Relatório 1: Detecção de Fraudes em Transações Financeiras

Bruna Matias de Lima
Departamento de Computação
Universidade Federal de São Carlos
São Carlos, Brasil
RA:820582

Larissa Dias da Silva
Departamento de Computação
Universidade Federal de São Carlos
São Carlos, Brasil
RA: 800204

Professor: Alan Demétrius Baria Valejo
Disciplina: Aprendizado de Máquina I

I. Introdução

A detecção de fraudes bancárias é um dos principais desafios enfrentados por instituições financeiras. Com o crescimento das transações digitais, torna-se cada vez mais necessário adotar técnicas eficientes para identificar atividades fraudulentas com rapidez e precisão. Este trabalho tem como objetivo aplicar algoritmos de aprendizado de máquina na detecção de transações bancárias suspeitas, utilizando um conjunto de dados com transações rotuladas como legítimas ou fraudulentas.

Neste trabalho, utilizamos o conjunto de dados *"Fraudulent Transactions Prediction"* [**?**], disponibilizado no Kaggle por Vardhan Siramdasu. O dataset contém 6.353.307 registros de transações bancárias.

O projeto foi desenvolvido no Google Colab utilizando a linguagem Python, com o uso das bibliotecas `Pandas`, `Scikit-learn`, `Matplotlib` e `Seaborn`. Seguiram-se as etapas clássicas da modelagem preditiva: exploração de dados, pré-processamento, treinamento e avaliação de modelos.

O notebook está disponível em: https://colab.research.google.com/drive/1w1Q5IQSyK1cOu4-ZMCARSx_uPVu7C5jj?usp=sharing.

II. Análise Exploratória dos Dados

Iniciamos a exploração do dataset verificando suas dimensões e o significado de cada coluna. O conjunto de dados possui 6.362.620 instâncias e 15 atributos, sendo eles:

- `step` (`int64`): hora relativa da transação (em incrementos de 1 hora).
- `type` (`object`): tipo de transação.
- `amount` (`float64`): valor monetário da transação.
- `nameOrig` (`object`): identificador do cliente que inicia a transação.
- `oldbalanceOrg` (`float64`): saldo do remetente antes da transação.
- `newbalanceOrig` (`float64`): saldo do remetente após a transação.
- `nameDest` (`object`): identificador do cliente que recebe a transação.
- `oldbalanceDest` (`float64`): saldo do destinatário antes da transação.
- `newbalanceDest` (`float64`): saldo do destinatário após a transação.
- `isFraud` (`int64`): indicador de fraude (1) ou legítima (0).
- `isFlaggedFraud` (`int64`): flag interna de suspeita elevada.

Percebemos que quatro novas variáveis poderiam agregar informação relevante sobre o fluxo de valores e potenciais anomalias:

```
data['diffOrig'] = data['oldbalanceOrg'] -
    data['newbalanceOrig']
data['diffDest'] = data['newbalanceDest'] -
    data['oldbalanceDest']

data['origZero'] = (data['oldbalanceOrg'] ==
    0).astype(int)
data['destZero'] = (data['oldbalanceDest'] ==
    0).astype(int)
```

Listing 1. Criação de novos atributos

Justificativa para os novos atributos:

- `diffOrig` e `diffDest`: capturam exatamente o montante que saiu da conta origem e entrou na conta destino, respectivamente. Em transações legítimas, espera-se que esses valores coincidam com `amount`, mas discrepâncias podem indicar manipulações ou inconsistências típicas de fraude.
- `origZero` e `destZero`: identificam casos em que o saldo antes da transação era zero. Muitas fraudes são executadas por contas recém-criadas ou com saldo zerado, visando evitar rastreamento ou sistemas de monitoramento tradicionais.

Por fim, verificamos os tipos de dados após a criação dos novos atributos, sendo eles:

- `step`: `int64`
- `type`: `object`
- `amount`: `float64`
- `nameOrig`: `object`

- `oldbalanceOrg`: `float64`
- `newbalanceOrig`: `float64`
- `nameDest`: `object`
- `oldbalanceDest`: `float64`
- `newbalanceDest`: `float64`
- `isFraud`: `int64`
- `isFlaggedFraud`: `int64`
- `diffOrig`: `float64`
- `diffDest`: `float64`
- `origZero`: `int64`
- `destZero`: `int64`

III. Pré-Processamento

Para preparar o conjunto de dados para treinamento, definimos um pipeline em três etapas: *limpeza*, *transformação* e *redução* de atributos.

a) 1. Limpeza: **Verificação de valores nulos e duplicados:** Confirmamos que não havia registros com `NaN` nem linhas repetidas.

Detecção de erros e ruídos: Executamos consultas para identificar valores negativos em atributos críticos, por exemplo:

```
data[data['amount'] < 0]
data[data['oldbalanceOrg'] < 0]
data[data['newbalanceDest'] < 0]
```

Listing 2. Exemplos de checagem de erros

Mesmo tendo encontrado alguns ruídos em nossos dados, optamos por mantê-los, pois comportamentos atípicos podem ser indicativos de fraudes, o que é significante para a nossa etapa de treinamento.

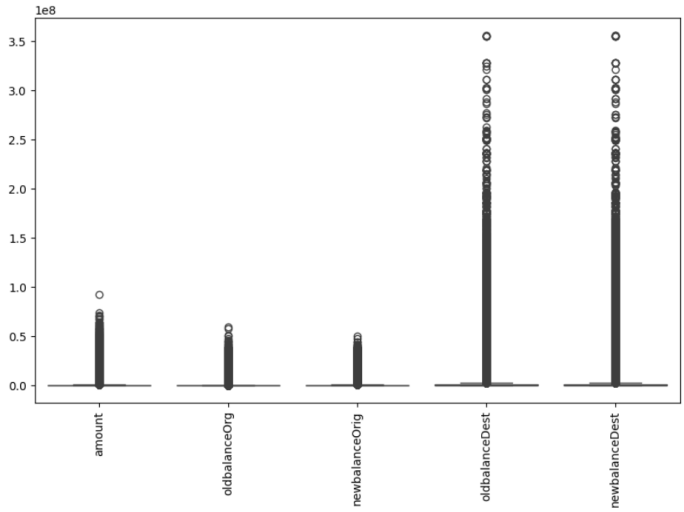

Fig. 1. Distribuição com ruídos identificados nos principais atributos numéricos.

b) 2. Transformação: **Normalização:** Aplicamos *z-scores* para padronizar as variáveis numéricas. Essa métrica é essencial para algoritmos baseados em distância (por exemplo, KNN) e para redes neurais, que se beneficiam de entradas com média zero e variância unitária. **Discretização (binning):** Testamos a criação de 10 faixas com frequência igual, para o atributo amount, mas descartamos esse passo porque a discretização reduz a dispersão natural dos dados, prejudicando a capacidade dos modelos de capturar valores suspeitos.

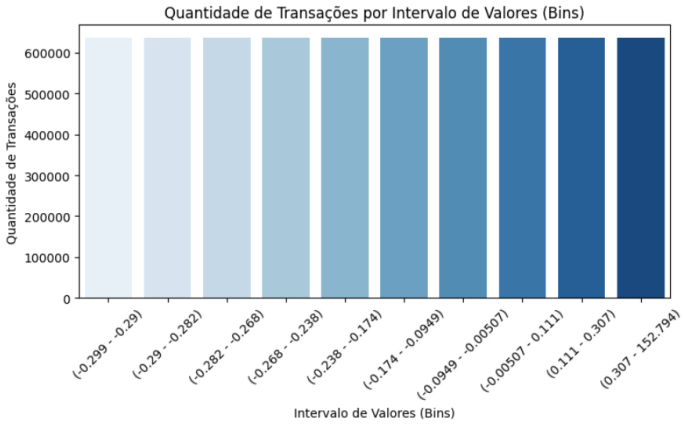

Fig. 2. Discretização em 10 faixas de intervalos com frequências iguais.

c) 3. Redução de Atributos: Para evitar multicolinearidade e reduzir o viés do modelo: Construímos a matriz de correlações e identificamos pares de atributos com correlação muito alta (maior que 0,80), que podem causar *overfitting* e instabilidade nos coeficientes.Avaliamos também a correlação de cada atributo com a variável-alvo (`isFraud`), removendo aqueles com correlação próxima a zero — pouco informativos.

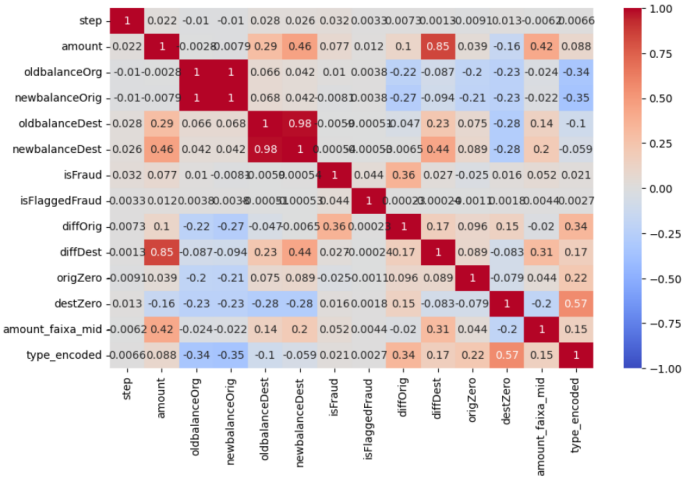

Fig. 3. Matriz de correlação antes da redução.

Assim, descartamos os seguintes atributos redundantes ou pouco úteis:

```
data_model = data_model.drop(columns=[
    'amount_faixa_mid',
    'newbalanceOrig',
    'oldbalanceDest',
    'diffDest'
])
```

Listing 3. Remoção de atributos

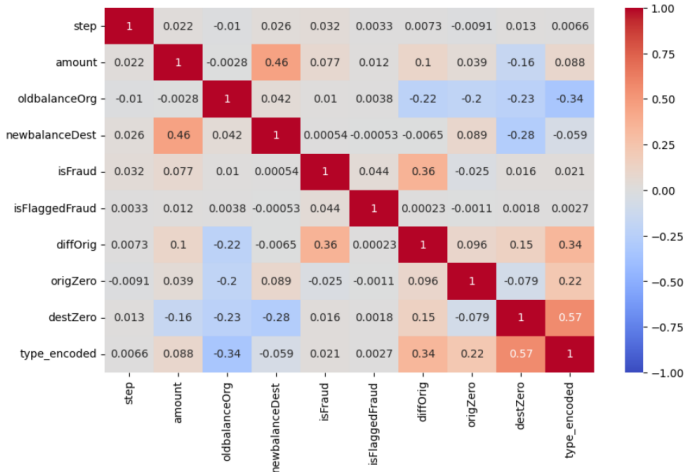

Fig. 4. Matriz de correlação depois da redução.

d) Amostragem Estratificada: Mesmo após remoção dos atributos, o dataset continuava muito grande. Para acelerar experimentos sem alterar o desbalanceamento original, realizamos uma amostragem estratificada de 20% das observações em cada classe:

```
data_reduzido = (
  data_model
    .groupby('isFraud', group_keys=False)
    .apply(lambda x: x.sample(frac=0.2,
        random_state=42))
)
```

Listing 4. Amostragem estratificada mantendo proporções de classe

O dataset resultante do processo de redução possui 1.272.524 instâncias com 10 atributos, preservando a proporção de transações fraudulentas e legítimas para garantir a representatividade em etapas subsequentes de treinamento e validação.

IV. Modelos de Classificação

Modelos de classificação são técnicas de mineração de dados utilizadas para organizar grandes volumes de informações em categorias previamente conhecidas, com base nas características observadas em conjuntos de dados rotulados. O objetivo central é prever a classe alvo de cada instância, permitindo, por exemplo, a distinção entre transações legítimas e fraudulentas. Esses modelos operam por meio de classificadores treinados com dados históricos, sendo amplamente utilizados em aplicações como análise de crédito e detecção de fraudes, como foi o foco deste trabalho (Mangat and Saini, 2022).

Neste projeto, aplicamos três algoritmos com abordagens distintas: o K-Nearest Neighbors (KNN), a Árvore de Decisão e a Rede Neural Artificial (MLP). O KNN classifica uma transação com base nas mais próximas do conjunto de treinamento, enquanto a árvore constrói uma estrutura hierárquica de regras de decisão com base nos atributos. Já a MLP, composta por múltiplas camadas de neurônios artificiais, é capaz de representar funções complexas e não lineares. Redes neurais como essa são extremamente versáteis, podendo aproximar qualquer função computável se corretamente estruturadas e treinadas — o que as torna especialmente eficazes para problemas complexos como a detecção de fraude, mesmo quando o processo não pode ser explicitamente modelado (Wallisch et al., 2014).

Quanto aos resultados, observamos que todos os modelos alcançaram alta acurácia geral, mas se diferenciaram bastante na detecção de fraudes. O KNN, embora conservador e preciso para transações normais, teve baixo recall, exibido na Figura 5, deixando muitas fraudes passarem despercebidas. A árvore de decisão, por outro lado, foi mais agressiva: detectou a maior quantidade de fraudes, mas cometeu mais erros em transações legítimas. A MLP se destacou pelo melhor equilíbrio entre precisão e recall, identificando a maioria das fraudes com poucos falsos positivos — o que a tornou o modelo com melhor desempenho geral neste cenário.

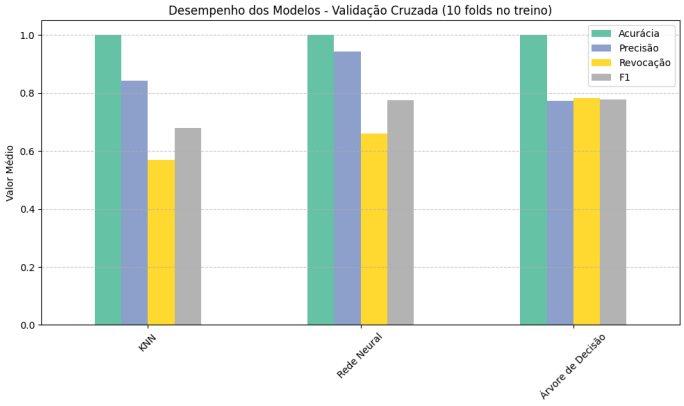

Fig. 5.

V. Avaliação

A avaliação dos modelos de classificação foi realizada utilizando validação cruzada com 10 folds, uma técnica que permite testar a capacidade de generalização dos modelos de forma robusta. No procedimento, o conjunto de dados de treino foi dividido em 10 partes, em que, a cada rodada, nove partes eram usadas para treinar o modelo e uma para validá-lo, repetindo o processo até que todas as partes tivessem sido usadas para validação. Essa abordagem minimiza o viés e a variância que podem ocorrer ao treinar e testar o modelo em uma única divisão dos dados.

Foram calculadas quatro métricas principais para medir o desempenho dos modelos: acurácia, precisão, recall e F1-score (vide Fig. 5). Embora a acurácia tenha sido alta para todos os modelos, isso é esperado devido ao forte desbalanceamento do conjunto de dados, onde a maioria das transações é legítima. Portanto, as métricas mais importantes para a análise foram precisão, recall e F1-score, que refletem a capacidade dos

modelos em identificar corretamente as fraudes, que são uma minoria.

A. Precisão

A precisão indica, dentre as transações classificadas como fraude, quantas realmente são fraudulentas; o recall mostra, entre todas as fraudes reais, quantas foram detectadas pelo modelo; e o F1-score representa um equilíbrio entre essas duas métricas, sendo especialmente relevante em problemas com classes desbalanceadas, como é o caso da detecção de fraudes bancárias.

B. Matrizes de Confusão

As matrizes de confusão médias complementaram a avaliação, fornecendo uma visão detalhada da quantidade de verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos para cada modelo. Essa análise ajudou a entender os tipos de erros cometidos e a agressividade dos classificadores na detecção de fraudes.

1) Matriz de Confusão do KNN: O modelo KNN apresentou um alto desempenho na identificação das transações legítimas, classificando corretamente mais de 88 mil casos em média. Contudo, no que diz respeito à detecção de fraudes, o KNN identificou apenas cerca de 65 casos fraudulentos, deixando escapar aproximadamente 50 fraudes, como exposto na Figura 6, que foram classificadas erroneamente como transações legítimas (falsos negativos). Isso indica que o modelo é conservador, priorizando a minimização de falsos positivos, porém à custa de um recall reduzido, o que implica uma baixa capacidade de identificar fraudes efetivamente.

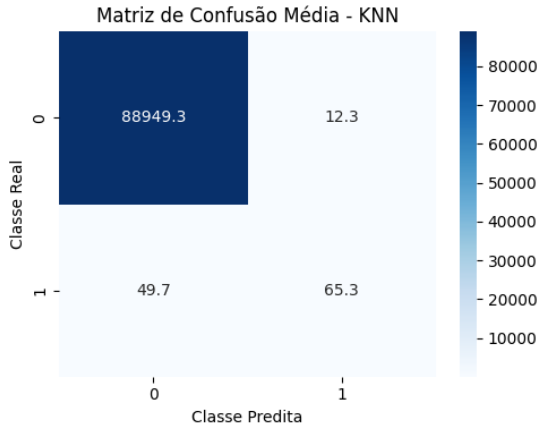

Fig. 6.

2) Matriz de Confusão da Rede Neural: Podemos ver na Figura 7 que a rede neural apresentou um desempenho superior na detecção de fraudes, com cerca de 76 fraudes corretamente identificadas e uma redução significativa no número de falsos negativos em comparação ao KNN. Além disso, o modelo manteve um baixo número de falsos positivos, com apenas cerca de 5 transações legítimas erroneamente classificadas como fraudes. Esses resultados refletem um equilíbrio mais eficaz entre precisão e recall, o que se traduziu no melhor valor de F1-score entre os modelos avaliados.

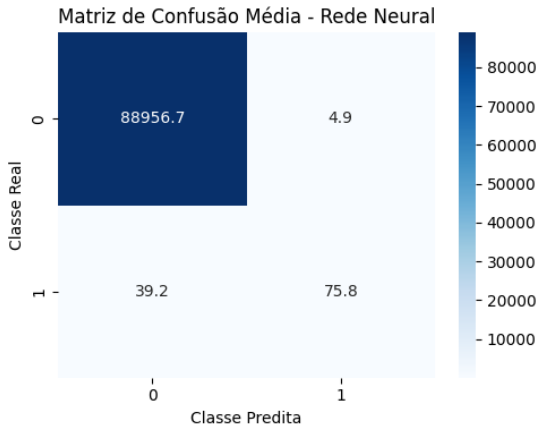

Fig. 7.

3) Matriz de Confusão da Àrvore de Decisão: Segundo a Figura 8, a árvore de decisão foi a mais eficaz na detecção de fraudes, classificando corretamente cerca de 90 casos fraudulentos e apresentando o menor número de falsos negativos, com aproximadamente 25 fraudes não detectadas. Entretanto, esse melhor desempenho no recall ocorreu às custas de um maior número de falsos positivos — cerca de 27 transações legítimas foram classificadas incorretamente como fraudes. Isso evidencia uma postura mais agressiva do modelo, que prefere aumentar os falsos positivos para maximizar a detecção de fraudes.

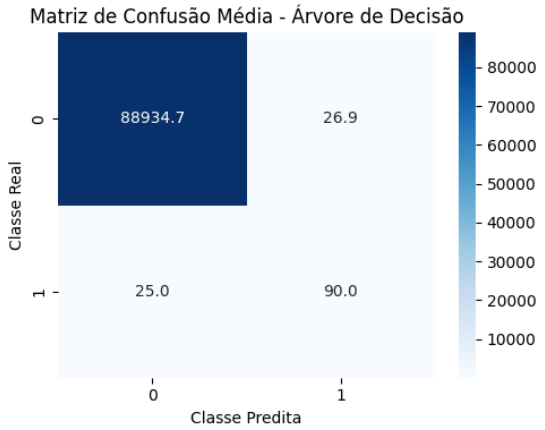

Fig. 8.

C. Ajuste de Hiperparâmetros

Por fim, o ajuste de hiperparâmetros, como o valor de k no KNN, foi realizado para otimizar o desempenho dos modelos, garantindo que os parâmetros escolhidos fossem os mais adequados para o conjunto de dados utilizado. Esse ajuste foi feito usando o GridSearchCV, com a métrica F1-score como critério de otimização.

D. Conclusão da Avaliação

Esse processo de avaliação rigoroso permitiu comparar os modelos de forma justa e confiável, destacando a importância de métricas específicas para detecção de fraude, e mostrando que, para esse problema, métricas como recall e F1-score são mais informativas do que a simples acurácia.

VI. Conclusões

Este trabalho analisou a aplicação de modelos de classificação na detecção de fraudes financeiras, utilizando um conjunto de dados simulado composto por transações bancárias realizadas ao longo de 30 dias (MANCHANDA, 2021). Foram construídos três modelos distintos: KNN, árvore de decisão e rede neural multicamada. A avaliação do desempenho foi conduzida por meio de validação cruzada com 10 folds, utilizando métricas como acurácia, precisão, recall e F1-score.

Os resultados mostraram que, embora todos os modelos apresentassem acurácia elevada — reflexo do desbalanceamento do conjunto de dados —, o desempenho real na detecção de fraudes foi melhor refletido pelas demais métricas. O modelo KNN obteve o menor recall, indicando que deixou de detectar diversas transações fraudulentas. A árvore de decisão alcançou o maior recall, mas apresentou mais falsos positivos, o que compromete a precisão. A rede neural apresentou o melhor equilíbrio entre precisão e recall, resultando no maior F1-score entre os três modelos avaliados.

A utilização da validação cruzada foi fundamental para garantir a confiabilidade da avaliação, ao reduzir a dependência de uma única divisão dos dados. Além disso, o ajuste de hiperparâmetros demonstrou ser uma etapa relevante, especialmente no modelo KNN, cujo desempenho foi sensível à variação de parâmetros.

Conclui-se que a escolha do modelo mais adequado depende não apenas do algoritmo utilizado, mas também da compreensão do problema, do tratamento apropriado dos dados e da aplicação de técnicas rigorosas de avaliação. Em cenários como a detecção de fraudes, nos quais os erros podem ter consequências significativas, o equilíbrio entre sensibilidade e especificidade é essencial. Nesse contexto, a rede neural multicamada se mostrou a abordagem mais promissora.

References

[1] MANCHANDA, Chitwan. Fraudulent Transactions Data. Kaggle, 2021. Disponível em: https://www.kaggle.com/datasets/chitwanmanchanda/fraudulent-transactions-data. Acesso em: 26 maio 2025.

[2] MANGAT, Palwinder Kaur; SAINI, Kamaljit Singh. Relevance of data mining techniques in real life. In: System Assurances: Modeling and Management - Emerging Methodologies and Applications in Modelling, 2022, p. 477-502. Springer. DOI: 10.1016/B978-0-323-90240-3.00026-6.

[3] WALLISCH, Pascal; LUSIGNAN, Michael E.; BENAYOUN, Marc D.; BAKER, Tanya I.; DICKEY, Adam S.; HATSOPOULOS, Nicholas G. MATLAB for Neuroscientists: An Introduction to Scientific Computing in MATLAB. 2. ed. Amsterdam: Academic Press, 2014. DOI: 10.1016/C2009-0-64117-9.