Detecção de Falhas em um Aplicativo Móvel Bancário

Daniel Schulz[1], Marcelo Marotta[1], Lucas Bondan[2,1], Marcos Caetano[1], Geraldo P. Rocha Filho[3,1], Aleteia Araujo[1]

[1]Departamento de Ciência da Computação
Universidade de Brasília – UnB – Brasília – DF – Brasil

[2]Rede Nacional de Ensino e Pesquisa – RNP

[3]Departamento de Ciências Exatas e Tecnológicas
Universidade Estadual do Sudoeste da Bahia – UESB – Vitória da Conquista – BA – Brasil

schulzdanielf@gmail.com, marcelo.marotta@unb.br, lucas.bondan@rnp.br
mfcaetano@unb.br, geraldo.rocha@uesb.edu.br, aleteia@unb.br

Abstract. *The internet has changed the way banks deliver services to customers. Due to the high number of digital accesses, interruptions in Information Systems cause great damage to the financial system. One of the strategies implemented to achieve a stable environment is the continuous monitoring of services, as described by ITIL. Given the above, this work proposes a failure detection approach through data mining techniques using the CRISP-DM reference model. The approach involves evaluating data extracted from a web analytics tool, in real time, to identify critical failures in a mobile banking application. The effects of different feature engineering techniques, such as variable filtering, data standardization and synthetic sample generation, were evaluated in 7 classification algorithms. Finally, the results were compared, and the support vector machine was the one that obtained the best result, with an F1-Score of 0.954 and a ROC-AUC of 0.989.*

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 [Malaquias and Hwang 2019]. Em 2022 no Brasil, o número de celulares inteligentes superou a quantidade de computadores e de habitantes [Meirelles 2022]. 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 [Afshan and Sharif 2016], em algumas instituições é o principal canal de interação entre os usuários e a empresa, e tem se tornado um importante aliado para a satisfação dos clientes.

Contudo, devido ao elevado número de acessos via meios digitais, interrupções em Sistemas de Informação (SI) no setor bancário têm provocado grandes prejuízos as organizações, 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.

Nesse contexto, diversos trabalhos na literatura abordam o tema de detecção de falhas em Sistema de Informação, por meio da análise de registros (*logs*) dos componentes ou consumo dos recursos [Du et al. 2017], [Zhang et al. 2019], [Munir et al. 2018]. 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, o objetivo deste trabalho é propor uma abordagem de detecção de falhas por meio das interações dos usuários com um aplicativo móvel bancário. A abordagem consiste na classificação, em tempo real, das interações dos usuários com o sistema para identificar falhas generalizadas na infraestrutura. Para atingir este objetivo foi modelado um conjunto de dados rotulados das interações dos usuários com a ferramenta, e da disponibilidade do aplicativo móvel. Na sequência, avaliou-se técnicas de engenharia de atributos para melhorar os resultados dos algoritmos de classificação. Por fim, implementou-se diferentes algoritmos para classificar as interações dos usuários, de acordo com a disponibilidade do serviço.

Para apresentar o trabalho redigido, este artigo está organizado, além desta seção, em mais cinco seções. Na Seção 2 são descritos os conceitos necessários para o entendimento deste trabalho. Em seguida, na Seção 3 são apresentados os trabalhos correlatos. Na Seção 4, o estudo de caso com a metodologia utilizada para avaliação é descrito em detalhe. Na Seção 5 são apresentados os resultados obtidos. Por último, na Seção 6 são apresentadas as conclusões e os trabalhos futuros.

2. Referencial Teórico

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 [Phippen et al. 2004]. Tal necessidade permitiu o desenvolvimento de ferramentas especializadas na extração de métricas em páginas na *web*, denominadas de *Web Analytics*. Estas ferramentas não capturam apenas a quantidade de visualizações nas páginas, mas medem o tempo de um clique para outro, onde o *mouse* de um computador está posicionado, qual mensagem o cliente recebeu do sistema ao realizar uma ação, e toda a jornada de um cliente, desde o momento que ele entra na página inicial até a saída da aplicação.

A Gestão de Serviços de TI provê uma estrutura para alinhar as atividades relacionadas a operação entre os usuários e a equipe técnica [Galup et al. 2009]. Um dos *frameworks* mais utilizados no contexto de infraestrutura é o ITIL *(Information Technology Infrastructure Library)*. Atualmente, o ITIL possui 34 práticas de gestão, dentre elas a gestão de incidentes e a gestão de eventos e monitoração. O objetivo da gestão de incidentes é minimizar o efeito ao negócio, restabelecendo o serviço o mais rápido possível, e garantindo o acordo de nível de serviço, além da satisfação daqueles que o consomem. Já a gestão da monitoração é responsável por identificar e priorizar eventos relacionados aos serviços e seus componentes, sistematicamente observando, registrando e reportando suas mudanças [Limited 2019].

Segundo o ITIL [Limited 2019] 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, hoje 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.

3. Trabalhos Relacionados

Com o objetivo de reduzir o tempo de resolução de uma falha ou até mesmo atuar proativamente para que a falha não ocorra, alguns trabalhos têm aplicado aprendizado supervisionado para que modelos consigam identificar, anomalias em Sistemas de Informação. Du et al. [Du et al. 2017] propuseram um sistema de detecção de anomalias em aplicações distribuídas como o *Hadoop Distributed File System* (HDFS) e o *Open Stack* utilizando os seus registros (*logs*) de sistema. Aplicando redes *Long Short-Term Memory* (LSTM) e processamento textual, foi possível identificar padrões de uma execução normal e detectar anomalias quando os registros estavam fora do comum por meio do aprendizado supervisionado. Zhang et al. [Zhang et al. 2019], para solucionar o problema da variação dos *logs* ao longo do tempo, propuseram um modelo de detecção de anomalias usando redes neurais Bi-LSTMs com mecanismo de atenção, permitindo a captura de informações contextuais e, automaticamente, aprendendo a importância de diferentes eventos de *logs*. Zhang et al. [Zhang et al. 2016] propuseram uma abordagem para reduzir o espaço de características da técnica TF-IDF, que permitiu uma otimização no processo de detecção de anomalias, quando combinado com redes neurais profundas LSTMs.

Com a popularidade da computação em nuvem, alguns trabalhos têm abordado a predição de falhas neste ambiente. Gao et al. [Gao et al. 2020] implementaram um sistema de detecção falhas em tarefas na nuvem do Google, processando dados referentes ao consumo dos recursos como processador, memória e disco. O sistema utilizou uma arquitetura com redes neurais Bi-LSTMs de duas camadas combinada com uma camada de Regressão Logística, apresentando resultados melhores quando comparado a outros modelos como Rede Oculta de Markov e Máquinas de Vetores de Suporte (SVM, do inglês *Support Vector Machine*). Chen et al. [Chen et al. 2014] desenvolveram um preditor de falhas em tarefas da nuvem do Google, aplicando Redes Neurais Recorrentes, o que reduziu o consumo de recursos em 10%. Yuan et al. [Yuan et al. 2019], por meio do processamento dos *logs* da ferramenta Open Stack, propuseram uma abordagem de detecção de falhas nas execuções de provisionamento de recursos da ferramenta. A abordagem utiliza a técnica *Word2Vec*, e aplica algoritmos de classificação para comparar os resultados, obtendo o melhor resultado o algoritmo de Floresta Aleatória. Dai et al. [Dai Vu et al. 2021] propuseram um modelo de aprendizado com Redes Neurais Profundas Bi-LSTMs para prever falhas em *containers* no *Kubernetes*.

Esses trabalhos aplicam com excelência a detecção e a previsão de falhas em aplicações específicas. No entanto, nenhum trabalho aborda a visão do usuário ou mesmo consideram uma estratégia que englobe uma análise geral de um sistema composto por diversos subsistemas, no qual seja possível avaliar o serviço final entregue. Dessa forma, o presente trabalho propõe uma abordagem de detecção de falhas considerando a visão do usuário e uma visão geral de sistemas e seus subsistemas.

4. Estudo de Caso

Neste trabalho considera-se o modelo de referência CRISP-DM (*Cross Industry Standard Process for Data Mining*) [Chapman et al. 2000], que possibilita uma visão global do ciclo de vida de um projeto de mineração de dados. O modelo contém as fases do projeto, suas respectivas tarefas e a relação delas entre si. A Figura 1 apresenta as fases do modelo de referência CRISP-DM e os seus relacionamentos.

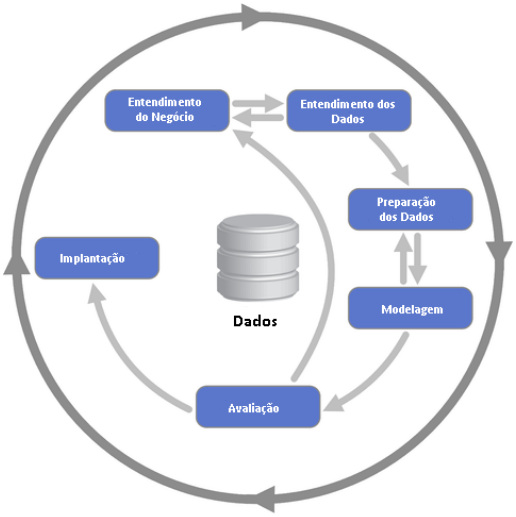

Figura 1. Fases do Modelo CRISP-DM (Adaptado de [Chapman et al. 2000]).

4.1. Entendimento do Negócio e Entendimento dos Dados

Na etapa de Entendimento do Negócio serão definidos os objetivos e as necessidades sob a perspectiva do negócio. Assim sendo, serão convertidas as necessidades para um problema de mineração de dados. Na etapa de Entendimento dos Dados será realizada a coleta, atividades de exploração para uma familiarização com os dados obtidos, e a formulação de hipóteses para modificar os dados.

4.1.1. Aplicativo Móvel Bancário

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, aplicação em investimentos, entre outros. Em novembro de 2020, o Banco Central do Brasil lançou um esquema de pagamentos instantâneos, denominado PIX. Esse esquema causou grande impacto no sistema financeiro, pois a sua quantidade de operações ultrapassou meios tradicionais de pagamentos, tais como cartões de débito, cartões de crédito, boletos e transferências [Duarte et al. 2022].

Ao longo deste trabalho foram coletados dados anonimizados referentes aos acessos dos clientes de uma organização financeira. Esses dados correspondem a quantidade de acessos a determinadas páginas, como a página de *login* e páginas de operações financeiras. Também foram coletados eventos gerados pelas ações dos clientes com o aplicativo, como mensagens dos sistemas, indicando o sucesso de operações ou erros. Por fim, foram utilizados os tempos de respostas de algumas operações e carregamento

de determinadas páginas. Tais dados correspondem a três dos quatro sinais de ouro [Beyer et al. 2016], que é estratégia de coleta de métricas implementada pelo time de engenharia de confiabilidade do Google. Essas métricas são referentes ao tráfego, aos erros e a latência dos serviços.

4.1.2. 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 melhoria contínua na gestão de incidentes.

Os registros de incidentes serão usados na etapa de rotulação dos dados, no qual serão mapeados os momentos de indisponibilidade do aplicativo bancário, e 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. Assim, os rótulos utilizados são referentes as falhas generalizadas no aplicativo. Tais falhas correspondem a problemas críticos na infraestrutura do sistema, que na maioria dos casos impede a autenticação do usuário e, consequentemente, o uso de todos os serviços relacionados as transações financeiras. Estes dados serão fundamentais para identificação de novas indisponibilidades pelo classificador proposto neste trabalho.

4.1.3. Extração e Transformação dos Dados

Os dados estavam armazenados em uma única tabela no Apache Hive. Inicialmente, foi realizada uma análise exploratória para identificar quais campos da tabela eram, de fato, significativos para o problema. Assim, foram selecionados campos que refletiam 3 das 4 métricas de ouro do time de engenharia de confiabilidade do Google, o tráfego, a taxa de erros e a latência. Esses campos eram compostos pelas URLs (*Uniform Resource Locator*) das páginas acessadas, texto das ações de eventos, e a categoria das ações dos eventos. As ações dos eventos representam ações executadas pelos usuários ou pelo sistema conforme a interação, tais como cliques em páginas ou mensagens dos sistema apresentadas ao usuário. A categoria das ações dos eventos representa a classe de cada ação como 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. Foram extraídos os 100 valores mais frequentes para os campos de URL e de ações de eventos, e os 50 valores mais frequentes para a categorias de ações de eventos. A partir dos valores mais frequentes de cada campo, foram agrupadas as quantidades de ocorrências de cada um dos 250 valores diferentes a cada um minuto.

A Figura 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. 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. Outro benefício é a anonimização

do conteúdo do texto para resguardar a organização.

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 2. Transformação dos dados.

4.2. Preparação dos Dados

Nesta etapa serão realizadas as tarefas para construir o conjunto final de dados. Tais tarefas incluem a limpeza dos dados, a construção de novas colunas, as transformações em geral, e a normalização dos dados, as quais são descritas a seguir.

4.2.1. Rotulação 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, foram buscados na base os incidentes da organização, e os períodos que houveram reclamações dos clientes referentes a problemas generalizados no acesso ao aplicativo móvel bancário.

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 3. Assim, para cada minuto no qual foram extraídas as ocorrências das páginas, eventos e categorias, foi rotulado o estado do serviço como disponível e não disponível, naquele momento.

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

Figura 3. Rotulação a partir da base de incidentes.

4.2.2. Seleção de Variáveis

Inicialmente foram construídas 250 variáveis a partir da base de dados da ferramenta de *Web Analytics*. Tais variáveis representam as interações mais frequentes dos usuários com o aplicativo móvel. No entanto, nem todas as variáveis possuem uma correlação forte com a disponibilidade do serviço e podem ter um impacto negativo no processo de classificação. Dessa forma, fez-se necessário aplicar técnicas de seleção de atributos para melhorar o desempenho dos modelos de Aprendizado de Máquina.

Assim, a seleção de variáveis foi aplicada por meio da filtragem de atributos. O método de filtragem ranqueia cada variável de acordo com sua pontuação em relação à variável objetivo. A filtragem foi aplicada por meio do método *SelectKBest* da biblioteca *Scikit-learn* [Pedregosa et al. 2011]. O algoritmo usa a função de pontuação *f_classif* que aplica a técnica de análise de variância estatística ANOVA, que é frequentemente usada em tarefas de classificação.

Em seguida, foram filtradas as variáveis com as *K* maiores pontuações, de forma que *K* representa a quantidade de atributos que se deseja filtrar. O valor ótimo de K foi encontrado por meio de um algoritmo de força bruta, o qual será descrito com mais detalhes na Seção 4.3.3.

4.2.3. Padronização

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. Muito autores validaram o seu impacto na melhoria de resultados [Singh and Singh 2020], [Mohamad and Usman 2013], [Raju et al. 2020]. Porém, apesar de apresentar uma melhoria em determinados algoritmos, a padronização pode também influenciar negativamente os resultados de outros algoritmos. Assim, foram comparados os resultados da aplicação e da ausência da padronização nos algoritmos avaliados neste estudo (Seção 5.1).

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 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.

$$z = \frac{x - u}{s} \tag{1}$$

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, no qual a padronização apresentou melhores resultados.

4.2.4. SMOTE

O desbalanceamento dos dados ocorre quando uma das classes dos rótulos possui uma representatividade menor em relação às demais classes. Tal ocorrência é comum em problemas de detecção de falhas e podem influenciar nos resultados dos modelos de classificação. Algumas estratégias são utilizadas para reduzir o desbalanceamento, como a repetição de valores da classe minoritária ou a remoção de valores da classe majoritária. Outra alternativa é a geração de amostras sintéticas SMOTE (*Synthetic Minority Over-Sampling Techinique*) [Chawla et al. 2002].

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 um valor aleatório entre 0 e 1, e adiciona esse valor ao vetor de atributos considerado [Chawla et al. 2002]. Assim, essa abordagem se demonstra mais eficiente na criação de novas amostras do que comparadas a outras estratégias de balanceamento. Também se optou por não utilizar a abordagem de remoção da classe majoritária para evitar a perda de informação.

Neste estudo foi avaliado o impacto da geração de amostras sintéticas nos algoritmos de classificação. Para implementação do SMOTE foi utilizada a biblioteca *Imblearn*. A classe minoritária foi incrementada de 0,41% para 1%, cujo o valor foi o melhor obtido em testes realizados com o conjunto de dados de treino e validação. Após obtida a melhor proporção, foram geradas as amostras sintéticas e os dados foram avaliados nos algoritmos de classificação.

4.3. 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.

4.3.1. Separação dos Dados e Seleção de Algoritmos

Ao longo do estudo foram avaliadas diferentes formas de separação dos dados. Inicialmente, os dados foram separados com embaralhamento, permitindo uma estratificação das classes em cada conjunto. Porém, foi verificado que o embaralhamento causou sobre ajuste (*overfitting*) nos modelos, visto que variáveis que possuíam uma baixa correlação com as indisponibilidades estavam sendo usadas na classificação para identificar os períodos. Dessa forma, foi descartado o embaralhamento, seguindo com a ordem cronológica dos dados.

Na sequência, os dados foram separados em 20% para teste e 80% para treinamento. Em tarefas de otimização o conjunto de treinamento foi dividido novamente em 65% para treinamento e 15% para validação, totalizando os 80%. Tal estratégia permitiu buscar melhores hiperparâmetros, sem enviesar os resultados devido as repetidas validações. As separações foram realizadas garantindo a sequência cronológica dos dados.

Para avaliar o conjunto de dados e as transformações realizadas foram selecionados alguns dos algoritmos mais comuns para a tarefa de classificação binária supervisionada. Dentre eles foram selecionados um algoritmo probabilístico, um não paramétrico, uma rede neural profunda, dois métodos compostos e dois algoritmos comuns. Dessa forma, foi possível avaliar como diferentes estratégias de Aprendizado de Máquina se comportam para detecção de falhas no conjunto de dados construído.

O Naive Bayes (NB) foi selecionado para representar um algoritmo probabilístico e o K-vizinho mais próximo (KNN) para representar um algoritmo não paramétrico. A Árvore de decisão e a Máquina de Vetores de Suporte (SVM) foram selecionadas para representar algoritmos comuns de classificação. Floresta Aleatória (RF) e o Adaboost foram selecionados para representar algoritmos compostos. Por fim, as redes LSTM foram selecionadas devido a sua capacidade de reconhecer eventos raros e representar as redes neurais. Cada modelo foi executado três vezes e calculada uma média simples dos seus resultados para avaliação.

4.3.2. Métricas de Avaliação

A escolha de métricas de avaliação são essenciais para medir o desempenho de cada modelo. A acurácia é uma métrica geral do modelo no qual os acertos são contabilizados independente 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 detecção de um falso positivo pode ser tão prejudicial quanto a detecção 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 eventos raros. 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, quando dois modelos obtiverem um *F1-Score* muito próximo.

4.3.3. Otimização dos Hiperparâmetros

Nesta etapa são utilizados os dados de treinamento e validação para encontrar os melhores hiperparâmetros para o conjunto de dados. A otimização de hiperparâmetros foi implementada nas etapas de definição do número de variáveis no método de filtragem, na definição do tamanho da classe minoritária no SMOTE, e em hiperparâmetros dos algoritmos de Aprendizado de Máquina. Essa etapa é fundamental para ajustar o pré-processamento dos dados e alguns dos algoritmos que possuem uma grande opção de hiperparâmetros.

Tabela 1. Arquitetura da Rede Neural.

Camada	Função de Ativação	Quantidade de Parâmetros
LSTM	ReLU	344832
Densa	ReLU	37056
Dropout (20%)		0
Densa	ReLU	37056
Densa	ReLU	37056
Densa	Sigmoid	386
Total de Parâmetros: 456,386		

A busca pelo valor ótimo de K na etapa de filtragem foi implementada por meio de um algoritmo de força bruta, no qual todos os valores possíveis de K foram testados. Para tal, foram avaliados os resultados do *F1-Score* do algoritmo de Máquina de Vetores de Suporte. O SVM foi selecionado para esta etapa porque demonstrou o melhor desempenho em relação aos demais algoritmos em experimentos iniciais. O valor ótimo de K encontrado na avaliação realizada foi de 19 atributos.

Os algoritmos foram aplicados a partir da biblioteca Sklearn [Pedregosa et al. 2011], com exceção da rede LSTM. Assim, foram avaliados todos os núcleos do algoritmo SVM, porém o núcleo padrão (rbf) apresentou o melhor resultado. A rede LSTM foi otimizada a partir do Keras Tuner [O'Malley et al. 2019], por meio do algoritmo de busca *Random Search*. Para isso, foram buscadas as funções de ativação, quantidade de unidades de processamento, tamanho do lote, taxa de aprendizado, e o uso ou não de uma camada de esquecimento. O modelo final da rede LSTM é apresentado na Tabela 1, no qual são descritas as camadas, as funções de ativação e a quantidade de parâmetros. Os valores para o tamanho do lote e taxa de aprendizado encontrados são de 256 e 0.000152, respectivamente. Apesar das redes LSTMs serem tipicamente utilizadas em séries temporais, para este estudo foi utilizado apenas um evento por vez.

5. Resultados Obtidos

Nesta seção serão avaliados os resultados obtidos nas análises de efeito realizadas entre as técnicas de engenharia de atributos aplicadas, e os melhores resultados dos algoritmos de classificação.

5.1. Análise de Efeito das Técnicas de Engenharia de Atributos

Diferentes técnicas de transformação de dados podem influenciar no desempenho dos algoritmos de Aprendizado de Máquina. Entretanto, as transformações possuem um impacto diferente para cada algoritmo, podendo inclusive, influenciar negativamente. Dessa forma, foi realizada a análise de efeito de cada técnica para as métricas de *F1-Score* e ROC AUC. A análise de efeito foi calculada conforme descrita por R. Jain [Jain 2008]. Tal análise indica o quanto cada técnica foi responsável pelo incremento do desempenho no algoritmo comparada a não utilização das técnicas.

Analisando a Tabela 2, para as três técnicas aplicadas individualmente, considerando o *F1-Score* como medida de avaliação, a filtragem foi a transformação de dados que obteve o maior impacto nos algoritmos testados. Por outro lado, para os algoritmos de K-vizinhos Mais Próximos e para as redes LSTM, a padronização dos dados obteve o maior impacto. Já para o algoritmo de Floresta Aleatória, o SMOTE foi técnica que individualmente obteve maior efeito. Porém, quando analisadas as técnicas em conjunto, foi possível verificar que a junção das três técnicas ocasionou um impacto maior para este algoritmo. Analisando a combinação de duas técnicas, a filtragem e a padronização apresentaram um efeito relevante para o algoritmo de K-vizinhos Mais Próximos.

Tabela 2. Análise de efeito das técnicas de engenharia de atributos no F1 Score.

Algoritmo	Filtragem	Padronização	SMOTE	FI+PA	FI+SM	PA+SM	FI+PA+SM
NB	**98.36%**	0.82%	0%	0.82%	0%	0%	0%
DT	**76.63%**	0.04%	14.58%	0.03%	7.49%	0.62%	0.61%
KNN	19.78%	**54.99%**	0.54%	23.86%	0.06%	0.01%	0.75%
RF	16.51%	1.71%	25.36%	5.49%	11.10%	12.72%	**27.11%**
ADABOOST	**96.78%**	0.08%	0.92%	0.08%	2.04%	0.06%	0.06%
SVM	**99.32%**	0%	0.04%	0%	0.04%	0.30%	0.30%
LSTM	37.96%	**50.63%**	0.08%	10.98%	0.08%	0.27%	0%

A análise de efeito das técnicas de engenharia de atributos para a ROC AUC, apresentada na Tabela 3, é bastante similar com os resultados apresentados para o *F1-Score*. A filtragem novamente foi a técnica que apresentou o maior efeito, com exceção para o algoritmo de K-vizinhos Mais Próximos, no qual a padronização obteve maior impacto. O SMOTE foi a técnica que individualmente menos refletiu no aumento da ROC AUC, não conseguindo atingir o objetivo de melhorar a diferenciação das classes com a sobre amostragem dos dados. Os resultados para as três técnicas em conjunto não obtiveram grande efeito, visto que o maior impacto foi para o algoritmo de Floresta Aleatória, com um efeito de 10% apenas.

Tabela 3. Análise de efeito das técnicas de engenharia de atributos na ROC AUC.

Algoritmo	Filtragem	Padronização	SMOTE	FI+PA	FI+SM	PA+SM	FI+PA+SM
NB	**37.81%**	31.10%	0%	31.10%	0%	0%	0%
DT	**95.48%**	0%	0.58%	0%	3.94%	0%	0%
KNN	12.73%	**53.39%**	0.09%	33.48%	0.21%	0.02%	0.08
RF	**66.64%**	0.17%	4.77%	1.94%	10.46%	6.01%	10%
ADABOOST	**99.93%**	0.01%	0.02%	0.01%	0%	0.01%	0.01%
SVM	**99.95%**	0.02%	0%	0.02%	0%	0%	0%
LSTM	**73.36%**	11.67%	0.99%	13.22%	0.47%	0.25%	0.04%

Os bons resultados da etapa de filtragem podem ser justificados pela remoção de variáveis irrelevantes ao problema que causam sobre ajuste (*overfitting*) aos modelos. Assim, o modelo no treinamento identifica padrões que considera relevante para o problema naquele momento, mas não são generalizáveis para outros períodos. Outra possível causa

é a redução da dimensionalidade para alguns algoritmos, que conseguem melhorar a performance com um número inferior de variáveis de entrada. Já a padronização apresentou o maior efeito para o algoritmo de K-Vizinho Mais Próximo, pois o seu algoritmo trabalha diretamente no cálculo das distâncias no plano amostral. Isso significa, quando as variáveis estão na mesma ordem de grandeza, o algoritmo consegue classificar um novo dado com mais acurácia.

5.2. Avaliação Final dos Algoritmos

Os melhores resultados da etapa de análise de efeito foram extraídos e estão apresentados na Figura 4. O SVM foi o algoritmo que apresentou o melhor *F1-Score* (0.954), e obteve o terceiro maior ROC AUC (0.989). Os algoritmos KNN, Floresta Aleatória e Adaboost apresentaram resultados próximos ao SVM, com um *F1-Score* variando de 0.915 a 0.884. A rede LSTM não obteve um resultado de destaque, apresentando o quinto melhor resultado para o *F1-Score* no experimento realizado.

O Algoritmo de Naive Bayes demonstrou o maior resultado para a ROC AUC, com 0.998. No entanto, o seu *F1-Score* obteve um valor baixo comparado aos demais. Tal divergência nas métricas é causada porque o algoritmo possui uma alta precisão e consegue acertar a grande maioria das falhas. Porém, apresenta um número alto de falsos positivos, o que acaba reduzindo o valor do *F1-Score*. Já a árvore de decisão é o algoritmo que pior performou analisando os resultados, com os menores valores para as duas métricas avaliadas.

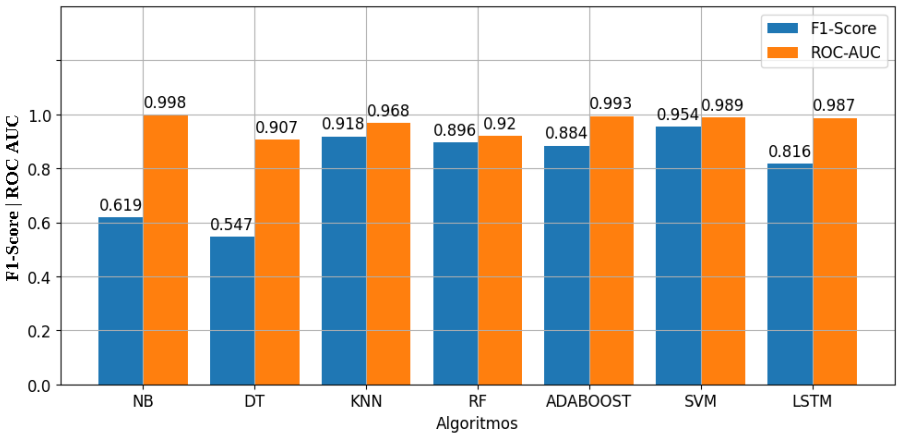

Figura 4. Resultados Obtidos.

O SVM gera diversas linhas 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. Tais características justificam a capacidade desse algoritmo de potencializar funções matemáticas, responsáveis pelos bons resultados de classificação apresentados.

6. Conclusões

Neste trabalho foi proposta uma abordagem para detecção de falhas em um aplicativo móvel bancário por meio de dados das interações dos usuários. Os dados foram extraídos, rotulados e transformados para que pudessem alimentar diferentes modelos de

Aprendizado de Máquina. Assim sendo, foi possível analisar os efeitos de três técnicas de transformação de dados ao aplicar os modelos.

A filtragem foi a técnica de engenharia de atributos que maior impactou nos resultados dos modelos, seguida pela padronização dos dados. O SVM apresentou o melhor *F1-Score* na classificação dos períodos avaliados, demonstrando que têm um bom desempenho na detecção de eventos raros. Obteve-se uma ROC AUC próxima a 1, reforçando que o modelo consegue diferenciar as classes do conjunto de dados. Também foi possível validar o uso de dados provenientes de uma ferramenta de *Web Analytics* para alimentar modelos de Aprendizado de Máquina.

Ao final foi criado um modelo de classificação que avalia as interações dos usuários e fornece uma informação única da disponibilidade de um serviço distribuído e composto por diversos sistemas. O modelo será utilizado como ferramenta de monitoração contínua, contribuindo para o processo de GSTI e em práticas do ITIL. A abordagem apresentada é composta majoritariamente por processos automatizáveis (com exceção da rotulação dos dados e da otimização), permitindo a fácil adaptação em outras organizações. Para trabalhos futuros se planeja avaliar o uso de técnicas de otimização nos demais algoritmos utilizados.

Referências

Afshan, S. and Sharif, A. (2016). Acceptance of mobile banking framework in pakistan. *Telematics and Informatics*, 33(2):370–387.

Beyer, B., Jones, C., Petoff, J., and Murphy, N. R. (2016). *Site reliability engineering: How Google runs production systems*. "O'Reilly Media, Inc.".

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R., et al. (2000). Crisp-dm 1.0: Step-by-step data mining guide. *SPSS inc*, 9(13):1–73.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Chen, X., Lu, C.-D., and Pattabiraman, K. (2014). Failure prediction of jobs in compute clouds: A google cluster case study. In *2014 IEEE International Symposium on Software Reliability Engineering Workshops*, pages 341–346. IEEE.

Dai Vu, D., Vu, X. T., and Kim, Y. (2021). Deep learning-based fault prediction in cloud system. In *2021 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1826–1829. IEEE.

Du, M., Li, F., Zheng, G., and Srikumar, V. (2017). Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 1285–1298.

Duarte, A., Frost, J., Gambacorta, L., Koo Wilkens, P., and Shin, H. S. (2022). Central banks, the monetary system and public payment infrastructures: lessons from brazil's pix. *Available at SSRN 4064528*.

Galup, S. D., Dattero, R., Quan, J. J., and Conger, S. (2009). An overview of it service management. *Communications of the ACM*, 52(5):124–127.

Gao, J., Wang, H., and Shen, H. (2020). Task failure prediction in cloud data centers using deep learning. *IEEE transactions on services computing*.

Jain, R. (2008). *The art of computer systems performance analysis*. john wiley & sons.

Limited, A. (2019). *ITIL foundation: ITIL 4 edition*. TSO (The Stationery Office), ein Unternehmen von Williams Lea.

Malaquias, R. F. and Hwang, Y. (2019). Mobile banking use: A comparative study with brazilian and us participants. *International Journal of Information Management*, 44:132–140.

Meirelles, F. d. S. (2022). Pesquisa anual do uso de ti. *Fundação Getúlio Vargas*.

Mohamad, I. B. and Usman, D. (2013). Standardization and its effects on k-means clustering algorithm. *Research Journal of Applied Sciences, Engineering and Technology*, 6(17):3299–3303.

Munir, M., Siddiqui, S. A., Dengel, A., and Ahmed, S. (2018). Deepant: A deep learning approach for unsupervised anomaly detection in time series. *Ieee Access*, 7:1991–2005.

O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L., et al. (2019). Kerastuner. `https://github.com/keras-team/keras-tuner`.

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

Phippen, A., Sheppard, L., and Furnell, S. (2004). A practical evaluation of web analytics. *Internet Research*.

Raju, V. G., Lakshmi, K. P., Jain, V. M., Kalidindi, A., and Padma, V. (2020). Study the influence of normalization/transformation process on the accuracy of supervised classification. In *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pages 729–735. IEEE.

Singh, D. and Singh, B. (2020). Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, 97:105524.

Yuan, Y., Shi, W., Liang, B., and Qin, B. (2019). An approach to cloud execution failure diagnosis based on exception logs in openstack. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, pages 124–131. IEEE.

Zhang, K., Xu, J., Min, M. R., Jiang, G., Pelechrinis, K., and Zhang, H. (2016). Automated it system failure prediction: A deep learning approach. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 1291–1300. IEEE.

Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., Xie, C., Yang, X., Cheng, Q., Li, Z., et al. (2019). Robust log-based anomaly detection on unstable log data. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 807–817.