DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Aprendizado de máquina para otimização das retentativas de pagamentos recorrentes com cartão de crédito de uma empresa SaaS

Relatório submetido à Universidade Federal de Santa Catarina
como requisito para a aprovação da disciplina:
DAS 5511: Projeto de Fim de Curso

Mateus Gabriel Bosa

Florianópolis, Maio de 2021

Aprendizado de máquina para otimização das retentativas de pagamentos recorrentes com cartão de crédito de uma empresa SaaS

Mateus Gabriel Bosa

Esta monografia foi julgada no contexto da disciplina

DAS 5511: Projeto de Fim de Curso

e aprovada na sua forma final pelo

Curso de Engenharia de Controle e Automação

Prof. Eric Aislan Antonelo

Banca examinadora:

Pedro Covolan Bachiega
Orientador na empresa

Prof. Eric Aislan Antonelo
Orientador no curso

Prof. Marcelo De Lellis Costa de Oliveira
Responsável pela disciplina

Prof. Jomi Fred Hübner
Avaliador

Agradecimentos

À minha família agradeço por todo o apoio, por me incentivarem nos momentos difíceis e compreenderem a minha ausência enquanto eu me dedicava aos estudos.

À minha mãe, Nadir, minha base nesses anos de estudo, agradeço por me dar todo o suporte necessário para que eu pudesse alcançar os meus sonhos. Obrigado também pela educação e valores que fui ensinado desde criança. Também agradeço por todas as nossas aventuras em Floripa que deixaram lembranças muito alegres nesses anos.

À minha irmã, Alini, e meu cunhado, Leandro, agradeço por todas as conversas e conselhos nas inúmeras vezes que me buscaram em Araranguá durante esses anos de graduação.

Aos meus tios, Pedro e Nilva, e meus primos, Cristine e Marcel, por acreditarem no meu potencial, terem me dado tanto suporte quando me mudei para Floripa e todos os conselhos para que eu pudesse ingressar num curso de Engenharia.

Aos meus colegas de curso, pela parceria e por todas as horas de estudo e compartilhamento de conhecimento. Obrigado pelas tantas memórias, com certeza a jornada foi mais leve graças a vocês.

Aos professores que participaram da minha formação, pois com certeza cada um contribuiu para que eu tivesse tantos aprendizados nesses últimos anos.

Ao meu orientador, Eric, por todo o auxílio na execução deste trabalho e por ter aceitado esse desafio em tempos tão difíceis.

Ao meu mentor, Ramon, por ter me ajudado a desenvolver as habilidades necessárias e por todas as discussões que resultaram no desenvolvimento deste projeto.

Ao meu time Commandos, que me acolheu nesse último ano e me ajudou na transição para um novo desafio profissional como desenvolvedor. Obrigado em especial ao Pedrinho, meu gestor, pelo ambiente que me proporcionou tantos aprendizados e desenvolvimento pessoal.

À RD Station, por ter me dado a oportunidade de subir nesse foguete em 2015 e ter acelerado meu desenvolvimento profissional ao trabalhar com um propósito que eu acredito tanto.

À minha esposa e eterna companheira, Rebeca, por ter me dado tanto apoio e carinho nesses últimos meses. Obrigado por ter me ajudado a revisar todo este trabalho e me incentivado na conclusão deste projeto. Obrigado por participar da conclusão desse momento tão importante na minha vida.

Resumo

Num modelo de receita recorrente, a inadimplência de clientes gera perda financeira para a empresa, então recusas de pagamento recorrentes no cartão de crédito são um problema sério para esses negócios. Desta forma, ao tentarem fazer uma cobrança para um cartão de um cliente e receberem uma recusa do banco emissor, essas empresas podem fazer retentativas até conseguirem efetivar o pagamento. No entanto, existe um custo para cada tentativa de cobrança feita pelas empresas. Além disso, um alto número de tentativas negadas pode fazer com que a empresa seja penalizada com multas pelas bandeiras de cartões. Alguns tipos de recusas são sensíveis e não devem receber retentativas pelas processadoras de pagamentos. Em casos graves as bandeiras podem até mesmo deixar de aceitar transações de uma determinada instituição. Dado o contexto, o presente projeto teve por objetivo o emprego e a avaliação de técnicas de classificação capazes de modelar a probabilidade de uma tentativa de pagamento ser aprovada. Após uma análise exploratória dos dados disponíveis e um maior entendimento do problema, diversos tipos de classificadores foram treinados, seus hiperparâmetros foram otimizados e analisados. Por fim uma nova estratégia de retentativas de pagamentos foi criada fazendo uso do modelo treinado com o método floresta aleatória. Experimentos com grupos controlados de clientes em produção obtiveram um resultado de 70% de acurácia e uma redução de 60% da quantidade de retentativas, resultando no aumento da taxa de aprovação de pagamentos com cartão de crédito. Com a implementação dessa solução, a empresa do caso de estudo tem uma expectativa de economizar até R$65.000,00 em custos com transações e aumentar em quase 20 pontos percentuais a taxa de aprovação de cartões de créditos dos seus clientes.

Palavras-chave: Aprendizado de máquina. Métodos de classificação. Floresta aleatória. Retentativas de pagamentos. Cartão de crédito.

Abstract

For companies with recurring revenue models, recurring credit card payment declines are a serious problem for these businesses. When trying to charge a customer and receive a refusal from the issuing bank, these companies can make retries until they can make the payment. However, there is a cost for each attempted collection made by them. Also, a high number of denied attempts may result in the company being penalized with fines for credit card brands. Some types of declines are sensitive and should not be retried by payment processors. But in severe cases, credit card brands may even refuse transactions from some companies. Given the context, the present project at employing and evaluating classification techniques capable of modeling the probability that an attempted payment will be approved. After an exploratory analysis of the available data and a greater understanding of the problem, several types of classifiers were trained, their hyperparameters were optimized and analyzed. Finally, a new payment retry strategy was created using the model trained with the random forest method. Experiments with controlled groups of customers in production achieved a result of 70% accuracy and a reduction of 60% in the number of retries, increasing the credit card payment approval rate. This solution made the company in this case study expect to save up to R$65,000.00 in transaction costs and increase its customers' credit card approval rate by almost 20 percentage points.

Keywords: Machine learning. Classification methods. Random forest. Payments retries. Credit card.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

Abecs	Associação Brasileira de Empresas de Cartões de Crédito e Serviços
API	Application Programming Interface
FN	False Negative
FP	False Positive
kNN	k-Nearest Neighbours
$Q2C$	Quote-to-Cash
RD	RD Station
$SaaS$	Software as a Service
SVM	Support Vector Machine
TN	True Negative
TP	True Positive

Sumário

1 Introdução

Os pagamentos eletrônicos no Brasil já representam 44% do consumo total, de acordo com a Associação Brasileira de Empresas de Cartões de Crédito e Serviços (Abecs) [1]. Projeções feitas por especialistas e pesquisas feitas pela Bain Company e pela Mastercard, apontam que não deve demorar para que os meios de pagamento digital liderem essa relação [2].

Em 2020, o isolamento social acelerou ainda mais a tendência dos pagamentos sem contato: não apenas a transação com cartão por aproximação, mas especialmente o pagamento online. De acordo com a pesquisa da Mastercard, 56% dos brasileiros mudaram o comportamento em relação à forma de pagamento durante o isolamento. E 75% dessa parcela estão usando meios de pagamento digitais.

O pagamento por meio do número de cartão de crédito foi o pioneiro no mundo das vendas online. Por fazer uso de uma forma de pagamento convencional, ficou mais simples implementar o pagamento online pelo cartão de crédito. Com isso, a aceitação do público para fazer pagamentos com cartão foi instantânea, principalmente pela sua facilidade. Bastando apenas colocar o número de cartão, o nome do portador e o código de segurança, e pronto: está feito o pagamento. Mas nem tudo é tão simples assim, já que o processamento de cartões de crédito é um serviço complexo que envolve vários componentes instáveis, tecnologias emergentes, redes de pagamento, órgãos reguladores e instituições financeiras.

As empresas de software foram altamente impactadas pelo novo comportamento de pagamentos do consumidor. Muitas dessas instituições têm usado um modelo de recorrência como forma de recebimento do seus clientes. Recorrência é um modelo de negócios onde o cliente realiza o pagamento contínuo para ter acesso a um produto ou serviço por um determinado período de tempo [3]. Um dos maiores benefícios deste modelo é permitir que a empresa tenha uma melhor previsão do quanto ela receberá nos próximos meses. Conforme a quantidade de planos e serviços oferecidos, é possível que ela saiba até quanto será o seu faturamento anual.

Dado esse cenário, a ocorrência de recusas e outros erros durante o pagamento recorrente prejudica as empresas que usam esse modelo de negócio. Em especial, pagamentos com cartão de crédito podem sofrer com recusas de pagamentos por diversos motivos, podendo citar expiração da data de validade do cartão, falta de limite, saldo insuficiente, cartão bloqueado ou desabilitado para compras on-line, entre outros. Grande parte das recusas de pagamentos em vendas online são feitas pelo banco ou pelo emissor do cartão de crédito e isso foge do controle das empresas processadoras de pagamento.

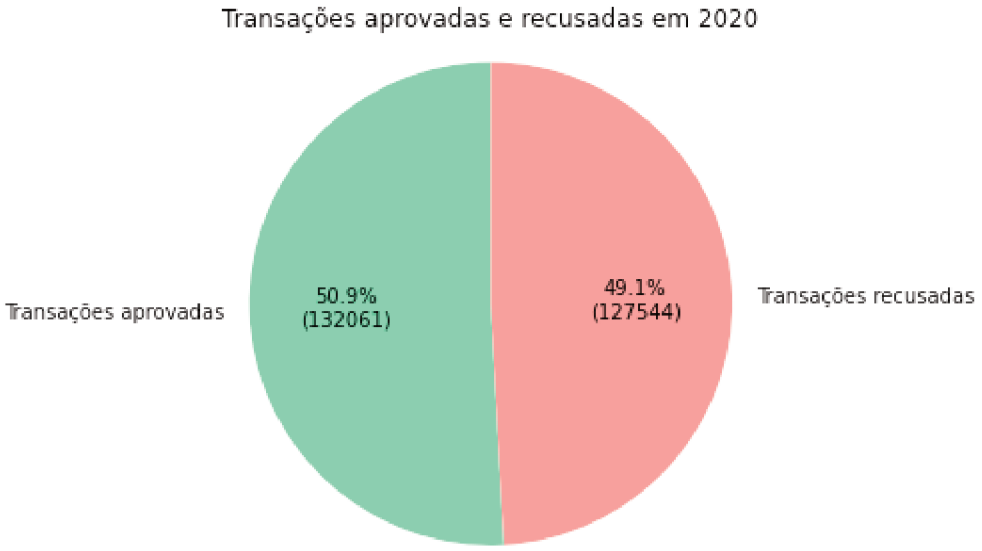

Figura 1 – Transações aprovadas e recusadas pela RD Station em 2020

1.1 Justificativa

Num modelo de receita recorrente, a inadimplência de clientes gera perda financeira para a empresa, então recusas de pagamentos recorrentes no cartão de crédito são um problema sério para esses negócios. Desta forma, ao tentarem fazer uma cobrança para um cartão de um cliente e receberem uma recusa do banco emissor, essas empresas podem fazer retentativas até conseguirem efetivar o pagamento.

No entanto, existe um custo para cada tentativa de cobrança feita pelas empresas. Além disso, um alto número de tentativas negadas pode fazer com que a empresa seja penalizada com multas pelas bandeiras de cartões. Alguns tipos de recusas são sensíveis e não devem receber retentativas pelas processadoras de pagamentos. Em casos graves as bandeiras podem até mesmo deixar de aceitar transações de uma determinada instituição.

A RD Station é a empresa de estudo do presente trabalho e a figura 1 representa a proporção entre transações aprovadas e negadas por cartão de crédito pelos seus clientes em 2020. No último ano sua taxa de aprovação ficou abaixo dos 51%, ou seja, praticamente metade das transações dos seus clientes foram recusadas.

Com base em pesquisas de mercado, a média da taxa de aprovação de cartão de crédito dos principais processadores de pagamento gira em torno de 75% [4], conforme tabela 1. Isso dá para o sistema uma grande margem de melhoria para estar compatível com seus competidores.

Tabela 1 – Taxas de aprovação por processadoras de pagamentos em 2019

Taxa de aprovação	Fevereiro	Março	Abril
Appmax	80,76%	**79,43%**	75,51%
hotmart	71,66%	70,26%	72,84%
mercado pago	71,77%	74,67%	73,42%
Monetizze	58,47%	56,84%	57,39%
pagar.me	60,38%	71,36%	**82,54%**
wirecard	**85,27%**	72,50%	73,97%

1.2 Objetivos

O objetivo geral deste projeto é empregar e avaliar técnicas de classificação que sejam capazes de modelar a probabilidade de uma tentativa de pagamento ser aprovada com base nos dados da transação. Dessa forma, busca-se otimizar o fluxo de retentativas de pagamentos da empresa RD Station. Também tem-se como objetivos específicos:

- Acabar com as retentativas indevidas de cobranças em cartões de crédito: identificar falhas de tentativas de transações que não devem ser retentadas e impedir novas tentativas para elas;
- Aplicar técnicas de aprendizado de máquina nas retentativas de cobranças em cartões de crédito: para as falhas de transações que podem receber novas tentativas, criar um modelo preditivo a partir dos dados disponíveis dos clientes e das suas transações sugerindo os melhores dias para serem feitas novas tentativas de transações com maiores chances de efetivação dos pagamentos.

Como principal resultado, espera-se um aumento da taxa de aprovação das transações de cartão de crédito para taxas próximas às aceitas no mercado.

1.3 Estrutura do documento

O trabalho é organizado em diferentes capítulos da seguinte maneira:

- O capítulo 2 aborda a fundamentação teórica dos conceitos de pagamentos e das ferramentas utilizadas neste projeto;
- O capítulo 3 faz uma introdução da empresa e formaliza o problema que o presente projeto se propõe a solucionar;
- O capítulo 4 apresenta o desenvolvimento da solução para recusas que não devem ser retentadas, impedindo que elas recebam novas tentativas;

- O capítulo 5 apresenta o desenvolvimento da solução para recusas que podem ser retentadas, gerando um modelo preditivo que otimiza o fluxo de pagamentos da empresa;
- O capítulo 6 apresenta os resultados obtidos com a implementação do projeto;
- Por fim, o capítulo 7 apresenta a conclusão do trabalho feito ao longo do programa do Projeto de Fim de Curso e perspectivas a respeito de trabalhos futuros.

2 Fundamentação teórica

2.1 Pagamentos com cartão de crédito

De acordo com a Abecs, o volume de compras pagas com cartões de crédito, débito e pré-pagos no Brasil deve superar 2,3 trilhões de reais em 2021. Isto representa uma alta de 18% a 20% sobre o volume do ano passado [1].

Apenas para o movimento com cartões de crédito a expectativa da Abecs é de que haja uma alta de 19% a 21%. Isso demonstra como o mercado brasileiro tem sido aquecido com o pagamento de cartões de crédito e por isso esse é um dos meios de pagamento mais populares, principalmente no que se refere a pagamentos recorrentes.

2.1.1 Recorrência de pagamentos

O pagamento recorrente é aquele feito no caso de um serviço prestado de forma contínua, em que existe uma assinatura. Sendo assim, enquanto o serviço continuar a ser prestado, o cliente deverá continuar fazendo os pagamentos, por um tempo determinado em contrato [3].

Em geral, existe um período mínimo para que o serviço seja prestado para que o cliente possa fazer um cancelamento, se desejar. Caso cancelado antes do final deste período, o cliente deverá pagar uma multa por rescisão. A recorrência é definida por períodos, na maioria das vezes mensais, trimestrais ou anuais.

Como num modelo de pagamentos recorrentes as cobranças são feitas automaticamente e renovadas periodicamente, até que o cliente decida cancelar sua assinatura, o número de clientes inadimplentes é muito menor que em modelos por pagamentos por boleto bancário, por exemplo.

Os pagamentos recorrentes automáticos são feitos por meio da tokenização dos cartões de crédito. Quando um cliente compra um produto ou um serviço online, ao colocar os dados do seu cartão numa página de pagamentos (conhecida como *checkout*) eles são enviados para um serviço que valida essas informações e gera um *token*, uma chave codificada. Então essa chave é salva de forma segura no site em que ele está fazendo a compra. Com essa chave, novas transações podem ser feitas regularmente debitando do cartão de crédito do cliente.

Cobrando de forma recorrente pelo cartão de crédito, as empresas não utilizam o limite total do cartão dos seus clientes, já que a recorrência é diferente do parcelamento. Na recorrência, é necessário que o cliente apenas tenha o valor da mensalidade no limite.

Enquanto isso, no parcelamento, o limite total da venda é retirado do cartão do cliente e volta aos poucos, de acordo com o pagamento das faturas. Desta forma a recorrência é uma excelente relação custo-benefício para o cliente, além de garantir faturamento mensal para empresa.

Além disso, um grande benefício da recorrência para as empresas é conseguir ter uma melhor previsão do quanto elas receberão nos próximos meses. De acordo com a quantidade de planos e serviços oferecidos, é possível que a empresa saiba até quanto será o seu faturamento anual, com dados muito mais reais do que em empresas que utilizam outras formas tradicionais de cobrança.

2.1.2 Adquirente

Empresas adquirentes, também conhecidas como credenciadores, são as responsáveis pelo processamento das operações de cartão de crédito e débito. A principal função delas é criar um canal de comunicação rápido e seguro entre lojas e o banco para checar os dados do consumidor e validar a compra [5].

2.1.3 Bandeiras de cartão de crédito

As bandeiras, como Mastercard, Visa e American Express, são órgãos reguladores que determinam regras do mercado de cartão de crédito. Alguns exemplos dessas regras são a quantidade de parcelas em que se pode dividir um pagamento e os estabelecimentos em que cada bandeira é aceita.

Em uma compra, a adquirente usada pelo vendedor se conecta com a bandeira de cartão de crédito. Então a bandeira aciona o emissor que responde autorizando ou recusando a transação. Desse modo, a bandeira regulariza o uso do cartão e faz a ponte entre a adquirente e o banco do consumidor.

2.1.4 Recusas de pagamentos em cartões de crédito

Ao tentar autorizar um pagamento, o emissor do cartão de crédito pode retornar a aprovação ou negação dessa tentativa. As falhas de tentativas de transações são classificadas em *hard declines* e *soft declines*.

Hard declines são recusas de pagamentos por problemas que exigem uma correção com o emissor do cartão, como quando um cartão está bloqueado ou vencido. Já *soft declines* são recusas de transações por motivo temporário, como por falta de limite de crédito.

Com base nisso, as adquirentes sugerem para as empresas que usam o modelo de pagamentos recorrentes que as tentativas que sofrem *hard decline* não sejam retentadas,

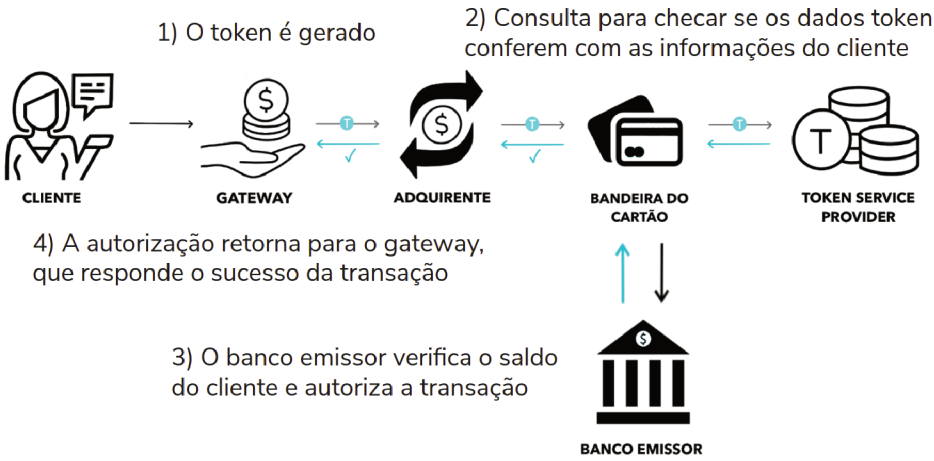

Figura 2 – Fluxo de uma transação via *gateway*

já que se faz necessária uma ação do dono do cartão. Já para as tentativas que sofrem *soft decline* é permitida retentativa. No exemplo de um cartão com limite estourado, após o pagamento de uma fatura ou com um pagamento antecipado o limite é reajustado, podendo assim ser feita uma nova cobrança.

2.1.5 *Gateway*

Assim como as maquininhas de cartão são para as lojas físicas, o *gateway* é para lojas virtuais, uma ferramenta que pode ser considerada como um terminal de cartão de crédito. O cliente fornece os dados da compra na página de *checkout* do vendedor, o *gateway* coleta essas informações e repassa aos bancos ou adquirentes. A figura 2 ilustra o fluxo de uma transação considerando todos os componentes apresentados nessa seção.

Além disso, o *gateway* possui diversas funcionalidades, como por exemplo, serviços antifraude, opções de crédito e débito, empresas conciliadoras, entre outros. Isso possibilita para as empresas terem acesso a todas as transações financeiras em um único lugar.

2.2 Aprendizado de máquina

O tema aprendizado de máquina se refere ao processo pelo qual sistemas computacionais geram aprendizados por meio de dados com instruções que são limitadas. Esses aprendizados se baseiam apenas em reconhecimento de padrões e inferências dos dados disponíveis [6]. Desta forma, o objetivo principal do aprendizado de máquina é fazer uma generalização além das observações contidas nos dados de treinamento. Além disso, também faz uma otimização contínua dessa generalização a partir da entrada de novos dados.

O objetivo do aprendizado de máquina é praticamente o mesmo para todas as suas aplicações, mas a maneira pela qual ele será implementado varia de acordo com a situação. Existe uma grande variedade de técnicas diferentes, desenvolvidas para solucionar tipos de problemas específicos. Em geral, os métodos de aprendizado de máquina são divididos em dois grupos distintos, dependendo do tipo de aprendizagem que espera ser realizada: aprendizagem não-supervisionada e aprendizagem supervisionada.

2.2.1 Aprendizado não-supervisionado

O aprendizado não-supervisionado é uma abordagem utilizada para descobrir de forma heurística e descrever padrões até então desconhecidos em um grupo de dados que não é categorizado [7]. Sendo assim, essa abordagem é classificada como não-supervisionada já que não existe um conhecimento sobre padrões existentes para o conjunto de dados analisados.

2.2.2 Aprendizado supervisionado

O aprendizado supervisionado é aquele que possui um supervisor, ou seja, um conhecimento prévio sobre os dados analisados. Com isso, a máquina pode avaliar a eficácia do seu aprendizado [7]. De forma geral, os principais métodos de aprendizagem supervisionada se dividem em duas categorias: regressão ou classificação.

2.2.2.1 Regressão

Problemas de regressão são aqueles que, para cada valor de entrada, existe um valor numérico de saída. Desta forma, o objetivo da máquina é de aprender a estimar um valor de saída para cada conjunto de valores de entrada.

2.2.2.2 Classificação

Problemas de classificação são aqueles que, para cada valor de entrada do conjunto de dados é atribuída uma categoria distinta. Assim, o objetivo da máquina é de aprender a classificar uma nova entrada, de acordo com as variáveis disponíveis, em uma destas classes.

2.2.2.2.1 Regressão Logística

Logistic Regression, em inglês, é uma técnica muito utilizada em problemas de classificação com o objetivo de prever a probabilidade de uma amostra pertencer a uma determinada classe.

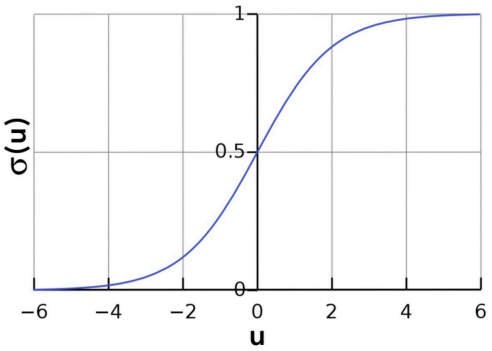

Figura 3 – Representação gráfica da função logística

O modelo pode ser representado pela equação 2.1, na qual $x = (x_0, x_1, ..., x_n)$ representa o valor de variáveis, $\theta = (\theta_0, \theta_1, ..., \theta_n)$ o vetor de parâmetros e σ a função logística.

$$h_\theta(x) = \sigma(x^T\theta) \tag{2.1}$$

A função logística σ é apresentada na equação 2.2 e demonstrada visualmente na figura 3. Essa função converte o valor que é produzido por $x^T\theta$ em uma probabilidade.

$$\sigma(u) = \frac{1}{1 + e^{-u}} \tag{2.2}$$

Desta forma, $h_\theta(x)$ representa que, dado o vetor de variáveis x, parametrizado por θ, o resultado é a probabilidade de a saída do modelo ser igual a 1.

$$h_\theta(x) = P(y = 1|x, \theta) \tag{2.3}$$

De acordo com a figura 3, a função logística pode variar entre 0 e 1. Assim, a predição usando a regressão logística pode ser feita de acordo com a equação 2.4, que define um limiar de 0,5.

$$\hat{y} = \begin{cases} 0, \text{ se } h_\theta(x) < 0,5 \\ 1, \text{ se } h_\theta(x) \geq 0,5 \end{cases} \tag{2.4}$$

Para a otimização do modelo é utilizada a função custo entropia cruzada binária (*binary cross-entropy*), com o objetivo de medir como o modelo está performando durante

o processo de treinamento. Essa função é apresentada na equação 2.5 em que $h_\theta(x)$ é a predição do modelo, y é a resposta desejada e m o número de amostras [8].

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))] \tag{2.5}$$

2.2.2.2.2 Árvores de Decisão

Decision Tree, em inglês, é um método que faz uso da modelagem de sistemas discretos na qual o modelo é obtido em forma de árvore. Esse é um dos métodos mais utilizados para a inferência indutiva por se tratar de um método de fácil entendimento e, se necessário, permite também a representação gráfica do modelo gerado [9]. Além da simplicidade em se montar as árvores, a maioria das técnicas podem trabalhar tanto com variáveis de entrada contínuas quanto categóricas.

A representação desse modelo é feita numa estrutura em que cada nodo interno corresponde a um teste sobre um determinado atributo. Um nodo possui ramos descendentes, e cada um deles representa uma possibilidade para o teste do atributo. Por fim, cada folha, que contém a classe respectiva às instâncias por ela classificada, representa a decisão obtida após testar os atributos. Dessa forma, o caminho que é percorrido para chegar até a uma determinada classe corresponde a uma regra de classificação [10].

A entropia é caracterizada pela organização de uma coleção de exemplos [11]. Para ilustrar, dado um conjunto S que contenha valores positivos e negativos para um determinado atributo, então a entropia de S é dada por:

$$Entropy(S) = -p_+\log_2 p_+ - p_-\log_2 p_- \tag{2.6}$$

em que p_+ é a proporção de exemplos na qual a saída é positiva e p_- na qual a saída é negativa. Assumindo que $0log_20$ como sendo 0, pode-se notar que a entropia é 1 quando a coleção contém um mesmo número de elementos cuja saída é positiva e negativa, e 0 quando todos os elementos de S pertencem à mesma classe.

Dado que o resultado da soma entre p_+ e p_- sempre será 1, pode-se escrever $p_- = 1 - p_+$. Então ao se generalizar o cálculo da entropia para casos em que o atributo analisado possui mais que dois valores possíveis, pode-se definir a entropia como:

$$Entropy(S) = \sum_{i=1}^{c} -p_i\log_2 p_i \tag{2.7}$$

em que p_i é a proporção de S que pertence à classe i e c é o número de classes do atributo analisado.

2.2.2.2.3 Florestas Aleatórias

Random Forest, em inglês, é a técnica que faz uma combinação de múltiplas árvores de decisões. Dado um conjunto de dados, são amostradas de forma aleatória o número de instâncias, podendo se repetir, em que n representa o número de instâncias do conjunto de dados original. Desta forma, cada árvore é induzida a partir de um desses subconjuntos, mas cada nó da árvore utiliza m atributos da quantidade total f de atributos. Os m atributos são escolhidos aleatoriamente e com repetição. Assim, $m = 1$ quando $f = 1$, e $m < f$ quando $f > 2$ [12].

Um dos parâmetros necessários para se definir nesse modelo é a quantidade desejada de árvores de decisão. Fazendo a combinação de uma variedade de árvores de decisão é possível convergir o valor de erro para um valor sem sobre-ajuste, que não sofreu *overfit*, em relação ao conjunto de dados fornecidos.

De forma prática, o conjunto de dados inicial é dividido em vários subconjuntos aleatórios e com repetição. Estes serão usados para realizar a indução de cada árvore de decisão correspondente. No final, a classificação da floresta aleatória é feita por meio de votação, em que cada árvore de decisão providencia uma classificação, e a classe que for maioria entre os votos é escolhida como a classificação final do modelo.

2.2.2.2.4 *Naive Bayes*

O classificador *Naive Bayes* é um método que considera as variáveis de entrada como independentes dada uma determinada classe. Por causa disso ele é considerado como "ingênuo", já que na maioria dos casos a suposição de independência das características de uma instância é falsa [13].

Esta técnica se baseia no teorema de *Bayes*, que trata sobre probabilidade condicional. Em suma este teorema representa a probabilidade de um evento A ocorrer dado um evento B, de acordo com a equação:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.8}$$

Esse teorema também pode ser utilizado para o cálculo das probabilidades necessárias para problemas de classificação. Para isso é necessário substituir um dos argumentos da equação pela classe a ser calculada. Assim é obtida a equação:

$$P(classe|B) = \frac{P(B|classe)P(classe)}{P(B)} \tag{2.9}$$

Para se determinar a classe mais provável da nova instância, ou seja, fazer uma predição, é feito o cálculo da probabilidade de todas as possíveis classes. Por fim é escolhida a classe com a maior probabilidade como rótulo da nova instância.

Sendo x_i um elemento pertencente ao vetor de N características de uma instância, para escolher a classe com maior probabilidade pode-se maximizar a $P(classe|x_1, ..., x_N)$, maximizando o valor do numerador $P(x_1, ..., x_N|classe) \times P(classe)$ e minimizando o valor do denominador $P(x_1, ..., x_N)$.

Como o denominador $P(x_1, ..., x_N)$ é uma constante, já que não depende da variável classe que se está procurando, pode-se considerá-lo nulo no teorema de Bayes. Como resultado se tem a seguinte equação:

$$argmaxP(classe|x_1, ..., x_N) = argmaxP(x_1, ..., x_N|classe) \times P(classe) \tag{2.10}$$

Como o classificador *Naive Bayes* faz uma suposição "ingênua" de que todos os atributos $x_1, ..., x_N$ da instância que se quer classificar são independentes, então o cálculo do valor do termo $P(x_1, ..., x_N|classe)$ é reduzido para o cálculo de $P(x_1|classe) \times ... \times P(x_N|classe)$. Assim, a equação utilizada por esse classificador é:

$$argmaxP(classe|x_1, ..., x_N) = argmax \prod_{i=1}^{N} P(x_i|classe) \times P(classe) \tag{2.11}$$

2.2.2.2.5 K-ésimo Vizinho mais Próximo

k-Nearest Neighbours, em inglês, é um dos mais populares para reconhecimento de padrões [14]. O método tem seu aprendizado baseado em instâncias e, diferente dos demais, ele não constrói modelos que representam o conjunto de treinamento durante o aprendizado, mas apenas armazena os dados para serem consultados na etapa de classificação. Por causa disso, ele também é chamado de algoritmo de aprendizado preguiçoso.

Para esse método todas as instâncias correspondem a pontos em um espaço n-dimensional. Desta forma, uma instância x é definida como sendo um vetor no espaço:

$$\langle a_1(x), a_2(x), ..., a_n(x) \rangle \tag{2.12}$$

em que $a_r(x)$ representa o valor do r-ésimo atributo de x.

Para fazer a classificação dos vizinhos mais próximos a certo ponto no espaço, é necessário fazer a definição de uma métrica que represente essa distância. Comumente se

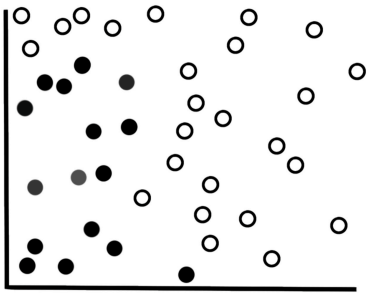

Figura 4 – Conjunto de dados inicial

utiliza a distância Euclidiana. Dadas duas instâncias x_i e x_j, a distância entre elas como:

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n}(a_r(x_i) - a_r(x_j))^2} \tag{2.13}$$

Com essa equação é possível obter as k instâncias mais próximas de um ponto desejado no espaço. É muito importante a escolha do parâmetro k, já que o seu valor afeta diretamente os resultados obtidos. No caso de valores grandes de k se identifica maior atenuação dos ruídos presentes nos dados, e características e tendências podem ser suprimidas quando presentes em pequenos grupos de dados. Geralmente é avaliada a qualidade do algoritmo para diferentes valores de k e escolhido o que apresentar melhor desempenho.

2.2.2.2.6 Máquina de Vetores de Suporte

Support Vector Machine, em inglês, é um método de aprendizagem computacional que funciona por meio do mapeamento dos dados para um espaço de parâmetros de maior dimensão, sendo possível assim fazer uma categorização dos pontos. Neste momento é feita uma separação entre as categorias definidas e os dados são transformados possibilitando o uso de um hiperplano como separador. Desse modo, futuros pontos amostrados podem ser classificados utilizando o modelo já treinado.

Para ilustrar visualmente, um exemplo é demonstrado na figura 4 em que pontos espalhados pertencem a uma de duas possíveis categorias. Tendo as categorias definidas, elas podem ser separadas por meio de uma curva, como demonstrado na figura 5. Finalmente, o limite entre as duas categorias estabelecidas pode ser definido de forma linear após a transformação por meio de um hiperplano, conforme figura 6.

Para fazer essa transformação, a função matemática utilizada é conhecida como função *kernel* [15]. Essa função pode ser de caráter linear, polinomial, entre outras.

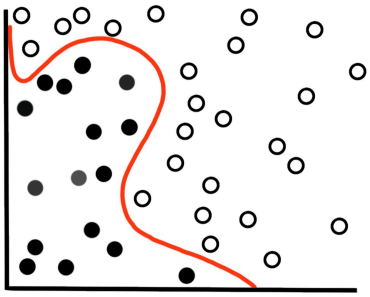

Figura 5 – Separador de dados adicionado

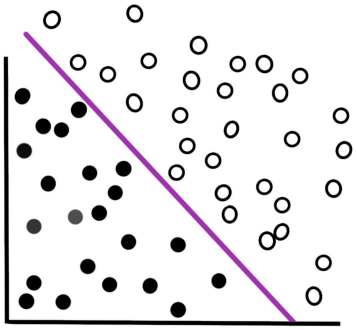

Figura 6 – Transformação por meio de um hiperplano

2.2.3 Metodologia para treinamento e avaliação dos modelos

Em geral, projetos de engenharia que fazem uso dos métodos de aprendizado de máquina são constituídos pelos seguintes componentes [6]:

- **Modelo**: de acordo com o método utilizado, existe uma forma específica de o conjunto de dados de treinamento serem interpretados e utilizados pelo método. Este modelo costuma ser atualizado com a entrada de novos dados e o resultado dos outros dois componentes com o objetivo de melhorar a eficácia do aprendizado.

- **Avaliação**: conforme a estrutura de dados e do problema estudado, existe uma função de avaliação da eficácia da técnica utilizada. O objetivo dessa função é mensurar a qualidade de aprendizado do método, fazendo uma comparação das saídas estimadas com as saídas reais do conjunto de dados de treinamento. Para cada técnica utilizado, essa função deve ser minimizada ou maximizada, obtendo um melhor resultado do modelo.

- **Otimização**: a função de avaliação pode ser otimizada de diferentes formas, dependendo da técnica utilizada. A eficácia do método pode ser melhorada com um bom processo de otimização, que continuamente atualiza os parâmetros do modelo até obter um resultado satisfatório.

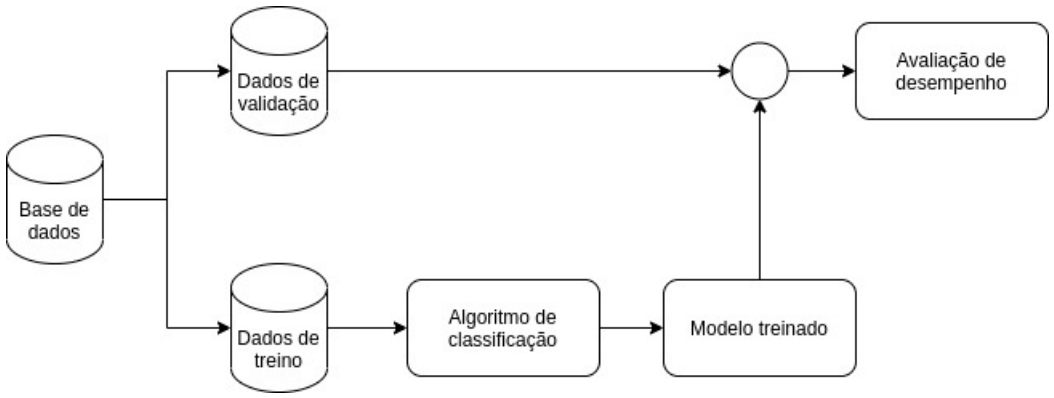

Figura 7 – Fluxo de separação dos dados em grupos de treino e validação

Além disso, para a execução desses algoritmos se faz necessário um pré-processamento dos dados, fazendo uma limpeza das entradas que possam alterar ou impedir a execução do algoritmo, como por exemplo dados ausentes ou a presença de *outliers*.

Após a definição de uma versão do método a ser utilizado, é preciso realizar e validar o aprendizado obtido. Para isso, normalmente os dados são separados em dados de treinamento, aqueles que são utilizados para que o algoritmo aprenda, e dados de validação, um conjunto de dados que ajuda na avaliação da eficácia do modelo em generalizar o seu aprendizado para entradas de dados que não estavam presentes no treinamento do modelo. Esta separação dos dados é muito importante para evitar que o modelo seja incapaz de generalizar o aprendizado para além dos dados de treinamento.

O método de avaliação do modelo é definido de acordo com as especificidades do problema estudado. Para isso, é importante ter em mente os objetivos que se espera alcançar com o seu desenvolvimento, como um valor mínimo de erro considerado aceitável. Em resumo, a validação espera definir a eficácia do modelo em um indicador de desempenho.

Após a separação dos dados e a escolha de um método de validação, o processo de treinamento e validação pode ser feito de acordo com o fluxograma da figura 7. De forma iterativa, esse fluxo pode ser seguido variando a forma que a base de dados é dividida entre dados de treino e de validação. A técnica de dividir a base de dados em dados de treino e de validação diversas vezes, com o objetivo de avaliar a eficácia do modelo, é conhecida como validação cruzada.

Em geral, projetos de aprendizado de máquina são feitos de forma iterativa seguindo as seguintes etapas:

- Identificação dos dados necessários: escolha e coleta dos dados;
- Pré-processamento de dados: tratamento de entradas nulas e *outliers* e formatação dos dados a serem treinados de acordo com o algoritmo a ser utilizado;

- Divisão do conjunto de dados em treinamento e validação: podendo ser feito múltiplas vezes ao optar pela técnica da validação cruzada;
- Seleção do algoritmo: escolha do algoritmo e modelo de aprendizado a ser utilizado;
- Treinamento do algoritmo com base nos dados de treinamento;
- Avaliação do modelo com dados de validação: análise do aprendizado do modelo com dados que não foram utilizados para treinamento;
- Avaliação de performance do modelo e comparação com modelos de parâmetros e algoritmos diferentes para que se possa escolher o que melhor atende as demandas do projeto.

2.2.4 Métricas de avaliação

As métricas de avaliação são usadas para se fazer uma avaliação de desempenho dos modelos de aprendizado de máquina. A matriz de confusão é a ferramenta mais usada para avaliações de modelos de classificação [16]. Um exemplo de uma matriz de confusão de um modelo com uma saída booleana é demonstrada na figura 8.

Conforme ilustrado, uma matriz 2x2 possui os seguintes componentes:

- **Verdadeiro negativo (*true negative* — TN)**: a classe é prevista corretamente, quando o valor previsto é de uma classe negada e o conjunto real também é de uma classe negada.
- **Falso positivo (*false positive* — FP)**: a classe é prevista incorretamente, quando o valor previsto é de uma classe positiva mas o conjunto real é de uma classe negada.
- **Falso negativo (*false negative* — FN)**: a classe é prevista incorretamente, quando o valor previsto é de uma classe negada mas o conjunto real é de uma classe positiva.
- **Verdadeiro positivo (*true positive* — TP)**: a classe é prevista corretamente, quando o valor previsto é de uma classe positiva e o conjunto real também é de uma classe positiva.

Com base nos parâmetros apresentados na matriz de confusão, existem algumas métricas de avaliação de desempenho que são utilizadas para analisar a qualidade de um modelo. As principais métricas utilizadas em projetos de aprendizado de máquina são apresentadas abaixo.

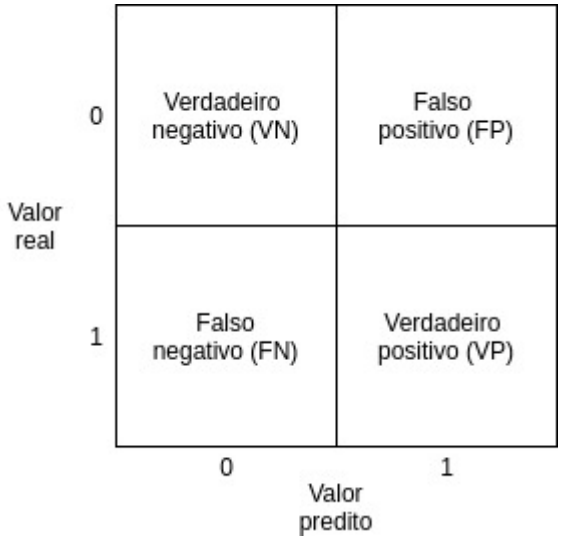

Figura 8 – Exemplo de uma matriz de confusão

Acurácia: métrica que indica, dentre todas as classificações, quantas o modelo classificou corretamente. É usada como um indicador da performance geral do modelo, já que corresponde a sua taxa de acertos.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.14}$$

Precisão: avalia a proporção de classificações de classe positiva classificadas corretamente, ou seja, dentre todas as classificações de classe positiva que o modelo fez, quantas estão corretas. Em geral, a precisão pode ser usada em uma situação em que os falsos positivos são considerados mais prejudiciais que os falsos negativos.

$$Precision = \frac{TP}{TP + FP} \tag{2.15}$$

Recall (sensibilidade): representa a proporção das predições positivas estarem corretas quando o valor real é positivo, ou seja, dentre todas as situações de classe positiva como valor esperado, quantas estão corretas. Em geral, o recall pode ser usado em uma situação em que os falsos negativos são considerados mais prejudiciais que os falsos positivos.

$$Recall = \frac{TP}{TP + FN} \tag{2.16}$$

F1-score: é uma métrica que combina a precisão e o recall, uma média harmônica entre elas.

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.17}$$

É importante ressaltar que cada projeto tem um objetivo próprio, por isso, com base no problema a ser estudado, o projetista deve ter um bom entendimento de qual dessas métricas representa a melhor solução que ele está buscando. É com base nisso que serão feitas melhorias e otimizações do modelo gerado.

3 RD Station

Nos últimos anos, o comportamento do consumidor passou por uma grande transformação. Ele não está mais preocupado se vai levar o produto "para casa", o que ele precisa é de agilidade. Um exemplo disso é que, se você está com fome, pode simplesmente pegar o seu celular, acessar um aplicativo de delivery e, em poucos minutos, fazer seu pedido.

Os modelos de negócios das empresas de software foram altamente impactados pelo novo comportamento do consumidor, e alinhado aos avanços tecnológicos, as empresas também tiveram que se adaptar. A maioria destas empresas deixou de oferecer produtos físicos e começou a oferecer eles de forma digital.

Estas empresas têm usado um modelo conhecido como SaaS (*Software as a Service*), e se diferenciam dos modelos tradicionais por distribuir e comercializar software por meio de serviços, licenças de uso e assinaturas. Desta forma seus clientes irão pagar somente pelos recursos utilizados do software, e não por toda a estrutura disponível que muitas vezes nem é de utilidade para o cliente [17].

Uma das principais vantagens desse modelo é a redução de custos, já que as empresas não precisam comprar diversas licenças ou possuir poderosos servidores para disponibilizar o seu produto. Com isso elas podem concentrar seus esforços no desenvolvimento do seu produto e talvez até terceirizar algumas responsabilidades com serviços de outras empresas. Assim a evolução do produto é acelerada e os custos são reduzidos gerando por fim uma maior vantagem competitiva em relação aos seus competidores de mercado.

Outra grande vantagem desse modelo é a facilidade na integração de dados. Para isso, são desenvolvidas APIs (*Application Programming Interface*), conjuntos de rotinas e padrões de programação para acesso a aplicativos de software ou plataformas baseadas na Web. Elas abstraem a implementação e fornecem especificações para rotinas, estruturas de dados, classes de objeto, variáveis e chamadas remotas que o desenvolvedor necessita. Dessa forma elas facilitam a conexão e comunicação entre os softwares permitindo a transferência de dados de forma confiável.

Nesse contexto surge a RD Station, antes conhecida como Resultados Digitais, uma empresa que atua no mercado de SaaS desde 2011 com o desenvolvimento de softwares que impulsionam os resultados de marketing e vendas de pequenas e médias empresas. Atualmente a empresa possui mais de 800 colaboradores que atendem a mais de 25.000 clientes, que além do Brasil estão espalhados em mais de 20 países, como Colômbia, México e Portugal [18].

3.1 RD Contas

Por se tratar de uma empresa que utiliza o modelo SaaS, a aquisição e utilização dos produtos pelos clientes se dá de forma digital por meio de uma assinatura de pagamento recorrente, podendo ser mensal, trimestral ou anual. Existe assim a demanda de uso de um sistema *Quote-to-Cash* (Q2C) [19] responsável por fazer a gestão das assinaturas dos seus clientes e disponibilizando as informações necessárias de planos e status dos serviços para os produtos desenvolvidos pela RD.

Desenvolvido internamente, o RD Contas surge da necessidade de gestão dos clientes e controle dos seus pagamentos. Primeiro emitindo boletos, painel de clientes, aceitando cartões de crédito, gestão de recebimentos e notas fiscais, expansão da gestão de clientes com novos controles e acompanhamentos, e diversas outras funcionalidades desenvolvidas até o momento. A figura 9 apresenta uma tela de *checkout* de pagamento por cartão de crédito do RD Contas.

Na sua estrutura de pagamentos, o modelo atual da aplicação suporta pagamentos nacionais e internacionais. Para isso são utilizados os seguintes meios de pagamentos:

- Boleto (aceito para transações nacionais);
- Cartões de crédito (aceito para transações nacionais e internacionais);
- Transferências bancárias (apenas para casos de exceção).

3.2 Formalização do problema

Atualmente, o pagamento recorrente via cartão de crédito se dá por meio de um *token* que é criado no *gateway* de pagamento. Este *token* é criado após a primeira transação realizada de forma manual pelo cliente, ou seja, o primeiro pagamento em que o cliente preenche os seus dados pessoais e do seu cartão de crédito.

Nos vencimentos de cada cobrança destes clientes com *token* salvo, acontece uma transação automática do RD Contas a fim de debitar da fatura do cliente. Com isso, o pagamento pode ser aprovado, quitando a cobrança, ou recusado, retornando um motivo de falha pelo *gateway* de pagamento.

São muitos os motivos que podem gerar erros na aprovação de uma transação como falha na conexão, limite insuficiente ou cartão inexistente, por exemplo. Hoje, independente do motivo das recusas de pagamento do cartão de crédito, são feitas retentativas diárias por no máximo 10 dias após o vencimento da fatura ou até o pagamento ser aprovado. Porém, esse cenário tende a comprometer o volume de inadimplência dos clientes da RD se não for gerenciado de uma maneira mais eficiente.

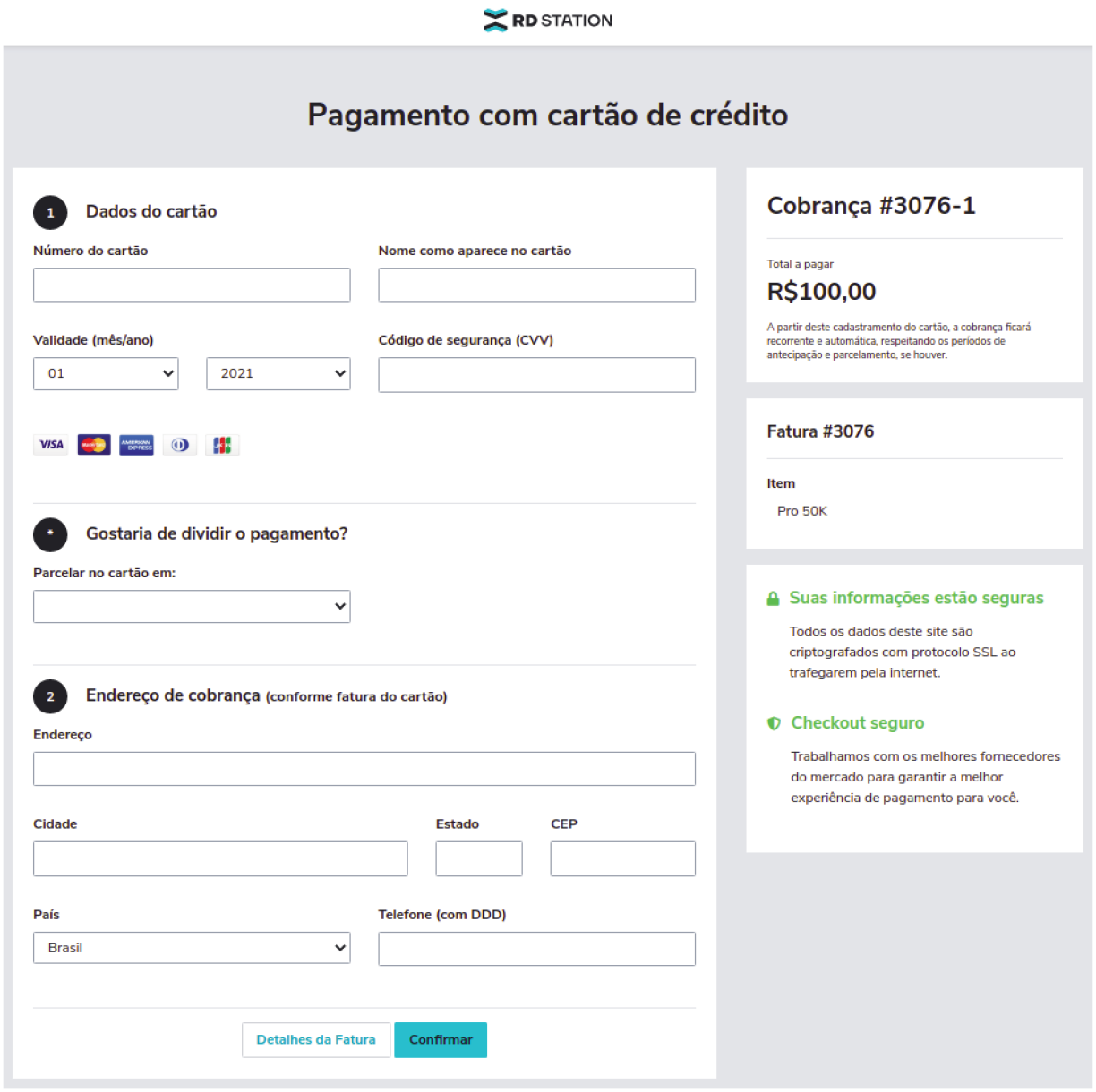

Figura 9 – Exemplo de tela de *checkout* de pagamento por cartão de crédito do RD Contas

Além de entender que a inadimplência de clientes num modelo de receita recorrente significa perda financeira para a empresa, o problema de não se ter um modelo eficiente nessas retentativas é que o número de requisições nos *gateways* de pagamento geram custos e um alto número de tentativas falhas podem prejudicar a imagem da empresa frente às bandeiras de cartões que podem impor multas ou bloqueios para próximas tentativas de transações.

Também, o RD Contas é um sistema que resolve problemas para outros modelos de empresas que trabalham com produtos de assinatura com receita recorrente. Assim, uma visão de futuro é que esse sistema também possa ser comercializado. Outros produtos do segmento possuem funcionalidades inteligentes nas retentativas de pagamentos [20], por isso existe uma necessidade de alterar a forma como o sistema trata esses casos, tanto para

atender melhor os clientes da empresa mas também para estar equiparado aos concorrentes de mercado.

3.3 Solução proposta

O retorno das requisições de pagamento fornecem o tipo de recusa quando a tentativa de pagamento não é autorizada. Entendendo que recusas do tipo *hard decline* não devem sofrer retentativas, a proposta de solução é fazer o cancelamento do *token* de cartão de crédito desses casos.

Analisando concorrentes do segmento foi identificado o uso de aprendizado de máquina para as retentativas de *soft declines*. Muitos afirmam aplicar soluções de inteligência artificial mas sem dar muitos detalhes sobre seus modelos, apenas indicam o uso de métodos de classificação supervisionada. Dessa forma, para as falhas de transações classificadas como *soft declines*, a proposta de solução é a criação de um modelo preditivo a partir dos dados disponíveis dos clientes e as suas transações sugerindo e fazendo as retentativas de pagamentos nos dias com as maiores chances de efetivação dos pagamentos.

4 Desenvolvimento da solução para *hard declines*

Hard declines são recusas de pagamentos por problemas que exigem uma correção com o emissor do cartão. Desse modo é altamente recomendado que parem de ser feitas retentativas em cartões que já sofreram algum *hard decline*. Além disso, por se tratarem de transações mais sensíveis, elas têm um peso maior em penalidades feitas pelas bandeiras de cartões de crédito, por isso é muito importante entender como essas recusas são tratadas no fluxo de pagamentos.

Dado esse contexto, foi feita uma análise das transações realizadas pela empresa e que sofreram recusas em 2020. A figura 10 representa a proporção de transações negadas pelo motivo de recusa.

Com isso, foi evidenciado que o grande detrator das transações negadas da empresa em 2020 foram as recusas *soft decline* e a solução para essas será detalhada no próximo capítulo. Mesmo assim as recusas *hard decline* representaram quase 25% das transações negadas do ano e merecem atenção. Ao fazer uma análise do fluxo que envolve recusas *hard decline* foram encontradas algumas oportunidades de melhorias que são apresentadas nas seções a seguir.

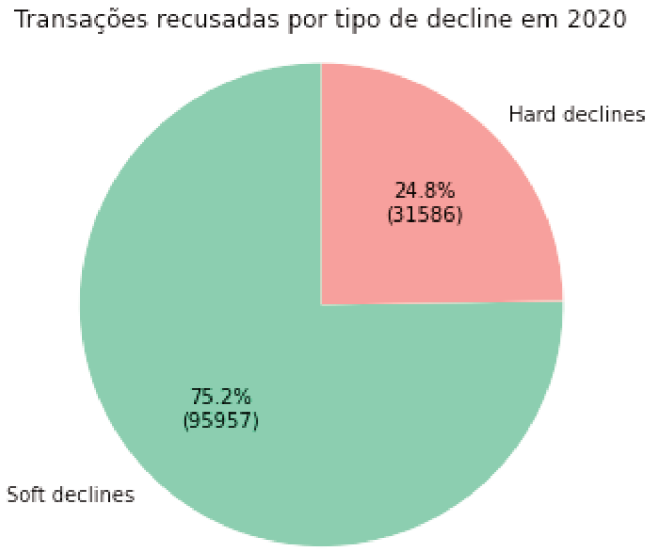

Figura 10 – Transações recusadas por tipo de *decline* em 2020

4.1 Separação das requisições enviadas para os *gateways*

Uma das características observadas nesse fluxo foi a forma que as requisições são feitas para os *gateways*. Basicamente as requisições de pagamento são divididas em duas partes: autorização e captura.

A autorização de uma transação representa o envio das informações do portador do cartão para o *gateway* de pagamentos e a reserva de saldo junto ao emissor. Então o emissor do cartão realiza a análise do limite de crédito e valida os dados enviados. Se houver alguma restrição no cartão do comprador, por exemplo a falta de saldo, ela será identificada no processo de autorização e retornará uma recusa da transação. Já no caso de uma autorização ser feita com sucesso, é realizada uma reserva do limite de crédito com o banco emissor e essa reserva fica pendente até a transação ser capturada.

A captura de uma transação consiste na confirmação de uma autorização que foi realizada previamente. É nessa etapa que é informado para o emissor do cartão que se deseja cobrar um determinado valor do seu portador. O valor a ser capturado deve ser igual ou inferior ao valor autorizado. Só depois da captura que o portador do cartão conseguirá visualizar um novo débito na sua fatura [21].

É com base no resultado das tentativas de capturas que as empresas são penalizadas pelas bandeiras de cartões de crédito, por isso uma das métricas mais importantes nesse segmento é a taxa de aprovação das transações, ou seja, o percentual de capturas que são aprovadas.

Até o presente projeto, as requisições de autorização e captura para os *gateways* de pagamento estavam sendo feitas independentemente do retorno da autorização, ou seja, mesmo com uma autorização negada, o RD Contas fazia uma tentativa de captura que também é negada. Desta forma a primeira oportunidade de melhoria encontrada foi a separação das requisições enviadas para os *gateways* de pagamentos.

4.2 Cancelamento de *tokens* que sofrem *hard decline*

Por se tratar de uma empresa SaaS, o modelo de negócio da RD consiste na cobrança recorrente dos seus clientes. Para os clientes que fazem pagamentos com cartão de crédito, um *token* é salvo no RD Contas após o primeiro pagamento manual, e nas cobranças seguintes o sistema é capaz de fazer cobranças automáticas no cartão do cliente.

Até o presente projeto, se a tentativa de cobrança do cartão do cliente tivesse um retorno de recusa, seja ela *soft* ou *hard decline*, o sistema estava configurado para continuar fazendo retentativas diárias por até 10 dias após o vencimento da fatura ou até o pagamento ser aprovado. Dado que uma recusa do tipo *hard decline* requer uma correção com o emissor do cartão, as retentativas nesses casos são inúteis e apenas prejudicam a

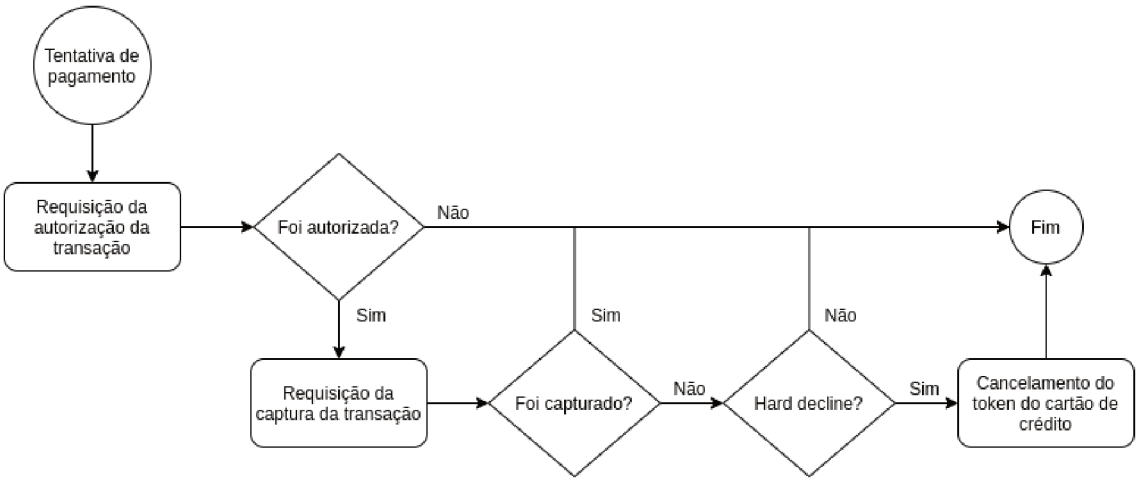

Figura 11 – Novo fluxo de requisições e cancelamento de *tokens* para recusas *hard decline*

taxa de aprovação das transações da empresa.

Desta forma, a solução encontrada para esses casos foi fazer o cancelamento dos *tokens* de cartões que sofrem uma recusa *hard decline*. A figura 11 ilustra o funcionamento do novo fluxo com a separação das requisições feita para os *gateways* e também o tratamento das recusas *hard decline* fazendo o cancelamento dos *tokens*.

5 Desenvolvimento da solução para *soft declines*

Soft declines são recusas de transações por motivo temporário e por isso permitem novas tentativas de pagamento. A análise das transações realizadas pela empresa e que sofreram recusas em 2020, apresentada pela figura 10, apontou que as recusas *soft decline* são as maiores detratoras das transações negadas. Desta forma, ao explorar as melhorias que podem ser implementadas no fluxo de pagamentos que envolve tais recusas foram encontradas as maiores oportunidades para atingir o objetivo desse projeto.

5.1 Análise exploratória

O primeiro passo para trabalhar nas melhorias das recusas *soft decline* foi a exploração e entendimento dos dados que envolvem os fluxos de pagamentos. Isso é muito importante porque é necessário ter um bom entendimento do problema antes de começar a obter dados de qualquer modo. Por isso a qualidade das análises está diretamente ligada com a qualidade dos dados disponíveis, e desta forma com boas análises se obtém bons dados para resolver um determinado problema.

Para iniciar a exploração, sem investir demasiado tempo em unir bases de informações ou tratar formatos de dados, foi feito um levantamento das transações feitas pelos clientes da RD no ano de 2020. O objetivo dessa exploração foi responder às seguintes perguntas:

- Qual a proporção entre transações aprovadas e negadas?
- Qual a proporção entre retentativas aprovadas e negadas?
- Quais os principais motivos de recusas *soft decline* que nossos clientes recebem?
- Quantos dias são necessários para aprovar um pagamento por retentativas?

Com base nessas perguntas, os dados disponíveis pelos *gateways* de pagamentos usados para as transações do RD Contas foram levantados e representados em alguns gráficos. Com a figura 1 pode-se concluir que praticamente metade das transações que passam pelo RD Contas é recusada e conforme a tabela 1 a média da taxa de aprovação de cartão de crédito dos principais processadores de pagamento gira em torno de 75% [4], dando para o sistema uma oportunidade de melhoria para elevar seus resultados.

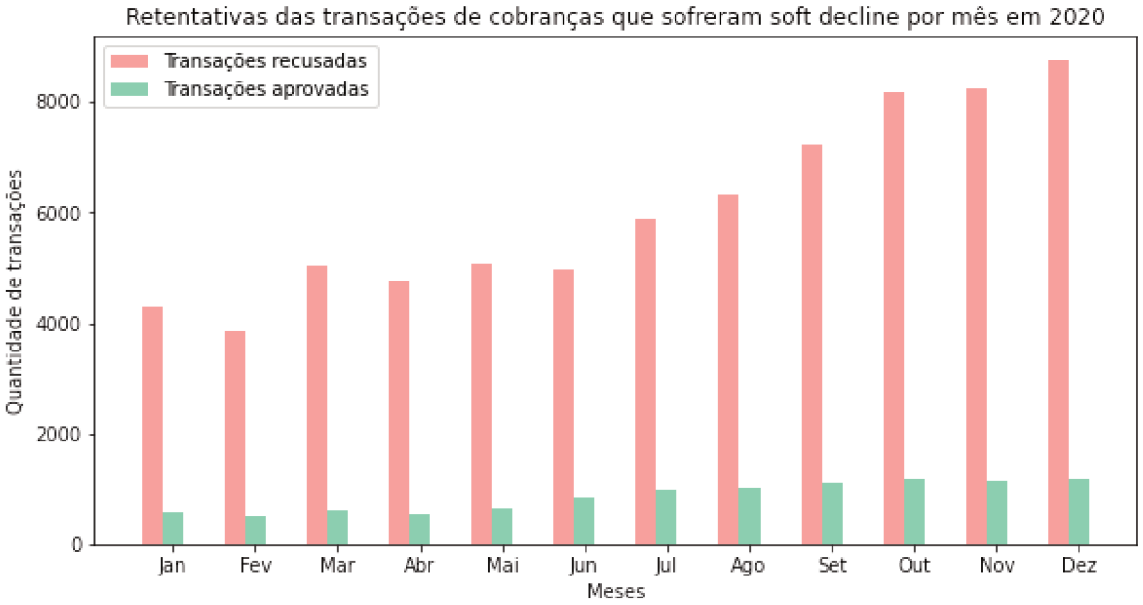

Figura 12 – Retentativas das transações de cobranças que sofreram *soft decline* em 2020

No que se refere a pagamentos com cartão de crédito não é possível se ter um controle sobre as informações preenchidas pelos clientes num *checkout*. O que acontece nessa etapa do fluxo são algumas validações de dados, por exemplo a quantidade de dígitos do cartão ou o endereço do portador são verificados. Esse tipo de validação já existe e garante um bom filtro antes de fazer requisições desnecessárias para os *gateways*. Mesmo assim, a figura 10 apontou que mais de 75% das transações negadas em 2020 foram por recusas do tipo *soft decline*. Por isso supõe-se que haja um problema maior nas requisições do processo de recorrência, ou seja, aquelas tentativas automáticas feitas pelo sistema. Com o intuito de analisar as transações das retentativas das cobranças que foram recusadas, foi feito um levantamento dos dados disponíveis e o resultado foi ilustrado na figura 12.

Os dados apontam que quase 90% das retentativas das cobranças que sofreram *soft decline* foram negadas, ou seja, a estratégia de retentativas está sendo muito ineficiente. Fica claro também que esse não é um comportamento sazonal, mas sim frequente, já que é observado em todos os meses do ano de 2020.

Determinado que há um problema na lógica de retentativas do RD Contas, buscou-se mais informações das tentativas de pagamento que passam por esse fluxo. Foram mapeados então os principais motivos de recusas do tipo *soft decline* com o objetivo de entender quais as principais causas de tantas transações negadas. A figura 13 determina que o grande detrator de pagamentos negados dos clientes da RD Station é o motivo "fundos insuficientes" com mais de 50.000 recusas. O segundo motivo com mais recusas é classificado como "*do not honour*", uma classificação genérica fornecida pelos *gateways* de pagamentos que pode indicar outros motivos, como problemas com saldo ou limite do cartão. Para a melhor classificação desse motivo existe um movimento das adquirentes de

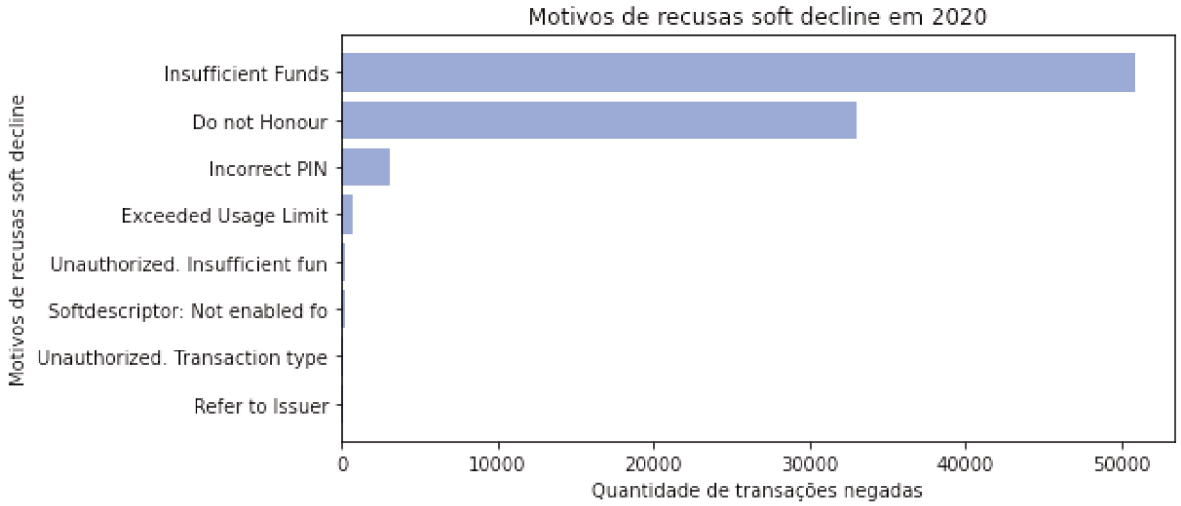

Figura 13 – Motivos de recusas *soft decline* em 2020

fornecer mais detalhes sobre essas recusas. Com o tempo esses motivos detalhados podem ajudar ainda mais na previsibilidade de aprovação ou recusa com os modelos a serem treinados. Todos os motivos apresentados são passíveis de retentativas e não há nenhuma limitação de dias para uma próxima tentativa de cobrança.

Por fim também foi analisado, com base nas cobranças que tiveram recusas *soft decline*, a quantidade de dias retentados até a aprovação do pagamento. A figura 14 apresenta tais informações. Pode-se perceber uma forte tendência aos dias iniciais, dado que a lógica de retentativas automática tenta cobrar o cliente diariamente até o dia 10 após o vencimento da fatura. Há então um pico no dia 11 e a justificativa para isso é que esse é o dia em que o serviço do cliente é bloqueado. Também são apresentadas novas retentativas após o dia 10, as quais representam tentativas manuais do time financeiro com o cartão salvo pelo cliente. Isso acontece no processo em que o time financeiro contata o cliente e este solicita uma nova tentativa após a regularização da sua conta.

Com base nas análises apresentadas nessa seção, pode-se concluir que o fluxo de retentativas das cobranças que sofreram *soft decline* é o grande detrator da taxa de aprovação das transações da RD Station. Tendo isso em vista, o projeto seguirá com o objetivo de reformular esse fluxo de pagamentos propondo uma lógica inteligente para tratar esses casos sugerindo com assertividade se a retentativa deve ou não ser realizada.

5.2 Processamento de dados

Com o problema bem claro e os objetivos definidos pode-se então buscar mais dados disponíveis e que tenham relação com o desafio proposto. Para este caso os dados de interesse estão relacionados com as informações enviadas para os *gateways* de pagamentos e também os dados dos clientes que podem influenciar na efetivação dos seus pagamentos.

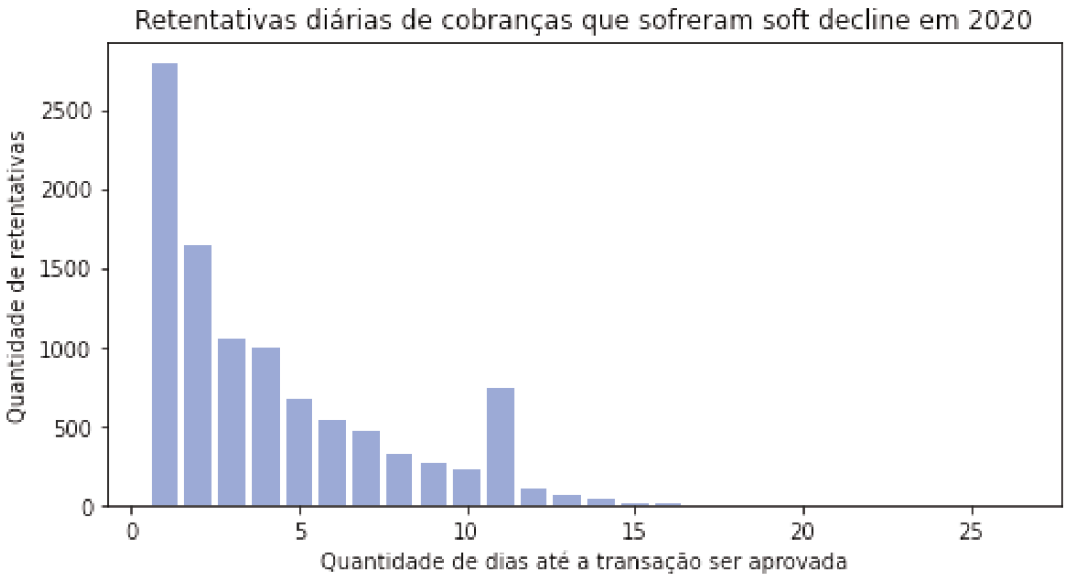

Figura 14 – Retentativas diárias de cobranças que sofreram *soft decline* em 2020

Segue abaixo uma lista dos dados relacionados com as transações e os dados dos clientes e uma descrição dos atributos. Além disso, também é explicado o interesse nesses atributos para a geração dos modelos e como eles são formatados para a aplicação das técnicas de classificação.

Dados das transações:

- **Retentativa**: inteiro que expressa o número de retentativas de pagamentos já realizadas até o momento. Por exemplo, se uma cobrança sofreu *soft decline* e recebeu mais 5 retentativas, o campo é preenchido pelo valor inteiro 5.
- **Data da transação**: *string* da data da tentativa de pagamento no formato $dd/mm/aaaa$.
- **Valor da transação**: *float* que representa o valor monetário que será cobrado do cliente.
- **Bandeira do cartão de crédito**: *string* que representa a bandeira do cartão do cliente.
- **Adquirente**: *string* que representa a adquirente a ser feita a transação.
- **Código de retorno da transação**: inteiro que representa o motivo de recusa da tentativa de pagamento, ou seja, a primeira tentativa que sofreu *soft decline*.

Dados dos clientes:

- **Preço do item de venda**: *float* que representa o valor tabelado do serviço contratado pelo cliente.

- **Quantidade de parcelas**: inteiro que expressa o número de parcelas que o cliente deseja fazer seu pagamento.
- **Dia do vencimento**: inteiro que representa o dia (do mês) do vencimento da fatura do cliente.
- **Data de ativação do serviço**: *string* da data em que o serviço do cliente foi ativado. Com essa data é possível identificar quanto tempo esse cliente faz uso dos serviços da RD Station.
- **Data de criação do *token* do cartão de crédito**: *string* da data em que o *token* do cartão de crédito do cliente foi criado no sistema. Com essa data é possível identificar quanto tempo esse cliente faz o pagamento com esse cartão de crédito.
- **Tipo da entidade**: booleano que representa se o cliente é uma pessoa física ou pessoa jurídica.
- **Nota fiscal antecipada**: booleano que representa se o cliente precisa da nota fiscal antecipada para o seu pagamento.

Como o objetivo do modelo deve prever com assertividade se uma retentativa deve ou não ser feita chegou-se ao entendimento de que esse caso poderia ser trabalhado como um problema de classificação. Assim o objetivo da máquina será de aprender a classificar uma entrada de dados em uma determinada categoria, ainda a ser definida.

Para a grande maioria dos métodos de classificação existe um pré-requisito que deve ser levado em consideração nesta etapa de tratamento de dados: a necessidade de todas as variáveis serem numéricas. Desta forma, alguns dos dados acima apresentados devem passar por algum tipo de transformação para cumprirem com esse requisito antes de avançarem para a validação com os algoritmos. Entre eles estão todas as variáveis relacionadas a datas e também as variáveis categóricas.

Para o atributo da data de ativação do serviço, o interesse foi descobrir quanto tempo o cliente usa os serviços da RD Station. Por isso a formatação desse dado foi feita ao transformar a data em quantidade de meses, identificando assim há quantos meses o cliente usa o serviço contratado. A hipótese para esse campo é de que um cliente que usa os serviços oferecidos há mais tempo não vai querer deixar de pagar a fatura correndo o risco de ter o seu acesso bloqueado. Algo parecido foi feito com o atributo da data de criação do *token* de cartão de crédito. O intuito desse atributo era fornecer a informação de quanto tempo (também em meses) o cliente usa esse cartão para fazer pagamentos. A hipótese nesse caso é de que um cliente que costuma fazer pagamentos num cartão a mais tempo, não o deixará ficar sem saldo ou ter outros problemas com ele. Já no atributo da data da transação foi feita uma simplificação ao se usar apenas o dia da transação. A

hipótese é de que exista certa sazonalidade com pagamentos, já que é comum pagamentos de salários caírem em determinado período do mês.

Como variáveis categóricas existem os atributos da bandeira do cartão de crédito e a adquirente que fará a transação. Para esses casos pode-se fazer o uso da técnica de variáveis *dummy*, a transformação de categorias em variáveis binárias (0 ou 1) criadas para representar uma variável [22]. O número de variáveis *dummies* a serem criadas sempre será $n-1$ categorias, já que uma das variáveis pode ser considerada a negação das demais. Para o tratamento de variáveis categóricas nesse projeto foi utilizado o método *get_ dummies* da biblioteca *Pandas* [23].

Com as variáveis nos formatos corretos foi possível começar a modelar o problema para a aplicação dos algoritmos. Nesse projeto foram propostas duas modelagens com saídas distintas como solução do problema em questão. As seções abaixo detalham a formulação dos modelos e a decisão de seguir com o que apresenta o melhor resultado.

5.3 Modelagem pela quantidade de retentativas por cobrança

A primeira concepção de modelo teve como intuito predizer, a partir de uma recusa *soft decline*, em quantos dias deveria ser feita uma nova tentativa para obter uma aprovação, podendo ser de 1 a 10 dias. Como os dados disponíveis são por transação, primeiro foi preciso fazer um agrupamento das transações referentes às mesmas cobranças buscando obter o número de retentativas necessárias até uma aprovação. A figura 15 representa o conjunto de entradas e saídas da modelagem apresentada nesta seção.

Nesse processo também foi identificado que, mesmo todas as variáveis sendo numéricas, havia uma discrepância grande entre elas que poderia impactar os algoritmos de classificação. Deste modo foi utilizada a técnica de normalização dos dados com o objetivo de transformar todas as variáveis para a mesma ordem de grandeza. Essa técnica faz com que as variáveis possuam uma média igual a 0 e um desvio padrão igual a 1 [24]. Para tratar a discrepância dos dados nesse projeto foi utilizada a classe *StandardScaler* da biblioteca *scikit-learn* [25].

Com os dados formatados e normalizados, foi feita uma separação em dados de treinamento e dados de teste para a validação com alguns algoritmos de classificação. Alguns concorrentes de mercado anunciam ferramentas de retentativas inteligentes que usam regressão logística e florestas aleatórias em suas lógicas [26]. Por causa disso foram usados esses dois classificadores para fazer esta primeira modelagem.

Usando os hiperparâmetros comuns desses algoritmos foram obtidos resultados de acurácia muito abaixo do esperado, entre 19% e 21%. Mesmo com a alteração dos hiperparâmetros desses modelos, a acurácia não chegou a ultrapassar dos 23%. Isso acontece

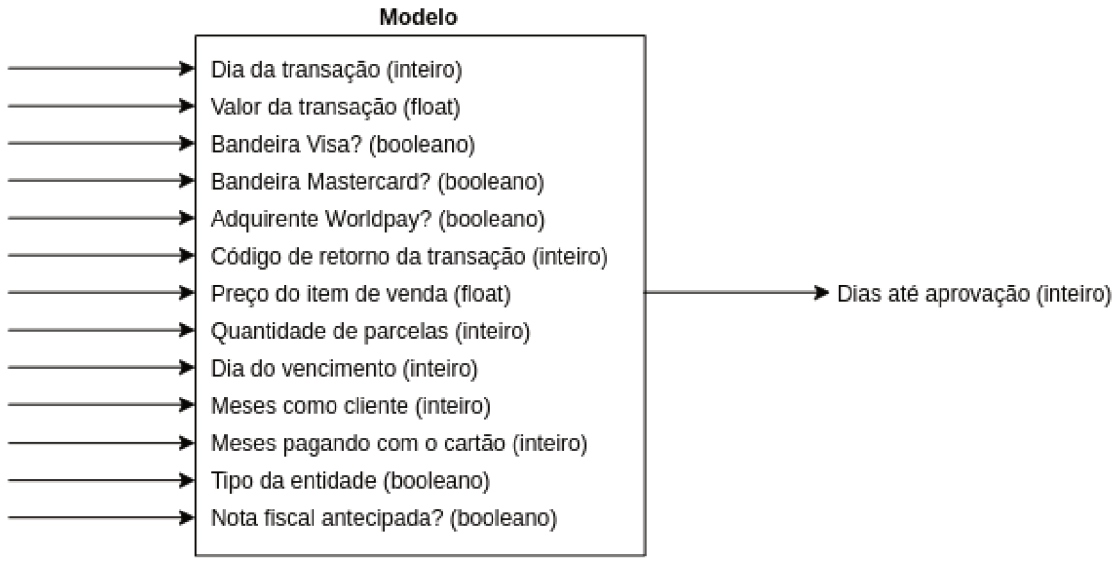

Figura 15 – Entradas e saídas da modelagem pela quantidade de retentativas por cobrança

porque nesse caso foi definida uma saída de multi classes com 10 possíveis valores de saída. Além disso, neste caso não seriam apenas os verdadeiros positivos da matriz de confusão considerados resultados aceitáveis. Por exemplo, se o modelo fez a previsão que uma retentativa seria aprovada no dia 5, mas na verdade ela foi aprovada no dia 4 isso pode ser considerado como um resultado positivo, já que a transação seria aprovada nos dias subsequentes. Apesar de não ser a previsão mais assertiva, o objetivo do projeto é fazer com que o número de retentativas negadas seja reduzido, e essa previsão contribui para isso.

As figuras 16 e 17 apresentam as matrizes de confusão da regressão logística e do classificador florestas aleatórias. Apesar de terem uma acurácia abaixo dos 21%, considerando aceitável uma previsão após o dia da real aprovação da transação, então o modelo da regressão logística teria um resultado positivo para 54,9% dos casos, e o modelo de florestas aleatórias para 45,9% deles.

Como os resultados para essa modelagem não foram satisfatórios foi proposta uma nova abordagem, apresentada a seguir.

5.4 Modelagem pela aprovação de transação

A segunda concepção de modelo teve como intuito predizer, para cada entrada de dados, se uma determinada transação seria aprovada ou negada. Desta vez, como os dados disponíveis são por transação, não foi necessário nenhum tipo de agrupamento de dados. Para esta abordagem foi adicionada uma nova entrada ao modelo, um valor inteiro que representa a retentativa de pagamento, e a saída foi alterada para um valor booleano representando o resultado da retentativa de pagamento: 1, se aprovada, e 0, se negada.

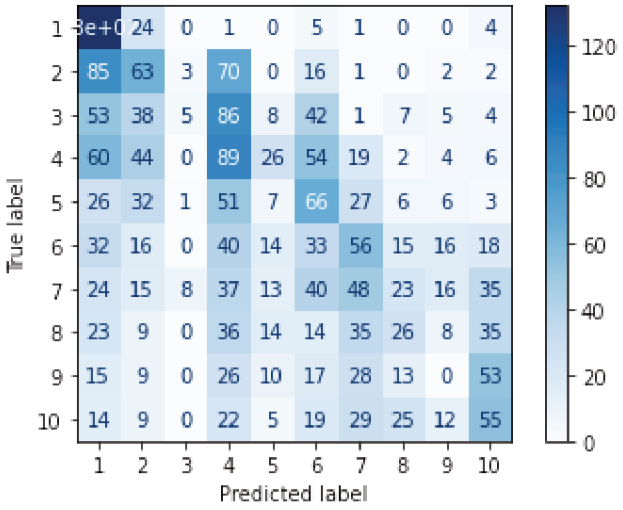

Figura 16 – Matriz de confusão do modelo com regressão logística

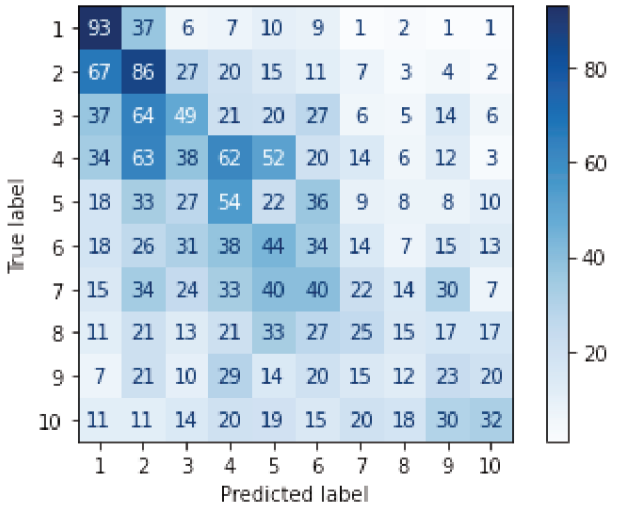

Figura 17 – Matriz de confusão do modelo com floresta aleatória

A figura 18 representa o conjunto de entradas e saídas da modelagem apresentada nesta seção.

Com os dados formatados e normalizados, novamente uma separação em dados de treinamento e dados de teste foi feita para uma validação inicial com regressão logística. O método teve uma acurácia de 71,65%, muito superior à modelagem anterior. Porém ao analisar sua matriz de confusão, representada na figura 19, pode-se perceber que poucas foram as predições de transações aprovadas.

Por ter um número muito superior de retentativas negadas em relação às retentativas aprovadas, o modelo é muito influenciado a predizer grande parte das transações como negadas. Observando a matriz de confusão é perceptível a discrepância entre as predições de retentativas aprovadas e negadas pelo modelo.

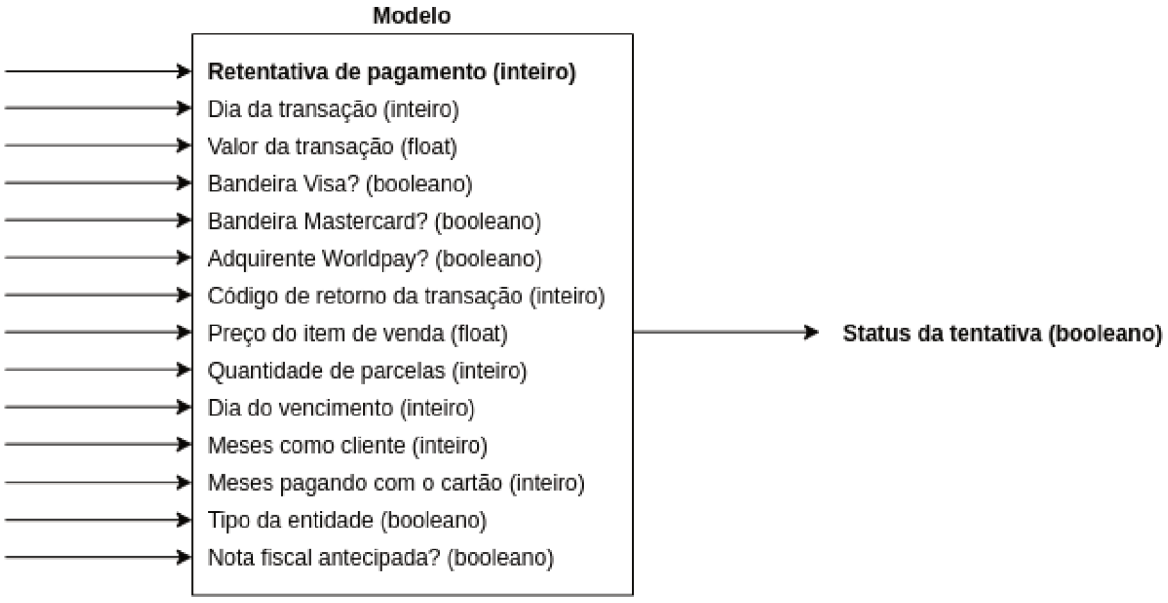

Figura 18 – Entradas e saídas da modelagem pela aprovação da transação

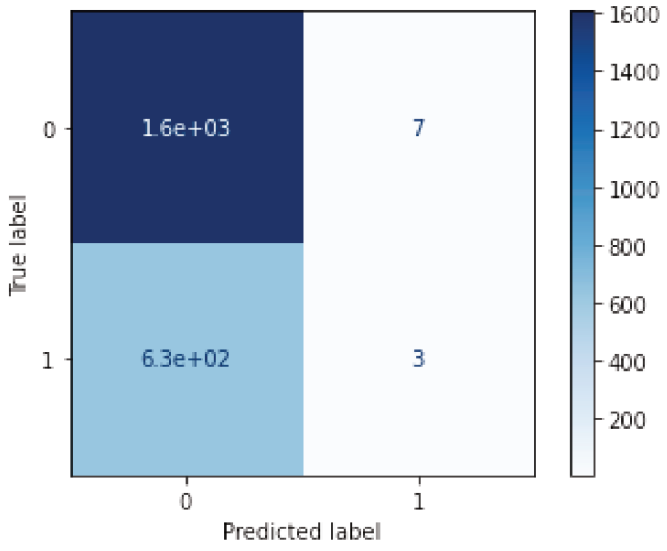

Figura 19 – Matriz de confusão do modelo com regressão logística

Esse problema é conhecido como desbalanceamento dos dados, quando um dataset possui muitos exemplos de uma classe e poucos exemplos da outra classe, o que aconteceu nesse caso. A grande consequência disso é que esse desequilíbrio causará no modelo uma tendência a dar muitos falsos positivos ou falsos negativos, que foi o que aconteceu nesse caso também. Para corrigir o problema de forma simples, existem basicamente dois caminhos:

- ***Over-sampling***: é uma técnica que cria novas observações da classe minoritária com base nas informações dos dados originais. A geração de novas entradas pode ser feita aleatoriamente com o auxílio de técnicas de clustering ou sinteticamente. A vantagem de usar esse método é que nenhuma informação é descartada, mas o custo computacional será elevado, além de poder deteriorar a performance do algoritmo para as classes minoritárias [27].

- ***Under-sampling***: é um método que irá reduzir o desbalanceamento do dataset eliminando aleatoriamente entradas da classe com maior número de ocorrências. Essa técnica extrai um subconjunto aleatório da classe majoritária. Com isso, são preservadas as características da classe minoritária, fazendo com que o método seja ideal para situações onde se tenha grandes volumes de dados.

Este projeto não está lidando com um grande volume de dados disponível, por isso descartar dados se torna uma opção inviável, apesar de ter sido testada como experimento. Desta forma optou-se por seguir com a técnica de over-sampling para corrigir o desbalanceamento dos dados de entrada e o código implementado é demonstrado abaixo:

```
# contar classes do atributo "approved"
count_class_0, count_class_1 = df_training.approved.value_counts()

# dividir dataset pelas classes
df_class_0 = df_training[df_training["approved"] == 0]
df_class_1 = df_training[df_training["approved"] == 1]

# oversampling da classe com poucos registros
df_class_1_over = df_class_1.sample(count_class_0, replace=True)
df_train_over = pd.concat([df_class_0, df_class_1_over], axis=0)

# print da nova distribuicao entre as classes
print("Random over-sampling:")
print(df_train_over.approved.value_counts())
```

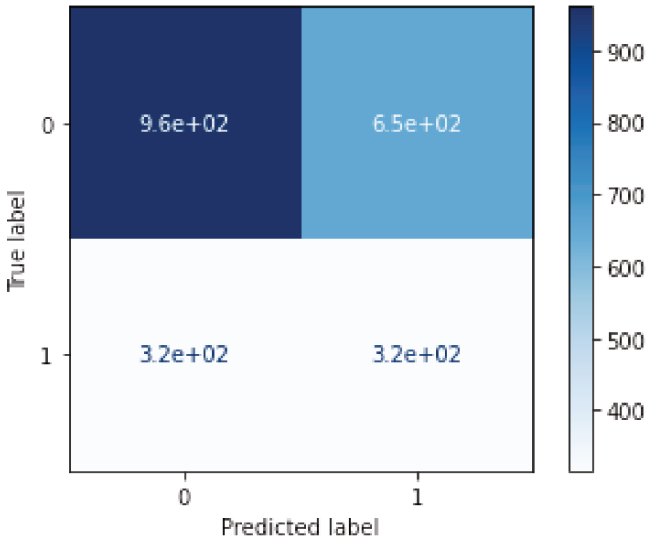

Figura 20 – Matriz de confusão do modelo com regressão logística e *over-sampling*

Com os dados balanceados, novamente uma validação foi realizada usando o método de regressão logística. A figura 20 representa a matriz de confusão obtida. O modelo de regressão logística obteve uma acurácia de 56,2% e a forma como foi definida a variável de saída possibilita um nível maior de experimentação que o modelo anterior. Dado o resultado obtido e a aplicação com o modelo de desenvolvimento de funcionalidades do RD Contas, foi decidido implementar os outros métodos de classificação com base nesse conjunto de dados de entradas e saída.

5.5 Aplicação dos algoritmos de classificação

Com o objetivo de se definir o melhor modelo a partir dos mais conhecidos métodos de classificação, o projeto teve o intuito de testar 6 algoritmos distintos. Os classificadores usados para a geração do modelo de aprendizagem de máquina foram: Regressão Logística, Árvores de Decisão, Florestas Aleatórias, Classificador Naive Bayes, K-ésimo Vizinho mais Próximo e Máquina de Vetores de Suporte de Suporte.

Todos os métodos de classificação utilizados foram testados inicialmente com seus hiperparâmetros padrões. A tabela 2 apresenta as métricas de avaliação para cada um deles.

5.6 Escolha e validação do modelo

Para a escolha do modelo de classificação foi importante entender qual resolveria da melhor forma o problema em estudo. Neste caso, o objetivo é a redução das retentativas de uma cobrança que sofreu uma recusa do tipo soft decline, ou mais especificamente,

Tabela 2 – Métricas de desempenho dos modelos de classificação

Classificador	Acurácia	Precisão	Recall	F1-score
Regressão Logística	56,9%	32,9%	50,1%	39,6%
Árvores de Decisão	55,6%	29,3%	40,9%	34,2%
Florestas Aleatórias	**62,2%**	**34,2%**	37,0%	35,5%
Naive Bayes	28,0%	28,1%	**99,5%**	**43,8%**
K-ésimo Vizinho mais Próximo	56,9%	32,7%	50,1%	39,6%
Máquina de Vetores de Suporte	60,7%	33,6%	40,4%	36,7%

aumentar a taxa de aprovação dessas retentativas. Para isso um algoritmo com uma acurácia mais elevada tende a atender melhor esse critério. Além disso, o número de falsos negativos não é o maior problema para o modelo, já que o seu comportamento fará apenas com que se leve mais dias a se conseguir cobrar o cliente. Os maiores vilões nesse caso são os falsos positivos, pois estes representam predições que um pagamento será aprovado enquanto na verdade é negado, ou seja, o sistema faria retentativas desnecessárias que iriam contra o objetivo do projeto.

Tendo esses pontos bem claros, não foi difícil identificar que o modelo de florestas aleatórias foi o classificador que melhor atendeu as expectativas do projeto. Também não foi uma grande supresa dadas as características desse modelo e também por ser um modelo usualmente utilizado por outras empresas que oferecem estratégias de retentativas inteligentes para *soft declines* [26]. Desta forma foi tomada a decisão de seguir com esse modelo e otimizar os seus parâmetros antes de desenvolver a lógica para o fluxo de pagamentos.

5.6.1 Otimização dos hiperparâmetros do modelo selecionado

Para fazer uma otimização do modelo selecionado foram testados diferentes valores dos hiperparâmetros [28]. As variações testadas estão demonstradas abaixo:

```
{
    "bootstrap": [True, False],
    "max_depth": [10, 50, 100],
    "max_features": ["auto", "sqrt"],
    "min_samples_leaf": [1, 2, 4],
    "min_samples_split": [2, 5, 10],
    "n_estimators": [10, 100, 200, 500, 1000]
}
```

Por meio de uma busca exaustiva com validação cruzada, os modelos tiveram que passar por uma nova validação das suas métricas. As tabelas 3 e 4 representam alguns

dos hiperparâmetros testados e as respectivas métricas obtidas para eles [29]. A busca exaustiva foi realizada usando a classe *RandomizedSearchCV* da biblioteca *scikit-learn* [30] e o código implementado é demonstrado abaixo:

```
from pprint import pprint
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier

# variacao dos hiperparametros
n_estimators = [10, 100, 200, 500, 1000]
max_features = ["auto", "sqrt"]
max_depth = [10, 50, 100]
min_samples_split = [2, 5, 10]
min_samples_leaf = [1, 2, 4]
bootstrap = [True, False]

random_grid = {"n_estimators": n_estimators,
               "max_features": max_features,
               "max_depth": max_depth,
               "min_samples_split": min_samples_split,
               "min_samples_leaf": min_samples_leaf,
               "bootstrap": bootstrap}
pprint(random_grid)

# criacao do modelo base e aplicacao dos hiperparametros
rf = RandomForestClassifier()
rf_random = RandomizedSearchCV(estimator = rf,
                               param_distributions = random_grid,
                               n_iter = 100,
                               cv = 3,
                               verbose = 2,
                               random_state = 0,
                               n_jobs = -1)
rf_random.fit(x_train, y_train)

# melhores parametros aplicados ao modelo
rf_random.best_params_
```

O modelo que obteve o melhor percentual de acurácia foi configurado com os hiperparâmetros listados a seguir:

```
{
    "bootstrap": False,
    "max_depth": 100,
    "max_features": "sqrt",
    "min_samples_leaf": 1,
    "min_samples_split": 2,
    "n_estimators": 10
}
```

Portanto, este foi o modelo escolhido para seguir a implementação desse projeto.

Tabela 3 – Hiperparâmetros do classificador de florestas aleatórias

	bootstrap	max depth	max features	min samples leaf	min samples split	n estimators
1	True	10	'auto'	1	2	10
2	True	10	'auto'	2	5	100
3	True	10	'auto'	1	5	200
4	True	50	'auto'	4	10	500
5	True	100	'sqrt'	1	2	1000
6	False	10	'auto'	1	2	1000
7	False	10	'auto'	2	5	500
8	False	10	'auto'	1	5	200
9	False	50	'auto'	4	10	100
10	False	100	'sqrt'	1	2	10

Tabela 4 – Métricas de desempenho do classificador de florestas aleatórias

	Acurácia	Precisão	Recall	F1-score
1	60,6%	34,9%	46,3%	39,8%
2	57,0%	32,8%	50,2%	39,7%
3	57,3%	33,0%	50,1%	39,8%
4	58,2%	34,0%	**51,3%**	**40,9%**
5	60,7%	34,4%	43,8%	38,5%
6	58,5%	33,7%	48,8%	39,9%
7	58,0%	33,4%	49,4%	39,9%
8	58,4%	33,3%	47,7%	39,2%
9	59,1%	34,2%	49,3%	40,4%
10	**66,1%**	**36,7%**	27,8%	31,6%

5.6.2 Proposta da lógica de implementação do modelo

Ao analisar a matriz de confusão do modelo escolhido na seção anterior, foi observado que o número de predições de retentativas negadas era muito superior às predições de

retentativas aprovadas. Esse é um comportamento esperado dado que o balanceamento dos dados é feito apenas no conjunto de dados de treinamento, e não no de teste. O problema disso é a possibilidade de o modelo sugerir que não haja nenhuma retentativa para uma determinada cobrança. Com esse cenário em mente, foi preciso criar uma lógica que utilizasse o modelo treinado, mas que pudesse adicionar regras para essa rotina de retentativas de pagamentos.

Dado o contexto, foi muito importante nessa etapa do projeto voltar a estudar o negócio da empresa e rever os objetivos no desenvolvimento desse modelo de aprendizado de máquina. Na concepção do projeto idealizou-se que as retentativas de pagamentos fossem feitas nos dias mais propensos a uma aprovação. A importância disso é justamente ao pensar que, se o modelo sugerir que nenhuma retentativa fosse feita, ou pelo contrário, que retentativas fossem feitas todos os dias, isso não atenderia aos requisitos desejados, já que se faz necessário um aumento na efetivação das retentativas e uma redução em sua quantidade.

O modelo até então tinha sido treinado para prever se uma determinada transação seria aprovada ou recusada, de forma que a saída do sistema fosse um valor booleano sugerindo ou não a retentativa ser feita. Mesmo implementado dessa forma, as métricas de avaliação desse modelo foram as mais positivas de todas as opções testadas, o que significa que esse modelo obteve a maior capacidade de predição de uma transação ser aprovada. Porém, tendo a saída um valor booleano não torna possível a possibilidade de identificar quais os melhores dias para que o sistema tente fazer uma retentativa de pagamento.

Revendo o fluxo de pagamentos foi relembrado que existe uma régua de retentativas que podem ser feitas em até 10 dias. Nesse período, com base nas regras estabelecidas pelas bandeiras dos cartões de crédito, seria recomendado que até 3 retentativas fossem realizadas. Portanto dois parâmetros foram estabelecidos nesse processo: a quantidade de dias da régua de retentativas e o número de retentativas desejada pelo sistema.

Tendo por base um requisito de se fazer 3 retentativas em um período de 10 dias, o modelo treinado até então deveria ser capaz de sugerir as datas mais propensas a aprovarem os pagamentos. Foi aí que uma alteração simples, mas necessária, precisou ser feita no modelo. Ao invés de fornecer um valor booleano, o modelo também é capaz de fornecer a probabilidade de uma transação ser aprovada, de acordo com os dados de entrada que se referem a uma retentativa de pagamento para um determinado dia. Para isso a alteração técnica envolveu deixar de usar o método *predict* para o método *predict_proba* do classificador de floresta aleatória da biblioteca *scikit-learn* [31].

Desta forma buscou-se desenvolver uma lógica que, ao fazer uso do modelo treinado nesse projeto, pudesse predizer e sugerir os melhores dias para uma retentativa ser aprovada. Para isso, a partir de uma cobrança que sofreu uma recusa do tipo *soft decline*, 10 conjuntos de dados são selecionados, cada um representando uma possível retentativa dentro dos 10

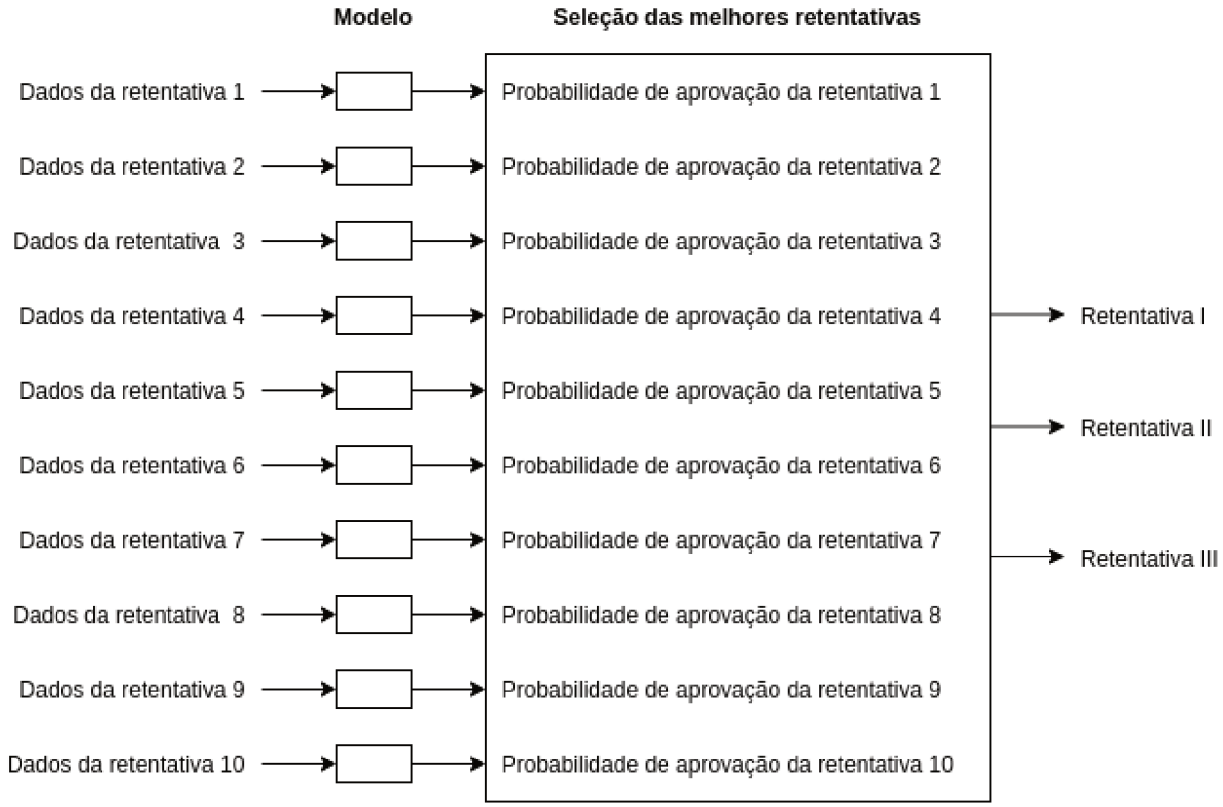

Figura 21 – Processamento dos conjuntos de entradas e seleção das melhores retentativas

dias da régua de pagamentos da empresa. Com base nessas entradas o modelo prevê a probabilidade de aprovação para cada uma das retentativas. Por fim devem ser selecionadas as 3 retentativas com as maiores chances de aprovação e essas devem ser agendadas. A figura 21 demonstra o processamento dos 10 conjuntos de entradas e a seleção das 3 retentativas com maiores probabilidades de aprovação.

Para validar a concepção da lógica proposta, um script foi desenvolvido para formatar os dados de entrada do modelo, receber as probabilidades da saída e definir os melhores dias para se fazerem as retentativas de pagamentos. A partir de dados reais das cobranças e transações dos clientes da RD Station, uma simulação foi realizada com o objetivo de verificar a eficácia dessa lógica. As métricas de validação dos classificadores não se aplicam de forma direta para avaliar essa solução, então elas foram adaptadas para:

- **Taxa de sucesso das retentativas**: proporção da quantidade de cobranças que tiveram retentativas de pagamentos aprovadas em relação ao total de cobranças.
- **Taxa de melhoria das retentativas**: comparação entre a quantidade de tentativas feitas com a nova lógica e o modelo desenvolvido, e a quantidade de tentativas da rotina usada pelo sistema vigente.
- **Acurácia**: proporção da quantidade de retentativas com aprovação do pagamento em relação a quantidade total de retentativas da nova lógica.

Com base nessas métricas adaptadas, a lógica proposta foi validada obtendo os seguintes resultados:

- **Taxa de sucesso das retentativas**: 98,9% das cobranças teriam alguma retentativa de pagamento aprovada.
- **Taxa de melhoria das retentativas**: a lógica proposta realizaria 60,3% menos tentativas que o sistema.
- **Acurácia**: das retentativas propostas pela nova lógica, 70,0% teriam a aprovação da transação.

Dado que foram obtidos resultados satisfatórios em todas as métricas adaptadas, a lógica de retentativas por agendamento foi validada com o uso do modelo de florestas aleatórias.

5.7 Implementação da solução em produção

Uma solução comumente utilizada para a implementação de modelos de classificação que devem se comunicar com aplicações web é a construção de APIs, as quais facilitam a conexão e comunicação entre os softwares permitindo a transferência de dados de forma confiável.

As APIs podem conter modelos por aprendizagem offline, quando o modelo é treinado uma vez em dados históricos, o que aconteceu até então neste projeto [32]. Geralmente se segue esse caminho para validar um fluxo ponta a ponta em produção, ao passo que com o tempo são feitos novos treinamentos do modelo com a coleta de novos dados.

Nesta etapa do projeto foram feitas análises sobre as tecnologias disponíveis e esboçadas diferentes formas de implementação de APIs para que pudessem se comunicar com o RD Contas sem gerar um acoplamento ao sistema. Desta forma foi idealizada uma solução de uma aplicação desacoplada ao sistema de contas da RD Station podendo assim ser desenvolvido de forma independente e possibilitando a sua conexão com outros sistemas. O esboço da aplicação e seu funcionamento são ilustrados na figura 22.

A API apresentada possui o seguinte funcionamento: partindo do processo de recorrência de pagamentos, quando o sistema tenta fazer a primeira tentativa de cobrança de uma fatura e recebe uma recusa do tipo *soft decline*, então os dados dessa fatura são enviados para a API. Ela recebe esses dados, faz uma normalização para que eles sejam formatados corretamente e processa 10 conjuntos de entrada no modelo, cada um representando uma possível retentativa nos próximos 10 dias. O modelo então responde com as probabilidades de cada tentativa de pagamento ser aprovada e em seguida a aplicação

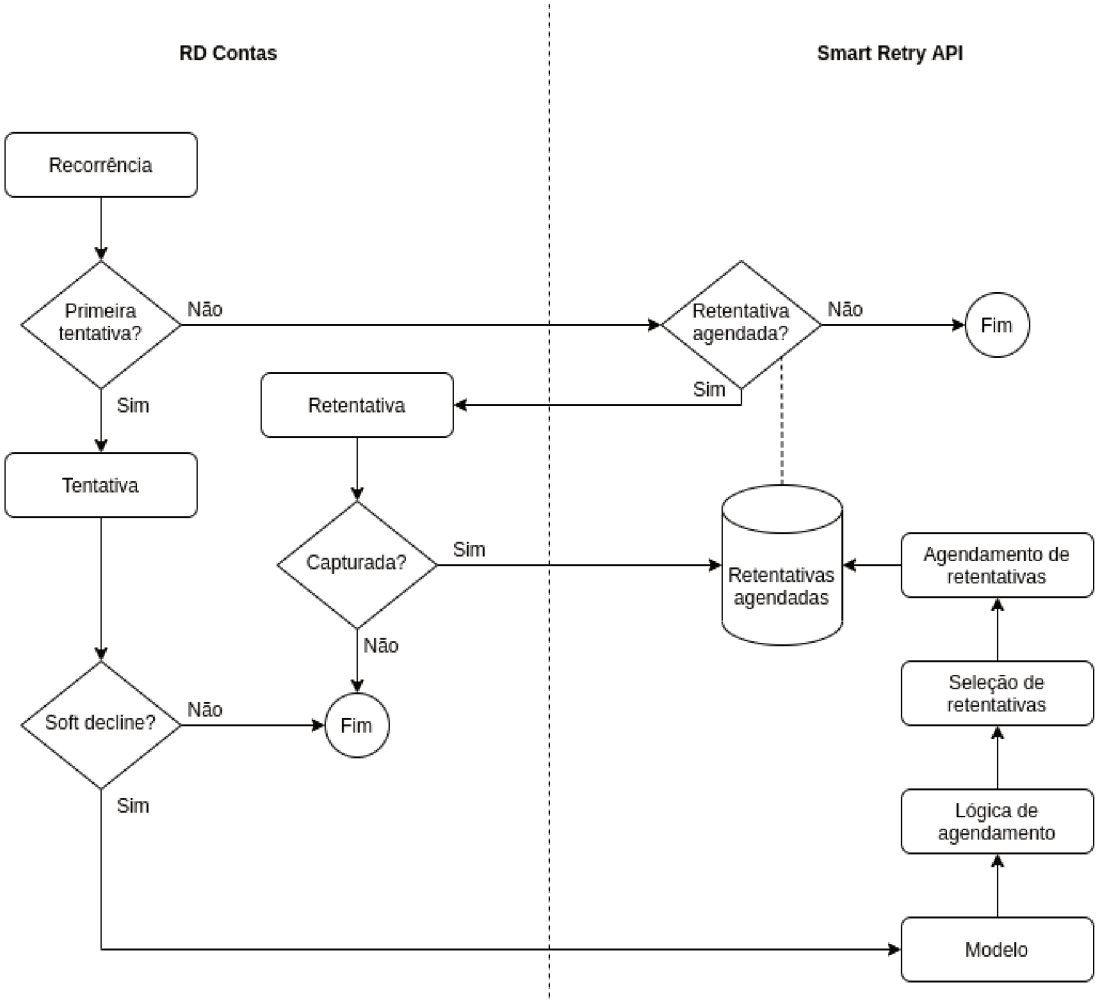

Figura 22 – Fluxograma do modelo e lógica *Smart Retry*

seleciona as 3 retentativas com as maiores probabilidades, registrando as informações numa tabela de um banco de dados de agendamento.

Para o sistema fazer uma nova tentativa é necessária uma requisição para a API consultando se há um agendamento de retentativa para um determinado dia. Caso haja uma retentativa agendada ela é executada no RD Contas e, se o pagamento for capturado, essa informação é persistida no banco de dados da aplicação.

Uma das vantagens dessa solução é manter o sistema de contas da RD Station com menor complexidade, além de possibilitar uma futura automação do treinamento do modelo com a coleta de novos dados que chegam na base de dados da aplicação, fazendo com que o modelo seja atualizado periodicamente.

6 Resultados em produção

6.1 Solução para *hard declines*

A primeira atuação desse projeto tinha como intuito a redução de recusas por *hard declines*. Em agosto de 2020 as requisições enviadas para os *gateways* de pagamentos foram divididas entre autorização e captura, acabando com a tentativa de capturas quando um pagamento não é autorizado pela emissora do cartão de crédito do cliente.

Num segundo momento, as transações com retorno do tipo *hard decline* começaram a ser tratadas de uma forma diferente. Para que estas não tivessem retentativas de pagamentos foi implementado um novo serviço para que os *tokens* de cartão de crédito fossem cancelados nesse cenário.

O resultado dessas ações pode ser visto na figura 23. Nela é possível notar uma primeira queda no número de recusas *hard decline* em agosto de 2020, com a divisão das requisições para os *gateways* de pagamento. Já nos meses seguintes a queda continua acontecendo com a implementação do serviço de cancelamento de *tokens* quando recebem um retorno do tipo *hard decline*.

No final de 2019 o time financeiro fez um levantamento de que a taxa de aprovação das cobranças em cartões de crédito da RD Station estava abaixo dos 45%, já no final de 2020 com essas primeiras melhorias implementadas **a taxa de aprovação da empresa subiu para 55%**.

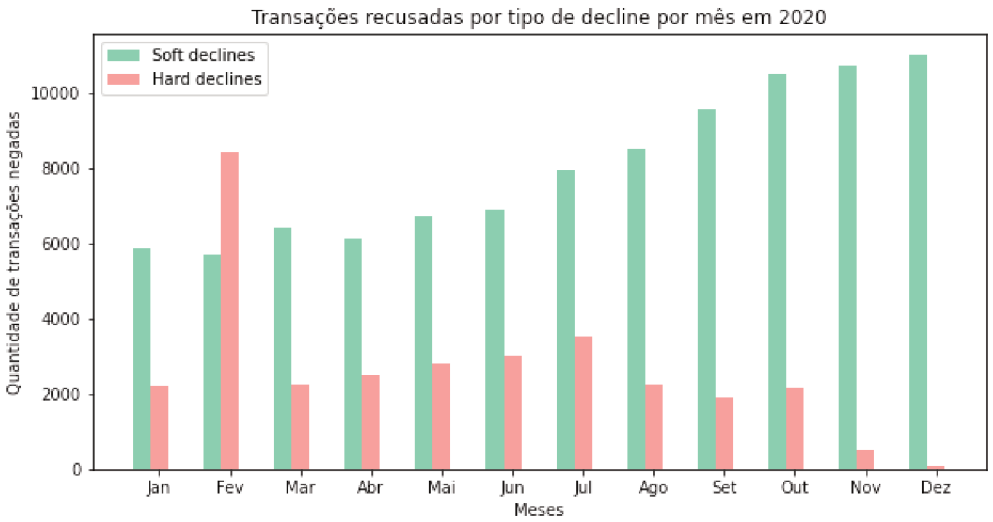

Figura 23 – Transações recusadas por tipo de decline por mês em 2020

6.2 Solução para *soft declines*

Apesar de as recusas do tipo *hard decline* serem mais sensíveis para os clientes, elas têm um impacto menor nas taxas de aprovação da RD Station. Enquanto isso, as recusas do tipo *soft decline* são as grandes detratoras no fluxo de pagamentos da empresa.

Até o momento da escrita deste trabalho, a solução que envolve o desenvolvimento de uma API, estava começando a ser testada em produção. Por isso, a validação da solução que envolve o modelo de florestas aleatórias e a nova lógica de retentativas de cobranças que sofreram recusas *soft decline* foi feita de forma experimental com clientes em um grupo controlado.

Para isso 90 clientes selecionados de forma aleatória e que tiveram cobranças com recusas *soft decline* foram removidos da rotina automática de recorrência de pagamentos do RD Contas, para que suas retentativas fossem realizadas de acordo com a lógica proposta neste trabalho. Após os dados desses clientes serem coletados e tratados, foram inseridos no modelo desenvolvido e escolhidas as 3 datas com maiores probabilidades de aprovação em uma janela de 10 dias. Com base nessas datas, as retentativas de pagamentos foram feitas de forma manual e os dados com o retorno dos *gateways* registrados. Ao final do experimento foram observados os seguintes resultados:

- Taxa de melhoria de retentativas: foram feitas **72,6% menos transações** com a lógica proposta nesse trabalho **em relação ao funcionamento do sistema vigente**.
- Acurácia: **75% das retentativas** realizadas pela lógica proposta **foram aprovadas**.

Em 2021 a previsão de custos com transações feitas pelos *gateways* de pagamentos da RD Station ultrapassa os R$200.000,00. A lógica apresentada nesse trabalho está nas fases finais de implementação mas a perspectiva é que, com base nos resultados experimentais e as proporções de recusas soft decline, esta solução **economize para a empresa um valor de cerca de R$65.000,00**. Além disso, com essa mesma redução nas retentativas de recusas soft decline, a perspectiva é que **a taxa de aprovação da empresa passe de 55% para 73%**, se mantendo livre das penalidades das bandeiras de cartões de crédito e se equiparando a outros processadores de pagamento do mercado.

7 Conclusões e Perspectivas

Este documento apresentou uma aplicação de técnicas de aprendizado de máquina com o objetivo de reduzir a quantidade de tentativas de pagamento recorrente com recusas do tipo *soft decline*. Foram estudados diversos métodos de classificação e gerados modelos para cada um deles com o intuito de entender as suas características, vantagens e desvantagens. Além disso, diversas técnicas de tratamento de dados foram realizadas para que fosse possível o desenvolvimento desse projeto.

O trabalho com métodos de classificação, e de aprendizagem de máquina em geral, é muito facilitado pelo amplo conhecimento disponível na internet. É muito fácil encontrar códigos abertos que ajudem o projetista e desenvolver uma solução nessa área. A maior dificuldade encontrada é justamente lidar com dados em plataformas distintas, com formatos diferentes ou até mesmo a falta deles.

Para se fazer uma modelagem adequada, foi necessário fazer uma análise exploratória em cima dos dados disponíveis, buscando ter um entendimento claro do problema antes de mexer em qualquer linha de código. Alinhado a isso, saber avaliar modelos de classificação com base em métricas foi essencial para se tomar boas decisões ao decorrer do projeto.

Os problemas relacionados a dados podem ser tratados com as diversas técnicas disponíveis. Neste projeto, usar os métodos de *over-sampling* para a ampliação do dataset disponível, alterar os hiperparâmetros para obter melhores resultados de acurácia, e validação cruzada para ampliar a capacidade de generalização do modelo, foi muito importante. Mas mesmo com todos esses passos, se o projetista não tiver capacidade analítica para lidar com os problemas fica muito difícil encontrar uma solução. Na etapa de validação do modelo de florestas aleatórias, entender que havia a oportunidade de olhar o problema de uma outra forma, fazendo o agendamento das retentativas com maiores probabilidades de aprovação foi essencial para se chegar ao resultado entregue ao final desse trabalho. Assim, mesmo sem uma base de dados ideal, sempre é possível encontrar alternativas que superem os obstáculos do projeto.

Este trabalho possibilitou o desenvolvimento de um modelo de aprendizado de máquina capaz de predizer a probabilidade de transações serem aprovadas ou negadas. Alinhado a uma nova estratégia no fluxo de pagamentos da RD Station, a empresa poderá obter uma economia de até R$65.000,00 em 2021 e um aumento de quase 20 pontos percentuais na taxa de aprovação de cartões de créditos dos seus clientes.

Como possibilidade de trabalhos futuros, sugere-se fazer novas modelagens do problema, já que a solução proposta mudará completamente o comportamento dos dados, por isso novos modelos devem ser treinados e implementados. Além disso, a solução

implementada foi de um modelo treinado por aprendizagem offline [32]. A expectativa é que essa solução evolua e o treinamento do modelo seja automatizado com a coleta de novos dados, fazendo com que o modelo seja atualizado periodicamente.

Ao final deste projeto, fica claro que a engenharia e a tecnologia conseguem contribuir muito em diversas áreas de trabalho em que seja necessária a análise matemática para otimização de processos internos. Com o progresso no desenvolvimento de novas tecnologias e o acesso a elas, existe uma grande perspectiva de que cada vez mais projetos de aprendizado de máquina sejam aplicados em frentes de diversas áreas de conhecimento, encontrando assim soluções inovadoras para todo tipo de problema [33].

Referências

1 ABECS. *MEIOS ELETRÔNICOS DE PAGAMENTO - BALANÇO 2020*. 2020. Disponível em: <https://api.abecs.org.br/wp-content/uploads/2021/02/Balan%C3%A7o-do-Setor-4%C2%BA-Trimestre-de-2020-Apresenta%C3%A7%C3%A3o.pdf>. Citado 2 vezes nas páginas 17 e 21.

2 ALVES, P. *Pesquisa da Mastercard indica aumento nos pagamentos digitais e crescimento do e-commerce*. 2020. Acessado em 25/04/2021. Disponível em: <https://www.mastercard.com/news/latin-america/pt-br/noticias/comunicados-de-imprensa/pr-pt/2020/julho/pesquisa-da-mastercard-indica-aumento-nos-pagamentos-digitais-e-crescimento-do-e-commerce/>. Citado na página 17.

3 RIBEIRO, R. *Recorrência SaaS: por que é tão importante e como implantar?* 2020. Acessado em 25/04/2021. Disponível em: <https://www.iugu.com/blog/recorrencia-saas>. Citado 2 vezes nas páginas 17 e 21.

4 DIGITAL MANAGER GURU. *[Infográfico]Taxa de aprovação dos cartões de crédito*. 2019. Acessado em 25/04/2021. Disponível em: <https://blog.digitalmanager.guru/pt/infografico-taxa-de-aprovacao-cartoes-de-credito/>. Citado 2 vezes nas páginas 18 e 43.

5 VINDI. *Adquirente, subadquirente e gateway: quais as diferenças?* 2021. Acessado em 25/04/2021. Disponível em: <https://blog.vindi.com.br/adquirente-subadquirente-e-gateway/>. Citado na página 22.

6 DOMINGOS, P. A few useful things to know about machine learning. *Communications of the acm*, v. 55, n. 10, p. 78–87, 2012. Citado 2 vezes nas páginas 23 e 30.

7 SATHYA, R.; ABRAHAM, A. Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence*, v. 2, n. 2, p. 34–38, 2013. Citado na página 24.

8 NG, A. Cs229 lecture notes. Stanford, v. 1, p. 1–3, 2000. Citado na página 26.

9 SU, J.; ZHANG, H. A fast decision tree learning algorithm. In: *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*. [S.l.]: AAAI Press, 2006. (AAAI'06), p. 500–505. ISBN 9781577352815. Citado na página 26.

10 BORDA, M. Statistical and informational model of an its. Spinger, p. 7–52, 2011. Citado na página 26.

11 QUINLAN, J. R. Induction of decision trees. Spinger, v. 1, n. 1, p. 81–106, 1986. Citado na página 26.

12 BREIMAN, L. Random forests. machine learning. Springer, v. 45, n. 1, p. 5–32, 2001. Citado na página 27.

13 RASCHKA, S. Naive bayes and text classification i. 2014. Citado na página 27.

14 SUGUNA N.; THANUSHKODI, K. An improved k-nearest neighbor classification using genetic algorithm. International Journal of Computer Science Issues, v. 7, n. 2, p. 18–21, 2010. Citado na página 28.

15 NG, A. Cs229 lecture notes. Stanford, v. 5, p. 13–19, 2000. Citado na página 29.

16 NARKHEDE, S. *Understanding Confusion Matrix*. 2018. Acessado em 25/04/2021. Disponível em: <https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>. Citado na página 32.

17 TOTVS. *Software as a Service: aprenda tudo sobre o modelo de negócio SaaS*. 2019. Acessado em 25/04/2021. Disponível em: <https://www.totvs.com/blog/negocios/software-as-a-service/>. Citado na página 35.

18 RD STATION. *RD Station*. 2021. Acessado em 25/04/2021. Disponível em: <https://www.rdstation.com/sobre-nos/>. Citado na página 35.

19 SALESFORCE. *The Basics of the Quote-to-Cash Process*. 2021. Acessado em 25/04/2021. Disponível em: <https://www.salesforce.com/products/cpq/resources/quote-to-cash-process-basics/>. Citado na página 36.

20 RECURLY. *Retry Logic*. 2021. Acessado em 25/04/2021. Disponível em: <https://docs.recurly.com/docs/retry-logic>. Citado na página 37.

21 JUNO. *O que é e como funciona a captura e autorização de uma compra no cartão de crédito?* 2021. Acessado em 25/04/2021. Disponível em: <https://blog.juno.com.br/o-que-e-e-como-funciona-a-captura-e-autorizacao-de-uma-compra-no-cartao-de-credito/>. Citado na página 40.

22 MISSIO, F.; JACOBI, L. F. Variáveis dummy: especificações de modelos com parâmetros variáveis. Ciência e Natura, v. 29, p. 111–135, 2007. Citado na página 48.

23 PANDAS. *get_dummies*. 2021. Acessado em 25/04/2021. Disponível em: <https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html>. Citado na página 48.

24 IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: BACH, F.; BLEI, D. (Ed.). *Proceedings of the 32nd International Conference on Machine Learning*. Lille, France: PMLR, 2015. (Proceedings of Machine Learning Research, v. 37), p. 448–456. Disponível em: <http://proceedings.mlr.press/v37/ioffe15.html>. Citado na página 48.

25 SCIKIT-LEARN. *StandardScaler*. 2021. Acessado em 25/04/2021. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>. Citado na página 48.

26 RECURLY. *Predicting Recurring Transaction Success*. 2021. Acessado em 25/04/2021. Disponível em: <https://recurly.com/blog/predicting-recurring-transaction-success/>. Citado 2 vezes nas páginas 48 e 54.

27 KRAWCZYK, B. Learning from imbalanced data: open challenges and future directions. Prog Artif Intell, v. 5, p. 221–232, 2016. Citado na página 52.

28 SILIPO, R. *Machine Learning Algorithms and The Art of Hyperparameter Selection*. 2020. Acessado em 25/04/2021. Disponível em: <https://towardsdatascience.com/machine-learning-algorithms-and-the-art-of-hyperparameter-selection-279d3b04c281>. Citado na página 54.

29 IPPOLITO, P. P. *Hyperparameters Optimization*. 2019. Acessado em 25/04/2021. Disponível em: <https://towardsdatascience.com/hyperparameters-optimization-526348bb8e2d>. Citado na página 55.

30 SCIKIT-LEARN. *RandomizedSearchCV*. 2021. Acessado em 25/04/2021. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html>. Citado na página 55.

31 SCIKIT-LEARN. *RandomForestClassifier*. 2021. Acessado em 25/04/2021. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>. Citado na página 57.

32 BEN-DAVID, S.; KUSHILEVITZ, E.; MANSOUR, Y. Online learning versus offline learning. Springer, v. 29, p. 45–63, 1997. Citado 2 vezes nas páginas 59 e 64.

33 BCHUI, M. Artificial intelligence the next digital frontier? McKinsey and Company, v. 47, 2017. Citado na página 64.