Análise de Técnicas de Machine Learning para Detecção de Vulnerabilidades em Contratos Inteligentes

Tiago Rios da Rocha[1,2], **Tobias de Abreu Kuse**[1], **Mariana Recamonde Mendoza**[1]

[1] Universidade Federal do Rio Grande do Sul (UFRGS)
[2] Instituto Federal do Rio Grande do Sul (IFRS)

{trrocha,takuse,mrmendoza}@inf.ufrgs.br

Abstract. *This study presents a comparative analysis of machine learning algorithms applied to the detection of vulnerabilities in smart contracts. Among the tested models, LightGBM achieved the best performance in identifying the vulnerable class.*

Resumo. *Este trabalho apresenta uma análise comparativa de algoritmos de aprendizado de máquina aplicados à detecção de vulnerabilidades em contratos inteligentes. Entre os modelos testados, o LightGBM obteve o melhor desempenho na identificação da classe vulnerável.*

1. Introdução

Contratos inteligentes (ou *"smart contracts"*) são peças importantes em aplicações descentralizadas na plataforma Ethereum. Esses contratos são programas autoexecutáveis que armazenam regras contratuais diretamente em código e, uma vez implantados na *blockchain*, operam de forma autônoma, transparente e imutável. A *blockchain*, por sua vez, é um livro-razão digital, descentralizado e distribuído, que registra transações de forma cronológica e imutável. O Ethereum permite a criação e execução desses contratos inteligentes através da *Ethereum Virtual Machine* (EVM). Essa imutabilidade, embora ofereça garantias de integridade, também representa um desafio em termos de segurança: qualquer vulnerabilidade inserida no momento da codificação torna-se permanente e pode ser explorada indefinidamente [Atzei et al. 2017].

O uso de *Machine Learning* (ML) permite extrair padrões complexos a partir de dados. Apesar do potencial, seu uso enfrenta desafios consideráveis. Um deles é a escassez de *datasets* rotulados e representativos, o que dificulta o treinamento de modelos generalizáveis. Outro ponto é a heterogeneidade estrutural dos contratos inteligentes — compostos por bytecode, árvores de sintaxe abstrata (AST), e componentes textuais — o que exige estratégias cuidadosas de extração e seleção de atributos [Durieux et al. 2020].

O presente trabalho investiga o uso de técnicas supervisionadas de ML para classificação de contratos inteligentes quanto à presença de vulnerabilidades. Para isso, foi utilizado o *dataset* proposto por [HajiHosseinKhani et al.], que empregou algoritmos genéticos para geração de subconjuntos representativos de contratos. O conjunto de dados contém contratos rotulados por meio de múltiplos scanners de segurança, enriquecido com atributos sintáticos, semânticos e estatísticos derivados de diferentes níveis de representação do código.

Neste trabalho, foram avaliados modelos com métodos baseados em distância (*K-Nearest Neighbors - KNN*), métodos baseados em árvores (Árvore de Decisão, *Random*

Forest), métodos probabilísticos (*Naive Bayes*), algoritmos de *boosting* (AdaBoost, XGBoost, LightGBM, CatBoost) e Máquina de Vetores de Suporte (*Support Vector Machine* - SVM), utilizando técnicas de padronização/normalização, redução de dimensionalidade e balanceamento de classes.

2. Trabalho Realizado

A metodologia foi estruturada em três etapas principais, abordadas adiante: primeiramente, uma análise exploratória e um pré-processamento do conjunto de dados; seguido por uma condução de experimentos para treinar e avaliar um conjunto diversificado de algoritmos de classificação; e, por fim, a seleção do modelo com melhor desempenho na detecção de vulnerabilidades.

2.1. Análise Exploratória dos Dados e Pré-processamento

O dataset de [HajiHosseinKhani et al.] foi gerado usando algoritmos genéticos para selecionar exemplos representativos e rotulado por múltiplos scanners de segurança. Ele contém uma variedade de atributos extraídos de diferentes níveis de análise de código.

Com um total de aproximadamente 36.670 contratos, o *dataset* abrange uma ampla variedade de casos, o que favorece a robustez no treinamento e na avaliação dos modelos supervisionados. Cada contrato está rotulado com uma classe binária: 0 para contratos seguros e 1 para contratos vulneráveis.

Existe no *dataset*, um desbalanceamento entre as classes 'Seguro' e 'Vulnerável', com 26% dos contratos classificados como 'Vulneráveis'. Este desequilíbrio não é severo (9.756 de um total de 36.671), de toda forma, foram aplicados métodos para mitigar os efeitos do desbalanceamento a fim de evitar um classificador enviesado em predizer somente a classe majoritária.

Outro aspecto relevante é a quantidade de atributos no conjunto de dados. São 265 variáveis, todas numéricas. Esse número elevado traz alguns desafios para a modelagem do problema: presença de atributos irrelevantes ou redundantes (colinearidade), risco de sobreajuste, aumento do tempo de treinamento, dificuldade de interpretação do modelo. Uma análise mais detalhada revelou que o conjunto de dados é altamente esparso, sendo de aproximadamente 70% das colunas possuem pelo menos metade de seus valores sendo zero. Adicionalmente, foi observado que o conjunto de dados está completo, isto é, não possui valores faltantes, não sendo necessário nenhum procedimento nesse sentido.

2.2. Pré-processamento

Com base na análise exploratória, o pré-processamento dos dados foi estruturado em um pipeline de várias etapas para otimizar a performance dos modelos. Primeiramente, para reduzir a dimensionalidade e a esparsidade, foram aplicados o agrupamento semântico de *opcodes* e caracteres do *bytecode*, a eliminação de variáveis com baixa variância e a remoção de atributos altamente correlacionados com um limiar de 0.8. Em seguida, foi realizada a seleção de features mais importantes com base no índice de Gini de uma Árvore de Decisão.

Para o tratamento de outliers em variáveis com distribuição assimétrica, utilizou-se a transformação logarítmica, e para lidar com a alta esparsidade, foram criadas *flags*

binárias que indicam a presença de valores não nulos. O balanceamento de classes foi tratado com o uso de parâmetros intrínsecos aos modelos, como *class_weight='balanced'* (para Random Forest e SVC) e *scale_pos_weight* ou *is_unbalance* (para os modelos de boosting), aplicados exclusivamente nos conjuntos de treinamento e validação para evitar o viés do classificador e o vazamento de dados.

Após estas etapas, os dados estavam adequadamente preparados para o treinamento dos modelos de aprendizado supervisionado. A combinação de seleção de atributos, normalização e balanceamento permitiu uma representação mais concisa, padronizada e informativa dos contratos inteligentes.

3. Experimentos, validação e resultados

Os experimentos foram conduzidos com a divisão estratificada do conjunto de dados em 80% para treino e validação, e 20% reservados exclusivamente para a avaliação final do modelo escolhido. A parte de treino e validação foi utilizada nos pipelines dos algoritmos com validação cruzada estratificada de 5 *folds*.

Foram avaliados diversos algoritmos de classificação supervisionada, incluindo: *K-Nearest Neighbors*, Naive Bayes, regressão logística, árvore de decisão, SVC linear, SVC com *kernel* não linear, *Random Forest*, AdaBoost, XGBoost, LightGBM e CatBoost, conforme apresentado na Tabela 1.

Tabela 1. Resultados médios e desvios padrão das métricas (validação cruzada estratificada 5-folds)

Modelo	Acurácia (%)	Precisão (%)	Recall (%)	F1-score (%)
KNN	78.35 ± 0.10	62.49 ± 0.33	47.29 ± 1.33	$\mathbf{53.82 \pm 0.76}$
Naive Bayes	67.50 ± 0.19	40.38 ± 0.40	45.68 ± 1.49	$\mathbf{42.86 \pm 0.85}$
Árvore de Decisão	72.79 ± 0.98	49.29 ± 1.39	64.13 ± 0.74	$\mathbf{55.73 \pm 1.02}$
Regressão Logística	71.39 ± 0.41	47.16 ± 0.59	59.74 ± 1.61	$\mathbf{52.70 \pm 0.85}$
Linear SVC	71.36 ± 0.61	47.18 ± 0.83	60.62 ± 1.50	$\mathbf{53.05 \pm 0.86}$
SVC (RBF)	71.55 ± 0.27	47.47 ± 0.34	61.59 ± 2.03	$\mathbf{53.60 \pm 0.78}$
Random Forest	76.19 ± 0.55	54.85 ± 1.02	61.30 ± 1.16	$\mathbf{57.88 \pm 0.72}$
AdaBoost	72.58 ± 0.37	48.75 ± 0.63	53.19 ± 1.69	$\mathbf{50.87 \pm 1.05}$
XGBoost	74.01 ± 0.62	50.95 ± 0.83	71.20 ± 1.24	$\mathbf{59.39 \pm 0.74}$
LightGBM	77.30 ± 0.66	56.21 ± 1.12	67.92 ± 1.03	$\mathbf{61.51 \pm 0.86}$
CatBoost	75.29 ± 0.63	52.76 ± 0.91	71.19 ± 1.23	$\mathbf{60.60 \pm 0.75}$

As métricas consideradas nos experimentos foram *accuracy*, *precision*, *recall* e *F1-score*. O *F1-score* foi adotado como métrica principal, por representar um equilíbrio entre *precision* e *recall*, aspecto crucial nesta tarefa. O foco em *recall* se justifica pela importância de detectar corretamente os contratos vulneráveis, enquanto a *precision* também é relevante, dado que o próprio artigo que propôs o conjunto de dados destaca a dificuldade de reduzir falsos positivos nesse domínio.

Com base nos resultados obtidos na validação cruzada, o modelo selecionado para avaliação final foi o LightGBM, por apresentar o melhor desempenho em termos de *F1-score*, além de manter um bom equilíbrio entre *precision* e *recall*.

O pipeline utilizado para este modelo incluiu diversas etapas de pré-processamento e seleção de atributos. Inicialmente, foi realizada uma agregação dos códigos de operação em categorias específicas, com o objetivo de reduzir a dimensionalidade e consolidar a informação. Em seguida, foram removidas variáveis com baixa variância, utilizando um ponto de corte mínimo, e atributos altamente correlacionados foram eliminados com base em um limiar de correlação, preservando-se aquele com maior variância em cada par.

Na sequência, foi aplicada uma transformação logarítmica para atenuar o impacto de valores extremos em variáveis com distribuição assimétrica. Também foi criada uma flag para identificar variáveis esparsas, com alto percentual de zeros, permitindo ao modelo tratá-las de forma diferenciada.

Por fim, foi utilizado o classificador LightGBM. Os hiperparâmetros foram definidos com base em testes manuais preliminares. Também foi conduzido um procedimento automatizado de busca por melhores combinações, porém sem ganhos relevantes de desempenho em relação à configuração manual adotada.

A Tabela 2 apresenta os resultados obtidos no conjunto de teste final. O modelo manteve desempenho consistente, refletindo a robustez do processo de validação cruzada.

Tabela 2. Desempenho do modelo LightGBM no conjunto de teste

Métrica	Precisão	Recall	F1-score	Acurácia
Valor	0,56	0,71	0,62	0,78

4. Considerações Finais [1]

Foi realizada uma análise comparativa entre diferentes algoritmos de aprendizado de máquina aplicados à tarefa de detecção de vulnerabilidades em contratos inteligentes. Os experimentos mostraram que modelos baseados em árvores, especialmente o LightGBM, apresentaram melhor desempenho na identificação da classe vulnerável. A avaliação foi feita utilizando-se validação cruzada estratificada com 5 folds. O código-fonte encontra-se disponível no repositório público GitHub (https://github.com/tiagoriosrocha/projeto-ml).

Referências

Atzei, N., Bartoletti, M., and Cimoli, T. (2017). A survey of attacks on ethereum smart contracts (sok). *Intern. Conf. on Principles of Security and Trust*.

Durieux, T., Ferreira, J. F., Abreu, R., and Cruz, P. (2020). Empirical review of automated analysis tools on 47,587 ethereum smart contracts. In *Intern. Conf. on Software Engineering*.

HajiHosseinKhani, S., Lashkari, A. H., and Mizani Oskui, A. Unveiling vulnerable smart contracts: Toward profiling vulnerable smart contracts using genetic algorithm and generating benchmark dataset. *Blockchain: Research and Applications*.

[1]O presente trabalho foi realizado com apoio do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul.