Curso de Ciência da Computação
Universidade Estadual de Mato Grosso do Sul

AUTENTICAÇÃO CONTÍNUA COM USO DE PADRÕES DE DIGITAÇÃO

Sthefany Conceição Duquini

Prof. Dr. Fabrício Sérgio de Paula (Orientador)

Dourados - MS
2023

AUTENTICAÇÃO CONTÍNUA COM USO DE PADRÕES DE DIGITAÇÃO

Sthefany Conceição Duquini

Este exemplar corresponde à redação final da monografia da disciplina Projeto Final de Curso, devidamente corrigida e defendida por Sthefany Conceição Duquini e aprovada pela Banca Examinadora, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Dourados, 17 de Outubro de 2023.

Prof. Dr. Fabrício Sérgio de Paula (Orientador)

D949a Duquini, Sthefany Conceição

Autenticação contínua com uso de padrões de digitação / Sthefany Conceição Duquini – Dourados, MS: UEMS, 2023.
74 p.

Trabalho de Conclusão de Curso (Graduação) - Ciência da Computação - Universidade Estadual de Mato Grosso do Sul (UEMS), 2023.

Orientador: Prof.º Dr.º Fabrício Sérgio de Paula

1. Inteligência artificial - Técnicas de aprendizado de máquina 2. Algoritmo de aprendizado - One-Class Support Vector Machine (OCSVM) 3. Computação - Medidas de segurança - Autenticação contínua 4. Dispositivos móveis - Android I. Paula, Fabrício Sérgio de. II. Título.

CDD 23 ed. 006.3

Ficha Catalográfica elaborada pela bibliotecária da Universidade Estadual de Mato Grosso do Sul (UEMS) Aline Perdomo Soutelo, inscrita sob o CRB Provisório n° 714 - 1ª Região.

Curso de Ciência da Computação
Universidade Estadual de Mato Grosso do Sul

AUTENTICAÇÃO DE USUÁRIO ATRAVÉS DE PADRÕES DE DIGITAÇÃO

Sthefany Conceição Duquini

Outubro de 2023

Banca Examinadora:

Prof. Dr. Fabrício Sérgio de Paula (Orientador)
Área de Computação – UEMS

Prof. Dr. Ricardo Luís Lachi
Área de Computação – UEMS

Prof. Dr. Cleber Valgas Gomes Mira
Área de Computação – UEMS

RESUMO

Neste projeto, foi feita uma exploração da detecção de anomalias em dispositivos móveis por meio da técnica One-Class Support Vector Machine (OCSVM), implementada em um aplicativo de simulação de mensagens. Os dados de digitação foram coletados em dispositivos Android usando um teclado personalizado, os quais foram utilizados para treinar o modelo e realizar verificações de usuários. Embora tenha sido alcançada uma taxa satisfatória na identificação de usuários legítimos, identificou-se a necessidade de aprimoramentos para lidar com padrões de digitação semelhantes. Durante o desenvolvimento do projeto houve uma busca por conhecimentos sobre a arquitetura de dispositivos móveis, bem como sobre a coleta e análise de dados em Android. Foram estudadas também técnicas avançadas de aprendizado de máquina, com foco especial na compreensão da técnica OCSVM, além do desenvolvimento prático no ambiente do Android Studio para a criação do aplicativo.

SUMÁRIO

1. INTRODUÇÃO

Dispositivos móveis, como *smartphones* e *tablets*, se tornaram ferramentas essenciais em nosso cotidiano. Conquistaram uma ampla base de usuários e enorme fatia do mercado econômico, sob diversas marcas e modelos com uma variedade de recursos (Haris et al, 2017). No entanto, à medida que nos tornamos cada vez mais dependentes desses dispositivos, surgem preocupações significativas relacionadas à segurança cibernética.

Um dos principais desafios em termos de segurança em dispositivos móveis é o roubo e a perda desses aparelhos. Quando um dispositivo é roubado, as informações pessoais armazenadas nele ficam vulneráveis. Dados sensíveis, como informações bancárias, senhas e documentos pessoais, podem ser explorados por indivíduos não autorizados. Isso pode resultar na violação da privacidade do usuário e até mesmo em consequências mais graves, como roubo de identidade e acesso indevido a contas pessoais (Android, 2023).

Além do roubo físico, a perda de dispositivos móveis também representa um risco significativo. Se um aparelho é extraviado, todas as informações armazenadas nele podem ser acessadas por terceiros, comprometendo a segurança e privacidade dos dados. Esse tipo de situação pode ter impactos adversos na vida pessoal e profissional do usuário.

Outro desafio comum em dispositivos Android diz respeito aos ataques de *malwares* e aplicativos maliciosos. Dispositivos móveis, especialmente aqueles que utilizam o sistema operacional Android, devido à sua ampla base de usuários, são frequentemente alvos de ataques cibernéticos. Esses ataques podem ocorrer por meio da instalação de aplicativos de fontes não confiáveis, resultando em acesso não autorizado a dados pessoais, espionagem e danos ao dispositivo.

Além disso, a engenharia social é uma tática amplamente utilizada para obter acesso não autorizado aos dispositivos móveis. Os cibercriminosos se aproveitam da falta de conscientização dos usuários e do descuido ao lidar com mensagens suspeitas. Táticas como *phishing*, mensagens falsas e *links* maliciosos são empregadas para enganar os usuários e obter informações confidenciais (UFRJ, 2021). Diante desses desafios, é crucial implementar medidas de proteção e autenticação em dispositivos móveis. Métodos de autenticação contínua em aplicativos tem se mostrado uma abordagem eficaz, envolvendo a monitorização constante do comportamento do usuário (Kaspersky, 2023). Isso inclui a análise de padrões de digitação, gestos e movimentos do dispositivo, além de outros fatores que possam ajudar a validar a identidade do usuário. Essa abordagem é particularmente importante em dispositivos

móveis, onde as tarefas são realizadas rapidamente e as medidas de autenticação tradicionais, como senhas e códigos PIN, podem não ser suficientes para garantir a segurança (Android, 2023).

Em resumo, a segurança em dispositivos móveis é um desafio constante que exige atenção e adoção de medidas de proteção adequadas. Os usuários devem estar cientes dos riscos e adotar práticas seguras, como o uso de senhas fortes, a atualização regular do sistema operacional e a instalação de aplicativos apenas de fontes confiáveis. Além disso, as empresas de tecnologia devem continuar investindo em melhorias de segurança e na implementação de técnicas avançadas, como análise de comportamento do usuário e detecção de anomalias, para garantir a segurança dos dispositivos móveis e proteger as informações pessoais dos usuários.

Para mitigar esses riscos, é crucial adotar medidas de segurança adequadas. Manter os dispositivos móveis atualizados com as últimas atualizações de segurança é fundamental para combater as vulnerabilidades conhecidas. Além disso, é importante utilizar senhas fortes, autenticação de dois fatores e aplicativos de segurança confiáveis para proteger os dispositivos e dados pessoais.

Diante dos desafios na segurança de dispositivos móveis, este projeto tem como objetivo desenvolver um método de autenticação contínua. Para simular uma situação real em que o método será aplicado, foi criado um aplicativo de mensagens em linguagem Java. O método de autenticação em questão se baseia na análise dos padrões de digitação, incluindo o tempo de pressionamento das teclas, a pressão aplicada e a área ocupada nas teclas. A abordagem principal utiliza técnicas de aprendizado de máquina para análise de padrões, com foco na detecção de anomalias, a fim de identificar características únicas do usuário. Essa estratégia permite a validação contínua da autenticidade do usuário durante o uso do aplicativo de mensagens, buscando resultados que reflitam em uma melhora significativa na segurança contra perda ou roubo. A lógica de captura de eventos é acionada em resposta direta às interações do usuário, minimizando o tempo entre a ação do usuário e a coleta dos dados correspondentes.

Esse método busca superar as limitações dos métodos tradicionais de autenticação, como senhas e biometria estática, que podem ser vulneráveis a ataques e comprometimento da segurança. Ao analisar o comportamento do usuário em tempo real, levando em consideração aspectos como padrões de digitação, movimentos do dispositivo e características de uso, o

sistema de autenticação contínua visa garantir uma maior confiabilidade e segurança na verificação da identidade do usuário.

A implementação desse método exige o desenvolvimento de um aplicativo móvel dedicado, que irá coletar os dados relevantes do comportamento do usuário. Além disso, é necessário treinar um algoritmo de aprendizado de máquina, a fim de reconhecer e classificar os padrões comportamentais identificados. A integração desse sistema de autenticação contínua com outros aplicativos permite a sua utilização de forma transparente e conveniente para o usuário, sem comprometer a experiência de uso do dispositivo móvel.

Dessa forma, esse projeto busca contribuir significativamente para a melhoria da segurança em dispositivos móveis, fornecendo um método de autenticação contínua mais robusto e eficiente. A análise comportamental baseada em redes neurais oferece uma abordagem promissora para mitigar os riscos associados à autenticação em dispositivos móveis, proporcionando uma maior proteção contra acessos não autorizados e potenciais ataques cibernéticos.

Este trabalho está organizado como segue:

- O Capítulo 2 faz uma revisão teórica envolvendo o sistema operacional Android e o aprendizado de máquina;
- O Capítulo 3 apresenta o método de autenticação contínua desenvolvido e os resultados obtidos;
- Ao final é apresentada a conclusão deste trabalho.

2. REFERENCIAL TEÓRICO

2.1 Sistema operacional Android

A plataforma foi originalmente criada pela Android Inc., que mais tarde foi adquirida pelo Google e lançada como o AOSP (*Android Open Source Project*) em 2007 (Gilski et al., 2015). Esse anúncio foi acompanhado pela fundação da OHA (*Open Handset Alliance*), um consórcio dedicado ao desenvolvimento e distribuição do Android.

O *software* foi lançado sob a licença Apache como uma licença de código aberto gratuita. Graças ao rápido desenvolvimento, uma nova versão principal do Android é lançada a cada poucos meses. A arquitetura em camadas do Android, juntamente com o suporte de uma comunidade de desenvolvedores ativa, tem permitido a expansão contínua e aprimoramento das funcionalidades da plataforma ao longo dos anos (Android, 2023).

A arquitetura do sistema operacional Android, chamada também de arquitetura AOSP é organizada em formato de pilha, a **Figura 2.1** representa essa organização.

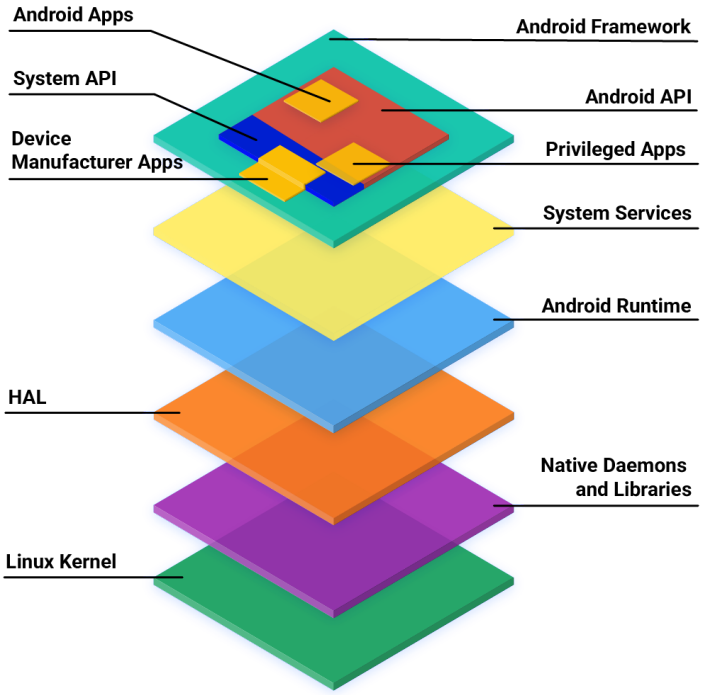

Figura 2.1: Arquitetura em pilha de software AOSP. Fonte: (Android, 2023)

A seguir, temos as definições de cada camada da pilha, todas de acordo com a documentação oficial do Sistema Operacional Android (Android, 2023):

- Aplicativo Android: Desenvolvido exclusivamente com a API do Android, geralmente disponível na Google Play Store, ou por meio de outras fontes. Em certos casos, fabricantes de dispositivos pré-instalam aplicativos Android para suportar funcionalidades específicas.
- Aplicativo privilegiado: Desenvolvido com uma combinação de APIs do Android e do sistema, esses aplicativos são pré-instalados como aplicativos privilegiados em um dispositivo.
- Aplicativo do fabricante do dispositivo: Criado com a utilização da API do Android, API do sistema e acesso direto à estrutura do Android. Tais aplicativos, por acessarem diretamente APIs instáveis na estrutura do Android, devem ser pré-instalados no dispositivo e podem ser atualizados apenas com as atualizações do software do sistema.
- API do sistema: APIs do Android disponíveis somente para parceiros e OEMs, marcadas como @SystemApi no código-fonte.
- API do Android: API publicamente disponível para desenvolvedores de aplicativos Android terceirizados.
- Estrutura do Android: Composta por classes Java, interfaces e códigos pré-compilados, sobre os quais os aplicativos são construídos. Partes da estrutura são acessíveis publicamente através da API do Android, enquanto outras são restritas aos OEMs via APIs do sistema, executando o código dentro do processo de um aplicativo.
- Serviços do sistema: Componentes modulares, como *system_server*, SurfaceFlinger e MediaService, que se comunicam com a funcionalidade exposta pela API da estrutura do Android para acessar o hardware subjacente.
- Tempo de execução do Android (ART): Ambiente de tempo de execução Java fornecido pelo AOSP, traduzindo o *bytecode* do aplicativo em instruções específicas do processador executadas no ambiente de tempo de execução do dispositivo.
- Camada de abstração de *hardware* (HAL): Interface padrão para os fornecedores de hardware implementarem, permitindo que o Android seja agnóstico sobre

implementações de driver de nível inferior, facilitando a implementação de funcionalidades sem modificar o sistema de nível superior.

- *Daemons* e bibliotecas nativas: *Daemons* incluem *init, healthd, logd* e *storaged,* interagindo diretamente com o *kernel* ou outras interfaces, e bibliotecas nativas, como libc, liblog, libutils, libbinder e libselinux, que não dependem de uma implementação HAL baseada no espaço do usuário.
- Núcleo: Parte central do sistema operacional, comunicando-se com o hardware subjacente em um dispositivo, muitas vezes dividido em módulos independentes de *hardware* e módulos específicos do fornecedor para uma operação eficaz e um gerenciamento adequado dos recursos do sistema.

2.2 Segurança cibernética em dispositivos móveis Android

O Android é projetado com uma arquitetura de segurança robusta, proporcionando uma plataforma flexível que protege a confidencialidade, integridade e disponibilidade dos dados e aplicativos. A equipe de segurança do Android trabalha para identificar possíveis vulnerabilidades em aplicativos e fornece atualizações de segurança para garantir a proteção contínua dos dispositivos (Android, 2023).

A plataforma oferece transparência e controle sobre as permissões dos aplicativos para os usuários, reduzindo riscos comuns de ataques, como engenharia social e *malware*. Além disso, são fornecidos patches de segurança para dispositivos Android que recebem atualizações regulares.

A ilustração na **Figura 2.2** mostra a estrutura de segurança, com a ênfase na restrição do código acima do *kernel* Linux pelo *Application Sandbox,* exceto por trechos de código do sistema operacional Android em execução como *root* (Android, 2023).

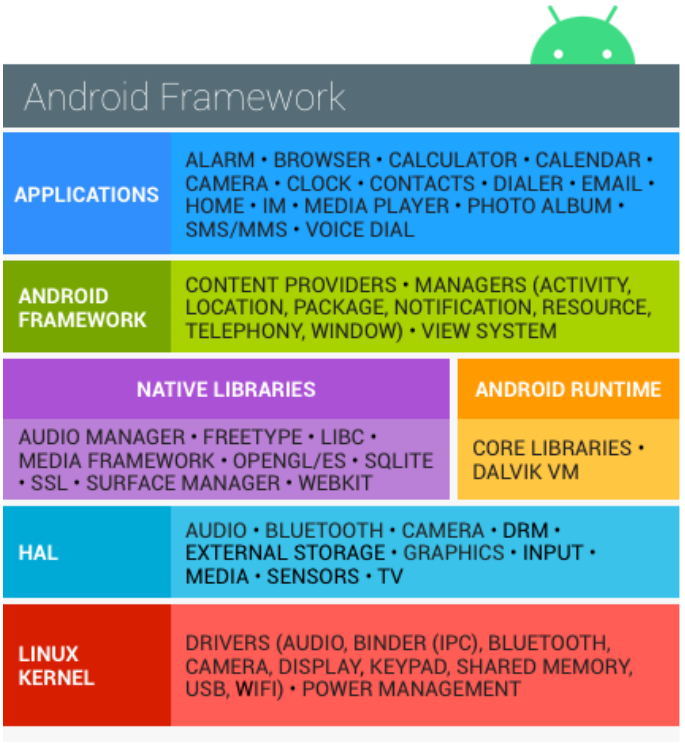

Figura 2.2: Pilha de software e as considerações dos vários níveis da pilha de software android. Fonte: (Android, 2023)

2.2.1 Visão geral do Programa de Segurança

O Programa de segurança do Android opera por meio de uma série de medidas fundamentais para garantir a integridade da plataforma, as medidas abaixo foram retiradas da documentação oficial do Sistema Operacional Android (Android, 2023):

2.2.1.1 Revisão de design

Desde as fases iniciais de desenvolvimento, a segurança é priorizada por meio da integração de controles de segurança específicos no modelo e *design* do Android, assegurando que cada recurso principal seja minuciosamente avaliado e revisado por especialistas em engenharia e segurança.

2.2.1.2 Teste de penetração e revisão de código

Durante o processo de desenvolvimento, todos os componentes de código aberto são submetidos a extensas revisões de segurança, realizadas não apenas pela equipe de segurança do Android, mas também por consultores independentes, com o objetivo de identificar

possíveis vulnerabilidades antes dos lançamentos principais e simular diferentes tipos de análises de segurança.

2.2.1.3 Revisão da comunidade e código aberto

A plataforma Android permite uma revisão de segurança abrangente por parte de todos os interessados, incluindo uma análise aprofundada de tecnologias de código aberto, como o *kernel* do Linux. Além disso, o Google Play oferece um espaço para os usuários e empresas fornecerem *feedback* direto sobre aplicativos específicos, promovendo assim uma comunidade de colaboração em prol da segurança.

2.2.1.4 Resposta a incidentes

Apesar das medidas preventivas, problemas de segurança podem surgir após o lançamento. Nesse sentido, o Android implementou um processo abrangente de resposta a incidentes, que inclui monitoramento contínuo, revisão de bugs e uma abordagem ágil para mitigar quaisquer vulnerabilidades identificadas, a fim de minimizar os riscos potenciais para todos os usuários.

2.2.1.5 Atualizações de segurança mensais

Com o compromisso de assegurar a proteção contínua dos dispositivos Android, a equipe de segurança do Android fornece atualizações mensais para dispositivos Google Android, assim como para seus parceiros de fabricação de dispositivos, garantindo a manutenção de um ambiente seguro e confiável para os usuários.

2.2.2 Serviço de segurança do Google

Para os dispositivos Android compatíveis com os Serviços Móveis do Google, a empresa disponibiliza uma gama de serviços baseados na nuvem. Alguns dos principais serviços de segurança do Google incluem (Android, 2023):

2.2.2.1 Google Play

Uma plataforma que permite aos usuários descobrir, instalar e adquirir aplicativos em dispositivos Android e na *web*. O Google Play também oferece serviços de segurança, como

revisão pela comunidade, verificação de licença de aplicativos e análise de segurança de aplicativos.

2.2.2.2 Atualizações do Android

Este serviço garante que dispositivos Android selecionados recebam regularmente atualizações de segurança e novos recursos, seja por meio da *web* ou de atualizações *over-the-air* (OTA).

2.2.2.3 Serviços de Aplicativos

Frameworks que permitem que aplicativos Android usem recursos de nuvem, incluindo *backup* de dados e configurações de aplicativos, bem como mensagens de nuvem para dispositivo (C2DM) para entrega de mensagens *push.*

2.2.2.4 Verificação de Aplicativos

Um sistema que alerta ou bloqueia automaticamente a instalação de aplicativos maliciosos e realiza verificações contínuas de segurança no dispositivo para detectar e remover aplicativos prejudiciais.

2.2.2.5 SafetyNet

Um sistema de detecção de intrusões que protege a privacidade, ajudando o Google a identificar e mitigar ameaças de segurança conhecidas, além de descobrir novas ameaças.

2.2.2.6 Atestado SafetyNet

Uma API de terceiros que verifica a compatibilidade do dispositivo com os padrões de teste de compatibilidade (CTS). Além disso, o atestado é capaz de identificar aplicativos Android que se comunicam com servidores de aplicativos.

2.2.2.7 Gerenciador de Dispositivos Android

Tanto um aplicativo da web quanto um aplicativo Android que auxilia na localização de dispositivos perdidos ou roubados.

2.3 Ameaças à segurança

Apesar dos cuidados para com a segurança em dispositivos Android, por parte dos desenvolvedores, estes aparelhos não estão livres de ameaças (Mayhofer, René et al., 2020). A seguir estão as principais ameaças e vulnerabilidades.

2.3.1 Malware

O crescimento no número de dispositivos móveis impulsionou uma verdadeira explosão de *malwares*, sendo os dispositivos Android particularmente vulneráveis a esse fenômeno (Malwarebytes, 2023). Entre os tipos de malwares que se disseminam neste ambiente, destacam-se os *Adwares, Spywares, Ransomwares e Trojans,* dentre outros (Knoll, Florian, 2019). Em termos concisos, um malware é definido como qualquer software que possui intenções maliciosas. Para uma compreensão mais aprofundada, a seguir, forneceremos uma análise mais detalhada dos principais tipos de malwares:

- Vírus: É um pedaço de código que possui a habilidade de se replicar, Os vírus se propagam com a ajuda de documentos *scripts*, e vulnerabilidades em aplicações *web* (Malwarebytes, 2023).
- *Worm*: Um *worm* é um tipo de *software* malicioso que tem a capacidade de se replicar rapidamente e se espalhar entre dispositivos em uma rede (CISCO, 2023). Conforme se propaga, um *worm* consome uma grande quantidade de largura de banda, sobrecarregando os sistemas infectados e muitas vezes tornando-os instáveis ou inacessíveis. Além disso, *worms* podem modificar e excluir arquivos, além de introduzir outros tipos de *malware* (CISCO, 2023).
- *Spywares e Adwares: Spywares* são programas projetados para rastrear todas as atividades do computador, desde quais aplicativos são usados até quais sites são visitados (Cornell University, 2023). *Adwares* são programas que exibem vários anúncios e ofertas, muitas vezes baseados nos sites visitados (Cornell University, 2023). Esses anúncios são independentes das janelas pop-up que aparecem ao visitar sites específicos.
- *Trojan*: Aplicação que se disfarça de benigna para atrair vítimas. Os *Trojans*, ou cavalos de Tróia, são programas maliciosos que executam ações não autorizadas pelo usuário. Essas ações podem incluir a exclusão, bloqueio, modificação ou cópia de dados, bem como a interrupção do desempenho de computadores ou redes de computadores. Ao contrário de vírus de computador e *worms*, os cavalos

de Troia não possuem a capacidade de se auto-replicar (University of Colorado, 2023).

- *Ransomwares: Ransomware* é uma forma de software malicioso que bloqueia os dados ou o dispositivo da vítima, ameaçando mantê-los bloqueados, a menos que um resgate seja pago ao atacante (IBM, 2023).

2.3.2 Softwares desatualizados

As atualizações do sistema do Google representam um componente essencial para aprimorar a experiência do usuário e garantir a segurança e estabilidade dos dispositivos Android. Com a evolução constante da tecnologia e a crescente sofisticação das ameaças cibernéticas, a atualização regular do sistema operacional desempenha um papel fundamental na proteção contra vulnerabilidades e na implementação de medidas de segurança avançadas (Google, 2023). Segundo a plataforma Google, priorizar as atualizações de aplicativos é crucial para que os usuários possam acessar novos recursos, aprimorar a segurança do aplicativo e garantir estabilidade geral (Google, 2023).

Essas atualizações abrangem uma ampla gama de dispositivos, desde *smartphones* e *tablets* até dispositivos para Google TV, Android TV, veículos com Android Auto, dispositivos Wear OS e dispositivos Chrome OS. Ao implementar atualizações regulares do sistema, o Google visa melhorar vários serviços essenciais, concentrando-se especialmente na segurança e privacidade dos usuários. A proteção da integridade do dispositivo é uma prioridade, e recursos como a localização de dispositivos perdidos oferecem tranquilidade aos usuários em caso de extravio ou roubo. Além disso, as atualizações do sistema auxiliam os usuários na concessão e gerenciamento de permissões, permitindo um controle mais preciso sobre os dados e informações compartilhados com os aplicativos no dispositivo (Google, 2023).

Em suma, a ausência de atualizações regulares pode deixar os dispositivos Android vulneráveis a potenciais ameaças e ataques maliciosos, enfatizando a importância crítica das atualizações do sistema do Google para manter a segurança e estabilidade dos dispositivos.

2.3.3 Acesso físico não autorizado

Diferentemente de computadores e servidores, os aparelhos móveis estão constantemente expostos a roubo ou acesso indevido devido à sua principal característica de portabilidade e uso em diversas situações. Segundo o jornal O GLOBO (Castro 2022), apenas

em 2022, cerca de um milhão de aparelhos móveis foram furtados no Brasil. Uma vez em posse do dispositivo, o acesso a aplicativos sensíveis pode ser obtido, concedendo ao indivíduo acesso a informações pessoais, como fotos, mensagens, documentos e outros tipos de dados sensíveis.

A vulnerabilidade dos dispositivos móveis é ainda mais agravada pela proliferação de dados pessoais armazenados nos dispositivos. Com o uso cada vez mais comum de dispositivos móveis para uma variedade de finalidades, desde comunicação até transações bancárias, os riscos associados ao roubo de dispositivos móveis vão muito além do custo do próprio dispositivo. A perda ou roubo de um dispositivo móvel pode resultar em consequências significativas, incluindo violações de privacidade, roubos de identidade e acesso não autorizado a contas pessoais e profissionais.

2.4 Aprendizado de máquina

A aprendizado de máquina, também conhecido como Machine Learning (ML), representa uma área crucial da inteligência artificial (IA) e da ciência da computação (IBM, 2023), cujo objetivo é desenvolver técnicas computacionais relacionadas ao aprendizado e construir sistemas capazes de adquirir conhecimento de forma automática (IBM, 2023). O termo "aprendizado de máquina" concentra-se na utilização de dados e algoritmos para imitar a maneira como os humanos aprendem, aprimorando gradualmente sua precisão (Escovedo, 2020). Nos últimos vinte anos, esses avanços tecnológicos na área de machine learning possibilitaram a criação de produtos inovadores, notavelmente algoritmos de recomendação e veículos autônomos (Mahesh, 2018).

Esses desenvolvimentos foram impulsionados pelo constante aprimoramento da capacidade de processamento e armazenamento de dados, permitindo que as máquinas aprendam e tomem decisões cada vez mais sofisticadas. A aprendizagem de máquina (ML) é uma disciplina que utiliza esses avanços para ensinar máquinas a lidar de forma mais eficiente com os dados. Em situações em que pode ser desafiador extrair informações significativas após uma análise dos dados, a aprendizagem de máquina se torna fundamental. Com a disponibilidade abundante de conjuntos de dados, a demanda por essa tecnologia está em ascensão em diversas indústrias (Escovedo, 2020).

Muitos setores aplicam a aprendizagem de máquina para extrair insights relevantes e acionáveis de seus dados, impulsionando a tomada de decisões baseadas em dados. O propósito essencial da aprendizagem de máquina é capacitar as máquinas a aprenderem de forma autônoma a partir dos dados que encontram. Inúmeras pesquisas foram conduzidas para avançar nessa área, visando tornar as máquinas capazes de aprenderem por si mesmas, sem a necessidade de programação explícita. Essa evolução nas técnicas de aprendizagem de máquina não só permite que as máquinas lidem com os dados de maneira mais eficaz, mas também as capacita a melhorar continuamente seu desempenho e se adaptar a novos desafios em um cenário de dados em constante mutação (Monard et al., 2003).

À medida que a tecnologia continua a evoluir, a aprendizagem de máquina se torna uma ferramenta essencial para aproveitar ao máximo o potencial dos dados em nosso mundo digital. Existem três tipos principais de métodos de machine learning, sendo eles o aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço (Ludemir, 2021).

2.4.1 Aprendizado de máquina supervisionado

Algoritmos de aprendizagem supervisionada estabelecem conexões entre entradas e saídas por meio de dados rotulados, chamados *datasets*. Os *datasets* são representados na forma de pares ordenados (entrada - saída desejada). Nesse contexto, o usuário fornece ao algoritmo pares de entradas e saídas conhecidas, normalmente apresentados como vetores. A cada saída é associado um rótulo, que pode ser um valor numérico ou uma classe. O algoritmo, por sua vez, busca identificar uma maneira de prever o rótulo de saída com base em uma entrada específica (Fontana, 2020)(Coutinho, 2019).

A aprendizagem supervisionada resolve problemas conhecidos e usa um conjunto de dados rotulados para treinar um algoritmo para realizar tarefas específicas. Os dados de entrada podem ser divididos em dois grupos (Coutinho, 2019):

- Conjunto X, composto por atributos ou características a serem empregados na definição da classe de resultado;
- Conjunto Y, englobando o atributo cujo valor de saída se pretende predizer, seja ele categórico ou numérico.

É aconselhável dividir os dados de entrada (*dataset*) em dois grupos distintos: o conjunto de treinamento, que será utilizado para construir o modelo, e o conjunto de teste, que será empregado para avaliar o desempenho do modelo em dados não utilizados anteriormente.

A aprendizagem supervisionada permite que os algoritmos "aprendam" a partir de dados históricos/de treinamento e os apliquem a entradas desconhecidas para derivar a saída correta (Tibco, 2023). A **Figura 2.4.1** ilustra o funcionamento de um modelo de aprendizagem supervisionado.

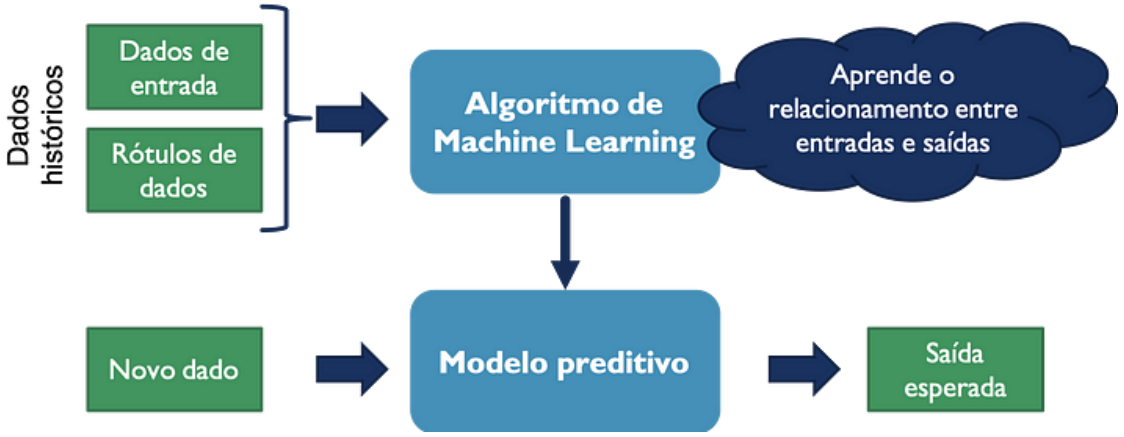

Figura 2.4.1: Esquema de aprendizado supervisionado. Fonte: (Escovedo 2020)

2.4.2 Aprendizado de máquina não supervisionado

O Aprendizado Não Supervisionado é uma técnica de machine learning que se baseia em algoritmos para analisar e agrupar conjuntos de dados não rotulados, tudo isso sem depender de intervenção humana (Lopes et al., 2014). Essa abordagem é especialmente útil para descobrir padrões ocultos ou grupos de dados que compartilham características semelhantes. Suas aplicações são variadas, incluindo análise exploratória de dados, estratégias de vendas cruzadas, segmentação de clientes e até mesmo o reconhecimento de padrões em imagens. Dentro do Aprendizado Não Supervisionado, encontramos três abordagens comuns (IBM, 2023):

2.4.2.1 Armazenamento em Cluster (Clustering)

Essa técnica visa agrupar dados que compartilham semelhanças com base em características específicas. Existem diferentes tipos de armazenamento em cluster, incluindo o exclusivo (onde um ponto de dados pertence a um único *cluster*), o sobreposto (onde um ponto pode pertencer a vários *clusters*), o hierárquico (que organiza clusters em uma hierarquia) e o probabilístico (baseado em probabilidades) (IBM, 2023).

2.4.2.2 Regras de Associação (Association Rules)

Regras de associação são utilizadas para encontrar relacionamentos entre variáveis em um conjunto de dados. Um exemplo prático disso é a identificação de produtos frequentemente adquiridos juntos em uma loja *online*. O algoritmo Apriori é amplamente empregado para essa finalidade (IBM, 2023).

2.4.2.3 Redução de Dimensionalidade (Dimensionality Reduction)

Essa técnica é aplicada quando se deseja reduzir a quantidade de variáveis ou dimensões em um conjunto de dados, especialmente quando o número de características é muito alto. Métodos comuns de redução de dimensionalidade incluem a Análise de Componente Principal (PCA), a Decomposição em Valores Singulares (SVD) e o uso de Codificadores Automáticos (IBM 2023).

Em contraste com o Aprendizado Supervisionado, no qual os dados de treinamento são rotulados e utilizados para ensinar ao modelo as respostas corretas, o Aprendizado Não Supervisionado lida com dados não rotulados. Isso torna essa abordagem adequada para explorar padrões em grandes volumes de dados. No entanto, trabalhar com algoritmos de aprendizado não supervisionado pode ser mais desafiador, uma vez que não há respostas corretas disponíveis para orientar o treinamento.

2.4.3 Aprendizado por reforço

O aprendizado por reforço é uma área de estudo que se concentra em três aspectos essenciais (Sutton, 2015).

Primeiramente, envolve a capacidade de um agente de aprendizado de realizar ações com o objetivo de maximizar um sinal numérico de recompensa, tornando-o um problema intrinsecamente de ciclo fechado, onde as ações influenciam as entradas futuras.

Em segundo lugar, diferencia-se de outros tipos de aprendizado, como o supervisionado, uma vez que o agente não recebe instruções diretas sobre quais ações tomar, mas deve descobrir quais ações levam à recompensa máxima por tentativa e erro.

Terceiro, o aprendizado por reforço lida com um dilema de exploração versus explotação, onde o agente deve equilibrar a escolha de ações que funcionaram no passado com a exploração de novas ações para melhorar seu desempenho futuro.

Em um sistema de aprendizado por reforço, além do agente e do ambiente, identificam-se quatro elementos principais: uma política, um sinal de recompensa, uma função de valor e, opcionalmente, um modelo do ambiente. A política define o comportamento do agente de aprendizado em um determinado momento, mapeando estados percebidos do ambiente para ações a serem tomadas (Sutton, 2015).

O sinal de recompensa estabelece o objetivo do agente, com o objetivo de maximizar a recompensa total ao longo do tempo. Por outro lado, a função de valor especifica o que é bom a longo prazo, representando o montante total de recompensa que um agente pode esperar acumular no futuro, a partir de um determinado estado. Além disso, alguns sistemas de aprendizado por reforço utilizam um modelo do ambiente para prever como o ambiente se comportará em resposta a ações específicas, permitindo o planejamento de ações futuras. O aprendizado por reforço abrange desde métodos de aprendizado baseados em tentativa e erro até métodos de planejamento mais elaborados, e desempenha um papel fundamental na tomada de decisões e no planejamento de ações (Sutton, 2015).

2.2.4 Diferenças entre Inteligencia Artificial, Machine Learning, Deep Learning e Redes Neurais

No nível mais fundamental, o *Machine Learning* é uma categoria de inteligência artificial que capacita os computadores a pensar e aprender por conta própria. Isso envolve a

capacidade de modificar ações com o objetivo de aumentar a precisão, medida em termos de quão frequentemente as ações escolhidas resultam em ações corretas (Alzubi et al., 2018).

As redes neurais, pertencentes ao subconjunto do aprendizado de máquina, buscam replicar o processo de comunicação entre neurônios no cérebro humano. Essa imitação permite a realização de tarefas complexas relacionadas ao aprendizado e ao reconhecimento de padrões (IBM, 2023).

O Deep Learning é uma subárea do Machine Learning que se concentra em redes neurais, compostas por múltiplas camadas, também conhecidas como redes neurais profundas (Lecun et al., 2015). Essas redes possuem a capacidade de aprender representações complexas dos dados e são particularmente adequadas para a resolução de tarefas de alto nível, como reconhecimento de fala, tradução automática, processamento de imagens e jogos. O termo "profundo" refere-se ao grande número de camadas ocultas presentes nessas redes neurais profundas (Lecun et al., 2015).

2.5 Sistemas biométricos

A biométrica em seres humanos engloba medições biológicas ou características físicas que podem ser empregadas para confirmar a identidade de pessoas. Exemplos notáveis desses atributos incluem as impressões digitais, o reconhecimento facial e a análise da íris (Kaspersky, 2023).

Os sistemas biométricos representam abordagens automatizadas que efetuam a verificação ou o reconhecimento da identidade de um indivíduo com base em traços físicos, fisiológicos ou comportamentais (Lopes et al., 2014). Quando combinados a outros métodos de segurança, eles provêem um nível extra de segurança.

A implementação de um sistema biométrico compreende algumas etapas, organizadas em camadas distintas, que desempenham funções no processo de verificação de identidade. A **Figura 2.5** a seguir mostra a arquitetura genérica de um sistema de reconhecimento biométrico.

Figura 2.5: Arquitetura genérica de um sistema de reconhecimento biométrico. Fonte: (Lopes et al., 2014)

2.5.1 Camada de entrada (*Input*)

A camada de entrada representa o ponto de entrada para o sistema biométrico, onde o usuário fornece os dados por meio de um dispositivo de coleta específico. Esses dados podem abranger uma variedade de informações biométricas, como impressões digitais, imagens faciais, padrões de íris, amostras de voz e até mesmo características de digitação (Lopes et al., 2014).

2.5.2 Camada de extração de características (*Feature extraction*)

Na camada de extração de características, que se segue à coleta dos dados biométricos, ocorre um processo de destaque das características mais pertinentes dentro do conjunto de informações capturadas. No contexto biométrico, essas características podem incluir a velocidade de digitação, a pressão aplicada nas teclas, o intervalo entre as pressões, bem como detalhes específicos, como a topologia das cristas e vales nas impressões digitais. Essa etapa é essencial, pois visa a redução da dimensionalidade dos dados, tornando-os mais concisos e representativos (Lopes, 2014).

2.5.3 Camada de classificação (*Classification*)

Após a extração das características biométricas, a camada de classificação assume o papel de tomar decisões com base nas informações previamente identificadas. Para isso, essa camada utiliza algoritmos de aprendizado de máquina ou técnicas estatísticas avançadas. Ela compara as características biométricas do usuário em questão com os dados armazenados no banco de dados biométrico. Esse processo visa determinar se as características extraídas correspondem a um usuário autorizado ou se há diferenças substanciais que possam indicar a presença de um usuário não autorizado (Lopes et a., 2014).

2.5.4 Camada de decisão (*Decision*)

A camada de decisão representa a etapa final do processo biométrico, onde uma decisão final é tomada com base nas saídas da camada de classificação. Em um cenário típico de um sistema de verificação biométrica, se a camada de classificação determina que as características biométricas do usuário correspondem às de um usuário autorizado, a camada de decisão concede o acesso. Por outro lado, se as características não forem compatíveis ou se houver alguma discrepância significativa, a camada de decisão negará o acesso, garantindo assim a segurança e a integridade do sistema biométrico (Lopes et al., 2014).

2.6 Autenticação contínua com base em padrão biométrico de digitação

A autenticação contínua com padrões de digitação é uma abordagem avançada para garantir a segurança em dispositivos móveis, especialmente em smartphones. Tradicionalmente, os métodos de autenticação utilizavam senhas, PINs ou autenticação biométrica para verificar a identidade do usuário. No entanto, essas medidas de autenticação estáticas podem ser contornadas por ataques sofisticados ou desbloqueadas no caso de um possível furto do aparelho. Nesse contexto, a autenticação contínua com base em métricas comportamentais se destaca como uma solução inovadora para fortalecer a segurança (Lu, 2020).

O monitoramento em tempo real se baseia na ideia de que o comportamento do usuário é único e pode ser utilizado como um fator adicional de autenticação. Em vez de depender apenas de informações estáticas, como senhas ou impressões digitais, a autenticação contínua monitora constantemente o comportamento do usuário durante o uso do dispositivo. Isso inclui padrões de digitação, gestos, movimentos do dispositivo e outras métricas comportamentais (Vemou, 2023).

São utilizados algoritmos de aprendizado de máquina e inteligência artificial para analisar e identificar padrões comportamentais específicos de cada usuário. Esses algoritmos são treinados para reconhecer características individuais e identificar possíveis anomalias ou comportamentos suspeitos (Vemou, 2023).

Ao longo do tempo, o sistema de autenticação contínua aprende e se adapta ao comportamento do usuário, criando um perfil comportamental único. Isso permite que o sistema diferencie entre o comportamento legítimo do usuário e atividades maliciosas realizadas por terceiros não autorizados (Lu, 2020).

A autenticação contínua com *behavioral metrics* representa um avanço promissor na segurança de dispositivos móveis, oferecendo uma abordagem mais robusta e eficaz para autenticar os usuários no que diz respeito à proteção contra o acesso físico não autorizado. Ao analisar o padrão de digitação individual do usuário, essa abordagem proporciona uma camada adicional de segurança, tornando mais difícil para os invasores acessarem informações confidenciais ou realizarem atividades maliciosas.

Como a segurança cibernética é um campo em constante evolução, espera-se que a autenticação contínua com *behavioral metrics* continue a se desenvolver e aprimorar, tornando-se uma parte cada vez mais importante da estratégia de segurança em dispositivos móveis. Ao adotar medidas inovadoras como essa, podemos fortalecer a segurança dos dispositivos móveis e proteger as informações sensíveis dos usuários.

2.7 Support Vector Machines

O *Support Vector Machine* (SVM), em portugues "Máquina de Vetores de Suporte", é um algoritmo de aprendizado de máquina amplamente reconhecido pela sua habilidade notável de encontrar uma linha ou hiperplano de separação altamente eficaz entre duas classes distintas em conjuntos de dados. Esse método se destaca pela maneira como considera atentamente os dois pontos mais próximos de cada classe em relação à outra, resultando na identificação de uma fronteira de decisão otimizada que maximiza a distância entre esses dois grupos, como ilustrado na **Figura 2.7** representada pela linha vermelha (Coutinho, 2019).

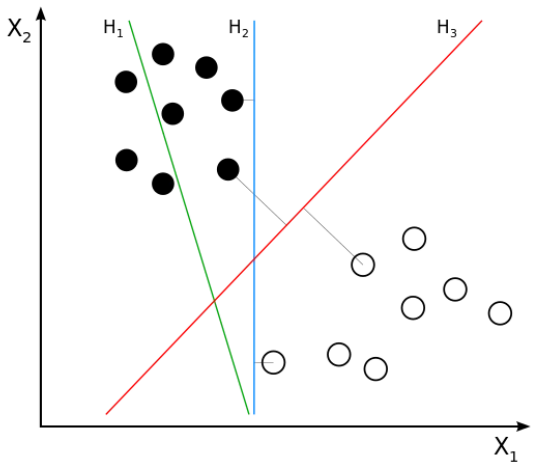

Figura 2.7: Esquema de aprendizado supervisionado. Fonte: (Coutinho, 2019)

Este algoritmo é amplamente empregado em diversas aplicações de classificação, tais como reconhecimento de padrões, detecção de spam, diagnóstico médico e muito mais, devido à sua notável capacidade de lidar com diferentes tipos de dados e problemas de classificação. O SVM destaca-se especialmente em cenários desafiadores, onde os dados podem ser altamente dimensionais e complexos, e a identificação de uma fronteira de decisão eficaz é crucial para o sucesso da tarefa de classificação. Portanto, o SVM é uma ferramenta essencial no campo do aprendizado de máquina.

Existem diferentes tipos de SVM, sendo os principais (Saini, 2021):

- SVM Linear: este tipo de SVM utiliza uma reta ou hiperplano linear para separar as classes e é adequado para conjuntos de dados que podem ser separados por uma linha reta. Quando os dados são linearmente separáveis, o SVM Linear encontra o hiperplano que maximiza a margem entre as duas classes. A margem representa a distância entre o hiperplano e o ponto mais próximo de cada classe. Os pontos de dados mais próximos do hiperplano, chamados de vetores de suporte, desempenham um papel fundamental na determinação da fronteira de decisão. O hiperplano é posicionado de forma a maximizar a distância entre os vetores de suporte e, consequentemente, a margem;
- SVM Não-Linear: em contraste, o SVM Não Linear utiliza curvas ou hiperplanos não lineares para separar as classes em espaços de alta dimensão. Ele é empregado quando os dados não podem ser eficazmente separados por uma reta. Quando os dados não são linearmente separáveis, o SVM Não Linear recorre a um truque de *kernel*, que mapeia os dados para um espaço dimensional superior onde podem ser separados linearmente. Os tipos de *kernel* mais comuns incluem o *kernel* polinomial, o *kernel* radial (RBF) e o *kernel* sigmoide, que aplicam transformações não lineares para encontrar uma fronteira de decisão em um espaço mais complexo. Com o uso desses *kernels*, o SVM pode separar dados que seriam intratáveis em sua forma original.

2.7.1 One-class Support Vector Machines: visão geral

A *One-Class Support Vector Machine* (OCSVM), também conhecida como Máquina de Vetores de Suporte para uma Única Classe, é um modelo amplamente empregado no campo do aprendizado não supervisionado, com destaque na detecção de anomalias (Alam, 2020). Ao contrário de uma *Support Vector Machine* (SVM) convencional, que normalmente

necessita de rótulos de classe alvo durante o treinamento, a versão de uma única classe opera de forma independente desses rótulos.

O algoritmo OCSVM é projetado para construir um hiperplano ótimo que divide amostras de dados semelhantes às utilizadas no treinamento daquelas que não são, essencialmente separando dados em duas classes distintas. Este hiperplano age como uma função de fronteira de decisão, atribuindo +1 às amostras de dados de um lado do hiperplano e -1 ao lado oposto. Para minimizar erros, como falsos positivos e falsos negativos, o OCSVM procura determinar essa fronteira de decisão resolvendo um problema de otimização que busca maximizar as margens de separação entre as classes (Wang, 2022).

Essa fronteira delimita a região que engloba a maioria dos dados considerados normais, e a OCSVM identifica qualquer ponto que se afaste significativamente desse limite, considerando-o uma possível anomalia (Medium, 2022). Em termos simplificados, a OCSVM aprende o conceito do que é considerado "normal" e distingue entre os elementos que se encaixam nesse padrão e aqueles que não o fazem.

A One-Class Support Vector Machine (OCSVM) possui uma ampla gama de aplicações em diversas áreas, destacando-se em cenários que envolvem classificação de documentos, diagnóstico médico, detecção de fraudes, segurança cibernética (detecção de intrusos), entre muitos outros (Alam, 2020). Seu poder de detecção de anomalias a torna uma ferramenta versátil e essencial em situações onde a identificação de padrões anômalos é crucial para tomar decisões informadas e proteger sistemas e processos contra ameaças e falhas. Portanto, a OCSVM desempenha um papel significativo em várias aplicações críticas, contribuindo para a segurança, a qualidade e a eficiência em diversos setores (Medium, 2022)(Alam, 2020).

2.7.2 *One-Class Support Vector Machines*: Algoritmos

No treinamento do OCSVM, procuramos estabelecer um hiperplano ótimo no espaço de características, seguindo o princípio de maximizar o valor do intervalo para alcançar a separação das amostras de treinamento e sua origem (Saini, 2021). Essa abordagem é a base do que é conhecido como Máquina de Vetores de Suporte de Uma Única Classe (OCSVM), um modelo criado para resolver problemas de classificação em que há apenas uma classe de interesse. O OCSVM constrói esse hiperplano de forma a separar os pontos de dados da classe de interesse com a maior margem possível em relação à origem, considerando todos os valores atípicos como pertencentes ao hiperplano (Hejazi, 2013).

O OCSVM resolve um problema de otimização para encontrar o hiperplano ótimo no espaço de características, mas esse problema de otimização é solucionável em tempo polinomial. A abordagem Lagrangiana, que será vista nas sessões que se seguem, utilizada no OCSVM, é uma técnica comum em otimização e não transforma o problema em NP-difícil.

Essa técnica é especialmente útil em cenários de detecção de anomalias, onde é essencial identificar dados que se desviam significativamente da norma. A **Figura 2.7.2** a seguir ilustra o esquema do classificador.

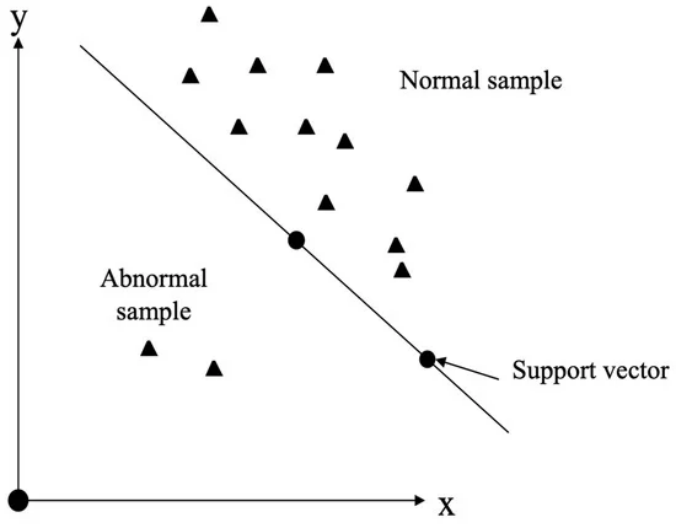

Figura 2.7.2: Diagrama esquemático de um classificador vetor de suporte. Fonte: (Saini, 2021)

Nos parágrafos a seguir, é apresentada uma exposição matemática breve sobre o funcionamento do OCSVM:

2.7.2.1 Hiperplano de Decisão e Seu Objetivo

Como abordado no tópico anterior, o OCSVM tem como objetivo encontrar um hiperplano ótimo no espaço de características. Matematicamente, esse hiperplano é representado como $f(x)=\omega\phi(x)-\varrho=0$, onde:

x representa um vetor de características.

ω é o vetor normal ao hiperplano.

ϕ é uma função que mapeia os dados para um espaço de dimensão superior.

ϱ é a distância entre o hiperplano e a origem.

2.7.2.2 Problema de Otimização

O OCSVM formula um problema de otimização com o objetivo de encontrar os valores ideais de ω, ξ (variáveis de folga) e ϱ que minimizam a expressão:

$$\omega\|\omega\|^2 + 1/N\text{v}\sum Ni=1\xi i-\varrho.$$

Nesse contexto:

- N é o tamanho do conjunto de dados usado no treinamento.
- v é um parâmetro de regularização.
- i são as variáveis de folga associadas a cada amostra.
- O objetivo é encontrar um hiperplano que maximize a margem entre as classes.

2.7.2.3 Multiplicadores de Lagrange

Para resolver esse problema de otimização, introduzimos multiplicadores de Lagrange $\alpha\text{i} \geq 0$ e $\beta\text{i} \geq 0$, relacionados às restrições das variáveis de folga. Esses multiplicadores são coeficientes associados às condições de igualdade e desigualdade no problema de otimização e são essenciais para a formulação do problema dual. A função Lagrangiana, representada por $\mathcal{L}$, é então construída incorporando os multiplicadores de Lagrange e as condições do problema original. A otimização da função Lagrangiana em relação às variáveis primais e aos multiplicadores de Lagrange resulta nos valores ótimos de ω, ξ (variáveis de folga) e ϱ. Essa abordagem de otimização Lagrangiana é fundamental para transformar o problema original em um problema dual mais gerenciável, simplificando assim a busca pela solução ótima.

2.7.2.4 Função de Decisão

Uma vez que o hiperplano ótimo é determinado, ele funciona como uma função de fronteira de decisão. Ao avaliar uma nova amostra x usando $f(x)$, o resultado positivo ou negativo determina se a amostra é considerada "normal" ou uma anomalia.

Em resumo, o OCSVM é uma ferramenta poderosa para identificar anomalias em conjuntos de dados não rotulados. Ele aprende o conceito do que é considerado "normal" e classifica qualquer elemento que se desvie significativamente desse padrão como uma anomalia. Essa abordagem é particularmente útil em cenários como detecção de fraudes, segurança cibernética e diagnóstico médico, onde a identificação de padrões anômalos é de extrema importância.

2.7.2.5 Exemplo de Aplicação

A seguir, encontra-se uma aplicação genérica do OCSVM (Máquina de Vetores de Suporte de Uma Única Classe) em problemas de detecção de anomalias, tal aplicação foi baseada em exemplos do site oficial da Scikit Learn, disponível em https://scikit-learn.org/, onde o objetivo é identificar elementos incomuns ou fora do padrão em um conjunto de dados, independentemente do domínio do problema.

Considere um conjunto de dados com várias características e deseja-se identificar amostras que sejam consideradas "anômalas" em relação à maioria das outras amostras. Essas amostras podem representar eventos raros, comportamentos suspeitos, falhas de equipamentos ou qualquer coisa que seja significativamente diferente da norma.

Nesta seção, estamos importando três bibliotecas fundamentais para o nosso código:

- “pandas” (abreviada como “pd”): Essa biblioteca é usada para manipulação de dados e permite trabalhar com estruturas de dados em formato de DataFrame (Pandas, 2023).
- “OneClassSVM” do “sklearn.svm”: Isso importa a classe One-Class SVM da biblioteca Scikit-Learn, que usaremos para criar nosso modelo de detecção de anomalias (Scikit Learn, 2023).
- “joblib”: Essa biblioteca é usada para salvar nosso modelo treinado em um arquivo (Joblib, 2023)

```
import pandas as pd
from sklearn.svm import OneClassSVM
import joblib
```

Nesta etapa, carregamos os dados do arquivo CSV fornecido, tal arquivo é representado por “arquivo_de_dados.csv”. Os dados são carregados em um DataFrame do pandas para facilitar a manipulação.

```
dados = pd.read_csv('arquivo_de_dados.csv')
```

A **Figura 2.7.2.5** a seguir mostra um exemplo de arquivo CSV adequado para o exemplo.

	Padrão	Padrão	Padrão
1	Feature1	Feature2	Feature3
2	1.0	4.0	7.0
3	2.0	5.0	8.0
4	1.5	4.5	7.5
5	3.0	6.0	9.0
6	2.5	5.5	8.5

Figura 2.7.2.5: Tela de definição de senha. Fonte: Autora

Aqui, estamos selecionando as colunas relevantes do DataFrame para serem usadas como recursos (features) para o treinamento do modelo. “Feature1”, “Feature2” e “Feature3” são os nomes das características que serão usadas no modelo.

```
X = dados[['Feature1', 'Feature2', 'Feature3']]
```

Esta linha de código cria um modelo One-Class SVM com a classe “OneClassSVM” do Scikit-Learn. O parâmetro “nu” é um hiperparâmetro que controla a sensibilidade do modelo. Valores menores de “nu” tornam o modelo mais sensível a anomalias, enquanto valores maiores tornam o modelo menos sensível.

```
modelo_ocsvm = OneClassSVM(nu=0.01)
```

Nesta etapa, estamos treinando o modelo One-Class SVM com os dados de treinamento “X”. Isso permite que o modelo aprenda a distinguir entre os dados normais e as possíveis anomalias com base nos recursos fornecidos.

```
modelo_ocsvm.fit(X)
```

Aqui, estamos salvando o modelo treinado em um arquivo usando a biblioteca “joblib”. Definimos o nome do arquivo como “modelo_ocsvm.pkl, mas você pode escolher qualquer nome de arquivo que desejar. O modelo treinado é salvo no arquivo especificado, e uma mensagem é impressa para confirmar que o modelo foi salvo com sucesso.

```
modelo_ocsvm_filename = 'modelo_ocsvm.pkl'
joblib.dump(modelo_ocsvm, modelo_ocsvm_filename)
```

O OCSVM é uma ferramenta versátil e amplamente aplicável em uma variedade de cenários, pois não requer conhecimento prévio sobre o que constitui uma anomalia. Ele aprende a partir dos dados e é capaz de identificar elementos incomuns em qualquer domínio, contribuindo para a detecção de problemas, segurança e qualidade em diversas áreas.

3. DESENVOLVIMENTO

Este trabalho foi desenvolvido seguindo a definição de etapas específicas que ajudaram a alcançar os objetivos desejados:

1. Um estudo detalhado sobre a arquitetura, recursos e funcionalidades dos dispositivos Android foi conduzido para compreender o funcionamento e identificar possíveis vulnerabilidades. Além disso, um estudo detalhado sobre o algoritmo One-Class Support Vector Machine também foi realizado.
2. Desenvolvimento de um aplicativo Android de troca de mensagens na IDE Android Studio, com implementação de funcionalidades para simular o envio e exibição de mensagens.
3. Criação de um teclado personalizado para o sistema, permitindo a captura de informações de eventos durante a digitação.
4. Implementação de um serviço no aplicativo para coletar dados relevantes enquanto o usuário digita, incluindo informações como tempo de pressão, pressão de clique e área ocupada no clique.
5. Treinamento de um modelo OCSVM para detecção de anomalias.
6. Integração do sistema de autenticação contínua ao aplicativo, utilizando os padrões de digitação aprendidos pelo modelo.

A Etapa 1 foi contemplada no Capítulo 2, enquanto o texto a seguir contempla de uma forma conjunta o desenvolvimento das demais etapas.

3.1 Visão geral do sistema de autenticação contínua

O sistema desenvolvido consiste em um aplicativo de mensagens que emprega um método de autenticação contínua com base nos padrões de digitação do usuário. Especificamente, são coletados e analisados dados como tempo de pressionamento, pressão aplicada e área de toque na tela durante a interação com o teclado personalizado. A partir desses dados, é possível identificar e verificar a legitimidade do usuário em tempo real.

Além disso, o aplicativo possui funcionalidades de conectividade com um servidor remoto, permitindo o *download* diário de um modelo aprimorado de aprendizado de máquina. Esse modelo é utilizado para aprimorar o processo de autenticação e detecção de comportamentos suspeitos. No caso de uma identificação de um padrão de digitação anômalo, o aplicativo é capaz de bloquear a tela e solicitar uma senha previamente definida pelo usuário, garantindo assim a segurança das interações e dos dados do usuário.

A **Figura 3.1** apresenta a janela de definição de uma senha, que é exibida durante a primeira execução do aplicativo. Esse recurso permite ao usuário configurar uma senha pessoal para garantir a autenticidade e a segurança do acesso ao aplicativo.

Figura 3.1.1: Tela de definição de senha. Fonte: Autora

Além disso, a **Figura 3.1.2** ilustra a tela de bloqueio, que é acionada quando um comportamento suspeito é identificado pelo modelo de treinamento, no caso, o One-Class Support Vector Machine (OCSVM). Esse método de machine learning, discutido anteriormente neste documento, é fundamental para a detecção e prevenção do uso do aplicativo por um usuário ilegítimo no contexto deste projeto..

Figura 3.1.2: Tela de bloqueio. Fonte: Autora

A aplicação do método em um aplicativo de mensagens é uma abordagem desenvolvida especificamente para este projeto, porém, métodos de autenticação contínua em aplicativos Android já foi realizada anteriormente, este projeto se baseou especialmente no fluxo utilizado no trabalho intitulado "*Keystroke Recognition Using Android Devices"* (Lopes et al.,, 2014).

Nas sessões que se seguem, será realizada uma explanação mais detalhada sobre a arquitetura e a implementação específica do aplicativo, abordando seus componentes e funcionalidades-chave.

3.2 Coleta de dados

Para viabilizar a coleta de dados durante a utilização do aplicativo, foi desenvolvido um teclado personalizado que replica a aparência e o funcionamento do teclado padrão do Android. Esse teclado customizado permite o monitoramento preciso dos eventos de toque na tela e a captura dos dados essenciais para análise e processamento.

A Figura 3.2 ilustra o *layout* desse teclado na interface do aplicativo.

Figura 3.2. Visão geral do aplicativo.

3.2.1 Captura de eventos

A coleta de dados ocorre quando o usuário interage com o teclado personalizado. Mais especificamente, quando um botão é pressionado e solto, os seguintes dados são coletados:

- Tempo de pressão (*Dwell Time*): É o tempo em milissegundos que o usuário pressiona um botão. Isso é calculado subtraindo o tempo de início da pressão do tempo de término.
- Diferença de pressão (*Pressure Difference*): É a diferença entre a pressão no início e no final da pressão do botão. Essa métrica pode ajudar a distinguir a pressão do usuário de outros fatores.
- Área de toque (*Touch Area*): Refere-se à área da superfície de toque do botão durante a pressão. Pode ser uma medida do tamanho da área de toque.

A captura de eventos no teclado personalizado é realizada através de ouvintes de toque anexados a cada botão. Esses ouvintes respondem a eventos de pressionamento e soltura de teclas, registrando as interações do usuário com o teclado.

O código inclui um método chamado “handleKeyPress”, que é invocado durante os eventos de toque dos botões. Ele monitora ações como pressionar e soltar e registra os dados relevantes durante essas interações.

A seguir, temos trecho de código que exemplifica como o método “handleKeyPress” é utilizado para capturar os eventos de toque:

```
private void handleKeyPress(char keyPressed, MotionEvent event) {
   if (onKeyPressListener != null) {
       switch (event.getAction()) {
           case MotionEvent.ACTION_DOWN:
               pressStartTime = System.currentTimeMillis();
               lastPressure = event.getPressure();
               lastTouchArea = event.getSize();
               break;
           case MotionEvent.ACTION_UP:
               if (pressStartTime != 0) {
                   // Captura de dados relevantes durante a interação
                   long dwellTime = System.currentTimeMillis() -
pressStartTime;
                   float pressureDifference = Math.abs(lastPressure -
event.getPressure());
                   float touchArea = event.getSize();
                   KeystrokeData keystrokeData = new KeystrokeData(dwellTime,
pressureDifference, touchArea);
                   // Adiciona os dados capturados à lista de dados
                   keystrokeDataList.add(keystrokeData);
                   temporaryDataList.add(keystrokeData);
                   charactersSinceLastSend++;
                   pressStartTime = 0;
                   lastPressure = 0;
                   lastTouchArea = 0;
                   // Chama o ouvinte de pressionamento com os dados
capturados
                   onKeyPressListener.onKeyPress(keyPressed, dwellTime,
pressureDifference, touchArea);
               }
               break;
       }
```

Neste trecho, o método “handleKeyPress” monitora o tempo de pressão, a diferença de pressão e a área de toque durante as ações de pressionar e soltar. Ele instancia um objeto “KeystrokeData” com esses valores e o adiciona à lista “keystrokeDataList”. Além disso, chama o ouvinte “onKeyPressListener” para notificar a ocorrência do evento com os dados coletados.

Essa abordagem de captura de eventos garante que cada interação do usuário com o teclado personalizado seja registrada e os dados relevantes sejam armazenados para análise e processamento posterior.

3.2.2 Armazenamento dos Dados temporários

Os dados coletados são armazenados em objetos da classe “KeystrokeData”, que contém esses três valores. Esses objetos são adicionados à lista “keystrokeDataList”, que mantêm o histórico de dados coletados.

3.2.3 Exportação para CSV

A exportação dos dados para um arquivo CSV é realizada quando certas condições são atendidas, quando a quantidade de dados atinge um limite predefinido (“exportThreshold”), que na versão atual do projeto foi definido como dez caracteres. O código utiliza o método “exportKeystrokeDataToCSV” para realizar essa exportação.

Aqui está uma parte do código que demonstra a exportação de dados para um arquivo CSV:

```
public void exportKeystrokeDataToCSV(List<KeystrokeData> keystrokeDataList,
String csvFilename) {
   try {
       FileWriter writer = new FileWriter(csvFilename);
       for (KeystrokeData data : keystrokeDataList) {
           // Escreve os dados no arquivo CSV no formato apropriado
           writer.write(data.getDwellTime() + "," +
data.getPressureDifference() + "," + data.getTouchArea() + "\n");
       }
       writer.close();
       Log.d("CustomKeyboard", "Dados exportados para CSV: " + csvFilename);
```

```
    } catch (IOException e) {
        Log.e("CustomKeyboard", "Erro ao exportar para CSV: " +
e.getMessage());
    }
}
```

Nesse trecho, os dados de tempo de pressão, diferença de pressão e área de toque de cada objeto “KeystrokeData” são escritos no arquivo CSV no formato apropriado, a **Figura 3.2.1** mostra um trecho deste arquivo. Cada linha no arquivo CSV representa uma interação do usuário com o teclado personalizado, com os valores separados por vírgula.

1	111	0.05098039	0.023529414
2	49	0.019607857	0.015686275
3	24	0.08627452	0.019607844
4	45	0.047058836	0.019607844

Figura 3.2.1:Trecho do arquivo CSV armazenado no servidor. A primeira coluna é o tempo de pressionamento da tecla, a segunda coluna é a área de pressão utilizada e a última coluna é a pressão aplicada no clique. Fonte: Autora.

Essa exportação para CSV permite que os dados sejam facilmente armazenados e posteriormente analisados pelo modelo, seja para fins de treinamento de modelos ou para verificação do usuário.

3.3 Servidor

O servidor desempenha um papel crucial na comunicação entre o aplicativo móvel de mensagens e o servidor web construído com o *framework* Flask em Python. Embora seja importante reconhecer que há oportunidades de melhoria na implementação da comunicação, é fundamental destacar que essas partes do código desempenham funções essenciais no sistema. Elas permitem a realização do *download* do modelo uma vez por dia, o envio de dados para a atualização do modelo e até mesmo a capacidade de treinar o modelo com base nos dados recebidos.

3.3.1 Recebimento dos dados

O servidor Flask define uma rota com a URL /receber_dados para receber dados enviados pelo cliente. Esse recebimento ocorre por meio de uma solicitação HTTP do tipo POST. Os dados são transmitidos ao servidor no formato CSV e, em seguida, anexados a um arquivo CSV local para processamento posterior. É realizada uma pequena verificação no recebimento dos dados. O trecho abaixo mostra como essa etapa foi realizada.

```
@app.route('/receber_dados', methods=['POST'])
def receber_dados():
    try:
        # Obtem os dados da solicitação POST
        dados_csv = request.form.get('dados')

        # Verifica se existem dados
        if dados_csv:
            # Abre o arquivo CSV em modo de anexação e escreve os dados
recebidos
            with open('dados_csv/meus_dados.csv', 'a') as arquivo:
                arquivo.write(dados_csv)
    except Exception as e:
        print("Erro ao receber dados:", str(e))
        return "Erro ao receber dados.", 500
```

3.4 Armazenamento dos dados

Os dados no sistema estão sendo armazenados tanto no cliente Android quanto no servidor Flask, e a maneira como são tratados em cada lado é essencial para o funcionamento do aplicativo. A seguir, será explicado em detalhes como ocorre o tratamento e armazenamento dos dados coletados.

3.4.1 Armazenamento de Dados no Cliente

No cliente Android, os dados estão sendo armazenados localmente antes de serem enviados para o servidor. Os dados são coletados e armazenados em uma lista denominada “keystrokeDataList”. Essa lista mantém informações relacionadas às ações de digitação, como o tempo de pressionamento das teclas, a diferença de pressão e a área de pressão.

Quando a lista “keystrokeDataList” atinge o limite de dez caracteres, os dados são exportados para um arquivo CSV local. Essa abordagem de armazenamento em lotes, visa reduzir a frequência de comunicação com o servidor, economizando largura de banda e

melhorando o desempenho do aplicativo. No entanto, é importante notar que a segurança dos dados durante o armazenamento local e o transporte para o servidor deve ser uma consideração crítica, que deve ser tratada em versões futuras. O limite de caracteres foi mantido em dez. Foram feitos testes de cinco a dez caracteres. O valor que mostrou maior acurácia, mantendo-se um valor pequeno, foi dez.

3.4.2 Armazenamento de Dados no Servidor Flask (Python)

No lado do servidor Flask, os dados enviados pelo cliente Android são recebidos em formato CSV e anexados a um arquivo CSV local, denominado "meus_dados.csv". Isso é feito na rota “/receber_dados”. O código utiliza o modo de anexação para adicionar os novos dados ao final do arquivo existente. Dessa forma, os dados são acumulados ao longo do tempo, criando um registro histórico das ações de digitação do usuário.

É importante observar que essa abordagem de armazenamento no servidor é simples e funcional, mas pode não ser a mais eficiente em cenários com grande volume de dados, pois a leitura e gravação em arquivos CSV podem se tornar lentas à medida que o arquivo cresce. Em casos de produção real, considerações de escalabilidade e desempenho podem levar à exploração de soluções de armazenamento de dados mais robustas, como bancos de dados NoSQL.

Além disso, a segurança dos dados armazenados no servidor deve ser uma prioridade. Medidas de autenticação, autorização e criptografia devem ser implementadas para garantir a integridade e a confidencialidade dos dados do usuário.

Em resumo, os dados neste sistema são armazenados localmente no cliente Android em formato de lote antes de serem enviados para o servidor Flask, onde são anexados a um arquivo CSV local. Embora funcional, essa abordagem pode ser otimizada para cenários de uso intensivo de dados e requer atenção especial à segurança dos dados armazenados no servidor.

3.5 Treinamento do modelo

O treinamento de modelos desempenha um papel crucial no sistema em questão, que envolve a comunicação entre um aplicativo móvel Android e um servidor Flask em Python. Esta seção do trabalho se concentra em analisar detalhadamente o processo de treinamento do modelo, explicando as etapas envolvidas e a teoria subjacente.

O treinamento do modelo começa com a carga de dados a partir de um arquivo CSV. Os dados, originários do usuário legítimo, são lidos a partir do arquivo "meus_dados.csv", localizado na pasta "dados_csv". A leitura do arquivo é feita utilizando a biblioteca Pandas, uma ferramenta amplamente empregada para manipulação de dados em Python. O arquivo contém três colunas: “Dwell Time” (Tempo de Pressão), “Pressure Difference” (Diferença de Pressão) e “Area” (Área).

```
dados_usuario_original = pd.read_csv('dados_csv/meus_dados.csv', usecols=[0,
1, 2], names=['Dwell Time', 'Pressure Difference', 'Area'])
```

Após o carregamento dos dados, inicia-se a etapa de pré-processamento, que visa preparar os dados para o treinamento do modelo. Um aspecto crucial nesse processo é a substituição de valores iguais a zero na coluna “Pressure Difference” por NaN (*Not-a-Number*) para lidar com dados inadequados ou ausentes que possam impactar negativamente o desempenho do modelo. Em seguida, os valores NaN são preenchidos com a média dos valores de pressão da coluna. A exclusão de valores de “Pressure Difference” zerados busca melhorar a precisão do modelo, eliminando possíveis interferências causadas por dados irrelevantes.

```
# Substitui os valores iguais a zero por NaN
dados_usuario_original['Pressure Difference'] =
dados_usuario_original['Pressure Difference'].replace(0, float('nan'))

# Preenche os valores NaN com a média dos valores de pressão
media_pressao = dados_usuario_original['Pressure Difference'].mean()
dados_usuario_original['Pressure Difference'].fillna(media_pressao,
inplace=True)
```

Com os dados devidamente pré-processados, eles são divididos em recursos (X) que serão utilizados para o treinamento do modelo. Neste contexto, os recursos consistem nas colunas “Dwell Time”, “Pressure Difference” e “Area” dos dados originais do usuário.

```
# Divide os dados em recursos (X)
X = dados_usuario_original[['Dwell Time', 'Pressure Difference', 'Area']]
```

O modelo escolhido para esse sistema é o *One-Class SVM*, abordado na seção 2.7.1, uma técnica de aprendizado de máquina projetada para identificar anomalias nos dados. A implementação do modelo é realizada com a classe “OneClassSVM” da biblioteca Scikit-Learn, com um parâmetro "nu" definido como 0.01. Esse parâmetro "nu" controla a sensibilidade do modelo à detecção de anomalias e pode ser ajustado conforme necessário para atender aos requisitos específicos do sistema.

```
# Cria o modelo One-Class SVM
modelo_ocsvm = OneClassSVM(nu=0.01)
```

Após a criação, o modelo é treinado com os dados do usuário original (X). Durante essa etapa, o modelo aprende os padrões normais nos dados, preparando-se para identificar anomalias posteriormente.

```
# Treina o modelo apenas com os dados do usuário original
modelo_ocsvm.fit(X)
```

Finalmente, o modelo treinado é persistido em um arquivo denominado "modelo_ocsvm.pkl". Isso permite que o modelo seja baixado pelo cliente Android. O arquivo do modelo é armazenado no mesmo diretório onde o servidor Flask está em execução.

```
# Salva o modelo treinado em um arquivo
modelo_ocsvm_filename = 'modelo_ocsvm.pkl'
joblib.dump(modelo_ocsvm, modelo_ocsvm_filename)
print(f"Modelo One-Class SVM treinado e salvo em {modelo_ocsvm_filename}")
```

Em resumo, a etapa de treinamento do código é responsável por carregar, pré-processar e treinar um modelo *One-Class* SVM com base nos dados do usuário original. O modelo treinado é salvo em um arquivo para uso posterior.

3.6 Aplicação do modelo de detecção de anomalias

O modelo de verificação, que é baixado pelo aplicativo uma vez a cada período de 24 horas, deveria ser aplicado aos dados dentro do próprio dispositivo. No entanto, devido a uma

série de inconsistências e incompatibilidades entre as bibliotecas capazes de carregar modelos treinados em Python, não foi possível concluir a verificação dos dados no dispositivo. Embora os módulos de *download* do modelo tenham sido devidamente implementados, a verificação com modelo ocorre no servidor. A correção para tal problema deve ser aplicada em versões futuras, a fim de evitar o uso excessivo de largura de banda e reduzir a dependência de conexão com a internet para que o método seja aplicado.

No caso deste trabalho, os dados para o primeiro treinamento do modelo foram coletados ao longo de um dia, período em que 1270 caracteres foram acrescentados no arquivo utilizado para treinamento. Durante esse período, o servidor recebeu constantemente lotes de caracteres, cada lote contendo 10 caracteres. Após completar 24 horas de recebimento de caracteres, o modelo foi treinado pelo servidor.

Com o modelo pronto para verificar os dados, o servidor passou a tratar os lotes recebidos de duas formas. Primeiro, aplicando o modelo nos dez caracteres recebidos e, em seguida, devolvendo uma resposta. Caso os dados sejam considerados de usuário legítimo, ou seja "Nao intruso", tais dados são acrescentados ao arquivo total, que seria parte do treinamento do modelo no próximo ciclo de treinamento, 24 horas após o último treinamento. Caso sejam identificados como "Intruso", os dados deste usuário são excluídos, a fim de evitar dados incongruentes no arquivo de treinamento.

O trecho de código a seguir mostra como foi feita a verificação com a aplicação do modelo no conjunto de dados.

```
import joblib
import pandas as pd

# Carrega o modelo One-Class SVM previamente treinado
modelo_ocsvm = joblib.load('modelo_ocsvm.pkl')

# Carrega o novo conjunto de dados
novo_conjunto_de_dados = pd.read_csv('dados_csv/dados_recebidos.csv',
usecols=[1, 2, 3], names=['Dwell Time', 'Pressure Difference', 'Area'])

# Substitui os valores iguais a zero por NaN e preencha com a média dos
valores de pressão
novo_conjunto_de_dados['Pressure Difference'] =
novo_conjunto_de_dados['Pressure Difference'].replace(0, float('nan'))
```

```
media_pressao = novo_conjunto_de_dados['Pressure Difference'].mean()
novo_conjunto_de_dados['Pressure Difference'].fillna(media_pressao,
inplace=True)

# Faz previsões usando o modelo One-Class SVM
previsoes_novo = modelo_ocsvm.predict(novo_conjunto_de_dados)

# Adiciona as previsões ao DataFrame do novo conjunto de dados
novo_conjunto_de_dados['Anomaly'] = previsoes_novo  # -1 para anomalias, 1
para dados normais

# Determina se cada ponto de dados é "intruso" ou não
novo_conjunto_de_dados['Resultado'] = ['Intruso' if anomalia == -1 else
'Normal' for anomalia in previsoes_novo]

# Verifica a maioria das previsões
contagem_intruso =
novo_conjunto_de_dados['Resultado'].value_counts().get('Intruso', 0)
contagem_normal =
novo_conjunto_de_dados['Resultado'].value_counts().get('Normal', 0)

# Define o resultado final com base na maioria
if contagem_intruso > contagem_normal:
   resultado_final = 'Intruso'
else:
   resultado_final = 'Nao Intruso'

# Exibe o resultado final
print(f"Resultado Final: {resultado_final}")
```

O código acima carrega um modelo OCSVM previamente treinado a partir do arquivo “modelo_ocsvm.pkl”. Em seguida, carrega o conjunto de dados recebidos que estão em formato CSV e realiza o pré-processamento dos dados. O pré-processamento envolve a substituição de valores iguais a zero na coluna “Pressure Difference” por NaN e, em seguida, preenchendo esses NaNs com a média dos valores de pressão.

Posteriormente, o modelo OCSVM é usado para fazer previsões com base no conjunto de dados lidos. As previsões são adicionadas ao *DataFrame* do novo conjunto de dados. Cada

ponto de dados é então classificado como "Intruso" ou "Normal" com base nas previsões feitas pelo modelo.

O código verifica a maioria das previsões e, com base nessa contagem, determina o resultado final. Se o número de classificações como "Intruso" for maior do que o número de classificações como "Normal", o resultado final será definido como "Intruso", caso contrário, será "Nao Intruso". Por fim, o código exibe o resultado final na forma de uma mensagem de texto.

3.7 Testes e resultados experimentais

A acurácia do modelo é um indicador do desempenho global do modelo na classificação dos dados. A saída do modelo é similar à sequência a seguir:

[1, 1, 1, 1, 1, -1, 1...]

Onde o valor "1" indica que o modelo classificou a interação como comportamento "normal" e o valor "-1" indica que o modelo classificou a interação como comportamento "anômalo" ou "intruso".

Para simplificar o entendimento dos resultados, a saída “1” foi trocada por “Normal”, e a saída “-1” foi trocada para “Anomalia”.

A fórmula utilizada para o cálculo da acurácia considera os resultados da validação do modelo:

```
acuracia=(resultados_validacao== 1).sum() / len(resultados_validacao)
```

Para avaliar com mais precisão a eficácia do modelo em um contexto real, foram coletados dados de digitação de cinco pessoas, incluindo o usuário original, que é o proprietário do dispositivo, e outras quatro pessoas não relacionadas. Um total de 18 testes foi conduzido, três com cada usuário. A seleção desse número de indivíduos foi feita com o intuito de abranger uma variedade maior nos padrões de digitação. Durante a interação com o aplicativo, os participantes digitaram no teclado personalizado até atingirem o limite de dez caracteres por três vezes.

Na **Tabela 1** a seguir, são apresentados os dados coletados de cada usuário, acompanhados da taxa de precisão do modelo em cada conjunto de dados. Conforme evidenciado na tabela, o modelo demonstrou uma acurácia satisfatória no reconhecimento do usuário legítimo do dispositivo, mantendo uma taxa de acerto de 80%. Os usuários não

legítimos 1 e 3 obtiveram elevadas taxas de acerto, variando de 70% a 100%, enquanto os usuários não legítimos 4 e 5 apresentaram as taxas de acerto mais baixas. Destaca-se que o modelo exibiu sua menor acurácia no usuário não legítimo 5, com uma taxa de acerto variando entre 20% e 60%. Além disso, o usuário não legítimo 2 apresentou uma variação considerável em sua taxa de acerto, oscilando entre 40% e 100%. No total, foram identificados quatro falsos negativos, nos quais o sistema erroneamente identificou usuários não legítimos como legítimos. Detalhes completos dos dados, juntamente com suas respectivas descrições, estão disponíveis no **Apêndice 1**.

Usuário	Taxa de precisão
Usuário original: Teste 1	80%
Usuário original: Teste 2	80%
Usuário original: Teste 3	80%
Usuário não legítimo 1: Teste 1	80%
Usuário não legítimo 1: Teste 2	100%
Usuário não legítimo 1: Teste 3	90%
Usuário não legítimo 2: Teste 1	60%
Usuário não legítimo 2: Teste 2	40%
Usuário não legítimo 2: Teste 3	60%
Usuário não legítimo 3: Teste 1	100%
Usuário não legítimo 3: Teste 2	60%
Usuário não legítimo 3: Teste 3	70%
Usuário não legítimo 4: Teste 1	50%
Usuário não legítimo 4: Teste 2	60%
Usuário não legítimo 4: Teste 3	60%
Usuário não legítimo 5: Teste 1	30%
Usuário não legítimo 5: Teste 2	20%
Usuário não legítimo 5: Teste 3	60%

Tabela 1: Dados de testes e experimentos. Fonte: Autora

4. CONCLUSÃO

O projeto alcançou com sucesso os objetivos propostos, fornecendo um conhecimento aprofundado sobre a arquitetura de dispositivos móveis e técnicas avançadas de aprendizado de máquina, com ênfase na detecção de anomalias usando o método One-Class Support Vector Machine. Durante o desenvolvimento, foram adquiridas habilidades essenciais de coleta de dados em dispositivos Android e o processo de construção de aplicativos na plataforma utilizando o Android Studio.

O sistema desenvolvido demonstrou resultados promissores ao lidar com usuários com padrões de digitação consideravelmente diferentes. No entanto, foi observada uma limitação ao lidar com usuários com padrões de digitação semelhantes, sugerindo a necessidade de aprimoramentos futuros. Possíveis melhorias incluem testes com um volume maior de caracteres e o treinamento do modelo com uma base de dados mais ampla. Além disso, consideramos a implementação de pesos diferenciados para dados mais ou menos relevantes como uma estratégia que poderia aprimorar a precisão do modelo.

Outra área de aprimoramento importante para futuras iterações é a aplicação do modelo diretamente no dispositivo, evitando a necessidade de verificações no servidor e reduzindo a dependência da conexão com a internet. Infelizmente, essa característica não pôde ser implementada nesta versão do projeto devido a desafios técnicos relacionados à incompatibilidade entre as versões recentes de bibliotecas capazes de carregar modelos treinados em Python. Apesar dessas limitações, o projeto apresentou pontos fortes, com uma taxa de identificação correta de usuários legítimos em 14 dos 18 testes realizados. Com atenção às questões levantadas nesta versão do projeto, acreditamos que versões futuras podem alcançar melhorias substanciais na precisão e eficácia do sistema, sendo uma delas a ampliação do número de caracteres analisados, uma vez que um volume mais amplo de dados pode contribuir para um aprimoramento na detecção de anomalias.

REFERÊNCIA BIBLIOGRÁFICA

ALAM, Mahbub. Support Vector Machine (SVM) for Anomaly Detection. Disponível em 23/11/2023 na URL: https://towardsdatascience.com/support-vector-machine-svm-for-anomaly-detection-73a8d676c331

ALAM, Shamshe et al. One-class support vector classifiers: A survey. Disponível em 23/11/2023 na URL: https://www.sciencedirect.com/science/article/pii/S0950705120301647?casa_token=dqbbXXhnH3gAAAAA:LfnQ883cOe5yg3P1kL7JVcel-YDyqqd6n2IAsW_LqcAjtu7VNDDfGxozpREh4XcKRuoEZDkb8g#b12

ALZUBI, Jafar. Intelligent Systems. Disponível em 23/11/2023 na URL: https://iopscience.iop.org/article/10.1088/1742-6596/1142/1/012012/pdf

ANDROID. Android Security Overview. Disponível em 23/11/2023 na URL: https://source.android.com/docs/security/overview?hl=pt-br

ANDROID. Configuração para desenvolvimento em Android. Disponível em 23/11/2023 na URL: https://source.android.com/docs/setup/about?hl=pt-br

ANDROID. Visão geral da arquitetura. Disponível em 23/11/2023 na URL: https://source.android.com/docs/core/architecture?hl=pt-br

CASTRO, Mayra. Um milhão de celulares foram roubados ou furtados no Brasil em 2022; São Paulo lidera ranking. Disponível em 23/11/2023 na URL: https://oglobo.globo.com/brasil/noticia/2023/07/20/um-milhao-de-celulares-foram-roubados-ou-furtados-no-brasil-em-2022-sao-paulo-lidera-ranking.ghtml

CISCO. What is a worm? Disponível em 23/11/2023 na URL: https://www.cisco.com/c/en/us/products/security/what-is-a-worm.html

COUTINHO, Bernardo. Turing Talks. Disponível em 23/11/2023 na URL: https://medium.com/turing-talks/turing-talks-12-classifica%C3%A7%C3%A3o-por-svm-f4598094a3f1

ESCOVEDO, Tatiana. Machine Learning: Conceitos e Modelos. Disponível em 23/11/2023 na URL: https://tatianaesc.medium.com/machine-learning-conceitos-e-modelos-f0373bf4f445

FONTANA, Eliton Introdução à Inteligência Artificial. Disponível em 23/11/2023 na URL: https://fontana.paginas.ufsc.br/files/2018/03/apostila_ML_pt2.pdf

GILSKI, Przemyslaw, STEFANSKI, Jacek. Android OS: A Review. Disponível em 23/11/2023 na URL: https://www.temjournal.com/content/41/14/temjournal4114.pdf

GOOGLE. How to update the Play Store & apps on Android. Disponível em 23/11/2023 na URL: https://support.google.com/googleplay/answer/113412?hl=en#:~:text=Updating%20your%20apps%20to%20the,improves%20app%20security%20and%20stability.&text=Important%3A,may%20make%20certain%20app%20updates.

GOOGLE. Learn more about Google System updates. Disponível em 23/11/2023 na URL: https://support.google.com/product-documentation/answer/11462338?sjid=11159412029028847387-SA

HARIS, Muhammad et al. International Conference on Advanced Robotics (ICAR). "Robotics and Automation". Disponível em 23/11/2023 na URL: https://apiar.org.au/wp-content/uploads/2017/12/16_ICAR_Nov_BRR708_ICT-125-135.pdf

HEJAZI, Maryamsadat. SINGH P, Yashwant. One-Class Support Vector Machines Approach to Anomaly Detection. Disponível em 23/11/2023 na URL: https://www.tandfonline.com/doi/full/10.1080/08839514.2013.785791

IBM. Machine Learning. Disponível em 23/11/2023 na URL: https://www.ibm.com/br-pt/topics/machine-learning

IBM. Neural Networks. Disponível em 01/11/2023 na URL: https://www.ibm.com/topics/neural-networks

IBM. Ransomware. Disponível em 23/11/2023 na URL: https://www.ibm.com/topics/ransomware

ILR SCHOLL. Spyware, Adware and Viruses. Disponível em 23/11/2023 na URL: https://www.ilr.cornell.edu/faculty-and-staff-resources/technology-services/services/desktop-support/spyware-adware-and-viruses

JOBLIB. Joblib Documentation. Disponível em 23/11/2023 na URL: https://joblib.readthedocs.io/en/stable/

KASPERSKY. Biometrics. Disponível em 23/11/2023 na URL: https://www.kaspersky.com.br/resource-center/definitions/biometrics

KNOLL, Florian. Continuous Authentication using Keystroke Dynamics on Android Devices. Disponível em 23/11/2023 na URL: https://epub.jku.at/obvulihs/download/pdf/4607508?originalFilename=true

LECUN, Yann. et al. Deep Learning. Disponível em 23/11/2023 na URL: https://www.nature.com/articles/nature14539

LOPES, João. CORREIA, Paulo. Keystroke Recognition Using Android Devices. Disponível em 23/11/2023 na URL: file:///home/sthefany/Downloads/Resumo%20alargado-1.pdf

LU, Xiaofeng. Continuous authentication by free-text keystroke based on CNN and RNN. Disponível em 23/11/2023 na URL: https://www.sciencedirect.com/science/article/pii/S0167404820301334

APÊNDICES

Primeiro teste com usuário legítimo

O primeiro teste realizado utilizou os dados de dez caracteres coletados do usuário legítimo para verificação, o teste resultou como “não intruso”. A **Figura 3.7.1** mostra o conjunto de dados verificados.

	Padrão	Padrão	Padrão
1	91	0.0039215684	0.011764707
2	75	0.007843152	0.019607844
3	147	0.035294116	0.011764707
4	96	0.0039215684	0.019607844
5	84	0.0	0.011764707
6	74	0.0039215684	0.019607844
7	85	0.082352936	0.011764707
8	108	0.0039215684	0.015686275
9	92	0.043137252	0.019607844
10	123	0.02352941	0.015686275

Figura 3.7.1.: Dados de digitação do usuário original para o teste 1. Fonte: Autora

A **Figura 3.7.2** a seguir mostra o resultado obtido após a aplicação do modelo, onde cada linha representa os dados da tecla digitada, ‘Dwell Time’, ‘Pressure Difference’ e ‘Area’ respectivamente, como no arquivo CSV. Como é possível observar na Figura 3.7.2, duas saídas do modelo foram classificadas como “Anomalia”, enquanto outras oito foram corretamente classificadas como “Normal”. O resultado final permanece correto, porém a taxa de acerto foi de 80%.

```
Dados: 91, 0.0039215684, 0.011764707, Resultado: Anomalia
Dados: 75, 0.007843152, 0.019607844, Resultado: Normal
Dados: 147, 0.035294116, 0.011764707, Resultado: Normal
Dados: 96, 0.0039215684, 0.019607844, Resultado: Normal
Dados: 84, 0.023093682177777783, 0.011764707, Resultado: Normal
Dados: 74, 0.0039215684, 0.019607844, Resultado: Normal
Dados: 85, 0.082352936, 0.011764707, Resultado: Normal
Dados: 108, 0.0039215684, 0.015686275, Resultado: Normal
Dados: 92, 0.043137252, 0.019607844, Resultado: Anomalia
Dados: 123, 0.02352941, 0.015686275, Resultado: Normal
Resultado Final: Nao intruso
```

Figura 3.7.2: Resultado após verificação do usuario original no teste 1.

Segundo teste com usuário original

O segundo teste realizado também utilizou os dados de dez caracteres digitados pelo usuário legítimo para verificação, o teste resultou como “não intruso”. A **Figura 3.6.3** mostra o conjunto de dados verificados.

	Padrão	Padrão	Padrão
1	86	0.015686274	0.023529414
2	64	0.007843144	0.023529414
3	183	0.03529413	0.023529414
4	117	0.019607842	0.019607844
5	90	0.0039215684	0.015686275
6	113	0.03921569	0.015686275
7	97	0.04313726	0.011764707
8	68	0.0039215684	0.015686275
9	69	0.007843137	0.015686275
10	85	0.0039215684	0.019607844

Figura 3.7.3: Dados de digitação do usuário original para o teste 2. Fonte: Autora

A **Figura 3.7.4** a seguir mostra o resultado obtido após a aplicação do modelo, nesta situação a taxa de acerto também foi de 80%, pois duas saídas foram classificadas como “Anomalia”, enquanto oito foram corretamente classificadas como “Normal”.

```
Dados: 86, 0.015686274, 0.023529414, Resultado: Normal
Dados: 64, 0.007843144, 0.023529414, Resultado: Normal
Dados: 183, 0.03529413, 0.023529414, Resultado: Anomalia
Dados: 117, 0.019607842, 0.019607844, Resultado: Normal
Dados: 90, 0.0039215684, 0.015686275, Resultado: Anomalia
Dados: 113, 0.03921569, 0.015686275, Resultado: Normal
Dados: 97, 0.04313726, 0.011764707, Resultado: Normal
Dados: 68, 0.0039215684, 0.015686275, Resultado: Normal
Dados: 69, 0.007843137, 0.015686275, Resultado: Normal
Dados: 85, 0.0039215684, 0.019607844, Resultado: Normal
Resultado Final: Nao intruso
```

Figura 3.7.4: Resultado após verificação do usuário original no teste 2. Fonte: Autora

Terceiro teste com usuário legítimo

O terceiro teste realizado com o usuário legítimo também resultou corretamente como "Nao Intruso". A **Figura 3.7.5** mostra o conjunto de dados analisados.

	Padrão	Padrão	Padrão
1	68	0.015686274	0.015686275
2	87	0.02352941	0.019607844
3	80	0.007843137	0.023529414
4	111	0.0627451	0.015686275
5	85	0.0039215833	0.015686275
6	90	0.0039215684	0.011764707
7	58	0.023529418	0.019607844
8	90	0.039215684	0.019607844
9	74	0.0039215684	0.011764707
10	100	0.050980397	0.015686275

Figura 3.7.5: Dados de digitação do usuário original para o teste 3. Fonte: Autora

A **Figura 3.7.6** a seguir mostra o resultado obtido após a aplicação do modelo, nesta situação a taxa de acerto também foi de 80%, como nos dois testes anteriores, pois duas saídas foram classificadas como "Anomalia", enquanto oito foram corretamente classificadas como "Normal".

```
Dados: 68, 0.015686274, 0.015686275, Resultado: Normal
Dados: 87, 0.02352941, 0.019607844, Resultado: Normal
Dados: 80, 0.007843137, 0.023529414, Resultado: Normal
Dados: 111, 0.0627451, 0.015686275, Resultado: Normal
Dados: 85, 0.0039215833, 0.015686275, Resultado: Normal
Dados: 90, 0.0039215684, 0.011764707, Resultado: Anomalia
Dados: 58, 0.023529418, 0.019607844, Resultado: Normal
Dados: 90, 0.039215684, 0.019607844, Resultado: Anomalia
Dados: 74, 0.0039215684, 0.011764707, Resultado: Normal
Dados: 100, 0.050980397, 0.015686275, Resultado: Normal
Resultado Final: Nao intruso
```

Figura 3.7.6: Resultado após verificação do usuário original no teste 3. Fonte: Autora

Primeiro teste com o primeiro usuário não legítimo

Nos dados de digitação do segundo usuário podemos perceber na **Figura 3.7.7** que o tempo de pressionamento é diversas vezes maior que o do usuário legítimo. O resultado do teste foi tido corretamente como “Intruso”.

	Padrão	Padrão	Padrão
1	831	0.015686274	0.027450982
2	765	0.07843138	0.027450982
3	1672	0.027450979	0.027450982
4	1655	0.054901958	0.019607844
5	1787	0.03921569	0.019607844
6	941	0.101960786	0.027450982
7	1497	0.10196079	0.027450982
8	1561	0.02352941	0.023529414
9	1846	0.0039215684	0.023529414
10	1141	0.035294116	0.027450982

Figura 3.7.7: Dados de digitação do usuário não legítimo 1 para o teste 1. Fonte: Autora

A **Figura 3.7.8** a seguir mostra o resultado obtido após a aplicação do modelo, nesta situação a taxa de acerto também foi de 80%, com oito saídas corretamente classificadas como “Anomalia” e duas saídas classificadas como “Normal”.

```
Dados: 831, 0.015686274, 0.027450982, Resultado: Anomalia
Dados: 765, 0.07843138, 0.027450982, Resultado: Anomalia
Dados: 1672, 0.027450979, 0.027450982, Resultado: Anomalia
Dados: 1655, 0.054901958, 0.019607844, Resultado: Anomalia
Dados: 1787, 0.03921569, 0.019607844, Resultado: Anomalia
Dados: 941, 0.101960786, 0.027450982, Resultado: Normal
Dados: 1497, 0.10196079, 0.027450982, Resultado: Anomalia
Dados: 1561, 0.02352941, 0.023529414, Resultado: Anomalia
Dados: 1846, 0.0039215684, 0.023529414, Resultado: Anomalia
Dados: 1141, 0.035294116, 0.027450982, Resultado: Normal
Resultado Final: Intruso
```

Figura 3.7.8: Resultado após verificação do usuário não legítimo 1 no teste 1. Fonte: Autora

Segundo teste com o primeiro usuário não legítimo

O segundo teste realizado com o usuário não legítimo 1 também resultou corretamente como “Intruso”. A **Figura 3.7.9** mostra o conjunto de dados analisados.

	Padrão	Padrão	Padrão
4	1179	0.027450979	0.019607844
5	925	0.09411765	0.023529414
6	811	0.086274505	0.019607844
7	783	0.027450979	0.015686275
8	1066	0.039215684	0.015686275
9	1290	0.027450979	0.023529414
10	843	0.031372555	0.015686275

Figura 3.7.9: Dados de digitação do usuário não legítimo 1 para o teste 2. Fonte: Autora

A **Figura 3.7.10** a seguir mostra o resultado obtido após a aplicação do modelo, a taxa de acerto neste teste foi de 100%, onde todas as saídas foram corretamente classificadas como “Anomalia”

```
Dados: 688, 0.048366012, 0.03137255, Resultado: Anomalia
Dados: 1232, 0.031372547, 0.023529414, Resultado: Anomalia
Dados: 1885, 0.07058823, 0.027450982, Resultado: Anomalia
Dados: 1179, 0.027450979, 0.019607844, Resultado: Anomalia
Dados: 925, 0.09411765, 0.023529414, Resultado: Anomalia
Dados: 811, 0.086274505, 0.019607844, Resultado: Anomalia
Dados: 783, 0.027450979, 0.015686275, Resultado: Anomalia
Dados: 1066, 0.039215684, 0.015686275, Resultado: Anomalia
Dados: 1290, 0.027450979, 0.023529414, Resultado: Anomalia
Dados: 843, 0.031372555, 0.015686275, Resultado: Anomalia
Resultado Final: Intruso
```

Figura 3.7.10: Resultado após verificação do usuário não legítimo 1 no teste 2. Fonte: Autora

Terceiro teste com o primeiro usuário não legítimo

Neste terceiro teste com o usuário não legítimo 1, o resultado também foi positivo, as Figuras **3.7.11** e **3.7.12** mostram respectivamente o conjunto de dados analisados e o resultado do teste, que apresentou 90% de acerto, com apenas uma saída classificada erroneamente como “Normal”.

	Padrão	Padrão	Padrão
1	787	0.07058823	0.023529414
2	935	0.113725506	0.023529414
3	568	0.031372547	0.015686275
4	1050	0.09019608	0.019607844
5	494	0.04705882	0.023529414
6	940	0.0745098	0.027450982
7	1015	0.0039215684	0.019607844
8	1343	0.06666666	0.023529414
9	1184	0.0039215684	0.015686275
10	891	0.015686274	0.011764707

Figura 3.7.11: Dados de digitação do usuário não legítimo 1 para o teste 3. Fonte: Autora

```
Dados: 787, 0.07058823, 0.023529414, Resultado: Anomalia
Dados: 935, 0.113725506, 0.023529414, Resultado: Anomalia
Dados: 568, 0.031372547, 0.015686275, Resultado: Anomalia
Dados: 1050, 0.09019608, 0.019607844, Resultado: Anomalia
Dados: 494, 0.04705882, 0.023529414, Resultado: Anomalia
Dados: 940, 0.0745098, 0.027450982, Resultado: Normal
Dados: 1015, 0.0039215684, 0.019607844, Resultado: Anomalia
Dados: 1343, 0.06666666, 0.023529414, Resultado: Anomalia
Dados: 1184, 0.0039215684, 0.015686275, Resultado: Anomalia
Dados: 891, 0.015686274, 0.011764707, Resultado: Anomalia
Resultado Final: Intruso
```

Figura 3.7.12: Resultado após verificação do usuário não legítimo 1 no teste 3. Fonte: Autora

Primeiro teste com o segundo usuário não legítimo

O terceiro usuário analisado, chamado de usuário não legítimo 2, apresenta um tempo médio de “Dwell Time” relativamente menor que o usuário não legítimo 1. A **Figura 3.7.13** mostra o conjunto de dados analisados.

	Padrão	Padrão	Padrão
1	110	0.0039215684	0.015686275
2	290	0.035294123	0.011764707
3	154	0.0039215684	0.019607844
4	106	0.025686281	0.019607844
5	291	0.0039215684	0.019607844
6	194	0.02352941	0.011764707
7	157	0.0	0.019607844
8	170	0.031372547	0.025686275
9	91	0.0039215684	0.011764707
10	210	0.03921569	0.011764707

Figura 3.7.13: Dados de digitação do usuário não legítimo 2 para o teste 1. Fonte: Autora

A **Figura 3.7.14** apresenta os resultados da verificação, onde foi obtida uma acurácia de apenas 60%, com quatro saídas erroneamente classificadas como “Normal”, e seis saídas corretamente classificadas como “Anomalia”.

```
Dados: 110, 0.0039215684, 0.015686275, Resultado: Normal
Dados: 290, 0.035294123, 0.011764707, Resultado: Anomalia
Dados: 154, 0.0039215684, 0.019607844, Resultado: Normal
Dados: 106, 0.025686281, 0.019607844, Resultado: Normal
Dados: 291, 0.0039215684, 0.019607844, Resultado: Anomalia
Dados: 194, 0.02352941, 0.011764707, Resultado: Anomalia
Dados: 157, 0.018976036066666663, 0.019607844, Resultado: Normal
Dados: 170, 0.031372547, 0.025686275, Resultado: Anomalia
Dados: 91, 0.0039215684, 0.011764707, Resultado: Anomalia
Dados: 210, 0.03921569, 0.011764707, Resultado: Anomalia
Resultado Final: Intruso
```

Figura 3.7.14: Resultado após verificação do usuário não legítimo 2 no teste 1. Fonte: Autora

Segundo teste com o segundo usuário não legítimo

O segundo teste com o usuário não legítimo 1 resultou em um falso negativo, onde a taxa de acerto foi de apenas 40%, ou seja, o modelo identificou erroneamente o usuário como “Nao Intruso”, tal resultado é observado na **Figura 3.7.16**.

	Padrão	Padrão	Padrão
1	138	0.015686274	0.019607844
2	145	0.027450979	0.015686275
3	176	0.007843137	0.011764707
4	219	0.07450981	0.023529414
5	122	0.031372547	0.015686275
6	237	0.007843137	0.019607844
7	138	0.027450979	0.015686275
8	265	0.0	0.015686275
9	161	0.019607842	0.019607844
10	80	0.027450979	0.015686275

Figura 3.7.15: Dados de digitação do usuário não legítimo 2 para o teste 2. Fonte: Autora

```
Dados: 138, 0.015686274, 0.019607844, Resultado: Normal
Dados: 145, 0.027450979, 0.015686275, Resultado: Normal
Dados: 176, 0.007843137, 0.011764707, Resultado: Anomalia
Dados: 219, 0.07450981, 0.023529414, Resultado: Anomalia
Dados: 122, 0.031372547, 0.015686275, Resultado: Normal
Dados: 237, 0.007843137, 0.019607844, Resultado: Anomalia
Dados: 138, 0.027450979, 0.015686275, Resultado: Normal
Dados: 265, 0.026579520444444442, 0.015686275, Resultado: Anomalia
Dados: 161, 0.019607842, 0.019607844, Resultado: Normal
Dados: 80, 0.027450979, 0.015686275, Resultado: Normal
Resultado Final: Nao intruso
```

Figura 3.7.16: Resultado após verificação do usuário não legítimo 2 no teste 2. Fonte: Autora

Terceiro teste com o segundo usuário não original

O terceiro teste com o usuário não legítimo 2 teve como conjunto de dados os valores contidos na **Figura 3.7.17**, e resultou em uma taxa de acerto de 60%, onde apenas seis saídas foram classificadas corretamente como “Anomalia”, como é observado na **Figura 3.7.18**.

	Padrão	Padrão	Padrão
1	99	0.0	0.023529414
2	228	0.0039215684	0.03137255
3	258	0.0039215684	0.027450982
4	244	0.0	0.03137255
5	309	0.0039215684	0.043137256
6	312	0.0	0.050980397
7	245	0.043137267	0.039215688
8	140	0.0039215684	0.03137255
9	281	0.0	0.04705883
10	131	0.0039215684	0.043137256

Figura 3.7.17: Dados de digitação do usuário não legítimo 2 para o teste 3. Fonte: Autora

```
Dados: 99, 0.010457518166666667, 0.023529414, Resultado: Normal
Dados: 228, 0.0039215684, 0.03137255, Resultado: Anomalia
Dados: 258, 0.0039215684, 0.027450982, Resultado: Anomalia
Dados: 244, 0.010457518166666667, 0.03137255, Resultado: Anomalia
Dados: 309, 0.0039215684, 0.043137256, Resultado: Anomalia
Dados: 312, 0.010457518166666667, 0.050980397, Resultado: Normal
Dados: 245, 0.043137267, 0.039215688, Resultado: Anomalia
Dados: 140, 0.0039215684, 0.03137255, Resultado: Normal
Dados: 281, 0.010457518166666667, 0.04705883, Resultado: Anomalia
Dados: 131, 0.0039215684, 0.043137256, Resultado: Normal
Resultado Final: Intruso
```

Figura 3.7.18: Resultado após verificação do usuário não legítimo 2 no teste 3. Fonte: Autora

Primeiro teste com o terceiro usuário não legítimo

O segundo teste realizado também utilizou os dados de dez caracteres digitados pelo usuário original para verificação, o teste resultou como “Intruso”. A **Figura 3.7.19** mostra o conjunto de dados verificados.

	Padrão	Padrão	Padrão
1	522	0.0	0.03529412
2	598	0.019607842	0.023529414
3	660	0.06666667	0.027450982
4	640	0.0	0.03529412
5	609	0.06666667	0.03137255
6	645	0.0039215684	0.03137255
7	624	0.035294116	0.027450982
8	614	0.0	0.03137255
9	531	0.011764705	0.03137255
10	416	0.007843137	0.023529414

Figura 3.7.19. Dados de digitação do usuário não legítimo 3 para o teste 1. Fonte: Autora

A **Figura 3.7.20** a seguir mostra o resultado obtido após a aplicação do modelo, nesta situação a taxa de acerto foi de 100%, com todas as dez saídas corretamente classificadas como “Anomalia”.

```
Dados: 522, 0.0302521012, 0.03529412, Resultado: Anomalia
Dados: 598, 0.019607842, 0.023529414, Resultado: Anomalia
Dados: 660, 0.06666667, 0.027450982, Resultado: Anomalia
Dados: 640, 0.0302521012, 0.03529412, Resultado: Anomalia
Dados: 609, 0.06666667, 0.03137255, Resultado: Anomalia
Dados: 645, 0.0039215684, 0.03137255, Resultado: Anomalia
Dados: 624, 0.035294116, 0.027450982, Resultado: Anomalia
Dados: 614, 0.0302521012, 0.03137255, Resultado: Anomalia
Dados: 531, 0.011764705, 0.03137255, Resultado: Anomalia
Dados: 416, 0.007843137, 0.023529414, Resultado: Anomalia
Resultado Final: Intruso
```

Figura 3.7.20: Resultado após verificação do usuário não legítimo 3 no teste 1. Fonte: Autora

Segundo teste com o terceiro usuário não original

Neste segundo teste com o usuário não legítimo 3, o modelo apresentou resultados não tão satisfatórios como no teste anterior, neste, apenas 60% das saídas foram corretamente classificadas como “Anomalia”. **A Figura 3.7.21** a seguir mostra o conjunto de dados utilizado neste teste, e a **Figura 3.7.22** apresenta os resultados encontrados.

	Padrão	Padrão	Padrão
1	514	0.0039215833	0.019607844
2	229	0.09019608	0.015686275
3	249	0.08235294	0.019607844
4	517	0.09803922	0.019607844
5	493	0.06666666	0.015686275
6	58	0.0	0.019607844
7	559	0.05098039	0.019607844
8	327	0.08627451	0.011764707
9	322	0.035294116	0.011764707
10	157	0.031372547	0.011764707

Figura 3.7.21: Dados de digitação do usuário não legítimo 3 para o teste 2. Fonte: Autora

```
Dados: 514, 0.0039215833, 0.019607844, Resultado: Anomalia
Dados: 229, 0.09019608, 0.015686275, Resultado: Anomalia
Dados: 249, 0.08235294, 0.019607844, Resultado: Anomalia
Dados: 517, 0.09803922, 0.019607844, Resultado: Anomalia
Dados: 493, 0.06666666, 0.015686275, Resultado: Anomalia
Dados: 58, 0.06056644958888889, 0.019607844, Resultado: Normal
Dados: 559, 0.05098039, 0.019607844, Resultado: Anomalia
Dados: 327, 0.08627451, 0.011764707, Resultado: Normal
Dados: 322, 0.035294116, 0.011764707, Resultado: Normal
Dados: 157, 0.031372547, 0.011764707, Resultado: Normal
Resultado Final: Intruso
```

Figura 3.7.22: Resultado após verificação do usuário não legítimo 3 no teste 2. Fonte: Autora

Terceiro teste com o terceiro usuário não legítimo

No último teste com o usuário não legítimo 3, foi obtida uma taxa de acerto de 70%, na **Figura 3.7.24** podemos ver que apenas três saídas foram erroneamente classificadas como "Normal".

	Padrão	Padrão	Padrão
1	467	0.0039215684	0.027450982
2	347	0.10588237	0.027450982
3	376	0.031372547	0.019607844
4	467	0.054901972	0.019607844
5	322	0.058823526	0.019607844
6	442	0.082352936	0.019607844
7	618	0.031372547	0.027450982
8	551	0.113725506	0.023529414
9	445	0.078431375	0.015686275
10	405	0.035294116	0.015686275

Figura 3.7.23: Dados de digitação do usuário não legítimo 3 para o teste 3. Fonte: Autora

```
Dados: 467, 0.0039215684, 0.027450982, Resultado: Anomalia
Dados: 347, 0.10588237, 0.027450982, Resultado: Normal
Dados: 376, 0.031372547, 0.019607844, Resultado: Normal
Dados: 467, 0.054901972, 0.019607844, Resultado: Anomalia
Dados: 322, 0.058823526, 0.019607844, Resultado: Normal
Dados: 442, 0.082352936, 0.019607844, Resultado: Anomalia
Dados: 618, 0.031372547, 0.027450982, Resultado: Anomalia
Dados: 551, 0.113725506, 0.023529414, Resultado: Anomalia
Dados: 445, 0.078431375, 0.015686275, Resultado: Anomalia
Dados: 405, 0.035294116, 0.015686275, Resultado: Anomalia
Resultado Final: Intruso
```

Figura 3.7.2: Resultado após verificação do usuário não legítimo 3 no teste 3. Fonte: Autora

Primeiro teste com o quarto usuário não legítimo

Durante os testes com o usuário não legítimo 4 foi possível perceber que o sistema se manteve constantemente dividido, entre “Normal” e “Anomalia”. No primeiro teste, o resultado foi um falso negativo, ou seja, o usuário foi classificado como “Nao Intruso”, apesar de se tratar de um usuário não legítimo. Os dados avaliados são encontrados na **Figura 3.7.25** a seguir.

	Padrão	Padrão	Padrão
1	163	0.08235294	0.027450982
2	205	0.105882354	0.019607844
3	203	0.058823533	0.027450982
4	138	0.09019608	0.023529414
5	216	0.058823533	0.019607844
6	161	0.10980392	0.019607844
7	201	0.06666666	0.019607844
8	66	0.027450979	0.023529414
9	140	0.0039215684	0.019607844
10	188	0.101960786	0.019607844

Figura 3.7.25: Dados de digitação do usuário não legítimo 4 para o teste 1. Fonte: Autora

A **Figura 3.7.26** a seguir mostra os resultados obtidos aplicando o modelo no conjunto de dados, observa-se que o modelo classificou 50% dos dados como “Anomalia” e 50% como “Normal” resultando em “Nao Intruso”, apesar de se tratar de um “Intruso”.

```
Dados: 163, 0.08235294, 0.027450982, Resultado: Normal
Dados: 205, 0.105882354, 0.019607844, Resultado: Anomalia
Dados: 203, 0.058823533, 0.027450982, Resultado: Anomalia
Dados: 138, 0.09019608, 0.023529414, Resultado: Normal
Dados: 216, 0.058823533, 0.019607844, Resultado: Anomalia
Dados: 161, 0.10980392, 0.019607844, Resultado: Normal
Dados: 201, 0.06666666, 0.019607844, Resultado: Anomalia
Dados: 66, 0.027450979, 0.023529414, Resultado: Normal
Dados: 140, 0.0039215684, 0.019607844, Resultado: Normal
Dados: 188, 0.101960786, 0.019607844, Resultado: Anomalia
Resultado Final: Nao intruso
```

Figura 3.7.26: Resultado após verificação do usuário não legítimo 4 no teste 1. Fonte: Autora

Segundo teste com o quarto usuário não legítimo

Neste segundo teste o intruso foi identificado, porém o modelo classificou erroneamente quatro saídas como “Normal”, e seis como “Anomalia”, chegando a uma taxa de acerto de 60%.

	Padrão	Padrão	Padrão
1	324	0.09411765	0.019607844
2	239	0.09019609	0.03137255
3	349	0.09411765	0.03529412
4	432	0.08627452	0.039215688
5	202	0.0	0.039215688
6	304	0.0039215684	0.03137255
7	152	0.015686274	0.027450982
8	136	0.019607842	0.019607844
9	229	0.101960786	0.023529414
10	201	0.058823526	0.027450982

Figura 3.7.27: Dados de digitação do usuário não legítimo 4 para o teste 2. Fonte: Autora

A **Figura 3.7.28** mostra os resultados para o conjunto de dados do teste 2 com o usuário não legítimo 4.

```
Dados: 324, 0.09411765, 0.019607844, Resultado: Normal
Dados: 239, 0.09019609, 0.03137255, Resultado: Anomalia
Dados: 349, 0.09411765, 0.03529412, Resultado: Normal
Dados: 432, 0.08627452, 0.039215688, Resultado: Anomalia
Dados: 202, 0.06274510071111111, 0.039215688, Resultado: Anomalia
Dados: 304, 0.0039215684, 0.03137255, Resultado: Anomalia
Dados: 152, 0.015686274, 0.027450982, Resultado: Normal
Dados: 136, 0.019607842, 0.019607844, Resultado: Normal
Dados: 229, 0.101960786, 0.023529414, Resultado: Anomalia
Dados: 201, 0.058823526, 0.027450982, Resultado: Anomalia
Resultado Final: Intruso
```

Figura 3.7.28: Resultado após verificação do usuário não legítimo 4 no teste 2. Fonte: Autora

Terceiro teste com o quarto usuário não legítimo

O terceiro teste com o usuário não legítimo 4 foi semelhante ao segundo teste do mesmo, onde foi encontrada uma taxa de acerto de 60%. As **Figuras 3.7.29** e **3.7.30** mostram respectivamente os dados coletados do usuário e os resultados obtidos com a aplicação do modelo.

	Padrão	Padrão	Padrão
1	219	0.101960786	0.027450982
2	227	0.0039215684	0.03137255
3	227	0.08627451	0.023529414
4	388	0.09411766	0.027450982
5	133	0.007843137	0.027450982
6	142	0.039215684	0.019607844
7	229	0.09019608	0.023529414
8	118	0.062745094	0.023529414
9	164	0.062745094	0.023529414
10	226	0.0039215684	0.023529414

Figura 3.7.29: Dados de digitação do usuário não legítimo 4 para o teste 3. Fonte: Autora

```
Dados: 219, 0.101960786, 0.027450982, Resultado: Anomalia
Dados: 227, 0.0039215684, 0.03137255, Resultado: Anomalia
Dados: 227, 0.08627451, 0.023529414, Resultado: Anomalia
Dados: 388, 0.09411766, 0.027450982, Resultado: Normal
Dados: 133, 0.007843137, 0.027450982, Resultado: Normal
Dados: 142, 0.039215684, 0.019607844, Resultado: Normal
Dados: 229, 0.09019608, 0.023529414, Resultado: Anomalia
Dados: 118, 0.062745094, 0.023529414, Resultado: Normal
Dados: 164, 0.062745094, 0.023529414, Resultado: Anomalia
Dados: 226, 0.0039215684, 0.023529414, Resultado: Anomalia
Resultado Final: Intruso
```

Figura 3.7.30: Resultado após verificação do usuário não legítimo 4 no teste 3. Fonte: Autora

Primeiro teste com o quinto usuário não legítimo

O primeiro teste com o quinto e último usuário não legítimo resultou no terceiro falso negativo do projeto. A **Figura 3.6.31** apresenta os dados coletados, neles é possível observar que os valores são mais próximos dos valores obtidos com o usuário legítimo, dono do aparelho, em comparação com os outros usuários.

	Padrão	Padrão	Padrão
1	165	0.019607842	0.03137255
2	87	0.04705883	0.027450982
3	117	0.06666668	0.027450982
4	91	0.04705882	0.015686275
5	107	0.054901958	0.023529414
6	122	0.007843137	0.023529414
7	86	0.04705883	0.019607844
8	130	0.007843137	0.027450982
9	75	0.035294116	0.019607844
10	91	0.0039215684	0.023529414

Figura 3.7.31: Dados de digitação do usuário não legítimo 5 para o teste 1. Fonte: Autora

A **Figura 3.7.32** apresenta os resultados obtidos ao aplicar o modelo no conjunto de dados, evidenciando o pior desempenho até então, com uma taxa de acerto de apenas 30%.

```
Dados: 165, 0.019607842, 0.03137255, Resultado: Anomalia
Dados: 87, 0.04705883, 0.027450982, Resultado: Normal
Dados: 117, 0.06666668, 0.027450982, Resultado: Normal
Dados: 91, 0.04705882, 0.015686275, Resultado: Anomalia
Dados: 107, 0.054901958, 0.023529414, Resultado: Normal
Dados: 122, 0.007843137, 0.023529414, Resultado: Normal
Dados: 86, 0.04705883, 0.019607844, Resultado: Normal
Dados: 130, 0.007843137, 0.027450982, Resultado: Normal
Dados: 75, 0.035294116, 0.019607844, Resultado: Normal
Dados: 91, 0.0039215684, 0.023529414, Resultado: Anomalia
Resultado Final: Nao intruso
```

Figura 3.7.32: Resultado após verificação do usuário não legítimo 5 no teste 1. Fonte: Autora

Segundo teste com o quinto usuário não legítimo

O segundo teste com o quinto usuário não legítimo também resultou em um falso negativo, porém, com uma taxa de acerto 20% maior que a do teste anterior, resultando em 50% de acerto. Os dados coletados podem ser encontrados na **Figura 3.7.33** a seguir, enquanto os resultados são apresentados na **Figura 3.7.34.**

	Padrão	Padrão	Padrão
1	90	0.04763605	0.023529414
2	133	0.00058823	0.027450982
3	75	0.001764705	0.019607844
4	94	0.04721507	0.022519515
5	108	0.031372547	0.011764707
6	89	0.04610405	0.022559717
7	95	0.035294116	0.015686275
8	88	0.04564705	0.024549224
9	214	0.03529413	0.029607844
10	143	0.07057823	0.023529414

Figura 3.7.33: Dados de digitação do usuário não legítimo 5 para o teste 2. Fonte: Autora

```
Dados: 90, 0.04763605, 0.023529414, Resultado: Anomalia
Dados: 133, 0.00058823, 0.027450982, Resultado: Normal
Dados: 75, 0.001764705, 0.019607844, Resultado: Normal
Dados: 94, 0.04721507, 0.022519515, Resultado: Anomalia
Dados: 108, 0.031372547, 0.011764707, Resultado: Normal
Dados: 89, 0.04610405, 0.022559717, Resultado: Anomalia
Dados: 95, 0.035294116, 0.015686275, Resultado: Normal
Dados: 88, 0.04564705, 0.024549224, Resultado: Anomalia
Dados: 214, 0.03529413, 0.029607844, Resultado: Anomalia
Dados: 143, 0.07057823, 0.023529414, Resultado: Normal
Resultado Final: Nao intruso
```

Figura 3.7.34: Resultado após verificação do usuário não legítimo 5 no teste 2. Fonte: Autora

Terceiro teste com o quinto usuário não legítimo

Neste último teste o modelo conseguiu identificar corretamente o usuário como "Intruso", na **Figura 3.7.35** são apresentados os dados coletados, e na **Figura 3.7.36** estão os resultados obtidos ao aplicar o modelo nos dados. Foi obtida uma taxa de acerto de 60%.

	Padrão	Padrão	Padrão
1	192	0.0039215684	0.027450982
2	143	0.0	0.023529414
3	92	0.011764705	0.019607844
4	85	0.0	0.019607844
5	109	0.007843137	0.023529414
6	178	0.007843137	0.023529414
7	119	0.0039215684	0.023529414
8	89	0.023215684	0.023529414
9	88	0.0039215684	0.04705883
10	178	0.0	0.043137256

Figura 3.7.35. Dados de digitação do usuário não legítimo 5 para o teste 3. Fonte: Autora

```
Dados: 192, 0.0039215684, 0.027450982, Resultado: Anomalia
Dados: 143, 0.00878431315, 0.023529414, Resultado: Normal
Dados: 92, 0.011764705, 0.019607844, Resultado: Anomalia
Dados: 85, 0.00878431315, 0.019607844, Resultado: Normal
Dados: 109, 0.007843137, 0.023529414, Resultado: Normal
Dados: 178, 0.007843137, 0.023529414, Resultado: Anomalia
Dados: 119, 0.0039215684, 0.023529414, Resultado: Normal
Dados: 89, 0.023215684, 0.023529414, Resultado: Anomalia
Dados: 88, 0.0039215684, 0.04705883, Resultado: Anomalia
Dados: 178, 0.00878431315, 0.043137256, Resultado: Anomalia
Resultado Final: Intruso
```

Figura 3.7.36: Resultado após verificação do usuário não legítimo 5 no teste 3. Fonte: Autora