DETECÇÃO INTELIGENTE DE ANOMALIAS EM REDES COM LLMS *OPEN SOURCE*: UM PROTÓTIPO DE CAPTURA POR PROTOCOLO

Ronaldo Crispim da Silva Gonçalves[1] - rcsg2@aluno.ifnmg.edu.br
Márcio Martins Ferreira Júnior[1] - mmfj@aluno.ifnmg.edu.br
Paulo Veloso Santos Junior[1] - paulo.junior@ifnmg.edu.br
Felipe Augusto Oliveira Mota[1] - felipe.mota@ifnmg.edu.br
João Pedro Santos Rodrigues[1] - joao.pedro@ifnmg.edu.br
[1]Instituto Federal do Norte de Minas Gerais, IFNMG - Januária, MG, Brazil

Abstract. *Este artigo descreve a arquitetura e a avaliação de um protótipo de sistema de captura de anomalias em redes por protocolo. O protótipo coleta dados por ICMP, TCP, HTTP, DNS e SNMP e combina heurísticas, modelos clássicos de aprendizado de máquina e classificação generativa em um ensemble ponderado. São apresentados objetivos, metodologia, fluxo de execução, métricas e resultados experimentais, além de uma análise estatística com média, desvio padrão e intervalo de confiança de 95% para os principais classificadores.*

Keywords: *Anomalias de rede, Aprendizado de máquina, Redes de computadores, Grandes modelos de linguagem*

1. INTRODUÇÃO

Redes de computadores modernas suportam aplicações cada vez mais heterogêneas e exigem elevados níveis de desempenho, disponibilidade e segurança. Nesse cenário, a identificação de anomalias, como congestionamentos, falhas de enlace e degradação de desempenho, é uma atividade essencial para manter a confiabilidade e a qualidade dos serviços oferecidos. De acordo com Kreutz et al. (2015); Kim & Feamster (2013); Buczak & Guven (2016), o aumento da complexidade das infraestruturas de rede torna o monitoramento operacional uma tarefa cada vez mais desafiadora, exigindo mecanismos capazes de analisar grandes volumes de dados de forma rápida e consistente.

As abordagens tradicionalmente empregadas para detecção de anomalias baseiam-se em regras pré-definidas, assinaturas ou modelos clássicos de aprendizado de máquina. Embora essas técnicas apresentem bons resultados em cenários específicos, sua capacidade de adaptação a comportamentos inéditos ou mudanças dinâmicas na rede permanece limitada, além de

frequentemente depender de conhecimento especializado para configuração, manutenção e interpretação dos resultados (Buczak & Guven, 2016).

Nos últimos anos, Grandes Modelos de Linguagem (*Large Language Models* – LLMs) passaram a demonstrar potencial para tarefas que envolvam interpretação contextual, raciocínio e suporte à tomada de decisão. Paralelamente, Redes Definidas por Software (*Software-Defined Networking* – SDN) consolidaram-se como uma arquitetura capaz de fornecer maior programabilidade, visibilidade e flexibilidade ao gerenciamento de redes. Apesar desse cenário favorável, ainda são limitados os trabalhos que investigam, de forma experimental, a integração entre LLMs e ambientes SDN para apoio à detecção e classificação de anomalias em redes (Bommasani et al., 2021; Zhao et al., 2023; Kreutz et al., 2015).

Diante desse contexto, este trabalho apresenta um protótipo de sistema para captura e classificação de anomalias em redes por meio de sondas específicas para diferentes protocolos. A proposta integra abordagens heurísticas, aprendizado de máquina tradicional e uma estratégia generativa local em uma arquitetura modular, com consolidação por ensemble ponderado. Como contribuição, o trabalho descreve a arquitetura desenvolvida, avalia experimentalmente diferentes estratégias de classificação e discute suas limitações e possibilidades de evolução, fornecendo uma base para pesquisas futuras sobre a aplicação de Inteligência Artificial (IA) na operação de redes.

2. METODOLOGIA

O trabalho desenvolvido corresponde a um protótipo de sistema de captura de anomalias por protocolo, com interface web para visualização das ocorrências e dos indicadores coletados. A implementação atual está organizada em uma arquitetura distribuída baseada em containers, na qual componentes especializados operam de forma coordenada por meio de comunicação assíncrona. O ambiente é composto por um container para a interface web, um container para a API FastAPI/Uvicorn, um container para o banco de dados PostgreSQL, um broker RabbitMQ para filas de mensagens e um container com o motor de automação n8n para encaminhamento de alertas. Embora o núcleo funcional permaneça concentrado em um backend modular, a arquitetura adotada permite a separação operacional dos serviços e facilita a implantação, manutenção e evolução do protótipo. A estrutura geral do ambiente é apresentada na Figura 1.

O funcionamento do protótipo é conduzido por um motor assíncrono, implementado com `asyncio`, que executa ciclos periódicos de captura e publica tarefas para processamento posterior. As sondas são organizadas por protocolo: ICMP verifica disponibilidade, latência e perda de pacotes; TCP testa a acessibilidade de portas; HTTP registra código de resposta e tempo de atendimento; DNS verifica a resolução de nomes; e SNMP coleta indicadores de dispositivos e interfaces. Quando ocorrências de falha ou degradação são identificadas, os eventos são encaminhados para um *pipeline* de notificações que utiliza *RabbitMQ* e o *n8n*, possibilitando o envio automatizado de mensagens para o *Telegram*. A classificação é disponibilizada pelo *endpoint* `POST//api/anomalies/classify`, que aceita métricas capturadas pelo protótipo ou amostras oriundas de cenários simulados.

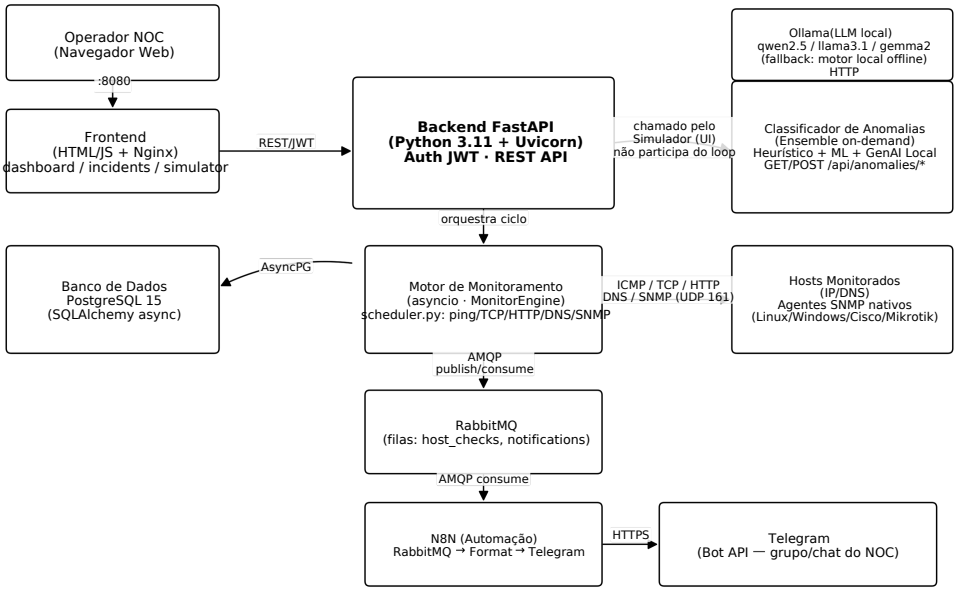

Figura 1: Arquitetura distribuída em contêineres do protótipo de sistema de captura de anomalias por protocolo. Fonte: Autor(es).

2.1 Fluxo de Classificação

O módulo de classificação de anomalias foi implementado como um *pipeline* composto por múltiplos motores de inferência, cada um responsável por analisar um mesmo vetor de características e produzir uma previsão sobre o tipo de ocorrência observada. O fluxo, ilustrado na Figura 2, parte das métricas capturadas pelas sondas de protocolo, segue para a extração de atributos numéricos e culmina na geração de uma decisão consolidada sobre o cenário operacional. O conjunto de classes considerado inclui `NORMAL`, `DDOS_ATTACK`, `PORT_SCAN`, `HOST_DOWN`, `MEMORY_LEAK`, `DATA_EXFILTRATION`, `DISK_EXHAUSTION`, `SERVICE_OUTAGE`, `COMBINED_FAULT`, `NETWORK_CONGESTION` e `CUSTOM`.

A primeira estratégia implementada é um motor heurístico baseado em regras de limiar definidas manualmente. Esse módulo avalia indicadores como perda de pacotes, latência, jitter, tráfego de rede, consumo de CPU, RAM e disco, e classifica a ocorrência com base em padrões operacionais previamente codificados. A segunda estratégia utiliza aprendizado de máquina clássico por meio de um modelo de `RandomForestClassifier`, complementado por um `IsolationForest` para apoio à detecção de anomalias. O treinamento é realizado a partir de dados sintéticos gerados pelo próprio protótipo, utilizando um conjunto balanceado de amostras para calibrar os modelos e permitir a comparação entre abordagens tradicionais e a estratégia generativa. A terceira estratégia incorpora IA generativa local por meio de um provedor baseado em Ollama. Nesse módulo, as métricas numéricas são convertidas em uma descrição textual estruturada e enviadas ao modelo para análise semântica do cenário. O protótipo também conta com um mecanismo de contingência, de modo que, em caso de indisponibilidade do serviço local, a decisão pode ser substituída por uma resposta baseada no motor heurístico. Por fim, o módulo de ensemble consolida as saídas dos três motores, produzindo uma decisão final acompanhada de indicador de concordância, nível de risco e recomendação de ação. Os

pesos foram definidos empiricamente, com o objetivo de refletir maior confiança na abordagem baseada em heurísticas, seguida pelo modelo de aprendizado de máquina e, por último, pelo modelo baseado em LLM.

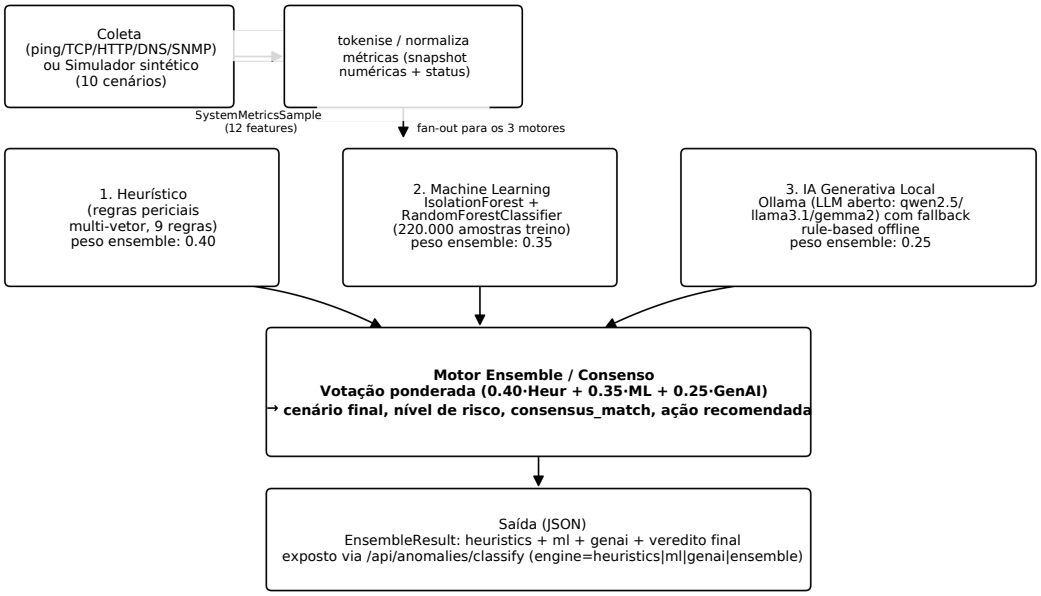

Figura 2: Fluxo de classificação de anomalias: coleta de métricas, processamento por múltiplos motores e consolidação da decisão final. Fonte: Autor(es).

2.2 Simulador

O módulo de simulação de anomalias, implementado em `anomaly_simulator.py`, tem por objetivo gerar cenários sintéticos representativos de condições normais e anormais de operação. A estrutura de dados utilizada para esse processo é a classe `SystemMetricsSample`, que encapsula métricas de latência, perda de pacotes, jitter, utilização de CPU, RAM e disco, tráfego de entrada e saída, código de resposta HTTP, tempo de resposta, estado de portas TCP e número de conexões ativas. As amostras são geradas por meio de distribuições probabilísticas estocásticas, com ajuste de parâmetros específicos para cada cenário, o que permite reproduzir comportamentos como sobrecarga de rede, vazamento de memória, esgotamento de disco, indisponibilidade de host e ataques de varredura de portas.

A simulação cobre tanto cenários saudáveis quanto cenários anômalos, permitindo que o protótipo gere dados para treinamento, testes e avaliação comparativa dos classificadores. As amostras sintéticas são consumidas pela API de classificação, pelo *pipeline* de treinamento do modelo de aprendizado de máquina e pela lógica de testes do protótipo, o que torna o processo independente de bases públicas e facilita a reprodução controlada de diferentes condições operacionais. Dessa forma, o simulador atua como componente central para a validação funcional do protótipo, pois possibilita a geração de dados com características conhecidas e a comparação do desempenho dos motores heurístico, estatístico e generativo sob os mesmos cenários.

3. Resultados e Discussão

Os experimentos foram conduzidos utilizando 140 amostras sintéticas balanceadas, com 20 amostras por cenário e 7 cenários principais avaliados. Os resultados foram agregados a partir de repetições independentes e resumidos na Tabela 1. Para a acurácia, foram reportadas média, desvio padrão e intervalo de confiança de 95%; para F1-macro e latência, foram reportadas as médias observadas nas repetições.

Table 1: Resumo estatístico dos classificadores avaliados.

Abordagem	**Acurácia**	**F1-macro**	**Latência média (ms)**	**Intervalo de Confiança 95%**
Heurística	1,0000	1,0000	0,0073	[1,0000; 1,0000]
ML	0,9403	0,9563	44,9579	[0,9050; 0,9700]
GenAI	1,0000	1,0000	0,0,0184	[1,0000; 1,0000]
Ensemble	1,0000	1,0000	45,0553	[1,0000; 1,0000]

A abordagem heurística obteve acurácia média de 1,0000 e F1-macro de 1,0000, com latência média de 0,0073 ms, indicando execução praticamente instantânea no cenário sintético utilizado. O classificador de machine learning apresentou acurácia média de 0,9403, F1-macro de 0,9563 e latência média de 44,9579 ms, com desvio padrão de 0,0166 na acurácia e intervalo de confiança de 95% entre 0,9050 e 0,9700. A componente generativa local apresentou acurácia média de 1,0000 e F1-macro de 1,0000, com latência média de 0,0184 ms, enquanto o ensemble também obteve acurácia e F1-macro iguais a 1,0000, com latência média de 45,0553 ms.

Em contraste, o classificador baseado em LLM, avaliado separadamente sobre 48 amostras sintéticas, alcançou acurácia aproximada de 0,46 e F1-macro de 0,34, com latência mediana de 3.557 ms. Além do custo computacional significativamente superior aos modelos tradicionais, foram observadas inconsistências decorrentes de ambiguidades nas respostas do modelo e de falhas no processamento do JSON retornado, o que comprometeu a confiabilidade da classificação automática.

Do ponto de vista comparativo, os resultados indicam que, para o conjunto de dados sintético utilizado, os classificadores supervisionados e a heurística são alternativas mais eficientes e estáveis do que a classificação puramente generativa baseada em LLM. A heurística apresentou desempenho máximo com custo computacional mínimo, enquanto o modelo de machine learning ofereceu um equilíbrio entre desempenho e generalização, ainda que com maior latência. O ensemble, por sua vez, consolidou as decisões com alta confiança e manteve o desempenho máximo observado para acurácia e F1-macro, porém com latência semelhante à do classificador de machine learning, em razão do custo de agregação das saídas dos motores.

Para complementar a avaliação experimental, os resultados foram reportados como média, desvio padrão e intervalo de confiança de 95%. Como as abordagens foram avaliadas sobre o mesmo conjunto de amostras, empregou-se o teste pareado de Wilcoxon para verificar a significância das diferenças entre os classificadores. Na métrica de acurácia, o classificador de machine learning apresentou diferença estatisticamente significativa em relação às abordagens heurística, generativa e ao ensemble (p = 0,0005), reforçando a robustez da comparação experimental.

De forma geral, os resultados evidenciam que o protótipo é capaz de capturar, estruturar e classificar métricas de rede por protocolo com alto grau de consistência no ambiente sintético proposto. Embora o LLM tenha se mostrado menos adequado como classificador principal, ele ainda apresenta utilidade como componente de apoio à decisão e explicabilidade, especialmente

quando integrado a uma arquitetura híbrida em que heurísticas e modelos supervisionados realizam a classificação principal.

4. CONCLUSÃO

Os resultados demonstram que a arquitetura do protótipo é capaz de capturar, estruturar e representar métricas provenientes de diferentes protocolos para a classificação de anomalias. Os modelos avaliados apresentaram bom desempenho sobre o conjunto sintético utilizado, enquanto a abordagem heurística mostrou-se altamente eficaz no cenário atual e o ensemble consolidou as decisões com confiança elevada. A conversão de atributos numéricos em representações textuais possibilitou a utilização de um modelo generativo no processo de classificação, embora a configuração atual esteja mais alinhada ao apoio à decisão e à explicabilidade do que à substituição dos classificadores tradicionais. Dessa forma, os experimentos sugerem que uma arquitetura híbrida, na qual métodos heurístico e supervisionados realizem a classificação principal e a componente generativa atue como ferramenta de interpretação dos resultados, representa uma estratégia promissora para trabalhos futuros.

Como continuidade deste trabalho, pretende-se estudar o comportamento isolado da classificação utilizando apenas LLM, ampliar a validação com novos conjuntos de dados e implementar mecanismos de mitigação automática em ciclo fechado. Além disso, pretende-se comparar o desempenho dessas abordagens com modelos proprietários, ampliando a validação experimental da proposta.

REFERÊNCIAS

Bommasani, R. et al. (2021), "On the Opportunities and Risks of Foundation Models", *arXiv preprint arXiv:2108.07258*.

Buczak, A.L.; Guven, E. (2016), "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection", *IEEE Communications Surveys & Tutorials*, vol. 18, n. 2, 1153–1176.

Kim, H.; Feamster, N. (2013), "Improving Network Management with Software Defined Networking", *IEEE Communications Magazine*, vol. 51, n. 2, 114–119.

Kreutz, D. et al. (2015), "Software-Defined Networking: A Comprehensive Survey", *Proceedings of the IEEE*, vol. 103, n. 1, 14–76.

Zhao, W.X. et al. (2023), "A Survey of Large Language Models", *arXiv preprint arXiv:2303.18223*.