AXEL VALENE ROCHA, GUSTAVO HORNIG DE MEIRA

UTILIZAÇÃO DE MODELOS DE MACHINE LEARNING PARA A IDENTIFICAÇÃO DE LAVAGEM DE DINHEIRO EM TRANSAÇÕES FINANCEIRAS

(*versão pré-defesa, compilada em 16 de dezembro de 2019*)

Trabalho apresentado como requisito parcial à conclusão do Curso de Ciência da Computação - Bacharelado, setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Eduardo Todt.

CURITIBA PR

2019

RESUMO

O crime da lavagem de dinheiro consiste na prática de tornar um capital financeiro adquirido de forma ilícita em aparentemente lícito. Este crime ganhou popularidade nos anos 80 e assola a economia global, financiando crimes como tráfico de drogas e terrorismo. Este trabalho de graduação tem como objetivo apresentar modelos de *machine learning* para a detecção de lavagem de dinheiro. Os modelos estudados são: Árvore de Decisão, XGBoost e Multilayer Perceptron. Estes foram treinados com um *dataset* sintético, visto a dificuldade de se obter dados reais desta natureza. Os resultados obtidos foram mensurados utilizando métricas como *precision*, *recall*, *f1-score* e AUPRC. As métricas demonstraram um alto valor de eficiência dos modelos, tendo como menor *f1-score* 84%, provando a capacidade de implementação em um cenário real. Por fim, são discutidas as comparações entre os algoritmos e possíveis melhorias futuras.

Palavras-chave: Aprendizado de Máquina. Lavagem de Dinheiro. Árvore de Decisão. XGBoost. Multilayer Perceptron

ABSTRACT

The crime of money laundering is the practice of turning illicitly acquired financial capital into seemingly lawful. This crime gained popularity in the 80's and ravages the global economy by financing crimes such as drug trafficking and terrorism. This thesis aims to present machine learning models to detect financial transaction fraud. The models presented are: Decision Tree, XGBoost and Multilayer Perceptron. These were trained with a synthetic dataset, given the difficulty of obtaining real data of this nature. The results obtained were measured using metrics such as precision, recall, f1-score and AUPRC. These demonstrated a high efficiency value of the models, as the smallest f1-score being 84%, proving the ability to implement in a real scenario. Finally, comparisons between the algorithms and possible future improvements to assist in fraud detection are discussed.

Keywords: Machine Learning. Money Laundering. Decision Tree. XGBoost. Multilayer Perceptron.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

AML	Anti-Money Laundering (Contra Lavagem de Dinheiro)
AUC	Area Under the Curve (Área embaixo da curva)
AUPRC	Area Under the Precision-Recall Curve
AUROC	Area Under the Receiver Operating Characteristics
DL	Deep Learning (Aprendizagem Profunda)
FATF	Financial Action Task Force
FPR	False Positive Rate(Taxa de false positive)
GCN	Graph Convolutional Network (Rede Neural Convolucional)
HVDM	Heterogeneous Value Difference Metric
ML	Machine Learning (Aprendizado de Máquina)
MLP	Multilayer Perceptron
PPV	Predicted Positive Values (Predição de valores positivos)
PRC	Precision-Recall Curve
ROC	Receiver Operating Characteristics
SVM	Support-Vector Machine
TNR	True Negative Rate (Taxa de True negative)
TPR	True Positive Rate (Taxa de True positive)
UIF	Unidade de Inteligência Financeira

SUMÁRIO

1 INTRODUÇÃO

A lavagem de dinheiro consiste em técnicas e procedimentos utilizados para ocultar bens financeiros e patrimoniais que foram obtidos ilegalmente, de tal forma que este capital possa ser utilizado em outras transações consideradas lícitas. Este crime ganhou popularidade na década de 80 quando os Estados Unidos criou a sua primeira legislação a respeito deste crime, decorrente da preocupação em criminalizar o tráfico de entorpecentes e o dinheiro provindo desta atividade, que na época, já eram estimados em bilhões de dólares (Hülsse, 2007).

A identificação de casos de fraude é extremamente importante, pois permite expor organizações criminosas, que se utilizam de transações financeiras para ocultar a origem do dinheiro ilícito, provinda do tráfico de drogas, corrupção e terrorismo. Em 1989, o grupo Financial Action Task Force (FATF) foi criado pelo G7 (grupo composto por Canadá, França, Alemanha, Itália, Japão, Reino Unido e Estados Unidos), com o intuito de desenvolver políticas contra a lavagem de dinheiro. O FATF define que este crime acontece em três etapas: a primeira é a inserção do dinheiro ilícito em um sistema financeiro, em seguida, cria-se sistemas de camadas para dispersar o dinheiro e impossibilitar a rastreabilidade, como por exemplo várias transações de valores menores utilizando-se de diferentes contas e bancos, e por fim, a legitimação do dinheiro para uso posterior (Levi, 2002).

No Brasil, o órgão responsável pelas políticas contra lavagem de dinheiro é a Unidade de Inteligência Financeira (UIF), o qual institui regras rígidas para as instituições financeiras na tentativa de evitar este crime. No entanto, na economia brasileira, estima-se que 17 bilhões de dólares por ano provêm de uma origem ilícita (Araújo, 2009).

Atualmente, há uma grande preocupação causada pelo impacto mundial gerado por este dinheiro fraudulento. Estima-se que anualmente são lavados entre 400 bilhões e 2,85 trilhões de dólares no mundo (Schneider e Windischbauer, 2008), e apenas 1% destes fundos ilegais são identificados e confiscados. Isso decorre do fato de que os conjuntos de dados financeiros são extremamente grandes e complexos, além de formas de anonimato que dificultam ainda mais a identificação destes problemas (Europol, 2017). Assim, existem poucas técnicas eficazes sendo a maioria medidas provisórias e limitadas que uma instituição pode adotar, que não garantem a ineficácia desta atividade criminosa.

Dessa maneira, este trabalho de graduação tem como objetivo apresentar uma análise na detecção de casos nos quais há lavagem de dinheiro em transações financeiras, utilizando algoritmos de *machine learning* que, dado um *dataset* (conjunto de dados) contendo inúmeras transações financeiras, leva em consideração um conjunto de características que são julgadas suspeitas. Assim, o modelo treinado é capaz de identificar se houve alguma transação fraudulenta que se encaixe nos critérios considerados.

No entanto, há uma grande dificuldade em acumular dados suficientes que possibilitem o treinamento do algoritmo, visto que as empresas ou instituições não compartilham seus dados, já que estes são sigilosos e divulga-los de maneira leviana poderia causar muitos problemas de privacidade. Além disso, criminosos poderiam utilizar dessa informação para aprimorar suas práticas ilegais. Por este motivo, utilizamos uma ferramenta que gera dados sintéticos, o Paysim (Lopez-Rojas et al., 2016).

No capítulo 2 deste trabalho, apresenta a fundamentação teórica, onde são definidos os algoritmos de *machine learning* responsáveis pelos resultados presentes neste trabalho de graduação e as métricas utilizadas na análise dos resultados. O capítulo 3 aborda os trabalhos relacionados que utilizam técnicas de *Anti-Money Laundering*(AML) para obter uma possível solução para o crime de lavagem de dinheiro. O capítulo 4 contém a nossa proposta de solução para o problema, explicando a abordagem escolhida. O capítulo 5 detalha os meios utilizados para alcançar os resultados obtidos pelos algoritmos, isto é os parâmetros relevantes e também uma explicação a respeito das métricas utilizadas para a análise. No capítulo 6 são apresentados os resultados obtidos pelos algoritmos, bem como os pontos fortes e fracos do mesmo. No capítulo 7 há uma comparação de resultados entre os algoritmos escolhidos, que acarreta em uma análise feita sobre os mesmos. No capítulo 8 possíveis melhorias para trabalhos futuros são discutidas. E finalmente, no capitulo 9, são apresentadas conclusões.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentadas as métricas responsáveis pela análise e os modelos de *machine learning* utilizados para a classificação dos conjuntos de dados como fraudulentos ou não fraudulentos.

2.1 MÉTRICAS

Precisamos estabelecer padrões e métricas para uma análise justa dos desempenhos dos modelos apresentados. Para isso utilizaremos os conceitos de Regra de Classificação, Matriz de Confusão e Curvas de Performance.

2.1.1 Regras de Classificação

Considerando um *dataset* com transações financeiras, cada dado deste conjunto é considerado fraude ou não fraude. O algoritmo irá determinar, baseado em sua aprendizagem, a sua interpretação para cada dado, julgando-o como fraudulento ou não. Assim, teremos quatro tipos de classificação: *True-Positive*, *True-Negative*, *False-Positive*, *False-Negative*.

- *True-Positive*(T_p) significa que o dado era considerado fraudulento e o algoritmo teve a mesma observação.

- *True-Negative*(T_n) significa que o dado era considerado não fraudulento e o algoritmo considerou como não fraude.

- *False-Positive*(F_p) significa que o dado era não fraudulento, mas o algoritmo considerou como fraude.

- *False-Negative*(F_n) significa que o dado era fraudulento, mas o algoritmo considerou como não fraude.

2.1.2 Matriz de Confusão

Tendo o conceito de Regras de Classificação em mente, iremos trabalhar com a Matriz de Confusão, a qual é uma tabela que usaremos para representar as regras. Por definição uma matriz de confusão C é tal que C_{ij} é igual ao número de indivíduos conhecidos no grupo i, previstos estarem no grupo j. O modelo utilizado em nossas métricas será o mostrado na imagem 2.1.

Outras métricas importantes são: *Precision* (precisão) ou *Predicted Positive Values* (PPV), *Recall* ou *True Positive Rate*(TPR), *Specificity* ou *True Negative Rate*(TNR), *Fall-out* ou *False Positive Rate* (FPR), *F1-score*, e *Accuracy*.

Precision ou PPV: é a habilidade intuitiva de não classificar o que é positivo, como negativo. (P) é definida da seguinte forma:

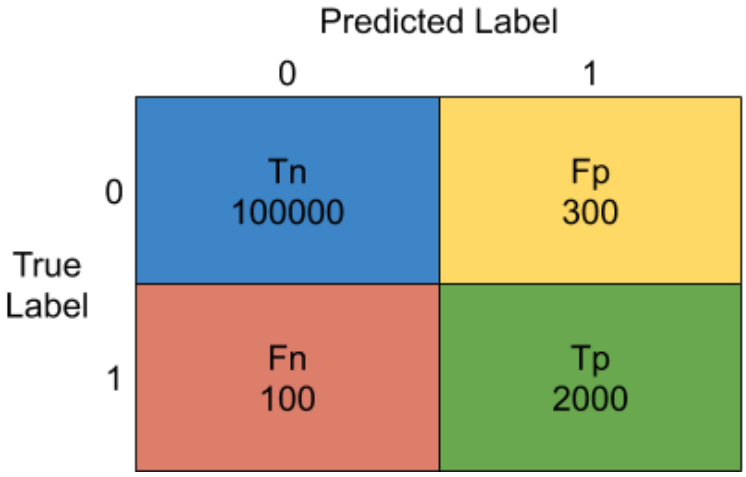

Figura 2.1: Matriz de confusão de exemplo, onde a quantidade de *True Negative* foi 100000, *False Positive* 300, *False Negative* 100 e *True Positive* 2000. Fonte: autores.

$$P = T_p/(T_p + F_p)$$

Recall ou TPR: é a habilidade intuitiva do classificador de encontrar todos as amostras positivas. (R) é definido da seguinte forma:

$$R = T_p/(T_p + F_n)$$

Specificity ou TNR: mede a proporção em que as classes negativas foram corretamente preditas como negativas. (E) é definido da seguinte forma:

$$E = T_n/(T_n + F_p)$$

Fall-out ou FPR: é a taxa que mede a quantidade de classes negativas identificadas incorretamente como positivas. (Fo) é definido da seguinte forma:

$$Fo = F_p/(F_p + T_n)$$

F1-score: é uma métrica simplificada que envolve a precisão e o *recall*, isto é, (F1) é definido como a media harmônica de (P) e (R), dada pela seguinte fórmula:

$$F1 = 2 * (P * R)/(P + R)$$

Accuracy: é a métrica que avalia quantos indivíduos foram classificados corretamente levando em conta todas as classificações. (AC) é definida da seguinte forma:

$$AC = (T_p) + (T_n)/((T_p) + (T_n) + (F_p) + (F_n))$$

2.1.3 Curvas de Performance

São várias as maneiras para se medir a performance de um modelo de *machine learning*, as mais conhecidas são *Area Under the Receiver Operating Characteristics* (AUROC) e *Area Under the Precision-Recall Curve*. AUROC é construída analisando a área da curva *Receiver Operating Characteristics(ROC)*. ROC é uma curva de probabilidade que descreverá a chance do modelo, dado uma entrada, de decidir entre as possíveis classes: fraudulento ou não fraudulento. Esta curva é criada relacionando o *Recall* com o *Fall-out*.

A *Area Under the Curve* (AUC) representa o grau ou a medida de separação do modelo. Com isso seríamos capazes de observar a capacidade do algoritmo de distinguir entre as classes. Os valores da AUC variam de zero a um, assim quanto maior o valor, melhor é a previsão de T_p e T_n. A curva ROC é criada relacionando a taxa de T_p sobre a taxa de F_p, como mostra a figura 2.2.

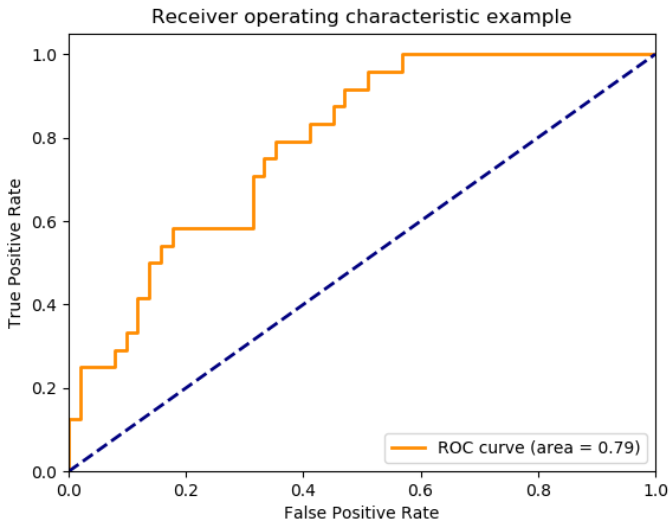

Figura 2.2: Esta curva ROC representa a correlação de F_p e T_p de um determinado modelo, onde ilustra a relação de crescimento da taxa de T_p conforme a taxa de F_p aumenta. Obteve-se o valor 0,79 como área embaixo da curva. Fonte: Pedregosa et al. (2011).

Quanto mais próximo de zero o valor do AUROC, menor é a sua capacidade de classificação. Ao aproximar do valor um, o modelo consegue distinguir entre não fraude e fraude. No entanto, quando está próximo de 0,5 significa que ele consegue na metade das vezes classificar corretamente ou apenas classificar os inputs como não fraude em um cenário onde não há quantidade suficiente de dados para o modelo aprender o que é fraude. Dessa maneira, o modelo não será capaz de identificar casos fraudulentos, demonstrando uma falsa ideia de assertividade. Assim, o AUROC não é uma métrica suficiente para a nossa análise, sendo necessária uma outra abordagem.

No capítulo 4.3.1 o *dataset* utilizado neste trabalho de graduação é apresentado e foi evidenciado a sua característica de desbalanceamento, já que a taxa de ocorrência de fraudes constitui aproximadamente 0,13% do *dataset*, e por esta razão utilizaremos a métrica AUPRC.

Segundo Saito e Rehmsmeier (2015) a curva *Precision-Recall* nos diz mais sobre a eficiência do modelo do que a curva *Receiver Operating Characteristic*. Isso se deve ao fato de que AUPRC não considera a taxa de *True Negative*, desta maneira, não é influenciada pela imensa quantidade de T_n.

A *Precision-Recall Curve* (PRC) esboça a relação entre o aumento da taxa de precisão em detrimento da taxa do *recall* e vice-versa. Seus valores nas abcissas e ordenadas variam de zero a um. Esta curva, diferente da ROC, começa no canto esquerdo de cima do gráfico e termina no canto direto de baixo, como mostra a figura 2.3.

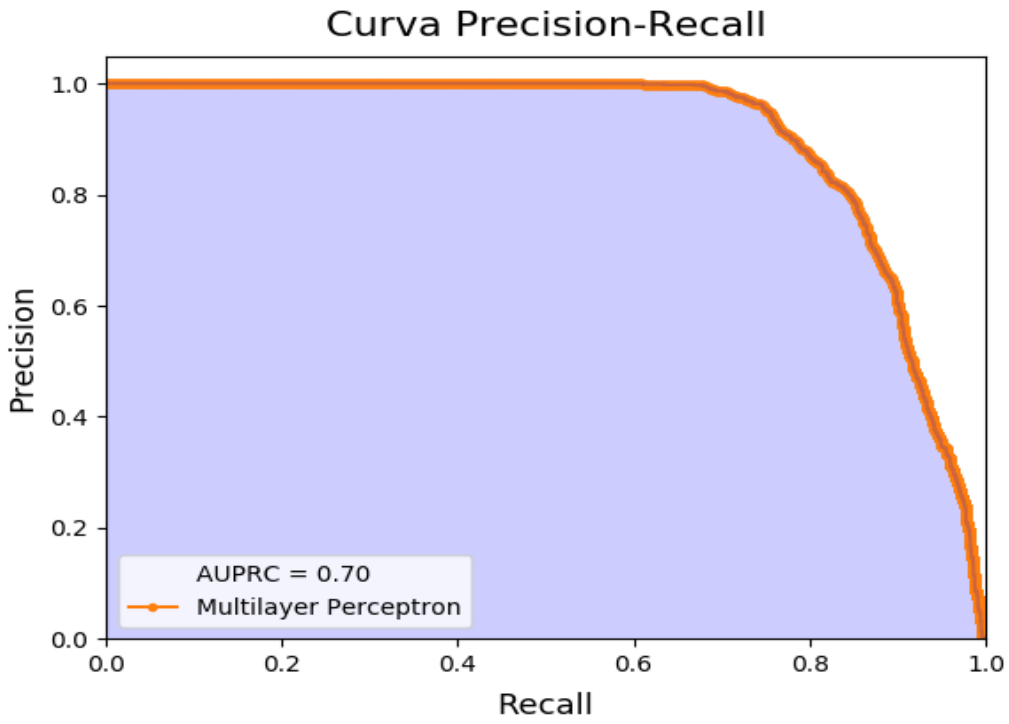

Figura 2.3: Curva de Precision-Recall de um modelo Multilayer Perceptron, ilustrando a relação de compensação entre o ganho de *recall* em detrimento de *precision*, onde se obteve um valor AUPRC de 0,7. Fonte: autores.

O fato da curva começar no ponto 1 em precisão e 0 em *recall*, ilustra o caso onde o modelo não aprendeu com o *dataset* e por isso predisse que somente existem casos onde não há fraude. Consequentemente gerou 100% de T_n e 100% de F_n. Deste modo, conforme o modelo foi aprendendo a classificar corretamente os casos onde houve fraude, este equivoca-se ao classificar alguns casos que não eram fraude, como fraudulentos.

Analogamente, quando temos *precision* em 0 e *recall* em 1, mostra que o algoritmo não aprendeu nada, porém desta vez ele não consegue identificar quando não é fraude, gerando 100% de T_p e 100% de F_p.

Intuitivamente, o objetivo é encontrar o ponto máximo da concavidade formada pela curva. Porém, esse objetivo pode mudar quando estivermos falando de aplicações práticas, como por exemplo, uma empresa não se importa em identificar todos os casos fraudulentos, mas querem o menor número de casos que não era fraude seja dito como fraude, assim, focaríamos

em *precision*. Outro caso é quando a empresa não se importa de identificar pessoas como fraudulentas quando não são, mas deseja diminuir ao máximo a quantidade de pessoas usando-a para cometer crimes.

A interpretação da AUPRC é mais simples e não possui vieses como AUROC. Seu valor também varia de zero a um, e quanto maior, melhor. A área abaixo da curva é calculada a partir do somatório da média ponderada alcançada a cada ponto (x, y) do gráfico, com o incremento do *recall* do ponto anterior usado como peso. A fórmula é a seguinte:

$$AUPRC = \sum_{n} (R_n - R_{n-1}) P_n$$

Onde R_n e P_n são o *precision* e o *recall* no N-ésimo ponto (x, y) do gráfico. Vale ressaltar que, pensando em evitar um resultado muito otimista, o cálculo desta área não é interpolado, isto é, não aproxima valores discretos No nosso caso significa que não são criados pontos novos além dos existentes a fim de se ter uma aproximação mais suave.

2.2 ÁRVORE DE DECISÃO

O algoritmo de árvore de decisão é considerado um modelo eficiente para aprendizagem de máquina supervisionada, já que consegue separar os dados de forma precisa e homogênea em subconjuntos, tornando possível a distinção e classificação dos dados (Song e Ying, 2015). Além disso, o modelo é comparável com o pensamento humano, o que traz uma facilidade na interpretação dos dados e na visualização do treinamento do modelo Sanjeevi (2017).

Assim, este foi um algoritmo utilizado para a classificação de dados neste trabalho. Na figura 2.4 podemos visualizar um grafo conexo que representa a estrutura de uma árvore aplicada ao *dataset* do Paysim. No início do processamento, o conjunto de dados está presente na raiz da árvore. A seguir, há uma divisão da raiz de tal forma que serão criados dois novos vértices que irão conter um sub-conjunto de dados. Neste momento, a raiz estará homogênea já que apresenta uma única informação relevante. Após esta etapa, o modelo irá continuar a divisão dos nós que apresentarem um sub-conjunto de dados até uma homogeneidade predeterminada ou até que um critério de parada seja satisfeito. O resultado será uma árvore completa com as informações necessárias para determinar se uma transação é fraudulenta ou não (Brid, 2018).

Para as folhas, ou seja, os vértices de grau um, não há como dividir em novos subconjuntos sem adicionar novos dados no conjunto, portanto não há mais divisões a partir deste ponto. As arestas representam fluxos de decisão, onde “se” e “se não” representam as bifurcações que o algoritmo pode fazer.

Após o treinamento do modelo, a árvore estará completa e inicia-se o processo de decisão recebendo as entradas. Para cada vértice é analisado a condição deste com o dado, caso as condições sejam verdadeiras para a entrada que está sendo observada, continua-se percorrendo na condição verdadeira da árvore. Caso contrário, irá para a condição negativa. Por fim, após

percorrer o caminho completo o destino é uma folha que contém o resultado, ou seja, uma conclusão a respeito do dado observado categorizando o dado como fraude ou não fraude.

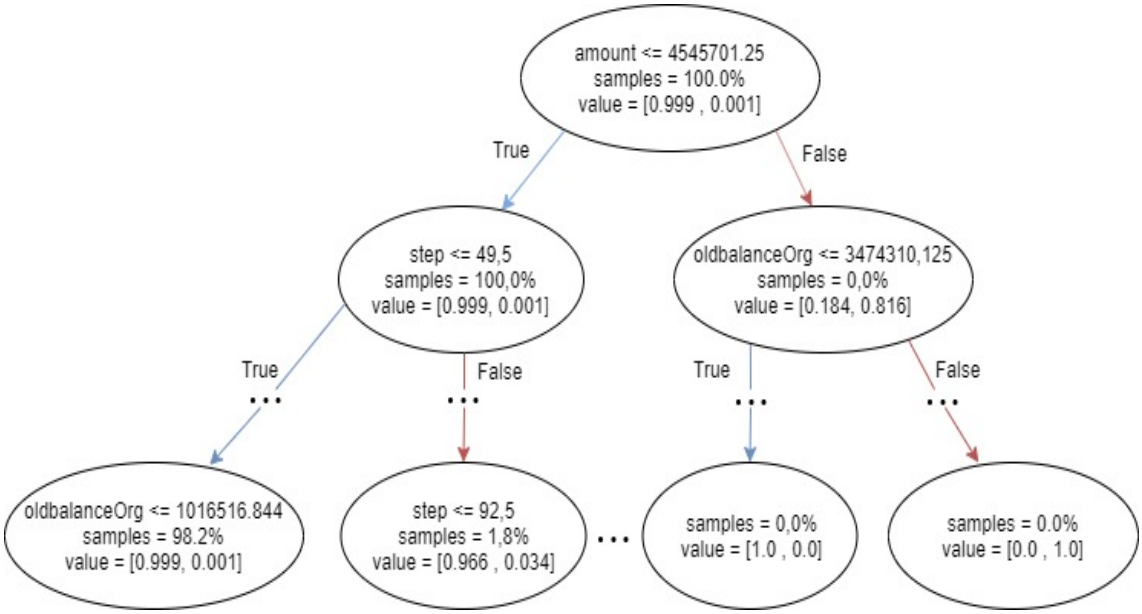

Figura 2.4: Árvore completa criada após o aprendizado do modelo de Árvore de Decisão utilizando todos os dados. Os três pontos expressam uma continuação finita da árvore. Fonte: autores.

2.3 XGBOOST

Tree boosting é uma técnica amplamente utilizada em diferentes áreas como, por exemplo, na medicina (Pezoulas et al., 2019) e predição de valores de criptomoedas (Li et al., 2019), com o intuito de aumentar a eficiência de árvores de decisão em vários casos de *machine learning* (Rushin et al., 2017). Um exemplo de algoritmo conhecido é o *Random Forest* (Liu et al., 2015), que utiliza uma técnica similar ao XGBoost, porém menos eficiente. Assim, como método de comparação, o XGBoost será usado neste trabalho de graduação para a classificação do *dataset*.

Nesta técnica de *tree boosting*, é utilizada a mesma ideia na criação de uma árvore de decisão, onde um conjunto de dados é representado por cada vértice em uma árvore, cada qual dividido da maneira mais homogênea possível e com um conjunto de arestas representando as decisões a serem tomadas pelo algoritmo. Porém, no algoritmo do XGBoost é criado um conjunto de árvores de decisão, cujo tamanho pode ser especificado.

A primeira árvore de decisão criada, que pode ser observada na figura 2.5, é uma árvore de pouca profundidade e de baixa eficiência de classificação e que corresponde a apenas alguns parâmetros do *dataset*, não levando em consideração todas as colunas de dados. A seguir, é criada a segunda árvore que é constituída pelos dados de novas colunas presentes no *dataset* e de colunas que foram utilizadas na primeira árvore. Esta árvore resultante tem propriedades parecidas com a primeira (pouca profundidade e baixa eficiência de classificação), porém, com

observações novas já que agregou informações diferentes do *dataset*. Este processo de criação é feito seguindo as mesmas regras até que ocorra uma condição de parada, neste caso, o tamanho do conjunto de árvores. As figuras 2.6 e 2.7 ilustram a quinta e a décima árvore gerada pelo algoritmo, respectivamente.

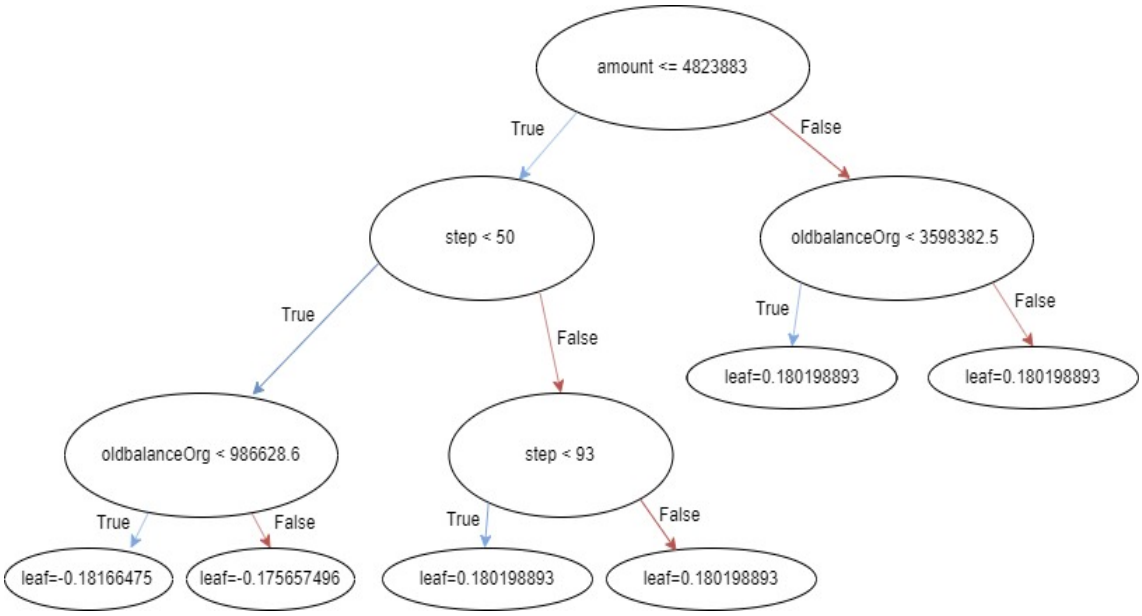

Figura 2.5: A primeira árvore de decisão criada pelo XGBoost, levando em consideração os parâmetros: amount, step, oldBalanceOrig. Fonte: autores.

Por fim, quando o algoritmo termina o seu treinamento, inicia-se a etapa de validação de um dado classificando-o como fraudulento ou não. Para cada árvore no conjunto, o algoritmo percorre as arestas com as tomadas de decisões baseadas nas propriedades de cada vértice, obtendo N resultados, sendo N o tamanho do conjunto de árvores. Para tomar a decisão final sobre um certo dado, o XGBoost faz uma média com os resultados de cada árvore do conjunto, então, se um houver um número maior de árvores de decisão atestando que um dado é fraude, este dado será considerado fraudulento.

2.4 MULTILAYER PERCEPTRON

Perceptron é um classificador linear, isto é, um algoritmo que classifica uma entrada separando-a em duas categorias com uma função linear (Stephen, 1990). Essa entrada é tipicamente um vetor de features multiplicada por pesos e adicionado a um bias. O Perceptron produz uma única saída baseada em vários valores de entrada, formando uma combinação linear usando os pesos de cada entrada.

O poder das redes neurais vem de sua capacidade de aprender a representação dos dados de treinamento e de como melhor relacioná-los à variável de saída que deseja-se prever. Nesse sentido, as redes neurais aprendem a fazer um mapeamento. Matematicamente, elas são capazes de aprender qualquer função de mapeamento (Hornik et al., 1989; Cybenko, 1989). Na

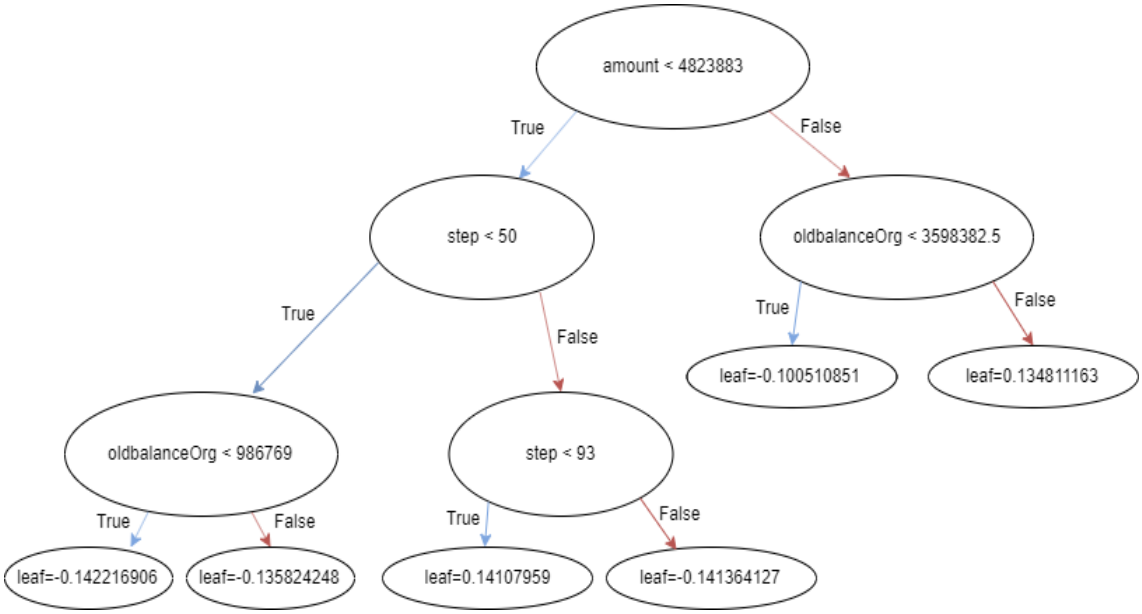

Figura 2.6: A quinta árvore de decisão criada pelo XGBoost, levando em consideração os parâmetros: amount, step, oldBalanceOrig. Fonte: autores.

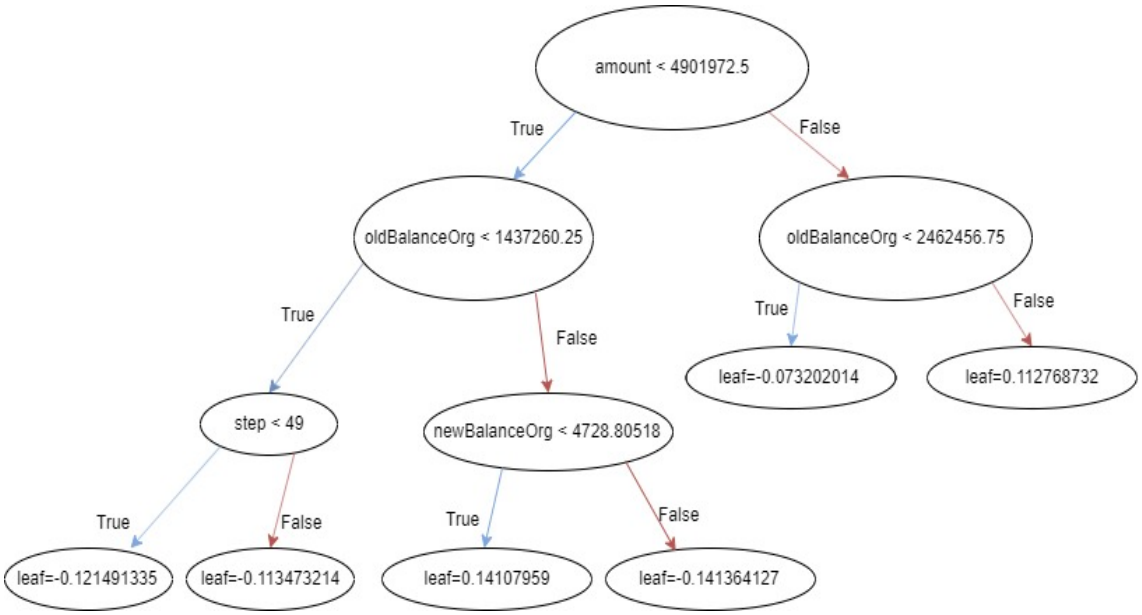

Figura 2.7: A décima árvore de decisão criada pelo XGBoost, levando em consideração os parâmetros das árvores anteriores: amount, step, oldBalanceOrig e com o parâmetro novo: newBalanceOrig. Fonte: autores.

área de *machine learning* um algoritmo que é amplamente utilizado é o Backpropagation para treinamento de redes neurais com conexões que não formam um ciclo entre elas. Este algoritmo calcula de maneira eficiente o gradiente da função de perda, respeitando os pesos existentes na rede para cada conjunto *input-output*, iterando uma camada por vez a partir da última para evitar cálculos redundantes em termos intermediários na rede.

Deste modo, o MLP foi criada sendo uma rede neural artificial a qual é composta por mais de um *Perceptron*, que por sua vez são compostos por uma entrada que recebem dados, chamada de *Input Layer*, uma saída, chamada de *Output Layer*, a qual tomará uma decisão ou previsão em relação a entrada, e entre essas, existe um numero arbitrário de camadas escondidas, chamadas de *Hidden Layers*, onde ocorrem os cálculos computacionais que decidirão o resultado do algoritmo (Popescu et al., 2009), como mostra a figura 2.8. O aprendizado no MLP ocorre nos *perceptrons*, utilizando do algoritmo de *backpropagation*, quando mudam seus respectivos pesos depois que uma parte da entrada de dados é processada, baseado no tamanho do erro no *output* comparado com o resultado esperado. Deste modo, o modelo vai se aperfeiçoando para prever de forma mais correta os próximos inputs.

De forma mais abrangente, MLP é um algoritmo de aprendizado supervisionado que aprende a função $f(\cdot) : R^m \rightarrow R^o$ treinando em um conjunto de dados, onde m é o numero de dimensões da entrada e o é o numero de dimensões do *output*.

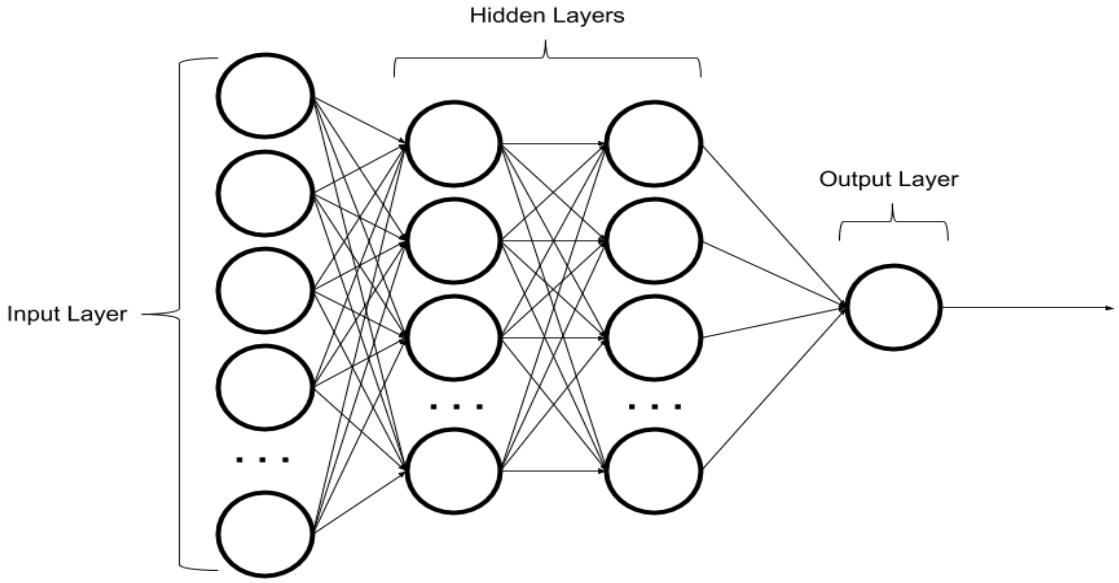

Figura 2.8: A camada de entrada, *Input Layer*, consiste de um conjunto de neurônios representando as features de entrada. Cada neurônio na camada escondida transforma os valores vindos das camadas anteriores utilizando somas ponderadas, seguidas de uma função de ativação não-linear. A camada de saída, *Output Layer*, recebe os valores vindos da última camada do *Hidden Layer* e transforma em um valor de saída, isto é, a previsão em relação ao dado de entrada gerada pelo modelo(Gardner e Dorling, 1998). Os três pontos representam uma continuação finita da quantidade de neurônios e camadas. Fonte: autores.

3 TRABALHOS RELACIONADOS

Neste capítulo é apresentada uma análise de trabalhos relacionados que contêm técnicas utilizadas na detecção de lavagem de dinheiro. Além disso, outros modelos foram citados com o intuito de comparar a efetividade de diferentes algoritmos.

3.1 TÉCNICAS PARA ANTI-MONEY LAUNDERING QUE NÃO UTILIZAM MACHINE LEARNING

Para a detecção de fraudes, Soltani et al. (2016) utilizam um método eficaz que consegue, em um grande volume de dados, detectar fraudulências de forma precisa. O método utilizado consiste em uma estrutura que aplica técnicas para diminuir progressivamente a quantidade de dados e obter um *dataset* pequeno. Por fim, o framework consegue encontrar padrões entre os dados e, por meio de agrupamento, decidir potenciais transações fraudulentas.

Utilizando uma técnica de *Data Mining*, o artigo de Zhang et al. (2003) apresenta um método chamado CORAL, que, dado um conjunto de dados sem relações entre si, consegue criar vínculos entre eles e seguindo um modelo, julgar se uma certa transação é considerada fraudulenta ou não. O artigo consegue aplicar o método em um caso real e demonstrar a sua aplicabilidade.

3.2 DEEP LEARNING APLICADO A ANTI-MONEY LAUNDERING

Deep Learning (DL) é uma técnica que ensina computadores a fazer o que para nós humanos vêm naturalmente: aprender a partir de exemplos, e quanto maiores a quantidade de exemplos melhores se tornam os resultados, como explica MathWorks (2016).

A análise de grafos se mostrou uma importante ferramenta para estudos sobre AML, já que a lavagem de dinheiro envolve relações de fluxo de dinheiro entre entidades, ou seja, pessoas e organizações. Deste modo, Weber et al. (2018) apresentam um estudo implementando DL utilizando uma estrutura de *Graph Convolutional Network*, considerando a complexidade do assunto, o desafio para esta abordagem é o tamanho que o *dataset* poderá chegar, já que em uma situação real uma empresa bancária lida com milhões de transações bancárias todos os dias. Assim, a ferramenta para detecção de fraudes deve ser eficiente e possuir boa performance.

Utilizando a rede neural AutoEncoder, o artigo Paula et al. (2016) propõe uma forma de detectar suspeitas de fraude em exportações, mais precisamente, brasileiras. A metodologia empregada visa, primeiramente, definir os dados a serem minerados e entendê-los para, a seguir, preparar o conjunto final de dados a ser utilizado no método. Por fim, com os dados avaliados e modelados há a etapa de verificação para comparar se os resultados obtidos são semelhantes aos esperados.

3.3 TÉCNICAS DE MACHINE LEARNING APLICADAS A ANTI-MONEY LAUNDERING

Um modelo de ML comumente utilizado é o *Support-vector Machine* (SVM), que em Jun Tang e Jian Yin (2005) foi treinado usando um *dataset* real obtido de um Banco localizado na China. Neste artigo, os autores utilizam o método *Heterogeneous Value Difference Metric* (HVDM), para tornar os dados obtidos em dados heterogêneos sendo possível a utilização do SVM. O artigo conclui que esta técnica utilizada se mostra eficiente na detecção de fraudes e pode ser usada em sistemas do mundo real sem a necessidade de muitas modificações no algoritmo.

A China vem enfrentando um severo desafio de detectar práticas de lavagem de dinheiro em seu país, cujo valor está estimado em 200 bilhões de Renminbi, relatam Wang e Yang (2007). Este artigo aborda a utilização de Árvores de Decisão para identificar os casos onde há maior risco de ocorrer as fraudes. Em seus experimentos e avaliações, aplicando o modelo no *dataset* exposto no artigo, é demonstrado que o algoritmo foi capaz de encontrar as características mais significativas, e deste modo, podendo identificar quais indivíduos eram mais prováveis de realizar operações de lavagem de dinheiro.

No artigo Zhang e Trubey (2019), são apresentados os modelos Bayes Logistic Regression, Árvore de Decisão, Random Forest, Support Vector Machine e rede neural aplicados a um *dataset* de uma instituição financeira nos Estados Unidos. Os autores apresentam o resultado obtido, demonstrando a eficiência dos modelos aplicados a AML, fomentando ainda mais a criação do nosso trabalho.

3.4 CONSIDERAÇÕES

Apesar dos trabalhos citados e dos seus positivos resultados, a busca para encontrar maneiras eficientes de combater a lavagem de dinheiro está apenas no início, considerando a dificuldade na identificação e a recente legislação deste crime. Sendo assim, este trabalho apresenta novas tentativas para solucionar este problema utilizando *machine learning*.

A não demonstração dos valores dos trabalhos relacionados se deve ao fato de, os autores utilizam diferentes métricas para avaliar o desempenho de seus trabalhos, não havendo uma maneira precisa de comparação entre estes.

4 PROPOSTA

Este capítulo tem como objetivo expor a técnica de lavagem de dinheiro e analisar as ferramentas e dados utilizados para a detecção de fraudes em transações financeiras.

4.1 LAVAGEM DE DINHEIRO

A lavagem de dinheiro consiste no processo de tornar um dinheiro ilícito provindo de atividades criminosas, como tráfico de drogas e armas, em um dinheiro aparentemente lícito podendo ser utilizado em qualquer negócio do mercado financeiro. Este processo de lavagem de dinheiro consiste em três etapas (Mulig e Smith, 2004), exemplificado na figura 4.1.

A primeira etapa conhecida é a inserção do dinheiro ilícito no mercado financeiro. Nesta etapa, o fraudador utiliza-se de depósitos e compra de produtos ou bens, com o dinheiro fraudulento. Nota-se a dificuldade de identificar a origem do capital utilizado na compra de um produto, já que este pode ser de um valor irrisório fazendo com que o estabelecimento, que está vendendo o produto, não questione sobre a origem do dinheiro ocasionando na aceitação deste capital no sistema financeiro podendo ser futuramente repassado para outros estabelecimentos.

A segunda etapa consiste em esconder a origem do dinheiro ilícito. Os criminosos utilizam transferências eletrônicas criando várias camadas complexas de transações, utilizando-se de anonimato ou empresas fictícias e de fachada. Além disso, uma técnica comum é dividir um valor em valores menores e executar múltiplas transferências. Considerando o altíssimo número de transações feitas diariamente, identificar o que pode ser fraudulento ou não é extremamente difícil.

A terceira etapa é a conclusão da lavagem de dinheiro, fazendo com que este seja aceito em transações consideradas legítimas. Assim, não haverá uma suspeita e o fraudador irá alcançar o seu objetivo.

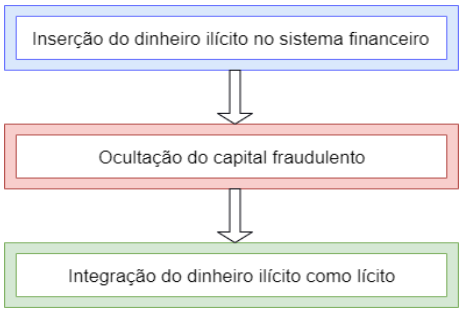

Figura 4.1: As três etapas para a realização da lavagem de dinheiro. A inserção é feita por métodos como depósitos e compra de bens. A ocultação tem como objetivo esconder a origem do dinheiro fraudulento. Por fim, a integração do dinheiro ilícito como um capital lícito podendo ser utilizado em qualquer negociação. Fonte: autores.

Considerando as três etapas, o presente trabalho de graduação tem como proposta agir sobre a segunda etapa da lavagem de dinheiro identificando possíveis transações financeiras ilícitas.

4.2 APRENDIZAGEM SUPERVISIONADA

A aprendizagem supervisionada consiste em modelar algoritmos de *machine learning* com dados que já contêm a resposta para o problema. No estudo deste trabalho de graduação, o *dataset* utilizado aponta quais dados são fraudulentos e quais não são, portanto, definimos os métodos como supervisionados (Cunningham et al., 2008).

Além disso, como o nosso objetivo é, após o treinamento de um modelo de *machine learning*, poder classificar um dado como fraudulento ou não, este será um problema de classificação.

4.3 PAYSIM

Para obter o sucesso no aprendizado de máquina, é necessário utilizar um conjunto de dados suficiente para treinar um algoritmo que seja capaz de processar diferentes informações. No entanto, empresas e instituições não compartilham dados financeiros, dificultando o agrupamento de informações para treinar uma máquina. Dessa maneira, o PaySim, desenvolvido por Lopez-Rojas et al. (2016), supre essa necessidade gerando dados sintéticos de transações financeiras.

A ferramenta gera dados sintéticos de operações feitas em um aplicativo *mobile* de transferência de dinheiro e é construída utilizando o conceito de *Multi Agent Based Simulation*, onde é implementada uma lógica de agentes simulados, na qual os mesmos têm a função de imitar o comportamento de transações financeiras reais, com o objetivo de que o *output* gerado pela ferramenta se assemelhe à um *dataset* real, e assim, podendo ser utilizado em estudos relacionados a transações financeiras. Nesta ferramenta, existem somente dois tipos de agentes, vendedores e clientes.

A sua construção se baseia em uma amostra real de transações financeiras, extraídas de um serviço *mobile* utilizado em um país da África. Ao analisar essa amostra, foi retirada informações essenciais para a construção de uma ferramenta que gerasse um *dataset* sintético similar a um conjunto de dados obtidos de uma empresa, contendo operações financeiras como débito, transferência, pagamento, etc., quem é o originário da operação e o destinatário, a quantidade transferida, como mostra a tabela 4.1. A imagem 4.2 mostra o quão próximo da realidade foram os dados gerados.

4.3.1 Análise dos Dados

Realizar uma análise dos dados que iremos trabalhar é extremamente importante para que possamos entender comportamentos corretos ou anormais quando estivermos implementando

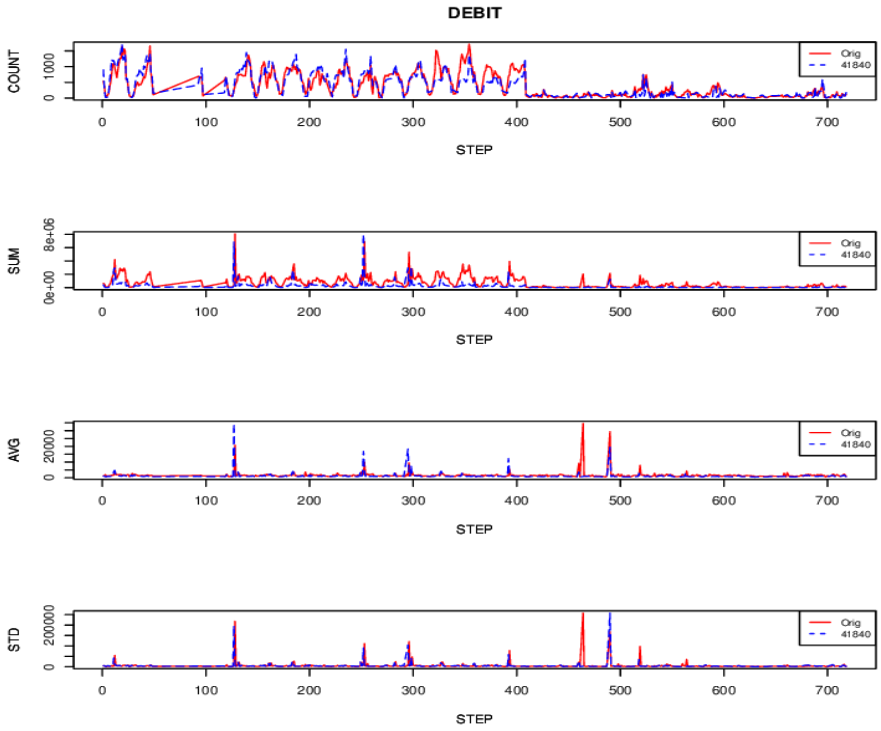

Figura 4.2: Comparação entre o comportamento de dados reais reprentados em vermelho e os dados gerados pela ferramenta Paysim representados em azul. Fonte: Lopez-Rojas et al. (2016).

os modelos de *machine learning*. Outro objetivo de se fazer uma analise é para poder realizar o processo chamado de *Data Cleaning* onde são retiradas as informações que podem ser prejudiciais ou que não apresentariam relevância para o aprendizado de modelos de *machine learning*.

A base de dados que usamos, gerada pelo Paysim, contém 6 milhões de dados, e possui onze colunas diferentes:

- *step*: coluna utilizada para mapear unidade de tempo em horas, dentro de um mês.

- *type*: tipos de transações financeiras possíveis, podendo ser *cash-in*, que é o processo de pagar um vendedor em dinheiro; *cash-out* receber dinheiro de um vendedor; *debit* que é similar ao *cash-out* mas envolve mandar dinheiro do serviço financeiro *mobile* a uma conta bancária; *payment* é o processo de pagamento por produtos ou serviços a um vendedor; e *transfer* é o processo de enviar dinheiro para outro usuário pela plataforma de serviço *mobile*.

- *amount*: valor que representa a quantidade transferida nas transações.

- *nameOrig*: identificação do cliente que iniciou a transação.

- *oldbalanceOrg*: balanço da conta do cliente antes de efetuar a transação.

- *newbalanceOrig*: balanço da conta do cliente depois de efetuar a transação.

- *nameDest*: identificação do cliente que recebe a transação.

- *oldbalanceDest*: balanço da conta do cliente antes de receber a transação.

- *newbalanceDest*: balanço da conta do cliente depois de receber a transação.

- *isFraud*: identifica se uma transação é fraudulenta (com o valor um) ou não fraudulenta (com valor zero).

- *isFlaggedFraud*: sinaliza uma tentativa de ilegalidade ao transferir mais de 200.000 em apenas uma transação.

type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	newbalanceDest	isFraud
PAYMENT	9839.64	C1231006815	170136.0	160296.36	0.0	0
TRANSFER	181.0	C1305486145	181.0	0.0	0.0	1
CASH_OUT	181.0	C840083671	181.0	0.0	0.0	1
DEBIT	5337.77	C712410124	41720.0	36382.23	40348.79	0

Tabela 4.1: Tabela representando as colunas *type*, *amount*, *nameOrig*, *oldbalanceOrig*, *newbalanceOrig*, *newbalanceDest*, *isFraud*, criadas pelo *PaySim* com seus respectivos valores. Fonte: adaptado de Lopez-Rojas et al. (2016).

O primeiro e mais importante fator neste *dataset* é o quanto os dados são desbalanceados. Em um *dataset* de 6.354.407 *inputs* que não são fraude e apenas oito mil, duzentos e treze casos de fraude. O que significa que temos 99,87% de não fraude e 0,13% de fraude.

A primeira coluna que analisamos foi *isFlaggedFraud*, o qual teria valor 1 caso houvesse uma tentativa de transferir um valor maior do que duzentos mil em uma única transferência. Contudo, notamos que existem alguns casos onde a variável não é ativada mesmo que a condição seja satisfeita, mas isso se deve ao fato de que alguns indivíduos são vendedores, por isso podem fazer transações maiores que o limite de duzentos mil e ainda serem legítimas. Outro ponto a se notar é que, quando *isFlaggedFraud* é ativada a transação é interrompida, fazendo com que o *oldBalanceDest* seja igual ao *newBalanceDest*.

Uma maneira comum de se detectar chance de transações suspeitas, é checar se o individuo está realizando ou recebendo uma quantidade de transações fora do normal, ou se logo após ser o destinatário de uma transição, ele iniciou outra. Não foram encontradas essas relações descritas quando *isFlaggedFraud* estava ativo.

Nota-se uma relação interessante entre duas colunas, *type* e *isFraud*. A partir delas, notamos que *isFraud* só é ativado em operações do tipo *transfer* e *cash_out*. Isso remete a como acontece no mundo real, onde um dos momentos finais antes do dinheiro se tornar lícito, ocorre a transferência do dinheiro, já injetado no mercado financeiro e livre de suspeita de fraude. Outro exemplo é a ocorrência de excedente de pagamento de uma compra onde o vendedor retorna um valor.

Por fim, verifica-se que o *dataset* apresenta comportamentos semelhantes ao mundo real, pois leva em consideração casos onde podem ocorrer operações que seriam classificadas como suspeitas, mas dado o contexto que estão inseridas, são legítimas, como no caso dos vendedores. Outras situações são as etapas finais de lavagem de dinheiro, onde ilustram o retorno do capital para o fraudário.

5 METODOLOGIA

Neste capítulo são abordadas as métricas utilizadas, e apresentadas discussões sobre parâmetros escolhidos nos modelos. Para os parâmetros não mencionados, não foram necessárias mudanças já que estes não alteraram significativamente ou afetaram negativamente o resultado final. A implementação dos modelos de *machine learning* utilizados provém da biblioteca do Scikit-learn (Pedregosa et al., 2011). Para o tratamento do arquivo que contém o *dataset* utilizamos a biblioteca Pandas (Virtanen et al., 2019). Por fim, a biblioteca Matplotlib Hunter (2007) foi usada para plotar os gráficos de resultado.

5.1 DATASET

Como explicado anteriormente, a análise dos dados é importante para identificar e retirar ou modificar informações que podem prejudicar o treinamento do modelo. Nesta sessão explicaremos as mudanças realizadas no conjunto de dados.

5.1.1 Data Cleaning

Ao analisar o *dataset*, percebeu-se que o *step* não era um dado adequado para se utilizar no treinamento. Este, representa uma unidade de tempo em horas, dentro de um período de um mês, assim, considerando um mês de 31 dias, o valor máximo do *step* é de 744. Note que em um caso real, poderia haver a seguinte situação: entre os dias 24 e 30 ocorre o pagamento dos clientes de uma empresa. Sabendo disso, um fraudador pode utilizar este período de numerosas transações e transferir um dinheiro fraudulento, passando despercebido. Neste cenário, a utilização do *step* seria útil e poderia resultar em um modelo excelente para prever a legitimidade da transação.

No entanto, essa natureza do *step* é muito volátil, isto é, este padrão pode mudar constantemente, visto que o fraudador dificilmente utiliza a mesma técnica, como pode ser observado nos diferentes *datasets* gerados pelo *PaySim*. Dessa maneira, ao treinar-se os modelos com a coluna *step* o modelo considerava esta como muito relevante, mas ao aplicar a técnica de validação cruzada e utilizar um *dataset* diferente para a verificação, era notável o *overfitting* que acabava ocorrendo nos três modelos de *machine learning*, como pode ser visto na figura 5.1 aplicado à árvore de decisão.

Assim, decidiu-se retirar a coluna *step* já que esta trazia resultados inconsistentes e não auxiliava no descobrimento de casos de fraude.

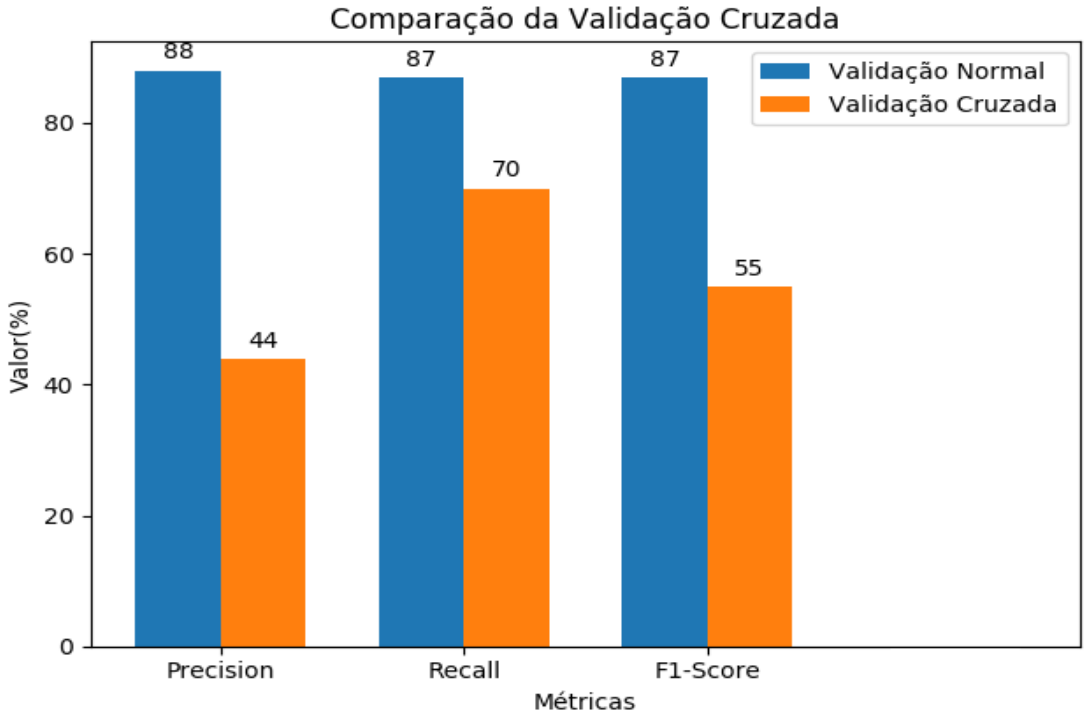

Figura 5.1: Este gráfico esboça as métricas de uma árvore de decisão com o erro de *overfitting*. Os valores acima das barras azuis representam a métrica calculada utilizando o mesmo *dataset* com o qual o modelo foi treinado. Os valores acima das barras laranjas representam a métrica calculada a partir da validação cruzada, ou seja, utilizando um *dataset* diferente do usado para treinamento do modelo. Fonte: autores.

5.1.2 Adequação dos Dados

Os modelos que utilizamos neste trabalho só aceitam valores numéricos, isto é, inteiros e racionais, por isso, no nosso *dataset*, foi necessário traduzir as informações que eram letras ou palavras para um equivalente numérico.

Desta maneira, as *features* afetadas foram type, nameOrig, nameDest. Onde os tipos de transações foram padronizados de 0 a 4, *cash-in*, *cash-out*, *debit*, *payment* e *transfer* respectivamente. Nas features de nome, foram trocados a letra que identificava o individuo como cliente ou como vendedor, para 1 e 2 respectivamente.

5.2 ÁRVORE DE DECISÃO

Foram abordados os parâmetros utilizados para a adequação do modelo sobre o *dataset*, evitando o problema de *overfitting*.

- *Criterion*[1]: este parâmetro define qual o método utilizado para a melhor divisão dos vértices na árvore de decisão. Os dois mais conhecidos são a impuridade de Gini e a entropia da informação. Nas observações dos resultados, a entropia da informação se mostrou mais eficiente, e isso se deve ao fato da entropia sempre buscar uma divisão mais homogênea, tendendo a não

[1]Criterion utilizado: Entropy

agrupar dados diferentes em um mesmo vértice utilizando a seguinte fórmula (Shannon e Weaver, 1998)

$$-\sum_{i=1}^{n} p_i log_2 p_i$$

onde n é a quantidade de classes diferentes e p_i a fração de itens rotulados como pertencentes a classe i. No entanto, apesar da diferença, o resultado obtido através dos dois métodos é de apenas 2%, como é explicado em (Raileanu e Stoffel, 2002).

- *min_samples_split*[2]: parâmetro que indica a quantidade de amostras necessárias para o modelo criar uma divisão de decisão em um vértice da árvore. Inicialmente, o valor deste parâmetro era 2, ou seja, o modelo precisava de no mínimo duas amostras para criar uma divisão de decisão, tornando sua árvore muito específica para aquele conjunto de treino, deste modo podendo gerar *overfitting*. Aumentamos esta quantidade para que o modelo crie vértices de decisões mais homogêneos, obrigando-o a aprender com um maior número mínimo de amostras antes de criar uma divisão.

- *class_weight*[3]: este parâmetro está associado com a importância de cada classe. Para o nosso *dataset*, as possíveis classes são não fraude (representado por 0) ou fraude (representado por 1). Como o objetivo deste trabalho é identificar a maior quantidade de casos de fraude, aumentamos o peso dos dados que são considerados fraudulentos com a intenção de dar ênfase aos casos T_p.

5.3 XGBOOST

- *learning_rate*[4]: constante atribuída a cada iteração do algoritmo, tal que quanto menor o seu valor, menor é o peso resultante a cada decisão. Os valores podem variar entre 0 e 1. Este parâmetro tem como objetivo diminuir o peso das *features* para evitar *overfitting*, contudo diminui-lo demasiadamente pode gerar *underfitting*.

- *max_depth*[5] : parâmetro utilizado para definir a profundidade máxima das árvores geradas pelo modelo. Note que, aumentar este valor acarreta em árvores mais profundas, sendo que o limite da profundidade será a maior homogeneidade da amostra utilizada, podendo causar *overfitting* no modelo. Sendo assim, aumentamos o valor da profundidade das árvores geradas pelo XGBoost, melhorando as decisões, sem causar *overfitting*.

5.4 MULTILAYER PERCEPTRON

Como os dados presentes no nosso *dataset* variam desde categorias (type), números de balanços iniciais e finais e nome dos indivíduos presentes nas transações, mostra que nosso

[2]Valor de min_samples_split utilizado: 45
[3]Valor de class_weight utilizado: {0:1, 1:2}
[4]Valor de learning_rate utilizado: 1
[5]Valor de max_depth utilizado: 16

dados não estão normalizados, e isso pode ser um problema para modelos que são sensíveis à *feature scaling*. Os modelos de redes neurais fazem parte do conjunto que apresentam problemas no aprendizado quando os dados estão demasiadamente desproporcionais.

Por exemplo, ao transformarmos os nomes de vendedores e clientes em números, estes tornaram-se inteiros de valores extremamente altos. O contrário aconteceu quando transformamos as categorias em números, estes viraram números de 0 a 4. Por causa dessa diferença de valores, os modelos podem concluir que as *features* "maiores" são mais importantes e relevantes para o problema, o que no nosso caso não é correto. Assim, para resolver este problema utilizamos o método StandardScaler.

- *early_stopping*[6]: este parâmetro é responsável por parar antecipadamente o treinamento do modelo, caso a pontuação de validação não esteja melhorando. Este método de parada auxilia na não ocorrência do problema de *overfitting*, já que o modelo irá parar de treinar em uma certa etapa, não permitindo que os dados fiquem muito específicos para um *dataset*. Então, para o treinamento do nosso modelo, o Multilayer Perceptron irá parar assim que a validação não estiver aumentando por um certo nível de tolerância.

- max_iter[7]: número de iterações que o algoritmo irá fazer até a convergência ou até um limite estabelecido. Para este parâmetro, optamos por diminuir a quantidade de iterações para evitar *overfitting*, fazendo com que a rede neural fique mais genérica para diferentes *datasets*, visto que a busca pela convergência pode utilizar pesos muito específicos nos neurônios acarretando em uma observação muito específica para o *dataset* que está sendo usado para treino.

- *hidden_layer_sizes*[8]: este parâmetro representa a quantidade de *Hidden Layers* e a quantidade de neurônios que estão presentes em cada camada. Para atingir os nossos resultados, precisamos escolher a melhor quantidade de camadas e neurônios. Existem métodos que auxiliam nesta busca, e para os nossos resultados, foi utilizado o método GridSearchCV. Este, é considerado uma busca exaustiva, que dado os parâmetros, irá treinar o modelo e comparar os melhores resultados obtidos com cada parâmetro. Por fim, o GridSearchCV irá retornar a melhor quantidade de camadas e neurônios para o modelo.

[6]Valor de early_stopping utilizado: True
[7]Valor de max_iter utilizado: 100
[8]Valor de hidden_layer_sizes utilizado: (8,8,8)

6 ANÁLISE DOS RESULTADOS OBTIDOS

Neste capítulo são apresentados os resultados obtidos com os modelos de *machine learning* escolhidos. Note que o nosso foco principal são T_n e T_p.

Em relação aos F_p e F_n, vale pontuar suas consequências e relações numa aplicação real, isto é, se uma empresa estivesse utilizando *machine learning* para identificar fraudes.

Aplicado à uma situação real, cada F_p identificado pelo algoritmo significa que uma pessoa foi identificada como um fraudador, porém é um cliente legítimo. Esse erro levaria a empresa a iniciar um processo interno de investigação deste cliente, podendo acarretar em custos operacionais e monetários. Do mesmo modo, cada F_n identificado pelo algoritmo significa que uma uma pessoa identificada como legítima, na verdade, era um fraudador. Neste último caso não haveriam gastos com investigações, porém, a fraude não seria identificada e o dinheiro fraudulento, de valor muito maior do que o custo de investigar um cliente, é passado despercebido.

Isso nos faz pensar que existe uma relação matemática entre eles que nos leve ao custo médio, isto é a proporção aceitável de F_n e F_p em relação a quantidade de T_p. Isso seria material para estudos futuros, já que não possuímos informações de custo real de uma investigação e a quantidade perdida em média por uma empresa, assim, sem estas informações não podemos montar uma análise proveitosa.

6.1 ÁRVORE DE DECISÃO

Observando a matriz de confusão, na figura 6.1, criada a partir modelo de árvore de decisão podemos concluir que houve bons resultados. Na identificação de casos onde não houve fraude, o modelo conseguiu predizer 99,97%, errando apenas 0,03%. Para os casos de identificação de dados fraudulentos, o modelo conseguiu predizer 89,03%, errando 10,97%.

Em um cenário real, para os casos de não fraude, uma instituição financeira iria investigar apenas 0,03%, reduzindo um custo, tanto financeiro como operacional, muito alto quando comparado com um cenário sem o modelo de *machine learning*. Em uma situação onde existam bilhões de transações, o modelo de árvore de decisão tornaria possível a verificação de fraudes neste sistema que antes poderia ser considerado inviável.

Para os casos de identificação de dados fraudulentos, o modelo conseguiu predizer 89,03%, não conseguindo identificar 10,97%. A eficiência do algoritmo deve ser analisado para cada ambiente que está sendo testado. Na nossa avaliação, 89% dos dados é extremamente eficiente já que representa um alto valor monetário fraudado que será descoberto, além disso, podendo levar a outros casos de fraude, que não foram identificados pelo modelo (até mesmo dentro dos 10,97%), cometidos pelo mesmo fraudador.

Observa-se que se o intuito for utilizar o modelo de árvore de decisão em um sistema onde há um maior interesse de descobrir fraudes sobre não fraudes, isto é, obter um menor

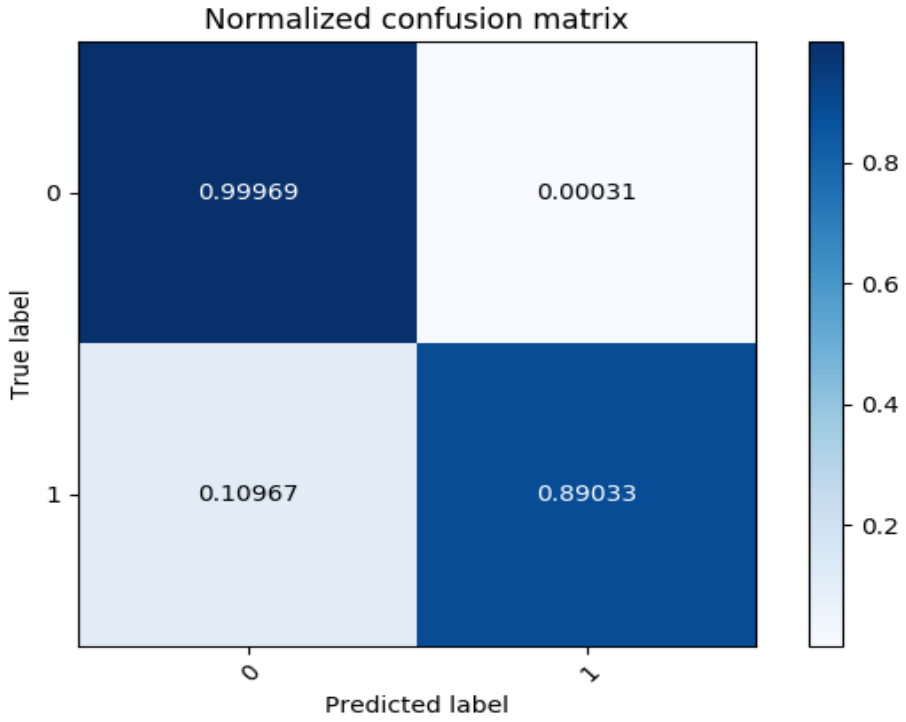

Figura 6.1: Matriz de confusão normalizada do modelo de Árvore de Decisão com 6 milhões de dados. Fonte: autores.

valor de F_n que acarretará em um aumento no valor de casos de F_p, basta alterar o parâmetro *class_weight* do modelo, explicado na metodologia deste trabalho.

Para as métricas, expostas na figura 6.2, podemos analisar com um maior cuidado a eficiência do modelo de árvore de decisão. Este apresentou valores aceitáveis, com um *recall* maior que a *precision*. Assim, concluímos que este modelo é ideal para um cenário onde há o intuito de descobrir transações fraudulentas, não havendo uma preocupação com uma menor taxa de F_n quando comparado a taxa de F_p.

A curva *Precision-Recall* a seguir 6.3 apresenta a relação entre a taxa de precisão e o *recall*, ela representa as perdas e ganhos do modelo conforme os valores mudam. Neste caso a curva não foi criada de modo suave, isto é, não há uma continuidade dos pontos saindo de (1,0; 0) até (0;1,0), isso nos indica que o modelo, no início de sua aprendizagem, possuía uma taxa de *recall* aproximadamente 0,7 e *precision* 0,98 aproximadamente.

6.2 XGBOOST

A matriz de confusão, da figura 6.4, que representa as classificações geradas pelo XGBoost, evidencia a qualidade do modelo aplicado ao problema de anti-money laundering (AML). Para a identificação de casos não fraudulentos o resultado obtido foi de 99,986%, demonstrando a sua alta capacidade de classificação.

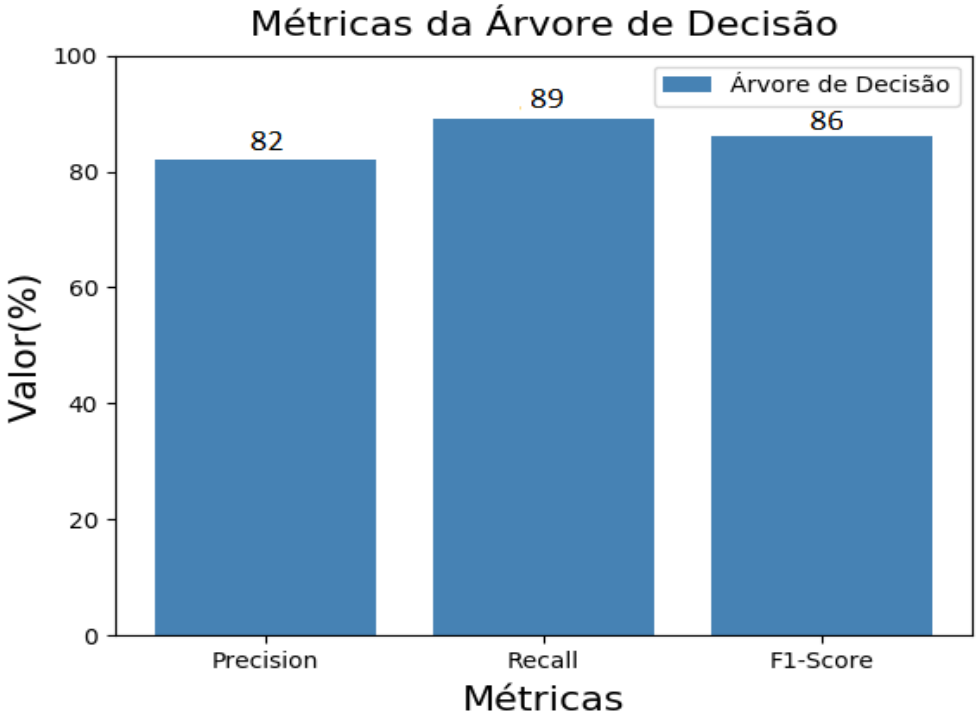

Figura 6.2: Representação gráfica do valor de *precision*, *recall* e *F1-Score* do modelo de Árvore de Decisão, após o treinamento do algoritmo utilizando 6 milhões de dados. Fonte: autores.

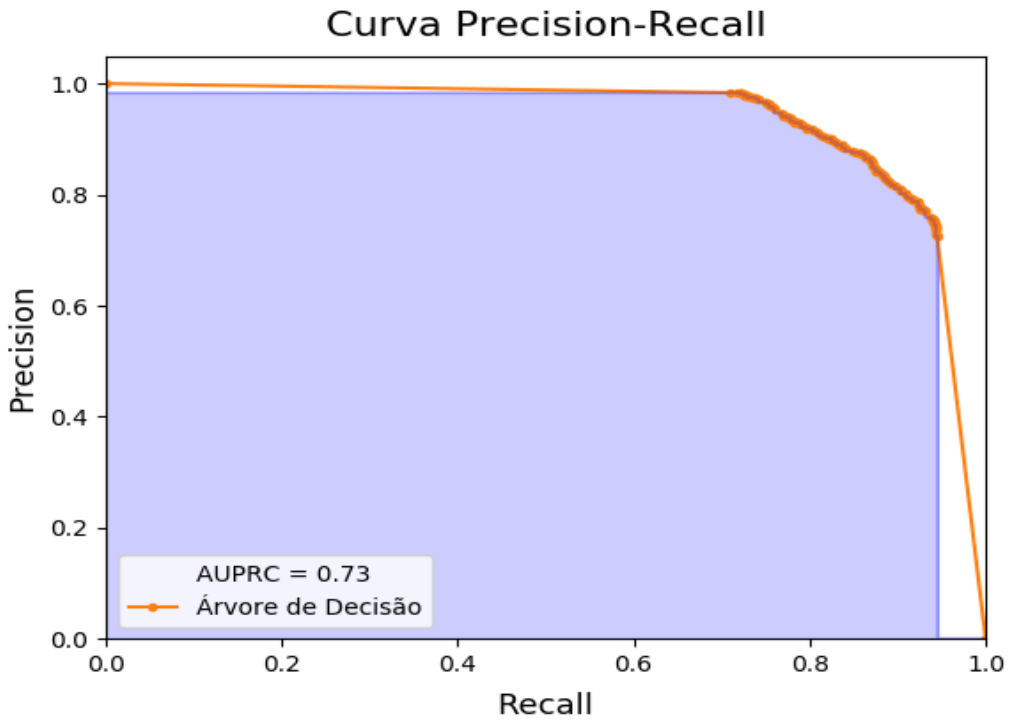

Figura 6.3: Curva de Precision-Recall da Árvore de Decisão gerada após treinamento do algoritmo utilizando 6 milhões de dados. Os pontos gerados começam por volta da posição (0,7;0,9) e termina em (0,95, 0,75), os outros pontos representam os *trade-offs* entre *recall* e *precision*, a curva criada forma uma área de 0,73. Fonte: autores.

Supondo cenários reais em que este modelo possa ser aplicado, o seu alto índice de identificação de casos T_n permite que uma instituição financeira, buscando resolver o problema de ML, invista o seu capital apenas na investigação de clientes considerados suspeitos, já que apenas 0,014% destes serão inocentes.

A taxa de 15,42% de transações fraudulentas consideradas lícitas, pode gerar um grande valor fraudado. Contudo, a possibilidade de identificação de casos T_p é extremamente alta (84,57%), sendo aplicável em cenários onde há grandes quantidades de transações.

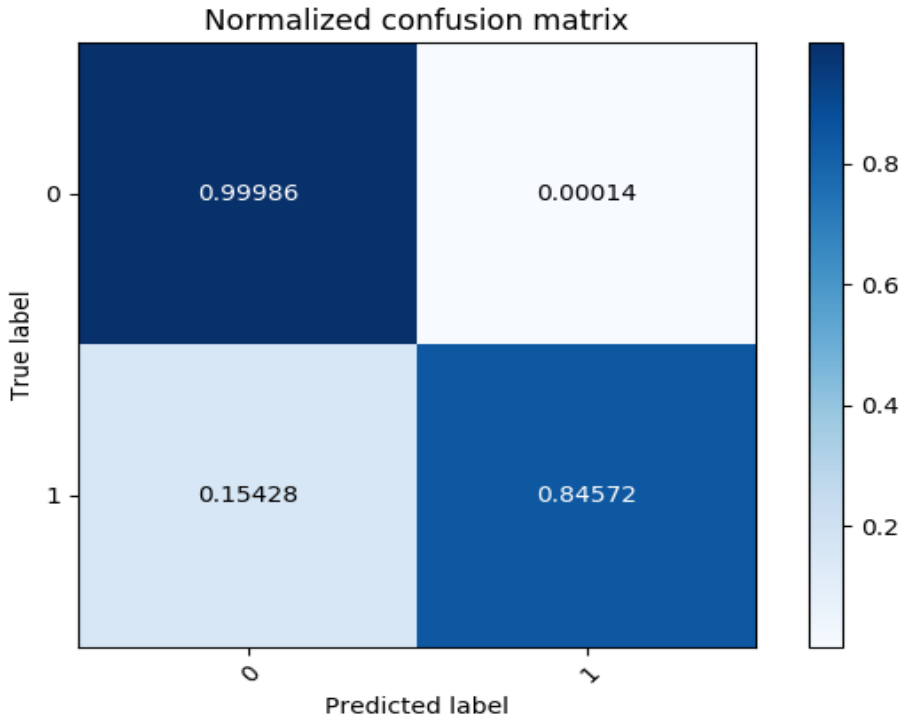

Figura 6.4: Matriz de confusão normalizada do modelo XGBoost, treinado com 6 milhões de dados. Fonte: autores.

Para as métricas, figura 6.5, o XGBoost apresenta um excelente desempenho. O seu valor de precisão é extremamente alto e a sua taxa de *recall* também se mantém alta. Com isso, o *F1-Score* chega a 88%, demonstrando a qualidade do modelo. Assim, o XGBoost demonstra grande aplicabilidade em situações reais, apresentando valores de métricas elevados e poucos pontos negativos.

A curva *Precision-Recall*, na figura 6.6, mostra como foi a relação *precision-recall* do modelo XGBoost, a curva é iniciada no ponto (0,1;0,92), e a lacuna entre o ponto inicial (1,0;0) significa que o modelo quando começou a aprender, apresentou um *recall* de 0,1 enquanto a *precision* estava em 0,92. A partir deste ponto a curva se torna mais suave, com diferenças minúsculas entre os pontos plotados, até parecendo uma linha continua. A área embaixo dessa curva é 0,77.

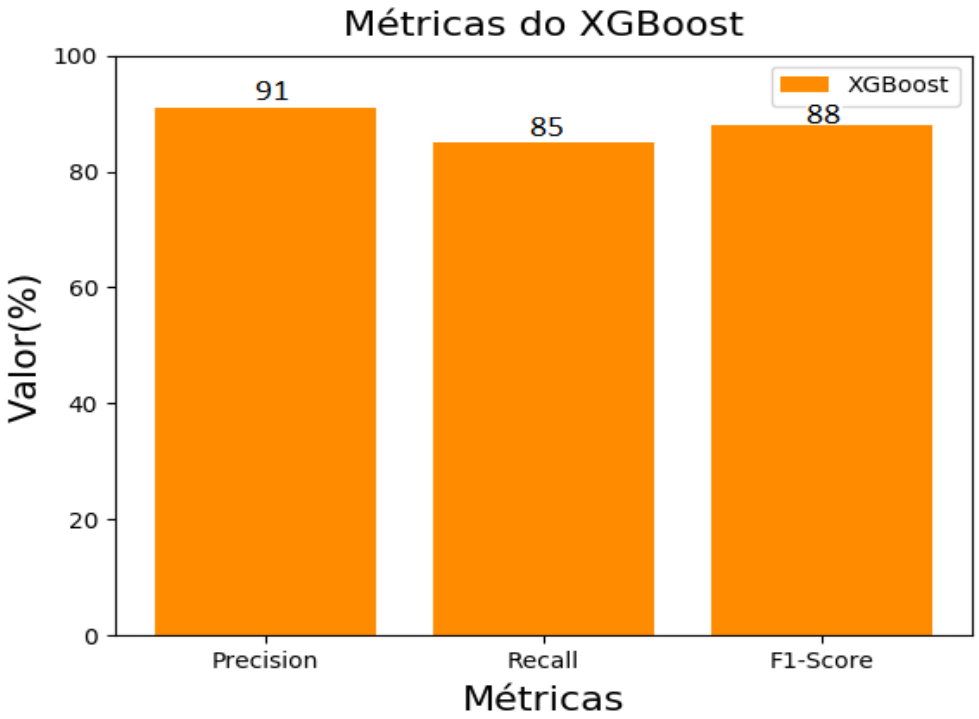

Figura 6.5: Representação gráfica do valor de *precision*, *recall* e *F1-Score* do modelo XGBoost, após o treinamento do algoritmo utilizando 6 milhões de dados. Fonte: autores.

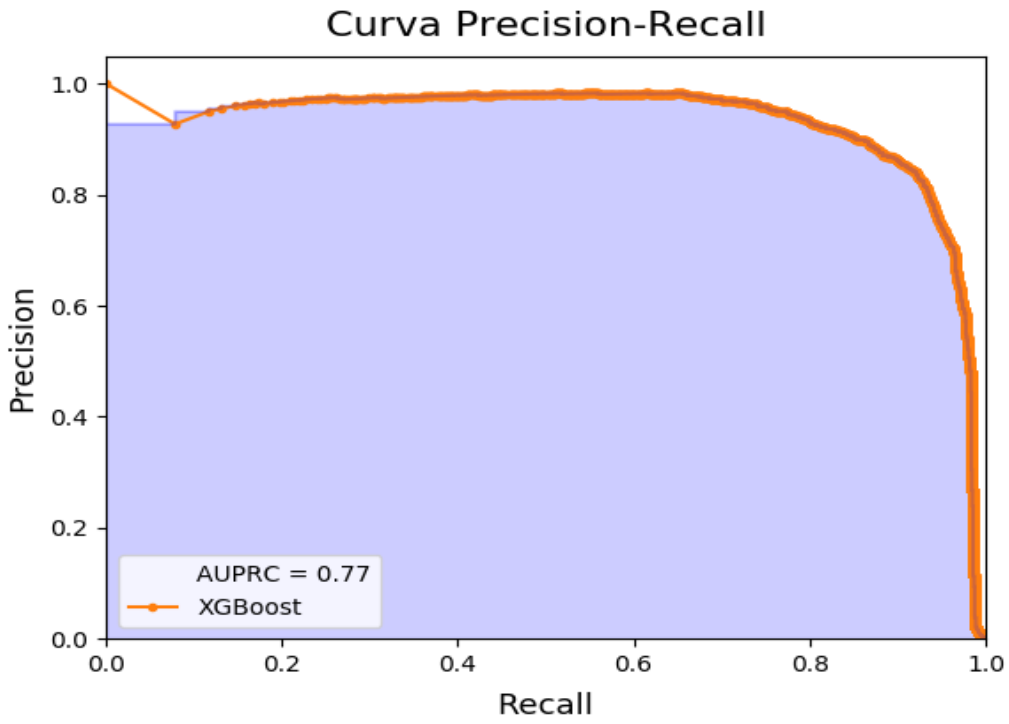

Figura 6.6: Curva de *Precision-Recall* do XGBoost gerada após treinamento do algoritmo utilizando 6 milhões de dados. Seu *trade-off* começa a partir do ponto (0,93; 0,1) e termina no ponto (0; 1,0), os outros pontos representam os *trade-offs* do *precision-recall*, a curva criada forma uma área de 0,77. Fonte: autores.

6.3 MULTILAYER PERCEPTRON

Como pode se observar na matriz de confusão da figura 6.7, o modelo *Multilayer Perceptron* (MLP) obteve resultados aceitáveis. Para transações não fraudulentas, o modelo de MLP obteve uma precisão de 99,989% em casos onde não houve fraude, garantindo assim a sua aplicabilidade em um cenário onde há uma grande quantidade de transações. Em casos onde há transações fraudulentas, o modelo conseguiu identificar 76,76%, demonstrando ainda um valor aceitável.

Explorando os cenários possíveis, para uma instituição financeira, eliminar 99,98% dos dados tendo a certeza que estes não são fraudulentos, torna viável a sua aplicação. No entanto, não identificar 23,23% pode inviabilizar a sua aplicabilidade em algumas situações, visto o alto valor que pode ser fraudado sem que o modelo identifique.

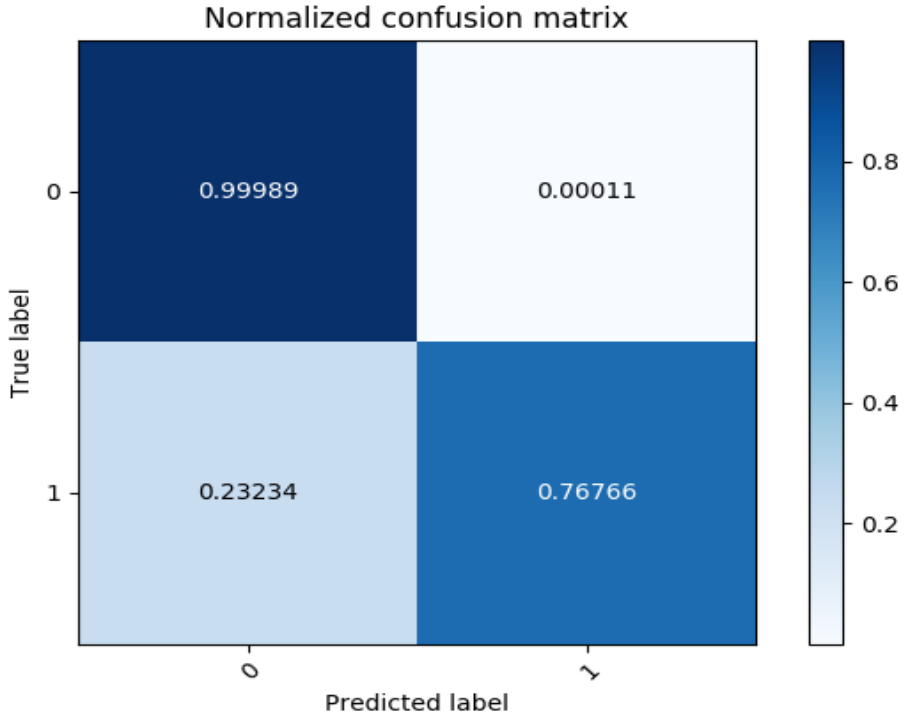

Figura 6.7: Matriz de confusão normalizada do modelo *Multilayer Perceptron*, treinado com 6 milhões. Fonte: autores.

Para as métricas apresentadas na figura 6.8, podemos ver que a taxa de precisão foi 13% mais alta que o *recall*, isso nos mostra uma maior taxa de F_n quando comparado aos casos de F_p. Assim, a aplicação do MLP deve ser utilizada em ambientes onde há um maior interesse em não gastar valor monetário investigando transações não fraudulentas.

A curva Precision-Recall do modelo *Multilayer Perceptron*, mostrada na imagem 6.9, teve uma relação de *trade-off* mais suave, não houveram pontos que estavam muito longe de seu ponto vizinho e por isso é a mais suave entre os três modelos, e teve como valor de área da curva 0,70.

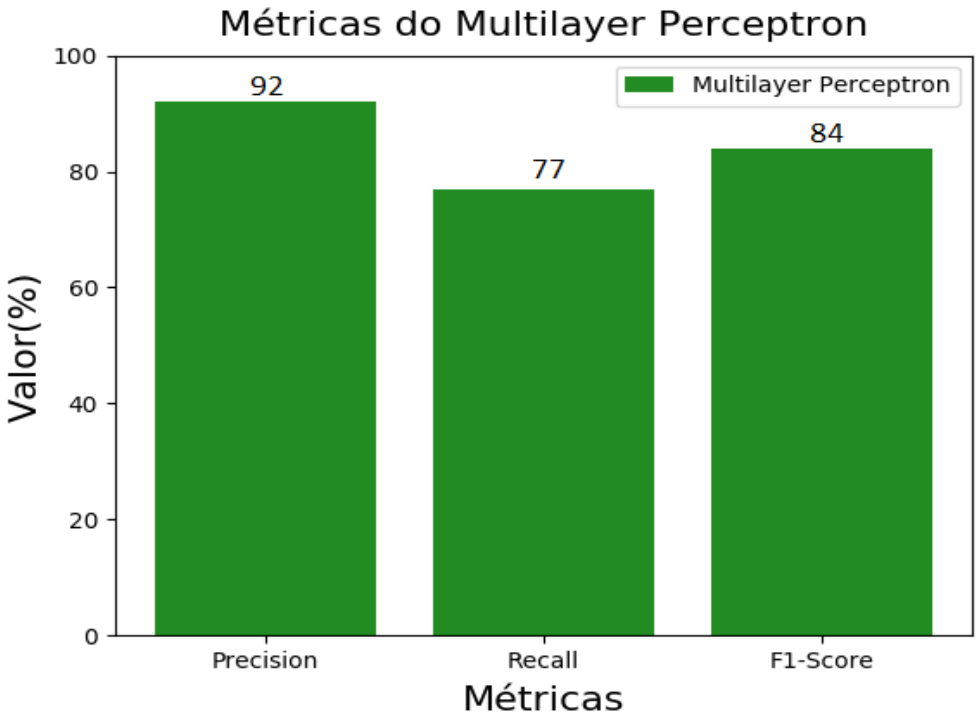

Figura 6.8: Representação gráfica do valor de *precision*, *recall* e *F1-Score* do modelo Multilayer Perceptron, após o treinamento do algoritmo utilizando 6 milhões de dados. Fonte: autores.

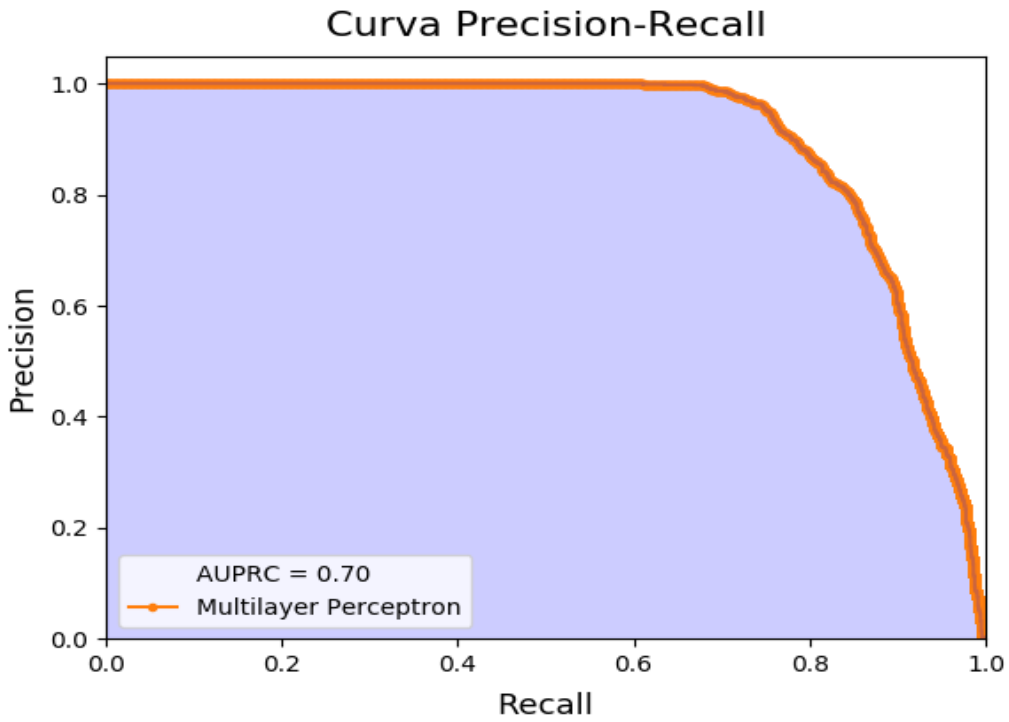

Figura 6.9: Curva de *Precision-Recall* do *Multilayer Perceptron* gerada após treinamento do algoritmo utilizando 6 milhões de dados. Os pontos de *trade-offs* iniciam-se no (1,0;0) e terminam no (0;1,0), curva criada forma uma área de 0,70. Fonte: autores.

7 COMPARAÇÃO DOS RESULTADOS

Considerando os resultados obtidos, os três modelos tiveram um excelente resultado. No entanto, é necessária uma discussão sobre possíveis cenários e em quais devem ser aplicados os modelos, como pode ser visto na figura 7.1, cada algoritmo de *machine learning* teve um ponto forte diferente.

Assim, em uma situação hipotética onde clientes de um banco realizam milhares de transações bancárias em menos de um mês, considera-se que para este banco o prejuízo econômico e operacional para investigar uma transação seja alto, portanto, o cenário mais interessante é aquele onde há um menor número de clientes para investigar. Dessa maneira, o *Multilayer Perceptron* será o modelo que excluirá a maior quantidade de clientes, gerando o menor prejuízo para o banco.

No entanto, se o intuito é a identificação do maior número de casos de fraude, ou seja, os casos classificados como T_p, o modelo mais recomendado será o de Árvore de Decisão, que possui o maior valor de *recall* entre os três modelos.

Por fim, se a ideia é manter um equilíbrio entre encontrar a maior taxa de fraude e gastar a menor quantidade possível de dinheiro investigando um cliente, o melhor modelo será o XGBoost. Este apresentou um *F1-score* de 88, visto que obteve um alto desempenho em *precision* e *recall*. Além disso, esta ideia é reforçada pela observação da comparação das curvas AUPRC na figura 7.2, indicando o equilíbrio que o XGBoost proporciona na identificação dos casos de T_p e T_n.

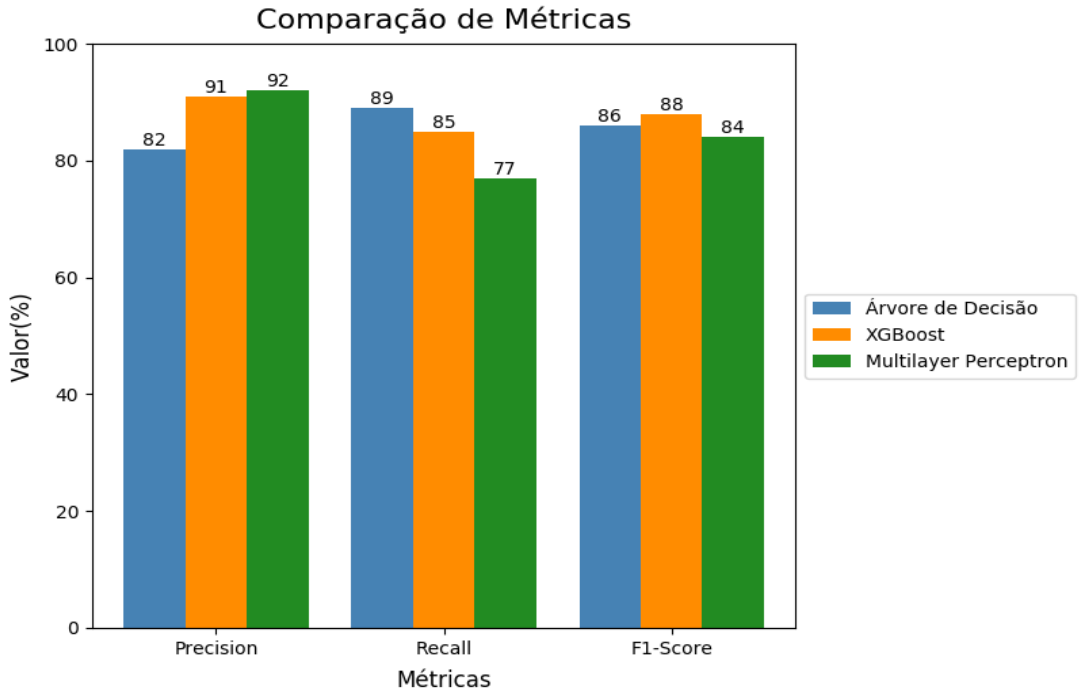

Figura 7.1: Representação do valor, em porcentagem, das métricas aplicadas aos três modelos de *Machine Learning*. Em azul, Árvore de Decisão. Em laranja, XGBoost. Em verde, *Multilayer Perceptron*. Fonte: autores.

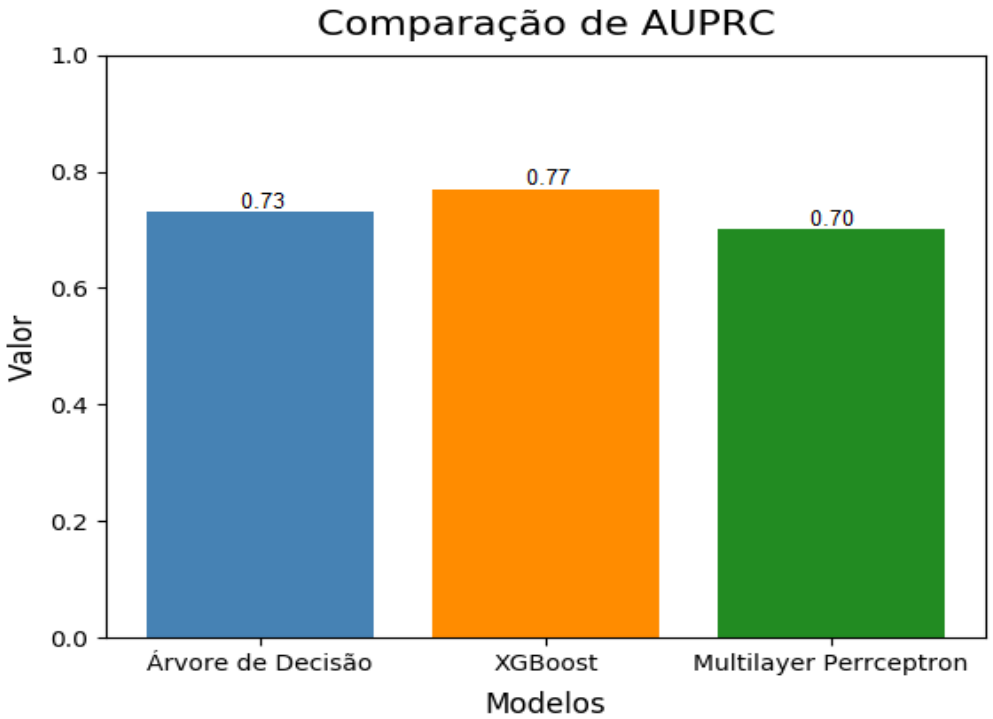

Figura 7.2: Representação do valor, em porcentagem, das métricas aplicadas aos três modelos de *Machine Learning*. Em azul, Árvore de Decisão. Em laranja, XGBoost. Em verde, *Multilayer Perceptron*. Fonte: autores.

8 CONCLUSÃO

Neste trabalho de graduação, foi abordado o crime de lavagem de dinheiro, cuja popularidade aumentou nos anos 80. Mesmo existindo órgãos regulamentadores que tentam evitar esta prática ilegal, como FATF (na Europa) e UIF (no Brasil), estes não se mostram suficientes.

Os trabalhos relacionados mencionados neste texto, demonstram tentativas bem-sucedidas de detectar a ocorrência de crimes de lavagem de dinheiro com diferentes técnicas.

A seguir, foi apresentada uma proposta de identificação de casos fraudulentos utilizando modelos de *machine learning*, sendo eles Árvore de Decisão, XGBoost e *Multilayer Perceptron*, os quais foram treinados utilizando um *dataset* criado pela ferramenta PaySim, já que a escassez de conjunto de dados, a respeito de transações financeiras, dificulta o estudo nesta área.

Os resultados obtidos alcançaram valores para F1-score maiores de 80%, tendo casos onde um modelo era melhor em precision e outro melhor em recall, desta maneira mostram-se satisfatórios, atestando a possibilidade de aplicação em diferentes cenários reais onde cada modelo individualmente teve a sua vantagem. A comparação que ocorre em seguida é necessária, a fim de discutir qual o melhor modelo para cada situação de diferentes instituições financeiras.

Contudo, apesar de alguns modelos terem sido bem sucedidos, ainda há espaço para melhorias nesse campo de pesquisa. Existem áreas na economia que são mais propicias à tentativas de lavagem de dinheiro, como o mercado de imóveis, ações, joias, gado, transações internacionais, obras de arte, etc., já que estes mercados naturalmente movimentam grandes quantidades de dinheiro. Deste modo, uma próxima proposta é utilizar o processo conhecido por *Know Your Customer* (KYC), utilizado por (Wang e Yang, 2007), em conjunto com o que foi desenvolvido aqui. Levando em consideração estas outras características e também analisar as movimentações monetárias, acreditamos que o desempenho dos modelos seria aprimorado significativamente.

Por fim, para uma análise mais profunda, seria interessante conhecer o custo os quais uma empresa teria que arcar para investigar seus clientes e também a quantidade de capital que seria perdido se fraudadores estivessem utilizando seus serviços para executarem ações ilegais. Com estes dados em mãos, poderia-se criar um cálculo de *trade-off* entre a quantidade de F_p e F_n, com o objetivo de se alcançar o prejuízo mínimo para a empresa.

REFERÊNCIAS

Araújo, R. A. (2009). Assessing the efficiency of the brazilian anti-money laundering regulation: a game theoretic approach. *Revista de Economia Mackenzie*, 7(1).

Brid, R. S. (2018). Decision trees - a simple way to visualize a decision.

Cunningham, P., Cord, M. e Delany, S. (2008). *Supervised Learning*, páginas 21–49.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

Europol (2017). From suspicion to action - converting financial intelligence into greater operational impact.

Gardner, M. W. e Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636.

Hornik, K., Stinchcombe, M. e White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Hülsse, R. (2007). Creating demand for global governance: The making of a global money-laundering problem. *Global Society*, 21(2):155–178.

Jun Tang e Jian Yin (2005). Developing an intelligent data discriminating system of anti-money laundering based on svm. Em *2005 International Conference on Machine Learning and Cybernetics*, volume 6, páginas 3453–3457 Vol. 6.

Levi, M. (2002). Money laundering and its regulation. *The Annals of the American Academy of Political and Social Science*, 582(1):181–194.

Li, T. R., Chamrajnagar, A. S., Fong, X. R., Rizik, N. R. e Fu, F. (2019). Sentiment-based prediction of alternative cryptocurrency price fluctuations using gradient boosting tree model. *Frontiers in Physics*, 7:98.

Liu, C., Chan, Y., Alam Kazmi, S. H. e Fu, H. (2015). Financial fraud detection model: Based on random forest. *International journal of economics and finance*, 7(7).

Lopez-Rojas, E., Elmir, A. e Axelsson, S. (2016). Paysim: A financial mobile money simulator for fraud detection. Em *28th European Modeling and Simulation Symposium, EMSS, Larnaca*, páginas 249–255. Dime University of Genoa.

MathWorks (2016). What is deep learning. `https://www.mathworks.com/discovery/deep-learning.html`.

Mulig, E. V. e Smith, M. (2004). Understanding and preventing money laundering.

Paula, E. L., Ladeira, M., Carvalho, R. N. e Marzagão, T. (2016). Deep learning anomaly detection as support fraud investigation in brazilian exports and anti-money laundering. Em *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, páginas 954–960.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. e Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pezoulas, V. C., Exarchos, T. P., Tzioufas, A. G., De Vita, S. e Fotiadis, D. I. (2019). Predicting lymphoma outcomes and risk factors in patients with primary sjögren's syndrome using gradient boosting tree ensembles. Em *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, páginas 2165–2168.

Popescu, M.-C., Balas, V. E., Perescu-Popescu, L. e Mastorakis, N. (2009). Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8(7):579–588.

Raileanu, L. E. e Stoffel, K. (2002). Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, 41:77–93.

Rushin, G., Stancil, C., Sun, M., Adams, S. e Beling, P. (2017). Horse race analysis in credit card fraud—deep learning, logistic regression, and gradient boosted tree. Em *2017 Systems and Information Engineering Design Symposium (SIEDS)*, páginas 117–121.

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):e0118432.

Sanjeevi, M. (2017). Chapter 4: Decision trees algorithms.

Schneider, F. e Windischbauer, U. (2008). Money laundering: some facts. *European Journal of Law and Economics*, 26(3):387–404.

Shannon, C. E. e Weaver, W. (1998). *The mathematical theory of communication*. University of Illinois press.

Soltani, R., Nguyen, U. T., Yang, Y., Faghani, M., Yagoub, A. e An, A. (2016). A new algorithm for money laundering detection based on structural similarity. Em *2016 IEEE 7th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, páginas 1–7.

Song, Y.-Y. e Ying, L. (2015). Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, 27(2):130.

Stephen, I. (1990). Perceptron-based learning algorithms. *IEEE Transactions on neural networks*, 50(2).

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C., Polat, İ., Feng, Y., Moore, E. W., Vand erPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P. e Contributors, S. . . (2019). SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. *arXiv e-prints*, página arXiv:1907.10121.

Wang, S.-N. e Yang, J.-G. (2007). A money laundering risk evaluation method based on decision tree. Em *2007 International Conference on Machine Learning and Cybernetics*, volume 1, páginas 283–286. IEEE.

Weber, M., Chen, J., Suzumura, T., Pareja, A., Ma, T., Kanezashi, H., Kaler, T., Leiserson, C. E. e Schardl, T. B. (2018). Scalable graph learning for anti-money laundering: A first look. *CoRR*, abs/1812.00076.

Zhang, Y. e Trubey, P. (2019). Machine learning and sampling scheme: An empirical study of money laundering detection. *Computational Economics*, 54(3):1043–1063.

Zhang, Z. M., Salerno, J. J. e Yu, P. S. (2003). Applying data mining in investigating money laundering crimes. Em *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 747–752. ACM.