evolutivos, e neste trabalho não somente um modelo

envolvendo algoritmos evolutivos foi implementado como seus resultados se mostraram afins de técnicas de estado-da-arte da literatura, mais especificamente o modelo Random Forest. Os resultados encontrados nesta pesquisa apontaram para precisão, sensibilidade e F-Measure semelhante para ambos os modelos, apesar de uma acurácia mais evoluída para o método de Random Forest. Mas como discutido, comparando os modelos pela F-measure a semelhança dos resultados é evidenciada. Para trabalhos futuros a coleta e criação de uma base de dados maior e mais diversa, envolvendo linguagens diferentes, desenvolvedores diferentes, a utilização de métricas diferentes e de fontes diferentes como open-source e códigos proprietário se mostram como o melhor curso de ação, essa coleta não se trata de uma simples tarefa, mas certamente deverá trazer melhores resultados para a predição de módulos de software defeituosos. Além disso a grande maioria das publicações tratam a tendência a falha como uma tarefa de classificação básica entre “o módulo possui falha” e “o módulo não possui falha”, assim a evolução para um problema de classificação um pouco mais complexo, considerando a severidade e o número de erros pode trazer resultados interessantes para a discussão.

## Referências bibliográficas


Araújo, M. A., et al. "**Métodos estatísticos aplicados em engenharia de software experimental**." XXI SBBD-XX SBES (2006).

Askari, Mohamad Mahdi, and Vahid Khatibi Bardsiri. "**Software defect prediction using a high-performance neural network**." International Journal of Software Engineering and Its Applications 8.12 (2014): 177-188.

Banzhaf, Wolfgang, et al. **Genetic programming: an introduction**. Vol. 1. San Francisco: Morgan Kaufmann Publishers, 1998.

Basili, Victor R., and Barry T. Perricone. "**Software errors and complexity: an empirical investigation**." Communications of the ACM 27.1 (1984): 42-52.

Breiman, Leo. "**Random forests**." Machine learning 45.1 (2001): 5-32.

Charniak, Eugene. **Introduction to artificial intelligence**. Pearson Education India, 1985.

Chidamber, Shyam R., and Chris F. Kemerer. "**A metrics suite for object-oriented design**." IEEE Transactions on software engineering 20.6 (1994): 476-493.

De Carvalho, Andre B., Aurora Pozo, and Silvia Regina Vergilio. "**A symbolic fault-prediction model based on multiobjective particle swarm optimization**." Journal of Systems and Software 83.5 (2010): 868-882.

Fenton, Norman E., and Martin Neil. "**Software metrics: roadmap**." Proceedings of the Conference on the Future of Software Engineering. 2000.

Gondra, Iker. "**Applying machine learning to software fault-proneness prediction**." Journal of Systems and Software 81.2 (2008): 186-195.

Guo, Lan, et al. "**Robust prediction of fault-proneness by random forests**." 15th international symposium on software reliability engineering. IEEE, 2004.

Harrison, Rachel, Steve J. Counsell, and Reuben V. Nithi. "**An evaluation of the MOOD set of object-oriented software metrics**." IEEE Transactions on Software Engineering 24.6 (1998): 491-496.

Isong, Bassey, and Ekabua Obeten. "**A systematic review of the empirical validation of object-oriented metrics towards fault-proneness prediction**." International Journal of Software Engineering and Knowledge Engineering 23.10 (2013): 1513-1540.

Kaur, Arvinder, and Ruchika Malhotra. "**Application of random forest in predicting fault-prone classes**." 2008 International Conference on Advanced Computer Theory and Engineering. IEEE, 2008.

Koza, John R. "Non-linear genetic algorithms for solving problems." U.S. Patent No. 4,935,877. 19 Jun. 1990.

Malhotra, Ruchika, and Ankita Jain. "**Fault prediction using statistical and machine learning methods for improving software quality**." Journal of Information Processing Systems 8.2 (2012): 241-262.

Murphy, Kevin P. **Machine learning: a probabilistic perspective**. MIT press, 2012.

Pedregosa, Fabian, et al. "**Scikit-learn: Machine learning in Python**." the Journal of machine Learning research 12 (2011): 2825-2830.

Promise Software Engineering Repository Public Datasets (2013). <http:// promise.site.uottawa.ca/SERepository/datasets/cm1.arff>

Shatnawi, Raed. "**Empirical study of fault prediction for open-source systems using the Chidamber and Kemerer metrics**." IET software 8.3 (2013): 113-119.

Singh, Ajmer, Rajesh Bhatia, and Anita Singhrova. "**Taxonomy of machine learning algorithms in software fault prediction using object-oriented metrics**." Procedia computer science 132 (2018): 993-1001.

Stephens, T. "**Gplearn Model, Genetic Programming**." (2015).