http://ensaios.usf.edu.br - ISSN 2595-1300

# DESENVOLVIMENTO E ANÁLISE DE MOTOR ANTIFRAUDE PARA PAGAMENTOS ONLINE

*DEVELOPMENT AND ANALYSIS OF AN ANTI-FRAUD ENGINE FOR ONLINE PAYMENTS*

ALVES DO NASCIMENTO, Felipe[1]; ANDRIJAUSKAS, Fabio[2]

[1]Graduando do Curso de Engenharia da Computação – Universidade São Francisco;
[2]Professor do Curso de Engenharia da Computação – Universidade São Francisco
**fabio.andrijauskas@usf.edu.br**

**RESUMO**. As compras *onlines* e as transações digitais estão cada vez mais presentes no dia a dia das pessoas. Por consequência, surgiram também novos comportamentos fraudulentos, visando o abuso de tais meios de pagamentos. Neste contexto, este estudo teve por objetivo desenvolver e demonstrar um fluxo de prevenção a esse tipo de fraude, utilizando-se de técnicas de *machine-learning (KNN)*, conjuntamente com o modelo de motor baseado em regras, que mostre desde o desenvolvimento inicial até o fluxo de dado transacional, evidenciando também, a análise dos dados criados. Com isso, foi possível realizar previsões e desenvolver regras para a prevenção dos diversos tipos de comportamentos fraudulentos, e não somente contra aqueles baseados em situações já conhecidas, mas também em cenários inéditos. Assim, torna-se viável a criação de aplicativos e plataformas de pagamentos confiáveis e seguros, tanto para o meio empresarial quanto para o usuário final.

**Palavras-chave**: *machine-learning*; motor de regras; *KNN*, fraude.

**ABSTRACT.** Online shopping and digital transactions are increasingly present in people's daily lives. Consequently, new fraudulent behavior has also emerged aimed at abusing such means of payment. The objective of this work is, therefore, the development and demonstration of a prevention flow for this type of fraud using machine-learning (KNN) techniques, together with the rule-based engine model, which shows from initial development to the transactional data flow, also demonstrating the analysis of the created data. With this, it is possible to make predictions and develop rules for preventing different types of fraudulent behavior, against those based on known situations and unprecedented scenarios. Thus, it becomes feasible to create reliable and secure payment applications and platforms for both the business environment and the final user.

**Keywords**: *machine-learning, rules engine, knn, fraud*

## INTRODUÇÃO

A transformação digital pela qual passava o mundo foi fortemente acelerada com a pandemia da Covid19, que tomou força a partir de 2020. A consequência foi um significativo aumento do comércio online, que passou a ser uma realidade presente na maioria da população: desde a compra de itens de valores expressivos, como eletrodomésticos e aparelhos eletrônicos, até os mais básicos, como os de higiene pessoal e refeições diárias. Por conseguinte, também se notou expressivo aumento dos casos de fraudes em pagamentos: “A taxa de ataques e pagamentos em *fintechs* aumentou em 70% em 2021” (*helpnetsecurity*, 2022). Ao contrário do que se pode deduzir, tais ocorrências não são provenientes de atividades de *hackers*, mas sim de pessoas comuns, que abusam de promoções, *chargeback* - prática que consiste em pedir estorno da compra -, dentre outras falhas de segurança. Desse modo, estudar a aplicação de ferramentas que atenuem tais práticas abusivas é de suma importância para o mercado, já que possibilita o aprimoramento considerável das técnicas utilizadas, refletindo diretamente no preço dos produtos para o consumidor final. O sistema

antifraude divide-se em dois grandes grupos: Motor de Regras, ou *Rule-Based Fraud Detection*, e *Machine Learning*.

Motor de Regras (*Rule-Based Fraud Detection*) consiste em um *software* robusto que, ao receber uma requisição (uma chamada de uma *API*), transmite esse dado por uma série de regras, podendo, no final, aprovar ou não a requisição, dependendo das pesquisas realizadas anteriormente pelo time que o desenvolveu. Para uma regra de detecção de fraude com cartão de crédito, por exemplo, a análise pode ser feita com base na localização da transação, bloqueando quaisquer operações que venham de um CEP suspeito. Outra possibilidade é o bloqueio de cartões que estão sendo utilizados em frequência anormal, tal como uma compra a cada curtos períodos (LIGEZA, 2006). Apesar de algumas dessas regras executarem sua função com excelência, outras tantas são responsáveis por muitos falsos alarmes, que podem ocorrer por diversos motivos: desde uma implementação inadequada, até não fazer sentido no contexto em que foi inserido. As maiores limitações dos sistemas baseados em regras são: limite fixo por regras; regras que não se adaptam ao comportamento fraudulento, que é dinâmico; dificuldade de estabelecer esse limite; resposta em "sim" ou não"; dificuldade em estabelecer os números de falsos positivos e negativos aceitáveis; incapacidade de capturar a interação de recursos. Supondo que, para determinar transações fraudulentas, o tamanho de certa transação só importe em combinação com a frequência, um sistema baseado em regras não conseguiria lidar com isso. Sendo assim, sua aplicação só pode ser realizada de maneira eficiente de acordo com o contexto em que está inserido, ou seja, onde as citadas limitações não sejam um problema crítico.

Por outro lado, os modelos *Machine Learning* não apresentam as limitações anteriormente citadas, sendo que possuem respostas ao comportamento fraudulento em forma de probabilidade, possibilitando o refinamento de acordo com a quantidade de respostas obtidas; tornam possível a definição do número aceitável de falsos positivos e negativos; adaptam-se aos novos dados, podendo, portanto, detectar novos comportamentos fraudulentos (MALINI, 2017). Essas características tornam sua aplicação mais adequada em ambientes com muitos dados. Nesse modelo, o problema é resolvido de maneira diferente. Como, normalmente, para executar alguma operação no computador, desde somar dois números até a construção de um motor de regras antifraude, é necessário descrever minuciosamente o que precisa ser feito, modelos de *machine learning* resolvem de maneira distinta, descobrindo sozinho o que precisa ser feito através da inferência de dados e, quanto maior a quantidade de dados, melhor seu comportamento (DOMINGOS, 2015).

Na Ciência da computação, entretanto, tudo depende do ambiente em que se trabalha, embora os modelos *Machine Learning* possuam uma eficácia maior, eles não são capazes de resolver todos os problemas. Algumas regras, mesmo as relativamente simples, são capazes de desempenhar de maneira consideravelmente melhor a detecção de fraudes. Conclui-se, portanto, que o melhor resultado é obtido quando há a combinação dessas duas tecnologias. Neste caso, aprimorar as técnicas de detecção de comportamentos fraudulentos torna-se uma questão de suma relevância para que o comércio digital possa continuar crescendo e evoluindo, sem que seja prejudicado por atividades de pessoas mal-intencionadas. Sendo assim, o presente estudo teve como objetivo observar e analisar os resultados obtidos através de motores de regras e *machine learning* quanto à detecção de falhas e, por conseguinte, verificar a melhor solução para que esta detecção seja aprimorada com o menor fator de erro possível. A detecção de fraude através de modelos baseados em regras possui baixa eficiência de processamento de dados, problema este que foi pensado pelo autor Wang (2017), que buscou trabalhar e desenvolver um motor de regras simples e eficaz, mas que pudesse ser usado em transações bancárias. Quando a amostra em questão possui um vasto número de dados, é possível realizar uma busca mais criteriosa e detalhada, além do uso de regras específicas e da combinação das informações obtidas. As funções do mecanismo de regras projetado incluem interface de aquisição de dados de transação, interface de armazenamento de dados de detecção, interface de serviços sobre mecanismo de regras, gerenciamento de regras, banco de dados de regras e correspondência de regras.

Para a resolução de um problema semelhante ao que foi abordado com o motor de regras, ou seja, pagamentos *online* com cartão de crédito, a autora Malini (2017), utilizou uma abordagem diferente, através de um algoritmo conhecido como *KNN (K-Nearest Neighbor*, ou, em português, K-vizinho mais próximo). Este é um dos mais populares algoritmos para utilização em *machine learning*, apresentando menor complexidade em comparação aos demais semelhantes a ele (YONG, 2009). Uma das características mais marcantes do *KNN* é poder ser utilizado tanto para classificação quanto para regressão. Seu funcionamento parte de uma variável chamada 'K' e o valor a ela atribuído definirá a quantidade de vizinhos. Por exemplo: ao se escolher um valor 'P1' que se necessita predizer, entre um grupo de duas classes, onde o valor atribuído a 'K' foi 1 (K=1), primeiro, é preciso identificar o ponto mais próximo a ele e, depois, qual marcação o identifica. Após identificar o ponto mais próximo e a marcação, é necessário a realização de uma votação para que a maioria diga a qual classe o ponto 'P1' realmente pertence (LUZ, 2019),

Em suma, a ideia principal do algoritmo *KNN* consiste em realizar previsões baseadas em dados pré-existentes. Desse modo, quando aplicado ao modelo antifraude apoiado no motor anteriormente citado, a situação seria análoga à previsão para a criação de uma regra mais próxima a um grupo de regras que resolva determinado problema e, como consequência, com características mais adequadas à detecção do comportamento fraudulento.

A autora Malini. 2017 aponta que as transações fraudulentas são muito próximas a qualquer outro pagamento comum. E, como esse tipo de comportamento sofre adaptações, surgem algumas dificuldades, tais como: o caráter sigiloso, já que os bancos não disponibilizam os dados nem sequer para testes; o comprometimento dos resultados obtidos, caso sejam adotados dados criados para fins de estudos que, por mais próximos à realidade, ainda assim, não correspondem totalmente a ela; o número pequeno de amostras, já que a quantidade de operações fraudulentas é bem menor do que as lícitas, tornando a obtenção de dados tarefa ainda mais árdua. A escolha do *KNN* pela autora, portanto, leva em consideração todos os problemas citados e como será possível resolvê-los. A detecção de fraudes é uma função de alta complexidade a ser executada, e não existe um sistema que possa garantir uma taxa de resultado com 100% de satisfação (MALINI, 2017). Esse modelo é interessante por fazer previsões de comportamentos fraudulentos e conseguir bloquear compras ilegais mesmo antes de sua ocorrência.

Outro modelo utilizado para a detecção de fraudes é conhecido como *HMM - Hidden Markov Model* - cuja abordagem foi feita pelo autor Shailesh (2012). *Softwares* desenvolvidos com base em modelos *HMM* são comuns em áreas como reconhecimento de fala, bioinformática e genômica, e podem ser definidos como um conjunto de vários elementos finitos. Nessa abordagem de estados, as transações são governadas por probabilidades de transição. Entrando, se, em cada estado, uma saída ou observação puder ser gerada de acordo com uma distribuição de probabilidade, ao invés das regras determinísticas, e se somente a saída ou observação (e não o estado que a gerou) for visível a um observador externo ao processo, então os estados estarão "escondidos" do exterior. Daí o nome Modelos de Markov Escondidos 1,4,5,6,7 (Hidden Markov ModelHMM)" (Andrade, 2000). A abordagem do autor Shailesh (2012) para a detecção de fraude utiliza a criação de *clusters* de conjunto de treinamento e identifica o perfil de gastos do titular do cartão, o número de itens comprados e os tipos de itens comprados em uma determinada transação que não são reconhecidos pelo sistema de Detecção de Fraudes. Esse sistema procura identificar qualquer variação no comportamento do proprietário do cartão de crédito, tal como o local da compra, o horário, a quantidade de itens comprados em um determinado período etc. As probabilidades do conjunto inicial foram escolhidas com base no perfil comportamental de gastos do titular do cartão e constroem uma sequência para processamento posterior (SHAILESH, 2012). Em síntese, o sistema busca por alterações bruscas no comportamento de usuários, seguindo a linha lógica: início do programa; login; compra; fornecimento dos dados do cartão de crédito; chamada do sistema antifraude, que é uma etapa "via de mão dupla", pois pode tanto fazer a verificação quanto realizar chamadas de um para o outro, e o sistema pode ou não continuar para a próxima; transação de pagamento; encerramento. Nesse caso, o autor optou por demonstrar o funcionamento do sistema de

pagamento como um todo, deixando a parte da detecção de fraudes como mais um processo a ser executado, semelhante ao que ocorre em pagamentos reais.

## METODOLOGIA

A escolha do desenvolvimento desse trabalho constitui-se em unir as ideias propostas por diferentes autores sobre motor baseado em regras e rede de inteligência artificial *KNN* , além de desenvolver um motor de regras que alimenta uma rede *KNN*. Assim, é possível demonstrar um fluxo completo de prevenção à fraude em pagamentos *online*. Ao utilizar esse método, o processo torna-se mais nítido e de fácil rastreamento, já que todo o fluxo é detalhado pelos micros-serviços que os compõem. Além disso, os gráficos gerados pela rede *KNN* (*Scikit-learn* (COURNAPEAU, 2021) representam fielmente o que foi retratado e, já que foram desenvolvidos para serem de fácil compreensão, somando-se ao fato de o projeto seguir a arquitetura *micro-service*, tem-se uma aplicação de maneira muito facilitada em situações reais.

Dados de compras são e devem ser sigilosos e protegidos por leis, como a Lei Geral de Proteção a Dados (LGPD) e, por questões de segurança, nenhum estabelecimento deve fornecê-los. Desta maneira, o presente projeto se utiliza de um ambiente de testes baseado na ferramenta *Postman, software* que gera e permite criar dados aleatórios e executa chamadas com todos os métodos *HTTP* a *APIs* (ASTHANA, 2022), onde a chamada principal, contém: nome, *e-mail*, documento, dados do cartão, endereço, data de nascimento e valor da transação.

Os conteúdos descritos após os dois pontos e entre chaves – exemplo: ‘document’: {{cpf}} - fazem parte do ambiente de teste e servem para gerar valores aleatórios, como “{{cpf}}”, por exemplo, que gera um número com características de um CPF real para fins de testes e assim sucessivamente para todos os demais atributos, como descrito na Figura 5. Esse documento está em formato *JSON* (*Javascript Object Notation)* e serve para simular os dados de uma compra. Esse dado abstrai todo o processo de um pagamento real - por exemplo: processo de escolha de produto, checagem e tentativa de pagamento - chegando à tomada de decisão final que aprova ou reprova a transação. Esse serviço denominado *tcc_random*, nome para se referir a qualquer sistema aleatório que recebe uma requisição com dados de pagamento, representado por um paralelepípedo, é um sistema de orquestração. Este recebe um determinado dado e escolhe para qual sistema ou banco esse dado deve ser enviado, em um fluxo de produção. Comumentemente, sua posição é logo após a checagem de pagamento. Então, essa *API*, do inglês Interface de Programação de Aplicação, recebe a chamada com todos esses dados de pagamento já processados para, assim, iniciar a análise de fraude. Posteriormente, esse conteúdo é enviado para um *software* de ‘tomada de decisão’ que, no caso desse projeto, foi escolhido um motor baseado em regras.

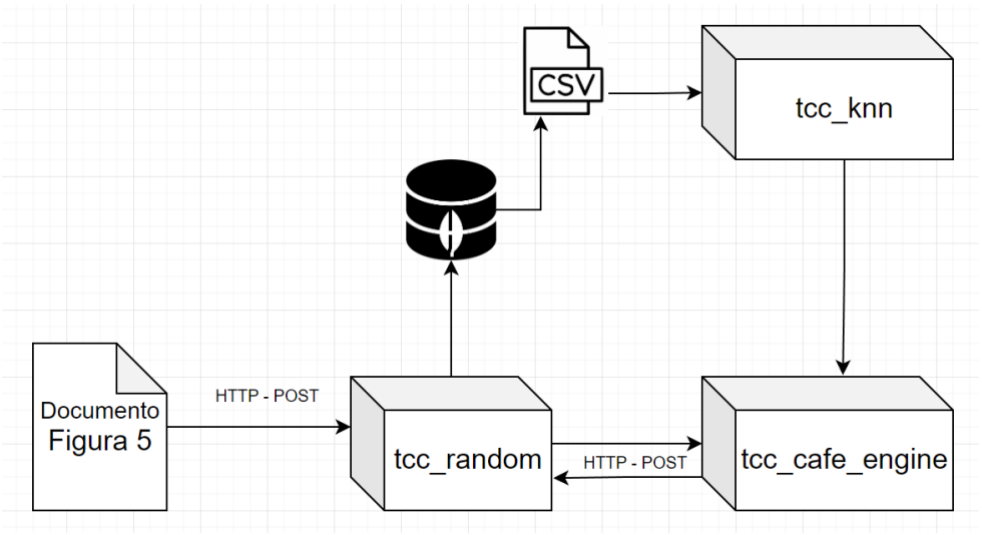
Neste fluxo, quem toma a decisão final da aprovação da compra é o motor (*tcc_cafe_engine* – motor de regras). As regras que o compõem devem ser sempre simples e diretas, para assegurar a fácil manutenção e possibilitar sua escrita não só por engenheiros de *software*, mas por qualquer pessoa que tenha o conhecimento dos dados transacionais. Uma vez que a decisão é tomada – o que não leva mais de um segundo – o motor faz uma chamada de volta para a *API (tcc_random)*, mas com um campo a mais, chamado de *transactionStatus*, com o valor de “*approve”* ou “*repprove*” que representa a decisão tomada pelo motor. Todo esse conteúdo é salvo num banco de dados (MongoDB)

Uma vez que esse dado se encontra no banco, já é possível realizar diversos tipos de tratamentos e consultas, além de algumas conclusões a partir de uma interface gráfica como *express mongodb.* Possuir somente essa métrica como parâmetro de decisão para a criação de futuras regras, entretanto, é uma tarefa árdua porque, com o crescimento dos dados, ela torna-se inviável, já que a base da pesquisa é feita a partir da análise do indivíduo que realiza a leitura dos dados. Para isso, a abordagem utilizada pelos autores mencionados é utilizar uma rede de inteligência artificial.

Ao utilizar essa rede, as métricas obtidas são direcionadas para tomar a melhor decisão. Essa inteligência artificial será ‘alimentada’ a partir de um documento *CSV* (do Inglês: Dados Separados por Vírgula) que será gerado a partir dos dados presentes no banco de dados não relacional. Uma vez

gerado, tal documento é enviado à rede *KNN*. Nesse projeto, por questões didáticas, esse processo é feito manualmente, mas poderia ser facilmente automatizado durante o processo de *deploy* da aplicação. Com isso, tomar uma decisão fica muito mais fácil, já que, com a própria rede, torna-se possível gerar gráficos e fazer previsões de futuros comportamentos fraudulentos que ainda não foram detectados. Dessa maneira, surge a possibilidade de serem escritas novas regras para o motor. O processo fica como o representado na Figura 1.

**Figura 1 –** Fluxo completo do projeto




Fonte: Próprio autor.

Todo esse sistema foi desenvolvido localmente, porém, utilizando-se de mecanismos que facilitam consideravelmente o processo de distribuição, caso seja necessário fazer um *deploy* em situação real, uma vez que este projeto está baseado em tecnologias como *container* facilitando a instalação (HYKES 2022).

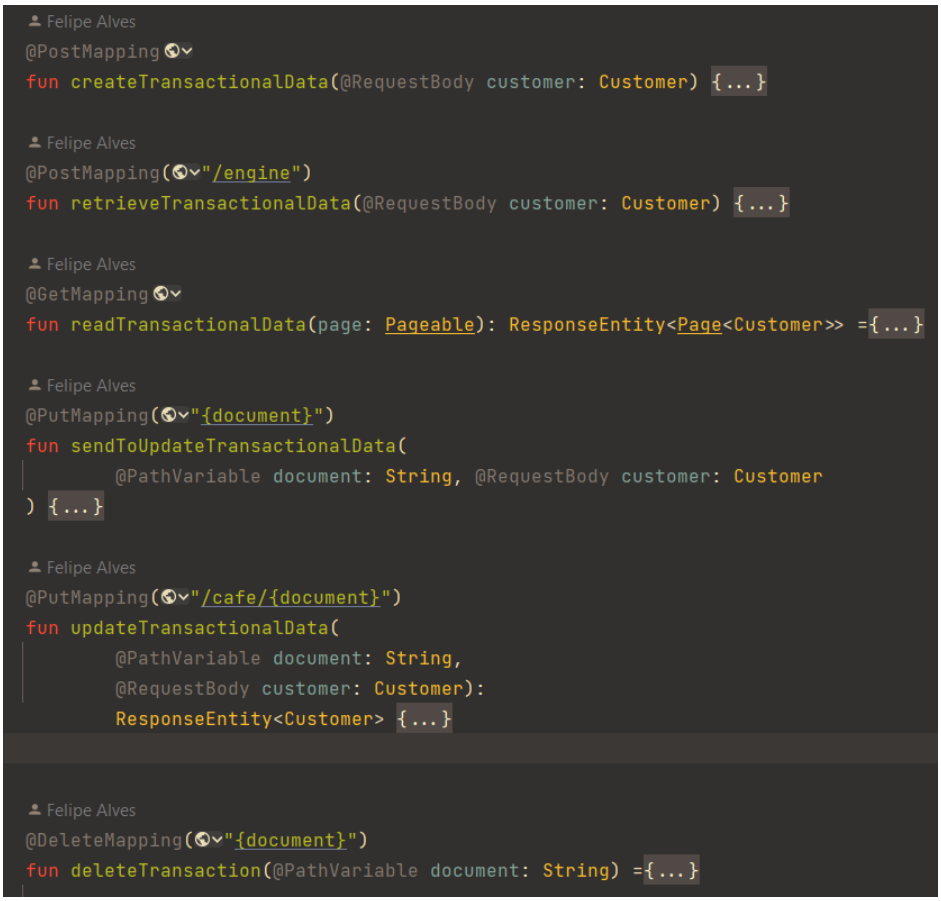
## RESULTADOS E DISCUSSÃO

*Motor de regras*

A implementação do banco de dados é um MongoDB do tipo não relacional. Para o fluxo transacional, o desenvolvimento foi feito em um *container Docker*, como pode ser visto no link pipe-felipe/tcc_mongodb: Tcc database (github.com). O repositório conta com mais dois *scripts* que têm a função de iniciar e parar o serviço. Já a *API* (tcc_random) pode ser acessada neste *link:* pipe-felipe/tcc_random (github.com). Este serviço é um orquestrador, como citado anteriormente, e depende diretamente do banco de dados tcc_mongodb. O sistema possui dois *endpoints*, que são os caminhos por onde pode ser feita uma chamada *http*. O núcleo do sistema é feito na classe '*controller'*. A estrutura desse núcleo é a implementação dos métodos contidos no *controller*. Como este projeto está rodando de maneira local, a base dos *endpoints* será o *localhost*. Ela pode, porém, sofrer alteração, a depender de onde a aplicação está rodando. O primeiro *endpoint* é o *localhost:8080/customer*, que executa os métodos *POST*, implementado por *createTransactionData* e preparado para receber os dados de um outro serviço (tcc_cafe_engine e fluxo transacional anterior ao antifraude); *GET*, implementado por *readTransactionData*, onde podem ser acessados todos os dados transacionais em formato *JSON; PUT*, implementado por *updateTransactionalDatal*, utilizado para atualização de cadastro; *DELETE*, implementado por *deleteTransaction* e com finalidade de apagar dados cadastrados. O segundo

*endpoint* é o *localhost:8080/engine*. Ele possui apenas um método, o *POST*, implementado por *retrieveTransactionalData* e que recebe os dados tratados provenientes do motor de regras. As assinaturas dos métodos podem ser vistas na Figura 2.

**Figura 2 –** Métodos principais TCC_RANDOM




Fonte: Próprio autor.

Quando recebe uma chamada em */customer*, a *API* espera uma resposta em */engine*, feita pela terceira implementação exigida por este projeto. Tal qual *tcc_random*, o *cafe_rules_engine* possui um *endpoint*, disponível em *http://localhost:8081/cafe*. Ele espera receber um dado transacional, Também espera obter uma requisição do tipo *POST* que, ao receber os dados, inicia o processamento das regras para análise de possíveis atos fraudulentos. Antes de devolver tais dados ao orquestrador, eles são passados por um método que realiza o tratamento ao utilizar-se de um conjunto de regras.

Uma vez que esses dados estão salvos, é possível gerar um documento *CSV* (*Comma-separated values*, do inglês; valores separados por vírgula. Este documento é utilizado como entrada de dados para a rede de inteligência artificial *KNN* que, baseando-se no conteúdo apresentado, é capaz de fazer previsões para que futuras regras sejam desenvolvidas.

A modelagem desses dados foi feita seguindo todas as etapas contidas nesses resultados, ou seja, fazendo várias chamadas aos serviços em execução (banco de dados; *tcc_random; tcc_cafe_engine)*, com dados que serão aprovados ou reprovados pela regra. Após fazer 1178 chamadas, o banco possui 1178 dados com conteúdo reprovados e aprovados. O documento exportado em *CSV*, com o nome *'customer.csv'*. Até aqui, foi descrito todo o processo de prevenção a fraudes utilizando o motor baseado em regras. Porém, a criação de novas regras através dos dados gerados torna-se praticamente impossível, já que a tendência é que eles cresçam de maneira exponencial, dificultando em demasia a sua análise.

*Inteligência artificial - KNN*

Em posse do documento *customer.csv*, é possível 'alimentar' a rede de inteligência artificial. O *KNN* foi utilizado por seu comportamento adaptável ao meio em que está inserido. Ao analisar, por exemplo, uma grande quantidade de dados gerados no processo transacional, ele é capaz de, por si só, tomar decisões que auxiliam no processo de criação de novas regras, já que esta rede é baseada na distância euclidiana, que diz que, quanto mais perto o dado vizinho está do dado de referência, maior a chance deste dado ser igual à referência. O código fonte da rede *KNN* pode ser acessado no link https://github.com/pipe-felipe/tcc_knn.

Essa rede foi desenvolvida usando *Python* com *Jupyter Notebook*. O primeiro passo é fazer a importação das bibliotecas necessárias, como *pandas*, para a manipulação e análise de dados, que oferece estruturas e operações para manipular tabelas numéricas e séries temporais; *numpy*, para cálculo numérico; *matplotlib*, para imprimir gráficos matemáticos; *seaborn*, para visualização de dados estatísticos; *sklearn*, para implementação da rede *KNN* de inteligência artificial; bibliotecas do *Google* para acesso ao *Google Drive*, já que o projeto utiliza *Jupyter* dentro do *Google Colab*.

Com os dados carregados no *Jupyter Notebook*, é necessário analisar a estrutura dos dados importados, já que eles devem ser sempre numéricos, pois a inteligência artificial não entende *strings*. Sendo assim, eles devem ser discretizados.

Em alguns casos, é necessária a remoção de algumas tabelas desnecessárias. No *data-set* utilizado pelo projeto, alguns campos foram removidos, como: *_id; _class*, que são valores gerados pelo banco de dados e que não têm impacto algum na análise estatística.

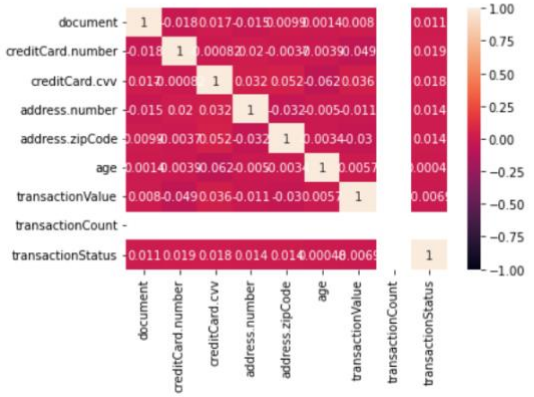
Agora, o ponto mais importante é fazer a transformação do dado mais significativo, o *transactionStatus*, pois esse é o conteúdo que retorna caso a transação tenha tido ou não risco de fraude. Porém, o conteúdo chega ao documento em forma de *string*, ou seja, '*APPROVE'* ou '*REPPROVE'*. Deve-se, então, percorrer-se todas as colunas e transformá-lo em 0 ou 1. Aqui, '0' significa 'sucesso' e '1' significa 'reprovado'.

Quando esse bloco de código é executado, os valores contidos na coluna *transactionStatus* são apenas '0' ou ''1'. Porém, há algumas variáveis que não podem ser transformadas em números, tais como nome, *e-mail* e cidade. Por isso, é necessário criar uma coluna específica para cada característica. Assim, a inteligência artificial consegue se situar e entender o tipo de dado com que ela está lidando. No mundo da computação, as tabelas em '0' ou '1' são conhecidas como *dummy*, ou, em português, fictício, já que bancos de dados assim são de difícil compreensão humana. Um exemplo desse tipo de tabela pode ser visto na coluna *name*, que contém o nome dos indivíduos que realizaram transações. Nela, se um dos nomes for, por exemplo, João, será criada uma coluna chamada 'nome.joão', que será preenchida com valores '0' ou '1', onde o único valor verdadeiro, ou seja, '1', será quando o conteúdo da linha for 'João'. Sendo assim, é possível fazer uma correlação entre as variáveis, para explicar as afinidades entre os atributos. Neste projeto, tais afinidades explicam o fato de as transações retornarem *'REPPROVE'*. Utilizando a biblioteca *seaborn*, é possível construir uma matriz de correlação, tal qual mostrado na Figura 3.

**Figura 3 –** Matriz de correlação

```
sns.heatmap(df.corr(), annot=True, vmin=-1, vmax=1)
plt.show()

# Isso quer dizer que a variável transactionCount explica 0.18% a variável transactionStatus
```




Fonte: Próprio autor.

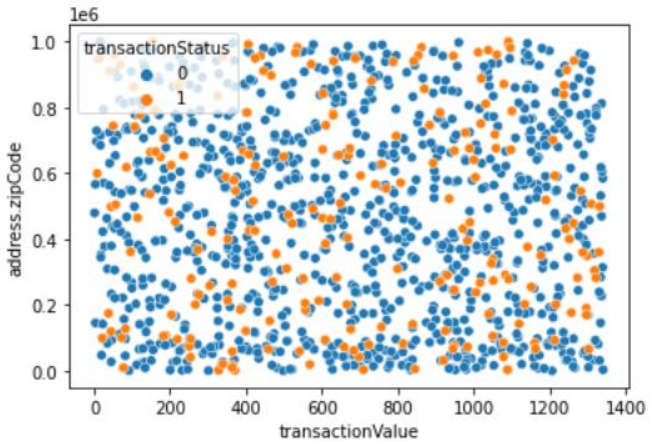
Tal correlação entre as variáveis é a base do *KNN*. A diferença é que o *KNN* irá, além de prever os dados, organizar o que já está disponível. Para fins didáticos, será mostrado um gráfico com valores de baixa correlação - Figura 4 - e outro com alta correlação - Figura 5. A base deste projeto é a criação de uma regra em que todos os indivíduos cujo sobrenome seja 'Furtado' é um ladrão. Além disso, uma outra maneira de ser bloqueado pelo motor é ter o *e-mail* '*ladrao@gmail.com'*. Como o sistema, entretanto, não permite a utilização de um mesmo *e-mail* por mais de um indivíduo, tal bloqueio realizou-se uma única vez.

É importante ressaltar, também, que o algoritmo utilizado cria *e-mails* da seguinte forma: *'nome_tcc@dominio'* e, sendo assim, nomes e *e-mails* são uma excelente maneira de correlacionar os demais dados, já que, se houver a palavra 'Furtado' em qualquer um desses dados, a transação será bloqueada.

**Figura 4 –** Baixa correlação - CEP com valor de transação

```
sns.scatterplot(data=df, x='transactionValue', y='address.zipCode', hue='transactionStatus')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a44fe1650>
```

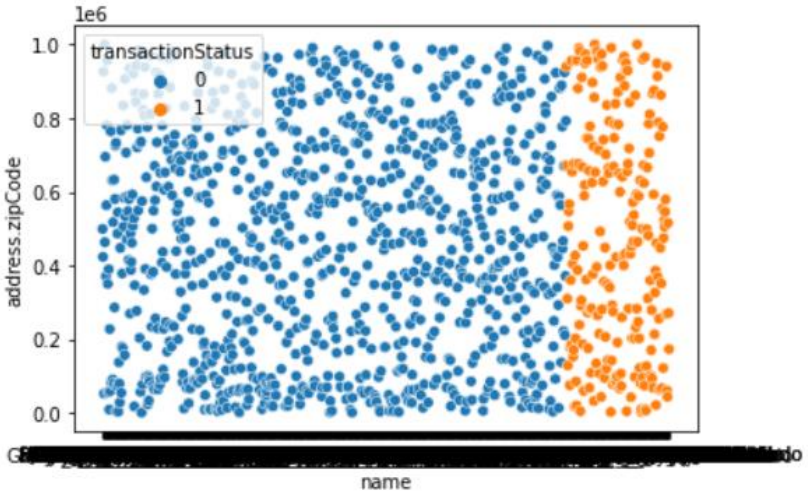



Fonte: Próprio autor.

**Figura 5 –** Alta correlação - nome com alguma outra variável, no caso CEP

```
sns.scatterplot(data=df, x='name', y='address.zipCode', hue='transactionStatus')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a44b6b150>
```




Fonte: Próprio autor.

A correlação não implica em causalidade. Se muitas transações bloqueadas vierem de uma mesma região do país, isso não necessariamente significa que todas as que vierem desse mesmo local serão fraudulentas. Mas, como o modelo do projeto está bastante enviesado, para melhor apresentação, esse tipo de correlação está acontecendo.

Todo esse conteúdo é baseado em dados já existentes, mas ao utilizar uma inteligência artificial, o objetivo é a previsão de dados, para que o sistema seja autônomo. Agora, é necessário fazer uma divisão entre os atributos dos dados da compra e a coluna que precisa ser prevista, no caso, a *transactionStatus*, que possui os valores de 'aprovado' ou 'reprovado'. Aqui, realizou-se a divisão do *dataset* em teste e treino, pois a própria biblioteca de inteligência artificial irá gerar os dados para o teste da precisão da previsão Uma vez normalizado, classificado e treinado, já é possível fazer a previsão dos dados. Conclui-se, portanto, que, com todos os dados utilizados para testes, percebeu-se que todas as pessoas com 'Furtado' no nome ou *e-mail* foram barradas. A inteligência artificial conseguiu ter uma acurácia de 77% de precisão em cenários semelhantes, não analisando apenas o nome e *e-mail*, mas sim todos os dados disponíveis, gerando, assim, uma precisão muito maior na decisão e confecção de novas regras.

## CONCLUSÃO

Conclui-se que, com o contínuo avanço do mundo digital e globalizado, é cada vez mais imprescindível ter sistemas de pagamentos com prevenção à fraude, já que o abuso de comportamentos fraudulentos mostra crescimento constante e significativo e, dessa maneira, estudos para desenvolvimento de *softwares* e análise de dados voltados a esse segmento apresentam cada vez mais relevância. Existem diversas abordagens para a prevenção de fraudes. Mas, no mundo real, não é necessário escolher apenas um modelo: o ideal é fazer a junção deles para que se obtenha o melhor proveito das tecnologias disponíveis. Essa foi a abordagem utilizada neste estudo, ao utilizar um modelo baseado em regras e o de inteligência artificial (*KNN*).

O modelo de regras apresentou um bom funcionamento, com precisão de mais de 70% da inteligência artificial. A arquitetura escolhida para o desenvolvimento das aplicações foi a de micro serviços, que torna possível o acoplamento deste projeto em outros projetos que podem já estar em

execução ou não, já que os sistemas são independentes e cada serviço se comunica um com o outro e não têm, necessariamente, relação de dependência.

Sendo assim, quanto mais dados passam pelo sistema transacional, melhor fica o processo em geral, já que tanto as regras quanto a inteligência artificial dependem desse volume de dados. Portanto, apesar de todo o processo ter sido feito com em média mil requisições, o que é considerado um volume pequeno, o desempenho mostrado e o resultado obtido foram muito satisfatórios.

## REFERÊNCIAS

ANDRADE, Marco Antônio Rocca de. **Fundamentos de Modelos de Markov Escondidos (HMM)**. Departamento de Engenharia Elétrica, Instituto Militar de Engenharia - IME, 2009. Disponível em <http://rmct.ime.eb.br/arquivos/RMCT_2_quad_2000/fund_modelos_Markov_escondidos.pdf> Acesso em: 20 de maio de 2022.

ASTHANA, Abhinav. **Postman**: Software desenvolvido 'em cima' do motor de javascript - NodeJS - que disponibiliza uma grande gama de ferramentas para testes em APIs. Versão V10, 11 agosto. 2022. Disponível em: https://www.postman.com/. Acesso em: 29 setembro. 2022.

COURNAPEAU, David. **Scikit-learn**: Framework de inteligência artificial desenvolvida em python. Versão v1.0.2, 28 abril. 2021. Disponível em: https://scikit-learn.org/stable/. Acesso em: 06 outubro. 2022.

DOMINGOS, Pedro. **O Algoritmo Mestre: Como a busca pelo algoritmo de machine learning**. São Paulo. Novatec Editora 1ª edição. 24, janeiro de 2017.

HYKES, Solomon. **Docker**: Software de gerenciamento de containers. Versão 20.10, 20 maio. 2022. Disponível em: https://www.docker.com//. Acesso em: 20 maio. 2022.

INC, Mongodb. **MongoDB**: Banco de dados não relacional orientado à documentos. Versão V10, 19 agosto. 2022. Disponível em: https://www.mongodb.com/. Acesso em: 03 outubro. 2022.

LIGEZA, Antoni. **Logical Foundations for Rule-Based Systems: Studies in computational intelligence**. Spinger Editora 2ª edição. 1, março de 2006.

LUZ, Felipe. **Algoritmo KNN para classificação**. inferir.com.br, 2019. Disponível em: <https://inferir.com.br/artigos/algoritimo-knn-para-classificacao/> Acesso em: 18 de maio de 2022.

MALINI N., Dr.M, et al. **Analysis on Credit Card Fraud Identification Techniques based on KNN and Outlier Detection**. Chennai, Índia: IEEE, 11, julho, 2017.

**PAYMENT FRAUD ATTACK RATE ACROSS FINTECH BALLOONED 70% IN 2021**. helpnetsecurity. Disponível em: <https://www.helpnetsecurity.com/2022/03/21/payment-fraud-increase/> Acesso em: 12 maio de 2022.

RUSSON, Guido Van. **Python**: Software linguagem de programação. Versão 3.10, 4 setembro. 2021. Disponível em: https://www.python.org/. Acesso em: 20 maio. 2022.

SHAILESH, S. Dhok. Credit Card Fraud Detection Using Hidden Markov Model. International **Journal of Soft Computing and Engineering**. Março, 2012. Disponível em

<https://mdatechsys.com/project/Mobile%20Computing/Credit%20Card%20Fraud%20Detection%20Using.pdf> Acesso em: 19 de maio de 2022.

STENBERG, Daniel. **cURL**: Software de linha de comando para transferir dados. Versão 7.83.1, 11 maio. 2022. Disponível em: https://curl.se/. Acesso em: 20 maio. 2022.

STONEBRAKER, Michael. **PostgreSQL**: Software de banco de dados procedural. Versão 13.4, 9 set. 2021. Disponível em: https://www.postgresql.org/. Acesso em: 20 maio. 2022.

WANG, Xiaoguo, et al. **Research and Design of a Rules Engine for Bank Anti-fraud Platform**. Tongji University, Shanghai, China. International Conference on Engineering Management Iconf-EM. Janeiro, 2017. Disponível em <https://www.atlantis-press.com/proceedings/iconfem-16/25868905> Acesso em: 17 de maio de 2022.

YONG, Zhou. **An Improved KNN Text Classification Algorithm Based on Clustering**. School of Computer Science & Technology, China University of Mining & Technology, Xuzhou, Jiangsu 221116, China Março, 2009. p 230-237. Disponível em <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.415.9683&rep=rep1&type=pdf> Acesso em: 18 de maio de 2022.

Publicado em 29/05/23