Detecção de Ataques de Front-Running na Blockchain Ethereum aplicando técnicas de Mineração de Dados e Aprendizado de Máquina

Marcelo Corni Alves
Universidade Federal de Juiz de Fora
marcelo.corni@estudante.ufjf.br

Figure 1: Front-runner. (Fonte: Bing Image Creator,2024)

RESUMO

Neste trabalho, foi proposta uma pipeline para a detecção de ataques de Front-Running, utilizando dados da Blockchain Ethereum, com foco nas transações realizadas entre 01/01/2024 e 07/01/2024. A pipeline envolve as etapas de pré-processamento de dados, processamento com técnicas de aprendizado de máquina, como Autoencoder e Isolation Forest, e avaliação dos resultados através de gráficos e métricas. A detecção dos ataques do tipo supressão, deslocamento e inserção foi identificada como um desafio de pós-processamento futuro, sendo o foco de adaptação de algoritmos presentes na literatura, como o código em Python de Frontrunner Jones and the Raiders of the Dark Forest[1]. Os resultados iniciais mostram um potencial significativo no uso de técnicas automatizadas para detecção de anomalias na Blockchain.

PALAVRAS-CHAVE

Blockchain, Ethereum, Front-Running, Anomalias, Autoencoder, Isolation Forest

Referência:
Marcelo Corni Alves. 2024. Detecção de Ataques de Front-Running na Blockchain Ethereum aplicando técnicas de Mineração de Dados e Aprendizado de Máquina. Universidade Federal de Juiz de Fora, 2024, 4 páginas. http://github.com/marcelocorni

1 INTRODUÇÃO

A Blockchain Ethereum, amplamente utilizada para transações financeiras, contratos inteligentes e aplicações descentralizadas, apresenta vulnerabilidades exploradas por ataques de Front-Running. Esses ataques consistem em manipular a ordem das transações na Blockchain, visando obter vantagens financeiras. Três tipos principais de ataques são destacados na literatura: supressão, deslocamento e inserção. Enquanto o Ethereum evolui como uma plataforma de contratos inteligentes, o monitoramento de transações e a detecção de ataques permanecem desafios críticos, especialmente em ambientes DeFi (Finanças Descentralizadas).

O aumento da utilização de redes descentralizadas, combinado com a complexidade crescente das transações, reforça a necessidade de desenvolvimento de ferramentas eficazes para monitoramento e detecção de ataques. A motivação deste trabalho está em desenvolver uma pipeline automatizada para identificar possíveis comportamentos maliciosos e otimizar a segurança na rede Ethereum.

2 METODOLOGIA E RESULTADOS

A pipeline desenvolvida segue quatro etapas principais: Pré-processamento Processamento, Avaliação e Pós-Processamento.

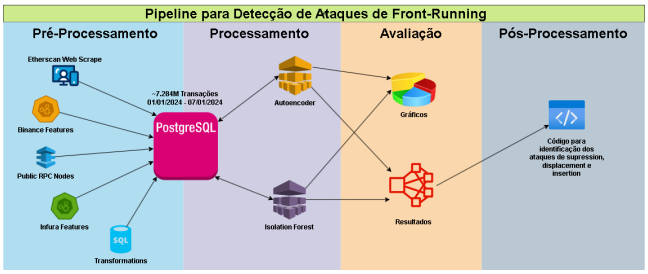

Figure 2: Pipeline para Detecção de Ataques de Front-Running

2.1 Pré-Processamento

As fontes de dados utilizadas incluem o Etherscan Web Scrape, características da Binance, e APIs de nós públicos de RPC e Infura. Esses dados fornecem informações sobre mais de 7.28M transações, realizadas entre 01/01/2024 e 07/01/2024, armazenadas em uma base de dados PostgreSQL. Nesse estágio, foram realizadas transformações para unificar os diferentes dados e integrá-los ao banco de dados para as etapas subsequentes. Abaixo são apresentadas as distribuições dos dados e de algumas features.

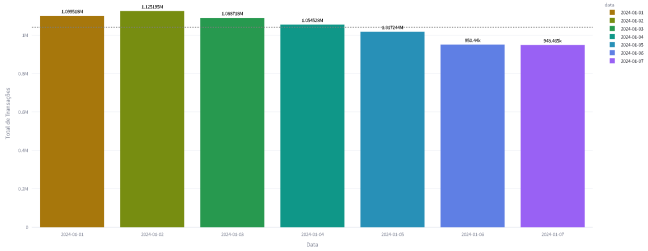

Figure 3: Quantidade de Transações por Dia

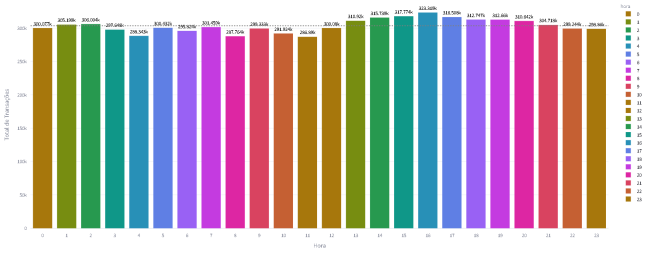

Figure 4: Quantidade de Transações Agrupadas por Hora do Dia

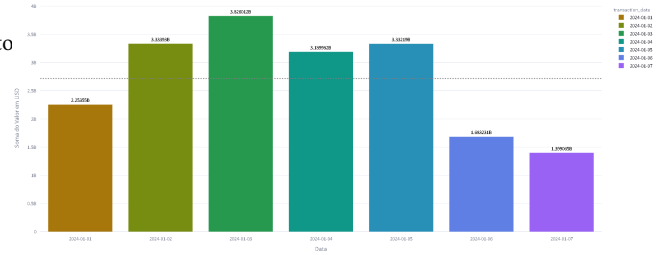

Figure 5: Valor Total em USD por Dia

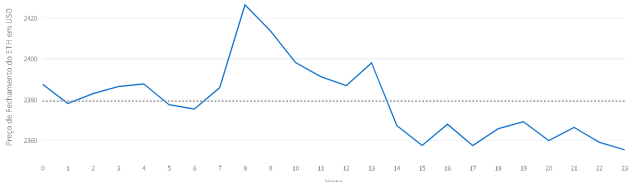

Figure 6: Variação Máxima do Preço de Fechamento do Ethereum por Hora do Dia

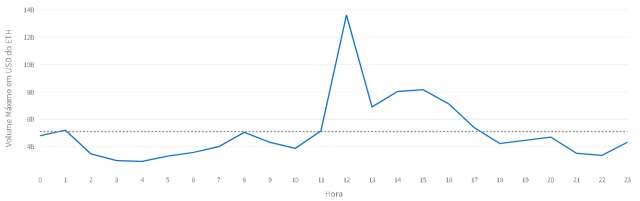

Figure 7: Volume Máximo em USD do Ethereum por Hora do Dia

2.2 Processamento

As técnicas de aprendizado de máquina escolhidas foram Autoencoder e Isolation Forest, ambas adequadas para detectar padrões anômalos. O Autoencoder foi utilizado para reduzir a dimensionalidade e extrair características relevantes, bem como para deteção de anomalias baseadas no erro de reconstrução. O Isolation Forest focou na detecção de anomalias. Ambas as técnicas apresentam resultados onde podem existir possíveis ataques de Front-Running. Cada modelo foi treinado com os dados pré-processados para identificar comportamentos anômalos nas transações.

Para o processamento foram selecionadas 12 features que tiveram correlações mais baixas para uma melhor qualidade nos resultados de detecção de padrões anômalos.

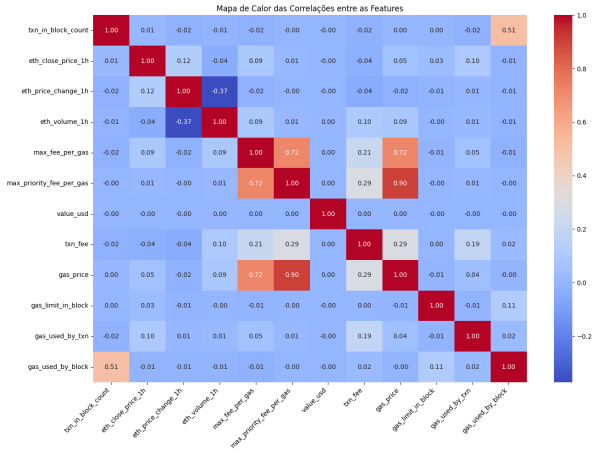

Figure 8: Mapa de Calor da Correlação entre as Features

O Isolation Forest foi parametrizado com *max_samples* igual 10% do total dos dados de transações, *estimators* igual à 150 e *contamination* igual 0.05. O tempo total para o processamento foi de 11 minutos e 37 segundos para 7.14M de transações com status igual à *Success*.

O Autoencoder foi parametrizado com *input_dim* igual à 12 features de entrada, *latent_space_dim* igual à 4, *learning_rate* igual à 0.05, *batch_size* igual à 32 e *epochs* igual à 20. O tempo total para o processamento foi de 1 hora, 3 minutos e 38 segundos para 7.14M de transações com status igual à *Success*.

Ambos os algoritmos tiveram separação de dados para treino, teste/validação na razão 70/30.

2.3 Avaliação

A avaliação dos resultados foi realizada com base em gráficos e relatórios gerados. As transações identificadas como anômalas foram marcadas para posterior análise. Além disso, foram gerados gráficos que mostram a distribuição das anomalias por feature, oferecendo insights visuais sobre os padrões detectados.

2.3.1 *Isolation Forest.*

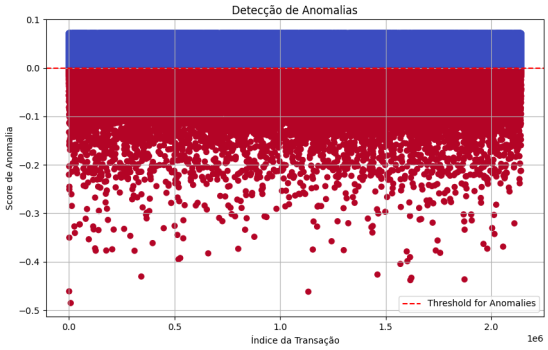

Figure 9: Detecção de anomalias

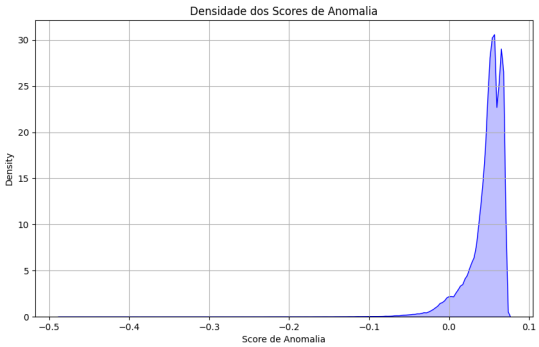

Figure 10: Densidade dos Scores de Anomalia

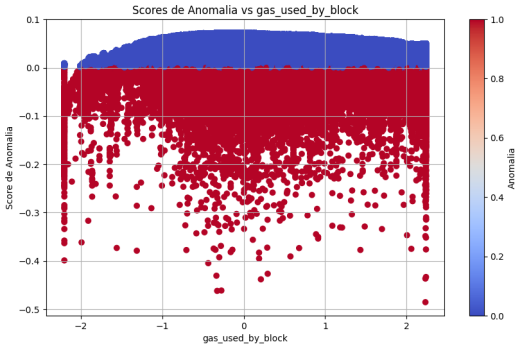

Figure 11: Scores de Anomalia vs gas_used_by_block

2.3.2 *Autoencoder.*

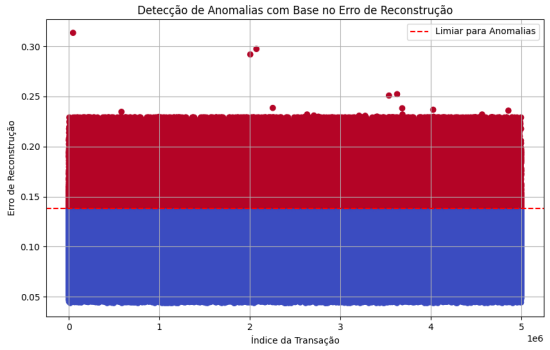

Figure 12: Detecção de anomalias por Erro de Reconstrução

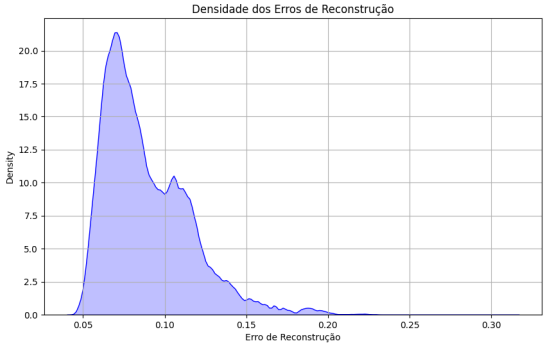

Figure 13: Densidade dos Erros de Reconstrução

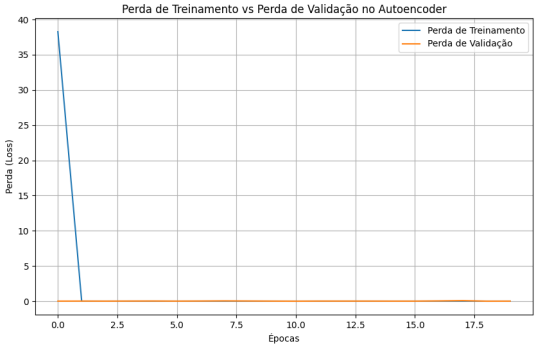

Figure 14: Perda de Treinamento vc Perda de Validação

3 CONCLUSÃO

4 TRABALHOS FUTUROS

REFERENCES

[1] Christof Torres. [n. d.]. Frontrunner-Jones. https://github.com/christoftorres/Frontrunner-Jones.