Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Detecção de Falhas em um Aplicativo Móvel Bancário

Daniel Ferreira Schulz

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientadora
Prof. Dr. Aletéia Patrícia Favacho de Araújo

Brasília
2023

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

Fd

Ferreira Schulz, Daniel
Detecção de Falhas em um Aplicativo Móvel Bancário / Daniel Ferreira Schulz; orientador Aleteia Patricia Favacho De Araujo Von Paumgartten. -- Brasília, .
p.

Dissertação(Mestrado Profissional em Computação Aplicada) -- Universidade de Brasília, .

1. Detecção de Falhas. 2. Monitoramento de Serviços de TI. 3. Mineração de Dados. 4. Aplicações Webs. I. Patricia Favacho De Araujo Von Paumgartten, Aleteia, orient. II. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Detecção de Falhas em um Aplicativo Móvel Bancário

Daniel Ferreira Schulz

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Prof. Dr. Aletéia Patrícia Favacho de Araújo (Orientadora)
CIC/UnB

Prof. Dr. Alan Demétrius Baria Valejo
Universidade Federal de São Carlos

Prof. Dr. Thiago de Paulo Faleiros
CIC/UnB

Prof. Dr. Gladston Luiz da Silva
Coordenador do Programa de Pós-graduação em Computação Aplicada

Brasília, 10 de Novembro de 2023

Dedicatória

Dedico este trabalho com imensa gratidão. À minha família, cujo apoio incondicional e constante incentivo sustentaram-me ao longo desta desafiadora jornada. À minha orientadora, Aletéia Patrícia Favacho De Araújo, cuja orientação perspicaz e inspiradora me impulsionou a atingir o meu máximo potencial. Aos dedicados professores e aos colegas do grupo de estudo de infraestrutura, cujas ideias e discussões enriqueceram cada etapa deste trabalho. À minha querida namorada, cujo estímulo inicial me trouxe até aqui e cujo apoio incansável me guiou durante todo o percurso. Cada um de vocês teve um papel crucial nesta conquista e sou profundamente grato por isso.

Agradecimentos

Gostaria de expressar minha sincera gratidão a todas as pessoas e entidades que desempenharam um papel fundamental na realização deste trabalho.

Primeiramente, à minha amada família, cujo apoio e encorajamento constante foram a base que sustentou minha jornada acadêmica. A vocês, meus pais, avós, irmãos e demais familiares, devo a motivação e a força para seguir em frente, mesmo nos momentos mais desafiadores.

À minha orientadora, Aletéia Patrícia Favacho De Araújo, quero expressar minha sincera gratidão. Sua orientação perspicaz, comprometimento inabalável e vasto conhecimento contribuíram significativamente para o enriquecimento deste trabalho. Cada correção em caneta vermelha, feita com dedicação ao longo das inúmeras revisões, foi fundamental para que eu pudesse alcançar esta etapa com sucesso

Aos dedicados professores e colegas do grupo de estudo de infraestrutura, minha admiração e agradecimento. As discussões e ideias compartilhadas ao longo do trabalho foram fundamentais para o desenvolvimento deste estudo. Cada um de vocês contribuiu de maneira significativa para o meu crescimento acadêmico e profissional.

À minha querida namorada, que não apenas me incentivou a me matricular no mestrado, mas também me apoiou incansavelmente durante toda a jornada. Você foi muito mais do que uma parceira de estudos; você foi uma referência constante em minha vida.

Agradeço aos meus superiores e colegas de trabalho por acreditarem em mim e no compromisso que assumi ao embarcar nesta jornada de dois anos. Seu apoio e confiança foram fundamentais para que eu pudesse conciliar minhas responsabilidades profissionais com os rigores do mestrado.

Por fim, a todos os amigos, colegas e instituições que contribuíram direta ou indiretamente para esta conquista, o meu mais profundo agradecimento. Seu apoio e encorajamento fizeram toda a diferença.

Cada um de vocês desempenhou um papel significativo na minha trajetória acadêmica e na realização deste trabalho. Sem sua presença e apoio, esta conquista não teria sido possível. Muito obrigado a todos.

Resumo

O avanço da Internet revolucionou a maneira como as empresas prestam serviços aos seus clientes. No entanto, a proliferação de pontos de acesso digital também introduziu desafios, já que interrupções nos Sistemas de Informação podem causar danos significativos. Para estabelecer um ambiente operacional e estável, a monitorização contínua dos serviços, conforme preconizado pelo ITIL, tornou-se uma estratégia fundamental. Com base nessas considerações, este estudo propõe uma abordagem inovadora para detectar falhas usando técnicas de mineração de dados, empregando o modelo de referência CRISP-DM. A abordagem envolve a avaliação em tempo real de dados obtidos de uma ferramenta de *Web Analytics* para identificar rapidamente falhas críticas em uma aplicação web. Para aprimorar a precisão e a eficácia do processo, várias técnicas de pré-processamento e hiperparâmetros foram otimizados utilizando a biblioteca *Hyperopt*. Além disso, foram desenvolvidos sete modelos de classificação binária para categorizar e identificar falhas. Entre esses modelos, o modelo LSTM demonstrou o melhor desempenho, alcançando uma pontuação ROC-AUC de 0,961 e um *F1-Score* de 0,827. Esses resultados validam a capacidade do modelo de discernir eficazmente entre casos de falha e sucesso.

Palavras-chave: Detecção de Falhas, Aplicativo Bancário, *Web Analytics*, CRISP-DM, Aprendizado de Máquina.

Abstract

The advancement of the internet has revolutionized how companies provide services to their customers. However, the proliferation of digital access points has also introduced challenges, as interruptions in Information Systems can cause significant damage. To establish a stable operational environment, continuous monitoring of services, as advocated by ITIL, has become a fundamental strategy. Based on these considerations, this study proposes an innovative approach to detect failures using data mining techniques, employing the CRISP-DM reference model. The approach involves real-time evaluation of data obtained from a Web Analytics tool to swiftly identify critical failures in a web application. To enhance the accuracy and effectiveness of the process, various preprocessing techniques and hyperparameters were optimized using the hyperopt library. Furthermore, seven binary classification models were developed to categorize and identify failures. Among these models, the LSTM model demonstrated the best performance, achieving an ROC-AUC score of 0.961 and an F1-Score of 0.827. These results validate the model's ability to effectively distinguish between failure and success cases.

Keywords: Failure Detection, Mobile Banking, Web Analytics, CRISP-DM, Machine Learning.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A expansão da Internet e o uso dos celulares inteligentes (*smartphones*) mudaram a forma como os bancos entregam seus serviços financeiros aos clientes [10]. Em 2022, no Brasil, o número de celulares inteligentes superou a quantidade de computadores e de habitantes [11]. Dessa forma, o *mobile banking* que é o uso de dispositivos móveis para se conectar a serviços bancários por meio de uma rede sem fio [12], em algumas instituições, é o principal canal de interação entre os usuários e a empresa.

Assim, em algumas instituições financeiras esse canal tem se tornado importante aliado na missão de aumentar a satisfação dos clientes. Uma grande quantidade de serviços pode ser prestada por meio de aplicativo móvel bancário, tais como a obtenção de extratos da conta corrente, contratação de empréstimos, transferência de valores entre contas, aplicações em investimentos, entre outros. Em novembro de 2020, o Banco Central do Brasil lançou um esquema de pagamento instantâneo denominado PIX, que tem tido um grande impacto no sistema financeiro, visto que sua quantidade de operações ultrapassou meios tradicionais de pagamentos como cartões de débito, cartões de crédito, boletos e transferências [13]. Com a ausência de taxas para pessoas físicas, possibilidade de efetuar transações 24h por dia e 7 dias por semana, o PIX foi projetado para aumentar a inclusão financeira e a popularização dos serviços bancários no Brasil.

Devido ao elevado números de acessos via meios digitais, interrupções em Sistemas de Informação (SI) no setor bancário, tem provocado grandes prejuízos às empresas, muitas vezes impedindo que novos negócios sejam realizados. Dessa forma, a imagem da empresa é afetada negativamente e a satisfação do cliente é reduzida. Assim, a implementação de estratégias de tolerância a falha se faz necessária para mitigar possíveis interrupções nos serviços. Porém, inevitavelmente falhas em sistemas ocorrem e devem ser identificadas no menor tempo possível, para que as equipes de suporte possam corrigir e restabelecer os serviços.

Dessa forma, a monitoração contínua é uma estratégia para acompanhar, em tempo

real, o estado dos serviços de TI prestados por uma organização. A monitoração contínua pode ser aplicada por meio de métricas extraídas dos SIs, tais como tráfego de usuários, tempo de resposta das requisições, porcentagem de erros apresentados aos clientes e saturação da infraestrutura que atende um determinado serviço. No entanto, identificar quais métricas estão mais correlacionadas a uma indisponibilidade, e definir o limite ideal em cada uma dessas métricas, para que a monitoração possua a maior acurácia possível, pode ser um desafio para os engenheiros de confiabilidade.

Assim, o uso de técnicas de mineração de dados se apresenta como uma solução para os desafios encontrados, devido a automação de processos de filtragem de métricas, ajustes de limites baseados em probabilidades, e construção de uma informação única de saída baseada em diversas métricas de entrada. Diante do exposto, este trabalho propõe o uso de técnicas de Aprendizado de Máquina para detectar falhas em Sistemas de Informação, em tempo real, por meio das interações dos usuários.

1.1 Objetivos

O objetivo geral deste trabalho é propor uma abordagem de detecção de falhas por meio das interações dos usuários com um aplicativo móvel bancário. Para atingir este objetivo geral, faz-se necessário cumprir os seguintes objetivos específicos:

- Modelar um conjunto de dados rotulados das interações dos usuários com a ferramenta de *Web Analytics*, e da indisponibilidade do aplicativo móvel;
- Avaliar diferentes técnicas de engenharia de atributos e hiperparâmetros;
- Analisar diferentes algoritmos na tarefa de classificar as interações dos usuários, de acordo com a disponibilidade do serviço;
- Gerar um classificador binário da disponibilidade do serviço.

1.2 Contribuições

Nesta seção são apresentadas as principais contribuições resultantes deste trabalho de dissertação. As contribuições abrangem diversas áreas, desde o desenvolvimento prático até a disseminação do conhecimento científico por meio de publicações.

Uma das contribuições deste trabalho é a concepção e a implementação bem-sucedida da proposta deste trabalho em modelo de programa de computador. O programa desenvolvido, AppFailureDetector, oferece diversas funcionalidades para tratar com um conjunto de dados de *Web Analytics*, treinar diferentes modelos e otimizar seus resultados.

Esta ferramenta representa uma solução prática para criar modelos de monitoração, e foi projetada levando em consideração as necessidades de detecção de indisponibilidades. O programa pode ser acessado por meio do *link*: `https://github.com/schulzdanielf/AppFailureDetector`.

Além do desenvolvimento do software, AppFailureDetector, esta dissertação conduziu uma análise experimental abrangente de algoritmos associados a classificadores binários. Os experimentos realizados proporcionaram percepções sobre o desempenho e a eficácia de diferentes abordagens, estabelecendo uma base sólida para pesquisas futuras e otimizações.

Para enriquecer a comunidade acadêmica e prática, uma base de dados rotulada, denominada *Mobile Web Analytics*, foi criada e disponibilizada como parte deste trabalho. Esta base de dados contém a ocorrência de páginas acessadas e mensagens apresentadas aos usuários, e está disponível publicamente para incentivar pesquisas futuras e promover a replicabilidade dos resultados obtidos. Informações detalhadas sobre a base de dados estão disponíveis na Seção 4.3 e pelo *link*: `https://ieee-dataport.org/documents/mobile-web-analytics`.

Este trabalho resultou na publicação de dois artigos científicos em conferências de renome nacional. Os artigos, intitulados Detecção de Falhas em um Aplicativo Móvel Bancário e Detecção de Indisponibilidades em Aplicações Webs, foram submetidos e aceitos nas conferências Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos e Escola Regional de Alto Desempenho do Centro-Oeste, respectivamente. Essas publicações destacam os principais resultados e conclusões alcançados ao longo deste estudo, fornecendo uma contribuição significativa para a literatura acadêmica.

1.3 Estrutura deste Trabalho

Este trabalho segue uma estrutura organizacional que compreende um total de seis capítulos. Assim, além deste capítulo introdutório, no Capítulo 2 são apresentados os fundamentos teóricos abrangendo tópicos cruciais, tais como Aprendizado de Máquina, Gestão de Serviços de Tecnologia da Informação e as técnicas relevantes empregadas. O Capítulo 3 se concentra na exploração detalhada do aprendizado supervisionado e nos algoritmos de classificação subjacentes, além de uma revisão abrangente da literatura, contextualizando o trabalho e destacando estudos relacionados. O Capítulo 4 descreve a proposta deste trabalho em profundidade, delineando a arquitetura, a coleta de dados e as etapas adotadas para alcançar os resultados almejados. No Capítulo 5 são minuciosamente apresentados os resultados obtidos, bem como a implantação do modelo desenvolvido. Finalmente,

o Capítulo 6 encerra o trabalho, trazendo as considerações finais e delineando possíveis direções para futuras pesquisas.

Capítulo 2

Fundamentação Teórica

Este capítulo tem como objetivo introduzir os conceitos fundamentais que embasam o trabalho proposto nesta dissertação. Inicialmente, na Seção 2.1 é apresentada a área de Aprendizado de Máquina, classificando os diferentes tipos de aprendizado. A Seção 2.2 aborda as técnicas e as ferramentas específicas empregadas no processo de mineração de dados. Em seguida, na Seção 2.3 é apresentado o conceito de Gestão de Serviços de Tecnologia da Informação (T.I.), com ênfase no *framework* ITIL, além de uma definição detalhada sobre o que é considerado um incidente de TI. Por fim, na Seção 2.4, faz-se uma revisão do conteúdo abordado neste capítulo e uma introdução ao próximo capítulo.

2.1 Aprendizado de Máquina

O Aprendizado de Máquina é um subcampo da Ciência da Computação e Estatística, que envolve o melhoramento automático dos resultados de um programa ao longo do tempo, além de inferência e tomada de decisão sob incerteza [14]. O aprendizado também pode ser descrito como uma tarefa de busca em um grande espaço de hipóteses previamente definidas, na qual o objetivo da busca é encontrar a hipótese que melhor se encaixa nos exemplos de treinamento [15]. A melhora nos resultados é obtida aplicando algoritmos que iterativamente aprendem a partir de um conjunto de dados, padrões, sem serem explicitamente programados [16].

O Aprendizado de Máquina pode ser classificado em três tipos, os quais são: Aprendizado Supervisionado, Aprendizado Não-Supervisionado e Aprendizado por Reforço. A seguir será descrito o Aprendizado Não Supervisionado, o Aprendizado Supervisionado, devido a sua relevância para este trabalho será abordado em mais detalhes no capítulo 3. O Aprendizado por Reforço é uma derivação do Aprendizado Supervisionado, no qual um agente deve interagir com um ambiente dinâmico [17]. Assim, caso as suas interações sejam positivas, o ambiente retorna uma recompensa, o que reforça o seu comportamento.

Já as tentativas mal sucedidas podem não trazer retornos, ou até penalidades ao programa [18].

2.1.1 Aprendizado Supervisionado

O Aprendizado Supervisionado é o conjunto de técnicas que constroem modelos preditivos que aprendem a partir de um grande conjunto de exemplos, no qual cada exemplo tem sua classe de saída verdadeira [19]. Dessa forma, para cada exemplo de dados, o objetivo é utilizar as variáveis de entrada para prever os resultados da saída. A principal tarefa do Aprendizado Supervisionado é construir um estimador que seja capaz prever uma saída, dado um conjunto de variáveis de entrada [20].

Ao longo do processo, esse estimador é treinado, recebendo variáveis de entradas e de saída, no qual o aprendizado é feito comparando a saída do estimador com a saída correta do conjunto de dados. As diferenças ou erros, entre as saídas, são usadas para alterar o estimador a cada iteração com os dados, ajustando as saídas para que tenham o menor erro possível [21].

O Aprendizado Supervisionado pode ser utilizado para resolver problemas de classificação ou problemas de regressão. A classificação será apresentada em detalhes no capítulo seguinte. A regressão é aplicada quando as variáveis de saída de um determinado problema são numéricas e contínuas. Assim, dado um conjunto de variáveis de entrada, o objetivo do algoritmo de regressão é prever o valor da saída, sendo este o mais próximo possível do seu valor real. Como exemplo é possível citar uma regressão para calcular o preço médio de uma casa, visto as variáveis de entrada como bairro, tamanho da casa em metros quadrados e número de cômodos. Alguns exemplos de algoritmos que fazem o uso da regressão são a regressão logística [22] e o Lasso [23]. Alguns algoritmos podem ser implementados tanto para problemas de regressão quanto de classificação, são estes a Máquina de Vetores de Suporte [24], o K-Vizinho mais próximo [25], as Árvores de decisão [26], o Bayes ingênuo **[27]** e a Floresta Aleatória [28].

2.1.2 Aprendizado Não-Supervisionado

Ao trabalhar com dados nem sempre é possível obter os rótulos ou as variáveis de saída de um determinado processo, sendo o Aprendizado Não-Supervisionado aquele que utiliza apenas as variáveis de entrada para gerar conhecimento [29]. Algoritmos não-supervisionados possuem o objetivo de gerar representações dos dados, sem receber qualquer retorno de uma variável de saída, que podem ser usadas na tomada de decisão, previsão de entradas futuras, entre outros [30].

O Aprendizado Não-Supervisionado também pode ser utilizado para resolver problemas de agrupamento e de redução de dimensionalidade. O agrupamento pode ser definido como uma tarefa na qual o seu objetivo é dividir um conjunto de dados não rotulados em grupos finitos e discretos em estruturas naturais, em vez de fornecer uma caracterização precisa dos dados a partir de uma mesma probabilidade de distribuição [31]. Um exemplo de algoritmo de agrupamento é o *K-Means* [32]. Na Figura 2.1 é exemplificada uma tarefa de agrupamento, na qual os dados são divididos em 6 classes distintas, de acordo com as características das suas variáveis de entrada, representadas pela posição espacial.

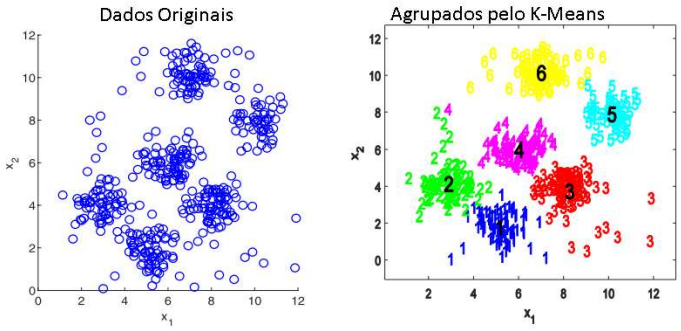

Figura 2.1: Algortimo de Agrupamento (Fonte: [33]).

A redução de dimensionalidade pode ser definida como a transformação de um conjunto de dados de dimensões D, em outro conjunto de dados de dimensão d, onde d $<$ D, e sua geometria é preservada o máximo possível [34], de forma que os dados ainda preservem informações importantes, mesmo ocupando um espaço menor. A *Principal Component Analysis* (PCA) [35] é uma técnica de redução de dimensionalidade popularmente utilizada, que transforma os dados para um novo sistema de coordenadas.

2.2 Técnicas e Ferramentas Utilizadas

Nesta seção, são explorados um conjunto de técnicas e ferramentas fundamentais no campo da ciência de dados e aprendizado de máquina. É apresentada a metodologia do CRISP-DM, uma estrutura amplamente utilizada para a condução de projetos de mineração de dados. Além disso, são investigadas a coleta de dados em uma ferramenta de *Web Analytics*, assim como técnicas de seleção de atributos e padronização, que desempenham papéis fundamentais na preparação e no processamento de dados. Para lidar com desequilíbrios nos dados, é discutido o uso da técnica *SMOTE*, enquanto é realizado o gerenciamento de experimentos e é feita a otimização de hiperparâmetros utilizando o MLflow e o Hyperopt, respectivamente. Por fim, a seção aborda conceitos necessários para a interpretabilidade

de modelos, utilizando a técnica SHAP para oferecer entendimento sobre as decisões dos algoritmos de aprendizado de máquina. A análise destas técnicas e ferramentas proporciona uma base para o desenvolvimento e aprimoramento dos conceitos adotados neste trabalho.

2.2.1 CRISP-DM

Em um mundo impulsionado pela tecnologia, a quantidade de dados disponível está crescendo exponencialmente. Nesse cenário, a mineração de dados desempenha um papel fundamental no entendimento e processamento desses dados. Porém, para obter resultados impactantes, é essencial abordar projetos de mineração de dados de maneira estruturada e seguindo as melhores práticas da área. O CRISP-DM (Cross Industry Standard Process for Data Mining) [36] é um modelo amplamente utilizado que fornece orientações e processos para que o projeto seja bem-sucedido.

O CRISP-DM é estruturado de forma hierárquica, sendo formado por um conjunto de tarefas em quatro níveis que variam do mais genérico ao mais específico. O primeiro nível é o mais genérico e é formado pela fases do processo. A Figura 2.2 apresenta as fases do CRISP-DM e o seus relacionamentos, no qual o processo se inicia pelo entendimento do negócio e se encerra na implantação. O segundo nível é formado por tarefas genéricas, enquanto o terceiro é formado por tarefas específicas. Por fim, o último nível é formado pelo próprio processo.

Além dos níveis de tarefas, horizontalmente o CRISP-DM é separado pelo modelo de referência e pelo guia de uso. O modelo de referência apresenta em resumo os procedimentos de cada fase, tarefas e saídas. Por outro lado, o guia de uso oferece sugestões para cada fase e tarefa, além de detalhar como executar um mini projeto [36].

2.2.2 Web Analytics

Os clientes são um ponto chave para o sucesso de um serviço na Internet. Logo, é preciso entender como esses clientes interagem com o sistema e, consequentemente, extrair métricas baseadas nesses clientes. Tal necessidade permitiu o desenvolvimento de ferramentas especializadas na extração de métricas em páginas na *web*, denominadas de *Web Analytics* [37]. Estas ferramentas não capturam apenas a quantidade de visualizações nas páginas, mas medem o tempo de um clique para outro, a localização do *mouse* de um computador, a mensagem que o cliente recebeu do sistema ao realizar uma ação, e toda a jornada de um cliente desde o momento que ele acessa a página inicial até o encerramento da aplicação.

O projeto Matomo [38] é uma ferramenta de *Web Analytics* amplamente utilizada, e uma solução de código aberto que tem sido relatada como uma alternativa a soluções

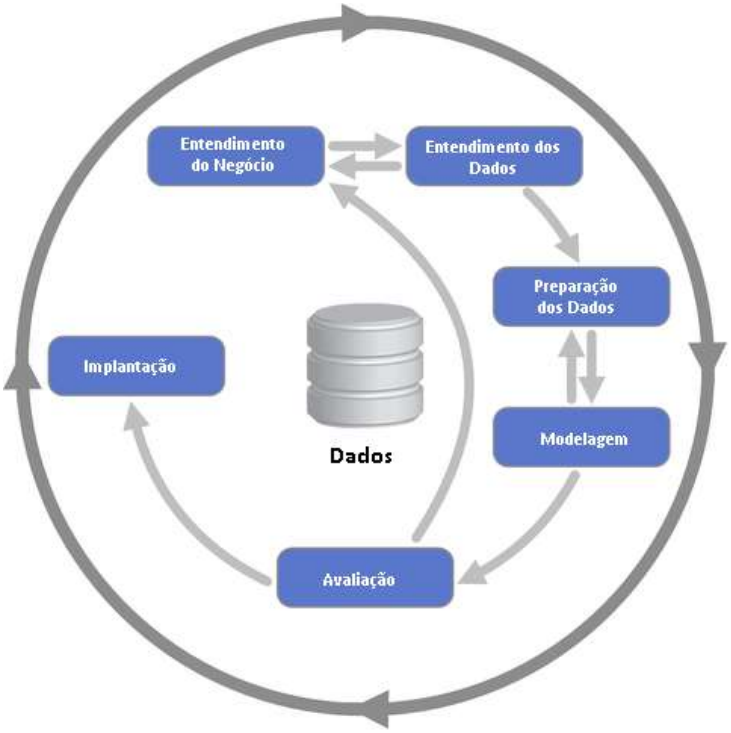

Figura 2.2: Fases do Modelo de Referência CRISP-DM (Fonte: [36]).

proprietárias. O Matomo é implementado via um pequeno trecho de código JavaScript que está incluído em cada página *web*, e pode ser personalizado para selecionar o que será capturado ou não [39]. Uma das vantagens do Matomo é que os dados gerados são totalmente da organização que escolhe utilizá-lo, garantindo, assim, a privacidade e a proteção dos dados, os quais são fundamentais no sistema financeiro.

2.2.3 Seleção de Atributos

O volume de dados gerado pelos Sistemas de Informação tem experimentado um crescimento significativo nos últimos anos. No entanto, os algoritmos de aprendizado de máquina enfrentam desafios ao lidar com grandes quantidades de atributos de entrada. Portanto, torna-se imprescindível identificar os dados relevantes para o problema a ser resolvido, por meio de um processo de seleção de atributos. Essa etapa é fundamental na mineração de dados, pois visa identificar os atributos mais importantes e eliminar informações redundantes, irrelevantes e ruidosas [40].

A seleção de atributos oferece diversos benefícios, notadamente aprimorando o desempenho dos algoritmos, reduzindo o tempo de processamento e melhorando a interpretabilidade dos resultados. Em situações específicas, a abundância de atributos no conjunto

de dados pode prejudicar o aprendizado dos algoritmos, tornando-se essencial eliminar aqueles que são indesejados [41].

Existem três principais estratégias de seleção de atributos: filtragem, embrulho e incorporação [42]. Neste trabalho, optou-se por utilizar a estratégia de filtragem devido à sua independência, ao passo que as outras estratégias são derivadas de algoritmos específicos de classificação. Outro benefício da filtragem é o seu menor tempo de processamento, visto que não requer o treinamento de um modelo de aprendizado. A abordagem de filtragem consiste em ranquear os atributos disponíveis e, em seguida, eliminar aqueles indesejáveis, tornando-a uma escolha apropriada para avaliar diversos algoritmos de maneira mais imparcial.

A filtragem pode utilizar diversas funções de ranqueamento baseadas em testes estatísticos, tais como correlação de Pearson [43], correlação de Spearman, distribuição Qui-quadrado, correlação tau de Kendall e a Análise de Variância (ANOVA) [44]. Devido aos dados de entrada serem numéricos e a saída do problema ser categórica, optou-se por utilizar a ANOVA para ranquear os atributos. A ANOVA possibilita a comparação da variação entre grupos e dentro dos grupos, permitindo medir a existência de diferenças entre as médias dos grupos e identificar os atributos mais relevantes para diferenciar as classes do problema.

2.2.4 Padronização

Os dados coletados dos Sistemas de Informação podem não estar em um formato ideal para serem utilizados por algoritmos de aprendizado de máquina [45]. Uma técnica de pré-processamento de dados amplamente utilizada na mineração de dados é o ajuste de escala, que visa reorganizar os diferentes atributos na mesma ordem de grandeza. Essa abordagem proporciona um equilíbrio maior entre os atributos, eliminando diferenças de magnitude existentes e tornando os dados mais adequados para análise por tais algoritmos.

A padronização (*Z-score standardization*) é um processo de engenharia de atributos no qual é aplicada a remoção da média, e os dados são escalados de acordo com a variância de cada métrica. Essa técnica permite a construção de modelos mais acurados, e tem sido examinada em diversos algoritmos [25] [24] [32]. Muito autores validaram o seu impacto na melhoria de resultados [46] [47] [48]. Porém, apesar de apresentar uma melhoria em determinados algoritmos, a padronização pode também influenciar negativamente os resultados de outros algoritmos.

O processo de padronização é semelhante a normalização, visto que as duas técnicas têm o objetivo de transformar as variáveis na mesma ordem de grandeza. Na padronização a média é transformada para o valor 0, e um desvio padrão para o valor 1. A fórmula do algoritmo de padronização é apresentada na Equação 2.1, na qual z é o valor final da

amostra transformada, x é o valor da amostra antes da transformação, u é a média de todas as amostras de um determinado atributo, e s é o desvio padrão das amostras.

$$z = \frac{x - u}{s} \tag{2.1}$$

2.2.5 SMOTE

O desbalanceamento dos dados ocorre quando uma das classes dos rótulos possui uma representatividade menor em relação às outras classes. Essa situação é bastante comum em muitos conjuntos de dados, especialmente em problemas de detecção de eventos raros. O desbalanceamento pode levar alguns modelos a apresentarem viés em direção à classe majoritária, resultando em uma baixa capacidade de generalização para a classe minoritária e influenciando no desempenho do modelo.

A redução do desbalanceamento é geralmente realizada na etapa de pré-processamento dos dados, empregando diversas estratégias. Algumas delas incluem a repetição de valores da classe minoritária ou a remoção de valores da classe majoritária. Outra alternativa eficaz é a geração de amostras sintéticas utilizando a técnica SMOTE (Synthetic Minority Over-Sampling Technique) [49]. O SMOTE aborda o desbalanceamento criando exemplos sintéticos da classe minoritária, aumentando assim o número de instâncias dessa classe no conjunto de treinamento

O SMOTE gera amostras sintéticas operando no espaço de características, no qual são extraídas amostras da classe minoritária, e são geradas novas amostras a partir dos seus vizinhos mais próximos. O algoritmo calcula as diferenças entre as amostras e os seus vizinhos mais próximos, e multiplica essa diferença por uma valor aleatório entre 0 e 1, e adiciona esse valor ao vetor de atributos considerado [49]. Assim, essa abordagem se demonstra mais eficiente na criação de novas amostras quando comparada a outras estratégias de balanceamento.

A Figura 2.3 ilustra o funcionamento da técnica SMOTE no plano cartesiano. Nela, é possível visualizar os pontos azuis representando a classe majoritária e os pontos vermelhos representando a classe minoritária. As linhas representam as distâncias entre os pontos da classe minoritária. Após serem multiplicadas por um valor aleatório, é criada a amostra sintética, representada pelo ponto amarelo. Essa amostra sintética é uma combinação ponderada dos pontos vizinhos da classe minoritária, e é fundamental para aumentar a representatividade dessa classe no conjunto de treinamento.

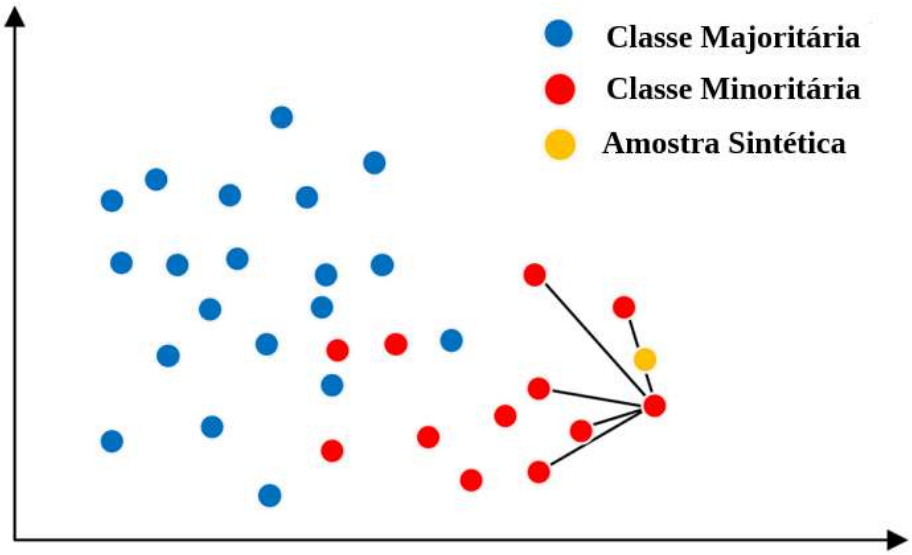

Figura 2.3: Funcionamento do SMOTE (Fonte: Adaptado de [1]).

2.2.6 MLFlow

Com o crescimento dos projetos de mineração de dados, o nível de complexidade também tem aumentado. Registrar os resultados em cada fase e acompanhar a evolução do modelo e dos dados pode ser um desafio. Nesse contexto, um conceito que tem ganhado força nos últimos anos é o de MLOps, que busca preencher essas lacunas e outras. O MLOps pode ser definido como um conjunto de boas práticas que envolvem a conceituação, implementação, monitoramento, implantação e escalabilidade dos produtos de aprendizado de máquina [50]. Essa abordagem visa integrar de forma eficiente o desenvolvimento de modelos de aprendizado de máquina com a operacionalização desses modelos em ambientes de produção.

Influenciado pelas práticas do DevOps [51], o MLOps traz entregas mais contínuas ao desenvolvimento de aplicações de mineração de dados. Com o MLOps, os projetos de mineração de dados podem obter benefícios significativos, como maior agilidade no ciclo de desenvolvimento, melhor controle de versões, maior rastreabilidade dos resultados e maior confiabilidade dos modelos em produção. Essa abordagem sinérgica entre desenvolvimento e operações permite uma integração mais fluida de todo o processo, garantindo maior eficiência e qualidade ao longo do caminho.

Existem diversas ferramentas que podem auxiliar na implementação do MLOps, e uma opção de código aberto muito popular é o MLFlow. Essa plataforma se fundamenta na filosofia de interface aberta, o que significa que os desenvolvedores têm a liberdade de incorporar seu próprio código e fluxos de projeto dentro da ferramenta [52]. Isso torna

o MLFlow altamente flexível e adaptável, permitindo uma integração perfeita com os processos existentes de desenvolvimento e mineração de dados.

O MLFlow é uma plataforma composta por três principais módulos, cada um desempenhando um papel fundamental no gerenciamento de experimentos e modelos de aprendizado de máquina. O primeiro módulo é o "Rastreamento", uma API que registra e monitora os dados dos experimentos, incluindo resultados, código-fonte, parâmetros utilizados e outras informações relevantes. Em seguida, tem-se o módulo "Projeto", que proporciona uma estrutura de empacotamento que facilita o reuso do código em diferentes contextos e experimentos. Por fim, o módulo "Modelo"oferece uma estrutura organizacional para gerenciar e implantar os modelos de aprendizado de máquina em diversas ferramentas e ambientes.

A Figura 2.4 mostra a interface gráfica do MLFlow, destacando a aba de experimentos onde as execuções são apresentadas junto com seus resultados e parâmetros. Uma característica notável do MLFlow é sua capacidade de integrar facilmente a base de dados dos resultados, permitindo que eles sejam utilizados por outras ferramentas ou incorporados diretamente ao código de desenvolvimento.

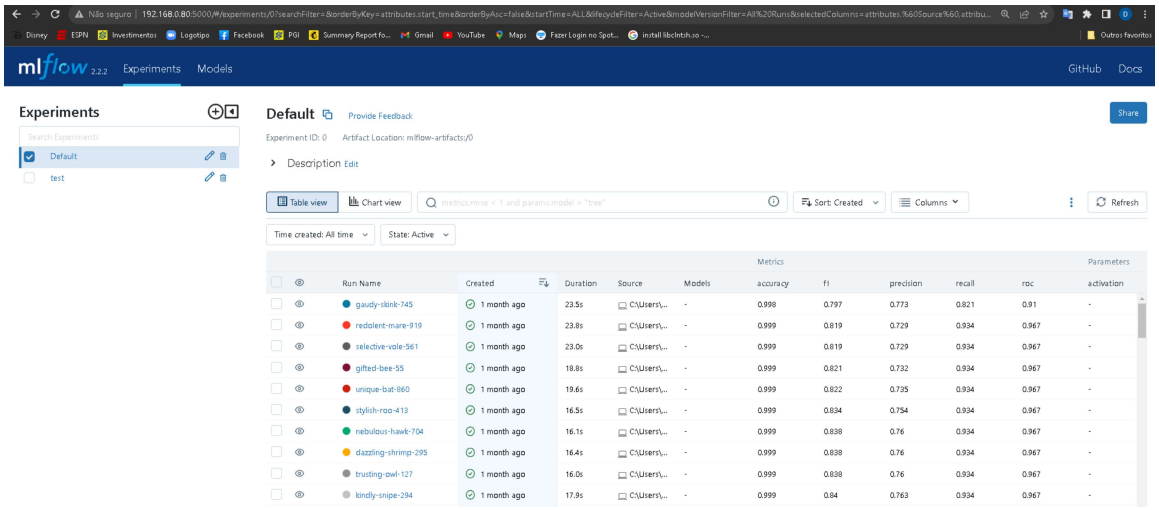

Figura 2.4: Interface Gráfica do MLFlow

2.2.7 HyperOpt

Uma das etapas cruciais da mineração de dados é o ajuste dos hiperparâmetros dos modelos. Essa atividade pode se tornar desafiadora em algumas situações, devido à vasta quantidade de opções disponíveis e/ou ao longo tempo de treinamento dos modelos [53]. Para contornar essa dificuldade, algumas ferramentas foram desenvolvidas com o objetivo de facilitar a otimização dos hiperparâmetros, buscando automaticamente o melhor con-

junto de configurações de acordo com um objetivo definido. Nesse contexto, o Hyperopt se destaca como uma opção de código aberto e eficiente.

O Hyperopt é uma biblioteca de otimização de hiperparâmetros desenvolvida para auxiliar a busca automática pelas melhores combinações de hiperparametros em modelos de aprendizado de máquina [2]. Essa ferramenta utiliza diferentes algoritmos de busca, como o algoritmo de árvore de decisão Parzen Estimator (TPE), para explorar de forma inteligente o espaço de busca e encontrar configuraçãoes que levem ao melhor desempenho do modelo. Com o Hyperopt é possível acelerar o processo de otimização dos hiperparametros e alcançar modelos mais eficientes e precisos. Além disso, a biblioteca é altamente configurável e pode ser integrada com outras aplicações de aprendizado de máquina, se tornando uma opção popular.

O Hyperopt opera a partir da definição de uma função objetivo e do espaço de busca. A função objetivo representa o objetivo final da otimização, que pode ser maximizar a acurácia ou minimizar o número de erros, por exemplo. Por sua vez, o espaço de busca consiste nos hiperparâmetros e suas combinações que serão testados pelo modelo durante a busca por um resultado ótimo. Dessa maneira, o Hyperopt explora as possíveis combinações dentro desse espaço, buscando identificar a configuração que proporcione o melhor valor para a função objetivo.

O método TPE utiliza densidades não paramétricas para modelar p(x|y), substituindo distribuições anteriores (como uniforme ou loguniforme) por uma variação de mistura gaussiana. Ao explorar o espaço de busca com diferentes observações nas densidades não paramétricas, o algoritmo busca encontrar os melhores parâmetros [54]. Isso o torna uma excelente opção para otimizar diferentes tipos de hiperparâmetros, sejam eles contínuos, discretos, categóricos ou ordinais.

A Figura 2.5 demonstra o funcionamento do Hyperopt, onde um modelo é criado através da otimização das etapas de pré-processamento e de um classificador. Nela, é possível visualizar as opções que estão sendo exploradas em um dado momento, destacadas pela cor verde, no qual o PCA e o KNN foram selecionados. Dentro do PCA, também é possível identificar dois hiperparâmetros: o número de componentes (discreto) e o branqueamento (categórico). Essa representação visual permite uma compreensão mais clara do processo de busca pelo melhor conjunto de hiperparâmetros para o modelo.

A escolha do Hyperopt foi fundamentada em alguns fatores, dentre eles sua fácil integração com diferentes bibliotecas de implementação dos algoritmos. Além disso, a capacidade do Hyperopt de adaptar dinamicamente sua busca com base nos resultados anteriores, utilizando métodos de modelagem probabilística, permite uma exploração mais eficiente do espaço de hiperparâmetros. Por fim, o Hyperopt permite lidar com diferentes tipos de hiperparâmetros, incluindo contínuos, discretos e condicionais, também contribui

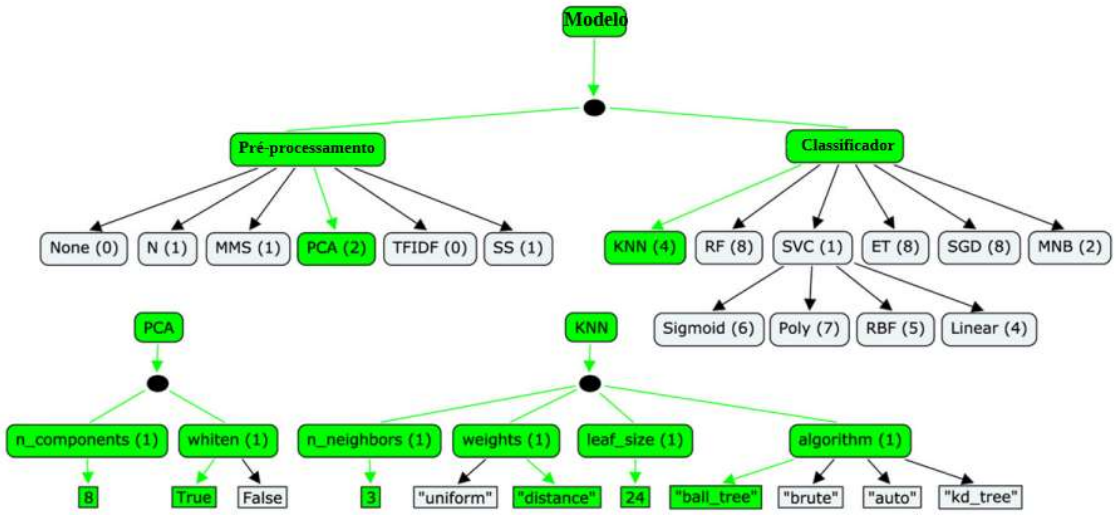

Figura 2.5: Hyperopt: Explorando o Espaço de Busca de Hiperparâmetros (Fonte: Adaptado de [2]).

para cenários onde a diversidade de algoritmos e configurações é essencial.

2.2.8 SHAP

O entendimento dos modelos de Aprendizado de Máquina é de suma importância para sua aplicação em diversos cenários. Por isso, a interpretabilidade desses modelos torna-se fundamental para garantir a confiança e aceitação de suas previsões. No entanto, muitos modelos de aprendizado de máquina são considerados caixas-pretas, ou seja, não fornecem de forma intrínseca o motivo por trás de suas decisões, o que dificulta sua compreensão. Diante desse desafio, uma alternativa é o SHAP (SHapley Additive exPlanations), uma abordagem que oferece explicações locais para as previsões dos modelos.

O SHAP auxilia na interpretabilidade do modelo através do calculo da importância dos atributos, quando estão correlacionados entre si. Isso é feito através de treinamentos repetidos do modelo, incluindo e excluindo cada atributo individualmente. Em seguida, é possível comparar todas as combinações possíveis nas previsões do modelo com e sem a presença de cada atributo. Com esse método, é possível medir a contribuição específica de cada característica para as previsões do modelo, proporcionando uma compreensão mais clara e detalhada das decisões tomadas pelo modelo [3].

O SHAP tem diversas aplicações, e um exemplo é a sua utilização para identificar quais pontos em uma imagem exercem maior influência na determinação de uma classe. Na Figura 2.6, é exibida a saída de uma Rede Neural Convolucional treinada com o conjunto de dados MNIST. O primeiro quadrado representa um exemplo de entrada do dígito 8, enquanto o segundo apresenta os pontos da imagem que influenciam positivamente o

resultado da classe 8 (representados pelos pontos vermelhos) e os pontos que influenciam negativamente (representados pelos pontos azuis). No último quadrante, são apresentados os valores SHAP para outra classe, que é a classe 3, onde é possível visualizar um agrupamento de pontos azuis à esquerda da imagem, que os diferencia do numeral 8. Essa visualização detalhada permite uma melhor compreensão do modelo e ajuda a identificar os fatores mais relevantes na tomada de decisão.

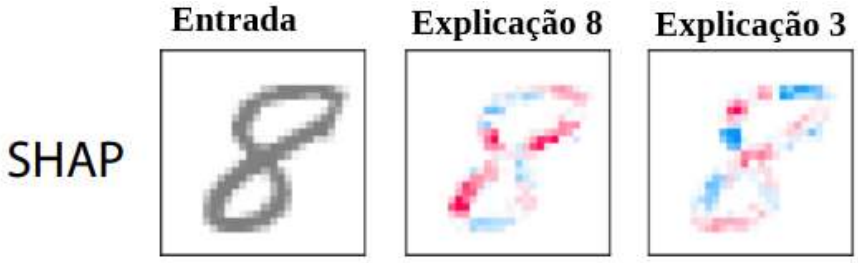

Figura 2.6: Explicação de dados do MNIST. (Fonte: Adaptado de [3])

2.3 Gestão de Serviços de Tecnologia da Informação

O Gerenciamento de Serviços de Tecnologia da Informação, do inglês *Information Technology Service Management (ITSM)*, é uma subdisciplina da Ciência de Serviços que foca na operação de TI, provendo uma estrutura para alinhar as atividades relacionadas a operação, entre os usuários e a equipe técnica [55]. Assim, existem diversos *frameworks* para auxiliar as empresas no gerenciamento de serviços, e um dos mais utilizados no contexto de infraestrutura é o ITIL [56]. Ele foi criado no final da década de 1980 pelo governo britânico, como um esforço para disciplinar e permitir a comparação entre as propostas dos diferentes prestadores de serviços.

O ITIL pode ser definido como um conjunto de boas práticas com a intenção de entregar serviços de alta qualidade a um custo justificável, sendo construído para controlar e gerenciar a operação de TI, incluindo o melhoramento contínuo e as métricas [55]. Assim, o ITIL objetivou garantir um mínimo de padronização de atendimento em termos de processos, desempenho, terminologia, qualidade e custo [57].

Inicialmente, o ITIL era composto por aproximadamente 40 livros, e ao longo dos anos foi sendo revisado até a versão atual, a qual é a quarta. Em sua quarta versão, os principais componentes do ITIL são o Sistema de Valor de Serviço (SVS) e o modelo de quatro dimensões. O SVS simboliza como diversas atividades e componentes de uma organização podem trabalhar conjuntamente na criação de valor por meio de serviços de TI. Já o modelo de quatro dimensões garante uma visão holística no que tange o gerenciamento de serviços, sendo elas [58]:

- Organização e pessoas;
- Informação e tecnologia;
- Parceiros e fornecedores;
- Fluxos de valor e processos.

A primeira dimensão diz respeito a como a organização está estruturada e sua cultura alinhada aos seus objetivos estratégicos, necessários para suportar o modelo de operação. A dimensão de tecnologia e informação abrange o conhecimento necessário para o gerenciamento dos serviços e suas tecnologias. A dimensão de parceiros e fornecedores inclui o relacionamento da empresa com outras organizações que estão ligadas ao processo de gestão de serviços de TI. A última dimensão engloba como diferentes partes da organização trabalham em conjunto para criar valor por meio dos produtos e serviços prestados [58].

O SVS do ITIL possui 14 práticas de gestão gerais, 17 práticas de gestão de serviços, e 3 práticas de gestão técnica. Todas essas práticas estão sujeitas as quatro dimensões do gerenciamento de serviço [58]. Duas práticas de gestão de serviço, a gestão de incidentes, e a gestão da monitoração são essenciais para o entendimento deste trabalho, e serão detalhadas nas próximas seções.

2.3.1 Gestão de Incidentes

Um incidente pode ser definido como uma degradação na qualidade de um serviço, a interrupção total deste ou mesmo a falha em um de seus componentes, ainda que não tenha impacto direto no serviço [59]. O objetivo da sua gestão é minimizar o efeito ao negócio, restabelecendo o serviço o mais rápido possível, e garantindo o Acordo de Nível de Serviço (do inglês *Service Level Agreement* - SLA) além da satisfação daqueles que o consomem [58].

O ciclo de vida de um incidente se inicia em sua abertura, que é realizada via chamado de um usuário ou identificado por um sistema de monitoração [60]. Após sua detecção, o incidente passa por equipes de suporte até ser aplicada uma solução, na qual são registrados os procedimentos que foram realizados para contorno do ocorrido, que podem ser utilizados no futuro, para auxiliar na condução de outros incidentes.

Segundo o ITIL, as organizações devem classificar os incidentes de acordo com o seu impacto, melhorando a alocação dos recursos envolvidos no processo. Incidentes com menor prioridade precisam ser gerenciados de maneira eficiente para não consumir recursos de forma excessiva. Por outro lado, incidentes de alto impacto necessitam de maiores recursos e de um gerenciamento mais complexo. A gestão de incidentes possui um grande

impacto nos clientes e na sua percepção sobre o serviço prestado. Assim, o seu gerenciamento é fundamental para a entrega de valor no processo de gestão de serviços de TI [58].

2.3.2 Gestão da Monitoração e Eventos

Esta prática identifica e prioriza eventos relacionados aos serviços e os seus componentes, sistematicamente observando, registrando e reportando suas mudanças [58]. O seu propósito, além de reduzir o impacto ao negócio, é prevenir a ocorrência de incidentes. Assim, eventos podem ser classificados em informativos, avisos e exceções, de acordo com a sua significância e impacto ao serviço [58].

Os informativos não necessitam de uma ação imediata, porém, devem ser analisados no longo prazo para aperfeiçoamento do processo e implementação de técnicas pró-ativas. Os avisos permitem que ações sejam tomadas antes de uma degradação do serviço, como um alto consumo de recursos em determinado item de configuração. Por último, exceções precisam de uma ação imediata e algumas podem se transformar em incidentes [58].

Um dos pontos chaves para o sucesso da Gestão da Monitoração é a automação, na qual as ferramentas de detecção de eventos se integram com outras práticas da gestão de serviços de TI, agilizando seus processos. Tais práticas incluem a gestão de incidentes, gestão de problemas, gestão de mudança, entre outros, além de uma integração com os meios de comunicação da organização. Assim, monitorações automatizadas quando precisas, podem ser utilizadas de gatilhos para a abertura de um incidente automático, envio de mensagens aos responsáveis por uma determinada aplicação, execução de um procedimento de auto recuperação, migração de serviços, entre outros procedimentos.

2.4 Considerações Finais

Ao longo deste capítulo foi realizada uma apresentação da área de Aprendizado de Máquina. Inicialmente foi definido o conceito de Aprendizado de Máquina e suas divisões. O Aprendizado Não-Supervisionado foi descrito e foram introduzidas as tarefas de clusterização e redução de dimensionalidade. Alguns algoritmos como o K-Means [32] e o PCA [35] foram brevemente comentados. O Aprendizado Supervisionado foi introduzido e teve suas tarefas revisadas com detalhes. A regressão foi abordada, assim como, alguns dos seus principais algoritmos.

Ao longo deste trabalho, foram apresentadas várias técnicas e ferramentas utilizadas para aprimorar o processo de mineração de dados, incluindo o CRISP-DM, seleção de atributos, padronização, SMOTE, MLFlow, HyperOpt e o SHAP. Cada uma dessas técnicas e ferramentas tem o propósito de facilitar a implementação do processo e aperfeiçoar

o desempenho dos resultados. Com a utilização conjunta dessas abordagens, é possível obter melhores resultados na análise dos dados, contribuindo para o sucesso do projeto de mineração de dados como um todo.

Além disso, a Gestão de Serviços de TI, com destaque para o *framework* mais amplamente utilizado no mercado, o ITIL. Explicamos a importância da implementação das boas práticas do ITIL e como elas estão diretamente relacionadas à satisfação dos clientes. Para entender melhor o contexto do negócio abordado nesta dissertação, também foram especificadas a Gestão de Incidentes e a Gestão da Monitoração e Eventos, focando nos objetivos que essas práticas buscam atingir. Ao combinar essas abordagens, temos uma base sólida para a condução deste trabalho e a melhoria dos serviços de TI de forma geral.

No próximo capítulo os algoritmos de classificação utilizados ao longo do trabalho serão apresentados, assim como as métricas de avaliação adequadas para o contexto do problema.

Capítulo 3

Algoritmos de Classificação e Trabalhos Relacionados

Este capítulo tem como objetivo introduzir os algoritmos de classificação utilizados ao longo deste trabalho, além de apresentar os trabalhos relacionados. Inicialmente, na Seção 3.1, são apresentados os conceitos de algoritmos de classificação e os classificadores utilizados, desde os mais simples, como o Bayes ingênuo, até as complexas redes *Long Short-Term Memory*. Em seguida, na Seção 3.2, são apresentados os trabalhos correlatos. O Aprendizado de Máquina na Gestão de Serviços de TI é abordado, seguido pela detecção de falhas em registros de sistemas e em tarefas na nuvem. Por fim, na Seção 3.3, realiza-se uma revisão do conteúdo deste capítulo e uma introdução ao próximo capítulo.

3.1 Algoritmos de Classificação

Os algoritmos de classificação são utilizadas quando o problema possui variáveis de saídas categóricas [61], ou seja, quando suas saídas são divididas em um número finito de classes. Um dos tipos mais simples de classificação é a binária, na qual apenas duas classes são utilizadas na saída. Um exemplo de uso é na monitoração de serviços de infraestrutura, que pode ser classificada em normal e anormal.

A Figura 3.1 mostra um caso de classificação binária, no qual os pontos azuis representam uma classe, e os pontos amarelos representam outra classe. Assim, o modelo de classificação estabelece uma linha de divisão entre o limite de cada classe. Diversos algoritmos utilizam a classificação como técnica de aprendizagem, entre eles é possível citar a Máquina de suporte de vetores (SVM), do inglês *Support-Vector Machine* [24], a Árvore de Decisão [26], o Bayes ingênuo (NB), do inglês *Naive Bayes* [27], o K-Vizinho mais Próximo [25] e a Floresta Aleatória [28].

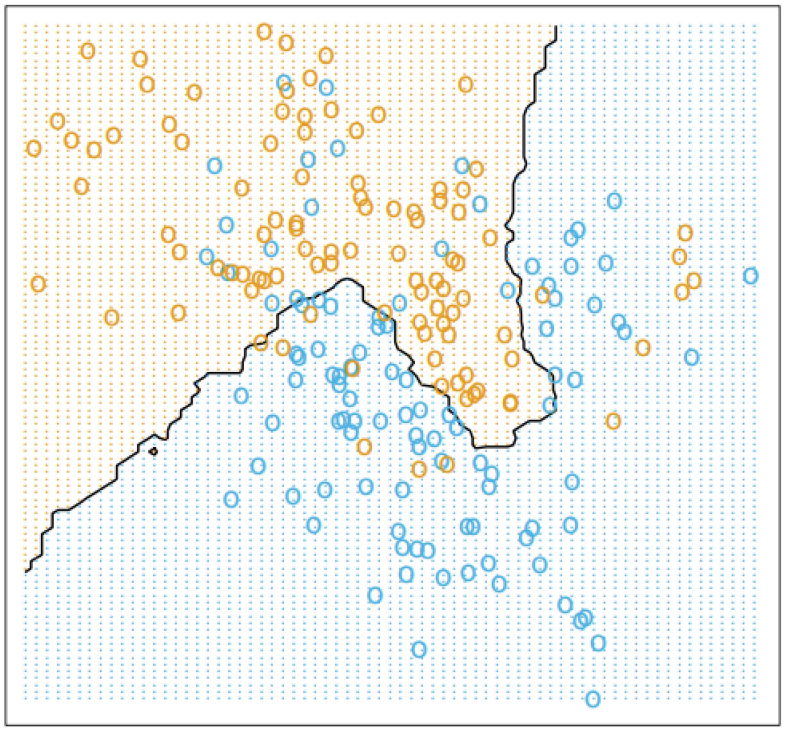

Figura 3.1: Representação de um algoritmo de classificação (Fonte: Adaptado de [4]).

3.1.1 Bayes Ingênuo

O classificador Bayes ingênuo é um algoritmo simples de aprendizado que se baseia no teorema de Bayes, e na forte suposição no qual os atributos utilizados são condicionalmente independentes [62]. Apesar da suposição não ser verdade em muitos casos, o resultado do classificador é notável e, muitas vezes competitivo, comparado a outros classificadores [27].

Assim, o Bayes ingênuo é um modelo probabilístico que atribui a cada possível saída uma certa probabilidade. As probabilidades são geradas a partir dos atributos de entrada, e ao final é escolhida a classe que possui a maior probabilidade. O modelo aprende a partir de uma amostra de dados, também chamado de conjunto de treinamento, e assim é estimada a probabilidade a posteriori P(Y|X) para cada classe de Y dado um X.

Na Equação 3.1 é apresentada a fórmula do Teorema de Bayes. Nela X e Y são eventos, P(X) é a probabilidade de um evento X, e P(Y) é a probabilidade de um evento Y. P(Y|X) é a probabilidade posteriori de Y condicional a X, e P(X|Y) é a probabilidade de posteriori de X condicional a Y.

$$P(Y|X) = \frac{P(Y)P(X|Y)}{P(X)} \tag{3.1}$$

3.1.2 K-Vizinho mais Próximo

O K-Vizinho mais Próximo (KNN, do inglês *K-Nearest Neighbor*) é um algoritmo de classificação não paramétrico, ou seja, ele não faz suposições sobre a amostra de dados [63]. É usado em circunstâncias no qual não se tem uma amostra representativa dos dados e, consequentemente, não se pode inferir a distribuição de suas probabilidades [64]. Este algoritmo foi desenvolvido inicialmente por Evelyn Fix e Joseph Hodges [65] e, posteriormente, melhorado por Thomas Cover [25].

O algoritmo funciona classificando as novas entradas a partir do conjunto de treinamento. Dessa forma, o novo dado não rotulado é inserido no plano amostral, e são calculadas as distâncias para os demais dados rotulados. A distância Euclideana normalmente é utilizada para medir quais classes estão mais próximas [63]. A Equação 3.2 calcula a distância Euclideana entre dois pontos, onde o vetor x = $(a_1, a_2, ..., a_n)$, n é o tamanho do vetor de entrada, a_r é o r^{o} elemento, e w_r é o peso do r^{o} elemento.

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} w_r(a_r(x_i) - a_r(x_j))^2} \tag{3.2}$$

O hiperparâmetro K é previamente definido para o valor de quantos vizinhos serão avaliados na comparação, sendo comum escolher um valor ímpar para evitar empates na decisão do resultado. O cálculo para determinar qual é a classe de uma nova entrada é apresentada na Equação 3.3, na qual d_i é uma entrada de exemplo, x_j é um dos k vizinhos mais próximos no conjunto de dados de treinamento, e $y(x_j, c_k)$ é quando x_j pertence a uma classe c_k. Assim, a classe que possuir a maioria dos vizinhos é a classe predita pelo classificador.

$$y(d_i) = argmax_k \sum_{x_j \epsilon kNN} y(x_j, c_k) \tag{3.3}$$

3.1.3 Máquina de Vetores de Suporte

A Máquina de Vetores de Suporte (SVMs, do inglês *Support Vector Machines*) é uma técnica de Aprendizado de Máquina que vem recebendo bastante atenção, e que seus resultados são comparáveis ou superiores a Redes Neurais [66]. Em sua essência, o SVM é um algoritmo que potencializa uma função matemática para um determinado conjunto de dados [5]. Essa técnica foi desenvolvida por Vapnik [67], e evoluída posteriormente em [68].

Para entender como funciona o SVM existem quatro conceitos básicos necessários, os quais são [5]: separação do hiperplano, o hiperplano de margem máxima, a margem suave, e a função de núcleo. O hiperplano de separação, é a linha que separa as classes em um

espaço de alta dimensão. O SVM gera diversas linhas ou hiperplanos de separação, porém a que possuir a maior distância entre os pontos das classes distintas é o hiperplano de margem máxima, que será usado pelo algoritmo. Já a margem suave é a capacidade do SVM permitir alguns erros, ou *outliers*, para manter a margem o mais larga possível. Por fim, a função de núcleo adiciona mais uma dimensão aos dados para facilitar a separação.

Na Figura 3.2 é exemplificado o algoritmo do SVM. Inicialmente, o objetivo é identificar se o ponto azul no plano A é da classe verde ou vermelha. Assim, é traçada a separação do hiperplano. O plano B apresenta como funciona o hiperplano de margem máxima, no qual é traçado a linha com maior distância entre as duas classes. O plano C mostra como a margem suave se mantém no meio das classes mesmo, com um *outlier* que é representado pela seta. Para finalizar, o Plano D indica a função do núcleo para aumentar uma dimensão e facilitar a separação das classes.

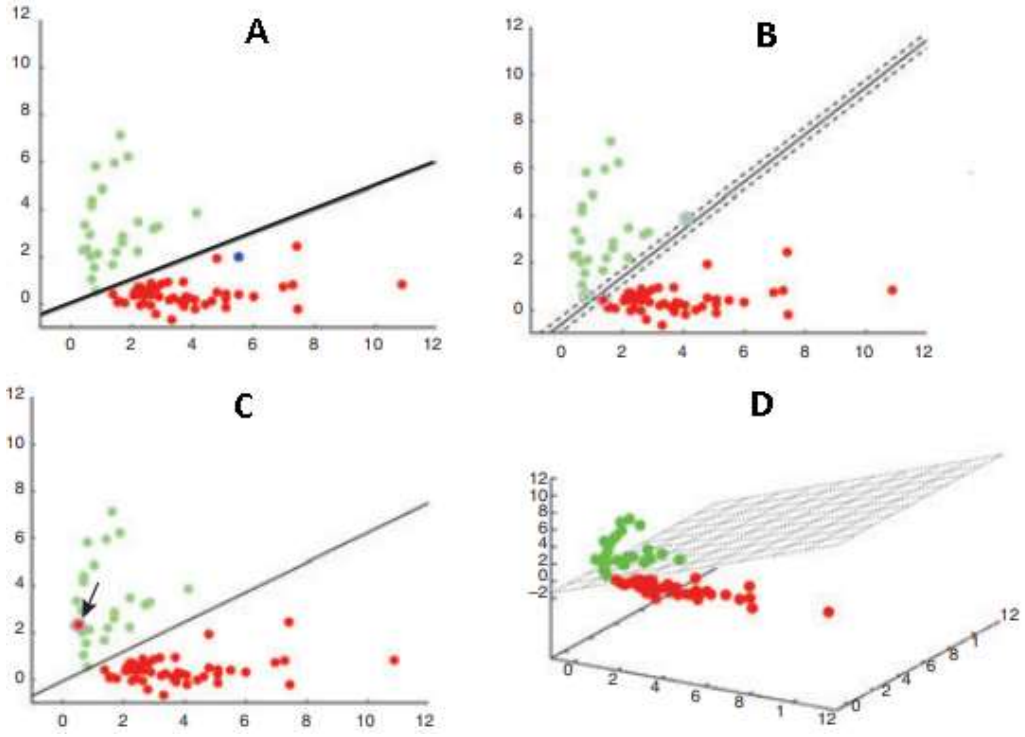

Figura 3.2: Exemplo do algoritmo de SVM (Fonte: Adaptado de [5]).

3.1.4 Árvore de Decisão

A Árvore de Decisão é um classificador caracterizado por seus dados não conhecidos serem rotulados em uma classe, a partir de funções de decisão de maneira sucessiva [69]. Essa classificação pode ser representada por um diagrama de árvore, o que traz um grande benefício para essa abordagem, a possibilidade de auditar as decisões e identificar quais

parâmetros são mais relevantes para a escolha de uma classe. Nessa estrutura, os ramos representam as operações lógicas das entradas que levam as folhas, que são representadas pelas classes.

A maioria das Árvores de Decisão realiza a classificação em duas etapas: construção da árvore e poda. Na etapa de construção do modelo, a árvore é gerada a partir da divisão recursiva do conjunto de dados de treinamento, baseado no critério do ótimo local. Os dados são divididos até que todos, ou a maioria, dos registros de uma partição estejam na mesma classe. Para otimizar a geração de Árvores de Decisão, a poda é utilizada para remover folhas e ramos responsáveis por classificar poucos dados [70].

A Figura 3.3 é um exemplo de uma árvore de decisão. Nela é possível visualizar os nós da árvore e como estes atuam em regras de decisão. Assim, novas entradas chegam a raiz, percorrem as regras de decisão e são direcionadas a novos nós, até que cheguem a uma folha e tenham sua classe predita.

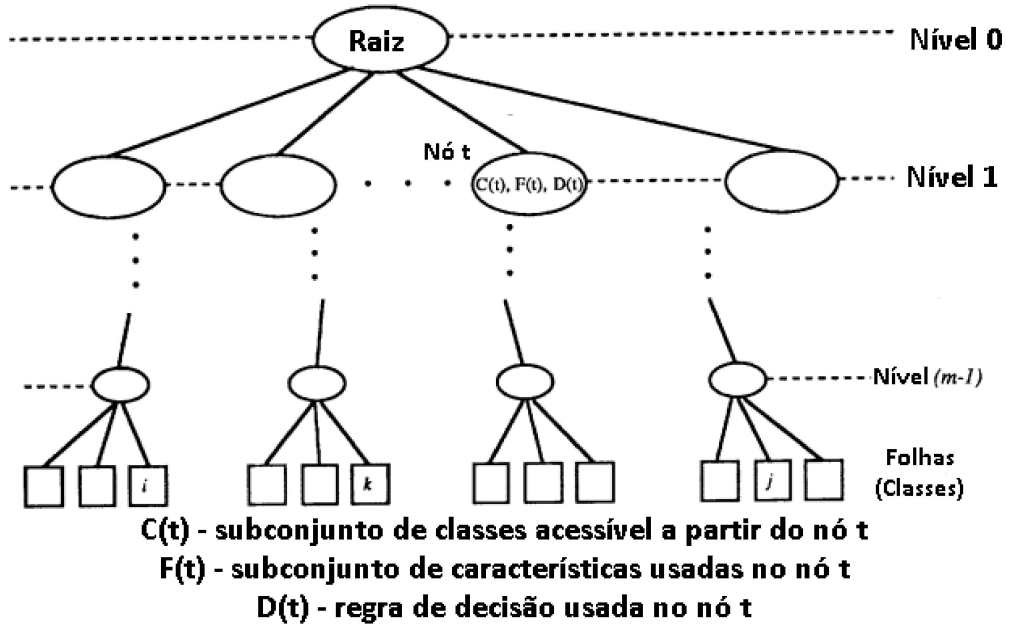

Figura 3.3: Exemplo de uma Árvore de Decisão (Fonte: Adaptado de [6]).

3.1.5 Floresta Aleatória

Floresta Aleatória é um modelo de classificação composto. Modelos compostos são um conjunto de outros classificadores que formam um único classificador final. Este modelo foi proposto por Breiman [28], que se inspirou no trabalho de Amit e Geman [71]. Apesar de parecer complexo, é um modelo simples que se adapta a diversos problemas de predição e possui poucos parâmetros de otimização, o que contribui para a sua popularidade [72].

O algoritmo funciona a partir do princípio de dividir para conquistar, de forma que os dados são separados em amostras menores, e são criadas Árvores de Decisão de forma aleatória para cada subconjunto de dados [72]. Ao final, cada um dos modelos realiza uma predição da classe, que são agregados em um sistema de votação. A classe que obter mais votos, é a classe final deste modelo composto.

3.1.6 Adaboost

Adaboost é um modelo de classificação composto, assim como a Floresta Aleatória. É baseado no *boosting*, uma abordagem que cria predições acuradas por meio da combinação de regras fracas [73]. O Adaboost foi proposto por [74] e, atualmente, é um dos algoritmos mais utilizados e estudados em diversos campos de aplicação.

O algoritmo funciona por meio de repetidas iterações na tentativa de aproximar um classificador de Bayes. Inicialmente, são gerados classificadores fracos, como árvores de decisão pequenas, no qual os dados de treinamento são classificados. Todos os dados iniciam com o mesmo peso, porém, se um determinado dado é classificado errado, o seu peso de treinamento é aumentado (*boosting*) [75]. Assim, novos classificadores são gerados para os novos pesos na iteração seguinte, o que diferem dos classificadores antigos.

Logo, conforme as iterações vão ocorrendo, exemplos de dados que eram difíceis de classificar recebem um aumento de influência, forçando os classificadores fracos a focarem nos exemplos errados anteriormente. Por fim, o algoritmo também utiliza um sistema de votação em que cada classificador emite uma predição, e a classe com mais votos é a classe final predita.

3.1.7 Redes Neurais Artificiais

Uma Rede Neural Artificial (RNA) é um modelo computacional que simula a rede de neurônios do nosso cérebro. Ela pode ser definida como um conjunto de neurônios (unidades de processamento), ligados entre si e implementados por vetores de pesos sinápticos [76]. A primeira publicação de um modelo matemático inspirado em um neurônio biológico foi feita por McCulloch e Pitts [77]. Nos anos seguintes foram desenvolvidas outras pesquisas relacionadas ao tema, e em 1958 Rosenblatt propôs o modelo mais básico de rede neural, o modelo Perceptron [78].

Um neurônio biológico é uma célula nervosa composta por dendritos, axônio e um corpo celular. A sua função é receber sinais, processá-los e emitir um novo sinal a outros neurônios. Os dendritos são responsáveis por receber os sinais, enquanto os axônios são responsáveis pela transmissão. A conexão entre o fim de um axônio e um dendrito vizinho é chamada de sinapse, quando ocorre a comunicação entre dois neurônios [8].

Computacionalmente, um neurônio pode ser representado como uma função matemática que recebe entradas, aplica pesos a essas entradas, calcula sua soma e emite uma saída. Após a soma dos valores, é aplicada uma função de ativação, a qual é avaliada se o resultado emitido está acima ou abaixo de um determinado limite, condicionando a uma saída binária. Na Figura 3.4 é apresentado o modelo de um neurônio proposto por McCulloch e Pitts [77], no qual as variáveis de entradas correspondem aos Xs, os pesos correspondem aos Ws, a soma das entradas e dos pesos é calculada pela fórmula no centro da figura e ,por fim, na etapa seguinte é aplicada uma função de ativação que irá definir o valor binário da saída.

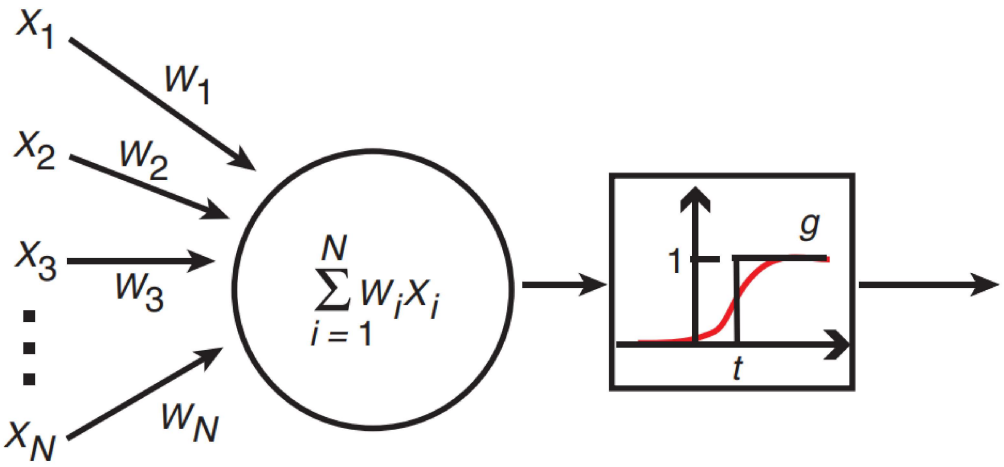

Figura 3.4: Neurônio artificial McCulloch-Pitts (Fonte: Adaptado de [7]).

As Redes Neurais Artificiais (RNAs) podem ser representadas por gráficos direcionados, de forma que os neurônios são os nós, e suas conexões são as arestas. Tendo em vista o seu padrão de conexões, elas podem ser classificadas em dois tipos: as redes sem realimentação e as redes com realimentação [79]. As redes sem realimentação (do inglês *feed-foward*) são caracterizadas por grafos sem ciclos, no qual os neurônios são organizados por camadas e o sinal percorre a rede em uma única direção. Na Figura 3.5 é ilustrada uma rede sem realimentação de múltiplas camadas. Assim, são exemplos populares de redes sem alimentação a *Perceptron*, a *Perceptron* de várias camadas [80], e a Rede Neural Convolucional [81].

As redes com realimentação ou recorrentes, são caracterizadas por ciclos, podendo seus neurônios alimentarem neurônios de camadas anteriores, da própria camada ou até realimentarem o próprio neurônio [82]. Devido a sua estrutura de retroalimentação este tipo de rede está frequentemente atualizando os estados dos seus neurônios, criando assim uma estrutura de memória em que os dados do passado influenciam no processamento dos dados atuais. Assim, são exemplos de redes recorrentes: as Redes Hopfield [83], Unidade

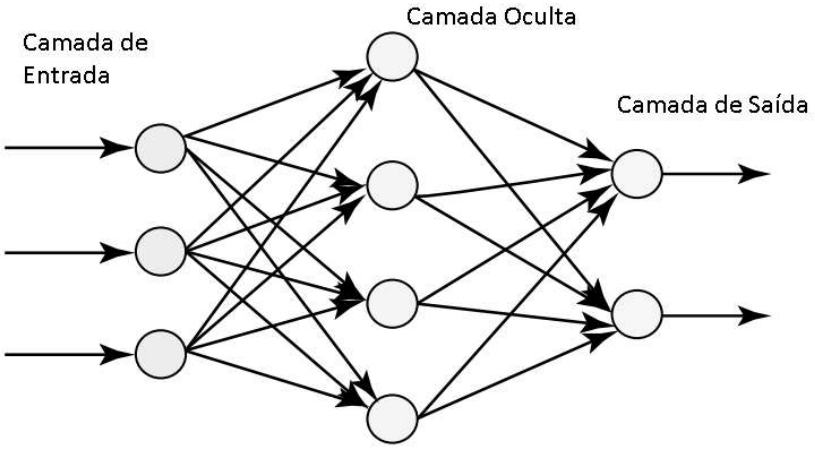

Figura 3.5: Rede neural sem realimentação (Fonte: Adaptado de [8]).

Recorrente Fechada (GRU) [84], e as LSTMs (*Longo Short-Term Memory*) que são objeto de estudo deste trabalho e serão abordadas com mais detalhes na próxima seção.

Long Short-Term Memory

Redes Neurais Recorrentes se destacam em relação as demais devido a sua habilidade de memória. Porém, a cada iteração, a influência dos dados antigos diminui em relação à novas entradas. Essa limitação é conhecida na literatura como o problema da dissipação do gradiente [85]. Para solucionar o problema da dissipação do gradiente, Hochreiter e Schmidhuber [86] propuseram a Rede de Memória Longa de Curto Prazo (do inglês *Long Short-Term Memory* - LSTM).

A arquitetura de uma LSTM é formada por um conjunto de sub-redes, denominadas de blocos de memória. Cada bloco de memória, em sua estrutura mais simples, é composto por uma célula de processamento, uma porta de entrada, uma porta de saída e uma porta de esquecimento [9]. Na Figura 3.6 é apresentado um bloco LSTM, no qual é possível visualizar sua estrutura composta pelas portas de entrada, saída e esquecimento, respectivamente, pelas letras *i*, *o* e *f*, além da sua estrutura de processamento representado pela função de multiplicação e soma das entradas. Essa estrutura previne que o estado da célula seja alterado por entradas irrelevantes, preservando informações por um longo período de tempo.

Devido a sua capacidade de processar dados sequenciais, as LSTMs podem ser usadas para diferentes tarefas, tais como a previsão de dados futuros, regressão, classificação, variados tipos de reconhecimentos e até geração de sequências [87]. Outra relevante usabilidade das LSTMs é na detecção de anomalias em séries temporais [88], visto que anomalias são eventos raros em um conjunto de dados, e podem ser mantidos na memória

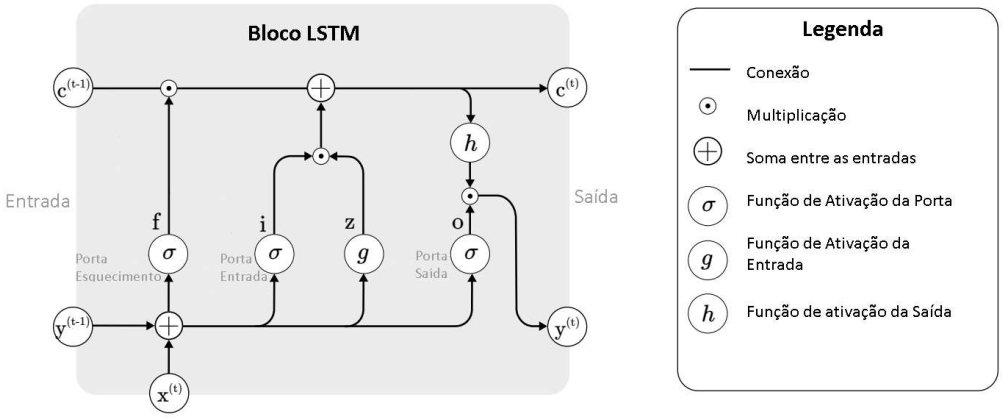

Figura 3.6: Bloco de memória LSTM (Fonte: Adaptado de [9]).

dos blocos por um longo período de tempo, facilitando assim a sua identificação em novas entradas.

3.1.8 Avaliação dos Algoritmos

Uma etapa fundamental do processo de Aprendizado de Máquina é a avaliação dos seus algoritmos. Nela são aplicadas diferentes métricas que irão avaliar características distintas dos resultados obtidos. Assim, sob a perspectiva de um Aprendizado Supervisionado para uma tarefa de classificação, tais métricas são focadas, principalmente, na habilidade do algoritmo de identificar a classe correta do dado.

Assim, os dados que possuem seus valores preditos iguais aos valores reais são considerados verdadeiros, enquanto os dados preditos que divergem dos valores reais são considerados como falsos. A matriz de confusão é uma tabela que permite o entendimento dos resultados de uma classificação binária, na qual é possível comparar os valores reais dos dados, com os valores preditos por um classificador [89]. Os dados que possuem valores reais positivos, e são preditos por um algoritmo de classificação também como positivos, são considerados como Verdadeiros Positivos (VP). Da mesma maneira, aqueles que possuem valor real como negativo e são preditos como verdadeiros, são classificados como Falsos Negativos (NF). Na Tabela 3.1 é apresentada uma exemplificação da matriz de confusão, no qual as colunas representam a classe predita e as linhas a classe original dos dados.

Nesse contexto, a acurácia serve como uma medida ampla para avaliar o desempenho do classificador. Ela se baseia na análise de todos os valores contidos na matriz de confusão durante seu cálculo, sendo uma das métricas mais empregadas para essa finalidade. Contudo, a acurácia não considera o desequilíbrio entre o número de registros em cada classe, o que pode levar a uma interpretação distorcida sobre o desempenho do algoritmo.

Tabela 3.1: Matriz de confusão.

Classe Original \Predita	Como Positiva	Como Falsa
Positiva	VP	FN
Negativa	FP	VN

Por exemplo, em cenários de detecção de anomalias, em que a classe de anomalias representa menos de 1% do conjunto total de dados, um classificador que não identifica corretamente nenhuma anomalia ainda apresentaria uma acurácia de 99%. A Equação 3.4 apresenta a fórmula da acurácia, de forma que os valores preditos corretamente são divididos pelo total de predições:

$$\text{Acurácia} = \frac{\text{VP} + \text{VN}}{\text{VP} + \text{VN} + \text{FP} + \text{FN}} \tag{3.4}$$

Para uma avaliação mais aprofundada dos resultados, especialmente focando nos valores mais relevantes, ou seja, os casos positivos, é comum recorrer às métricas de precisão e *recall*. O *recall* se destaca ao medir a capacidade do modelo de identificar corretamente os exemplos positivos em relação ao total de valores verdadeiros. Em contextos em que identificar casos positivos é de extrema importância, como na monitorização de eventos, um alto valor de *recall* se torna essencial para minimizar falsos negativos. Em outras palavras, sua relevância está na redução das predições em que ocorreu uma falha, mas o modelo classificou como normal. A Equação 3.5 apresenta a fórmula do *recall*, no qual os valores verdadeiros positivos são divididos pela soma do total de valores verdadeiros.

$$\text{Recall} = \frac{\text{VP}}{\text{VP} + \text{VN}} \tag{3.5}$$

A precisão, por sua vez, é expressa como a proporção de verdadeiros positivos em relação ao total de valores classificados como positivos, conforme evidenciado na Equação 3.6. Nesse sentido, para alcançar uma maior precisão, é imperativo minimizar a ocorrência de falsos positivos. Esse princípio torna-se especialmente crucial em cenários de monitoramento, onde a prevenção de alarmes incorretos é essencial, especialmente quando esses alarmes podem resultar em ações disruptivas, como a reinicialização de uma aplicação.

$$\text{Precisão} = \frac{\text{VP}}{\text{VP} + \text{FP}} \tag{3.6}$$

O *F1-Score* é uma métrica mais robusta, pois combina tanto a precisão quanto o *recall* em uma única medida, proporcionando uma avaliação mais abrangente do desempenho de um modelo de classificação. Essa métrica é especialmente útil em cenários nos quais se busca equilibrar a ocorrência de falsos positivos e falsos negativos, como na monitorização.

O *F1-Score* é calculado como a média harmônica entre a precisão e o *recall* [90], conforme apresentado na Equação 3.7, resultando em um valor que reflete a capacidade do modelo em identificar exemplos positivos e classificá-los corretamente. Por esse motivo, o *F1-Score* foi escolhido como a principal medida de avaliação deste trabalho.

$$F_1 = \frac{2 \text{ x Precisão} \times \text{Recall}}{\text{Precisão} + \text{Recall}} \tag{3.7}$$

A Área sob a Curva ROC (ROC-AUC) é uma medida de avaliação essencial na análise de modelos de classificação. Ela quantifica a capacidade global do modelo em distinguir entre as classes positiva e negativa, independentemente do ponto de corte escolhido para a classificação. Quanto maior a ROC-AUC, mais eficaz é o modelo em classificar corretamente os exemplos positivos em relação aos negativos. A ROC-AUC é frequentemente utilizada em problemas de desbalanceamento de classes devido a sua resiliência ao desequilíbrio, o que a torna uma métrica confiável no cenário deste trabalho.

Em contextos de classificação binária, o cálculo da ROC-AUC segue a fórmula apresentada na Equação 3.8. Nessa equação, S_p soma todas as instâncias positivas, ao passo que n_p e n_n representam, respectivamente, a quantidade de exemplos positivos e a quantidade de exemplos negativos[91]. Conforme se aproxima do valor 1, a ROC-AUC sinaliza a capacidade do modelo de discernir entre as classes, enquanto valores próximos de 0.5 indicam que o modelo apresenta dificuldade em distinguir as classes de forma significativa.

$$AUC = \frac{S_p - n_p(n_n + 1)/2}{n_p n_n} \tag{3.8}$$

3.2 Trabalhos Relacionados

Segundo o ITIL, a automação de processos na gestão de serviços de TI é um ponto chave no ganho de eficiência operacional. Diversos processos que no início do ITIL eram realizados manualmente, atualmente podem ser implementados pela automação computacional. Assim, o uso de sistemas de Aprendizado de Máquina tem se mostrado vantajoso para reduzir o tempo e o custo financeiro de determinadas tarefas. A seguir, serão apresentados trabalhos que implementam processos de Aprendizado de Máquina no contexto de Gestão de Serviços de TI, e trabalhos que têm aprimorado o estado da arte, especificamente, na área de detecção de falhas (monitoração de incidentes) em Sistemas de Informação.

3.2.1 Aprendizado de Máquina na Gestão de Serviços de TI

A categoria de um incidente é fundamental na priorização e no encaminhamento para a correta equipe de solução. Assim, quando implementada de forma automática e eficiente,

auxilia as organizações nas fases iniciais do ciclo de vida do incidente, reduzindo o seu tempo de resolução. Cada categoria pode ser vista como a classe de serviços que determinado incidente se encaixa, que também está relacionada a qual equipe de suporte será responsável pela atuação.

Al-Hawari e Barham [92] propuseram um sistema central de ajuda (*help desk*) que usa técnicas de Aprendizado de Máquina na classificação da categoria dos incidentes. Shao et al. [93] [94] propuseram um sistema de roteamento dos incidentes por meio de um modelo baseado na Cadeia de Markov [95], que reduz a quantidade de encaminhamentos, chegando a equipe solucionadora mais rápido e, consequentemente, reduzindo o tempo de resolução de um incidente. Miao et al. [96], por meio da máxima verossimilhança, desenvolveram um algoritmo probabilístico de recomendação de encaminhamento em uma rede de especialistas. Assim, calcularam possíveis rotas para potenciais solucionadores, reduzindo o número de encaminhamentos. Diversos outros trabalhos abordaram a classificação e o roteamento de incidentes por meio de técnicas de processamento de linguagem natural, utilizando o texto dos incidentes como [97] [98] [99].

A recuperação de procedimentos de solução também é uma importante tarefa no processo de gestão de incidentes, auxiliando as equipes de suporte no restabelecimento dos serviço. Quando um operador recebe uma recomendação assertiva e de forma automática, economiza-se o tempo da busca manual em uma base de conhecimentos e, consequentemente, o tempo de resolução de um incidente é reduzido.

Neste contexto, Zhou et al. [100] desenvolveram um sistema de recomendação de solução por meio do processamento dos títulos dos eventos de monitoração e o algoritmo do k-vizinho mais próximo [25]. Em outro trabalho, Zhou et al. [101] usou técnicas de redes neurais profundas para melhorar os resultados dos algoritmos de recomendação. Yun et al. [102] aplicou as técnicas de processamento de texto TF-IDF [103] e algoritmos de classificação para recomendar soluções e itens de configuração afetados pelos incidentes.

Os trabalhos apresentados atuam no campo de otimização de processos da GSTI. No entanto, não abordam o tema de identificação de falhas, processo essencial e prévio a classificação de incidentes. Tal lacuna, será abordada ao longo deste estudo.

3.2.2 Detecção de Falhas em Registros de Sistemas

Com o intuito de reduzir o tempo necessário para resolver falhas ou, ainda, de atuar proativamente para prevenir o surgimento de tais falhas, diversos estudos têm adotado abordagens de Aprendizado Supervisionado. Isso possibilita que modelos identifiquem possíveis anomalias em Sistemas de Informação o mais cedo possível. Um exemplo é o trabalho de Du et al. [104], que propôs um sistema de detecção de anomalias em aplicações distribuídas como o *Hadoop Distributed File System* (HDFS) [105] e o *Open*

Stack [106], usando os registros de sistema (logs). Por meio da aplicação de redes LSTMs [86] e processamento textual, eles conseguiram identificar padrões de execução normal e detectar anomalias quando os registros se desviavam do padrão, utilizando o Aprendizado Supervisionado.

Outra abordagem significativa foi a de Zhang et al. [107], que propuseram um modelo de detecção de anomalias para lidar com a variação dos logs ao longo do tempo. Eles utilizaram redes neurais Bi-LSTMs [108] com mecanismo de atenção, permitindo a captura de informações contextuais e o aprendizado automático da importância de diferentes eventos de logs.

Adicionalmente, Zhao et al. [109] integraram diversas técnicas de detecção de anomalias em registros de sistemas, resultando na criação de um modelo composto que aprimora consideravelmente a capacidade de generalização em diversos cenários. Eles também construíram uma base de conhecimento para o pré-processamento de dados, no qual cada padrão de registro é tratado com o modelo mais apropriado. Essa abordagem permitiu que o modelo lidasse eficazmente com uma ampla gama de registros, identificando segmentos anômalos que, se não fossem detectados, poderiam resultar em problemas de disponibilidade no futuro.

Uma abordagem amplamente explorada na literatura no contexto de detecção de anomalias envolve a implementação de algoritmos de Aprendizado Não-Supervisionado. Um exemplo é o trabalho de Munir et al. [110], que desenvolveu um sistema de detecção de anomalias utilizando redes neurais convolucionais profundas (CNN) [111] e Aprendizado Não-Supervisionado em séries temporais. O modelo foi aplicado a dez conjuntos de dados para a detecção de anomalias e superou os outros métodos do estado da arte em termos de desempenho. Por sua vez, Meng et al. [112] adotaram a técnica *Template2Vec*, uma variação do *Word2vec* [113], em conjunto com redes neurais LSTMs para a detecção de anomalias em registros de sistemas distribuídos utilizando Aprendizado Não-Supervisionado. Essa abordagem permitiu evitar falsos alarmes decorrentes da ocorrência de novos padrões de logs entre os períodos de reajuste do modelo.

3.2.3 Detecção de Falhas na Nuvem

Com a popularidade da computação em nuvem, alguns trabalhos têm abordado a predição de falhas neste ambiente. Islam et al. [114] propuseram uma arquitetura para detecção de anomalias nos componentes do ambiente de nuvem. Seu trabalho envolve a captura de dados de telemetria dos recursos e seu processamento como uma série temporal usando uma rede autoencoder baseada em GRU (Unidade Recorrente com Portões). Essa abordagem possibilita a identificação eficaz de anomalias nos dados dos componentes, acionando automaticamente um alerta à equipe de suporte para análise imediata da ocorrência.

Girish e Rao [115] desenvolveram um modelo avançado de rede neural Bi-LSTM para detectar anomalias em seus componentes. O modelo é alimentado com dados de recursos consumidos e tráfego de rede de aplicativos. Essa abordagem permitiu a identificação pontual de mudanças prejudiciais nos dados, possibilitando medidas proativas para evitar falhas no aplicativo.

Além disso, Dai et al. [116] introduziram um modelo de aprendizado profundo com redes neurais Bi-LSTMs para prever falhas em contêineres no ambiente Kubernetes. Esses estudos exemplificam a aplicação de arquiteturas avançadas de redes neurais e técnicas de aprendizado de máquina para aprimorar a previsão de falhas em sistemas baseados em nuvem.

Gao et al. [117] implementaram um sistema de detecção de falhas em tarefas na nuvem do Google, processando dados referentes ao consumo dos recursos consumidos como processador, memória e disco. O sistema utilizou uma arquitetura com redes neurais Bi-LSTMs [86] de duas camadas, combinada com uma camada de regressão logística, no qual apresentou resultados melhores do que comparado a outros modelos, tais como rede oculta de Markov [95] e Máquinas de Vetores de Suporte (SVM) [68].

Chen et al. [118] também desenvolveram um preditor de falhas em tarefas da nuvem do Google, aplicando redes neurais recorrentes, o que reduziu o consumo de recursos em 10% segundo os autores. Yuan et al. [119], por meio do processamento dos *logs* da ferramenta *Open Stack* [106], propuseram uma abordagem de detecção de falhas nas execuções de provisionamento de recursos da ferramenta. A abordagem utiliza a técnica *Word2Vec* para representar as palavras colhidas nos registros de *logs*, e aplica alguns algoritmos de classificação para comparar os resultados. O algoritmo que obteve o melhor resultado foi a Florestas Aleatória [28]. Dai et al. [116] propuseram um modelo de aprendizado com redes neurais profundas Bi-LSTMs para prever falhas em *containers* no *Kubernetes*.

Os trabalhos apresentados abordam o tema de detecção de falhas em Sistema de Informação, por meio da análise de registros (*logs*), dos componentes ou consumo dos recursos [104] [107] [110]. No entanto, em sistemas complexos que combinam diferentes infraestruturas como *mainframe*, nuvem e virtualização, faz-se necessária adotar uma abordagem que englobe uma visão geral do serviço. Tal estratégia pode ser obtida por meio da aplicação de monitoração na camada do usuário, na qual são avaliadas as interações dos usuários com o sistema. Assim, é possível ter uma avaliação da disponibilidade total do serviço e não apenas de alguns dos seus componentes. Tal abordagem possibilita o acionamento automático de fluxos de gestão de serviços de TI, e uma comunicação mais eficaz, permitindo o contorno mais rápido da queda dos serviços e, consequentemente, aumentando a satisfação dos clientes.

Além disso, alguns desses estudos não utilizam dados provenientes de contextos do

mundo real, o que pode resultar em disparidades quando aplicados em ambientes de produção. Outro aspecto fundamental que não é abordado na maioria das pesquisas revisadas é a implementação de estratégias de monitoramento para serviços na nuvem. Esses pontos críticos são analisados e comparados na Tabela 3.2, evidenciando as lacunas existentes nos estudos da área.

Tabela 3.2: Tabela comparativa entre os trabalhos relacionados.

Trabalho	**Redução do Tempo Médio de Reparo**	**Redução do Tempo Médio de Detecção**	**Redes Neurais Artificiais**	**Otimização de algoritmos**	**Dados de contextos reais**	**Aplicações na Nuvem**	**Dados de experiência dos usuários**
Al-Hawari e Barham [92]	X						
Shao et al. [93]	X			X	X		
Miao et al. [96]	X			X	X		
Zhou et al.[100]	X				X		
Zhou et al. [101]	X		X		X		
Yun et al.[102]	X		X	X	X		
Du et al.[104]	X	X	X				
Zhang et al.[107]	X	X	X		X		
Zhao et al.[109]	X	X	X		X		
Munir et al.[110]	X	X	X	X	X		
Meng et al.[112]	X	X	X		X	X	
Islam et al.[114]	X	X	X	X	X	X	
Girish et al.[115]	X	X	X			X	
Dai et al.[116]	X	X	X			X	
Gao et al.[117]	X	X	X		X	X	
Trabalho proposto	X	X	X	X	X	X	X

A Tabela 3.2 oferece uma visão sobre se os estudos analisados se concentram na otimização do processo de gerenciamento de TI, com o intuito de reduzir o Tempo Médio de Reparo, ou se estão direcionados para o campo da monitoração de serviços, visando também a diminuição do Tempo Médio de Detecção. Além disso, as pesquisas foram exa-

minadas para verificar se expõem estratégias de otimização de algoritmos e o uso de Redes Neurais, tecnologias atualmente relevantes em termos de algoritmos de classificação. Essa comparação proporciona uma compreensão das abordagens adotadas nos estudos, permitindo uma avaliação das contribuições de cada pesquisa para o campo da detecção de falhas em Sistemas de Informação. Ao evidenciar essas lacunas e tendências, este trabalho se posiciona como uma peça essencial para preencher essas necessidades, destacando sua relevância e importância no cenário acadêmico e profissional.

3.3 Considerações Finais

No transcurso deste capítulo, foram explorados diversos algoritmos de classificação, cada um com suas características únicas para a tarefa de classificação. Desde o simples, porém eficaz Bayes ingênuo, até o poderoso SVM que traça fronteiras de decisão complexas, passando pela interpretabilidade das Árvores de Decisão e a potência do AdaBoost em melhorar o desempenho de algoritmos mais simples. As Redes Neurais e, mais especificamente, as LSTMs, demonstraram a habilidade de lidar com dados sequenciais.

Além disso, foi explorada uma variedade de abordagens e pesquisas relacionadas ao uso de Aprendizado de Máquina na Gestão de Serviços de TI, detecção de falhas em registros de sistemas e detecção de falhas na nuvem. A aplicação de técnicas de Aprendizado de Máquina na Gestão de Serviços de TI tem demonstrado grande potencial para melhorar a eficiência, a confiabilidade e a qualidade dos serviços prestados. A capacidade de analisar grandes volumes de dados em tempo real permite a detecção precoce de problemas, a prevenção de interrupções e a tomada de decisões mais informadas.

A detecção de falhas em registros de sistemas é uma abordagem promissora para manter a integridade e a segurança das operações de TI. Os estudos discutidos destacaram abordagens baseadas em análise de padrões, detecção de anomalias e processamento de linguagem natural. Com o aumento da adoção de serviços em nuvem, a detecção de falhas nesse ambiente tornou-se crucial. Abordagens que envolvem monitoramento contínuo, análise de desempenho e modelagem preditiva têm sido exploradas para identificar problemas que afetam a disponibilidade e a performance dos serviços em nuvem.

Os estudos apresentados efetuam a detecção de falhas com excelência, permitindo, em alguns casos, a adoção de estratégias proativas para evitar impactos sobre o cliente. No entanto, em um ambiente complexo e heterogêneo, formado por diversos subsistemas de Tecnologia da Informação, se faz necessária uma abordagem que englobe uma visão global dos serviços. Uma estratégia que foi explorada ao longo deste trabalho, é análise de dados do comportamento dos usuários, permitindo um acompanhamento da persona mais

importante ao negócio, o cliente. No próximo capítulo, essa estratégia será apresentada com mais detalhes.

Capítulo 4

Proposta

Neste capítulo será apresentada, em detalhes, a proposta deste trabalho, descrevendo como foi aplicada a metodologia e as etapas de mineração de dados. Inicialmente, na Seção 4.1 será apresentada a arquitetura da proposta. Na Seção 4.2 será abordado o entendimento do negócio, no qual serão apresentados os objetivos sob o ponto de vista do negócio. Na Seção 4.3 serão apresentadas informações para o entendimento dos dados, tais como sua origem, a descrição estatísticas e as informações para facilitar a compreensão. Na Seção 4.4 é introduzida a etapa de preparação dos dados, na qual são apresentadas as transformações para alcançar o conjunto de dados finais. Na Seção 4.5 é apresentada a etapa de modelagem da proposta, na qual serão aplicadas técnicas de mineração de dados. Por fim, na Seção 4.6 serão apresentadas as considerações finais deste capítulo.

4.1 Arquitetura

A estrutura deste projeto de mineração de dados é composta por duas etapas principais: treinamento em lote e classificação em tempo real. Essas fases foram concebidas para atender a distintas demandas e fluxos de dados. O esquema completo pode ser observado na Figura 4.1, no qual são apresentadas as principais atividades de cada etapa.

Na etapa de treinamento em lotes, o objetivo é construir um modelo de classificação a partir do conjunto de dados históricos. Essa etapa deve ser executada em lotes periódicos, utilizando todo o conjunto de dados disponível. Durante o treinamento em lotes, os dados são processados de forma *off-line*, permitindo uma análise mais aprofundada e complexa.

A etapa de treinamento em lotes começa com a obtenção dos dados históricos, que são divididos em conjuntos de treinamento, validação e teste. Em seguida, são realizadas etapas de otimização do pré-processamento, que envolvem mapeamento dos atributos de entrada do modelo e controle da proporção de sobre-ajuste. Os modelos pré-selecionados passam por otimização de parâmetros e, ao final, são avaliados no conjunto de testes.

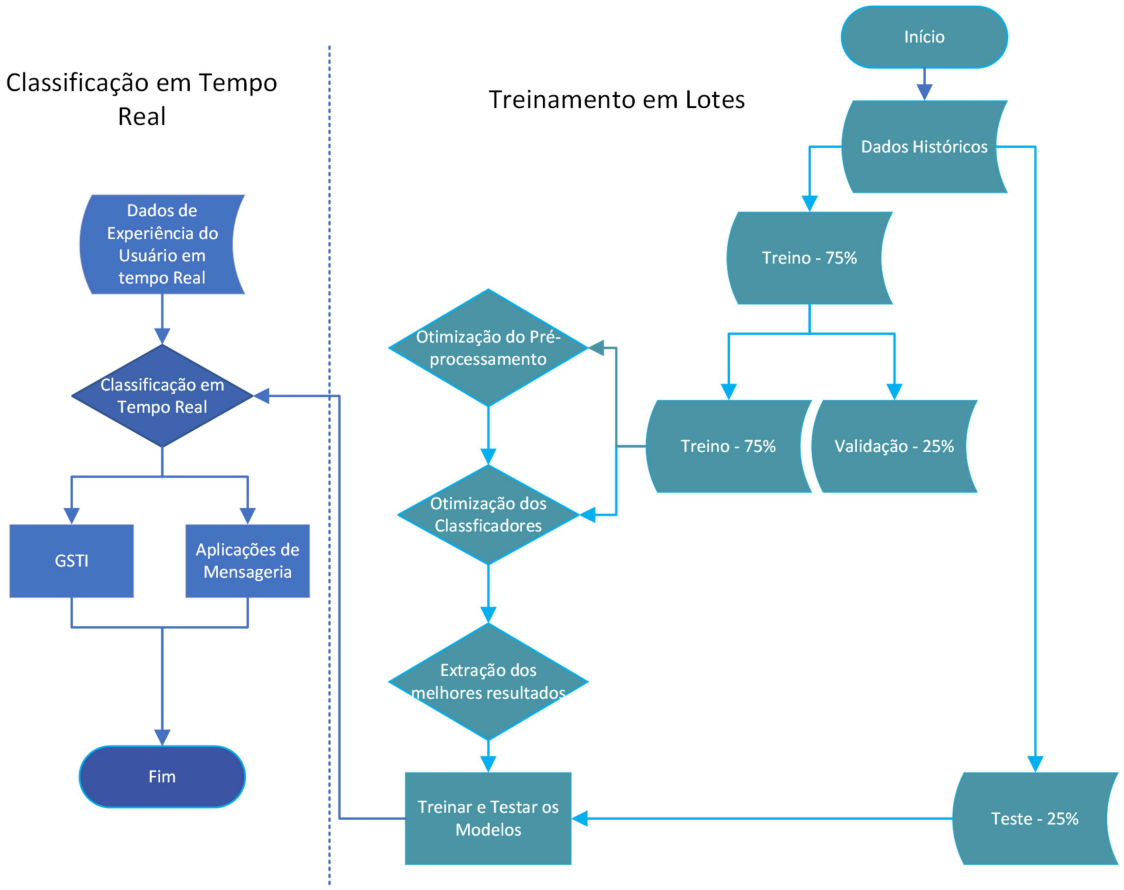

Figura 4.1: Arquitetura da solução proposta neste trabalho.

Dessa forma, o modelo que apresentar o melhor desempenho no conjunto de testes é selecionado para a etapa de classificação em tempo real.

Na classificação em tempo real, os dados recebidos dos clientes são agrupados a cada um minuto. Após o agrupamento, esses dados passam por uma etapa de pré-processamento, na qual podem ser filtrados atributos e as métricas podem ter seus valores escalados por meio da padronização. Em seguida, os dados são inseridos no modelo de classificação, gerando uma saída que indica o estado do serviço naquele minuto específico. Caso o serviço seja classificado como indisponível, são acionados fluxos de comunicação com os intervenientes, que podem incluir a abertura de um incidente para o time de suporte, bem como o envio de mensagens aos executivos e responsáveis técnicos da aplicação.

A arquitetura do projeto foi desenvolvida seguindo o modelo de referência CRISP-DM. Esta abordagem oferece uma perspectiva abrangente do ciclo de vida do projeto, ao mesmo tempo em que fornece uma estrutura sólida para o planejamento eficaz. O CRISP-DM é composto por seis fases distintas, as quais serão exploradas nas seções seguintes deste capítulo e no subsequente.

4.2 Entendimento do Negócio

Na etapa de Entendimento do Negócio serão definidos os objetivos e as necessidades sob a perspectiva do negócio. Assim, essas informações serão convertidas em um problema de mineração de dados. As próximas seções descrevem essas etapas em detalhes.

4.2.1 Definição de Falha

No contexto deste trabalho, a falha refere-se a uma interrupção ou disfunção em um sistema ou componente que resulta na indisponibilidade ou no mau funcionamento do serviço prestado. Essa falha pode ocorrer devido a vários fatores, tais como problemas técnicos, erros de configuração, sobrecarga, falhas de energia, erros humanos, entre outros.

A maioria das falhas está associada a problemas técnicos e sobrecarga, que podem ocorrer isoladamente ou em conjunto. Um exemplo concreto envolve a queda de um *link* da operadora, que resultou em um acúmulo de clientes na fila e sobrecarregou o sistema de processamento quando o *link* voltou a funcionar. Tal ocorrência afetou o sistema de autenticação impedindo o acesso as principais funcionalidades do aplicativo.

As falhas identificadas e abordadas ao longo deste trabalho abrangem uma parcela significativa, se não total, dos clientes. Elas não se limitam a funcionalidades específicas, mas comprometem o funcionamento do sistema como um todo, resultando na indisponibilidade e no mau desempenho do serviço prestado.

4.2.2 Objetivos do Negócio

Atualmente, os aplicativos móveis são o principal canal de interação entre as instituições financeiras e os seus clientes. Devido a facilidade de consumir serviços bancários pela Internet, a adesão ao *Mobile Banking* tem crescido a cada ano, tornando esse serviço essencial no dia a dia dos brasileiros. Dessa forma, quando o aplicativo deixa de funcionar por causa de alguma falha técnica, os clientes são impactados e sua satisfação é reduzida. Assim, faz-se necessária a monitoração contínua dos serviços prestados para que em momentos de falhas a organização possa iniciar os procedimentos de recuperação o quanto antes.

Logo, instituições financeiras necessitam de uma visão da disponibilidade do serviço prestado em um canal de atendimento como o *Mobile Banking*. A informação da disponibilidade do serviço é necessária para iniciar atividades de recuperação, mobilização das equipes intervenientes, acionamento de automações de reciclagem do ambiente e migração de processos, além da comunicação interna e externa da instituição. Assim, o objetivo deste trabalho é fornecer um serviço que esteja continuamente avaliando as interações

dos usuários com a instituição, e consiga informar quando o sistema está disponível ou indisponível.

Para atingir com sucesso o objetivo do negócio é necessário que o sistema identifique as indisponibilidades em um tempo próximo ao real. Também se faz necessário garantir a acurácia do modelo de maneira que seja minimizada a geração de falsos positivos e falsos negativos, evitando que sejam realizados acionamentos do modelo desnecessários à corporação.

4.2.3 Objetivos de Mineração de Dados

Para desenvolver o problema de negócio será criado um classificador binário. O modelo será alimentado por valores referentes as interações dos usuários (páginas acessadas e mensagens de sistema recebidas pelos clientes), e retornará uma saída com o estado do serviço, disponível ou indisponível. Para que o modelo consiga aprender os diferentes comportamentos das interações dos usuários com o sistema, em momentos de disponibilidade e indisponibilidade, será realizada a rotulação das interações passadas de acordo com os registros de incidentes que relatam falhas generalizadas no serviço.

O modelo, após treinado com interações passadas, será aplicado para classificar novos eventos em tempo real. O modelo proposto não realiza a previsão de falhas. Para atingir o objetivo de mineração de dados, faz-se necessária a elaboração de um fluxo de coleta de dados em tempo real para alimentar o modelo, assim como sua integração aos sistemas da organização.

Dado que a ocorrência específica de indisponibilidade em sistemas de TI é uma situação pouco comum, aferir seu desempenho frequentemente emprega o uso do *F1-Score*. Além disso, na avaliação de problemas de detecção de anomalias, também se recorre à Área sob a curva ROC (ROC-AUC). Consequentemente, esta pesquisa adotará o *F1-Score* como métrica principal e a ROC-AUC como secundária para avaliar a eficácia dos modelos.

4.3 Entendimento dos Dados

Na etapa de Entendimento dos Dados será realizada a coleta, a descrição dos dados, as atividades de exploração para uma familiarização com os dados obtidos, e a formulação de hipóteses para modificar os dados. Para um melhor entendimento desta etapa, as próximas seções desenvolvem as sub-etapas necessárias para a fase de Entendimento dos Dados.

4.3.1 Base de Incidentes

A Base de Incidentes é um banco de dados no qual estão armazenados todos os registros de ocorrências de degradação, falha nos serviços, reclamações dos usuários ou mudanças de estados identificadas pelas ferramentas de monitoração. Os incidentes registrados são persistidos ao longo do tempo para alimentar processos de melhorias e esteiras do ITIL, tais como a gestão de problemas e a melhoria contínua na gestão de incidentes.

Dessa forma, estão armazenados na base de incidentes os eventos de monitoração que foram identificados como prejudiciais aos serviços ofertados pela organização. Além desses, também estão armazenadas todas as reclamações dos clientes externos, correntistas do banco, cliente internos, e funcionários que usam as aplicações de TI em seus trabalhos. Na base os incidentes são catalogados de acordo com o serviço afetado, número de usuários impactados e o tipo de indisponibilidade.

Os registros de incidentes serão usados na etapa de Rotulação dos Dados, na qual serão mapeados os momentos de indisponibilidade do aplicativo bancário. Esses momentos de indisponibilidades serão repassados para o algoritmo de classificação, que irá identificar os padrões em um momento de crise e os padrões em momentos de normalidade do aplicativo. Tais dados serão fundamentais para identificação de novas anormalidades pelo classificador proposto neste trabalho.

4.3.2 Extração e Transformação dos Dados

Ao longo deste trabalho foram coletados dados anonimizados referentes aos acessos dos clientes de uma organização financeira por uma ferramenta de *Web Analytics*. Esses dados correspondem a quantidade de acessos a determinadas páginas, tais como a página de *login* e as páginas de operações financeiras. Também foram coletados eventos gerados pelas páginas, por exemplo as mensagens dos sistemas, indicando o sucesso de operações ou erros. Por fim, foram utilizados os tempos de respostas de algumas operações e o carregamento de determinadas páginas. Tais dados correspondem a três dos quatro sinais de ouro [120], a qual é uma estratégia de coleta de métricas implementada pelo time de engenharia de confiabilidade do Google. Essas métricas são referentes ao tráfego, os erros e à latência dos serviços.

Devido a quantidade de registros que são armazenados pelo projeto Matomo, foi necessário o uso de uma solução de *Big Data*, que é um conjunto amplo de dados caracterizado pelo seu volume, variabilidade e velocidade que são gerados [121]. O Apache Hive [122] é uma infraestrutura de *data warehouse* baseada no Apache Hadoop [123]. Ele foi projetado para permitir fácil sumarização, consultas e análise de grandes volumes de dados. Já o *Hadoop Distributed File System* (HDFS) [105] é um sistema de arquivos distribuí-

dos construído para executar sobre computadores de baixo custo. Logo, trata-se de uma abordagem altamente tolerante a falhas. Além disso, o HDFS provê alta vazão de dados, além de ser adequado para aplicações que manipulam grande volume de dados. Por fim, foi utilizado o Apache Spark [124], uma solução de processamento em memória, que é construído sobre a abstração *Resilient Distributed Datasets* (RDDs), que fornece um compartilhamento eficiente de dados durante o seu processamento.

Os dados estavam armazenados em uma única tabela no Apache Hive [122]. Inicialmente, foi realizada uma análise exploratória para identificar quais campos da tabela eram, de fato, significativos para o problema. Assim, três campos foram selecionados: o campo referente ao texto da URL (*Uniform Resource Locator*) das páginas acessadas; o campo referente aos texto das ações de eventos; e o campo referente a categoria das ações dos eventos. As ações dos eventos representam ações executadas pelos usuário ou pelo sistema conforme a interação, tais como cliques em páginas ou mensagens dos sistemas apresentadas ao usuário. A categoria das ações dos eventos representam a classe de cada ação, como por exemplo uma mensagem de erro de negócio ou uma mensagem de erro do sistema.

Para cada um dos três campos selecionados na etapa anterior, fez-se uma análise de frequência do conteúdo, com o objetivo de identificar os valores mais frequentes de cada campo. Como exemplo, no campo da URL do texto foram identificadas as 200 páginas mais acessadas pelos clientes no período. Também foram mapeados os 200 valores mais frequentes para as ações de eventos, e os 200 valores para as categorias. A partir dos valores mais frequentes de cada campo, foram extraídas as quantidades de ocorrências de cada um dos 600 valores diferentes a cada um minuto em um período de aproximadamente 18 meses, totalizando 800 mil registros.

A Figura 4.2 exemplifica o processo de transformação aplicado para criar o conjunto de dados. A primeira tabela representa como os dados são armazenados na origem, e a segunda como foram organizados após a transformação. Assim, com o agrupamento realizado, o conjunto de dados era composto por 600 campos referentes a quantidade de ocorrências a cada um minuto, mais os valores referentes as datas das capturas. A estratégia de agrupamento das ocorrências dos valores mais frequentes é utilizada em problemas que lidam com processamento textual, para traduzir os dados em ocorrências numéricas que podem ser interpretadas por programas de Aprendizado de Máquina.

4.3.3 Descrição dos Dados

A Análise Descritiva dos Dados é uma técnica estatística que consiste em resumir os dados. Descrever os dados estatisticamente é importante para fornecer uma compreensão geral dos dados e permitir identificar tendências, padrões e relações. Assim, esta etapa

Hora_min	Campo_2	...	TX_CMT_URL_ASCD	TX_ACAO_EVT	TX_CTGR_EVT	...	Campo_n
00:01			HOME (URL1)	PIX (EVT2)	Clique(CTGR1)		
00:01			HOME (URL1)	PIX (EVT2)	Clique(CTGR1)		
00:01			Login (URL2)	PIX (EVT2)	Clique(CTGR1)		
00:02			HOME (URL1)	PIX (EVT2)	Clique(CTGR1)		
00:02			Login (URL2)	Problema(EVT1)	Erro(CTGR2)		
00:03			Login (URL2)	PIX (EVT2)	Clique(CTGR1)		
...	...	...	...	...	...	...	...

Hora_min	URL1	URL2	...	EVT1	EVT2	...	CTGR1	CTGR2	...
00:01	2	1		0	3		3	0	
00:02	1	1		1	1		1	1	
00:03	0	1		0	1		1	0	
...	...	...	...	...	...	...	...		

Figura 4.2: Transformação dos dados aplicada neste trabalho.

pode auxiliar na formulação de hipóteses e tomada de decisões. Nas Tabelas 4.1 e 4.2 são descritos 19 campos selecionados, do total de atributos, no qual estão apresentadas medidas tais como valores máximos e mínimos, média, desvio padrão e os quartis do conjunto de dados. Os dados foram separados em duas tabelas para facilitar a visualização.

Tabela 4.1: Análise descritiva dos Dados 1.

	CTGR5	CTGR8	CTGR14	CTGR36	ACAO21	ACAO26	ACAO27	ACAO28	ACAO29	ACAO38
count	477504.00	477504.00	477504.00	477504.00	477504.00	477504.00	477504.00	477504.00	477504.00	477504.00
mean	11065.81	5621.79	834.71	9.63	702.36	481.04	480.13	325.76	416.28	182.92
std	15956.39	5111.76	1600.97	19.57	1622.40	1958.94	1504.39	811.82	2258.70	321.58
min	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
25%	4072.75	1767.00	185.00	3.00	164.00	83.00	119.00	70.00	88.00	41.00
50%	10605.00	4899.00	646.00	8.00	565.00	325.00	386.00	260.00	307.00	144.00
75%	15202.00	8813.00	1229.00	14.00	1004.00	638.00	658.00	474.00	526.00	276.00
max	709030.00	156223.00	73646.00	825.00	79597.00	126555.00	60346.00	46900.00	100698.00	14339.00

Assim sendo, ao interpretar os resultados apresentados nas Tabelas 4.1 e 4.2, é possível verificar que no geral os dados possuem uma distribuição assimétrica a direita. Em resumo, a distribuição assimétrica a direita é caracterizada por ter uma cauda mais longa na direita, indicando que há mais valores extremos a direita do que à esquerda. Como os dados são referentes a ocorrências de determinados registros, não possuem valores negativos, com o mínimo de zero. Já os valores de máximo se distanciam muito da média, tal comportamento é causado devido a estes campos selecionados estarem relacionados a geração de erros no sistema. Isso significa que em momento normais eles tendem a serem

Tabela 4.2: Análise descritiva dos Dados 2.

	ACAO44	ACAO46	ACAO51	ACAO54	ACAO56	ACAO62	ACAO82	ACAO84	ACAO97
count	477504.00	477504.00	477504.00	477504.00	477504.00	477504.00	477504.00	477504.00	477504.00
mean	233.74	181.28	101.68	141.61	100.46	92.56	41.86	31.38	20.80
std	2644.79	1652.73	133.67	1486.06	294.77	296.18	74.11	60.57	77.69
min	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
25%	32.00	27.00	27.00	20.00	25.00	23.00	9.00	10.00	2.00
50%	118.00	99.00	87.00	73.00	80.00	76.00	32.00	24.00	15.00
75%	227.00	189.00	157.00	140.00	137.00	124.00	63.00	37.00	30.00
max	160151.00	81222.00	6405.00	81624.00	16051.00	16192.00	7040.00	3627.00	5559.00

baixos, porém, em momentos de indisponibilidades eles tendem a uma elevação acentuada. Esse comportamento pode ser visualizado na Figura 4.3, onde são apresentados os histogramas para 20 atributos selecionados aleatoriamente, onde o eixo Y foi plotado na escala logarítmica para facilitar a visualização das distribuições.

A Figura 4.3 revela outro aspecto importante: a escala dos atributos apresenta um alto grau de variabilidade. Enquanto o atributo TX_ACAO_EVT_194 possui um valor máximo próximo a 1500, o atributo TX_CMT_URL_ACSD_5 apresenta um valor máximo próximo a 500.000, o que representa uma diferença de mais de 3 mil vezes. Essa grande variação pode dificultar o processamento de certos algoritmos de aprendizado de máquina, o que indica a necessidade de aplicar técnicas de ajuste de escala nos dados.

O conjunto de dados foi explorado a fim de facilitar o seu entendimento e gerar percepções do que pode ser trabalhado. Foi gerada um mapa de calor para a Correlação de Pearson [43] para avaliar como os atributos se relacionam, apresentada na Figura 4.4. Quanto mais claro o valor de cada célula, maior a correlação entre os atributos. Em razão do elevado número de atributos e para facilitar a visualização foi gerada uma amostra das colunas, contendo 20 atributos selecionados de forma aleatória.

Ao analisar a Figura 4.4 é perceptível que uma quantidade significativa de atributos apresenta uma correlação elevada entre si. Contudo, ao examinar a correlação entre o rótulo e os demais atributos, é notório que há uma baixa correlação, com exceção do penúltimo atributo. Com base nessas observações, pode-se inferir que o conjunto de dados possui uma margem considerável para aplicação de técnicas de seleção de atributos, o que possibilita refinar quais atributos serão utilizados nos modelos.

4.3.4 Exploração dos Dados

Em uma indisponibilidade generalizada nos Sistemas de Informação que atendem o aplicativo móvel, o comportamento dos usuários é modificado. Isso significa que em um período

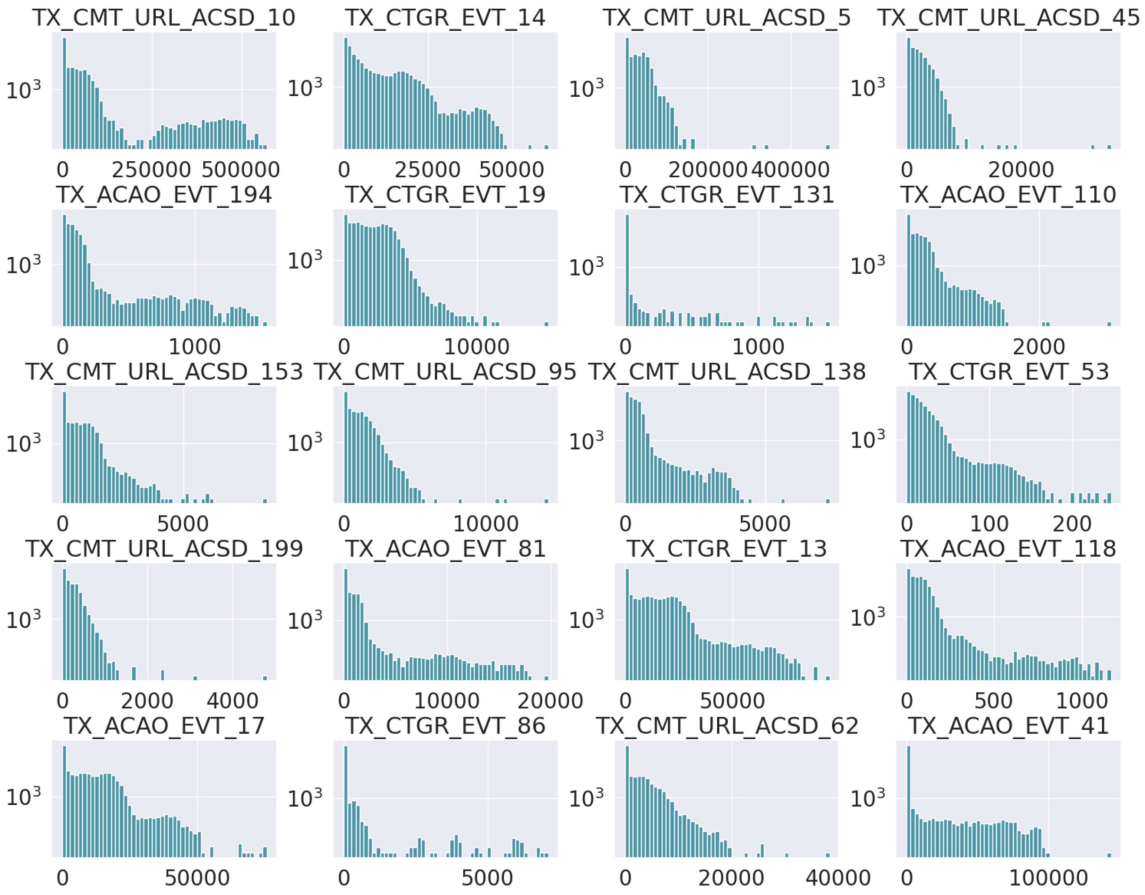

Figura 4.3: Distribuição dos atributos.

de normalidade, os clientes acessam a página de entrada, fazem a autenticação e realizam algumas operações financeiras. Em momentos de anormalidade, os clientes são surpreendidos com mensagens de erro logo ao tentarem se autenticar com o banco. O número de operações financeiras é reduzido, em um momento de crise, para praticamente zero e as páginas que necessitam de uma autenticação prévia ficam com seus acessos reduzidos.

A Figura 4.5 apresenta quatro métricas extraídas da base de dados do Matomo em um período de 800 minutos que contém uma das crises mapeadas na base de incidentes. As duas primeiras métricas indicam a quantidade de ações realizadas pelos clientes relacionadas a duas operações financeiras; a terceira indica a quantidade de ações para uma determinada página do aplicativo; e a quarta indica a quantidade de acessos a uma página de autenticação. Assim, é possível visualizar na Figura 4.5 que próximo ao período de 150 minutos houve uma forte oscilação nas métricas e, posteriormente, uma queda grande nos seus valores, até o período de 550 minutos quando a situação foi normalizada. Nas Figuras 4.5 e 4.6 os valores estão escalados para facilitar a visualização devido a variabilidade de cada métrica. Os valores foram escalados utilizando a técnica de padronização, conforme descrito na Seção 2.2.4.

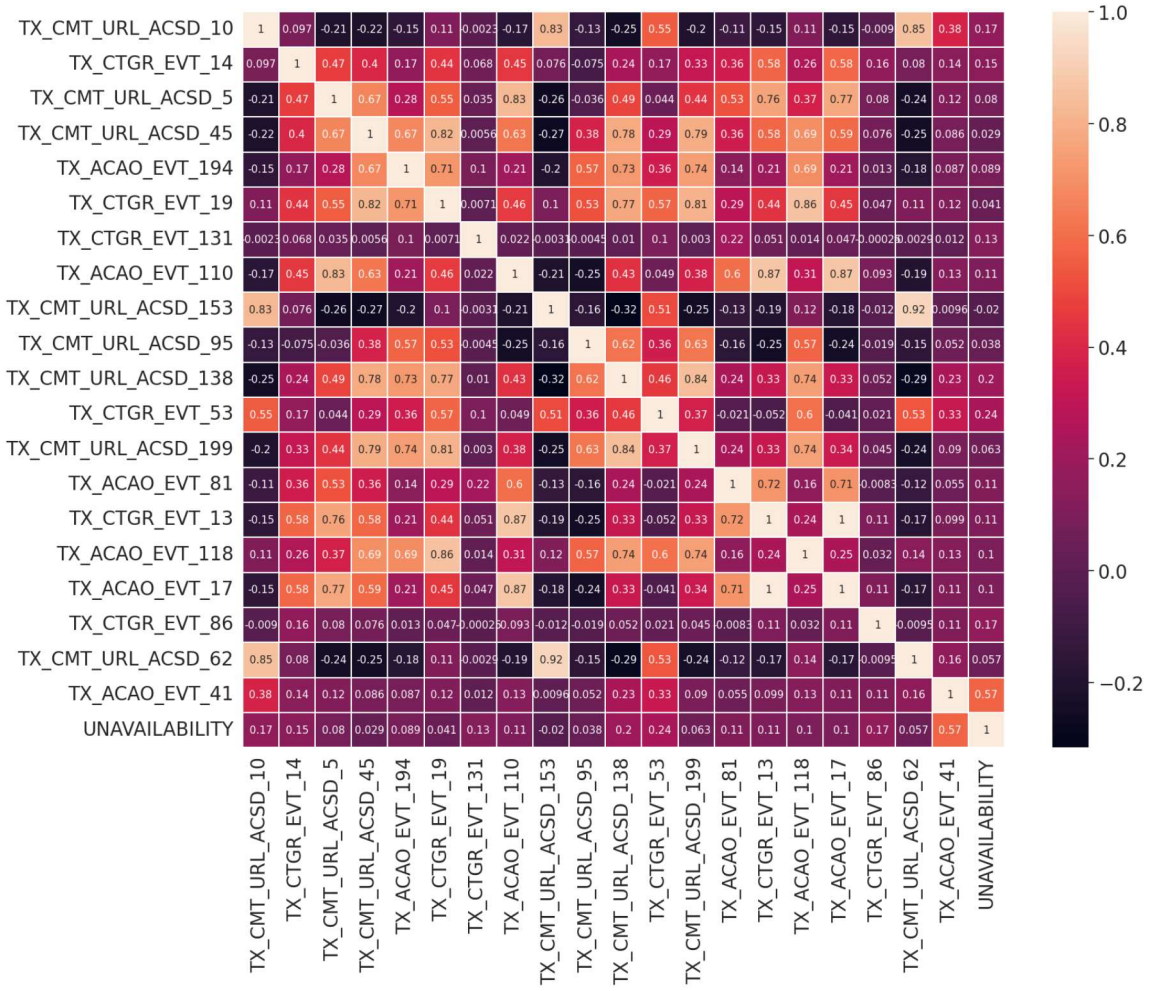

Figura 4.4: Correlação de atributos com a indisponibilidade.

Na Figura 4.6 são apresentadas três métricas que têm seu valor incrementado no momento de uma indisponibilidade. As Figuras 4.5 e 4.6 foram separadas com o objetivo de facilitar a leitura devido a escala das métricas. As métricas apresentadas na Figura 4.6 são referentes ao tempo de resposta de uma determinada operação, quantidade de eventos categorizados como erro de transação e a quantidade de eventos categorizados como erros. Também é possível visualizar que um pouco depois do minuto 150, a quantidade de eventos da categoria erro, e erro de transação sobem consideravelmente, assim como o tempo de resposta da operação.

4.3.5 Qualidade dos Dados

A avaliação da qualidade dos dados é um passo fundamental em qualquer projeto de análise e modelagem, uma vez que a precisão e a confiabilidade das conclusões dependem diretamente da integridade dos dados utilizados. Nesta seção serão exploradas a

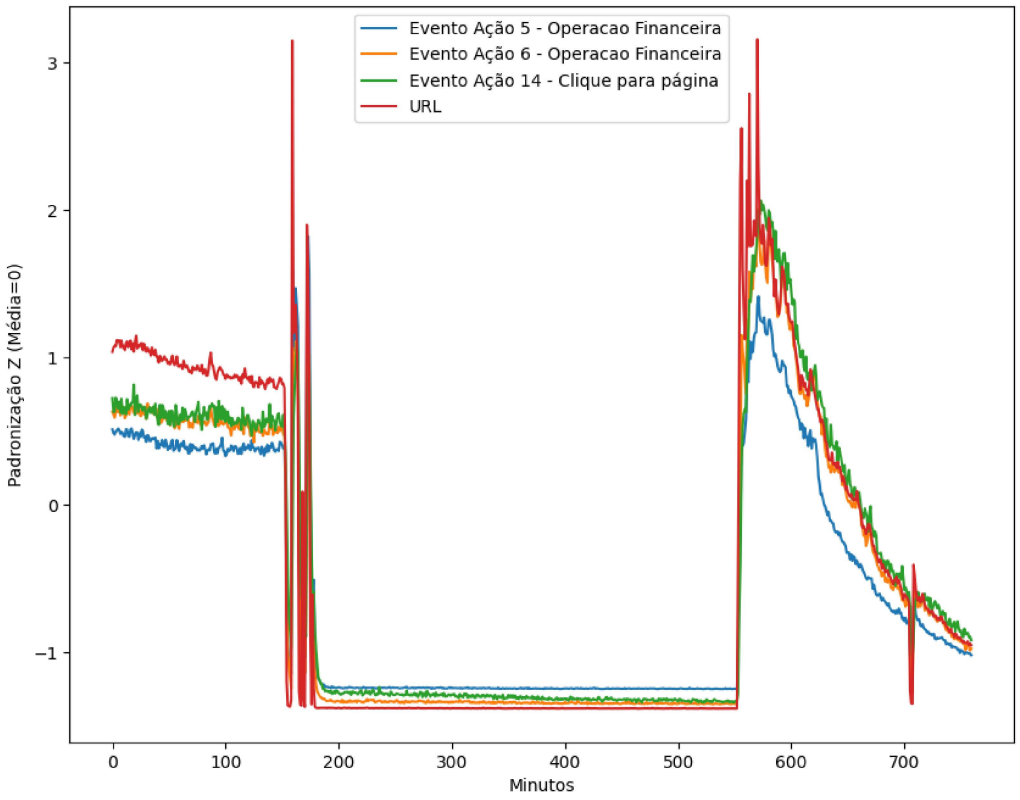

Figura 4.5: Operações financeiras em uma indisponibilidade.

qualidade dos dados coletados para este estudo, os métodos de extração empregados e as considerações feitas em relação à valores faltantes, nulos e inconsistentes.

Devido à abordagem de coleta de dados por meio de consultas SQL no Apache Hive, os dados adquiridos foram isentos de valores nulos ou faltantes. O processo de consulta resultava na obtenção de valores específicos em um dos três campos selecionados, o que culminava em uma saída numérica, variando como números inteiros positivos. Como parte dos testes de integridade realizados, foi verificado se algum atributo continha exclusivamente valores zerados, contudo, essa circunstância não foi encontrada.

No entanto, foi notado que em ocasiões excepcionais, devido a falhas na aplicação de *Web Analytics*, os dados coletados registraram valores zerados. Contudo, esses valores ausentes não se correlacionaram com períodos de indisponibilidade do aplicativo móvel e, dada a sua insignificante proporção, não será implementada qualquer abordagem de tratamento específico.

4.4 Preparação dos Dados

Nesta etapa serão realizadas as tarefas para construir o conjunto final de dados. Tais tarefas incluem a integração de dados de outras tabelas, a construção de novas colunas,

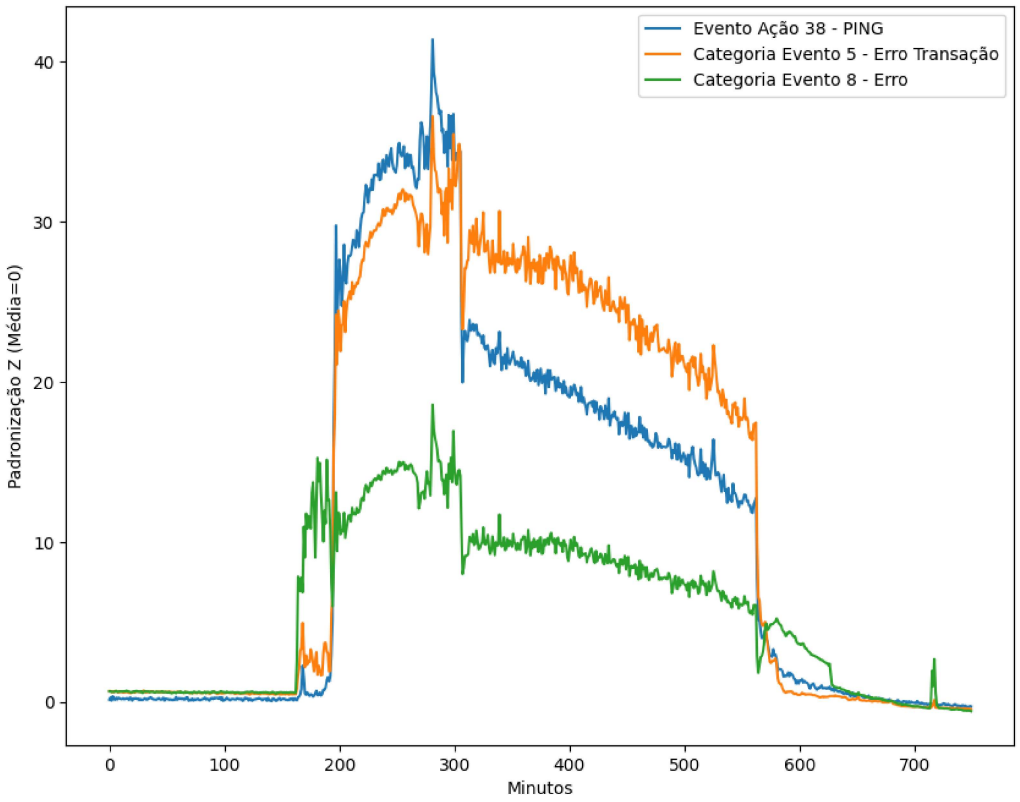

Figura 4.6: Erros e tempo de resposta em uma indisponibilidade.

a normalização dos dados e transformações em geral, que serão descritas nas próximas seções.

4.4.1 Construção e Integração dos Dados

A rotulação dos dados é um processo fundamental no treinamento de modelos de Aprendizado Supervisionado. Tal etapa consiste na classificação manual do conjunto de dados no atributo alvo que se pretende atingir. Porém, a informação da disponibilidade do serviço não se encontra na ferramenta de *Web Analytics*. Dessa forma, os incidentes da organização e os períodos que houveram reclamações generalizadas dos clientes referentes a problemas no acesso ao aplicativo móvel bancário foram pesquisados na base.

Após a identificação dos incidentes relacionados ao serviço, foram extraídos dos seus registros as datas e os horários de indisponibilidades reportadas pelos clientes até o momento da aplicação de uma solução, e a validação do seu restabelecimento. Em seguida, obtidos os períodos de indisponibilidades no ambiente, foi criado um campo binário no conjunto de dados referente a disponibilidade, conforme ilustrado na Figura 4.7. Assim, para cada minuto no qual foram extraídos as ocorrências das páginas, eventos e categorias, foi rotulado o estado do serviço como disponível e não disponível, naquele momento.

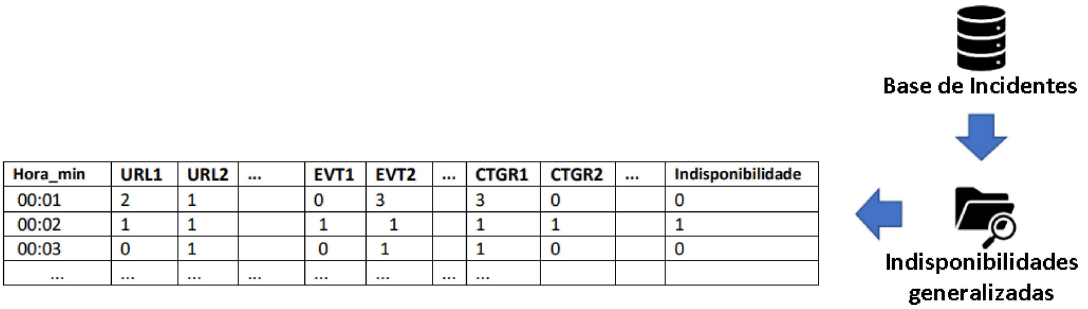

Figura 4.7: Rotulação a partir da base de incidentes

Após a construção do campo de indisponibilidade, esse foi integrado ao conjunto de dados da ferramenta de *Web Analytics*. Tal integração permitirá que os modelos de classificação aprendam os padrões contidos no conjunto de dados de disponibilidade e no conjunto de indisponibilidades. Possibilitando a classificação de novos dados não rotulados no futuro pelo classificador.

4.4.2 Padronização

Ajustar a escala dos dados é uma etapa essencial no processo de preparação dos dados, permitindo que variáveis com diferentes escalas numéricas sejam comparáveis e influenciem igualmente na análise. Ao aplicar esse ajuste, a análise de padrões, relações e tendências se torna mais confiável, evitando distorções provocadas por variações extremas nos valores. Dentre as técnicas de ajuste de escala mais comuns estão a normalização, a padronização e a transformação linear.

A padronização é mais eficiente em distribuições normais. No entanto, a maioria dos dados capturados possuem a distribuição assimétrica à direita. Assim, foram comparadas técnicas de ajuste de escala no conjunto de dados, como a transformação linear e a normalização em testes preliminares, no qual a padronização apresentou melhores resultados.

A padronização foi implementada por meio do método StandardScaler da biblioteca Scikt Learn [125], que aplica a transformação dos dados em cada atributo separadamente. O processo de padronização é feito em duas etapas, sendo que na primeira um modelo é treinado, utilizando apenas os dados de treinamento. Na segunda etapa, os valores de treinamento e teste são transformados de acordo com o modelo construído na primeira etapa. Dessa forma, quando aplicada, todas as variáveis numéricas do conjunto de dados são escaladas.

4.4.3 Seleção de Atributos

Inicialmente, foram selecionadas 600 atributos a partir da base de dados da ferramenta de *Web Analytics*. Tais atributos representam as interações mais frequentes dos usuários com o aplicativo móvel. No entanto, nem todas os atributos possuem uma correlação significativa com a disponibilidade do serviço, e podem ter um impacto negativo no processo de classificação. Dessa forma, foi necessário aplicar técnicas de seleção de atributos para melhorar o desempenho dos modelos de Aprendizado de Máquina.

A seleção de atributos foi realizada em duas fases distintas. Primeiramente, devido à extensão do conjunto de dados e às complexidades associadas ao processamento integral, optou-se por uma redução inicial pela metade, resultando em 300 atributos de maior relevância. Em seguida, implementou-se um processo de otimização para determinar o número ótimo de atributos. Dessa forma, os modelos de classificação foram alimentados com os 300 atributos previamente selecionados ou com uma seleção ainda mais criteriosa. A Figura 4.8 apresenta os *scores* dos 300 atributos gerados pela função *selectKBest*.

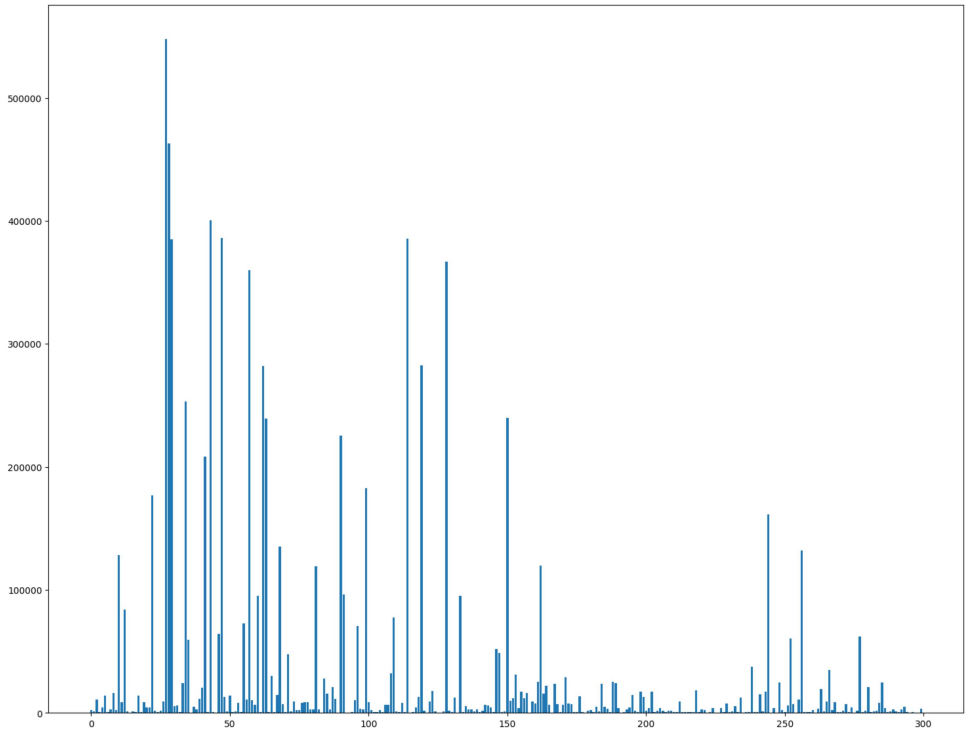

Figura 4.8: *Scores* dos atributos.

Para encontrar o número ideal de atributos, criou-se uma função de otimização utilizando a biblioteca *Hyperopt* [126]. O objetivo dessa função é maximizar o valor do F1-Score, aplicando o algoritmo SVM nos dados de treinamento. Definiu-se um espaço de busca de 10% a 50% da quantidade de atributos. Verificou-se em testes preliminares que com menos de 10% poderiam ocorrer casos de sobre-ajuste e com mais de 50% o valor

da função de otimização apenas reduzia. Novamente para evitar o sobre-ajuste dos dados, dessa vez em um número específico de atributos, os resultados foram suavizados por meio da média entre os três valores mais próximos. Assim, o melhor valor encontrado em 100 tentativas no conjunto de treinamento e validação foi de 87 atributos. Os atributos selecionados são descritos em detalhes nas Tabelas em anexo I.1, I.2, I.3.

A Figura 4.9 visualiza os resultados alcançados na etapa de otimização de atributos. Cada ponto na figura representa o valor do *F1-Score* correspondente a um número específico de atributos selecionados. É evidente que, após atingir 120 atributos, ocorre uma acentuada redução no valor do F1-Score, sugerindo que um excesso de atributos pode degradar o desempenho dos classificadores. De maneira geral, os resultados permanecem consistentes, exceto por uma discrepância no intervalo de 90 a 100 atributos.

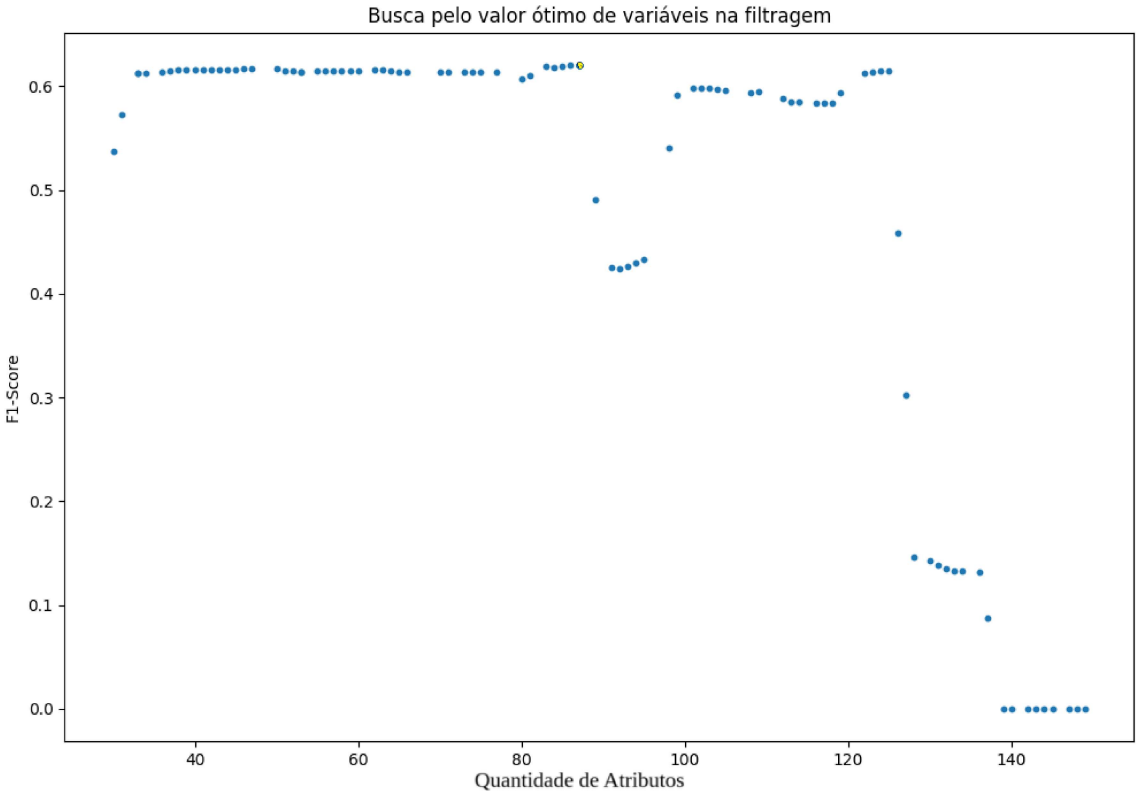

Figura 4.9: Otimização da seleção de atributos.

4.4.4 SMOTE

Em termos gerais, falhas representam eventos raros no âmbito da Tecnologia da Informação, em alguns cenários chegando a serem classificadas como anomalias. Essa particularidade resulta em conjuntos de dados altamente desequilibrados, especialmente quando se trata de dados reais. Essa disparidade de proporções pode comprometer a eficácia de

determinados algoritmos de classificação, tornando essencial a exploração de estratégias de balanceamento. Entre as abordagens disponíveis, uma que vem recebendo notável atenção é a técnica de geração de amostras sintéticas conhecida como SMOTE [49].

A proposta do uso do SMOTE reside em sua aplicação no conjunto de dados de treinamento, com o intuito de ampliar as instâncias das indisponibilidades. Essas amostras são sinteticamente geradas com base nos vizinhos da classe de indisponibilidade. Assim, por meio dessa abordagem evita-se a perda de dados, como ocorre na eliminação de exemplos da classe majoritária, e também a criação de registros duplicados, uma situação que pode ocorrer em outras táticas de equilíbrio.

A biblioteca *Imblearn* [127] foi empregada para a implementação do SMOTE. Para determinar a proporção mais eficaz de sobre-amostragem, mais uma vez recorreu-se ao Hyperopt. Nesse contexto, o objetivo da função no Hyperopt foi otimizar o *F1-Score* máximo, explorando diversas variações de sobre-amostragem do SMOTE. Novamente, a implementação contou com o SVM, escolha fundamentada em resultados positivos obtidos em avaliações iniciais.

Um espaço de busca foi delimitado, variando da proporção original de indisponibilidades até 50%, correspondendo a um equilíbrio entre as classes. Após 100 tentativas, o resultado mais notável atingiu 0.466%. Esse valor representa um aumento de aproximadamente 20% na sobre-amostragem, quando comparado ao índice inicial de 0.388% presente no conjunto de dados.

A Figura 4.10 ilustra os resultados obtidos durante a etapa de otimização da técnica de sobre-amostragem. A figura apresenta os valores do *F1-Score* que foram avaliados dentro do intervalo de busca definido. É notável que as performances mais promissoras são observadas em níveis moderados de sobre-amostragem, com um aumento leve notado a partir de um acréscimo de 0.05%. No entanto, um avanço significativo não é observado até que um equilíbrio de 50% seja alcançado.

4.5 Modelagem

Nesta seção são apresentados os detalhes da construção do plano de execução do experimento, no qual são abordados a estratégia de separação dos dados, os algoritmos escolhidos para classificação, e as métricas utilizadas na avaliação.

4.5.1 Estratégia de Separação dos Dados

Durante o estudo, foram investigadas diferentes abordagens para a separação dos dados, com o objetivo de encontrar a estratégia mais adequada. Inicialmente, optou-se por realizar a separação dos dados utilizando embaralhamento, buscando uma estratificação

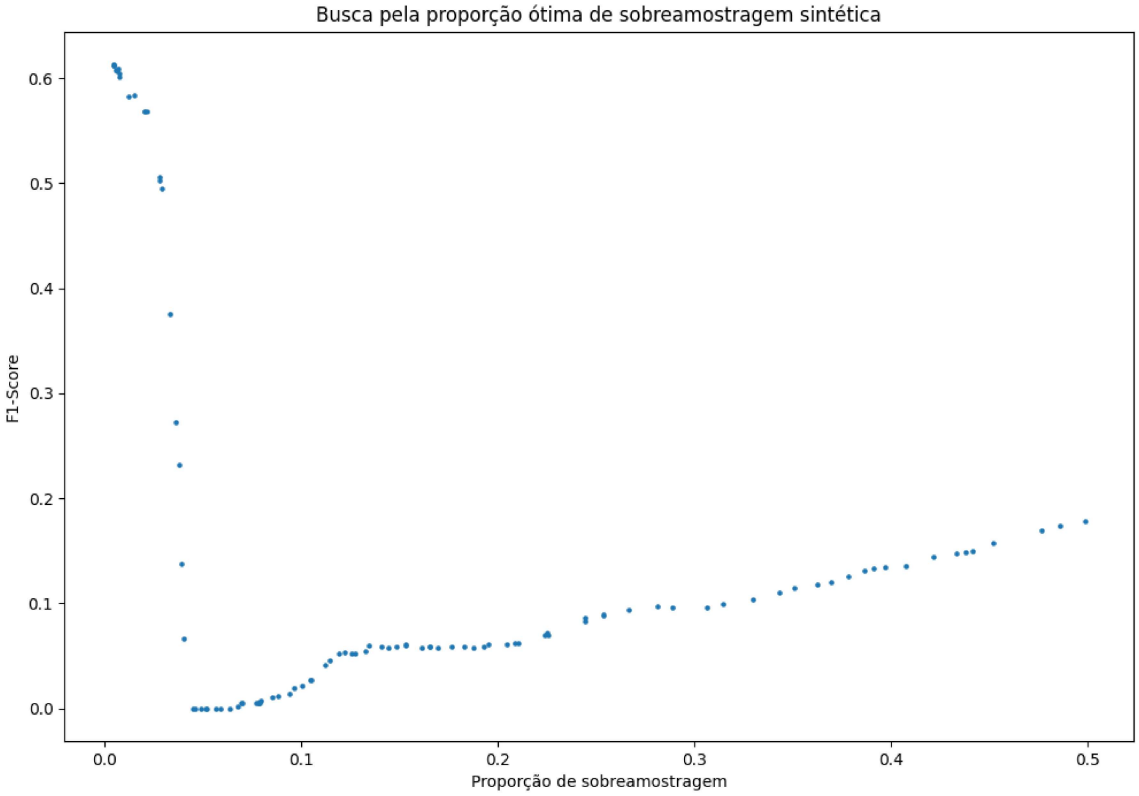

Figura 4.10: Otimização da proporção de sobre-amostragem.

equilibrada das classes em cada conjunto. No entanto, observou-se que essa abordagem resultava em sobreajuste (overfitting) nos modelos.

Ao analisar mais detalhadamente, constatou-se que os modelos estavam aprendendo as indisponibilidades de forma isolada, tornando difícil a generalização do problema para outros cenários. Além disso, variáveis com baixa correlação com o rótulo estavam recebendo uma importância desproporcional, o que levava os modelos a identificar apenas os períodos de crise individualmente.

Diante dessas observações, decidiu-se descartar o embaralhamento e adotar a ordem cronológica dos dados como estratégia de separação. Essa abordagem permitiu considerar a evolução temporal das informações, gerando resultados mais coerentes e facilitando a generalização dos modelos para diferentes cenários.

Além disso, foram conduzidos testes utilizando a estratégia de validação cruzada. No entanto, durante essa avaliação identificou-se que manter a ordem cronológica dos dados conferia uma vantagem adicional. O uso da sequência temporal dos dados contribuiu para que os modelos fossem expostos a padrões mais próximos àqueles encontrados na fase de teste. Isso proporcionou uma melhor adaptação dos modelos aos dados reais, produzindo resultados mais confiáveis.

Assim, os dados foram divididos em uma proporção de 25% para teste e 75% para

treinamento. No conjunto de treinamento os dados foram novamente divididos na mesma proporção de 25% para validação e 75% para treinamento. Essa estratégia permitiu explorar diferentes combinações de hiperparâmetros sem viés nos resultados decorrente de validações repetidas.

4.5.2 Seleção de Algoritmos

Para avaliar o conjunto de dados e as transformações realizadas, foram selecionados alguns dos algoritmos mais comuns para a tarefa de classificação binária supervisionada. Essa seleção incluiu um algoritmo probabilístico, um não paramétrico, uma rede neural profunda, dois métodos compostos e dois algoritmos comuns. Essa diversidade de escolhas permitiu uma avaliação abrangente de diferentes estratégias de Aprendizado de Máquina na detecção de falhas no conjunto de dados construído. Ao final do experimento os algoritmos selecionados serão comparados e o que obtiver o melhor resultado será utilizado na fase de implantação do projeto em um ambiente real.

Dessa forma, foi adotada uma variedade de algoritmos para representar diferentes abordagens de Aprendizado de Máquina. O Bayes ingênuo (NB) foi escolhido como um modelo probabilístico, enquanto o K-vizinhos mais próximos (KNN) foi selecionado como um algoritmo não paramétrico. A Árvore de Decisão e a Máquina de Vetores de Suporte (SVM) foram escolhidas como representantes de algoritmos de classificação convencionais. A Floresta Aleatória (RF) foi escolhida por ser um algoritmo composto amplamente utilizado, e o *Adaboost* foi selecionado devido ao seu bom desempenho em problemas de classificação binária e desbalanceamento de classes. Por fim, as redes LSTM foram escolhidas por sua capacidade de reconhecer eventos raros e por serem uma representação das redes neurais.

Os algoritmos Bayes ingênuo, Árvore de Decisão e *Adaboost* foram implementados utilizando a biblioteca Scikit-Learn. Para os algoritmos K-vizinhos mais próximos, *Support Vector Machine* e Florestas Aleatórias, optou-se pela biblioteca cuML [128]. Essa decisão foi motivada pelo fato de que a biblioteca cuML permite o treinamento e predição dos modelos em placas gráficas, proporcionando uma redução significativa no tempo de execução dos experimentos. Além disso, para a implementação do algoritmo LSTM, escolheu-se a biblioteca Keras [129], também aproveitando o potencial de processamento das placas gráficas.

4.5.3 Métricas de Avaliação

A escolha de métricas de avaliação é essencial para medir o desempenho de cada modelo. A acurácia é uma métrica geral do modelo no qual os acertos são contabilizados indepen-

dentes das classes. Porém, em um conjunto de dados desbalanceados é necessário avaliar quanto um algoritmo conseguiu classificar em cada classe. O *F1-Score* se apresenta como uma métrica adequada, pois leva em conta o peso de cada classe em seu cálculo.

Na monitoração de falhas, a classificação de um falso positivo pode ser tão prejudicial quanto a classificação de falsos negativos. Um alarme falso pode encadear um conjunto de processos para restabelecer o serviço. Tais processos incluem a reciclagem do ambiente com o reinicio de serviços críticos, que em períodos de normalidade geram indisponibilidades aos clientes. Assim, o *F1-Score* se apresenta como a medida adequada que atende aos requisitos do negócio para avaliar os resultados dos modelos de aprendizado.

A área sob a curva ROC (ROC AUC) é uma medida muito utilizada em problemas de detecção de anomalias. Tal métrica possibilita avaliar o quão bem um algoritmo conseguiu diferenciar cada classe. Assim, essa métrica será uma importante aliada para avaliar os modelos na classificação binária, como métrica secundária em critério de desempate ou quando dois modelos obtiverem um *F1-Score* muito próximo.

4.5.4 Otimização de Hiperparâmetros

Na etapa de otimização dos hiperparâmetros, foi realizado um processo iterativo para encontrar a combinação ideal de hiperparâmetros para os modelos de classificação no conjunto de dados avaliado. Primeiramente, foi criado um espaço de busca, incluindo os hiperparâmetros e as etapas de pré-processamento relevantes. Em seguida, a métrica de desempenho selecionada foi o *F1-Score*. Os experimentos foram conduzidos seguindo a estratégia de separação, os conjuntos de treinamento e validação foram utilizados, respeitando a ordem cronológica dos dados. Por fim, todas as iterações foram registradas utilizando a ferramenta MLFlow [130] para extrair as métricas do experimento.

Dentre os algoritmos selecionados, alguns possuem performance melhor de acordo com determinada etapa de pré-processamento e outros podem ter uma performance reduzida. Dessa forma, consideraremos as três etapas de pré-processamento e suas combinações ao buscar o melhor modelo no conjunto de treinamento para cada algoritmo otimizado durante a etapa de ajuste. No entanto, a rede LSTM será uma exceção, pois não aplicamos o método SMOTE devido à necessidade de manter uma sequência temporal consistente.

Os hiperparâmetros selecionados para cada algoritmo foram definidos no espaço de busca, que é apresentado na Tabela 4.3. Vale ressaltar que a etapa de pré-processamento foi incluída em todos os algoritmos para facilitar a visualização e evitar repetições desnecessárias. Os principais hiperparâmetros foram cuidadosamente escolhidos com o objetivo de otimizar os modelos para o problema específico e a métrica definida. Essa abordagem permite obter a melhor combinação de técnicas de pré-processamento e modelos otimizados para a classificação das indisponibilidades.

Tabela 4.3: Espaço de busca.

Classificador	**Hiperparâmetros**
ADA	Número de Estimadores: [50, 25, 100],
	Taxa de Aprendizado: [1, 0.5, 2]
DT	Critério: [gini, entropy, log_loss]
KNN	Número de Vizinhos: [1, 3, 5, 7, 9],
	Métrica de Distancia: [euclidean, manhattan, minkowski]
LSTM	Dropout [True, False], Unidades[512:1024]
	Taxa de Aprendizado [1e-5:6e-3]
NB	Suavização de variância: [1e-9, 1e-5, 1e-20]
RF	Número de Estimadaores: [25, 50, 100, 200, 500, 1000]
SVM	C: [0.01, 0.1, 0.5, 1, 10, 100], Núcleo: [rbf, sigmoid]

O Hyperopt foi escolhido como a biblioteca de otimização devido aos seus algoritmos de busca, que vão além da busca aleatória ou em grade, incluindo o TPE (Tree-structured Parzen Estimator). O TPE utiliza estimadores de Parzen adaptativos, estruturados em uma árvore. Essa abordagem suporta não apenas variáveis contínuas, mas também variáveis categóricas, discretas e condicionais, o que pode ser desafiador em outras estratégias de otimização Bayesiana, como o Kriging. Em resumo, o TPE gera uma estimativa da probabilidade de um conjunto de hiperparâmetros e prioriza conjuntos considerados mais promissores.

Considerando o tamanho do espaço de busca, as bibliotecas utilizadas e os tempos de treinamento e predição de cada algoritmo, os espaços de busca foram divididos em três grupos distintos. No primeiro grupo, encontram-se os algoritmos da biblioteca Sklearn, como Adaboost, Árvores de Decisão e Bayes ingênuo, com um número fixo de 50 tentativas definidas. Para o segundo grupo da biblioteca cuML, que inclui o K-Nearest Neighbor, Support Vector Machine e Random Forest, foram alocadas 500 tentativas, aproveitando a velocidade de processamento das placas gráficas. Por fim, o último grupo é composto exclusivamente pela LSTM, que passou por uma otimização de hiperparâmetros com 30 tentativas. A arquitetura final da rede LSTM é apresentada na tabela 4.4, composta por duas camadas LSTMs, uma camada de dropout para evitar o overfitting, uma camada densa e uma camada de saída.

Cada iteração executada pelo Hyperopt é registrada no MLflow, uma ferramenta de tracking de aprendizado de máquina, que armazena os parâmetros, modelos, tempo de execução e as métricas de resultados. Essa funcionalidade permite uma análise detalhada dos resultados de cada etapa do processo, fornecendo percepções valiosas para melhorias futuras.

Os resultados da fase de otimização dos hiperparâmetros para cada algoritmo estão detalhados na Tabela 4.5. Nessa tabela, não apenas os hiperparâmetros mais eficazes são

Tabela 4.4: Arquitetura da rede neural.

Camada	Função de Ativação	Quantidade de Parâmetros
LSTM	ReLU	2841600
LSTM	ReLU	5123200
Dropout (20%)		0
Densa	ReLU	640800
Densa	Sigmoid	1602
Total Parameters: 8,607,202		

exibidos, mas também as combinações de pré-processamento que demonstraram um desempenho superior para cada algoritmo. É relevante ressaltar a diversidade de técnicas de pré-processamento empregadas para cada algoritmo, variando desde a ausência completa de aplicação em alguns até a utilização de todas em outros. Essa observação enfatiza a necessidade intrínseca de avaliar cada técnica em conjunto com seu classificador específico, dada a possibilidade de variações nos resultados entre diferentes classificadores.

Tabela 4.5: Melhores hiperparâmetros.

Classificador	**Preprocessamento**	**Hiperparâmetros**
ADA	Padronização e SMOTE	Número de Estimadores: 50 Taxa de Aprendizado: 1
DT	Filtragem e SMOTE	Critério: *entropy*
KNN	Todas	Número de Vizinhos: 1 Métrica de Distancia: Manhattan
LSTM	Filtragem	Taxa de Aprendizado: 0.000569 Unidades: 768, Lotes: 2048, Épocas 50
NB	Nenhuma	Suavização de variância: 1e-5
RF	Nenhuma	Número de Estimadaores: 100
SVM	Filtragem	C: 1, Núcleo: rbf

4.6 Considerações Finais

Este capítulo introduziu uma proposta para a detecção de falhas em Sistemas de Informação. Inicialmente, delineou-se a arquitetura da proposta e a organização geral do projeto. Em seguida, delinearam-se as necessidades do negócio e como estas foram traduzidas em um problema de mineração de dados. A descrição detalhada dos dados disponíveis permitiu a exploração e a formulação de hipóteses para as etapas subsequentes. Além disso,

investigaram-se estratégias de transformação dos dados, preparando-os adequadamente para os algoritmos de aprendizado de máquina.

Durante a fase de modelagem, foi abordada a estratégia de divisão dos dados, bem como a seleção dos algoritmos. Uma ampla gama de algoritmos foi escolhida para avaliar qual deles demonstra o desempenho mais eficaz. Por fim, examinaram-se as métricas de avaliação dos modelos e a otimização dos hiperparâmetros que influenciam o desempenho dos modelos.

No capítulo subsequente, toda a estratégia elaborada será efetivamente aplicada. Nesse sentido, serão compartilhados os resultados alcançados nos conjuntos de validação e teste. Adicionalmente, os resultados serão analisados em detalhes e interpretados para proporcionar uma compreensão dos mesmos.

Capítulo 5

Resultados Obtidos

5.1 Avaliação

Nesta seção foram avaliados os resultados obtidos na etapa de modelagem e otimização de hiperparâmetros. Em seguida, os modelos foram testados no conjunto de dados de teste e sua performance foi comparada com base nas métricas de interesse previamente definidas. Por fim, os resultados foram analisados para verificar se os objetivos do negócio foram alcançados.

5.1.1 Experimentos e Resultados no Conjunto de Validação

A Figura 5.1 apresenta inicialmente um gráfico de dispersão com as 580 iterações do Hyperopt. Os valores do F1-Score para cada execução são exibidos no *eixo y*, enquanto as iterações são apresentadas no *eixo x*, conforme sua evolução. Devido à execução separada em três grupos, de acordo com a biblioteca de implementação, é possível observar a rede LSTM no canto esquerdo, os algoritmos da biblioteca CuML no centro e, por fim, os algoritmos implementados com o Scikit-Learn à direita.

Ao analisar a Figura 5.1, torna-se evidente que os ajustes dos hiperparâmetros podem alterar consideravelmente os resultados para a maioria, se não todos, os algoritmos, destacando a necessidade desta etapa. Observa-se que, devido à priorização dos melhores resultados pelo Hyperopt, alguns algoritmos dentro de um determinado grupo foram executados em maior quantidade do que outros, como é o caso do KNN e do Adaboost. No caso do algoritmo KNN, é possível visualizar que várias tentativas alcançam valores superiores a 0.65, mas não conseguem ultrapassar o valor de 0.7, indicando uma limitação nos resultados para este algoritmo.

A Figura 5.2 apresenta uma comparação dos valores do F1-Score obtidos na etapa de otimização em relação às etapas de pré-processamento. Ela fornece uma visualização da

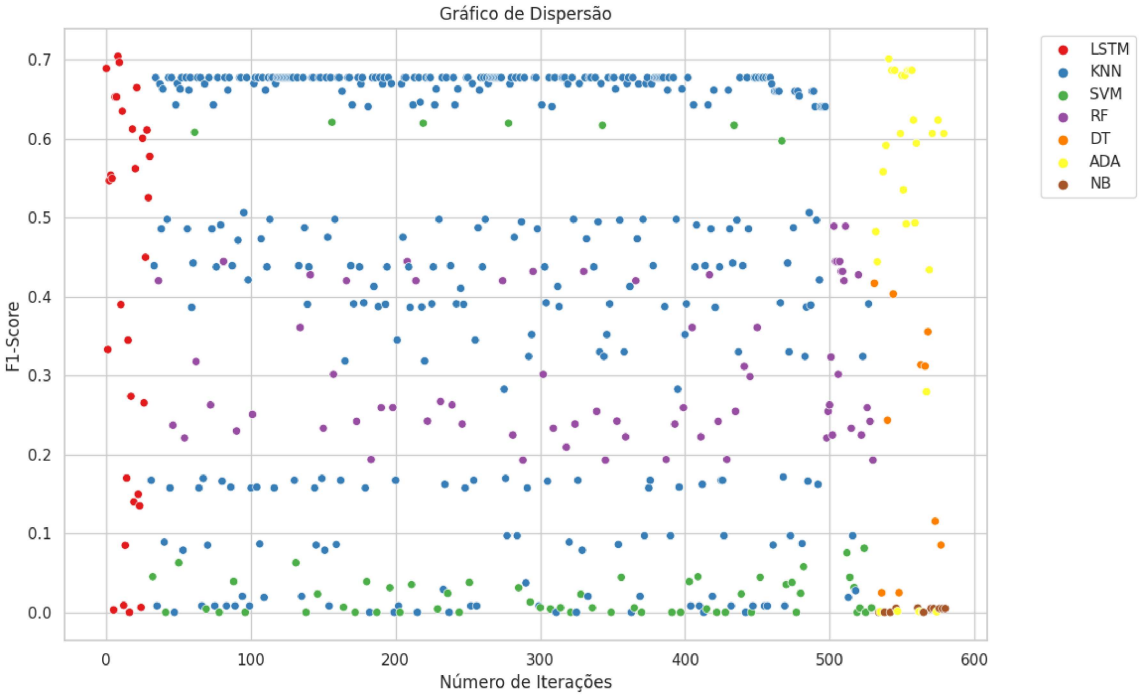

Figura 5.1: Gráfico de dispersão dos modelos em Treino/Validação.

performance de cada algoritmo em relação às técnicas de engenharia de atributos aplicadas. Alguns algoritmos aparecem mais de uma vez para cada etapa devido à avaliação de diferentes hiperparâmetros.

Ao analisar os resultados da Figura 5.2, podemos observar que os algoritmos que incluem a etapa de filtragem geralmente apresentam resultados melhores em comparação aos demais, com exceção do algoritmo *AdaBoost* com aplicação de padronização e *SMOTE* que obteve um dos melhores resultados. Além disso, nesta etapa de análise, podemos identificar que os algoritmos que se destacaram em termos de performance foram o *Adaboost*, a rede LSTM e o KNN. Os melhores hiperparametros obtidos podem são apresentados na Tabela 4.5.

A Figura 5.3 exibe um gráfico de *boxplot* para os resultados de validação agrupados por algoritmo. Este gráfico oferece uma representação visual clara da variação dos valores de *F1-Score* ao longo da etapa de otimização. Vale ressaltar que a rede LSTM se destaca por apresentar a maior variabilidade, apesar de possuir uma mediana consistente acima de 0.5. Por outro lado, o algoritmo de Bayes Ingênuo exibe a menor variação. Entre os demais algoritmos, o SVM se destaca ao revelar a maioria dos resultados abaixo de 0.1; no entanto, é notável a presença de três *outliers* que ultrapassam 0.6, posicionando-o entre os melhores resultados. Essa particularidade, como mencionado anteriormente, é atribuída à etapa de filtragem, que exerce uma influência significativa neste classificador.

Finalmente, a Figura 5.4 ilustra o tempo médio de treinamento para cada algoritmo.

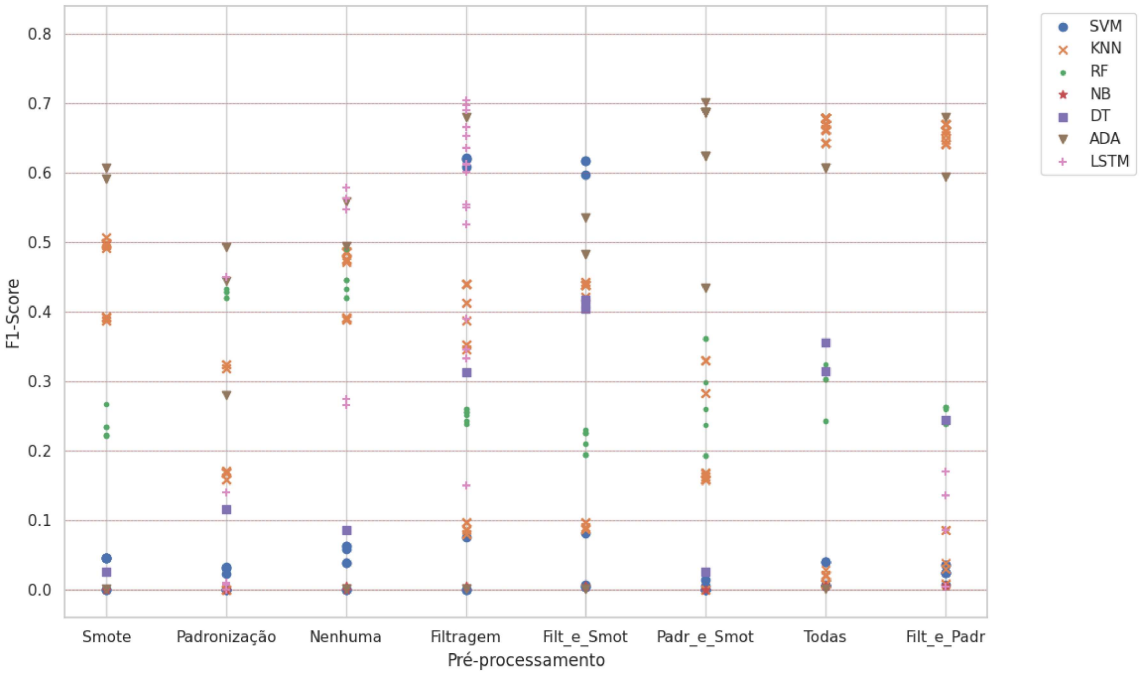

Figura 5.2: Comparação de modelos em Treino/Validação.

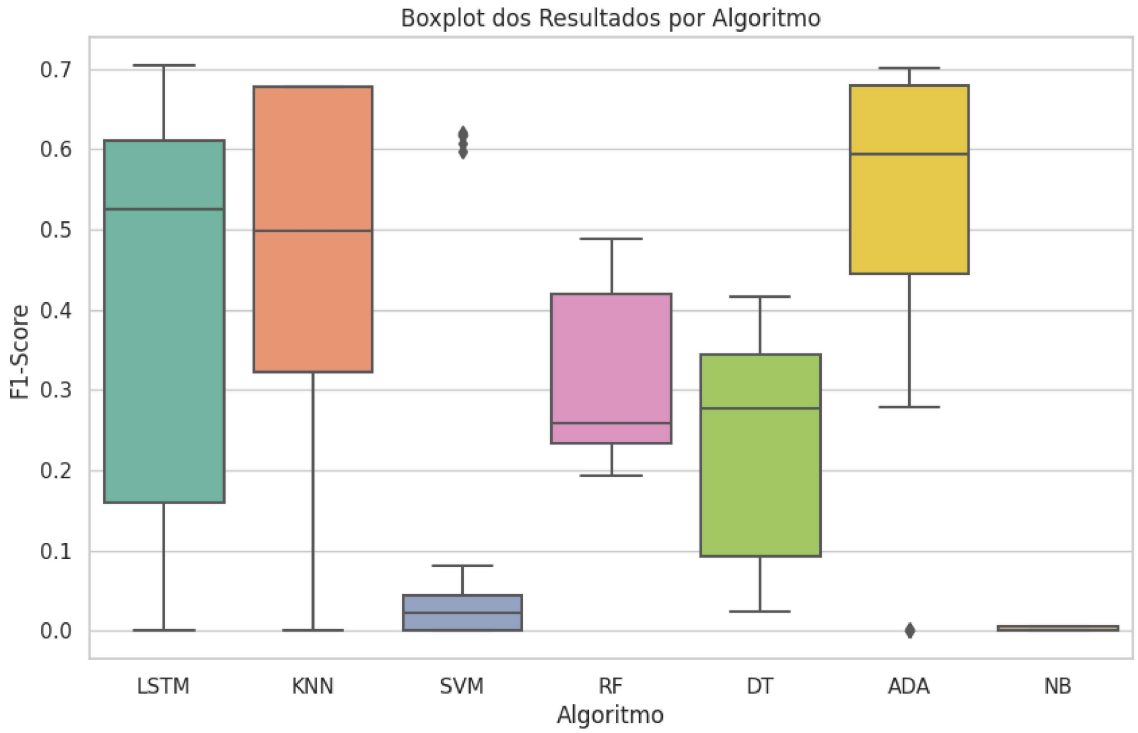

Figura 5.3: Boxplot dos algoritmos em Treino/Validação.

Nesta fase, torna-se evidente o significativo ganho de eficiência nos algoritmos implementados pela biblioteca CuML (SVM, KNN e RF), os quais executam o treinamento em

unidades de processamento gráfico (GPU, do inglês *Graphics Processing Unit*). Devido à disparidade nos valores de tempo, o *eixo y* foi representado em escala logarítmica, proporcionando uma visualização mais equilibrada. No entanto, destaca-se que a rede LSTM, embora também tenha se beneficiado do treinamento em GPU, apresentou o maior tempo de treinamento. Esse fenômeno é atribuído à complexidade de uma Rede Neural Artificial Profunda, composta por diversas camadas ocultas e milhões de parâmetros, conforme detalhado na Tabela 4.4.

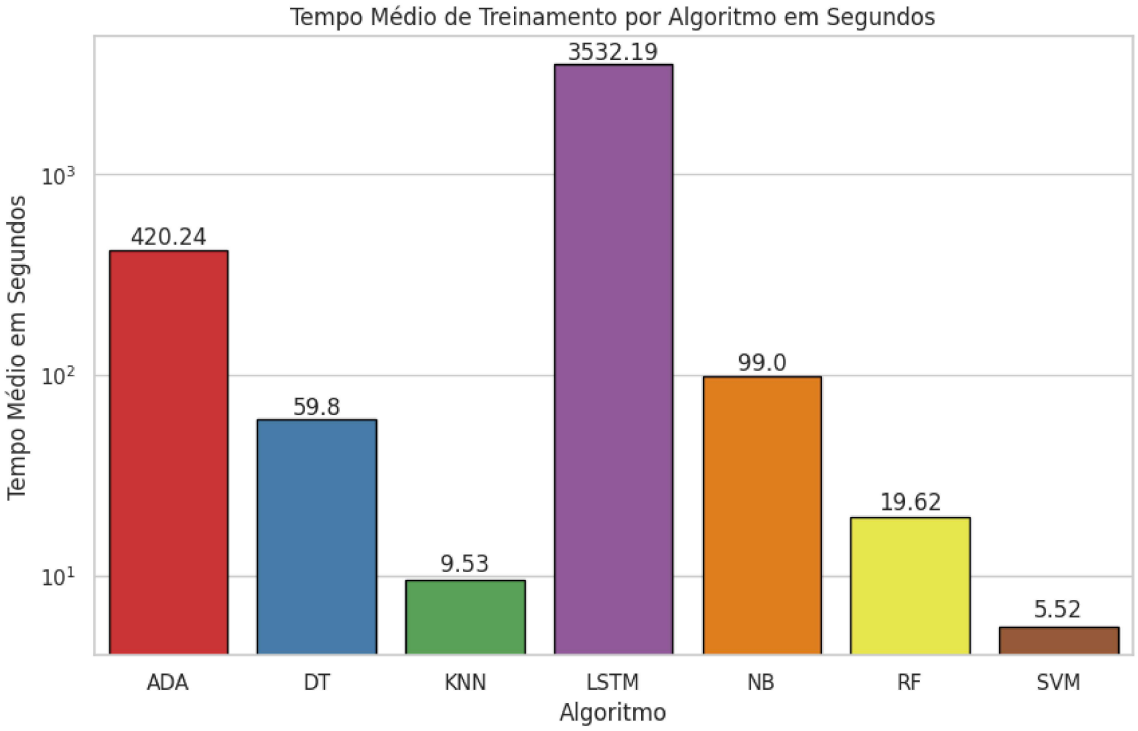

Figura 5.4: Tempo de treinamento em Treino/Validação.

5.1.2 Experimentos e Resultados no Conjunto de Teste

Após identificar os melhores resultados para cada algoritmo, foram registrados os respectivos hiperparâmetros e a estratégia de pré-processamento utilizada. Em seguida, os modelos foram treinados com o conjunto completo de dados de treinamento e submetidos a uma avaliação final no conjunto de testes. Cada algoritmo foi avaliado em 10 amostras diferentes, cada uma contendo 50% dos dados do conjunto de teste, utilizando a mesma semente para garantir a replicabilidade das amostras.

No contexto da rede LSTM, a avaliação ocorreu de maneira incremental para preservar a sequência temporal dos dados. Inicialmente, a primeira avaliação foi conduzida utilizando 60% do conjunto de teste. Nas nove avaliações subsequentes, incrementou-se 4%

dos dados do conjunto de teste em cada iteração, até alcançar a utilização total dos dados disponíveis. Essa estratégia foi adotada com o objetivo específico de manter a consistência temporal dos dados durante a análise do desempenho da rede LSTM.

A Figura 5.5 apresenta uma comparação entre os algoritmos avaliados, em relação a métrica de F1-Score. No geral, os três primeiros algoritmos (SVM, *Adaboost* e LSTM) obtiveram resultados próximos, com seus intervalos de confiança se intersectando. Isso indica que não existem evidências estatisticamente significativas para afirmar que um algoritmo é superior ao outro. Da mesma forma, os algoritmos Random Forest e KNN, que obtiveram o quarto e quinto melhores resultados, também apresentam uma intersecção nos seus intervalos de confiança. Esses resultados mostram que, considerando a métrica F1-Score, esses algoritmos são comparáveis em termos de desempenho.

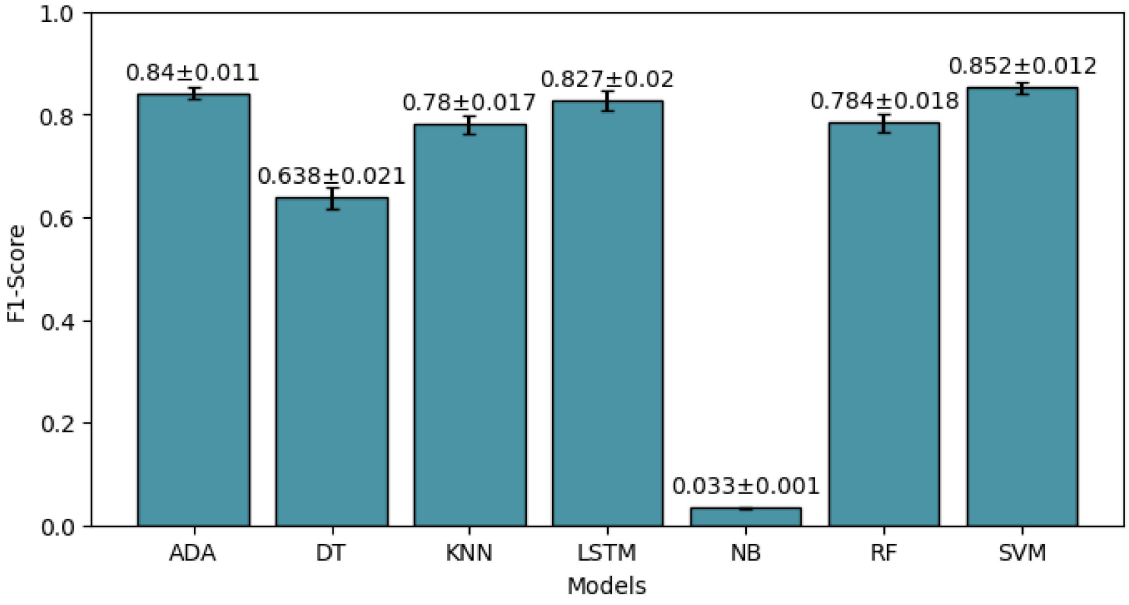

Figura 5.5: Comparação de modelos Treino/Teste.

Os resultados são também exibidos de forma tabular na Tabela 5.1, visando a clareza e compreensão visual. Além das médias e intervalos de confiança, os dados da tabela abrangem informações sobre o desvio padrão e a variância dos resultados. Tais dados proporcionam uma perspectiva mais completa sobre a dispersão e a consistência das medições efetuadas.

Após avaliar o *F1-Score*, comparamos os modelos com as seguintes métricas, Curva ROC, precisão e *recall*. Na Tabela 5.2, são apresentados os resultados de precisão, *recall* e Curva ROC de todos os algoritmos no conjunto de teste. Em relação à métrica de Curva ROC, é possível observar que o AdaBoost e o SVM novamente se encontram próximos, ocupando a segunda posição. À frente deles, em primeiro lugar na curva ROC, há a rede LSTM. A rede LSTM se destaca pelo alto *recall*, indicando a habilidade em diferenciar as

Tabela 5.1: Análise de desempenho: resultados do *F1-Score*.

Modelo	Média	Desvio Padrão	Variância	Intervalo de Confiança
AdaBoost	**0.840091**	0.007534	5.675562e-05	**0.011362**
Árvore de Decisão	0.637963	0.013850	1.918267e-04	0.020887
Floreta Aleatória	0.783764	0.011823	1.397775e-04	0.017830
KNN	0.780082	0.011396	1.298694e-04	0.017186
LSTM	**0.826631**	0.013168	1.733892e-04	**0.019858**
Bayes ingênuo	0.033377	0.000711	5.053078e-07	0.001072
SVM	**0.852374**	0.007858	6.175536e-05	**0.011851**

classes e identificar a maioria das indisponibilidades ocorridas. No entanto, é importante notar que o modelo LSTM gera um número ligeiramente maior de falsos positivos.

Assim como a rede LSTM, o algoritmo Bayes ingênuo demonstrou um alto *recall*, identificando quase todos os casos de indisponibilidades. No entanto, sua precisão muito baixa inviabiliza o seu uso, resultando em um *F1-Score* baixo. Ao analisar os dados relacionados à precisão, o algoritmo Random Forest se destaca, apresentando o melhor resultado devido à baixa ocorrência de falsos positivos. Outros algoritmos, tal como o KNN, também obtiveram um desempenho satisfatório, com ênfase na precisão. Já os algoritmos AdaBoost e SVM mostraram resultados balanceados ao serem avaliados em termos de *recall* e precisão, indicando um compromisso entre a ocorrência de falsos positivos e verdadeiros negativos.

Tabela 5.2: Análise de desempenho: precisão, *recall* e ROC-AUC.

Modelo	Precisão	*Recall*	ROC-AUC
AdaBoost	0.827 ± 0.015	0.852 ± 0.013	0.926 ± 0.007
Árvore de Decisão	0.837 ± 0.012	0.515 ± 0.023	0.757 ± 0.011
Floresta Aleatória	**0.956 ± 0.009**	0.663 ± 0.024	0.831 ± 0.012
KNN	0.901 ± 0.009	0.687 ± 0.024	0.843 ± 0.012
LSTM	0.749 ± 0.024	0.922 ± 0.050	**0.961 ± 0.025**
Bayes ingênuo	0.017 ± 0.000	**0.999 ± 0.001**	0.866 ± 0.001
SVM	0.855 ± 0.011	0.849 ± 0.018	0.924 ± 0.009

5.1.3 Discussão dos Resultados

O modelo SVM demonstrou um bom desempenho na detecção de indisponibilidades. Sua capacidade de traçar hiperplanos de separação eficientes entre as classes permitiu uma boa discriminação entre amostras de indisponibilidade e disponibilidade. Isso é evidenciado pela alta pontuação no índice F1-Score, indicando uma boa capacidade de classificação do modelo.

Por outro lado, o modelo *AdaBoost* também demonstrou um desempenho promissor na detecção de indisponibilidades. O algoritmo de *boosting* permitiu combinar vários classificadores fracos para formar um classificador forte. A capacidade do *AdaBoost* de focar nas amostras mal classificadas em iterações anteriores ajudou a melhorar a sensibilidade na detecção de indisponibilidades.

Destaca-se também o desempenho superior do modelo LSTM na detecção de indisponibilidades, conforme evidenciado por seu maior ROC-AUC em comparação com os outros modelos. A LSTM é uma arquitetura de rede neural recorrente especialmente projetada para lidar com sequências de dados, como séries temporais. Sua capacidade de capturar dependências de longo prazo nas sequências pode ter sido um fator chave para seu desempenho superior. Além disso, a LSTM é capaz de aprender automaticamente padrões complexos e não lineares presentes nos dados, o que pode ser crucial para a detecção de indisponibilidades.

5.1.4 Interpretabilidade dos Resultados

Nesta etapa, optou-se pelo uso do modelo SVM para interpretar os resultados, devido ao seu alto desempenho e simplicidade em comparação à rede LSTM, que utiliza apenas um ponto temporal. A biblioteca SHAP [131] foi empregada para realizar uma avaliação das saídas e identificar a relevância de cada atributo de acordo com o modelo. Dado o grande volume de dados e o tempo de processamento exigido pelo SHAP, uma amostra contendo 10% dos dados de teste foi utilizada para fins de avaliação.

A Figura 5.6 exibe o gráfico de *beeswarm* [132], que ilustra os valores de cada saída em relação aos seus atributos. Os atributos são ordenados de acordo com sua relevância, indicando que a categoria de evento 5 é o atributo mais influente neste modelo. Essa categoria representa a quantidade geral de erros categorizados pela ferramenta de *web analytics*, abrangendo os principais eventos de erro e se caracterizando por ser genérica e correlacionada as indisponibilidades.

A análise dos atributos por meio do gráfico *beeswarm* revelou que a maioria deles apresenta valores SHAP positivos, indicando uma influência significativa no modelo. Esses valores predominam à direita do gráfico, o que sugere que altos valores nesses atributos estão associados a predições mais positivas. Por outro lado, observou-se que certos atributos possuem valores SHAP negativos, indicando um impacto negativo nas predições do modelo. Esses valores são visualizados à esquerda do gráfico e são representados pela coloração vermelha dos pontos, indicando a sua importância na contribuição para as predições negativas.

Dentre os atributos relevantes, destaca-se a URL 101, que representa uma página de erro de login para a qual os clientes são redirecionados após receberem uma falha na au-

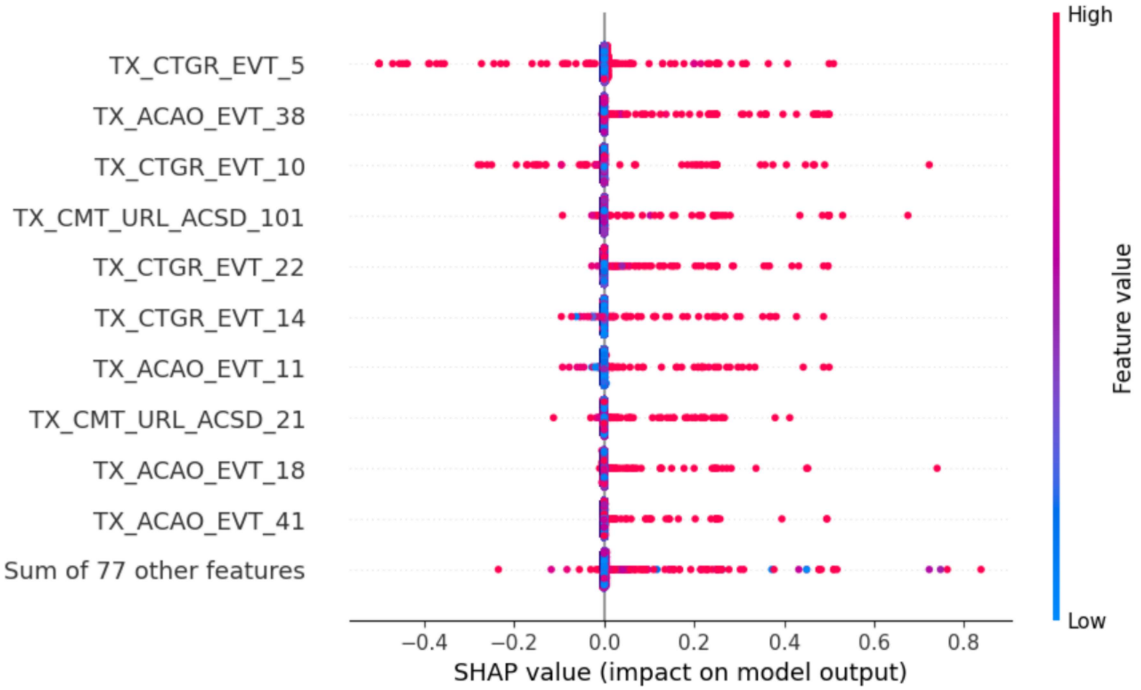

Figura 5.6: Interpretabilidade do Modelo.

tenticação. O evento 38, o segundo mais relevante, indica uma mensagem de erro recebida pelo usuário quando o aplicativo não consegue estabelecer uma conexão com os servidores. Além disso, com base no conhecimento interno da aplicação, é possível inferir que a falta de resposta nas requisições inicia uma operação transparente ao cliente chamada de *ping*, na qual o aplicativo realiza uma verificação de rede. Esse comportamento é caracterizado pelo evento 18.

5.2 Implantação

Nesta seção serão considerados os resultados obtidos para a criação de um plano de implantação do modelo proposto. Tal plano descreve como a proposta deverá ser integrada ao processo de monitoração de uma empresa do ramo financeiro. Também serão apresentadas estratégias para acompanhar o modelo ao longo do tempo visando monitorar o seu desempenho e criando um plano de manutenção, no qual será descrito quando o modelo deve ser retreinado para acompanhar as evoluções dos demais sistemas envolvidos.

5.2.1 Plano de Implantação

O modelo desenvolvido demanda sua integração no fluxo de monitoração contínua da organização. Para atingir esse objetivo, elaborou-se um código em linguagem Python

que realiza consultas à base de dados a cada intervalo de um minuto. Para acessar os recursos do ecossistema *Hadoop*, efetua-se uma chamada ao *Apache Spark* [124], uma ferramenta de processamento distribuído que se conecta ao *Apache Hive* [122]. Esta conexão permite a consulta das entradas previamente filtradas para o modelo. O resultado dessa consulta é então direcionado ao modelo, onde os dados são processados e a saída é gerada, apresentando o estado atual do serviço.

A saída do modelo desencadeia dois processos subsequentes. No primeiro cenário, em que o serviço está disponível, ocorre somente a gravação das entradas do modelo, saída e horário de processamento. Esses dados são registrados em um banco de dados relacional para permitir o acompanhamento futuro do desempenho do modelo.

Por outro lado, quando a saída do modelo indica uma indisponibilidade, o programa efetua uma chamada à API da ferramenta de monitoração da organização. Esta ação inicialmente gera um alerta à equipe de monitoração. Além disso, são iniciados outros mecanismos de comunicação, incluindo o acionamento de aplicativos de mensagem instantânea diretamente para executivos e técnicos responsáveis pelo sistema. Essas estratégias visam a ampliar a eficiência na detecção e notificação de eventos críticos.

Após a implementação do modelo, observou-se um atraso na chegada das mensagens ao ambiente *Apache Hive*. Como resultado, as consultas efetuadas abrangem um intervalo de tempo entre dois a três minutos a menos que o horário real da consulta. Similarmente, o desempenho do *Apache Spark* ficou acima das expectativas, com um tempo médio de processamento para retorno de resultados de cerca de 27 segundos. Isso contribui para um tempo médio de atraso de aproximadamente 3 minutos no processamento dos dados antes de serem inseridos no modelo.

Diante disso, está em análise a implementação de consultas em uma fila no *Apache Kafka* [133] por intermédio do *Apache Spark* [124], visando a redução do tempo médio de consulta para aproximadamente dois minutos. Por outro lado, os demais processos envolvidos não apresentam um tempo de espera significativo, pois são executados em milissegundos.

5.2.2 Monitoração e Manutenção

Para assegurar o funcionamento preciso do processo é imperativo estabelecer um monitoramento contínuo dos dados que alimentam o modelo, com verificação de saídas ocorrendo a cada intervalo de 1 minuto. Com esse objetivo, será desenvolvido um programa de verificação dedicado a avaliar a pontualidade da chegada dos dados dos clientes à base de dados. Simultaneamente, um *script* será implementado para monitorar a consistente gravação das saídas do modelo no banco de dados relacional.

Em situações de falhas, seja em um dos fluxos de dados ou em ambos, um alerta será automaticamente gerado e encaminhado à equipe de suporte para análise. Essa abordagem garante uma intervenção rápida em caso de qualquer anomalia, contribuindo para a funcionalidade ininterrupta do modelo de monitoramento.

Ao longo dos anos, é uma expectativa natural que o aplicativo móvel passe por evoluções, levando a modificações nos dados coletados. Novas funcionalidades e páginas podem ser introduzidas no aplicativo, bem como outras podem ser eliminadas. Isso implica na necessidade de monitorar continuamente a eficácia do modelo ao longo do tempo.

Dentre as atividades realizadas, o processo de rotulação dos dados é o único realizado manualmente. Como resultado, é necessário realizar a rotulagem dos dados históricos pelo menos uma vez a cada semestre, introduzindo novos incidentes. Explora-se a possibilidade de criar um procedimento automatizado que identifique incidentes críticos relacionados ao aplicativo móvel e, em seguida, atribua rótulos automaticamente aos conjuntos de dados correspondentes.

Após a rotulagem dos dados, será possível confrontar os resultados registrados no banco de dados relacional com os dados reais de indisponibilidades, permitindo uma reavaliação do modelo. Caso o desempenho indicado apresente um *F1-Score* abaixo do valor de corte estabelecido como 0.8, será necessário iniciar um novo ciclo de mineração de dados, buscando identificar novas ações e URLs que representem o atual funcionamento do aplicativo.

Importante ressaltar que todo o processo de descoberta de atributos e retreinamento do modelo deve ser automatizado, ainda que supervisionado por um cientista de dados, a fim de acompanhar os resultados de cada etapa de forma contínua.

5.3 Considerações Finais

Neste capítulo foram avaliados os resultados do processo modelado anteriormente. Inicialmente, as saídas dos modelos no conjunto de validação foram examinadas e os melhores hiperparâmetros foram extraídos. É importante mencionar que a filtragem desempenhou um papel significativo, demonstrando uma presença marcante nos resultados mais promissores.

Em seguida, os melhores hiperparâmetros foram aplicados ao conjunto de teste, no qual os modelos passaram por sua avaliação final. Assim sendo, foi notável que três algoritmos apresentaram resultados bastante próximos, evidenciando suas habilidades em detectar falhas e discernir momentos de disponibilidade e indisponibilidade. Os resultados foram discutidos e uma interpretação foi oferecida para facilitar a compreensão.

Além disso, o plano de implantação do modelo foi apresentado, delineando sua integração com os sistemas legados da organização. A monitoração contínua do modelo foi abordada para assegurar o seu correto funcionamento. Por fim, a manutenção do modelo foi explorada, delineando como ele se adaptará às mudanças nos dados ao longo do tempo. Esse conjunto de ações visa assegurar a eficácia e a utilidade contínuas do modelo em um ambiente em constante evolução.

No próximo capítulo serão apresentadas as conclusões obtidas ao longo deste trabalho, assim como as suas limitações e como superá-las em trabalhos futuros.

Capítulo 6

Conclusão

Durante o desenvolvimento deste trabalho foi apresentada uma abordagem inovadora para a detecção de indisponibilidades em aplicações web, utilizando dados das interações dos usuários com o sistema. Essa abordagem proporciona uma visão direta sobre a experiência do usuário final, colocando o foco na persona mais crucial para o sucesso do negócio: o cliente. Assim, por meio desta dessa perspectiva centrada no usuário, o objetivo foi identificar as indisponibilidades de forma mais ágil, resultando em uma redução do tempo de inatividade e um aumento geral na satisfação dos clientes.

Para alcançar os objetivos estabelecidos foram desenvolvidos e avaliados diversos modelos de classificação. Os resultados obtidos foram promissores, com um *F1-Score* próximo de 0,81 e um ROC-AUC superior a 0,92. Esses valores destacam a eficácia dos modelos na detecção de indisponibilidades, reduzindo significativamente a ocorrência de alertas falsos. Além disso, o modelo LSTM apresentou um *recall* de 0,922, o que reforça ainda mais a robustez da abordagem na identificação da grande maioria das indisponibilidades. Esses resultados indicam que a abordagem tem um desempenho notável na identificação e classificação de eventos de indisponibilidade, fornecendo uma base sólida para a melhoria da qualidade e confiabilidade de aplicações web.

Dessa forma, este estudo oferece contribuições significativas para o campo da detecção de falhas em sistemas web. A abordagem baseada nas interações dos usuários com a aplicação demonstrou eficácia na identificação de anomalias em sistemas complexos, preenchendo uma lacuna na pesquisa que tradicionalmente se concentra em *logs* de sistema ou aplicações específicas na nuvem. Além disso, a otimização de técnicas de pré-processamento e hiperparâmetros utilizando a biblioteca Hyperopt apresenta uma contribuição metodológica no contexto de mineração de dados. Essas contribuições têm o potencial de melhorar a confiabilidade e a estabilidade das aplicações web em um cenário digital em constante evolução, beneficiando tanto a pesquisa acadêmica quanto as práticas da indústria de tecnologia da informação.

O estudo proposto se baseia em um cenário real, com um conjunto de dados extraído de uma grande instituição financeira brasileira. Isso se traduz em benefícios tangíveis para empresas e usuários finais, permitindo uma identificação mais rápida e precisa de problemas, o que resulta em tempos de resposta mais curtos e uma experiência *online* mais confiável. A capacidade de mitigar impactos negativos e reduzir interrupções não apenas melhora a satisfação do cliente, mas também potencialmente economiza recursos financeiros e operacionais. Isso consolida a importância prática dessa abordagem na garantia da estabilidade das aplicações web em um cenário tecnológico cada vez mais interconectado.

A hipótese desta pesquisa, que postulava que a detecção de falhas com base nas interações dos usuários em aplicações web poderia oferecer uma estratégia eficaz para reduzir os impactos negativos das falhas, foi confirmada pelos resultados obtidos. A evidência empírica, respaldada pelo desempenho do modelo LSTM e a otimização das técnicas de pré-processamento, fortaleceu a validade dessa hipótese central. Portanto, esta pesquisa não apenas validou a hipótese original, mas também destacou seu potencial prático e a sua relevância para a área de detecção de falhas em sistemas web.

Assim sendo, este trabalho proporciona perspectivas importantes para a área de pesquisa em monitoramento e gerenciamento de aplicações web. Os modelos desenvolvidos demonstraram excelente capacidade de identificar a ocorrência de indisponibilidades. A abordagem proposta, ao colocar o usuário como figura central na análise, mostrou-se promissora ao permitir a detecção e mitigação ágil das indisponibilidades, resultando em uma experiência aprimorada para os clientes e vantagens significativas para as empresas.

6.1 Trabalhos Futuros

A abordagem proposta apresentou um excelente desempenho na detecção em tempo real das indisponibilidades. Todavia, é importante ressaltar que não foi possível realizar previsões de falhas ou identificar a causa raiz. Essas lacunas podem ser abordadas por meio da incorporação de dados provenientes da camada de infraestrutura, como registros de aplicativos (*logs*), métricas de desempenho (por exemplo, uso de CPU, consumo de memória, atividade de disco) e latência de requisições.

Um conjunto adicional de dados que está ganhando relevância e que pode ser incorporado ao estudo são os dados de telemetria. A inclusão desses elementos poderia levar a uma redução ainda maior ou até mesmo à prevenção de indisponibilidades. É crucial destacar que essas limitações serão investigadas em estudos futuros, com o intuito de aprimorar ainda mais a abordagem proposta.

6.2 Aplicabilidade

Esta dissertação não se limita apenas a contribuições teóricas dentro do escopo estabelecido, mas também almeja fornecer benefícios em diversos cenários. A aplicabilidade dos resultados alcançados pode ser ampliada para abranger outras aplicações baseadas em sistemas *web*. Assim, contribuindo para a melhoria de confiabilidade de sites que estão presentes em nosso cotidiano.

Além disso, a solução proposta apresenta a flexibilidade necessária para se adaptar à identificação de padrões que vão além da detecção de indisponibilidades, abrangendo a detecção de fraudes, ataques ou comportamentos anômalos em sites. Considerando que os dados de experiência do usuário são ricos em informações, eles podem servir como base para abordar diferentes problemas de mineração de dados em ambientes web.

A base de dados rotulada, disponibilizada publicamente como parte deste trabalho, oferece amplas possibilidades de aplicação em pesquisas futuras. Seja para avaliação de algoritmos, treinamento de modelos ou desenvolvimento de novas técnicas, a disponibilidade dessa base de dados representa um recurso valioso, tanto para a comunidade acadêmica quanto para a prática.

Referências

[1] Zhang, Biao, Xinyan Dong, Yuwei Hu, Xuchu Jiang e Gongchi Li: *Classification and prediction of spinal disease based on the smote-rfe-xgboost model.* PeerJ Computer Science, 9:e1280, 2023. xi, 12

[2] Bergstra, James, Dan Yamins, David D Cox *et al.*: *Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms.* Em *Proceedings of the 12th Python in science conference*, volume 13, página 20. Citeseer, 2013. xi, 14, 15

[3] Lundberg, Scott M e Su In Lee: *A unified approach to interpreting model predictions.* Advances in neural information processing systems, 30, 2017. xi, 15, 16

[4] Hastie, Trevor, Robert Tibshirani e Jerome Friedman: *Overview of supervised learning.* Em *The elements of statistical learning*, páginas 9–41. Springer, 2009. xi, 21

[5] Noble, William S: *What is a support vector machine?* Nature biotechnology, 24(12):1565–1567, 2006. xi, 22, 23

[6] Safavian, S Rasoul e David Landgrebe: *A survey of decision tree classifier methodology.* IEEE transactions on systems, man, and cybernetics, 21(3):660–674, 1991. xi, 24

[7] Krogh, Anders: *What are artificial neural networks?* Nature biotechnology, 26(2):195–197, 2008. xi, 26

[8] Abraham, Ajith: *Artificial neural networks.* Handbook of measuring system design, 2005. xi, 25, 27

[9] Van Houdt, Greg, Carlos Mosquera e Gonzalo Nápoles: *A review on the long short-term memory model.* Artificial Intelligence Review, 53(8):5929–5955, 2020. xi, 27, 28

[10] Malaquias, Rodrigo F e Yujong Hwang: *Mobile banking use: A comparative study with brazilian and us participants.* International Journal of Information Management, 44:132–140, 2019. 1

[11] Meirelles, Fernando de Souza: *Pesquisa anual do uso de ti.* Fundação Getúlio Vargas, 2022. 1

[12] Afshan, Sahar e Arshian Sharif: *Acceptance of mobile banking framework in pakistan.* Telematics and Informatics, 33(2):370–387, 2016. 1

[13] Duarte, Angelo, Jon Frost, Leonardo Gambacorta, Priscilla Koo Wilkens e Hyun Song Shin: *Central banks, the monetary system and public payment infrastructures: lessons from brazil's pix.* Available at SSRN 4064528, 2022. 1

[14] Jordan, Michael I e Tom M Mitchell: *Machine learning: Trends, perspectives, and prospects.* Science, 349(6245):255–260, 2015. 5

[15] Mitchell, Tom M e Tom M Mitchell: *Machine learning*, volume 9. McGraw-hill New York, 1997. 5

[16] Bishop, Christopher M e Nasser M Nasrabadi: *Pattern recognition and machine learning*, volume 4. Springer, 2006. 5

[17] Kaelbling, Leslie Pack, Michael L Littman e Andrew W Moore: *Reinforcement learning: A survey.* Journal of artificial intelligence research, 4:237–285, 1996. 5

[18] Dayan, Peter e Yael Niv: *Reinforcement learning: the good, the bad and the ugly.* Current opinion in neurobiology, 18(2):185–196, 2008. 6

[19] Zhou, Zhi Hua: *A brief introduction to weakly supervised learning.* National science review, 5(1):44–53, 2018. 6

[20] Nasteski, Vladimir: *An overview of the supervised machine learning methods.* Horizons. b, 4:51–62, 2017. 6

[21] Liu, Bing e Bing Liu: *Supervised learning.* Springer, 2011. 6

[22] Seber, George AF e Alan J Lee: *Linear regression analysis.* John Wiley & Sons, 2012. 6

[23] Tibshirani, Robert: *Regression shrinkage and selection via the lasso.* Journal of the Royal Statistical Society: Series B (Methodological), 58(1):267–288, 1996. 6

[24] Cortes, Corinna e Vladimir Vapnik: *Support-vector networks.* Machine learning, 20(3):273–297, 1995. 6, 10, 20

[25] Cover, Thomas e Peter Hart: *Nearest neighbor pattern classification.* IEEE transactions on information theory, 13(1):21–27, 1967. 6, 10, 20, 22, 31

[26] Breiman, Leo, Jerome H Friedman, Richard A Olshen e Charles J Stone: *Classification and regression trees.* Routledge, 2017. 6, 20

[27] Rish, Irina *et al.*: *An empirical study of the naive bayes classifier.* Em *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, páginas 41–46, 2001. 6, 20, 21

[28] Breiman, Leo: *Random forests.* Machine learning, 45(1):5–32, 2001. 6, 20, 24, 33

[29] Hastie, Trevor, Jerome Friedman, Robert Tibshirani, Trevor Hastie, Jerome Friedman e Robert Tibshirani: *Unsupervised learning.* The elements of statistical learning: Data mining, inference, and prediction, páginas 437–508, 2001. 6

[30] Ghahramani, Zoubin: *Unsupervised learning.* Em *Summer school on machine learning*, páginas 72–112. Springer, 2003. 6

[31] Xu, Rui e Donald Wunsch: *Survey of clustering algorithms.* IEEE Transactions on neural networks, 16(3):645–678, 2005. 7

[32] Davies, David L e Donald W Bouldin: *A cluster separation measure.* IEEE transactions on pattern analysis and machine intelligence, 2:224–227, 1979. 7, 10, 18

[33] Sinaga, Kristina P e Miin Shen Yang: *Unsupervised k-means clustering algorithm.* IEEE access, 8:80716–80727, 2020. 7

[34] Van Der Maaten, Laurens, Eric Postma, Jaap Van den Herik *et al.*: *Dimensionality reduction: a comparative.* J Mach Learn Res, 10(66-71):13, 2009. 7

[35] Tipping, Michael E e Christopher M Bishop: *Probabilistic principal component analysis.* Journal of the Royal Statistical Society: Series B (Statistical Methodology), 61(3):611–622, 1999. 7, 18

[36] Chapman, Pete, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, Rudiger Wirth *et al.*: *Crisp-dm 1.0: Step-by-step data mining guide.* SPSS inc, 9(13):1–73, 2000. 8, 9

[37] Phippen, Andrew, L Sheppard e Steven Furnell: *A practical evaluation of web analytics.* Internet Research, 2004. 8

[38] Gamalielsson, Jonas, Björn Lundell, Simon Butler, Christoffer Brax, Tomas Persson, Anders Mattsson, Tomas Gustavsson, Jonas Feist e Erik Lönroth: *Towards open government through open source software for web analytics: The case of matomo.* JeDEM-eJournal of eDemocracy and Open Government, 13(2):133–153, 2021. 8

[39] Ripp, Saskia e Stefan Falke: *Analyzing user behavior with matomo in the online information system grammis.* Em *Proceedings of the XVIII EURALEX International Congress Lexicography in Global Contexts 17-21 July 2018, Ljubljana*, páginas 87–100. Znanstvena založba Filozofske fakultete Univerze v Ljubljani/Ljubljana ..., 2018. 9

[40] Kumar, Vipin e Sonajharia Minz: *Feature selection: a literature review.* SmartCR, 4(3):211–229, 2014. 9

[41] Dash, Manoranjan e Huan Liu: *Feature selection for classification.* Intelligent data analysis, 1(1-4):131–156, 1997. 10

[42] Li, Jundong, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang e Huan Liu: *Feature selection: A data perspective.* ACM computing surveys (CSUR), 50(6):1–45, 2017. 10

[43] Cohen, Israel, Yiteng Huang, Jingdong Chen, Jacob Benesty, Jacob Benesty, Jingdong Chen, Yiteng Huang e Israel Cohen: *Pearson correlation coefficient.* Noise reduction in speech processing, páginas 1–4, 2009. 10, 44

[44] St, Lars, Svante Wold *et al.*: *Analysis of variance (anova).* Chemometrics and intelligent laboratory systems, 6(4):259–272, 1989. 10

[45] García, Salvador, Julián Luengo e Francisco Herrera: *Data preprocessing in data mining*, volume 72. Springer, 2015. 10

[46] Singh, Dalwinder e Birmohan Singh: *Investigating the impact of data normalization on classification performance.* Applied Soft Computing, 97:105524, 2020. 10

[47] Mohamad, Ismail Bin e Dauda Usman: *Standardization and its effects on k-means clustering algorithm.* Research Journal of Applied Sciences, Engineering and Technology, 6(17):3299–3303, 2013. 10

[48] Raju, VN Ganapathi, K Prasanna Lakshmi, Vinod Mahesh Jain, Archana Kalidindi e V Padma: *Study the influence of normalization/transformation process on the accuracy of supervised classification.* Em *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, páginas 729–735. IEEE, 2020. 10

[49] Chawla, Nitesh V, Kevin W Bowyer, Lawrence O Hall e W Philip Kegelmeyer: *Smote: synthetic minority over-sampling technique.* Journal of artificial intelligence research, 16:321–357, 2002. 11, 52

[50] Kreuzberger, Dominik, Niklas Kühl e Sebastian Hirschl: *Machine learning operations (mlops): Overview, definition, and architecture.* IEEE Access, 2023. 12

[51] *DevOps: Documentation.* `https://www.devops.com/`, Available online: devops.com/ (accessed on 03/08/2023). 12

[52] Zaharia, Matei, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe *et al.*: *Accelerating the machine learning lifecycle with mlflow.* IEEE Data Eng. Bull., 41(4):39–45, 2018. 12

[53] Putatunda, Sayan e Kiran Rama: *A comparative analysis of hyperopt as against other approaches for hyper-parameter optimization of xgboost.* Em *Proceedings of the 2018 international conference on signal processing and machine learning*, páginas 6–10, 2018. 13

[54] Bergstra, James, Rémi Bardenet, Yoshua Bengio e Balázs Kégl: *Algorithms for hyper-parameter optimization.* Advances in neural information processing systems, 24, 2011. 14

[55] Galup, Stuart D, Ronald Dattero, Jim J Quan e Sue Conger: *An overview of it service management.* Communications of the ACM, 52(5):124–127, 2009. 16

[56] Gërvalla, Muhamet, Naim Preniqi e Peter Kopacek: *It infrastructure library (itil) framework approach to it governance.* IFAC-PapersOnLine, 51(30):181–185, 2018. 16

[57] Magalhães, Ivan Luizio e Walfrido Brito Pinheiro: *Gerenciamento de serviços de TI na prática: uma abordagem com base na ITIL: inclui ISO/IEC 20.000 e IT Flex.* Novatec Editora, 2007. 16

[58] Limited, AXELOS: *ITIL foundation: ITIL 4 edition.* TSO (The Stationery Office), ein Unternehmen von Williams Lea, 2019. 16, 17, 18

[59] Cannon, David, David Wheeldon, Sharon Taylor e Office of Government Commerce Großbritannien: *ITIL.[4]. Service operation.* TSO (The Stationery Office), 2007. 17

[60] Gupta, Rajeev, K Hima Prasad e Mukesh Mohania: *Automating itsm incident management process.* Em *2008 International Conference on Autonomic Computing*, páginas 141–150. IEEE, 2008. 17

[61] Kesavaraj, Gopalan e Sreekumar Sukumaran: *A study on classification techniques in data mining.* Em *2013 fourth international conference on computing, communications and networking technologies (ICCCNT)*, páginas 1–7. IEEE, 2013. 20

[62] Webb, Geoffrey I, Eamonn Keogh e Risto Miikkulainen: *Naïve bayes.* Encyclopedia of machine learning, 15:713–714, 2010. 21

[63] Sun, Shiliang e Rongqing Huang: *An adaptive k-nearest neighbor algorithm.* Em *2010 seventh international conference on fuzzy systems and knowledge discovery*, volume 1, páginas 91–94. IEEE, 2010. 22

[64] Keller, James M, Michael R Gray e James A Givens: *A fuzzy k-nearest neighbor algorithm.* IEEE transactions on systems, man, and cybernetics, (4):580–585, 1985. 22

[65] Fix, Evelyn: *Discriminatory analysis: nonparametric discrimination, consistency properties*, volume 1. USAF school of Aviation Medicine, 1985. 22

[66] Lorena, Ana Carolina e André CPLF De Carvalho: *Uma introdução às support vector machines.* Revista de Informática Teórica e Aplicada, 14(2):43–67, 2007. 22

[67] Vapnik, Vladimir: *On the uniform convergence of relative frequencies of events to their probabilities.* Em *Doklady Akademii Nauk USSR*, volume 181, páginas 781–787, 1968. 22

[68] Vapnik, Vladimir: *The nature of statistical learning theory.* Springer science & business media, 1999. 22, 33

[69] Swain, Philip H e Hans Hauska: *The decision tree classifier: Design and potential.* IEEE Transactions on Geoscience Electronics, 15(3):142–147, 1977. 23

[70] Du, Wenliang e Zhijun Zhan: *Building decision tree classifier on private data.* 2002. 24

[71] Amit, Yali e Donald Geman: *Shape quantization and recognition with randomized trees.* Neural computation, 9(7):1545–1588, 1997. 24

[72] Biau, Gérard e Erwan Scornet: *A random forest guided tour.* Test, 25:197–227, 2016. 24, 25

[73] Schapire, Robert E: *Explaining adaboost.* Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik, páginas 37–52, 2013. 25

[74] Freund, Yoav e Robert E Schapire: *A decision-theoretic generalization of on-line learning and an application to boosting.* Journal of computer and system sciences, 55(1):119–139, 1997. 25

[75] Hastie, Trevor, Saharon Rosset, Ji Zhu e Hui Zou: *Multi-class adaboost.* Statistics and its Interface, 2(3):349–360, 2009. 25

[76] Da Silva, Ivan Nunes, Danilo Hernane Spatti, Rogerio Andrade Flauzino, Luisa Helena Bartocci Liboni e Silas Franco dos Reis Alves: *Artificial neural networks.* Cham: Springer International Publishing, 39, 2017. 25

[77] McCulloch, Warren S e Walter Pitts: *A logical calculus of the ideas immanent in nervous activity.* The bulletin of mathematical biophysics, 5(4):115–133, 1943. 25, 26

[78] Rosenblatt, Frank: *The perceptron: a probabilistic model for information storage and organization in the brain.* Psychological review, 65(6):386, 1958. 25

[79] Jain, Anil K, Jianchang Mao e K Moidin Mohiuddin: *Artificial neural networks: A tutorial.* Computer, 29(3):31–44, 1996. 26

[80] Werbos, Paul: *Beyond regression:" new tools for prediction and analysis in the behavioral sciences.* Ph. D. dissertation, Harvard University, 1974. 26

[81] LeCun, Yann, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard e Lawrence D Jackel: *Backpropagation applied to handwritten zip code recognition.* Neural computation, 1(4):541–551, 1989. 26

[82] Zou, Jinming, Yi Han e Sung Sau So: *Overview of artificial neural networks.* Artificial neural networks: methods and applications, páginas 14–22, 2009. 26

[83] Hopfield, John J: *Neural networks and physical systems with emergent collective computational abilities.* Proceedings of the national academy of sciences, 79(8):2554–2558, 1982. 26

[84] Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho e Yoshua Bengio: *Empirical evaluation of gated recurrent neural networks on sequence modeling.* arXiv preprint arXiv:1412.3555, 2014. 27

[85] Hochreiter, Sepp: *The vanishing gradient problem during learning recurrent neural nets and problem solutions.* International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6(02):107–116, 1998. 27

[86] Hochreiter, Sepp e Jürgen Schmidhuber: *Long short-term memory.* Neural computation, 9(8):1735–1780, 1997. 27, 32, 33

[87] Smagulova, Kamilya e Alex Pappachen James: *A survey on lstm memristive neural network architectures and applications.* The European Physical Journal Special Topics, 228(10):2313–2324, 2019. 27

[88] Malhotra, Pankaj, Lovekesh Vig, Gautam Shroff, Puneet Agarwal *et al.*: *Long short term memory networks for anomaly detection in time series.* Em *Proceedings*, volume 89, páginas 89–94, 2015. 27

[89] Hasnain, Muhammad, Muhammad Fermi Pasha, Imran Ghani, Muhammad Imran, Mohammed Y Alzahrani e Rahmat Budiarto: *Evaluating trust prediction and confusion matrix measures for web services ranking.* IEEE Access, 8:90847–90861, 2020. 28

[90] Sasaki, Yutaka *et al.*: *The truth of the f-measure.* Teach tutor mater, 1(5):1–5, 2007. 30

[91] Hossin, Mohammad e Md Nasir Sulaiman: *A review on evaluation metrics for data classification evaluations.* International journal of data mining & knowledge management process, 5(2):1, 2015. 30

[92] Al-Hawari, Feras e Hala Barham: *A machine learning based help desk system for it service management.* Journal of King Saud University-Computer and Information Sciences, 33(6):702–718, 2021. 31, 34

[93] Shao, Qihong, Yi Chen, Shu Tao, Xifeng Yan e Nikos Anerousis: *Efficient ticket routing by resolution sequence mining.* Em *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 605–613, 2008. 31, 34

[94] Shao, Qihong, Yi Chen, Shu Tao, Xifeng Yan e Nikos Anerousis: *Easyticket: a ticket routing recommendation engine for enterprise problem resolution.* Proceedings of the VLDB Endowment, 1(2):1436–1439, 2008. 31

[95] Ching, Wai Ki e Michael K Ng: *Markov chains.* Models, algorithms and applications, 2006. 31, 33

[96] Miao, Gengxin, Louise E Moser, Xifeng Yan, Shu Tao, Yi Chen e Nikos Anerousis: *Generative models for ticket resolution in expert networks.* Em *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 733–742, 2010. 31, 34

[97] Paramesh, SP e KS Shreedhara: *Automated it service desk systems using machine learning techniques.* Em *Data Analytics and Learning*, páginas 331–346. Springer, 2019. 31

[98] Silva, Sara, Rúben Pereira e Ricardo Ribeiro: *Machine learning in incident categorization automation.* Em *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, páginas 1–6. IEEE, 2018. 31

[99] Kikuchi, Shinji: *Prediction of workloads in incident management based on incident ticket updating history.* Em *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, páginas 333–340. IEEE, 2015. 31

[100] Zhou, Wubai, Liang Tang, Chunqiu Zeng, Tao Li, Larisa Shwartz e Genady Ya Grabarnik: *Resolution recommendation for event tickets in service management.* IEEE Transactions on Network and Service Management, 13(4):954–967, 2016. 31, 34

[101] Zhou, Wubai, Wei Xue, Ramesh Baral, Qing Wang, Chunqiu Zeng, Tao Li, Jian Xu, Zheng Liu, Larisa Shwartz e Genady Ya. Grabarnik: *Star: A system for ticket analysis and resolution.* Em *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, páginas 2181–2190, 2017. 31, 34

[102] Yun, Mingchun, Yuqing Lan e Tao Han: *Automate incident management by decision-making model.* Em *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, páginas 217–222. IEEE, 2017. 31, 34

[103] Sparck Jones, Karen: *A statistical interpretation of term specificity and its application in retrieval.* Journal of documentation, 28(1):11–21, 1972. 31

[104] Du, Min, Feifei Li, Guineng Zheng e Vivek Srikumar: *Deeplog: Anomaly detection and diagnosis from system logs through deep learning.* Em *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, páginas 1285–1298, 2017. 31, 33, 34

[105] Borthakur, Dhruba: *Hdfs architecture.* Document on Hadoop Wiki. URL http://hadoop. apache. org/common/docs/r0, 20, 2010. 31, 41

[106] Sefraoui, Omar, Mohammed Aissaoui, Mohsine Eleuldj *et al.*: *Openstack: toward an open-source solution for cloud computing.* International Journal of Computer Applications, 55(3):38–42, 2012. 32, 33

[107] Zhang, Xu, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xinsheng Yang, Qian Cheng, Ze Li *et al.*: *Robust log-based anomaly detection on unstable log data.* Em *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, páginas 807–817, 2019. 32, 33, 34

[108] Graves, Alex e Jürgen Schmidhuber: *Framewise phoneme classification with bidirectional lstm and other neural network architectures.* Neural networks, 18(5-6):602–610, 2005. 32

[109] Zhao, Nengwen, Honglin Wang, Zeyan Li, Xiao Peng, Gang Wang, Zhu Pan, Yong Wu, Zhen Feng, Xidao Wen, Wenchi Zhang *et al.*: *An empirical investigation of practical log anomaly detection for online service systems.* Em *Proceedings of the 29th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*, páginas 1404–1415, 2021. 32, 34

[110] Munir, Mohsin, Shoaib Ahmed Siddiqui, Andreas Dengel e Sheraz Ahmed: *Deepant: A deep learning approach for unsupervised anomaly detection in time series.* Ieee Access, 7:1991–2005, 2018. 32, 33, 34

[111] Albawi, Saad, Tareq Abed Mohammed e Saad Al-Zawi: *Understanding of a convolutional neural network.* Em *2017 international conference on engineering and technology (ICET)*, páginas 1–6. Ieee, 2017. 32

[112] Meng, Weibin, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun *et al.*: *Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs.* Em *IJCAI*, volume 19, páginas 4739–4745, 2019. 32, 34

[113] Mikolov, Tomas, Kai Chen, Greg Corrado e Jeffrey Dean: *Efficient estimation of word representations in vector space.* arXiv preprint arXiv:1301.3781, 2013. 32

[114] Islam, Mohammad S, William Pourmajidi, Lei Zhang, John Steinbacher, Tony Erwin e Andriy Miranskyy: *Anomaly detection in a large-scale cloud platform.* Em *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, páginas 150–159. IEEE, 2021. 32, 34

[115] Girish, L e Sridhar KN Rao: *Anomaly detection in cloud environment using artificial intelligence techniques.* Computing, 105(3):675–688, 2023. 33, 34

[116] Dai Vu, Dinh, Xuan Tuong Vu e Younghan Kim: *Deep learning-based fault prediction in cloud system.* Em *2021 International Conference on Information and Communication Technology Convergence (ICTC)*, páginas 1826–1829. IEEE, 2021. 33, 34

[117] Gao, Jiechao, Haoyu Wang e Haiying Shen: *Task failure prediction in cloud data centers using deep learning.* IEEE transactions on services computing, 2020. 33, 34

[118] Chen, Xin, Charng Da Lu e Karthik Pattabiraman: *Failure prediction of jobs in compute clouds: A google cluster case study.* Em *2014 IEEE International Symposium on Software Reliability Engineering Workshops*, páginas 341–346. IEEE, 2014. 33

[119] Yuan, Yue, Wenchang Shi, Bin Liang e Bo Qin: *An approach to cloud execution failure diagnosis based on exception logs in openstack.* Em *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, páginas 124–131. IEEE, 2019. 33

[120] Beyer, Betsy, Chris Jones, Jennifer Petoff e Niall Richard Murphy: *Site reliability engineering: How Google runs production systems.* " O'Reilly Media, Inc.", 2016. 41

[121] Oussous, Ahmed, Fatima Zahra Benjelloun, Ayoub Ait Lahcen e Samir Belfkih: *Big data technologies: A survey.* Journal of King Saud University-Computer and Information Sciences, 30(4):431–448, 2018. 41

[122] *Apache hive.* `https://hive.apache.org/`. 41, 42, 67

[123] *Apache hadoop.* https://hadoop.apache.org/. 41

[124] *Apache spark.* https://spark.apache.org/. 42, 67

[125] Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot e E. Duchesnay: *Scikit-learn: Machine learning in Python.* Journal of Machine Learning Research, 12:2825–2830, 2011. 49

[126] Bergstra, James, Daniel Yamins e David Cox: *Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures.* Em *International conference on machine learning*, páginas 115–123. PMLR, 2013. 50

[127] Lemaître, Guillaume, Fernando Nogueira e Christos K. Aridas: *Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning.* Journal of Machine Learning Research, 18(17):1–5, 2017. http://jmlr.org/papers/v18/16-365.html. 52

[128] Raschka, Sebastian, Joshua Patterson e Corey Nolet: *Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence.* Information, 11(4):193, 2020. 54

[129] Chollet, François *et al.*: *Keras.* https://keras.io, 2015. 54

[130] Zaharia, Matei, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe *et al.*: *Accelerating the machine learning lifecycle with mlflow.* IEEE Data Eng. Bull., 41(4):39–45, 2018. 55

[131] Lundberg, Scott M e Su In Lee: *A unified approach to interpreting model predictions.* Advances in neural information processing systems, 30, 2017. 65

[132] *Beeswarm plot.* https://shap.readthedocs.io/en/latest/example_notebooks/api_examples/plots/beeswarm.html. 65

[133] *Apache kafka.* https://kafka.apache.org/. 67

Apêndice A

Fichamento de Artigo Científico

Anexo I

Documentação Original UnB-CIC (parcial)

Tabela I.1: Descrição das Ações Selecionadas

Atributo	Média	Desvio Padrão	Mín	25%	50%	75%	max
ACAO_EVT_11	2375.884940	4485.148392	0.0	305.0	824.0	1843.00	60657.0
ACAO_EVT_16	1865.880461	2276.275402	0.0	538.0	1509.0	2536.00	255934.0
ACAO_EVT_18	1664.873056	2574.004855	0.0	468.0	1187.0	2053.00	437153.0
ACAO_EVT_29	929.273288	701.405470	0.0	271.0	841.0	1458.00	16842.0
ACAO_EVT_30	885.438545	2150.349985	0.0	178.0	605.0	1042.00	110087.0
ACAO_EVT_37	518.911636	459.201485	0.0	100.0	428.0	840.00	14599.0
ACAO_EVT_38	518.729349	1721.516769	0.0	94.0	370.0	704.00	126562.0
ACAO_EVT_40	478.440947	1413.831547	0.0	119.0	397.0	666.00	102974.0
ACAO_EVT_41	424.589900	2098.763539	0.0	91.0	324.0	550.00	144108.0
ACAO_EVT_47	293.271980	287.226953	0.0	79.0	252.0	441.00	9615.0
ACAO_EVT_49	290.728251	765.187064	0.0	60.0	232.0	405.00	85592.0
ACAO_EVT_51	242.958898	899.246578	0.0	50.0	180.0	341.00	108196.0
ACAO_EVT_57	204.064845	183.019265	0.0	52.0	157.0	329.00	4558.0
ACAO_EVT_58	203.214381	2082.936174	0.0	31.0	119.0	224.00	160151.0
ACAO_EVT_61	194.936226	304.514403	0.0	48.0	164.0	297.00	14339.0
ACAO_EVT_64	178.871518	150.368926	0.0	56.0	167.0	278.00	5505.0
ACAO_EVT_65	178.672556	1492.515577	0.0	27.0	104.0	196.00	131948.0
ACAO_EVT_72	171.871724	174.182326	0.0	26.0	120.0	297.00	4200.0
ACAO_EVT_76	161.193115	395.035616	0.0	10.0	24.0	49.00	3653.0
ACAO_EVT_78	158.436489	1277.578488	0.0	0.0	0.0	108.00	110150.0
ACAO_EVT_80	143.215130	1325.853880	0.0	21.0	80.0	150.00	113008.0
ACAO_EVT_88	101.632347	135.541807	0.0	26.0	88.0	156.00	8327.0
ACAO_EVT_90	89.521505	306.391003	0.0	21.0	71.0	117.00	19474.0
ACAO_EVT_92	79.320408	118.671293	0.0	16.0	64.0	119.00	13184.0
ACAO_EVT_93	68.605243	434.306369	0.0	10.0	47.0	89.00	33633.0
ACAO_EVT_95	68.181555	105.786627	0.0	10.0	36.0	104.00	11264.0
ACAO_EVT_97	63.924037	232.694844	0.0	6.0	21.0	99.00	16051.0
ACAO_EVT_126	40.734073	66.414127	0.0	9.0	32.0	62.00	7040.0
ACAO_EVT_134	36.480908	47.243776	0.0	6.0	25.0	56.00	6324.0
ACAO_EVT_146	31.762825	43.726306	0.0	6.0	21.0	41.00	3802.0
ACAO_EVT_157	28.002868	382.855776	0.0	2.0	5.0	9.00	26654.0
ACAO_EVT_161	23.776325	160.947635	0.0	2.0	9.0	23.00	23223.0
ACAO_EVT_167	23.862661	67.113320	0.0	5.0	18.0	34.00	8334.0
ACAO_EVT_175	25.691126	831.111451	0.0	0.0	0.0	0.00	66899.0
ACAO_EVT_189	21.071680	46.873900	0.0	5.0	14.0	24.00	700.0
ACAO_EVT_194	42.704352	46.289504	0.0	7.0	27.0	69.00	1542.0
ACAO_EVT_198	18.621803	30.980093	0.0	4.0	14.0	27.00	4017.0
ACAO_EVT_199	19.171139	73.130137	0.0	2.0	12.0	29.00	9882.0

Tabela I.2: Descrição das Categorias Selecionadas

Atributo	Média	Desvio Padrão	Mín	25%	50%	75%	max
CTGR_5	10224.357514	15397.504191	0.0	3042.0	9280.0	14068.00	1045098.0
CTGR_10	5767.312814	5299.696863	0.0	1809.0	5086.0	8952.00	162286.0
CTGR_14	2617.285435	4492.236338	0.0	365.0	1000.0	2395.00	60641.0
CTGR_22	1007.509156	1566.952665	0.0	233.0	772.0	1416.00	73646.0
CTGR_42	27.113929	52.933870	0.0	6.0	19.0	32.00	785.0
CTGR_48	9.016101	13.079582	0.0	1.0	4.0	11.00	354.0
CTGR_52	12.179915	16.501754	0.0	2.0	5.0	18.00	497.0
CTGR_53	10.062445	9.744912	0.0	3.0	8.0	15.00	247.0
CTGR_61	6.124385	15.766438	0.0	0.0	2.0	10.00	825.0
CTGR_63	4.926551	6.709884	0.0	0.0	2.0	8.00	137.0
CTGR_64	4.591571	6.175522	0.0	0.0	2.0	8.00	180.0
CTGR_65	4.367992	4.648602	0.0	1.0	3.0	6.00	185.0
CTGR_68	3.492866	4.779434	0.0	0.0	1.0	6.00	120.0
CTGR_69	3.308511	4.454498	0.0	0.0	1.0	5.00	95.0
CTGR_70	3.186276	4.402280	0.0	0.0	1.0	5.00	125.0
CTGR_75	2.014366	2.940061	0.0	0.0	1.0	3.00	107.0
CTGR_79	1.194485	2.739574	0.0	0.0	0.0	1.00	444.0
CTGR_80	1.734311	3.172547	0.0	0.0	1.0	3.00	145.0
CTGR_81	1.614915	2.277092	0.0	0.0	1.0	3.00	65.0
CTGR_82	1.649786	2.869019	0.0	0.0	0.0	2.00	89.0
CTGR_83	1.543168	2.310434	0.0	0.0	0.0	2.00	44.0
CTGR_106	0.240924	0.617855	0.0	0.0	0.0	0.00	13.0

Tabela I.3: Descrição das URLs Selecionadas

Atributo	Média	Desvio Padrão	Mín	25%	50%	75%	max
URL_8	12031.761458	18551.993423	0.0	436.0	2016.0	19659.00	983594.0
URL_10	5592.952069	17219.750389	0.0	242.0	663.0	1826.00	570040.0
URL_11	8830.041161	13692.089990	0.0	107.0	1059.0	13947.00	146936.0
URL_21	3090.422440	4428.504634	0.0	0.0	336.0	5864.00	126510.0
URL_27	2019.616550	1574.643257	0.0	595.0	1822.0	3179.00	35337.0
URL_31	932.993704	1168.228783	0.0	99.0	306.0	1490.25	29035.0
URL_56	764.883499	703.985285	0.0	97.0	608.0	1317.00	17213.0
URL_84	389.881051	765.321053	0.0	17.0	192.0	568.00	54236.0
URL_89	363.594412	602.050552	0.0	18.0	212.0	582.00	37488.0
URL_94	351.380589	337.097140	0.0	35.0	247.0	647.00	5144.0
URL_97	334.730529	562.492640	0.0	18.0	195.0	535.00	36891.0
URL_101	330.603825	2530.410823	0.0	0.0	26.0	458.00	184640.0
URL_107	317.263329	333.386465	0.0	35.0	218.0	511.00	7846.0
URL_111	305.909207	438.742650	0.0	2.0	35.0	564.00	11812.0
URL_112	304.312571	307.051079	0.0	43.0	239.0	483.00	8529.0
URL_113	157.280530	454.932627	0.0	4.0	14.0	50.00	11476.0
URL_115	173.511850	442.073681	0.0	9.0	31.0	86.00	16027.0
URL_117	290.455157	408.503751	0.0	33.0	188.0	444.00	20543.0
URL_131	240.937610	284.918473	0.0	20.0	131.0	378.00	6335.0
URL_137	139.172206	450.644544	0.0	0.0	2.0	10.00	12682.0
URL_138	217.877514	199.966740	0.0	32.0	188.0	363.00	7293.0
URL_143	206.560184	265.829448	0.0	6.0	83.0	337.00	5086.0
URL_149	189.207404	306.042131	0.0	9.0	93.0	293.00	8386.0
URL_157	176.208350	240.260902	0.0	17.0	90.0	245.00	6779.0
URL_164	156.591224	228.272318	0.0	8.0	86.0	260.00	9730.0
URL_168	153.860527	223.718945	0.0	9.0	86.0	255.00	5776.0