DIEGO AMARAL SOUZA
JONATHAN YUITI SUZUKI
THIAGO CORDEIRO DA FONSECA

BIOMETRIA COMPORTAMENTAL PARA IDENTIFICAÇÃO DE USUÁRIOS EM PLATAFORMA MOBILE

São Paulo

2018

DIEGO AMARAL SOUZA
JONATHAN YUITI SUZUKI
THIAGO CORDEIRO DA FONSECA

BIOMETRIA COMPORTAMENTAL PARA IDENTIFICAÇÃO DE USUÁRIOS EM PLATAFORMA MOBILE

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do título de bacharel em Engenharia Elétrica

São Paulo

2018

DIEGO AMARAL SOUZA
JONATHAN YUITI SUZUKI
THIAGO CORDEIRO DA FONSECA

BIOMETRIA COMPORTAMENTAL PARA IDENTIFICAÇÃO DE USUÁRIOS EM PLATAFORMA MOBILE

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do título de bacharel em Engenharia Elétrica

Área de Concentração:

Inteligência Artificial

Orientador:

Profª. Drª. Anarosa Alves Franco Brandão

São Paulo

2018

Resumo

Os métodos de autenticação adotados atualmente em smartphones, como PINs e padrões de desbloqueio, possuem vulnerabilidades como “shoulder surfing” (ato de espiar uma pessoa colocar a sua senha) e roubos ocorridos após a autenticação do usuário. Abordagens de autenticação com foco no comportamento do usuário tem sido alvo de estudo, uma vez que elas contornam estas vulnerabilidades. A partir de dados produzidos através dos diversos sensores presentes nos dispositivos móveis, propõe-se desenvolver um aplicativo capaz de coletar dados biométricos do usuário. Dados como pressão do toque, área de contato com a tela, velocidade do ponto de contato, inclinação do dispositivo, entre outros serão utilizados para treinar um modelo de inteligência artificial capaz de identificar um usuário específico. Essa identificação será dada baseando-se nas sutis diferenças de comportamento, identificadas pela análise dos dados coletados, que caracterizam o usuário de forma única.

São Paulo

2018

Lista de Ilustrações

São Paulo

2018

Lista de Tabelas

São Paulo

2018

Sumário

São Paulo

2018

São Paulo

2018

1 INTRODUÇÃO

A cada dia cresce a utilização de computadores e celulares para acessar informações confidenciais como contas bancárias, realizar atividades críticas como compras online com cartão de crédito e transferências bancárias [2][3]. Em todos estes casos, a autenticação do usuário é necessária. Tal autenticação é geralmente feita por meio de senhas. Entretanto, sabe-se que a maior parte dos usuários utiliza senhas não-seguras e/ou as reutiliza em diversos outros serviços [1]. Aind, mesmo que a senha seja segura, não impede que o cliente seja assaltado e coagido a informá-la para o fraudador ou que sua senha seja capturada via *shoulder surfing* (ato de espiar uma pessoa colocar a sua senha) [8]. Além disso, não é incomum que sistemas sejam invadidos e vazem as informações, inclusive senhas, de seus usuários [7].

O poder computacional crescente [4], o grande número de sensores presentes em dispositivos móveis [5] e os avanços nas áreas de inteligência artificial, notadamente em aprendizado de máquina [6] constituem um solo fértil para desenvolvimento de novas técnicas de autenticação mais seguras e transparentes para o usuário.

O objetivo deste trabalho é amenizar os problemas de autenticação pelo uso de senhas quando o atacante obtém a senha do usuário ou quando o dispositivo já desbloqueado é roubado. Esses problemas podem ser atacados através da autenticação contínua por meio de biometria comportamental, dessa maneira o atacante teria que além de roubar a senha do usuário ser capaz de mimicar o modo de uso do real dono do aparelho.

Para isto pretende-se elaborar um protótipo de um aplicativo para dispositivo móvel capaz de coletar dados comportamentais de um usuário, como toque e inclinação de um smartphone. Planeja-se utilizar técnicas de inteligência artificial para criar um modelo de aprendizado de máquina capaz de identificar e autenticar um usuário a partir dessas características, permitindo (ou não) o acesso a informações sensíveis. Ao final espera-se que esse sistema seja capaz de fornecer uma experiência mais segura ao usuário, sem prejudicar a usabilidade ou adicionar camadas de complexidade.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão introduzidos os aspectos fundamentais para a compreensão deste trabalho. Primeiramente será introduzido o conceito de autenticação, seguindo para como biometria comportamental ajuda a minimizar os problemas comuns de autenticação. Em seguida, discute-se a coleta de dados necessários para o estabelecimento de um perfil biométrico comportamental e quais os algoritmos de inteligência artificial que podem ser utilizados para tratar os dados do perfil biométrico e maximizar a segurança. Por fim, descrevem-se as métricas utilizadas para avaliar o desempenho da solução.

2.1 Autenticação

Pelo dicionário, a palavra “autêntico” significa “de origem (época, fabricação, localidade) comprovada” ou “cuja autoria é atestada”. Em computação, autenticar é provar a identidade fornecida pelo usuário. Normalmente, a identidade é fornecida através de um *login* e a prova de identidade pode ser feita de três métodos diferentes: por algo que o usuário sabe (como senhas ou perguntas pessoais), por algo que o usuário possui (como tokens ou cartões) ou por biometria (como digitais ou reconhecimento de padrões de digitação) [10].

O método mais comum e amplamente conhecido é o que o usuário sabe. Um dos problemas comuns dessa abordagem é a criação de senhas não seguras e o reuso das senhas [1]. A abordagem adotada pretende acrescentar uma camada de segurança à autenticação com uso de senha, ao introduzir a biometria associada à digitação da senha.

2.2 Biometria

Biometria é a medição de características biológicas. Essas características podem ser fisiológicas (estáticas), como reconhecimento da íris, ou comportamentais (dinâmicas), como a maneira que uma pessoa utiliza um smartphone [10]. Este trabalho tem como foco a biometria comportamental. A vantagem de se combinar biometria com o que o usuário sabe é que mesmo que o invasor consiga obter a senha, dificilmente ele será capaz de mimicar as nuances do comportamento do usuário verdadeiro, sendo possível bloquear o invasor nesses casos.

2.3 Captura de dados

Para realizar a medição e a autenticação por meio de biometria é necessária a captação de dados dos usuários que possibilitem a definição de seu comportamento. Tal captação é feita com o uso de sensores. Um sensor pode ser definido como um dispositivo que recebe um estímulo físico e responde com um sinal elétrico [9].

Os sensores comumente disponíveis nos dispositivos móveis atuais são acelerômetro, giroscópio, magnetômetro e de toque. Cada um deles capta um estímulo físico e o traduz para um sinal elétrico interpretado pelo smartphone. O acelerômetro mede a aceleração linear do dispositivo nas direções *x*, *y* e *z* e gera sinais elétricos possíveis de serem convertidos para metros por segundo ao quadrado. O giroscópio e o magnetômetro medem, respectivamente, a aceleração angular do dispositivo e o campo magnético nas direções *x*, *y* e *z* da mesma maneira que o acelerômetro. O sensor de toque responde a variação de capacitância sentida na tela com estímulos elétricos que determinam a localização, área e pressão do toque.

Durante a utilização do smartphone, o aplicativo realizou a captura dos dados de cada um desses sensores ao longo do tempo. Os dados coletados através desses sensores permitem a construção de um perfil biométrico do usuário.

2.4 HMOG

Uma das características possíveis de se extrair dos dados que fazem parte do perfil biométrico são as características de *Hand Movement, Orientation and Grasp (HMOG)* introduzidas na referência [11]. Essas características são criadas utilizando combinações dos dados levantados pelos sensores e foram divididas em resistência ao toque e estabilidade ao toque.

2.4.1 Resistência ao toque

Mede o comportamento de resistência a cada toque. Mais especificamente:

- Valor médio de cada um dos atributos dos sensores de posição (acelerômetro, giroscópio e magnetômetro) durante o toque;
- Desvio padrão de cada um dos atributos durante o toque;

- A diferença da média de leitura de cada um dos atributos 100ms antes e 100ms depois do toque;
- A mudança em cada um dos atributos causados por um toque, diferença da média durante o toque com a média de 100ms antes do toque;
- Mudança máxima em cada um dos atributos causados por um toque, diferença do máximo atingido com a média de 100ms antes do toque.

2.4.2 Estabilidade ao toque

Mede os tempos para a estabilização do dispositivo durante o toque:

- Tempo para atingir estabilidade após o evento de toque. A estabilidade é definida como o ponto em que as leituras dos atributos de posição após o toque são mais parecidas com as leituras precedentes ao toque.
- Quantidade de tempo normalizada para a leitura do sensor mudar em uma unidade antes e depois de um toque.
- Quantidade de tempo normalizada para a leitura do sensor mudar em uma unidade até o evento de mudança máxima do toque.

2.5 Aprendizado de máquina

Para conseguir distinguir o perfil biométrico levantado para o usuário e conseguir determinar se o usuário realmente é quem diz ser, serão utilizados algoritmos de aprendizado de máquina.

Em uma definição geral, é possível dizer que uma máquina aprende quando ela é capaz de mudar o seu comportamento através de um estudo diligente de suas próprias experiências [12] [13] de maneira que seu desempenho melhore.

Os algoritmos de aprendizado de máquina podem ser divididos em três grandes categorias: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço [13] [14]. Neste trabalho a ênfase foi o aprendizado supervisionado.

2.6 Aprendizado supervisionado

Nessa abordagem, o algoritmo observa exemplos de pares entrada-e-saída e aprende uma função de mapeamento da entrada para a saída. No caso de estudo, as entradas serão os dados fornecidos pelos sensores e as características adicionais como o HMOG, que terão a classificação de um usuário autêntico ou não. Existem diversos algoritmos que podem ser aplicados e estudaremos a viabilidade de vários deles.

Um resumo sobre o funcionamento deles encontra-se a seguir:

2.6.1 Árvore de decisão

Algoritmo que classifica os registros em categorias por meio de sucessivos testes sobre os atributos dos mesmos [13].

Para criar uma árvore de decisão é necessário acumular registros em um conjunto de treinamento que será separado entre preditores e atributo alvo. Seguindo o paradigma de aprendizado supervisionado os preditores são as entradas e o atributo alvo a saída. A partir desse conjunto cria-se um grafo onde cada nó corresponde a um preditor e, baseado no valor que o registro tem desse preditor, o registro é encaminhado para um dos nós filhos. Este processo se repete até que se atinja uma das folhas da árvore, onde então o registro é finalmente classificado.

O preditor usado em cada um dos nós é aquele que provê um maior ganho de informação, este ganho de informação é calculado através da diminuição de entropia. Entropia é a medida de desordem em um conjunto de dados relativo a uma das suas características (no caso o atributo alvo). Um registro é classificado como positivo se o seu valor é o igual ao desejado no atributo alvo e negativo caso contrário. Com a proporção de registros positivos e negativos é possível calcular a entropia de um conjunto, conforme a fórmula a seguir:

$$entropia(S) = -\big(pp(S)\big) * log_2(pp(S) - \big(pn(S)\big) * log_2(pn(S))$$

Em que:

$$S = conjunto\ de\ dados\ em\ um\ nó$$

$$pp(S) = proporção\ de\ positivos\ em\ S$$

$$pn(S) = proporção\ de\ negativos\ em\ S$$

Tendo calculado a entropia em um nó é possível calcular a entropia dos próximos nós em função de um dos preditores, e determinar o ganho para cada um dos preditores:

$$ganho(S, P) = \ entropia(S) - \sum_{S_P \in S} \left(\frac{|S_P|}{|S|}\right) * entropia(S_P)$$

Em que:

$$P = preditor$$

$$S_p = subconjuntos\ de\ S\ criados\ pelo\ preditor\ P$$

$$|S_P| = número\ de\ elementos\ do\ subconjunto$$

$$|S| = número\ de\ elementos\ do\ conjunto$$

Determinando o ganho para cada um dos possíveis preditores é possível obter aquele que traz o maior ganho. Esse será o preditor selecionado para aquele nó. Aplicando recursivamente esse algoritmo é possível obter um grafo que determina a árvore de decisão. Quantos nós e ramificações a árvore deve possuir é específico de cada problema e será tratado durante a implementação do projeto na seção de hiperparâmetros.

2.6.2 *Random forest*

Algoritmo classificador no qual são utilizadas diversas árvores de decisão, porém na construção de cada árvore, a decisão do teste que será aplicado em cada nó utiliza apenas um conjunto parcial aleatório de todas as características disponíveis. A classificação é dada por uma votação entre todas elas, sendo a resposta final aquela com o maior número de votos.

2.6.3 Rede neural

Abordagem que classifica registros através de camadas de neurônios ajustáveis. Cada neurônio recebe uma entrada e decide se ele será ativado, ou seja, se passará o sinal para a próxima camada. A última camada é, normalmente, composta pelas possíveis categorias e a saída é a probabilidade do registro pertencer a cada uma das categorias.

Um neurônio presente em uma camada da rede funciona como uma simples função: ela recebe alguns valores de entrada (denotados aqui de "x_i") e retorna um valor de saída baseada nos seus pesos (denotados de "w_i") e em sua função de ativação (denotado de σ). Portanto um neurônio pode ser representado como:

$$y(x, w) = \sigma(\sum_{i=0}^{n} x_i * w_i)$$

A quantidade “n” de entradas e de pesos que um neurônio tem em uma rede totalmente conectada é o número de neurônios da camada anterior. Para que uma rede neural consiga “aprender” (ou seja, aproximar qualquer função para a função ótima para o problema), a função de ativação deve ser não linear, caso contrário uma rede seria uma composição de funções lineares o que resultaria em uma função também linear. Funções comumente citadas na literatura são a sigmoide [13], tangente hiperbólica [17] e ReLU [18].

O treinamento de uma rede é dividido em duas etapas o *feedforward* e o *backpropagation* sendo que a primeira etapa é bem simples: alimenta-se a rede com as entradas do problema (no caso deste projeto, informações sobre o ângulo do dispositivo, tempo de toque, coordenada do toque, pressão, etc.) e obtém-se uma saída (probabilidade do indivíduo ser quem diz ser). Durante o treinamento tem-se a resposta verdadeira, sabe-se se o indivíduo é quem ele diz ser, portanto é possível calcular o erro da previsão.

$$Erro(x, w) = \frac{1}{2}(y_{real} - y_{previsto}(x, w))^2$$

Como o erro é uma função dos pesos é possível determinar qual foi a influência individual de cada um deles. Utilizando o método de gradiente descendente, calcula-se a derivada parcial do erro em função de cada peso e itera-se sobre cada um eles, conforme a fórmula:

$$w_{i\,novo} = w_i - \alpha * \frac{\partial Erro(x, w)}{\partial w_i}$$

Dessa maneira, a cada iteração, o erro tende a ser minimizado, aproximando a rede neural da função ótima para o problema apresentado. Um hiperparâmetro apresentado na função anterior foi o α, conhecido como taxa de aprendizado. Para que cada iteração aproxime e não distancie a função da aproximação desejada é necessário que o ajuste não seja muito grande, pois caso contrário a taxa de erro tende a oscilar de forma intermitente.

2.6.4 *XGBoost*

Algoritmo muito popular devido a seu bom desempenho e rápido processamento [19]. Assim como o Random Forest, consiste no Ensemble de diversas pequenas árvores.

O Algoritmo busca encontrar o conjunto de parâmetros θ que minimiza a função Objetivo dada por:

$$obj(\theta) = L(\theta) + \Omega(\theta) \quad (1)$$

Onde L é a parcela de perda, que indica o quão boas são as previsões realizadas pelo modelo. É muitas vezes definida como sendo o erro médio quadrático entre os valores alvo (y_i) e os previstos pelo modelo $(\ddot{y}_i)$:

$$L(\theta) = \sum_i (y_i - \ddot{y}_i)^2 \quad (2)$$

A segunda parcela é de regularização, ela penaliza a complexidade do modelo evitando que ocorra *overfitting*. Sendo ω o vetor de pesos das folhas, q a função que associa cada entrada a uma folha e T o número total de folhas de uma árvore, a previsão $f_t(x)$ para um dado de entrada x é denotada por:

$$f_t(x) = \omega_{q(x)} \quad (3)$$

Sendo então γ o coeficiente de penalização de número de folhas e λ o coeficiente de regularização L2, temos que a regularização é:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2 \quad (4)$$

Denotando por $\ddot{y}_i^{(t)}$ a previsão para o i-ésimo dado do treino e para a t-ésima iteração do modelo, temos que ele começa com uma previsão vazia para todas as entradas:

$$\ddot{y}_i^{(0)} = 0 \quad (5)$$

Então, a cada iteração, adicionamos uma nova árvore que corrige os erros da iteração anterior prevendo um valor $f_t(x_i)$, onde x_i é a entrada do modelo, que é adicionado à predição anterior:

$$\ddot{y}_i^{(t)} = \ddot{y}_i^{(t-1)} + f_t(x_i) = \sum_{k=1}^{t} f_k(x_i) \quad (6)$$

A arvore que se deseja adicionar a cada iteração é aquela que minimiza a função objetivo. Substituindo os valores das parcelas apresentadas acima, obtém-se:

$$obj^{(t)} = \sum_{i=1}^{n} l(y_i, \ddot{y}_i^{(t)}) + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2 \quad (7)$$

Substituindo o valor do último previsor adicionado:

$$obj^{(t)} = \sum_{i=1}^{n} l(y_i, \ddot{y}_i^{(t-1)} + f_t(x_i)) + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2 \quad (8)$$

Fazendo a expansão de Taylor da função de perda em torno do ponto da última previsão:

$$obj^{(t)} = \sum_{i=1}^{n} g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2 \quad (9)$$

Com g_i e h_i:

$$g_i = \partial_{\ddot{y}_i^{(t-1)}} l(y_i, \ddot{y}_i{}^{(t-1)}) \; ; \; h_i = \partial^2{}_{\ddot{y}_i^{(t-1)}} l(y_i, \ddot{y}_i{}^{(t-1)}) \quad (10)$$

Substituindo a definição de $f_t(x_i)$ conforme a equação (3):

$$obj^{(t)} = \sum_{i=1}^{n} g_i \omega_{q(x)} + \frac{1}{2} h_i \omega_{q(x)}{}^2 + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2 \quad (11)$$

Sendo i, o conjunto dos dados que são direcionados à folha j:

$$obj^{(t)} = \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i)\omega_j + \frac{1}{2}(\sum_{i \in I_j} h_i + \lambda)\omega_j{}^2] + \gamma T \quad (12)$$

Definindo então G_j e H_j:

$$G_j = \sum_{i \in I_j} g_i \; ; \; H_j = (\sum_{i \in I_j} h_i) \quad (13)$$

Concluímos que a função objetivo é:

$$obj^{(t)} = \sum_{j=1}^{T} [G_j \omega_j + \frac{1}{2}(H_j + \lambda)\omega_j{}^2] + \gamma T \quad [(4)$$

Então, o valor que minimiza a função objetivo e consequentemente indica a melhor árvore teoricamente possível é aquela cujos pesos das folhas têm os valores:

$$\omega_j^* = -\frac{G_j}{H_j + \lambda} \quad (15)$$

Utilizando a função objetivo da equação (14), é possível calcular facilmente a qualidade de uma árvore. Idealmente, todas as estruturas de árvore seriam avaliadas, selecionando a melhor, porém esta abordagem é computacionalmente inviável.

Na implementação real, o algoritmo ordena o conjunto de dados em relação a uma característica e avalia todas as estruturas de árvores formadas pelas divisões em dois subconjuntos.

Nesta avaliação é analisado o ganho produzido pela divisão de uma folha em duas novas. Este ganho pode ser calculado da seguinte forma:

$$Ganho = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right] - \gamma \quad (16)$$

Onde a primeira parcela indica o ganho da folha esquerda, a segunda indica o ganho da folha direita e o terceiro é o ganho da folha original.

2.6.5 Métricas de desempenho

Para avaliar o desempenho do algoritmo escolhido e a sua efetividade em termos de segurança foram utilizadas a Acurácia (taxa de acertos global do classificador) e também a métrica conhecida como F-Score.

$$Acurácia = \frac{tp + tn}{tp + fp + fn + tn} \qquad F - score = \frac{(\beta^2 + 1)Precisão * Revocação}{\beta^2 * Precisão + Revocação}$$

Onde tp, tn, fp e fn indicam o número de verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos respectivamente. Os valores de Precisão (taxa de classificações positivas corretas) e Revocação (taxa de positivos classificados corretamente) são calculados conforme indicado abaixo:

$$Precisão = \frac{tp}{tp + fp} \qquad Revocação = \frac{tp}{tp + fn}$$

A F-Score permite combinar outras duas métricas em uma métrica única, dando diferentes relevâncias para a Precisão e Revocação que a compõe por meio da variação do parâmetro β [34].

Devido ao contexto do problema abordado, existe um claro desbalanceamento entre a relevância da Precisão e da Revocação. O sucesso na autenticação de um usuário ilegítimo (falso positivo) é muito mais grave do que a falha de autenticação de um usuário legítimo (falso negativo).

Como o desejado é aumentar a relevância da Precisão sobre a Revocação, calcula-se a derivada parcial do F-Score com relação à Precisão:

$$\frac{\partial(F - score)}{\partial Precisão} = \frac{\beta^2 * \text{Revocação}^2 + \text{Revocação}^2}{(\beta^2 * Precisão + Revocação)^2}$$

Como o coeficiente β possui um expoente maior no denominador da derivada indicada acima, para aumentar a influência da Precisão deve-se diminuir o valor deste coeficiente, por este motivo o β adotado foi de 0.5.

Outra métrica comumente utilizada relaciona-se a curva "*Receiver Operating Characteristic*" (ROC) do modelo. Esta indica no mesmo gráfico as taxas de verdadeiros positivos e falsos positivos, com cada ponto da curva representando um valor de limiar de classificação, podendo ser utilizada para definir qual o limiar mais adequado.

A área abaixo desta curva é utilizada como medida de desempenho agregada para todos os valores de limiar de classificação, sendo conhecida como "Area Under *Receiver Operating Characteristic Curve*" (AUROC). Em problemas nos quais, como o nosso, existe uma diferença entre a relevância de falsos positivos e negativos, a AUROC não é uma boa métrica, pois não considera um limiar de classificação preferencial.

3 TECNOLOGIAS

A seguir estão descritas as tecnologias que foram estudadas para a implementação do projeto.

3.1 Linguagens de Programação

Nesta seção serão introduzidas as linguagens utilizadas no projeto.

3.1.1 Python

Python é uma linguagem de programação de propósito geral que possui diversas estruturas de dados de alto nível (como *strings*, listas e dicionários) embutidas. O código é automaticamente compilado para *bytecode* e executado, tornando-o uma linguagem viável para scripts, facilitando a implementação e teste de pequenos trechos de códigos. [15]

Além dessa facilidade, uma grande quantidade de ferramentas para análise de dados e bibliotecas para aprendizado de máquina (como *scikit-learn*, *TensorFlow* e *Keras*) se encontram implementadas em Python.

O Python foi utilizado para a implementação do treino do modelo de inteligência artificial, além do seu respectivo teste.

3.1.2 Java

Java é uma linguagem de programação de propósito geral, orientada a objetos [23]. Java é a linguagem nativa para aplicativos Android [24] e, por isso, ela foi utilizada para desenvolver os aplicativos para coleta de dados e teste da autenticação.

3.2 Ambientes de desenvolvimento

Nesta seção serão introduzidos os ambientes utilizados para facilitar o desenvolvimento do projeto.

3.2.1 Jupyter

Jupyter é uma ferramenta *open-source* desenvolvida para se criar e compartilhar documentos que contenham código, equações, gráficos e outras coisas. A estrutura dos *notebooks* (como são chamados os arquivos) é composta por células de código que podem ser

executadas independente umas das outras, enquanto um *kernel* é mantido, gravando os dados de todas as variáveis sendo utilizadas no ambiente [25]. A utilização dele se dá pelo fato de que certas partes de um código de treinamento de um modelo de aprendizado de máquina são mais demoradas que outras. Pela possibilidade de se utilizar células separadas, caso mude-se algum trecho do código, só é necessário mudar aquela parte.

3.2.2 Android Studio

O Android Studio é uma IDE *(Integrated Development Environment)* para o desenvolvimento de aplicativos para a plataforma Android. Foi utilizado pela facilidade em testar e exportar o aplicativo desenvolvido direto para o celular [24].

3.3 Tecnologias para inteligência artificial e aprendizado de máquina

Nesta seção serão introduzidas as tecnologias que auxiliariam na implementação dos algoritmos de inteligência artificial e aprendizado de máquina.

3.3.1 Scikit-learn

Scikit-learn é uma biblioteca de aprendizado de máquina desenvolvida para Python e permite a implementação rápida de algoritmos como árvores de decisão, random forests, regressões logísticas e *clustering* [26]. Além disso, possui integração com outras bibliotecas de Python de análise numérica como *NumPy* e *SciPy*.

3.3.2 TensorFlow

TensorFlow é uma biblioteca *open-source* que fornece um *framework* para computação numérica de alto desempenho, que possui um alto suporte para aplicações relacionadas a aprendizado de máquina e aprendizado profundo. Ela foi inicialmente desenvolvida pelo time do *Google Brain* [27].

3.3.3 Keras

Keras [28] é uma biblioteca de aprendizado profundo para Python que pode utilizar, como *backend*, o *TensorFlow*. Ela implementa diversas funções que possibilitam uma fácil

implementação e rápidos experimentos com ferramentas de aprendizado profundo. Da mesma forma que o *TensorFlow*, pode ser utilizada para implementar redes neurais.

3.4 Tecnologias para servidor

Nesta seção são introduzidas as tecnologias utilizadas para o desenvolvimento do servidor.

3.4.1 Django

Django é um *web-framework* (software que auxilia o desenvolvimento e a implantação de um servidor capaz de servir páginas webs e api) escrito em Python que segue a arquitetura de MVT (Model View Template). Ele é conhecido pelo seu rápido tempo de escrita e fácil gestão de banco de dados permitindo um ciclo de desenvolvimento curto [29].

O *web-framework* foi utilizado como uma api para as requisições de autenticação, além disso, seu ecossistema em Python facilitou a integração com os modelos de inteligência artificial criados ao longo do projeto.

3.4.2 Requisições HTTP

O *Hypertext Transfer Protocol* é um protocolo da camada de aplicação para comunicação do tipo cliente-servidor. O protocolo funciona através do modelo de requisição-resposta. Uma requisição pode ser especificada pelos seguintes parâmetros: o método, a URL, o cabeçalho e o corpo.

O método define qual o tipo de operação o cliente realizará. Os métodos HTTP são padronizados: o método GET é capaz de retornar os dados solicitados e o POST é capaz de submeter dados ao servidor. Existem outros métodos, porém estes não foram utilizados durante o desenvolvimento do projeto.

A URL define qual a fonte de informação deve ser acessada. Nela ainda é possível passar algumas informações adicionais, sendo no entanto limitada a 2048 caracteres.

O cabeçalho permite passar informações relacionadas ao controle da requisição, como o tipo do conteúdo, tamanho do corpo, origem, controle de cache, etc.

O corpo permite a submissão de informações adicionais, sem limitação de tamanho. No entanto, nem todas as requisições possuem corpo.

3.4.3 JSON

O *JSON* é um formato de arquivo para a troca de dados de forma legível para humanos. Ele transmite informações em formato chave-valor.

3.5 Tecnologias auxiliares

Nesta seção serão apresentadas as tecnologias que auxiliaram ao longo do desenvolvimento do projeto.

3.5.1 Matplotlib

Matplotlib é uma biblioteca de auxílio a impressão de gráficos em 2D para Python. Possui uma boa integração com o Jupyter Notebook, facilitando o trabalho de visualização de dados [30].

3.5.2 Numpy

Biblioteca capaz de auxiliar em cálculos complexos através de métodos numéricos para Python, fornecendo diversas funções que operam sobre vetores e matrizes multidimensionais. É extremamente poderosa devido a seu caráter paralelo (capacidade de paralelizar cálculos com vetores e matrizes) e seu alto grau de integração com a biblioteca Pandas [31].

3.5.3 Pandas

Pandas é uma biblioteca para Python para manipulação de dados que fornece estruturas de dados flexíveis e fáceis de manipular. Possui um suporte muito bom e otimização para a manipulação de um alto número de dados, sendo ideal para o trabalho com conjunto de dados grandes. Possui também funções que auxiliam o carregamento e exportação de tabelas para arquivos de texto [32].

3.5.4 Ngrok

Ngrok é um *software* que facilita o acesso de ambientes de desenvolvimento local, através de uma URL pública. Foi utilizada para facilitar que o aplicativo android fosse capaz de se comunicar com o servidor local desenvolvido em Django [33].

4 METODOLOGIA DE TRABALHO

Este projeto foi desenvolvido em cinco partes principais, estruturadas da seguinte forma:

- Estudo de técnicas atuais de autenticação do usuário;
- Levantamento de dados (através de um aplicativo e/ou um conjunto de dados);
- Construção de um modelo capaz de realizar biometria comportamental e implementação de um servidor que possui o modelo;
- Criação de um aplicativo capaz de consumir o resultado do modelo;
- Realização de testes para validar o funcionamento do sistema.

4.1 Levantamento de Dados

Na obtenção dos dados necessita-se de voluntários e, dada essa dificuldade, buscou-se, em trabalhos anteriores, um conjunto de dados com informações sobre a utilização dos *smartphones*.

Dos conjuntos de dados encontrados, selecionou-se um deles [11], tendo em vista as características presentes, o número de usuários, o número de amostras e também a presença de uma documentação robusta sobre como tais amostras foram geradas.

Foi realizado um estudo da documentação, para selecionar as características mais relevantes do conjunto de dados e replicar a coleta de dados utilizada. Isso permitiu que os novos dados coletados fossem incorporados aos dados pré-existentes, reduzindo-se assim o número de voluntários necessários para o aplicativo.

Desenvolveu-se então um aplicativo para a plataforma Android capaz de capturar, de forma transparente ao usuário, os toques na tela e a posição do aparelho durante o seu uso. O aplicativo foi então utilizado por um número reduzido de voluntários mediante a assinatura do termo de consentimento livre e esclarecido presente no anexo A. Os dados capturados foram salvos localmente em arquivos com extensão *.csv*. Posteriormente eles foram extraídos e integrados aos dados dos demais usuários. É importante também dizer que optou-se por utilizar um único dispositivo para todos os voluntários, para que pudéssemos isolar

completamente as características referentes a digitação de uma pessoa (e não características do dispositivo).

4.2 Implementação de um modelo

Com os dados coletados e os dados pré-existentes, a implementação do modelo foi dividida em 5 partes:

- Tratamento dos dados brutos, agrupando-os em janelas de tempo baseado nos *timestamps* de eventos de toque;
- Definição das características a serem utilizadas;
- Definição do tipo de modelo;
- Treino com validação;
- Treino com todos os dados para importação para o servidor.

Essas etapas devem ser vistas como cíclicas, já que os resultados parciais de uma etapa posterior podem fazer com que mudanças aconteçam em etapas prévias.

O servidor implementado não possui requisitos referentes à robustez dado que é só um protótipo para facilitar a implementação do aplicativo protótipo na etapa seguinte.

4.3 Aplicativo protótipo

Nesta etapa, adicionou-se ao aplicativo uma nova tela para testar, de maneira prática, o modelo adotado. O teste consistia em uma “simulação de invasão”, na qual o *login* de um dos usuários, que havia sido submetido à coleta de dados, tentará ser acessado por outro usuário. Dessa forma, o modelo deve identificar que o comportamento não é o mesmo que o aprendido e não autorizar, enviando uma resposta negativa ao aplicativo.

4.4 Teste

Nesta última etapa, voluntários que haviam participado da etapa de levantamentos de dados foram convidados para uma nova sessão em que o aplicativo protótipo e o modelo foram colocados em teste. Cada voluntário tentou se autenticar em um usuário diferente do

seu modelo. Novos voluntários também participaram, a fim de se testar a robustez do modelo a participantes desconhecidos ("*out-of-sample*").

5 ESPECIFICAÇÃO DE REQUISITOS DE SISTEMA

Nessa seção serão descritos os requisitos do sistema. É importante notar que o sistema deve trazer o mínimo de impacto possível para o usuário, sendo praticamente invisível para o consumidor final, por isso o número reduzido de casos de uso. A maior parte do trabalho do sistema está concentrada no modelo produzido que é capaz de garantir os requisitos não funcionais.

5.1 Arquitetura do Sistema

A arquitetura adotada foi a de cliente-servidor para melhor dividir as responsabilidades de cada equipamento. O aplicativo móvel (cliente) deve captar os dados do usuário e o servidor deve autenticar o usuário com base no modelo de aprendizado e dos dados captados.

5.2 Requisitos Funcionais

Essa seção apresenta os requisitos funcionais do sistema, funções que o sistema deve cumprir.

5.2.1 Modo de Coleta

O aplicativo possui um modo de coleta no qual é solicitado que o usuário utilize o aparelho para responder algumas perguntas aleatórias. Essas perguntas estão especificadas no item 5.5.1. Durante essa tarefa o sistema deve coletar e armazenar dados comportamentais que serão utilizados para treinar o modelo de aprendizado de máquina responsável por identificar o usuário durante o modo de autenticação.

5.2.2 Modo de Autenticação

Para verificar a legitimidade do usuário o aplicativo solicita o seu nome de usuário e a resposta de uma pergunta aleatória. A resposta da pergunta aleatória provê dados comportamentais que são utilizados como uma "senha", determinando se o usuário é legítimo ou não.

Assim que a resposta é submetida, o sistema só deve liberar o acesso aos recursos protegidos se os dados comportamentais se mostrarem consistentes com aqueles obtidos durante o modo de coleta.

5.3 Casos de Uso

Abaixo são apresentados os dois casos de uso implementados pelo aplicativo:

Tabela 1 – Autenticar usuário

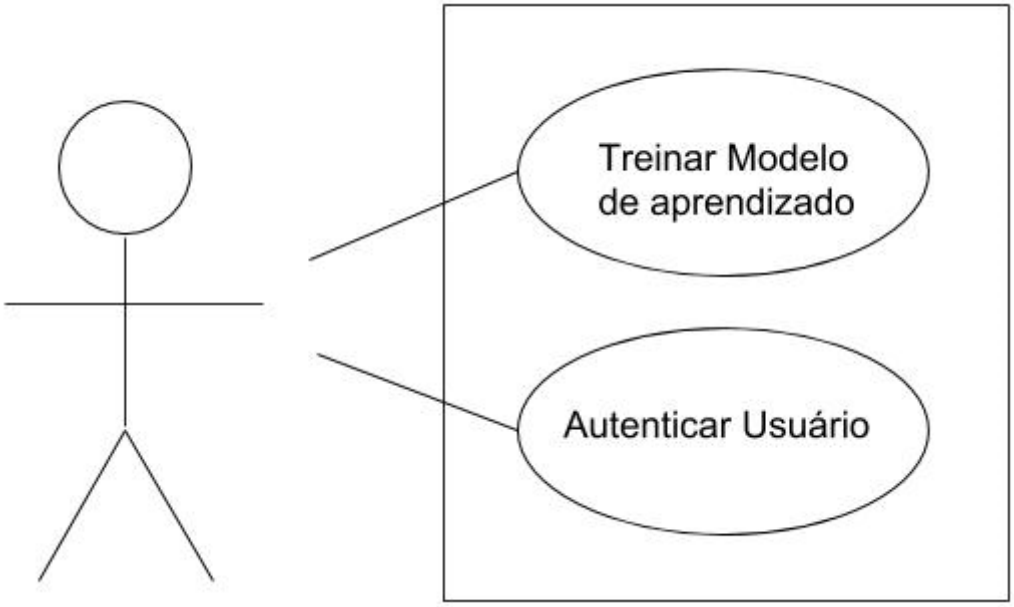
Atores: Usuário Pré-Condição: -
Fluxo Principal: 1. Usuário inicia o sistema no modo de autenticação; 2. Sistema solicita nome de usuário e a resposta de uma pergunta simples selecionada aleatoriamente; 3. Usuário insere as informações solicitadas e, durante esta inserção, o sistema coleta os dados de comportamento do usuário; 4. Sistema verifica que o texto digitado está correto e que os dados comportamentais são consistentes com o histórico do usuário; 5. Sistema libera os recursos protegidos para o usuário legitimo.
Cenário Alternativo (5) 5.a. Caso os dados comportamentais coletados sejam inconsistentes com o histórico do usuário, o sistema exibe uma mensagem informando que o serviço está indisponível e retorna o passo 2.
Pós-Condição: Usuário legítimo tem acesso aos recursos protegidos do sistema.

Tabela 2 – Treinar modelo

Atores: Usuário Pré-Condição: -
Fluxo Principal: 1. Usuário inicia o sistema no modo de coleta; 2. Sistema apresenta três perguntas selecionadas aleatoriamente e uma caixa de texto para cada uma delas; 3. Usuário responde as perguntas enquanto o sistema coleta os dados de comportamento do usuário; 4. Usuário pressiona o botão de submeter; 5. Sistema armazena os dados comportamentais em um arquivo local;
Pós-Condição: Arquivo com os dados comportamentais encontra-se armazenado localmente.

O respectivo diagrama de caso de uso é apresentado a seguir:

Figura 1 - Diagrama de Caso de Uso



Fonte: Autoria Própria

5.4 Requisitos Não Funcionais

Nesta seção serão apresentados os requisitos não funcionais do sistema, ou seja, os requisitos qualitativos do mesmo.

5.4.1 Autenticação Rápida

Não dificultar excessivamente o processo de autenticação do usuário, evitando aumento no tempo de acesso e irritação do cliente.

5.4.2 Eficácia do Modelo de aprendizado

Identificar e impedir de maneira eficaz acessos de usuários não legítimos sem impactar de forma significativa as falhas de autenticação de usuários legítimos.

5.5 Especificações

Nesta seção serão apresentadas as especificações do aplicativo, do modelo de aprendizado e do servidor.

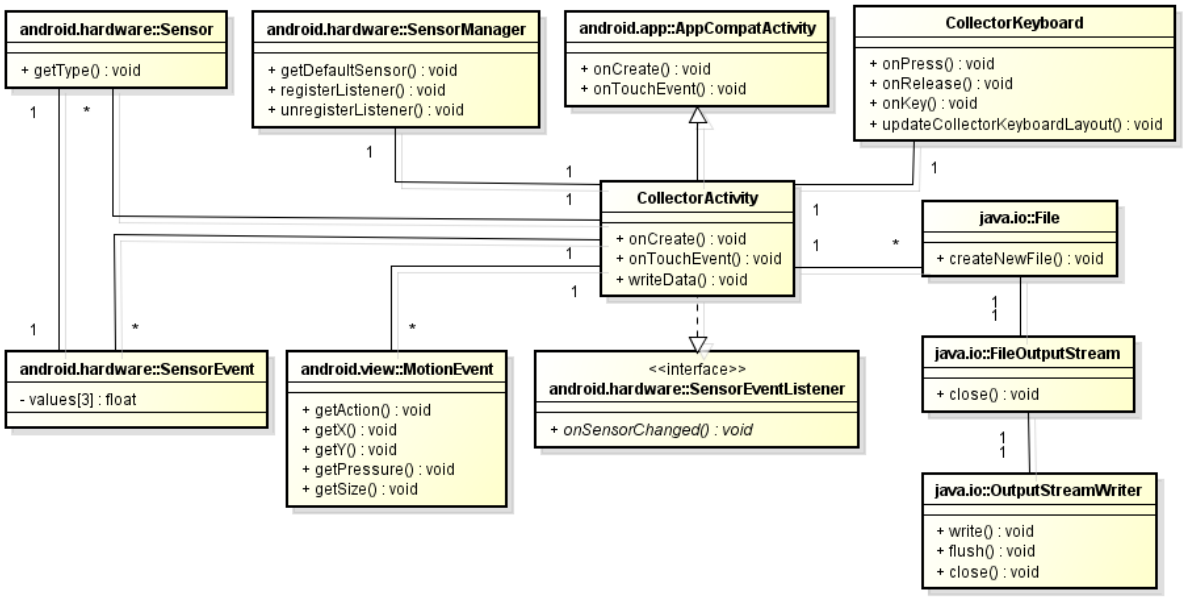
5.5.1 Especificação do Aplicativo para Coleta de Dados e Autenticação

Conforme discutido no item 4.3, foi necessário desenvolver um aplicativo capaz de coletar dados provenientes de diversos sensores dos smartphones. Estes sensores são acelerômetro, giroscópio, magnetômetro e de toque, sendo que os sensores de toque são das categorias *KeyPress* e *KeyBoardTouch*. Esse aplicativo tem dois modos de operação: o modo de coleta de dados e o modo de autenticação.

Para melhorar a legibilidade dos diagramas de classes dividimos em duas figuras: uma apresenta o modo de coleta e outra o modo de autenticação.

A figura a seguir mostra de maneira simplificada a estrutura de classes do modo de coleta do aplicativo:

Figura 2 - Diagrama de Classes (Coleta)



Fonte: Autoria Própria

A classe *CollectorActivity* presente no centro da figura 2 é a responsável por ser a tela principal desse modo e capturar os dados necessários para a construção do conjunto de dados. Para isso, ela possui instâncias das classes *Sensor*, *SensorEvent*, *SensorManager*, *MotionEvent* e *CollectorKeyboard*, que são responsáveis pela captura de tais dados, a partir dos sensores. Além disso, para a escrita de dados, haverá instâncias de objetos da classe *File*, cada uma representando um arquivo diferente, cada qual responsável por armazenar os dados dos sensores.

O *CollectorKeyboard* tem a sua parte visual especificada por um arquivo *xml*, sendo possível determinar o tamanho de cada uma das teclas e o código atrelado a cada uma delas.

Ele foi desenvolvido procurando obter a simulação mais fiel possível ao uso do teclado nativo do Google, presente nos dispositivos *Android*. Com esse novo componente de teclado é possível capturar os dados de toque e a duração deles, dados essenciais para o projeto em desenvolvimento.

Para incentivar os usuários a escrever textos longos, garantindo assim uma coleta de dados rica, três perguntas são selecionadas aleatoriamente de um conjunto com vinte e uma sempre que uma nova seção é iniciada. O conjunto de perguntas estabelecido foi:

• Você acredita que provas convencionais medem de maneira assertiva as capacidades dos alunos?

• Explique para um amigo mais jovem como estudar para o Vestibular.

• Explique como chegar na sua casa sem utilizar nenhum nome de rua.

• Você tem a oportunidade de aprender a tocar um novo instrumento? Qual você escolheria e por quê?

• Você acredita que o Facebook melhora a qualidade de vida das pessoas? Desenvolva.

• Você acredita que vale a pena investir em exploração espacial?

• Desenvolva um argumento a favor ou contra o uso de celulares enquanto dirige.

• O que você gosta e não gosta em relação ao transporte público?

• Explique para uma criança como realizar uma tarefa (e.g. atravessar a rua).

• Desenvolva um argumento a favor ou contra a seguinte declaração: violência televisiva é apropriada para públicos de todas as idades.

• Você acredita que a liberação do porte de armas aumentaria o número de crimes violentos?

• Você acredita que restrições de idade para o consumo de álcool são efetivas?

• As faculdades públicas deveriam continuar sendo gratuitas para todos? Explique sua resposta.

• Você acredita que é aceitável o governo ler o seus emails por motivos anti-terrorismo?

• Descreva o que você fez no último fim de semana.

• Como você convidaria seus amigos para uma festa se telefones e a Internet não existissem?

• Escreva um resumo do enredo do seu filme favorito.

• Os restaurantes deveriam ser obrigados a colocar as informações nutricionais no cardápio?

• Decida uma festa ou evento que você gostaria de organizar e escreva detalhes de como você gostaria de organizar o evento (música, local, convidados, etc.).

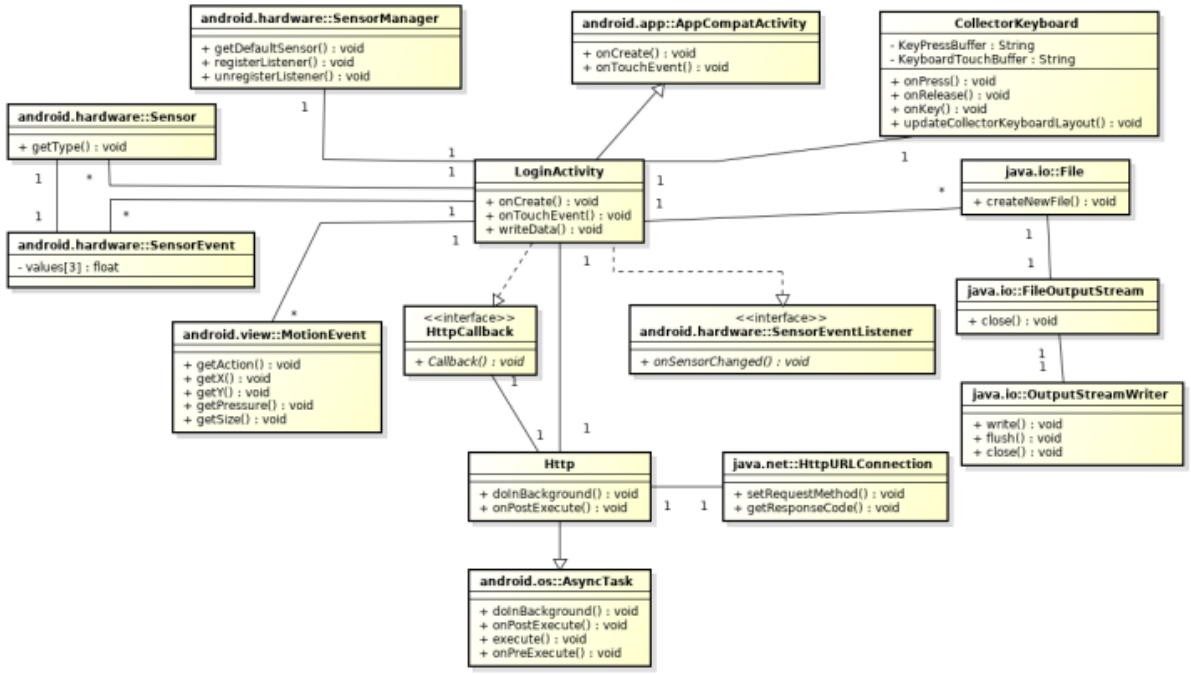
• Você é chamado para uma entrevista de emprego no Google. Como você se prepararia?

• Explique para sua avó como usar o smartphone para achar um novo restaurante no bairro.

As respostas das perguntas não são registradas.

A figura a seguir mostra a estrutura de classes do modo de autenticação:

Figura 3 - Diagrama de Classes (Autenticação)



Fonte: Autoria Própria

A classe *LoginActivity* é análoga a classe *CollectorActivity*, sendo a tela principal do modo de autenticação e capaz de capturar dados através de instâncias das classes *Sensor*, *SensorEvent*, *SensorManager*, *MotionEvent* e *CollectorKeyboard*.

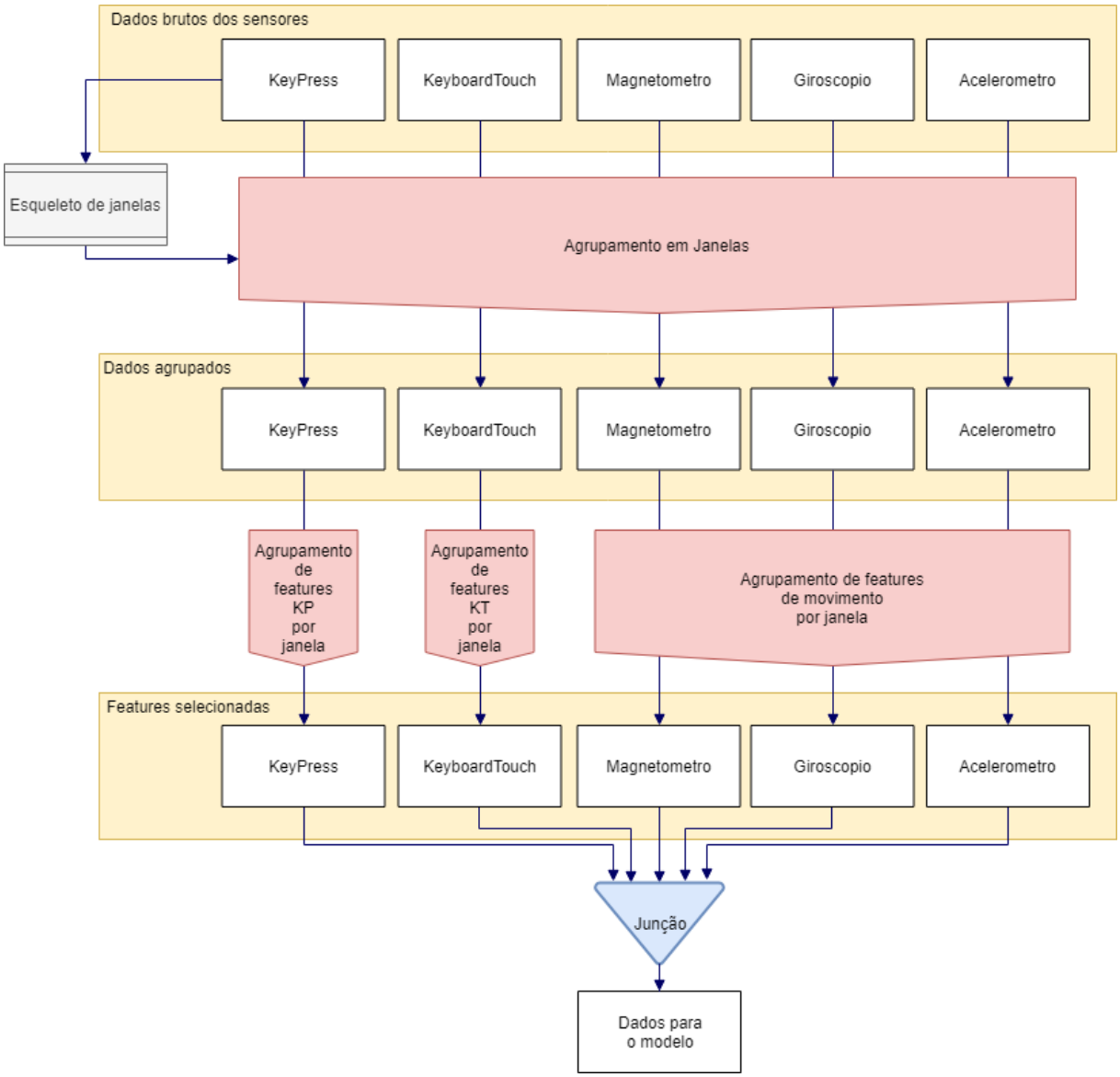
Nessa tela o usuário já selecionou qual o seu nome de usuário e deve responder uma das perguntas definidas no conjunto anterior. A principal diferença dessa tela é a sua capacidade de adicionar a classe Http para fazer uma requisição.

Essas requisições são feitas para o servidor enviando os dados coletados a partir dos sensores. O servidor então responde autenticando ou não o usuário baseado nos dados comportamentais da pessoa.

5.5.2 Especificação do Modelo.

Antes de utilizar o modelo é necessário tratar os dados recebidos pelos sensores. Esse tratamento segue o fluxo demonstrado no diagrama a seguir:

Figura 4 - Fluxo de Tratamento de Dados



Fonte: Autoria Própria

Os dados não tratados, capturados por cada um dos sensores, tem o seguinte formato:

Tabela 3 – Especificação dos dados do Acelerômetro (*Accelerometer*)

Nome do atributo	**Descrição**
Systime	*Timestamp* absoluto
ActivityID	Nome do usuário + Identificador de sessão
X	Aceleração menos a gravidade no eixo x
Y	Aceleração menos a gravidade no eixo y

Z	Aceleração menos a gravidade no eixo z
Phone_orientation	0: Modo retrato 1: Modo paisagem (rotacionado de 90 graus no sentido anti-horário) 3: Modo paisagem (rotacionado de 90 graus no sentido horário)

Tabela 4 – Especificação dos dados do Giroscópio (*Gyroscope*)

Nome do atributo	**Descrição**
Systime	*Timestamp* absoluto
ActivityID	Nome do usuário + Identificador de sessão
X	Velocidade angular em torno do eixo x
Y	Velocidade angular em torno do eixo y
Z	Velocidade angular em torno do eixo z
Phone_orientation	0: Modo retrato 1: Modo paisagem (rotacionado de 90 graus no sentido anti-horário) 3: Modo paisagem (rotacionado de 90 graus no sentido horário)

Tabela 5 – Especificação dos dados do Magnetômetro (*Magnetometer*)

Nome do atributo	**Descrição**
Systime	*Timestamp* absoluto
ActivityID	Nome do usuário + Identificador de sessão
X	Campo magnético ambiente no eixo x
Y	Campo magnético ambiente no eixo y
Z	Campo magnético ambiente no eixo z
Phone_orientation	0: Modo retrato 1: Modo paisagem (rotacionado de 90 graus no sentido anti-horário) 3: Modo paisagem (rotacionado de 90 graus no sentido horário)

Tabela 6 – Especificação dos dados de toque (*KeyPress*)

Nome do atributo	**Descrição**
Systime	Timestamp absoluto
ActivityID	Nome do usuário + Identificador de sessão

PressType	0: Apertar a tecla 1: Desapertar a tecla
KeyID	Código da tecla pressionada (Ex.: "97" corresponde a tecla "a"
Phone_orientation	0: Modo retrato 1: Modo paisagem (rotacionado de 90 graus no sentido anti-horário) 3: Modo paisagem (rotacionado de 90 graus no sentido horário)

Tabela 7 – Especificação dos dados de toque (*KeyboardTouch*)

Nome do atributo	**Descrição**
Systime	Timestamp absoluto
ActivityID	Nome do usuário + Identificador de sessão
Pointer_count	1: Toque único 2: Toque múltiplo
PointerID	0: Toque único (ou primeiro toque em toques simultâneos) 1: Segundo toque em toques simultâneos
ActionID	0 ou 5: Apertar a tecla 1 ou 6: Desapertar a tecla 2: Deslizar o dedo
X	Coordenada X do toque
Y	Coordenada Y do toque
Pressure	Pressão do toque
Contact_size	Área de contato do toque
Phone_orientation	0: Modo retrato 1: Modo paisagem (rotacionado de 90 graus no sentido anti-horário) 3: Modo paisagem (rotacionado de 90 graus no sentido horário)

O primeiro passo do fluxo é utilizar os dados de *KeyPress* para criar o "esqueleto de janelas". Esse arquivo define qual o *timestamp* de início da primeira janela que será considerado de todos os sensores. A partir desse *timestamp* de início são definidos os tempos de início e fim de todas as janelas subsequentes dessa sessão.

Tabela 8 – Esqueleto das Janelas

Nome do atributo	**Descrição**
ActivityID	Nome do usuário + Identificador de sessão

WindowNumber	Número da janela (n)
WindowStart	Timestamp de inicio da janela n
WindowEnd	Timestamp de fim da janela n

Os dados brutos então recebem as novas colunas do esqueleto através do processo de agrupamento em janelas. Os dados que não possuem um *WindowNumber* são descartados (um exemplo de dado descartado são os dados do acelerômetro antes do pressionar a primeira tecla).

Em seguida os dados são selecionados e agrupados de acordo com as características que utilizamos no modelo. Para os dados dos sensores de movimento (acelerômetro, giroscópio e magnetômetro), calculam-se os valores:

Tabela 9 – Features dos Sensores de Movimento

Nome do atributo	**Descrição**
ActivityID	Nome do usuário + identificador de sessão
WindowNumber	Número da janela (n)
X_mean	Média do sensor de movimento no eixo X durante a janela n
X_std	Desvio padrão do sensor de movimento no eixo X durante a janela n
Y_mean	Média do sensor de movimento no eixo Y durante a janela n
Y_std	Variância do sensor de movimento no eixo Y durante a janela n
Z_mean	Média do sensor de movimento no eixo Z durante a janela n
Z_std	Variância do sensor de movimento no eixo Z durante a janela n

Para os dados de *KeyPress* calculam-se os valores:

Tabela 10 – Features do KeyPress

Nome do atributo	**Descrição**
ActivityID	Nome do usuário + identificador de sessão
WindowNumber	Número da janela (n)
Press_count	Número de toques em teclas durante a janela n

Deltas_0_mean	Média dos tempos em que uma tecla fica pressionada durante a janela n
Deltas_0_median	Mediana dos tempos em que uma tecla fica pressionada durante a janela n
Deltas_0_std	Desvio padrão dos tempos em que uma tecla fica pressionada durante a janela n
Deltas_1_mean	Média dos tempos entre toques durante a janela n
Deltas_1_median	Mediana dos tempos entre toques durante a janela n
Deltas_1_std	Desvio padrão dos tempos entre toques durante a janela n

Para os dados de *KeyboardTouch* calculam-se os valores:

Tabela 11 – Features do KeyboardTouch

Nome do atributo	**Descrição**
ActivityID	Nome do usuário + identificador de sessão
WindowNumber	Número da janela (n)
Contact_size_mean	Média da área de contato de um toque durante a janela n
Contact_size_std	Desvio padrão da área de contato de um toque durante a janela n
Pressure_mean	Média da pressão de um toque durante a janela n
Pressure_std	Desvio padrão da pressão de um toque durante a janela n

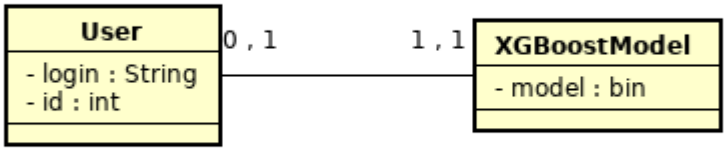
Após obter as features selecionadas agrupam-se todos os dados em uma tabela através do *ActivityID* e *WindowNumber*. A tabela obtida contém os dados de entrada do nosso modelo.

O modelo de aprendizado utilizado foi o algoritmo de XGBoost com os parâmetros: *n_estimators=90, max_depth=9, random_state=31, colsample_bytree=0.6, colsample_bylevel=0.5, learning_rate=0.11 e subsample=0.9.* Preferiu-se o uso de árvores de decisão pela sua simplicidade de implementação [20] e similar desempenho [21] quando comparada com redes neurais na tarefa de classificação de conjuntos de dados desbalanceados. A escolha entre a abordagem de Random Forest e XGBoost deu-se pela sua capacidade de lidar com dados nulos dentro do conjunto de dados [22].

5.5.3 Especificação do Servidor

Para não sobrecarregar o aplicativo com o processamento de dados e simplificar utilização do modelo treinado em 5.5.2 foi projetado um servidor simples capaz de receber os dados do aplicativo em modo de autenticação e retornar uma probabilidade do usuário ser autêntico. Cada usuário cadastrado no sistema pode ter um modelo de autenticação associado a ele, conforme a figura a seguir:

Figura 5 - Modelo Entidade Relacionamento



Fonte: Autoria Própria

Os *endpoints* necessários para o fluxo de autenticação foram:

Tabela 12 – Endpoint single_request_login

Título	**single_request_login** Realiza a autenticação do usuário
URL	/pignus/single_request_login
Método	POST
Parâmetros na URL	--

Parâmetros de dados	{
	Accelerometer: <String>,
	Gyroscope: <String>,
	Magnetometer: <String>,
	KeyPress: <String>,
	KeyboardTouch: <String>,
	Login: <String>
	}

Resposta de sucesso	Código: 200 Conteúdo: {

	auth: <Float> }

Resposta de erro	Código: 404
	Conteúdo:
	{
	error: "User with login not found"
	}
	Código: 401
	Conteúdo:
	{
	error: "Login not unauthorized"
	}
	Código: 500
	Conteúdo:
	{
	error: "Something went wrong. Please try again"
	}

Tabela 13 – Enpoint get_users

Título	**get_users** Mostra uma lista com os nomes de todos os usuários cadastrados
URL	/pignus/users
Método	GET
Parâmetros na URL	--
Parâmetros de dados	--
Resposta de sucesso	Código: 200 Conteúdo: [<String>,]

Resposta de erro	Código: 404
	Conteúdo:
	{
	error: "No users registered"
	}
	Código: 500
	Conteúdo:
	{
	error: "Something went wrong. Please try again"
	}

Durante a chamada do *single_request_login* os dados de cada um dos sensores devem ser tratados conforme ao modelo definido 5.5.2.

6 IMPLEMENTAÇÃO

6.1 Desenvolvimento do Aplicativo de Coleta de Dados

O aplicativo foi gradualmente desenvolvido por meio de sucessivas fases de pesquisa e implementação de pequenas funções isoladas, estas funções foram posteriormente integradas para garantir a funcionalidade especificada no item 5.5.1.

O desenvolvimento começou pela implementação da funcionalidade de criar e salvar novos arquivos no cartão de memória do dispositivo móvel. Vale ressaltar que para executar essa função é necessário conceder ao aplicativo permissão de gravação à memória do mesmo.

O próximo passo foi realizar a coleta dos dados dos sensores previamente selecionados na fase de estudo do dataset. Os dados do acelerômetro, magnetômetro e giroscópio foram facilmente obtidos, enquanto que os toques no teclado apresentaram certa dificuldade: o teclado nativo do *Android* consumia os eventos de toque na tela durante seu processamento natural, impedindo a captação de dados por meio das funções *OnTouch* que realizavam as leituras dos sensores de interesse.

Para contornar este problema, a primeira abordagem adotada foi tentar utilizar uma camada transparente que cobrisse a tela inteira do dispositivo. Embora esta camada permitisse a captura dos dados, não foi encontrada uma maneira fácil de passar adiante os eventos que eram recebidos por ela, inutilizando todos os demais componentes da tela.

A estratégia adotada então foi implementar um teclado personalizado (*CollectorKeyboard*). Utilizando esta abordagem o aplicativo passou a obter os valores dos sensores de interesse e posteriormente escrever os caracteres correspondentes nas caixas de texto associadas a ele.

Uma vez com todos os dados disponíveis, foi realizada uma integração dos dois módulos já implementados, passando a gravar os dados dos sensores formatados de maneira adequada em arquivos com extensão *csv*.

Por fim foi incluída a seleção aleatória de três das perguntas citadas no item 5.5.1 que passaram a ser exibidas para incentivar os usuários a escreverem durante a coleta de dados. Vale ressaltar que uma pequena correção foi necessária para evitar que estas perguntas fossem aleatorizadas sempre que o smartphone mudasse sua orientação de retrato para paisagem ou vice e versa.

6.2 Coleta de Dados

Utilizando o aplicativo apresentado no item 5.5.1, realizaram-se sessões de coleta de dados com os membros da equipe. Cada sessão durou em média seis minutos durante os quais os entrevistados responderam três perguntas aleatórias. Os dados coletados foram salvos em arquivos de extensão *csv* dentro de pastas homônimas aos entrevistados no próprio cartão de memória do *smartphone*.

Posteriormente, o mesmo procedimento foi aplicado sobre um pequeno grupo de entrevistados não pertencentes à equipe (mediante assinatura de um termo de consentimento livre e esclarecido presente no anexo A deste documento).

6.3 Compilação e Tratamento dos Dados

Os dados obtidos de ambas as fontes, *dataset* [11] e coleta, estavam segregados por sessão e por sensor em arquivos diferentes. Para realizar as manipulações, análises e por fim o treino do modelo de aprendizado, estes arquivos precisavam ser consolidados em um *csv* único. Um pequeno script foi desenvolvido em Python e aplicado sobre os dados para realizar uma agregação de todas as sessões e todos os usuários em diversos arquivos ainda segregados por sensor.

Uma vez que nem todos os sensores possuem uma taxa de atualização fixa, como por exemplo, o sensor de toque, não seria possível agrupá-los utilizando como chave o tempo absoluto do dado coletado. Em vez disso, optou-se por agregar os dados em pequenos intervalos sucessivos de 10 segundos cada, que serão referenciadas como janelas. As janelas são identificadas unicamente por seu número de sessão e número de janela.

Cada uma destas janelas possui valores de médias, variâncias e outras agregações relevantes dos diversos sensores coletados, conforme indicado abaixo:

- Acelerômetro
    - Média da aceleração no eixo X;
    - Média da aceleração no eixo Y;
    - Média da aceleração no eixo Z;
    - Desvio Padrão da aceleração no eixo X;
    - Desvio Padrão da aceleração no eixo Y;

    - Desvio Padrão da aceleração no eixo Z;
- Giroscópio
    - Média da velocidade no eixo X;
    - Média da velocidade no eixo Y;
    - Média da velocidade no eixo Z;
    - Desvio Padrão da velocidade no eixo X;
    - Desvio Padrão da velocidade no eixo Y;
    - Desvio Padrão da velocidade no eixo Z;
- Magnetômetro
    - Média do campo magnético no eixo X;
    - Média do campo magnético no eixo Y;
    - Média do campo magnético no eixo Z;
    - Desvio Padrão do campo magnético no eixo X;
    - Desvio Padrão do campo magnético no eixo Y;
    - Desvio Padrão do campo magnético no eixo Z;
- KeyboardTouch
    - Média da área de contato com a tela;
    - Desvio Padrão da área de contato coma tela;
- KeyPress
    - Média do intervalo de tempo entre pressionar e soltar um botão;
    - Média do intervalo de tempo entre soltar um botão e pressionar outro;
    - Média do número de botões pressionados;

É importante ressaltar que a primeira janela de cada sessão é criada com base no instante do primeiro toque do usuário, descartando quaisquer dados espúrios coletados antes de o usuário começar efetivamente a digitar.

Uma estratégia alternativa à divisão em sucessivas janelas seria consolidar os dados de toda a sessão, mas isto reduziria muito o volume de dados de treino disponível e também diminuiria drasticamente a resolução obtida em relação à estratégia anterior.

6.4 Treino do Modelo

Treinou-se um modelo por pessoa para permitir a flexibilidade de inserir novos usuários ao protótipo sem alterar os modelos já validados por outros usuários. O primeiro passo foi treinar modelos para os usuários provenientes do conjunto de dados provido pela referência [11]. Para ver a validação preliminar do modelo, carregavam-se os dados e definia-se aleatoriamente um usuário para ser o alvo do modelo, atribuindo a suas entradas *target* 1. Então, dividiu-se o conjunto de dados em uma parte para treino e uma parte de validação. A validação era composta por uma sessão do usuário alvo (com *target* 1) e outras 10 sessões compostas por outros usuários. Observou-se que o modelo já conseguia separar bem uma sessão das outras. Cada sessão é composta por diversas janelas e a previsão de uma sessão específica é definida pela média das janelas

Em seguida foi introduzido ao conjunto de treino os nossos dados. O modelo conseguia separar muito bem os dados coletados com os dados vindo de [11], mas não estava conseguindo classificar com acurácia entre os usuários com os dados coletados para esse projeto. Havia também uma limitação para realizar uma validação cruzada com muitos dados. A validação foi composta de 1 sessão do usuário alvo, 1 sessão de outro usuário que foi alvo de outro modelo, 2 sessões de usuários que tiveram os dados coletados para esse projeto e 10 sessões de usuários com dados coletados por [11]. Diversas mudanças foram realizadas para se otimizar o modelo.

Estas foram:

- A alteração do período considerado para representar uma janela de 10 segundos melhorou o resultado da previsão do modelo.
- Aumentou-se a coleta de dados para os usuários alvos do modelo (no início desta etapa só haviam sido realizadas 5 sessões). Isso fez com que o modelo conseguisse prever melhor as sessões com os dados coletados.
- Retirada das *features* de tamanho de toque e pressão de toque pelo fato delas diferenciarem os dispositivos e não as pessoas. A retirada fez com o que o modelo

pudesse aproveitar melhor os dados provenientes de [11] para melhor classificar pessoas com base no seu comportamento ao invés do dispositivo.

6.5 Desenvolvimento do Servidor

O servidor foi desenvolvido de forma iterativa. Primeiramente foi implementada uma requisição simples de teste, capaz de retornar um *JSON* do formato {"auth": 0.5}, com o valor "0.5" simulado. Isso foi feito para certificar o funcionamento correto dos cabeçalhos e o formato da resposta.

Em seguida foi desenvolvida a requisição "*get_users*" especificada no item 5.5.3. Essa requisição é capaz de retornar a lista com o *login* de todos os usuários presentes no servidor. É importante notar que o servidor foi utilizado como apoio para facilitar o desenvolvimento do aplicativo e a experiência de teste. Em uma aplicação real, retornar o *login* dos usuários é uma má prática e deve ser evitada, pois ela facilita que ataques de força bruta sejam realizados em seus usuários.

Uma vez que o aplicativo tem a lista de usuários, é possível que ele especifique qual dos usuários está tentando se autenticar. De início começou-se implementando a requisição de *single_request_login* com o usuário fixo (*hardcoded*) e com um *GET*, pois essa requisição não alterava o estado do servidor, apenas passava os dados coletados pelo sensor pelo modelo e retornava a probabilidade do usuário ser quem ele diz ser. Desse modo foi possível isolar os erros de seleção no banco de dados dos erros de tratamento de dados e instanciação do modelo. Entretanto logo notou-se que o limite de caracteres possíveis dentro de uma *URL* é de 2048 caracteres, muito abaixo do necessário para transmitir os dados comportamentais, portanto os dados comportamentais começaram a ser passados através do *body* da requisição, ao invés de parâmetros *URL*.

Ao longo do desenvolvimento provou-se mais fácil gravar os dados obtidos através da requisição no servidor, sendo assim alterado o método da requisição para um *POST*.

Validado o fluxo de autenticação com o usuário fixo, adicionou-se o parâmetro de *login*. Desse modo o servidor busca em seu banco de dados qual o usuário correspondente ao *login*, carrega o modelo correspondente e confirma a identidade do usuário através dos dados comportamentais.

6.6 Desenvolvimento do Aplicativo Protótipo para testes

O aplicativo existente foi estendido para garantir a funcionalidade especificada no item 5.5.1. Inicialmente foi implementada a função que realizava uma requisição *HTTP* do tipo *GET* para o servidor, garantindo assim que a comunicação com o servidor funcionava corretamente. Esta requisição foi utilizada para recuperar do servidor uma lista de usuários que possuem modelos disponíveis, para que o *login* pudesse ser realizado.

Logo depois foi implementada outra requisição *HTTP*, agora do tipo *POST*, que permitia enviar dados para o servidor.

Utilizando a mesma estrutura de coleta de dados já apresentada no item 6.1, os dados dos sensores foram coletados. Então, a requisição *POST* já implementada, é utilizada para enviar estes dados para o servidor. Após análise dos dados através do modelo presente no servidor, o mesmo responde com a probabilidade de os dados enviados pertencerem a um determinado usuário e esta probabilidade é indicada na tela do dispositivo.

6.7 Realização dos testes

Para se validar o protótipo, realizou-se um teste que representasse o real caso de uso. Para isso, foram utilizados os voluntários alvos do modelo e outros voluntários nunca antes vistos pelos modelos (validação *out-of-sample*).

Em todos os casos foi explicado como o teste funcionava *a priori*, ressaltando o objetivo da coleta de dados e como o teclado e as telas funcionavam. Também foi sempre feita uma sessão exemplo para mostrar como funcionaria.

O grupo de teste consistiu da seguinte combinação:

Alvos do modelo/Número de modelos: 3
Novas pessoas: 5

Os alvos do protótipo realizaram 10 sessões cada e as novas pessoas realizaram entre 3 a 6 sessões. Essas sessões, consequentemente, podem ser agrupadas da seguinte forma:

Sessões do alvo do modelo (*target = 1*): 10
Sessões de pessoas conhecidas ao modelo (*target = 0*): 20

Sessões de pessoas nunca vistas pelo modelo (*target = 0)*: *15*

Vale ressaltar que esses dados foram gravados para que em casos futuros também possam servir como dados para um futuro re-treino.

Durante estas sessões também se percebeu a necessidade de um feedback ao usuário ao apertar o botão de *login* enquanto o servidor calcula e envia o resultado da análise.

Uma vez feitas as sessões, todos os dados foram agrupados em um conjunto de dados do mesmo formato do conjunto de dados de treino e foram utilizados como entrada dos modelos preditivos para obter-se as métricas de desempenho.

7 RESULTADOS

Aqui serão apresentadas as métricas referentes ao modelo e aos testes, com análises sobre a relevância delas e possíveis conclusões

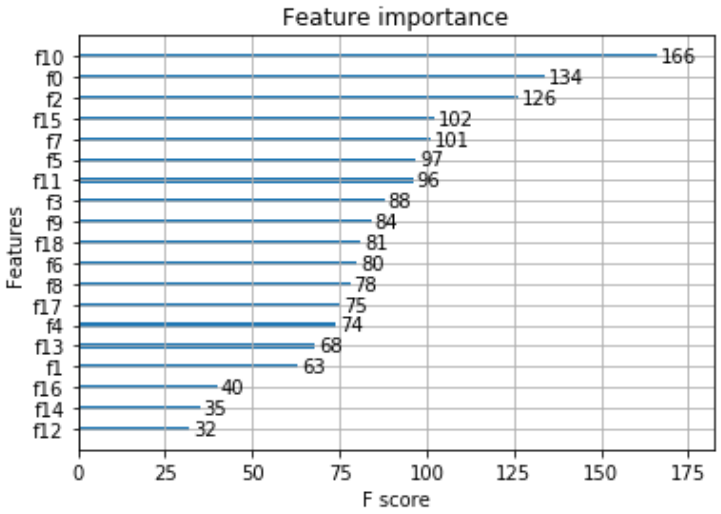
7.1 Importância das *features*

Os gráficos de importância das *features* pode ser visto a seguir:

Tabela 14 - Correspondência de *Features*

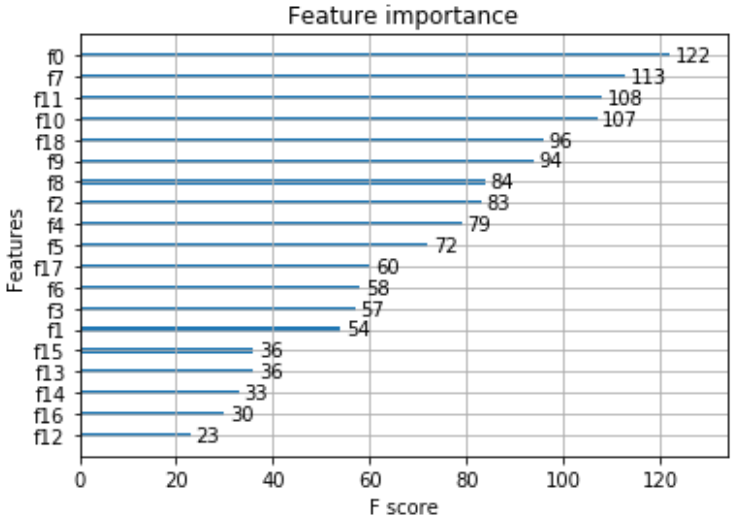
Legenda	*Feature*
F0	*Acc_X_mean*
F1	*Acc_X_std*
F2	*Acc_Y_mean*
F3	*Acc_Y_std*
F4	*Acc_Z_mean*
F5	*Acc_Z_std*
F6	*Deltas_0_mean*
F7	*Deltas_0_median*
F8	*Deltas_0_std*
F9	*Deltas_1_mean*
F10	*Deltas_1_median*
F11	*Deltas_1_std*
F12	*Gyr_X_mean*
F13	*Gyr_X_std*
F14	*Gyr_Y_mean*
F15	*Gyr_Y_std*
F16	*Gyr_Z_mean*
F17	*Gyr_Z_std*
F18	*Press_count*

Figura 6 - Importância de Features - Modelo 1



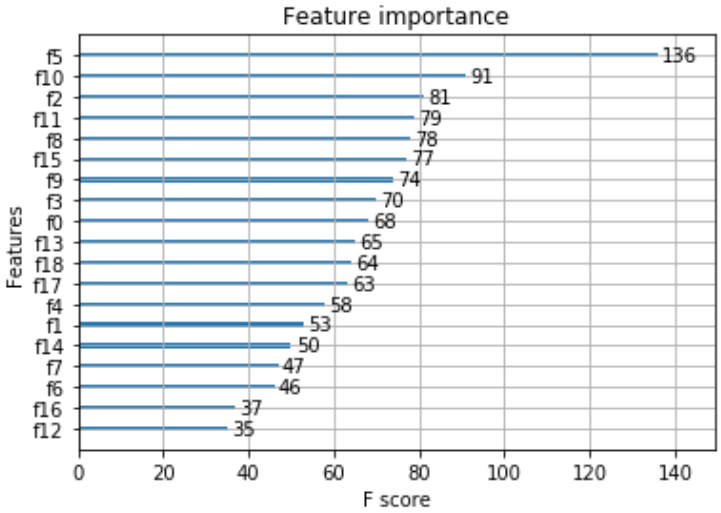
Fonte: Autoria Própria

Figura 7 - Importância de Features - Modelo 2



Fonte: Autoria Própria

Figura 8 - Importância de Features - Modelo 3



Fonte: Autoria Própria

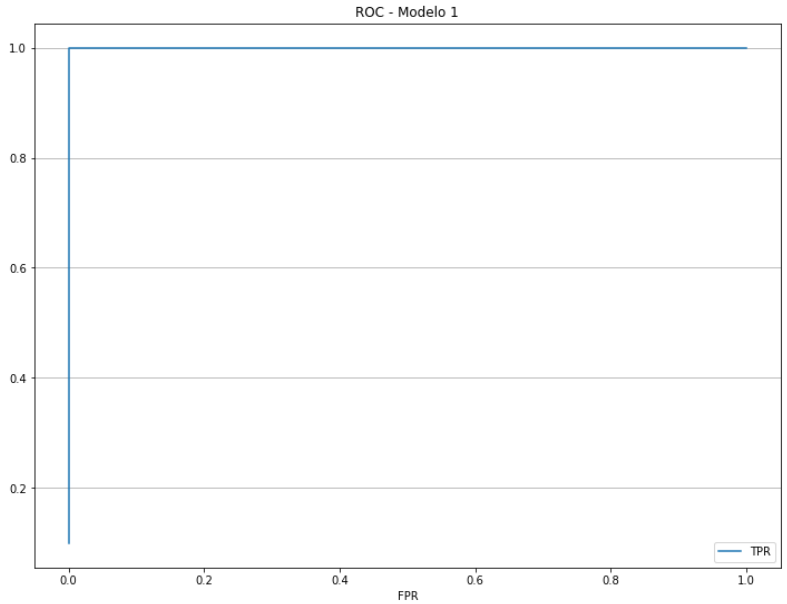
A importância das *features* é calculada automaticamente a partir de um método da classe *XGBoostClassifier* e ela é calculada a partir da importância de cada *feature* em cada árvore construída pelo modelo, levando em conta o quanto aquela *feature* contribui para o aumento da função de ganho. Então, todas essas importâncias são ponderadas para obter a importância agregada do modelo [16].

Pode-se ver que, em todos os modelos, as *features* relacionadas a duração do toque e/ou duração entre toque apareceram entre as 5 melhores *features*. Também é interessante observar que cada modelo também possui *features* importantes que não são tão bem ranqueadas em comparação com os outros modelos, o que mostra que certas características são bem específicas de cada pessoa.

7.2 Curva *ROC*

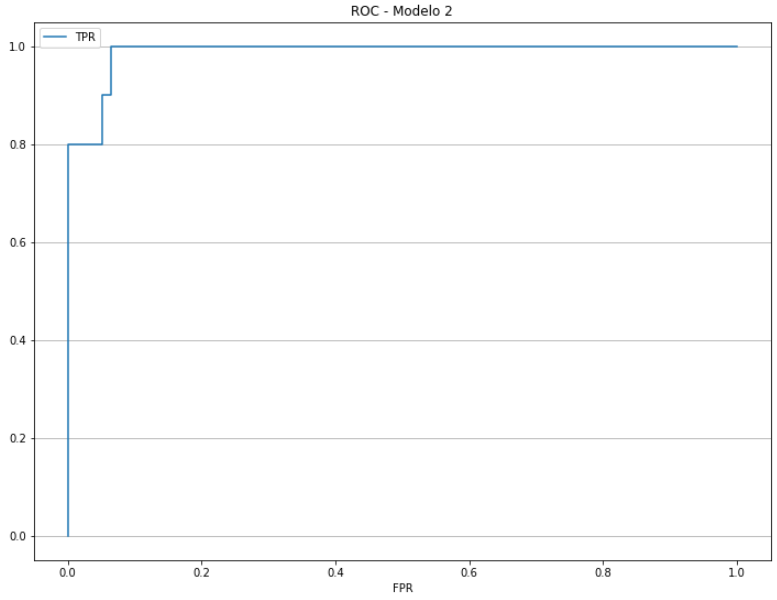
A curva *ROC* de cada um dos modelos pode ser observada a seguir:

Figura 9 - ROC das Sessões - Modelo 1



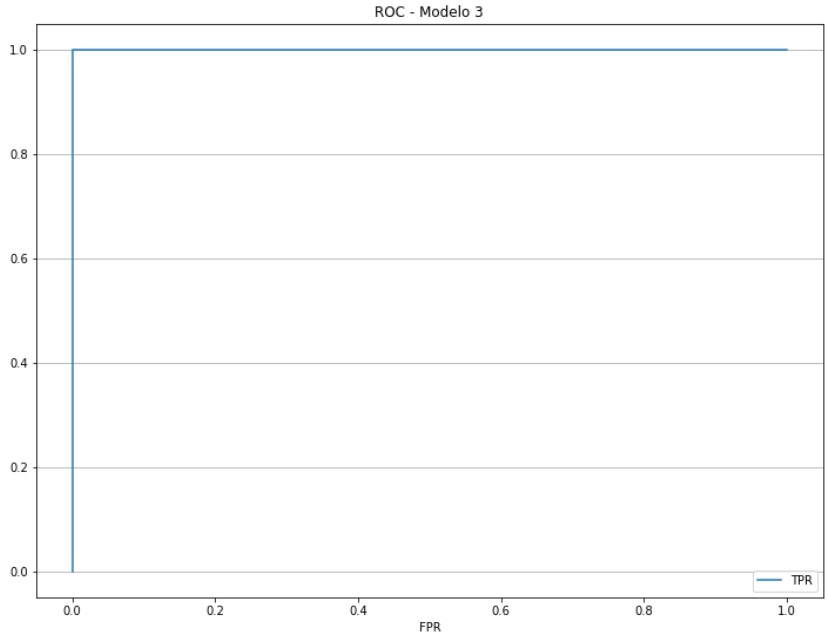
Fonte: Autoria Própria

Figura 10 - ROC das Sessões - Modelo 2



Fonte: Autoria Própria

Figura 11 - ROC das Sessões - Modelo 3



Fonte: Autoria Própria

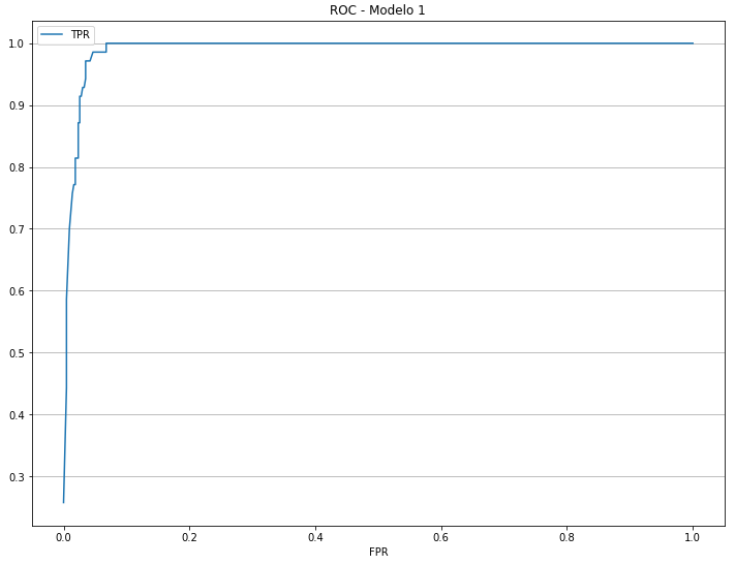
A aparência da curva dos modelos 1 e 3 pode ser explicada pelo fato de que existe um limiar no qual a taxa de falso positivos (FPR) é 0 ao mesmo tempo que a taxa de verdadeiros positivos é 1.0. Explicando-se de outra maneira, a maior previsão em um caso no qual o *target* era 0 ainda era menor do que a menor predição de um caso no qual o *target* era 1. Isso pode ser visto na tabela a seguir:

Tabela 15 - Previsões de limiar dos modelos

Modelo	Maior previsão com *target* 0	Maior previsão com *target* 1
1	0.907	0.982
2	0.971	0.867
3	0.833	0.878

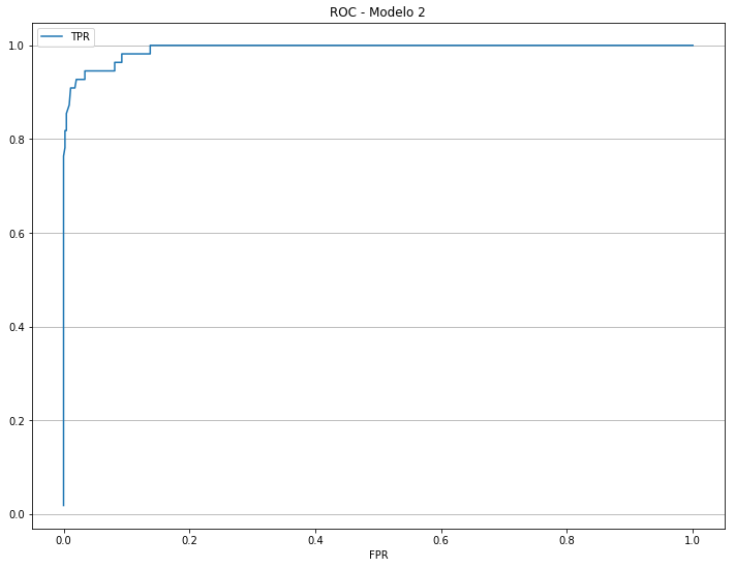
Essa situação se dá pelo fato de que estamos fazendo a média entre janelas para saber qual a predição de uma sessão, o que faz com que janelas erroneamente identificadas sejam compensadas pelas outras janelas da mesma sessão. Traçando-se a curva ROC para cada um dos modelos a partir da classificação das janelas, o resultado obtido pode ser observado nas figuras 12, 13 e 14:

Figura 12 - ROC das Janelas - Modelo 1



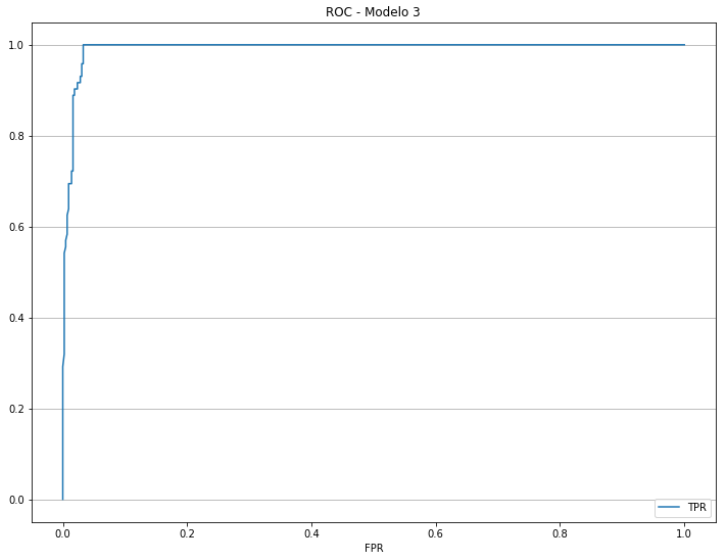
Fonte: Autoria Própria

Figura 13 - ROC das Janelas - Modelo 2



Fonte: Autoria Própria

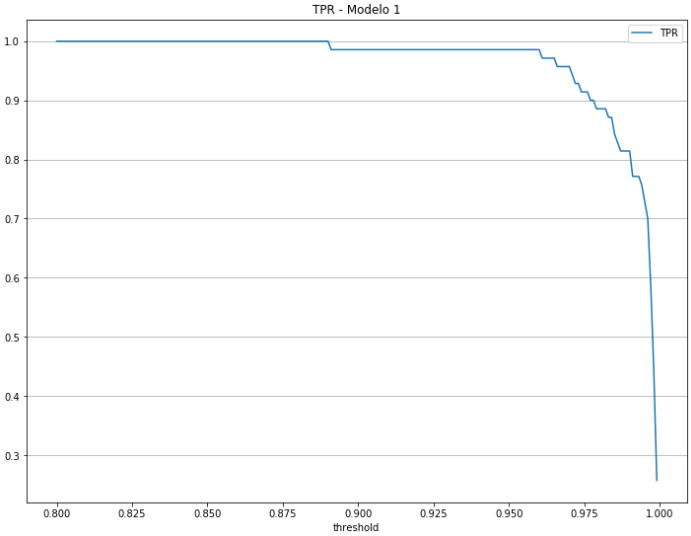
Figura 14 - ROC das Janelas - Modelo 3



Fonte: Autoria Própria

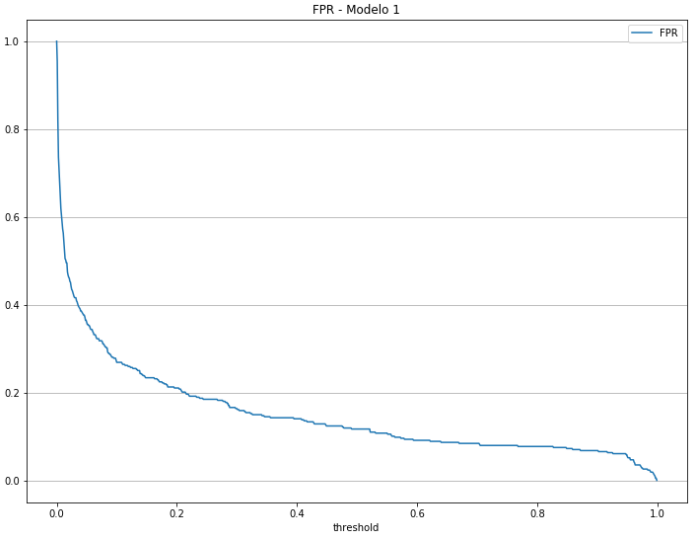
É possível observar melhor os efeitos isolados do limiar nas taxas de falso positivos e verdadeiros positivos nas sessões através dos gráficos presentes nas figuras 15 a 20:

Figura 15 - Taxa de Positivos Verdadeiros - Modelo 1



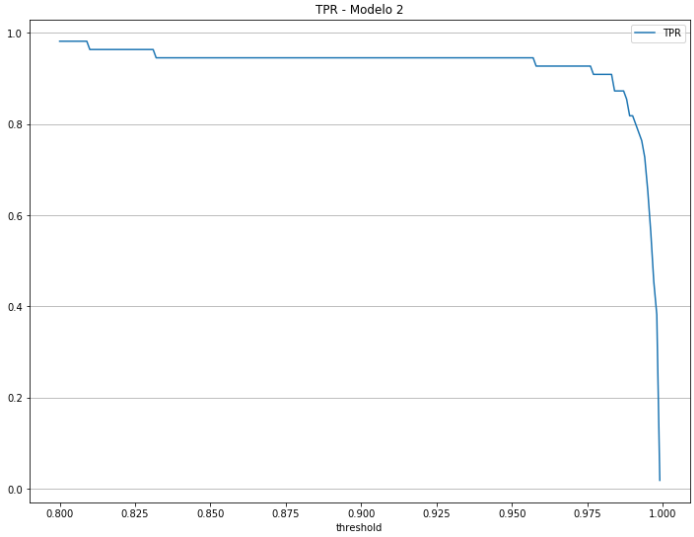
Fonte: Autoria Própria

Figura 16 - Taxa de Falsos Positivos - Modelo 1



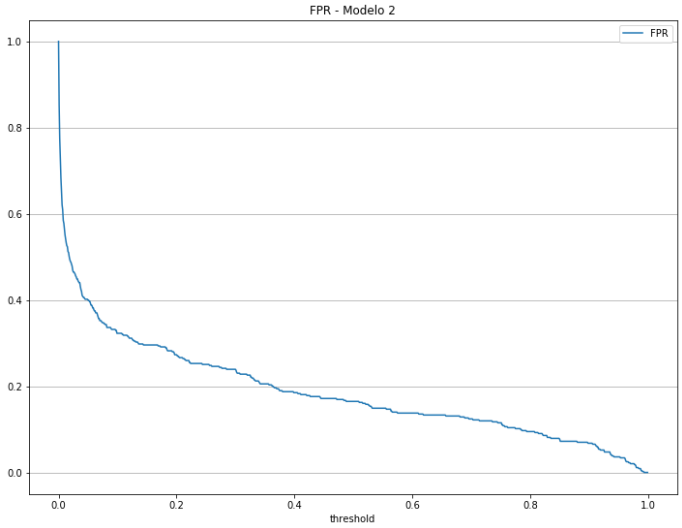
Fonte: Autoria Própria

Figura 17 - Taxa de Positivos Verdadeiros - Modelo 2



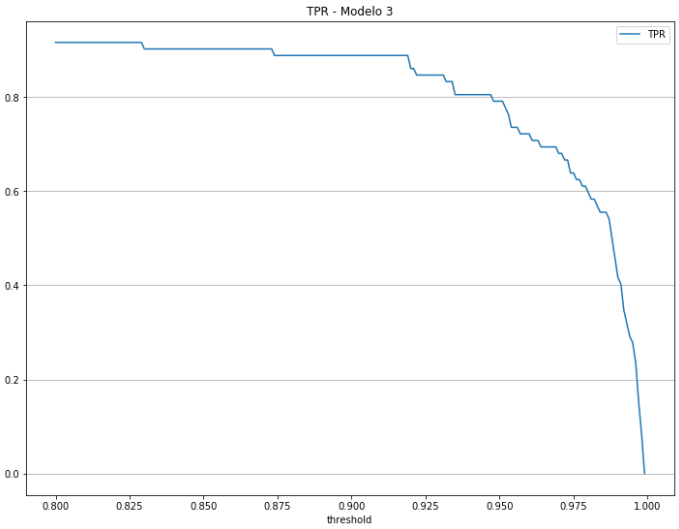
Fonte: Autoria Própria

Figura 18 - Taxa de Falsos Positivos - Modelo 2



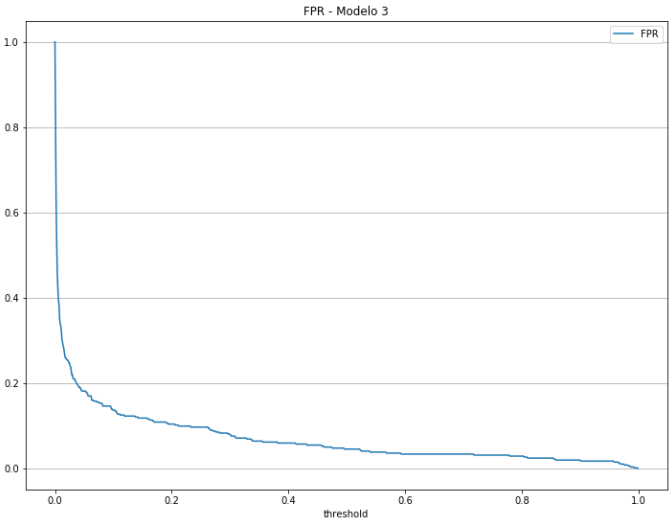
Fonte: Autoria Própria

Figura 19 - Taxa de Positivos Verdadeiros - Modelo 3



Fonte: Autoria Própria

Figura 20 - Taxa de Falsos Positivos - Modelo 3



Fonte: Autoria Própria

Pode-se perceber que a maior parte das sessões com alvo real possui uma previsão muito alta, só tendo quedas perceptivas na taxa a partir de limiares acima de 85%, o caso mais crítico pode ser observado na figura 19. Já na taxa de falsos positivos, a principal queda também acontece nos limiares mais baixos (menor que 10%), como visto na figura 20, mas existem sessões com previsões com diversos tipos de probabilidades, conforme figura 16 e 18. Isso se deve principalmente às sessões realizadas por usuários invisíveis ao modelo.

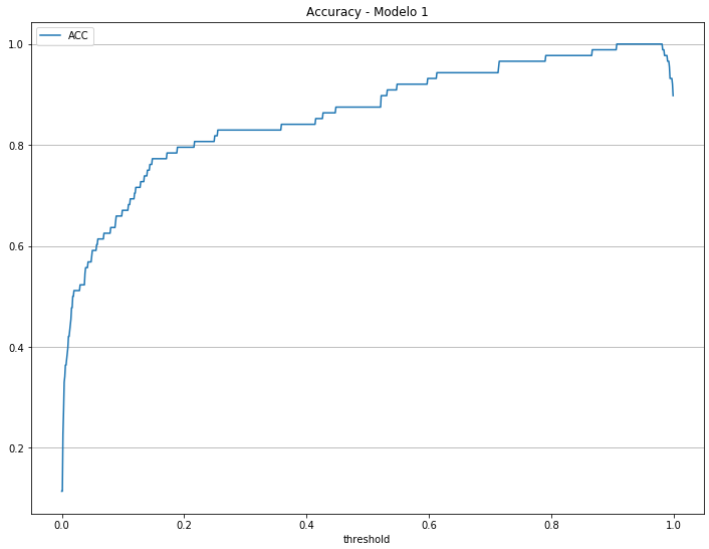
7.3 Acurácia

A acurácia obtida pelos três modelos 1, 2 e 3 treinados em função do *threshold* são apresentadas, respectivamente, pelas figuras 21, 22 e 23.

Para valores baixos de *threshold,* o modelo prevê muitos falsos positivos e poucos verdadeiros negativos, apresentando um valor baixo de acurácia como pode ser visto através do início dos gráficos 21, 22 e 23. Ao gradualmente elevar o *threshold*, o número de falsos positivos diminui e o de verdadeiros negativos aumenta, elevando assim a acurácia.

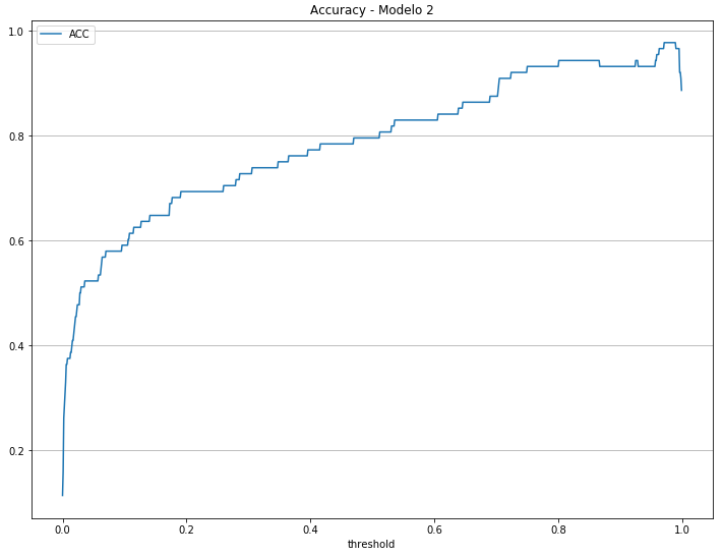
Entre 80% e 95% de *threshold* temos a máxima acurácia dos modelos. Elevar o *threshold* muito acima desse valor faz com que o modelo preveja muitos falsos negativos e poucos verdadeiros positivos, derrubando novamente a acurácia do modelo.

Figura 21 - Acurácia - Modelo 1



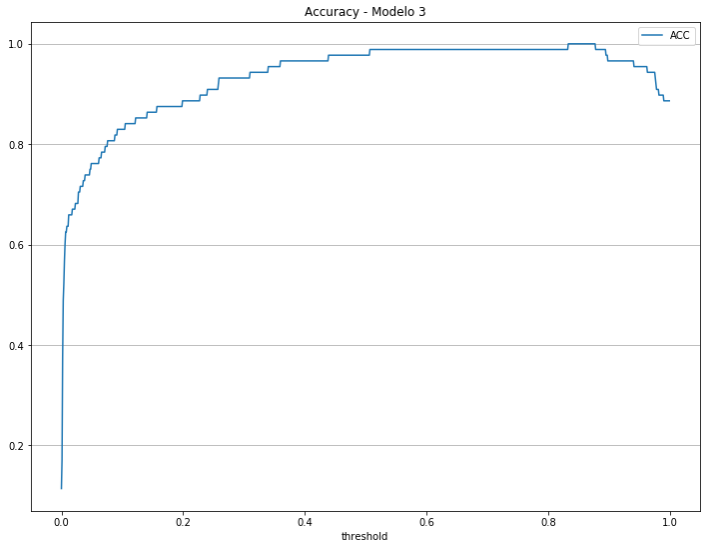
Fonte: Autoria Própria

Figura 22 - Acurácia - Modelo 2



Fonte: Autoria Própria

Figura 23 - Acurácia - Modelo 3



Fonte: Autoria Própria

7.4 F-Score

Como visto anteriormente, escolheu-se o *F-Score* com *beta = 0.5* para avaliar-se o modelo dada a característica dessa pontuação de valorizar modelos que necessitam falhar pouco. O valor de beta faz com que a taxa de falsos positivos seja mais relevante que a taxa de falsos negativos para a pontuação, já que, no caso desse projeto, o falso positiva significa um invasor sendo bem sucedido ao entrar em uma conta.

Calculou-se o F-Score para diversos limiares com passos de 0.1% para se encontrar o limar no qual a pontuação é máxima para cada modelo. O resultado obtido pode ser visto na tabela a seguir:

Tabela 16 - F-Score dos Modelos

Modelo	Limiar Ótimo	Precisão	*Recall*	**F0.5-Score**	F2-Score	F1-Score
1	0.916	1.0	1.0	**1.0**	1.0	1.0
2	0.973	1.0	0.8	**0.95**	0.83	0.89
3	0.833	1.0	1.0	**1.0**	1.0	1.0

8 CONCLUSÃO

Gradativamente, no decorrer do projeto, percebeu-se que o objetivo inicial de se criar um modelo para autenticação de pessoas a partir de dados de biometria comportamental necessitava de diversas outras etapas e componentes adicionais que, por si só, podem ser considerados pequenos projetos. Tendo isso em vista, essa conclusão também se dividirá da mesma forma.

8.1 Estudo de modelos e da literatura

Logo no início, uma das primeiras dificuldades foi se encontrar material na literatura referente à biometria comportamental e como relacioná-la com algoritmos de aprendizado de máquina de uma maneira mais prática.

O principal artigo de apoio, já que mais se assemelhava com o que era pretendido construir nesse projeto foi [11]. O seu estudo se provou benéfico ao longo de todo o projeto, principalmente nos seguintes fatores:

- Definição de quais dados deveriam ser coletados no aplicativo;
- A indicação de que era necessário construir um teclado próprio para se coletar dados referentes ao toque;
- Os dados coletados e publicamente disponibilizados que ajudaram tanto no treino do modelo tanto quanto na possibilidade de paralelização de esforços enquanto os dados próprios do projeto ainda não haviam sido coletados;
- A confirmação que dados do magnetômetro eram os menos relevantes e dados referentes à duração dos toques eram mais.

Reitera-se que mesmo com esses encaminhamentos, muitas coisas tiveram que ser desenvolvidas sem muito apoio da literatura como a implementação de um modelo de aprendizado de máquina em um servidor e o agrupamento de dados em janelas de tempo para facilitar o tratamento destes.

8.2 Aplicativo de coleta de dados

Idealmente, nessa etapa, os dados coletados viriam de uma experiência completamente fluida para os voluntários, não havendo mudanças perceptíveis entre as nossas

sessões de coletas e outros tipos de aplicativos nos quais há escrita de texto. Entretanto, por restrições do sistema operacional, foi necessário desenvolver um teclado próprio que, aos usuários, causava alguma estranheza nas primeiras sessões.

A grande preocupação dessa etapa era que os dados coletados não fossem consistentes ou que possuíssem alguma falha que fizesse com que o tempo gasto com os voluntários fosse desperdiçado. As práticas que ajudaram a prevenir esse problema foram:

- Padronização das instruções do que aconteceria na coleta de dados, mas ainda deixando o voluntário confortável o suficiente para escrever da maneira mais natural possível;
- Testes prévios para ver se os dados coletados estavam com ordens de grandeza e tamanhos consistentes com os esperados.

Além disso, pensando em uma aplicação comercial, um dos gargalos que foi observado nessa etapa seria a alta quantidade de dados que teriam que ser enviados por sessão de coleta (ao redor de 1MB por sessão).

8.3 Processamento de dados e treino do modelo

A validação cruzada foi essencial para verificar a eficácia do modelo e a ideia de se utilizar um modelo classificatório para cada usuário também se provou válida. O aumento da janela de 5 para 10 segundos diminuindo a variância entre as janelas e a retirada de *features* relacionadas a localização (Magnetômetro) e relacionadas diretamente ao dispositivo (tamanho do contato e pressão) se provaram úteis para a melhora na validação *out-of-sample*.

8.4 Realização do teste

A implementação do servidor em Django agilizou muito a velocidade para conseguirmos realizar os testes já que se utilizava a mesma linguagem de programação de implementação dos modelos offline.

Sobre o teste em si, as mesmas guardas utilizadas em 8.2. mais uma vez se provaram úteis evitando o retrabalho. Também tentou-se simular por completo uma situação de autenticação real para que os resultados do teste fossem consistentes com demonstrações ao vivo de funcionamento.

8.5 Resultados

Por fim, os resultados provaram a viabilidade da construção de um modelo de biometria comportamental com boas métricas, principalmente referentes a baixa taxa de falso positivos. Pode-se argumentar que a maneira como os dados foram coletados e os testes foram realizados diminui o valor prático que tal projeto pode possuir, mas uma breve reflexão faz perceber que estas são situações similares a aplicativos de troca de mensagens. Uma extrapolação seria coletar dados de quando um usuário digita a senha. A desvantagem é o curto período de tempo, com a clara vantagem que a autenticação é feita talvez no momento mais crítico do usuário.

Por fim, ainda há um grande espaço para inovação e outras *features* como o cálculo de ângulos e inclusive a inferência de com qual mão o usuário está segurando o dispositivo. Isso tudo, aliado a outras técnicas modernas de segurança, mostram como podemos caminhar para uma direção onde nossas informações estejam cada vez mais seguras.

Lista de Referências

[1] A. Hanamsagar, S. Woo, C. Kanich and J. Mirkovic. How Users Choose and Reuse Passwords, 2016.

[2] W. Weng. Mobile banking seen to overtake internet banking.

<http://www.theasianbanker.com/updates-and-articles/mobile-banking-seen-to-overtake-internet-banking>, 2017.

[3] D. Chaffey. Forecast growth in percentage of online retail / Ecommerce sales. <https://www.smartinsights.com/digital-marketing-strategy/online-retail-sales-growth/?>, 2017.

[4] M. Halpern, Y. Vijay J. Reddi. Mobile CPU's Rise to Power: Quantifying the Impact of Generational Mobile CPU Design Trends on Performance, Energy, and User Satisfaction, 2016.

[5] S. Ali, S. Khusro, A. Rauf and S. Mahfooz. Sensors and Mobile Phones: Evolution and State-of-the-Art, 2014.

[6] M. Feldman, 10 Real-World Examples of Machine Learning and AI, 2018.

[7] S. Gibbs. Dropbox hack leads to leaking of 68m user passwords on the internet <https://www.theguardian.com/technology/2016/aug/31/dropbox-hack-passwords-68m-data-breach>, 2016.

[8] M. Eiband, M. Khamis, E. Zezschwitz, H. Hussmann, F. Alt. Understanding Shoulder Surfing in the Wild: Stories from Users and Observers, 2017.

[9] M. J. McGrath, C. N. Scanaill. Sensing and Sensor Fundamentals, 2014.

[10] H. Saevanee, P. Bhatarakosol. User Authentication using Combination of Behavioral Biometrics over the Touchpad acting like Touch screen of Mobile Device, 2008.

[11] Z. Sitov, J. Sedenka, Q. Yang, G. Peng, G. Zhou, P. Gasti, K. Balagani. HMOG: New Behavioral Biometric Features for Continuous Authentication of Smartphone Users, 2016.

[12] N. Nilsson. Introduction to Machine Learning, 1998.

[13] S. Russel, P. Norvig. Inteligência Artificial 3ª edição, 2013.

[14] J. Si, A. Barto, W. Powell, D. Wunsch. Handbook of Learning and Approximate Dynamic Programming, 2004.

[15] D. Kuhlman. A Python Book: Beginning Python, Advanced Python, and Python Exercises, 2012.

[16] T. Hastie, R. Tibshirani, J. Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2001.

[17] B. Karlik, A Olgac. Performance Analysis of Various Activation Functions in

Generalized MLP Architectures of Neural Networks, 2011.

[18] A. Krizhevsky, I. Sutskever, G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks, 2012.

[19] D. Nielsen. Tree Boosting With XGBoost, 2016.

[20] M. Ahmad, M. Mourshed, Y. Rezgui. Trees vs Neurons: Comparison between random forest and ANN for high-resolution prediction of building energy consumption, 2016

[21] C. Hsieh, R. Lu, N. Lee, W. Chiu, M. Hsu, Y. Li. Novel solutions for an old disease: Diagnosis of acute appendicitis with random forest, support vector machines, and artificial neural networks, 2010.

[22] XGBOOST. XGBoost documentation. Disponível em

<https://xgboost.readthedocs.io/en/latest/index.html>. Acessado em 17 de Novembro de 2018.

[23] G. James, J. Bill, S. Guy, B. Gilad, B. Alex. *The Java® Language Specification*, 2014.

[24] DEVELOPERS ANDROID. Android Studio's User Guide. Disponível em

<https://developer.android.com/studio/intro/>. Acesso em 17 de Novembro de 2018.

[25] JUPITER. Project Jupyter Organization home page. Disponível em

<http://jupyter.org/index.html>. Acesso em 17 de Novembro de 2018.

[26] INRIA. Scikit-learn home page. Disponível em <https://scikit-learn.org/stable>. Acesso em 17 de Novembro.

[27] TENSORFLOW. TensorFlow home page. Disponível em

<https://www.tensorflow.org/>. Acesso em 18 de Novembro de 2018.

[28] KERAS. Keras Documentation. Disponível em <https://keras.io/>. Acesso em 18 de Novembro de 2018.

[29] DJANGO SOFTWARE FOUNDATION. Django Project home page. Disponível em <https://www.djangoproject.com/>. Acesso em 18 de Novembro de 2018.

[30] NUMFOCUS. Matplotlib Documentation. Disponível em <https://matplotlib.org/>. Acesso em 19 de Novembro de 2018.

[31] NUMFOCUS. Numpy home page. Disponível em <http://www.numpy.org/>. Acesso em 19 de Novembro de 2018.

[32] NUM FOCUS. Pandas home page. Disponível em <https://pandas.pydata.org/>. Acesso em 19 de Novembro de 2018.

[33] S. Shreve. Ngrok home page. Disponível em <https://ngrok.com/>. Acesso em 19 de Novembro de 2018.

[34] M. Sokolova, N. Japkowicz, S. Szpakowicz. Beyond Accuracy, F-score and ROC: a Family of Discriminant Measures for Performance Evaluation, 2006.

ANEXO A - TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Nós, Diego Amaral Souza, Jonathan Yuiti Suzuki e Thiago Cordeiro da Fonseca**,** estudantes de graduação de engenharia elétrica com ênfase em computação da POLI-USP, estamos realizando uma pesquisa intitulada de "Biometria comportamental pra identificação de usuários em plataforma móvel", sob a orientação de Anarosa Alves Franco Brandão, docente do curso de graduação de engenharia elétrica com ênfase em computação. Essa pesquisa tem como objetivo: desenvolver uma forma de autenticação mais segura com base na forma em que o usuário utiliza o seu dispositivo móvel, para tal será coletado dados de uso do dispositivo durante uma sessão de digitação.

Para tanto, gostaria que você participasse desta pesquisa, respeitando o seu direito de:

1- Ter liberdade de participar ou deixar de participar do estudo, sem que isso lhe traga algum prejuízo ou risco,
2- Manter o seu nome em sigilo absoluto, sendo que os dados coletados não lhe resultará em qualquer dano à sua integralidade,
3- Interromper a participação na pesquisa caso se sinta incomodado (a) com a mesma,
4- Garantia de receber uma resposta a alguma dúvida durante ou após a sessão.

O risco dessa pesquisa é mínimo, devido a anonimização de todos os dados coletados durante a pesquisa. Você não receberá qualquer pagamento pela sua participação, sendo ela voluntária. Você pode desistir da participação a qualquer momento do estudo sem penalidades

Esclareço-lhe ainda que o tempo estimado de sua participação será de aproximadamente 20 minutos, em que será solicitado que utilize um aparelho móvel disponibilizado pelos pesquisadores para responder 3 perguntas aleatórias dentro do aplicativo desenvolvido para a pesquisa.

A sua participação nesta pesquisa irá colaborar com o desenvolvimento da área de autenticação e segurança da informação, tópicos que estão cada vez mais presentes com a alta penetração que a tecnologia tem tido na vida das pessoas no século 21.

Este Termo de Consentimento será emitido em duas vias, sendo que uma via ficará em poder do pesquisador e a outra em poder do participante.

Deixamos telefone e email para contato do Jonathan Yuiti Suzuki: (41) 99243-2765 – jonathan.suzuki@usp.br, para que possa obter mais esclarecimentos ou informações sobre o estudo e sua participação.

Grato (a) pela atenção

_____ Assinatura do (a) pesquisador (a)

_____ Assinatura do (a) pesquisador (a)

_____ Assinatura do (a) pesquisador (a)

_____ Assinatura do (a) orientador (a)

Declaro que, após esclarecido pelos pesquisadores e ter entendido o que me foi explicado, consinto em participar do presente Projeto de Pesquisa.

São Paulo, ____/_____ /_____.

Nome do voluntário:

_____
Assinatura do voluntário