DETECÇÃO DE FRAUDE EM TRANSAÇÕES ONLINE DE CARTÕES DE CRÉDITO

Lucas de Toledo Barreto[2] e Victor Afonso Bauler[3]
Departamento de Engenharia Elétrica e Eletrônica - Programa de Pós-Graduação em Engenharia Elétrica (PPGEEL)
[1] Universidade Federal de Santa Catariana, Florianópolis - Santa Catarina, Brasil
[2] Endereço de e-Mail: lucasdetoledo4@gmail.com
[3] Endereço de e-mail: victor.bauler@gmail.com

Resumo – Devido ao rápido desenvolvimento da tecnologia, o crescimento do número de bancos online se tornou exponencial e, paralelamente, o fluxo de transações online seguiu o mesmo caminho. Posto isto, observa-se um crescimento no fluxo de dados, expondo vulnerabilidades de tais entidades bancárias e acarrentanso em maiores possibilidades de fraudes em transações. De acordo com a Associação Brasileira das Empresas de Cartão de Crédito e Serviços (Abces), no Brasil, no primeiro trimestre de 2022, ocorreu um aumento de 46,5% nas compras online. Visando garantir maior segurança aos usuários, cada vez mais, empresas ao redor do mundo buscam soluções para minimizar ou sanar este problema. Portanto, com o objetivo de mitigar o problema de detecção de fraude em cartões de crédito em compras online, neste artigo, aplicam-se modelos de aprendizado de máquina de Random Forest, XGBoost, LightGBM no *dataset* IEEE-CIS Fraud Detection, disponibilizado pela Vesta Corporation, com o intuito de classificar se uma transação é fraudulenta ou não. Além disso, o modelo final também apresenta as *features* mais importantes para definir a fraudulência.

***Index Terms*—Detecção de fraude, Random Forest, XGBoost, LightGBM.**

I. Introdução

Recentemente, o uso de cartões de crédito e débito têm crescido significativamente, devido a inúmeras razões. Alguns desses fatores incluem: maior conveniência, visto que a compra por meio de cartões de crédito acarretam na não necessidade da utilização de dinheiro em espécie; promoções cedidas pelos bancos, como sistema de cashback e pontos de fidelidade, milhas e outro benefícios; mudança nos hábitos de consumo, devido ao crescimento do mercado de compras online (*e-commerce*) e a popularidade de aplicativos de pagamento móvel.

Dessa forma, observa-se que o crescimento do uso de cartões de crédito influe diretamente no número de compras online realizadas, visto que o processo de comprar remotamente (em que o consumidor possui uma imensa gama de produtos disponíveis), torna-se mais conveniente e cômodo. Entretanto, comprar online também tem seu ônus: os riscos, visto que as informações dos consumidores estão mais expostas. Isso possibilita que fraudadores utilizem-se de técnicas avançadas para acessar informações pessoais e financeiras dos consumidores. Tal acesso malicioso pode ocorrer por meio de inúmeras formas, sendo algumas delas: fraudes de leilão, em que utilizam-se de anúncios de leilão falsos; *phishing*, fraude que envolve o envio de e-mail e mensagens de texto fingindo serem intituições financeiras e pessoas confiáveis.

Nesses casos, os bancos são responsáveis por reembolsar o valor da transação fraudulante ao titular do cartão. Ademais, fraudes em cartão de crédito podem afetar negativamente a reputação, dado que faz com que os clientes percam a confiança na instituição. Sendo assim, para minimizar o risco de prejuízos, bancos buscam investir fortemente em medidas de segurança (como forma de cuidado preditivo) e medidas para identificação de fraudes (para que, caso ocorra uma transação fraudulenta, possa-se identificar rapidamente e tomar as medidas cabíveis àquela situação). Portanto, tem-se que os desafios relacionadas a detecção de fraude ainda são grandes e, assim, inúmeros estudantes e pesquisadores voltam-se para este assunto, utilizando, principalmente, técnicas de *machine learning*, *deep learning* e mineração de dados.

Nesse trabalho, será apresentado uma abordagem utilizando modelos de *machine learning* e *deep learning* para enfrentar o problema de detecção de fraudes de cartões de crédito em transações online. Os modelos foram treinados e aperfeiçoados no *dataset* IEEE-CIS Fraud Detection, disponibilizado pela empresa Vesta Corporation, na plataforma de *data science* Kaggle. Para o restante deste artigo, tem-se a estruturação nos seguintes tópicos: Seção II, trabalhos e artigos relacionados; Seção III, apresentação do *dataset*, Seção IV, metodologia desenvolvida durante o estudo; Seção V, descrição dos resultados apresentados pelo modelo; Seção VI, conclusão geral.

II. Trabalhos Relacionados

Nesta seção, será feita uma breve recapitulação de trabalhos previamente publicados sobre o tema ”detecção de fraudes”. Com isso, os trabalhos a seguir aplicam diferentes modelos de *machine learning* ao *dataset* IEEE-CIS Fraud Detection.

Em [1], os autores propuseram, devido ao desbalanceamento do *dataset*, uma etapa de pré-processamento, em que foram utilizadas 3 técnicas: *SMOTE (Syntethic Minority Oversampling Technique)*, *random over sampling* e *random under sampling*. Em sequência, o *dataset* foi treinado com os seguintes modelos: Naive Bayes, Random Forest, Regressão Logística e modelos de aprendizado profundo (no qual o modelo BiLSTM-MaxPooling-BiGRU-MaxPooling performou melhor). Para as métricas de avaliação, utilizou-se: área abaixo da

curva ROC (AUC-ROC Score), Precision, Recall e F1-score. Por fim, observou-se que o modelo BiLSTM-MaxPooling-BiGRU-MaxPooling, utilizando *random over sampling* na etapa de pré-processamento, obteve melhor performance, atingindo 91.37% de AUC-ROC Score.

Em [2], na etapa de pré-processamento, os autores propuseram a remoção das *features* altamente correlacionadas, em que eles descartaram as que possuíam correlação maior que 95%. Além disso, posteriormente, também foi aplicada a técnica de *RFECV (recursive feature elimination with cross-validation)* para selecionar as features mais importantes e descartar o restante de modo automático. Em sequência, o *dataset* foi treinado em um modelo de *Gradient Boosting*, o LightGBM. Para as métricas de avaliação, utilizou: AUC-ROC Score e acurácia. Por fim, observou-se que o modelo obteve 95.5% de AUC-ROC Score.

III. Dataset

O *dataset* utilizado nesse artigo chama-se IEEE-CIS Fraud Detection, o qual encontra-se no Kaggle. O *dataset* contém 4 arquivos ".csv": *train transaction*, com 394 colunas; *train identity*, com 41 colunas; *test transaction*, com 393 colunas; *test identity*, com 41 colunas. Para o estudo, os datasets identity e transaction foram unidos por meio de um *merge*, orientado pela *feature* TransactionID (comum em ambos *datasets*).

Ademais, observa-se que o *dataset* apresenta classes altamente desbalanceadas. Um exemplo claro disso é observado na variável alvo "isFraud", a qual, no treino, possui 569875 transações legítimas, enquanto há 20663 transações fraudulentas, que representam apenas 3.62% do total de transações.

IV. Metodologia

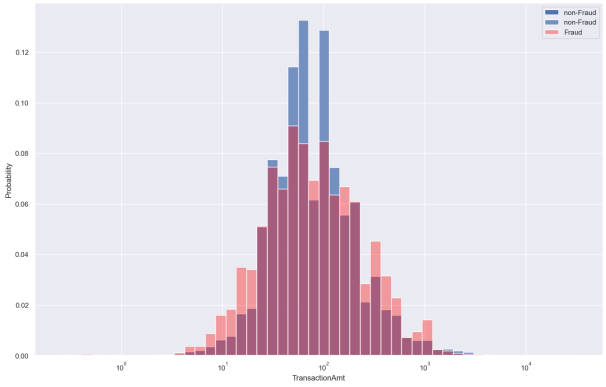
O presente trabalho tem como objetivo mitigar o problema de detecção de fraude em cartões de crédito em compras online, para tal será utilizado o *dataset* descrito no capítulo anterior, junto a modelos de aprendizado de máquina. Durante este capítulo serão apresentadas as métricas de avaliação para tais modelos, como foi executado o pré-processamento dos dados e como foi realizado o treinamento e otimização dos hiperparâmetros dos modelos escolhidos.

A. Análise Exploratória de Dados (EDA)

Como de praxe, foi realizada uma EDA do *dataset* supracitado, com intuito de entender a fundo as *features* encontradas neste desafio.

Para isso, primariamente, foram traçados histogramas correlacionando a variável-alvo "isFraud" com algumas features, sendo elas: TransactionAMT (valor da transação) e TransactionDT (tempo de duração desde um tempo especificado pela fornecedora do *dataset* até a realização da transação). Abaixo, os histogramas descritos anteriormente:

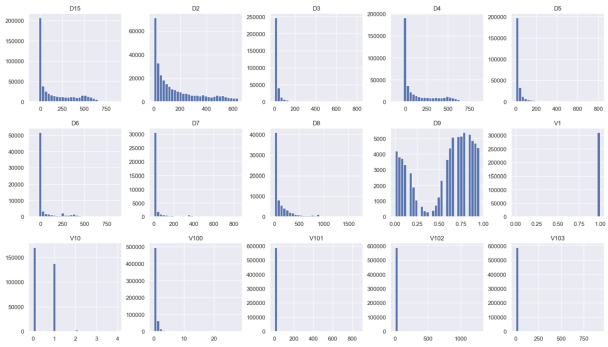
'TransactionAMT' x 'isFraud'



'TransactionDT' x 'isFraud'

Posteriormente, foi traçado um histograma para cada *feature*, os quais continham a frequência/contagem de cada amostra única. A partir disso, foi possível identificar que o dataset, além de ser desbalanceado, possuía uma gama imensa de outliers. Contudo, tais *outliers* foram mantidos para predição do modelo, visto que o modelo deseja generalizar a predição para quaisquer dados reais. Abaixo, uma noção geral dos histogramas:

Frequência x Amostra única



B. Métricas de Avaliação

Para o estudo de como o modelo se comporta, foram utilizadas diferentes métricas, visando obter uma melhor abordagem e entendimento dos resultados obtidos, são elas: ROC-AUC Score e acurácia balanceada.

Para a AUC-ROC Score, tem-se que a implementação desta é adequada para problemas de classificação binária ou multiclasse, em que haja interesse em calcular a área abaixo

da curva ROC. Ademais, para construírmos a curva ROC, necessita-se calcular a taxa de falso positivo (FPR) e a taxa de verdadeiro positivo (TPR).

$$\text{True Positive Rate (TPR)} = \frac{TP}{TP+FN}$$

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP+TN}$$

Para a acurácia balanceada, tem-se que esta não é influenciada pelo desbalanceamento das classes, visto que os cálculos se dão em cima da taxa de verdadeiros positivos (VP) e verdadeiros negativos (VN).

$$\text{Acurácia Balanceada} = \frac{1}{2}*(\frac{VP}{VP+FN} + \frac{VN}{VN+FP})$$

C. Pré-processamento

O pré-processamento pode ser dividido em 4 etapas, sendo elas:

- *Feature Engineering*
- Substituição de valores faltantes
- Aplicação de transformação de features
- Transformação das *features* com OneHotEncoder
- Remoção de *features* numéricas altamente correlacionadas

Na etapa de *Feature Engineering* foi realizado o agrupamento de atributos categóricos com elevado número de valores unicos, devido ao elevado aumento destes atributos no custo de processamento. Além disso, optou-se por descartar as colunas id-31 e id-33. As quais tinham informações repetidas ou aspectos não considerados relevantes pelo grupo.

Já na substituição de valores faltantes, a etapa foi realizada de maneira distinta para dados numéricos e categóricos. De tal forma, para dados categóricos, houve a substituição pela *string* 'Unknown'. Para os dados númericos foi adicionada na *pipeline* uma etapa de SimpleImputer em que será atribuído a mediana para os valores faltantes.

Na etapa de transformação dos atributos está contemplado tanto OneHotEncoder quanto SimpleImputer, sendo o primeiro para dados categóricos e o segundo para dados numéricos.

Na etapa de remoção de *features* altamente correlacionadas, foi utilizada a função df.corr(), apenas nas colunas de dados numéricos. Foi feita uma busca e removação para colunas com uma correlação maior que 0.95, resultando na remoção de 130 atributos.

D. Treinamento

1) Baseline

Em *machine learning*, a *baseline* consiste em um modelo básico, o qual serve como ponto de referência. Sendo assim, a *baseline* serve como o mínimo de acurácia alcançada, servindo apenas para comparar a performance de modelos futuros. Para este trabalho, como *baseline*, foi utilizado um modelo de Random Forest (Floresta Aleatória), com alguns hiperparâmetros pré-definidos para evitar *overfitting*.

Random Forest	Hiperparâmetros
min-samples-split	10
n-estimators	50
n-jobs	-1
class-weight	'balanced'

2) Modelos e Otimização de Hiperparâmetros

Além do modelo baseline de Random Forest, no estudo, o *dataset* também foi treinado utilizado outros modelos. Dentre os outros modelos treinados temos: XGBoost e LightGBM, escolhidos devido a alta performance que apresentaram em outros trabalhos utilizando o mesmo *dataset*.

Para a otimização dos hiperparâmetros, tem-se que está ocorreu por meio do método de *random search*, utilizando a função RandomizedSearchCV, a qual define a escolha dos hiperparâmetros aleatoriamente e treina o modelo com esta escolha. Tal método foi escolhido devido ao tamanho do *dataset* IEEE-CIS Fraud Detection, que é muito grande, o que exigiria muito de processamento caso quiséssemos testar todos os hiperparâmetros. Abaixo, os hiperparâmetros definidos para a função RandomSearchCV.

RandomizedSearchCV	Hiperparâmetros
cv	3
n-iter	35
scoring	['balanced-accuracy', 'roc-auc']
refit	'roc-auc'
n-jobs	-1
return-train-score	True

A seguir serão apresentados a faixa de hiperparâmetros escolhida para a otimização de cada um dos modelos, e qual a combinação que gerou o maior ROC-AUC Score. Obs.: em todos utilizou-se random_state=42 e n_estimators=50 (e apenas no LightGBM e no Random Forest class_weight='balanced').

Sendo assim, os hiperparâmetros para os modelos Random Forest, XGBoost e LightGBM, respectivamente:

Random Forest	Faixa de Valores
ccp_alpha	np.logspace(-4, -2, 5)
criterion	['entropy', 'gini']
max_depth	int(np.logspace(3, 7, 5, base=2))
max_features	['sqrt', 'log2']
min_samples_leaf	np.logspace(-5, -2, num=4)
min_samples_split	np.logspace(-4, -2, num=4)

Random Forest	Melhores Valores
ccp_alpha	0.0001
criterion	'entropy'
max_depth	32
max_features	sqrt
min_samples_leaf	0.00001
min_samples_split	0.000464

XGBoost	Faixa de Valores
learning_rate	np.logspace(-4, -1, 5)
max_depth	int(np.logspace(3, 7, 5, base=2))
gamma	np.logspace(-3, -1, 5)
reg_alpha	np.logspace(-4, 2, 6)
reg_lambda	np.logspace(-4, 2, 6)
subsample	np.linspace(0.5, 1, 3)

XGBoost	Melhores Valores
learning_rate	0.1
max_depth	32
gamma	0.1
reg_alpha	0.025119
reg_lambda	0.001585
subsample	0.75

LightGBM	Faixa de Valores
learning_rate	np.logspace(-4, -1, 5)
max_depth	int(np.logspace(3, 7, 5, base=2))
min_split_gain	np.logspace(-3, -1, 5)
reg_alpha	np.logspace(-4, 2, 6)
reg_lambda	np.logspace(-4, 2, 6)
subsample	np.linspace(0.5, 1, 3)

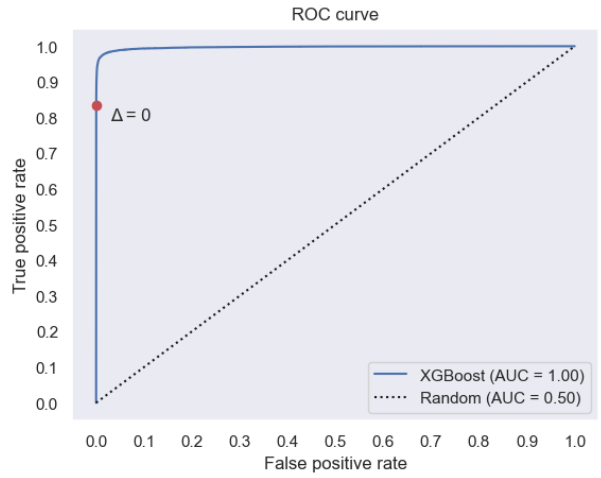
LightGBM	Melhores Valores
learning_rate	0.1
max_depth	16
min_split_gain	0.001
reg_alpha	0.0001
reg_lambda	1.0
subsample	0.5

V. Resultados

Após a otimização de hiperparâmetros e retreinamento no conjunto de validação mais treino completos, foi realizado o cálculo da ROC-AUC Score para o conjunto de teste, obtendo os resultados apresentados a seguir.

Modelo	ROC-AUC	Acurácia Balanceada
RandomForest	94.13%	86.07%
XGBoost	99.72%	91.63%
LightGBM	98.18%	80.86%

Sendo assim, observa-se que o modelo que possui melhor performance é o XGBoost. Dado este fato, a seguir, a curva ROC do modelo:



	Porcentagem
TNR	100.0%
FPR	0.0%
FNR	16.7%
TPR	83.3%

VI. Conclusão

Embora este artigo tenha como objetivo descrever o melhor modelo, o qual desempenha melhor ROC AUC para detecção de fraudes, tem-se que este é um trabalho muito mais complexo do que o que foi abordado.

Nesse artigo, foram utilizados modelos de Random Forest, XGBoost e LightGBM, os quais apresentaram acurácias satisfatórias, contudo, ocultamente, há um problema no *dataset* que não foi observado, o possível vazamento de informações do número do cartão de crédito entre o *dataset* de treino e teste. Avaliando alguns artigos e o melhor modelo proposto no desafio do Kaggle, foi relatado que o modelo, muitas vezes, após uma única fraude na transação, identificava que aquele cartão somente faria transações fraudadas, o que pode não ser uma verdade, porém, como o trabalho já abordava os *datasets* de treino e teste separadamente, tal fator passou desapercebido.

References

[1] H. Najadat, O. Altiti, A. A. Aqouleh and M. Younes, "Credit Card Fraud Detection Based on Machine and Deep Learning," 2020 11th International Conference on Information and Communication Systems (ICICS), 2020, pp. 204-208, doi: 10.1109/ICICS49469.2020.239524.

[2] D. Ge, J. Gu, S. Chang and J. Cai, "Credit Card Fraud Detection Using Lightgbm Model," 2020 International Conference on E-Commerce and Internet Technology (ECIT), 2020, pp. 232-236, doi: 10.1109/ECIT50008.2020.00060.

[3] MOHAN, Arun. IEEE-CIS Fraud Detection - Top 5% Solution. Disponível em:¡https://towardsdatascience.com/ieee-cis-fraud-detection-top-5-solution-5488fc66e9 5f¿. Acesso em: 11/12/2022.

[4] DELAMAIRE, Linda; ABDOU, Hussein; POINTON, John. Credit card fraud and detection techniques: a review. Banks and Bank systems, v. 4, n. 2, p. 57-68, 2009.

[5] DEOTTE, Chris. How to Find UIDs. Kaggle - IEEE-CIS Fraud Detection, 2019. Disponível em:¡https://www.kaggle.com/c/ieee-fraud-detection/discussion/111284¿