Melhorando a Acurácia da Detecção de Lavagem de Dinheiro na Rede Bitcoin

Gabriel Fontes Rebello[1], **Yining Hu**[2], **Kanchana Thilakarathna**[3], **Gustavo E. A. P. A. Batista**[2], **Aruna Senenviratine**[2] **e Otto Carlos M. B. Duarte**[1]

[1] Universidade Federal do Rio de Janeiro, RJ, Brasil
[2]University of New South Wales, Sydney, Austrália
[3]University of Sydney, Austrália

Resumo. *A rede Bitcoin permite a transferência de criptomoeda com um baixo custo, de forma rápida, sem limites geográficos e sem a intervenção de um banco intermediador. É uma possível solução para mais de um bilhão de pessoas que não possui acesso ao sistema financeiro por causa dos altos custos. Por outro lado, a rede Bitcoin é pseudônima e tem sido usada para uma enorme variedade de atividades financeiras dúbias e ilegais. Este artigo investiga as atividades de lavagem de dinheiro na rede Bitcoin através de diversos mecanismos que procuram melhorar o desempenho de classificadores na análise de um conjunto de dados desbalanceado devido a uma classe minoritária com poucas amostras. Os experimentos realizados mostram a eficácia de cada estratégia na melhora da classificação das atividades de lavagem de dinheiro tais como: i) descoberta automática de características; ii) o uso de diferentes classificadores de aprendizado de máquina, inclusive com o uso de algoritmo de reforço de aprendizado adaptativo; iii) o percentual de repartição do conjunto de dados em treino e teste e iv) heurísticas de sobreamostragem. Os resultados mostram um bom desempenho do algoritmo de sobreamostragem ADASYN e que o maior ganho em desempenho foi com o classificador floresta aleatória.*

1. Introdução

A rede Bitcoin vem revolucionando o mundo ao criar uma camada de confiança para transferências seguras de ativos. A criptomoeda Bitcoin [Nakamoto 2008] consiste em um ecossistema financeiro completo no qual cerca de 130 bilhões de reais são movimentados diariamente [CoinMarketCap 2019] em 300 mil transações realizadas todos os dias [Blockchain.com 2019] por mais de 30 milhões de usuários [Spilotro 2019]. Isto corresponde a uma movimentação diária maior do que o PIB anual de mais de 95 países [World Bank 2019]. Ainda, o Bitcoin induziu a criação de um grande número de *startups* de transferência de dinheiro que utilizam a criptomoeda para eliminar a necessidade de intermediários e realizar transferências monetárias com baixo custo. Essas *startups* já competem pelo mercado com serviços tradicionais de transferência de dinheiro, como o Western Union. A criptomoeda tem sido considerada como uma possível solução de baixo custo para os mais 1,7 bilhões de pessoas que vivem à margem do sistema financeiro [Demirguc-Kunt et al. 2018].

O Bitcoin também é amplamente conhecido por ser utilizado como forma de pagamento em atividades ilegais, como serviços de terrorismo cibernético, lavagem de dinheiro, *malware*, sequestro de dados, esquemas de pirâmide, etc. Estima-se que até 25% dos usuários na rede Bitcoin estejam envolvidos em atividades maliciosas ou suspeitas [Foley et al. 2019]. Atraídos pelo pseudoanonimato provido pelas características

Este trabalho foi realizado com recursos do CNPq, CAPES, FAPERJ e FAPESP (18/23292-0, 2015/24514-9, 2015/24485-9 2014/50937-1).

do Bitcoin, criminosos e *hackers* utilizam a criptomoeda para conduzir ataques e operar mercados negros de comercialização de drogas, armas e produtos ilegais.

O mecanismo de lavagem de dinheiro [Elliptic Inc. 2019] é simples: o criminoso converte o dinheiro sujo em Bitcoin através de uma corretora, anonimiza suas transações na rede através de serviços de misturadoras (*Bitcoin Mixers*)[1], aguarda por um período de tempo e converte os Bitcoin de volta para dinheiro real, agora limpo como se fosse fruto da supervalorização monetária. Alternativamente, como em casos de lavagem de dinheiro com dinheiro físico, o criminoso pode fundar uma companhia que recebe pagamentos em Bitcoin e gera lucro exagerado para justificar a posse do dinheiro. As medidas para se evitar lavagem de dinheiro são responsáveis por boa parte dos custos financeiros para transferência de moeda que acaba atingindo os mais carentes, os imigrantes e os refugiados. O escritório de drogas e crime da organização das nações unidas estima que a lavagem de dinheiro corresponda de 2% a 5% do produto interno bruto mundial, ou seja, entre 715 bilhões e 1,87 trilhões de dólares anuais [United Nations Office on Drugs and Crime 2019], a maior parte em dinheiro líquido.

Paradoxalmente à atratividade da rede Bitcoin para atividades ilegais devido ao pseudoanonimato, a rede Bitcoin registra de forma permanente todas as transações da rede e as disponibiliza publicamente para quem quiser acessar. Mesmo que se utilize o serviço de misturadoras, todo ato ilícito na rede Bitcoin tem um traço. Além disso, para evitar gasto duplo, o Bitcoin utiliza o modelo de saídas não-gastas (*Unspent Transactions Outputs* – UTXO), que gera enlaces entre todas transações registradas na rede. Isto dá rastreabilidade à origem das transações e permite representar o conjunto de todas as transações como um grafo direcionado. Portanto, ainda que as misturadoras ofusquem a origem de uma lavagem de dinheiro, é factível buscar traços e identificar transações ilícitas com técnicas de aprendizado de máquina.

Este artigo foca na detecção de atividades de lavagem de dinheiro na rede Bitcoin através de técnicas de aprendizado de máquina. A lavagem de dinheiro faz parte da classe de conjunto de dados desbalanceados, uma importante classe de problemas de aprendizado de máquina na qual a classe que se procura no conjunto de dados é composta de amostras raras que ficam sub representadas. A maioria dos algoritmos de mineração de dados apresenta desempenho abaixo do ideal quando se trata de raridade, porque objetos raros são geralmente muito mais difíceis de identificar do que objetos comuns [Chawla et al. 2004, Weiss 2004]. Por exemplo, dados desbalanceados são um problema para aplicações como: fraude em cartões de crédito [Chan e Stolfo 1998], detecção de intrusão [Tan et al. 2019], derramamento de óleo [Kubat et al. 1998], gerenciamento de riscos, falha no equipamento de telecomunicações [Weiss e Hirsh 1998], diagnóstico/monitoramento médico, etc.

O objetivo do artigo é analisar e verificar o desempenho, através de experimentos, da detecção de lavagem de dinheiro obtida por diferentes técnicas disponíveis na literatura para contornar o problema de desbalanceamento de dados. Os experimentos concentram-se em um conjunto de dados específico, o Elliptic Dataset de uma empresa estadunidense de segurança e criptomoeda, que é o maior conjunto de dados rotulados e publicamente disponível de uma criptomoeda. O artigo se serve da técnica de aprendizado por grafo obtido das saídas não-gastas de transações (*Unspent Transaction Outputs* - UTXO).

Os experimentos realizados objetivam melhorar o desempenho das técnicas de

[1]Misturadoras são companhias que ofuscam a origem de uma quantia em Bitcoin realizando uma série de transações entre diferentes endereços criados aleatoriamente.

aprendizado de máquina na obtenção da detecção de atividades de lavagem de dinheiro, considerando o desbalanceamento do conjunto de dados. O primeiro experimento se serve do algoritmo node2vec para aprender as características usando a técnica de caminhadas aleatórias. O segundo experimento compara o desempenho de diferentes algoritmos de aprendizado de máquina, como Naive Bayes, floresta aleatória (*random forest*), árvore de decisão, etc. O ganho de desempenho do algoritmo AdaBoost como técnica de reforço de aprendizagem associada a um algoritmo de árvore de decisão é obtido. O terceiro experimento avalia o desempenho dos classificadores quando se altera o percentual de partição do conjunto de dados para teste e treino. O quarto experimento se serve dos algoritmos SMOTE [Chawla et al. 2002] e ADASYN [He et al. 2008] como técnica de balancear o conjunto de dados com sobreamostragens.

O restante do artigo está organizado da seguinte forma. A Seção 2 descreve a geração do grafo de transações a partir da rede Bitcoin e o conjunto de dados Elliptic. A Seção 3 descreve a aprendizagem por grafos usando o algoritmo node2vec. A Seção 4 apresenta as métricas utilizadas para avaliar os classificadores em conjuntos de dados desbalanceados. A Seção 5 apresenta a avaliação dos algoritmos de aprendizado de máquina. A Seção 6 mostra o impacto do particionamento no desempenho dos classificadores. A Seção 7 apresenta os métodos utilizados para balancear o conjunto de dados. Por fim, a Seção 8 conclui o artigo.

2. Geração do Grafo de Transações a Partir da Rede Bitcoin

A rede Bitcoin possui uma característica intrínseca para impedir o gasto duplo, que facilita a representação de todas as transações em um grafo: o modelo de saídas não-gastas de transações (*Unspent Transaction Outputs* - UTXO). Uma transação neste modelo possui como componentes principais um *hash* identificador da transação (*TXID*) e uma lista de entradas e saídas. As entradas da transação referenciam saídas de transações passadas que ainda não tenham sido gastas na rede, através do *hash* identificador da transação precedente (*prevTx*) e de um índice que localiza a saída correspondente na lista de saídas de transações passadas. Cada saída de uma transação pode ser usada apenas uma vez como entrada em toda a corrente de blocos. Referenciar a mesma saída duas vezes é considerado uma tentativa de "gastar duas vezes a mesma moeda" e, portanto, não é permitido. Devido a essa propriedade, cada saída de uma transação é chamada de uma saída de transação não gasta (*Unspent Transaction Output* - UTXO), caso não tenha sido referenciada por outra transação até o momento, ou como saída de transação gasta (*Spent Transaction Output* - STXO) caso contrário.

As UTXOs são pedaços indivisíveis de Bitcoin pertencentes a um proprietário. As ações de pagar em Bitcoin e receber em Bitcoin correspondem a transferir UTXOs do pagador para o destinatário que vai receber o valor em Bitcoin. O pagador desbloqueia sua UTXO e assina a transferência para o destinatário com sua chave privada, garantindo a autenticidade da operação, e encripta o valor com a chave pública do receptor. Logo, só o receptor consegue desbloquear o valor transferido.

Toda transação Bitcoin consome UTXOs, desbloqueando-as com a chave privada do atual proprietário da UTXO, e cria novas UTXOs que só podem ser desbloqueados pela chave privada do destinatário. Assim, quantidades de Bitcoin são transferidas de proprietário para proprietário, criando uma cadeia de transações. Essas cadeias de transações que estão ligadas entre si por UTXOs de entrada e saída geram um grafo de transações, como ilustrado na Figura 1. Portanto, é possível abstrair em forma de grafo todas as transações em uma corrente de blocos que utilize o modelo UTXO. Cada vértice no

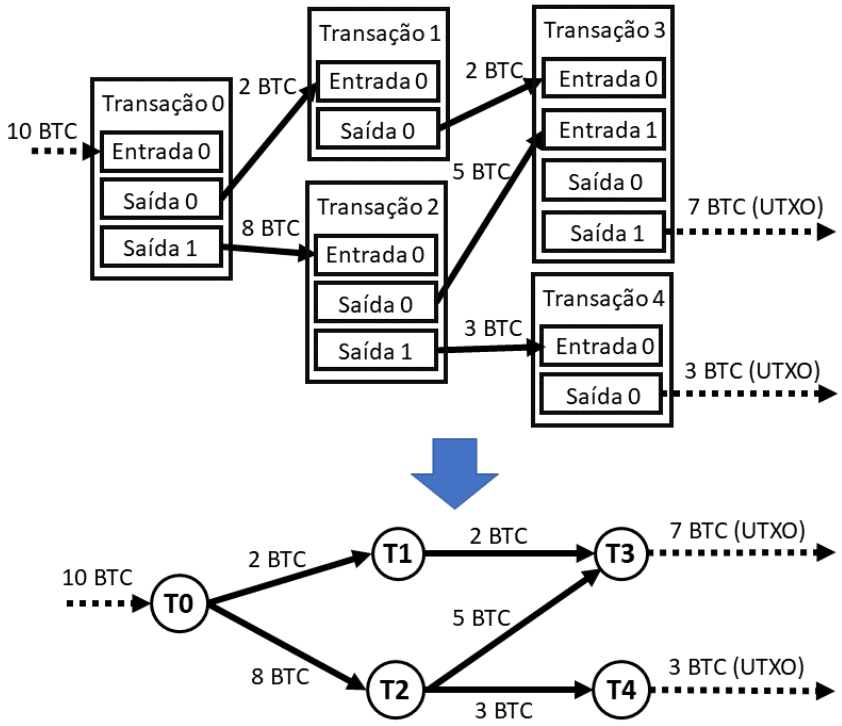


Figura 1. Grafo gerado por transações no modelo de saídas não-gastas de transações (*Unspent Transaction Outputs* - UTXO) [Nakamoto 2008] da criptomoeda Bitcoin (BTC). Cada transação corresponde a um vértice e cada fluxo de BTC corresponde a uma aresta no grafo.

grafo representa uma transação e cada aresta direcionada entre dois vértices representa uma saída não-gasta de uma transação passada que foi utilizada pela transação seguinte. Assim, todas as transações e quantidade de Bitcoins que são pagos e recebidos estão registradas de forma permanente e transparente na rede. O proprietário é identificado pelo seu endereço ou chave pública. Logo, a identificação na rede Bitcoin é considerada pseudônima, mas não anônima.

2.1. O Conjunto de Dados Elliptic

Weber *et al.* disponibilizaram o conjunto de dados Elliptic[2], que mapeia as transações da rede Bitcoin para entidades reais que realizam operações lícitas (conversão de moedas, provedores de carteira, mineradoras, serviços lícitos, etc.) ou ilícitas (golpes, *malware*, organizações terroristas, *ransomware*, esquemas de pirâmide,etc.) [Weber et al. 2019]. O mapeamento para cada tipo de entidade é realizado pela ferramenta de análise forense da Elliptic, empresa estadunidense de segurança em criptomoedas. A ferramenta combina dados públicos da corrente de blocos do Bitcoin com um conjunto de dados proprietário de endereços associados a entidades conhecidas para identificar cada usuário através dos seus conjuntos de chave pública e ferramentas de desanonimização. A partir dos dados da corrente de blocos (*blockchain*) do Bitcoin, o grafo é construído e rotulado como descrito na Seção 2, de forma que os vértices representam transações e as arestas representam um fluxo de Bitcoin passando de uma transação para a próxima através das UTXOs. Uma transação é considerada ilícita se a entidade

[2]O conjunto de dados está disponível em https://www.kaggle.com/ellipticco/elliptic-data-set

que inicia a transação, i.e. a entidade que controla as chaves privadas associadas com os endereços de entrada da transação, pertence a uma categoria ilícita no banco de dados proprietário da Elliptic. Caso contrário, a transação é lícita.

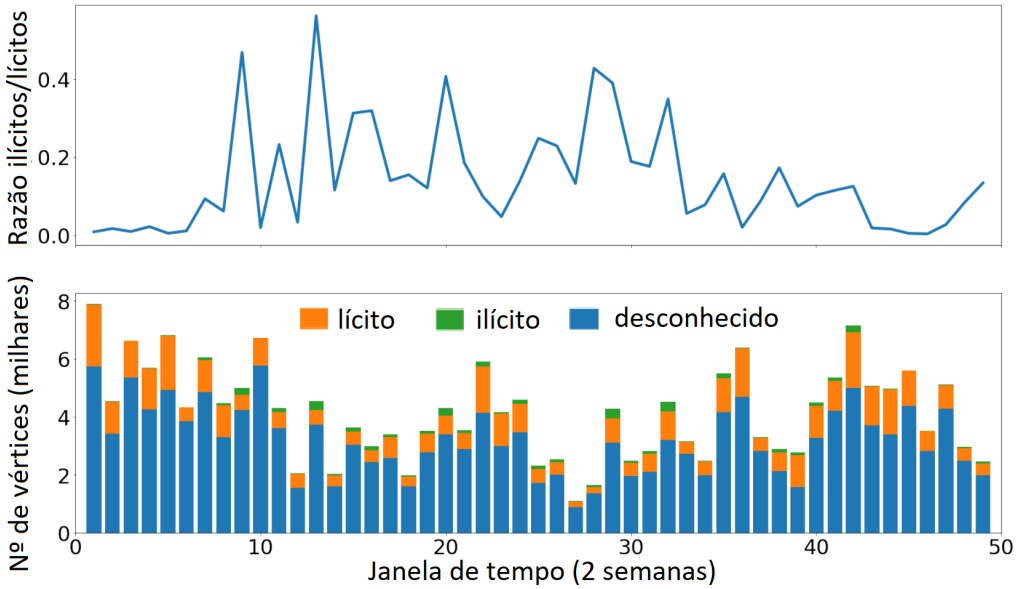


Figura 2. Série temporal do conjunto de dados Elliptic com 49 janelas de tempo com duas semanas cada, coletadas entre 2014 e 2016. A parte inferior ilustra o total de transações ocorridas em cada janela de tempo e a parte superior representa a proporção entre vértices ilícitos em cada janela (adaptado de [Weber et al. 2019]).

No total, o conjunto de dados Elliptic possui 203.769 vértices (transações) e 234.355 arestas direcionadas (fluxos de pagamentos de Bitcoin) coletadas entre os anos de 2014 e 2016. O grafo completo utilizando toda a rede Bitcoin possui aproximadamente 438 milhões de vértices e 1,1 bilhão de arestas. No conjunto de dados Elliptic, dois por cento dos vértices (4.545) correspondem a transações ilícitas, vinte e um por cento (42.019) correspondem a transações lícitas, e o restante dos vértices não é rotulado pois o resultado da ferramenta de análise forense é inconclusivo quanto à classe.

O conjunto de dados Elliptic também associa uma estampa de tempo (*timestamp*) a cada vértice contendo uma estimativa da data e hora em que a rede Bitcoin confirmou a transação correspondente ao vértice. O conjunto de dados divide-se em 49 janelas de tempo distintas, espaçadas uniformemente e contendo cerca de duas semanas cada. Cada janela de tempo possui uma componente conexa do grafo completo contendo todas as transações correspondentes àquele intervalo. Não há arestas conectando diferentes etapas de tempo. A Figura 2 ilustra a variação no número de vértices e na proporção entre vértices ilícitos e lícitos ao longo da série temporal do conjunto de dados.

3. Aprendizagem Automática de Características em Grafos com node2vec

Todo algoritmo de aprendizado de máquina supervisionado, para distinguir o máximo possível as classes e prover um bom desempenho de classificação, requer um conjunto de características bem-definido e independente. Logo, é preciso converter a representação dos dados em grafo, com vértices e arestas, em vetores de características associados a uma classe para serem utilizados no treino do classificador. A solução típica

para esta tarefa de engenharia de características (*feature engineering*) é a identificação manual por um especialista das melhores características. No entanto, a abordagem manual [LeCun et al. 2015]: i) depende da experiência e dos critérios adotados pelo especialista, gerando características particulares ao especialista; ii) possui baixa capacidade de generalização, pois a escolha das características é específica para aquele problema e, se o problema muda, deve-se reiniciar o processo de definir características relevantes; iii) limita a quantidade de características à criatividade do especialista e iv) é menos eficiente em conjuntos de dados muito grandes, pois o excesso de dados aumenta a probabilidade de o especialista negligenciar informações possivelmente úteis ao classificador.

As técnicas automáticas de aprendizado de características (*feature learning*) em grafos buscam definir, de maneira otimizada, as melhores características para classificação a partir do conjunto de dados bruto de vértices e arestas. O aprendizado de características em grafo substitui o aprendizado manual utilizando técnicas baseadas em buscas automáticas no grafo e redes neurais que ajustam seus pesos conforme recebem novas amostras corretas. O algoritmo de aprendizado de características pode utilizar métricas simples do grafo, como grau de um vértice, ou métricas mais complexas como vizinhança, centralidade, conectividade, etc.

Este artigo utiliza o algoritmo de aprendizado automático em grafos node2vec [Grover e Leskovec 2016] para gerar o vetor de características do problema de classificação de lavagem de dinheiro. O node2vec foi escolhido por ser o algoritmo mais eficiente para o problema de classificação de vértices em grafos [Goyal e Ferrara 2018, Grover e Leskovec 2016] e um dos mais populares no geral. O problema de lavagem de dinheiro na rede Bitcoin, tratado neste artigo, é um problema de classificação de vértices em um grafo, pois o objetivo é identificar se cada transação é lícita ou ilícita. Para o problema deste trabalho, o node2vec gera 80 características numéricas e com valor real. Não há valores ausentes.

4. Métricas de Desempenho de Classificadores para Conjuntos de Dados Desbalanceados

Duas métricas muito usadas para avaliar o desempenho de classificadores são a acurácia e a taxa de erro, definidas por

$$A_{cc} = \frac{VP + VN}{VP + FN + FP + VN} \quad e \quad E_{rr} = \frac{FP + FN}{VP + FN + FP + VN}, \tag{1}$$

onde VP é o número de verdadeiros positivos, VN é o número de verdadeiros negativos, FN é o número de falsos negativos e FP é o número de falsos positivos. No entanto, é fácil verificar que a acurácia, A_{cc}, e a taxa de erro, E_{rr}, não são métricas ideais para se utilizar em conjuntos desbalanceados de dados, uma vez que são extremamente influenciadas pela classe majoritária. Um exemplo trivial que deixa o problema claro é considerar um classificador que classifica todas as amostras como pertencentes à classe majoritária, portanto errando todas as classificações da classe minoritária. Este classificador, se utilizado em um conjunto de dados com 1.000 amostras, dentre as quais 999 amostras da classe majoritária e apenas uma amostra da classe minoritária, atinge uma acurácia de 99,9% mesmo que tenha errado todas as classificações da classe minoritária. Equivalentemente, a taxa de erro seria de apenas 0,1%. Outro fator contra o uso da acurácia e da taxa de erro é que estas métricas consideram que diferentes erros de classificação são igualmente importantes. Entretanto, nos problemas de conjuntos de dados altamente desbalanceados os custos de errar não são uniformes, o que desfavorece a identificação da classe minoritária

que é a classe de maior interesse.

Este artigo utiliza a métrica de área sob a curva característica de operação do receptor (*Area Under the Curve of the Receiver Operating Characteristics* – AUC-ROC) como métrica principal de avaliação dos classificadores testados. A área sob a curva ROC é a métrica mais utilizada para avaliar o desempenho de classificadores em conjuntos de dados desbalanceados [Batista et al. 2004], pois provê uma avaliação específica para a classe que possui menos amostras. Ainda que existam outras métricas com bom desempenho para conjuntos de dados balanceados, como a (*Area Under the Curve - Precision-Recall* – AUC-PR) [Saito e Rehmsmeier 2015], o foco deste artigo é melhorar a predição da classe positiva sem perder desempenho da classe negativa. AUC-ROC é uma medida mais balanceada que AUC-PR e considera os acertos nas duas classes igualmente, enquanto AUC-PR enfatiza a classe positiva e, com isso, considera menos as classificações corretas que ocorrem na classe negativa. Ainda, Saito *et al.* mostram que a AUC-PR é melhor que a AUC-ROC para observar a perda de desempenho de um mesmo classificador em função do desbalanceamento do conjunto de dados, mas não afirmam que a AUC-ROC é menos indicada que AUC-PR para comparação de desempenho entre classificadores diferentes em conjuntos de dados igualmente desbalanceados. O mesmo vale para diferentes particionamentos. Portanto, o uso da área sob a curva ROC permite comparar, de forma justa, o desempenho dos classificadores no conjunto de dados original e no conjunto de dados balanceado pelas técnicas de sobreamostragem. Outras métricas mais simples e específicas para uma classe, como $F_1 score$, geraram resultados similares à curva ROC e foram omitidas por restrições de espaço.

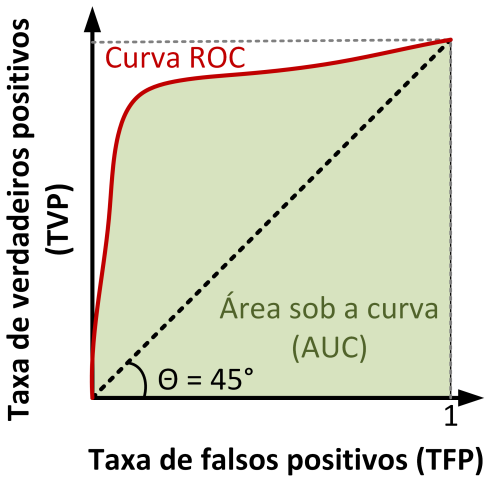


Figura 3. Exemplo de curva ROC de um classificador hipotético. A reta com inclinação de 45º indica o desempenho de um classificador totalmente aleatório.

A curva ROC, representada na Figura 3, possui como eixos a taxa de verdadeiros positivos (TVP) e a taxa de falsos positivos (TFP) da classe minoritária. Cada ponto na curva indica a probabilidade de se classificar amostras raras corretamente (taxa de verdadeiros positivos) em função da probabilidade de gerar alarmes falsos (taxa de falsos positivos), isto é, de classificar amostras da classe majoritária como amostras raras. Portanto, um classificador perfeito possui $TVP = 1$ independentemente da quantidade de amostras da classe majoritária, gerando uma curva em forma de degrau. Um classificador aleatório é incapaz de distinguir as classes e, portanto, possui $TVP = TFP$, gerando uma reta com inclinação de 45º em relação ao eixo x. A área sob a curva ROC (AUC) re-

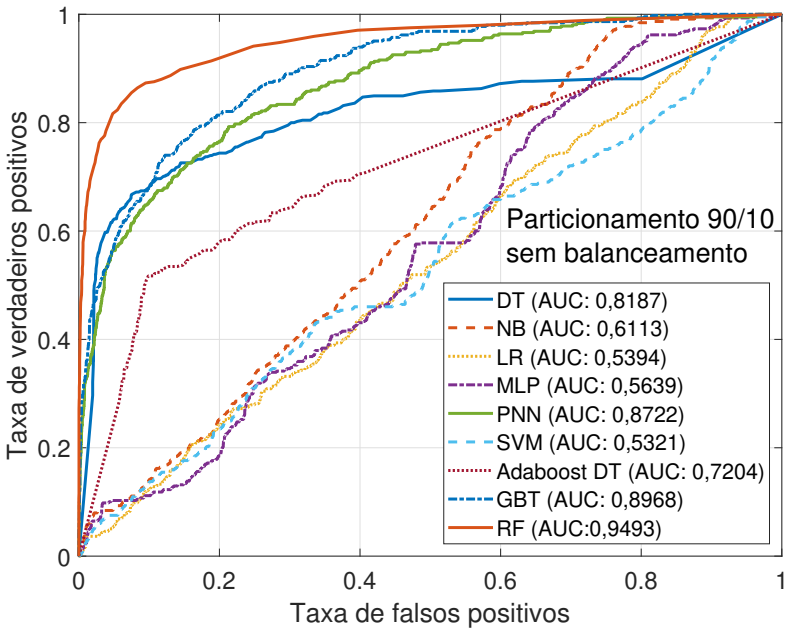


Figura 4. Comparativo de desempenho de nove classificadores para o conjunto de dados Elliptic.

presenta visual e numericamente a capacidade do modelo distinguir uma classe da outra. Quanto maior o valor da área sob a curva, melhor é o modelo de classificação. Assim, um classificador ideal possui $AUC = 1$ e um classificador aleatório possui $AUC = 0.5$.

5. Avaliação usando Algoritmos de Aprendizado de Máquina

Um dos experimentos realizados avalia o desempenho de nove classificadores para o problema de identificação de transações ilícitas na rede Bitcoin: árvore de decisão (*Decision Tree* - DT); Bayes ingênuo (*Naive Bayes* - NB); regressão logística (*Logistic Regression* - LR); máquinas de vetor de suporte (*Support Vector Machines* - SVM); perceptron multicamadas (*Multilayer Perceptron* - MLP); redes neurais probabilísticas (*Probabilitic Neural Networks* - PNN); floresta aleatória (*Random Forest* - RF); árvores reforçadas por gradiente (*Gradient Boosted Trees* - GBT) e árvores de decisão Adaboost (*Adaboost Decision Trees* - Adaboost DT). Todos os classificadores são implementados na plataforma de código aberto KNIME[3] e utilizando os hiperparâmetros padrão da plataforma. Todas os resultados utilizando particionamento simples (*holdout*) correspondem a uma média de 10 rodadas de particionamento, visando minimizar a variância dos resultados. A avaliação dos classificadores utiliza um particionamento simples com 90% de dados para treino e 10% de dados para teste. A informação de estampa de tempo das transações não foi utilizada para o particionamento, uma vez que todas as amostras pertencem à mesma janela de tempo.

A Figura 4 ilustra o desempenho de cada um dos classificadores analisados. Os classificadores convencionais, como Bayes ingênuo (NB), regressão logística (LR) e máquinas de vetor de suporte (SVM) apresentaram um péssimo desempenho, perto da classificação aleatória. A exceção foi o bom desempenho do algoritmo de árvore de decisão (DT), o que confirma a boa reputação deste algoritmo e a sua preferência na maio-

[3]Disponível em https://www.knime.com/

ria das aplicações de aprendizado de máquina por sua simplicidade, fácil compreensão e eficácia. Em relação aos algoritmos de redes neurais o algoritmo perceptron multicamadas (MLP) apresentou um péssimo resultado, similar aos citados anteriormente, confirmando que este algoritmo está caindo em desuso e sendo substituído por outros algoritmos de redes neurais mais eficientes. As redes neurais probabilísticas (PNN) apresentaram um bom desempenho, sendo o terceiro melhor resultado com $AUC = 0,8722$. No entanto, as redes neurais precisaram de um tempo muito maior para o treino que os demais.

O algoritmo de reforço de aprendizado (*learn algorithm boosting*) é um método geral para melhorar a acurácia de qualquer algoritmo de aprendizado de máquina. O algoritmo Adaptive Boosting (AdaBoost)[4] [Freund e Schapire 1999] é um meta algoritmo cuja a ideia chave é atribuir pesos diferentes às amostras o que permite "reforçar" o peso das amostras minoritárias. O AdaBoost é rápido, simples e pode ser usado com diversos algoritmos de aprendizado de máquina que usam pesos. A Figura 4 mostra que a aplicação do reforço de aprendizado junto com o método de árvore de decisão resulta em desempenho melhor que os algoritmos mais simples, mas, surpreendentemente, o resultado do algoritmo de reforço piora o resultado da árvore de decisão puro.

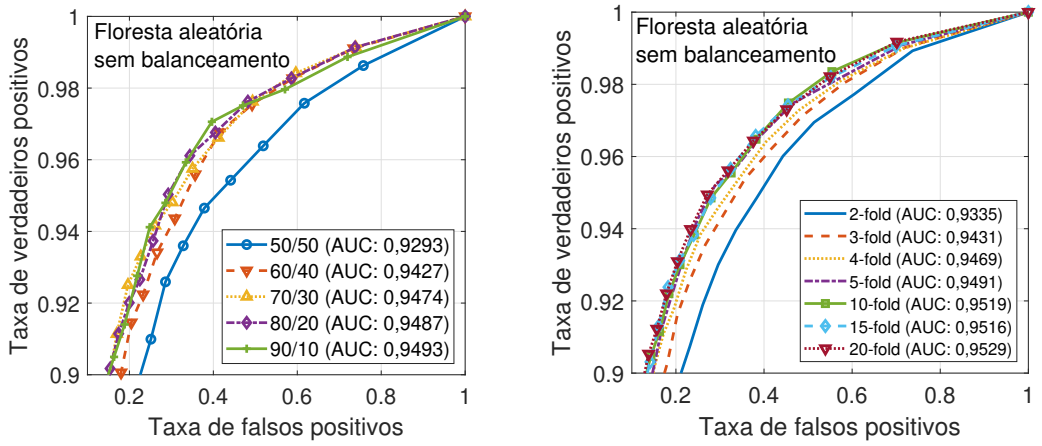
Os melhores resultados de desempenho foram os de classificadores com aprendizado em conjunto (*ensemble*), que empregam mais de um algoritmo para mitigar a variância causada nos classificadores com apenas um algoritmo. Normalmente, rodam-se diversos algoritmos e seleciona-se a moda dos valores. O melhor classificador, com $AUC = 0,9493$, é a floresta aleatória (RF). O classificador GBT, também da categoria *ensemble*, foi o segundo melhor classificador com $AUC = 0,8968$.

6. Repartição Treino/Teste do Conjunto de Dados

Para se avaliar um classificador é comum repartir o conjunto de dados em uma parte para treino e outra parte para o teste de classificação. Em conjuntos de dados altamente desbalanceados com poucas amostras da classe minoritária, é possível imaginar que quanto maior a partição reservada para treino, melhor será o desempenho do classificador. De fato, a Figura 5(a) mostra que o particionamento de 90% de dados para treino e 10% para teste é o que produz o melhor resultado de área sob a curva ROC. O experimento utiliza o classificador de floresta aleatória sem balanceamento. No entanto, o ganho em área sob a curva é cada vez menor, indicando que o desempenho se estabiliza com $AUC = 0,95$ aproximadamente. Isto significa que particionamentos com proporções extremas geraram poucos ganhos.

A Figura 5(b) analisa o desempenho do classificador de floresta aleatória sem balanceamento utilizando o algoritmo k-vezes (*k-fold*). A diferença do k-vezes para o particionamento simples é que, com objetivo de reduzir possíveis vícios (*bias*) do conjunto de dados, o k-vezes realiza k rodadas nas quais particiona o conjunto de dados. Em cada rodada, o algoritmo particiona o conjunto de dados em $\frac{1}{k}$ das amostras para teste e $1 - \frac{1}{k}$ para treino, treina o classificador e obtém suas métricas de avaliação. Ao final da última rodada, o algoritmo calcula a média das métricas de cada rodada para fornecer a métrica final sem vícios. Assim como no particionamento simples, o melhor resultado de área sob a curva ROC ocorre quando $k = 20$, que corresponde a particionamentos de 95%/5% por rodada. A AUC novamente estabiliza em cerca de 0,95. Assim como no particionamento simples, isto indica que é pouco eficiente utilizar a abordagem *leave-one-out*, um caso

[4]Adaboost é um meta-algoritmo de aprendizado de máquina proposto por Yoav Freund e Robert Schapire, que ganharam em 2003 o Prêmio Gödel por este trabalho.



(a) Efeito no desempenho da classificação para diferentes valores de particionamento.

(b) Efeito no desempenho da classificação devido a diferentes valores de k na validação cruzada.

Figura 5. Ganhos de desempenho devido a utilização de diferentes valores de a) partição treino/teste simples e b) *k* para o algoritmo *k*-vezes.

específico do k-vezes no qual k é igual ao número de amostras do conjunto de dados e apenas uma amostra é utilizada para teste por rodada.

7. Balanceamento de Conjuntos de Dados com Sobre e Subamostragem

A comunidade de aprendizado de máquina concorda com a hipótese de que o desbalanceamento entre classes é um importante obstáculo na indução de classificadores em domínios desbalanceados, afetando o seu desempenho. Os métodos mais simples de balanceamento de dados são a subamostragem aleatória da classe majoritária, que consistem em eliminar dados da classe majoritária selecionados aleatoriamente. Este método tem o inconveniente de descartar dados potencialmente úteis para o processo de indução. Não costuma ser muito usado por este motivo. O método de sobreamostragem aleatória copia amostras da classe minoritária e este procedimento tem o inconveniente de aumentar a possibilidade de o treino apresentar um sobreajuste (*overfitting*). Logo, um classificador pode construir regras que podem parecer acuradas, mas que na verdade cobrem amostras replicadas. Para contornar estas limitações da sobre e subamostragem aleatória alguns métodos heurísticos foram propostos e são descritos a seguir.

O *Synthetic Minority Oversampling Technique* (SMOTE) é um método estatístico de sobreamostragem sintética, ou seja, o SMOTE gera novas amostras da classe minoritária para balancear um conjunto de dados. O SMOTE, proposta por Chawla *et al.*, se baseia no cálculo dos vizinhos mais próximos, medidos pela distância Euclideana entre as amostras no espaço de características, o que permite melhorar o desempenho dos classificadores [Chawla et al. 2002]. O SMOTE não altera as amostras da classe majoritária. O pseudo código do balanceamento de dados através do SMOTE é mostrado no Algoritmo 1. O ADASYN é outra técnica de sobreamostragem que gera amostras da classe minoritária [He et al. 2008] assim como o SMOTE. A diferença da ADASYN para o SMOTE é que a geração de amostras do ADASYN considera a densidade de distribuição das amostras minoritárias para decidir o número de amostras sintéticas a serem geradas para um ponto particular, enquanto que a SMOTE considera pesos uniformes. O pseudo código do balanceamento de dados através do SMOTE é mostrado no Algoritmo 2.

O SMOTE e ADASYN são métodos propostos há alguns anos e muita pesquisa

Algoritmo 1: Balanceamento de dados através do algoritmo SMOTE.

Entradas: $T \rightarrow$ conjunto desbalanceado de dados de treino contendo N amostras do tipo $\{x_i, y_i\}, i \in [1, N], i \in \mathbb{Z}$, onde x_i é um vetor de F características e y_i é a classe do vetor x_i
$k \rightarrow$ número de vizinhos mais próximos a ser utilizado

Saída : $T' \rightarrow$ conjunto balanceado de dados de treino

Inicializar conjunto $T' = T$;
Inicializar os conjuntos N_{min} e N_{maj}, correspondentes às amostras da classe minoritária e majoritária, respectivamente ($T = N_{min} \cup N_{maj}$);
Calcular a taxa de sobreamostragem $\delta = \frac{|N_{maj}| - |N_{min}|}{|N_{min}|}$;
Calcular matriz D de distâncias entre todas as amostras no conjunto N_{min};
para cada $x_i \in N_{min}$ **faça**
    Inicializar o conjunto $A = \{\}$;
    Obter $x_j \in N_{min}, j = 1, 2, .., k$ vizinhos mais próximos a x_i na matriz D;
    Selecionar aleatoriamente e com reposição δ vizinhos da lista de vizinhos mais próximos e adicionar em A;
    para cada $x_j \in A$ **faça**
        $x' = x_i + random(0, 1)|x_i - x_j|$, onde $random(0, 1)$ representa um número aleatório entre 0 e 1;
        $T' = T' \cup x'$;
    fim
fim
Retornar T';

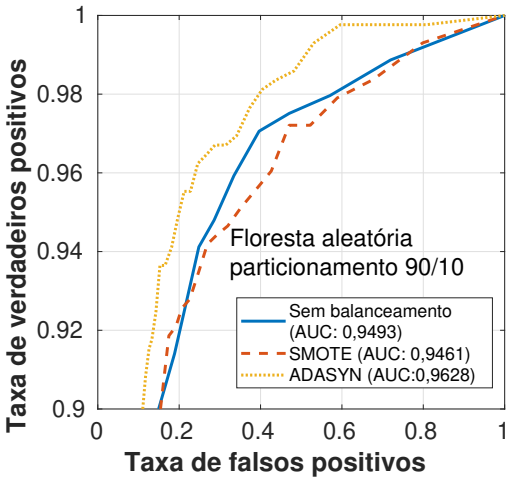
Algoritmo 2: Balanceamento através do algoritmo adaptativo ADASYN.

Entradas: $T \rightarrow$ conjunto desbalanceado de dados de treino contendo N amostras do tipo $\{x_i, y_i\}, i \in [1, N], i \in \mathbb{Z}$, onde x_i é um vetor de F características e y_i é a classe do vetor x_i;
$k \rightarrow$ número de vizinhos mais próximos a ser utilizado

Saída : $T' \rightarrow$ conjunto balanceado de dados de treino

Inicializar conjunto $T' = T$;
Inicializar os conjuntos N_{min} e N_{maj}, correspondentes às amostras da classe minoritária e majoritária, respectivamente ($T = N_{min} \cup N_{maj}$);
Calcular total de amostras a serem geradas $g = |N_{maj}| - |N_{min}|$;
Calcular matriz D de distâncias entre todas as amostras no conjunto N_{min};
para cada $x_i \in N_{min}$ **faça**
    Inicializar o conjunto $A = \{\}$;
    Obter o conjunto V contendo os $x_j \in \{N_{min} \setminus x_i \cup N_{max}\}, j = 1, 2, .., k$ vizinhos mais próximos a x_i baseado na matriz D;
    Calcular a quantidade de vizinhos pertencentes à classe majoritária Δ_i;
    Calcular a razão de vizinhos da classe majoritária $r_i = \frac{\Delta_i}{k}$;
    Calcular a razão normalizada $\hat{r}_i = \frac{r_i}{\sum_{x_n \in N_{min}} r_n}$;
    Calcular a quantidade $g_i = \hat{r}_i.g$ de amostras a serem geradas a partir de x_i;
    Selecionar aleatoriamente e com reposição g_i vizinhos da lista de vizinhos mais próximos e adicionar em A;
    para cada $x_j \in A$ **faça**
        $x' = x_i + random(0, 1)|x_i - x_j|$, onde $random(0, 1)$ representa um número aleatório entre 0 e 1 ;
        $T' = T' \cup x'$;
    fim
fim
Retornar T';

tem sido realizada desde então. Por exemplo, Fernandez *et al.* mapeiam mais de 100 variações conhecidas de SMOTE [Fernández et al. 2018]. Entretanto, nenhuma delas reconhecidamente provê melhores resultados que o SMOTE nem compartilham da mesma notoriedade. No caso deste trabalho, o objetivo é avaliar se e quanto as técnicas de balanceamento podem ajudar no domínio de aplicação. Portanto, acreditamos haver uma contribuição mais clara provendo resultados de um método amplamente conhecido que podem ser facilmente replicados pela comunidade.



(a) Efeito no desempenho de classificação para diferentes técnicas de sobreamostragem.

Figura 6. Desempenho de classificação por tipo de sobreamostragem.

Os resultados ilustrados na Figura 6(a) mostram que a utilização do algoritmo de sobreamostragem ADASYN melhora significativamente o desempenho do classificador de floresta aleatória, chegando a obter uma área sob a curva de 0,9628. Por outro lado, o SMOTE não permitiu ganho significativo. Os resultados dos demais classificadores foram omitidos por restrições de espaço. Os bons resultados obtidos por alguns classificadores e o mau resultado do SMOTE significam que o desbalanceamento do conjunto de dados é um problema, mas não é o único problema da detecção de lavagem de dinheiro.

8. Conclusão

Este artigo focou na detecção de transações de lavagem de dinheiro na rede Bitcoin, objetivando melhorar o desempenho de um classificador usando aprendizado de máquina. Para a obtenção das características a serem usadas na classificação, evitou-se a engenharia de características manual que é dependente de um especialista, e optou-se pelo algoritmo node2vec, que é um método automático e otimizado. O artigo avaliou como o desbalanceamento do conjunto de dados afeta o desempenho do classificador. Os resultados dos classificadores mostram três diferentes patamares de desempenho: os algoritmos Bayes ingênuo, SVM, MLP e regressão logística, com desempenho perto do aleatório, a árvore de decisão, com bom desempenho, e os algoritmos de redes neurais probabilísticas e os algoritmos de conjunto como os melhores resultados. Os três melhores classificadores foram a floresta aleatória, as árvores reforçadas por gradiente e a rede neural probabilística. Um resultado inesperado é a queda de desempenho apresentada pelo algoritmo de árvore de decisão com o AdaBoost.

O experimento de particionamento do conjunto de dados mostrou que há uma melhora do desempenho de classificação ao se aumentar o tamanho das partições para o treino, sendo então a partição 90/10 a de melhor resultado. Acredita-se que a maior partição para treino aumente o número de amostras da classe minoritária ajudando o aprendizado e melhorando a classificação. O experimento com validação cruzada k-vezes (*k-fold*) confirma a observação acima e mostra que os ganhos de desempenho diminuem conforme aumenta a diferença entre os conjuntos de treino e teste, gerando um teto para o desempenho máximo do classificador. O resultado indica que estratégias de particionamentos com diferenças muito grandes entre treino e teste, como o *leave-one-out*, são pouco eficientes para este caso.

Os trabalhos futuros focarão na utilização de métodos mistos de balanceamento de dados, como o SMOTE+Tomek Link e SMOTE+CNN propostos por Batista *et al.*, que combinam as técnica de do SMOTE com as técnicas de subamostragem conhecidas por *Tomek links* e *Condensed Nearest Neighbor* (CNN) [Batista et al. 2004]. Outra vertente importante é a substituição do node2vec pelo algoritmo transdutivo GCN, e o uso de algoritmos de aprendizagem indutivos, como o GraphSage e o Evolve-GCN.

Referências

[Batista et al. 2004] Batista, G. E. A. P. A., Prati, R. C. e Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. Em *ACM SIGKDD Explorations Newsletter - Special issue on learning from imbalanced datasets*, volume 6, páginas 20–29, New York, NY, USA. ACM.

[Blockchain.com 2019] Blockchain.com (2019). Confirmed transactions per day. Disponível em https://www.blockchain.com/charts/n-transactions. Acessado em 19 de dezembro de 2019.

[Chan e Stolfo 1998] Chan, P. K. e Stolfo, S. J. (1998). Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. Em *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, KDD'98, páginas 164–168. AAAI Press.

[Chawla et al. 2002] Chawla, N. V., Bowyer, K. W., Hall, L. O. e Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.

[Chawla et al. 2004] Chawla, N. V., Japkowicz, N. e Kol, A. (2004). Editorial: Special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations Newsletter - Special issue on learning from imbalanced datasets*, 6(1):20–29.

[CoinMarketCap 2019] CoinMarketCap (2019). Top 100 cryptocurrencies by market capitalization. Disponível em https://coinmarketcap.com/pt-br/all/views/all/. Acessado em 19 de dezembro de 2019.

[Demirguc-Kunt et al. 2018] Demirguc-Kunt, A., Klapper, L., Singer, D., Ansar, S. e Hess, J. R. (2018). The global findex database 2017: Measuring financial inclusion and the fintech revolution. Disponível em http://documents.worldbank.org/curated/en/332881525873182837/The-Global-Findex-Database-2017-Measuring-Financial-Inclusion-and-the-Fintech-Revolution. Acessado em dezembro de 2019.

[Elliptic Inc. 2019] Elliptic Inc. (2019). Bitcoin money laundering: How criminals use crypto. Disponível em https://www.elliptic.co/our-thinking/bitcoin-money-laundering. Acessado em 19 de dezembro de 2019.

[Fernández et al. 2018] Fernández, A., Garcia, S., Herrera, F. e Chawla, N. V. (2018). Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of artificial intelligence research*, 61:863–905.

[Foley et al. 2019] Foley, S., Karlsen, J. R. e Putniņš, T. J. (2019). Sex, drugs, and bitcoin: How much illegal activity is financed through cryptocurrencies? *The Review of Financial Studies*, 32(5):1798–1853.

[Freund e Schapire 1999] Freund, Y. e Schapire, R. E. (1999). A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780.

[Goyal e Ferrara 2018] Goyal, P. e Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94.

[Grover e Leskovec 2016] Grover, A. e Leskovec, J. (2016). node2vec: Scalable feature learning for networks. Em *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 855–864. ACM.

[He et al. 2008] He, H., Bai, Y., Garcia, E. A. e Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. *IEEE International Joint Conference on Neural Networks - IEEE World Congress on Computational Intelligence*, páginas 1322–1328.

[Kubat et al. 1998] Kubat, M., Holte, R. C. e Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30(2):195–215.

[LeCun et al. 2015] LeCun, Y., Bengio, Y. e Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

[Nakamoto 2008] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Acessado em 19 de dezembro de 2019.

[Saito e Rehmsmeier 2015] Saito, T. e Rehmsmeier, M. (2015). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3).

[Spilotro 2019] Spilotro, T. (2019). Coinbase crypto milestone: Amasses 30M users, 5M in last 10 months. Disponível em https://www.newsbtc.com/2019/07/23/coinbase-crypto-bitcoin-users-adoption/. Acessado em 19 de dezembro de 2019.

[Tan et al. 2019] Tan, X., Su, S., Huang, Z., Guo, X., Zuo, Z., Sun, X. e Li, L. (2019). Wireless sensor networks intrusion detection based on smote and the random forest algorithm. *Sensors*.

[United Nations Office on Drugs and Crime 2019] United Nations Office on Drugs and Crime (2019). Money-laundering and globalization. Disponível em https://www.unodc.org/unodc/en/money-laundering/globalization.html. Acessado em 19 de dezembro de 2019.

[Weber et al. 2019] Weber, M., Domeniconi, G., Chen, J., Weidele, D. K. I., Bellei, C., Robinson, T. e Leiserson, C. E. (2019). Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics.

[Weiss 2004] Weiss, G. M. (2004). Mining with rarity: A unifying framework. *ACM SIGKDD Explorations Newsletter - Special issue on learning from imbalanced datasets*, 6(1):7–19.

[Weiss e Hirsh 1998] Weiss, G. M. e Hirsh, H. (1998). Learning to predict rare events in event sequences. Em *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, KDD'98, páginas 359–363. AAAI Press.

[World Bank 2019] World Bank (2019). World development indicators: GDP (current US$). Disponível em https://data.worldbank.org/indicator/NY.GDP.MKTP.CD?year_high_desc=true. Acessado em 19 de dezembro de 2019.