Antônio de Jesus Moraes Neto

Inferência de atividades físicas por meio de dispositivos móveis com processamento em nuvem

São Luís - MA

2019

Antônio de Jesus Moraes Neto

Inferência de atividades físicas por meio de dispositivos móveis com processamento em nuvem

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Universidade Federal do Maranhão - UFMA

Curso de Ciência da Computação

Orientador: Prof. Dr. Mário Antonio Meireles Teixeira

São Luís - MA

2019

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Moraes Neto, Antônio de Jesus.
Inferência de atividades físicas por meio de dispositivos móveis com processamento em nuvem / Antônio de Jesus Moraes Neto. – 2019.
44 f.

Orientador(a): Mário Antonio Meireles Teixeira.
Monografia (Graduação) – Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís – MA, 2019.

1. Atividades Físicas. 2. Computação em Nuvem. 3. Dispositivos Móveis. 4. Modelos Preditivos. 5. Weka. I. Teixeira, Mário Antonio Meireles. II. Título.

AGRADECIMENTOS

Acima de tudo quero agradecer a Deus, que esteve comigo desde o início. Apesar das distâncias ocasionais, Seu amor me proporcionou o caminho e a força necessária para seguir em frente.

Agradeço aos meus pais, Nizélia e José Mário, pelo amor incondicional e por todo o apoio ao longo de todos estes anos. Seus ensinamentos foram o alicerce do homem que sou hoje e devo todas as minhas conquistas a eles.

Também gostaria de agradecer aos meus irmãos, Vanessa, Magno, Luciana e Susana. Seu carinho sempre me envolveu de alegria. Aos meus familiares, meus avós Antônio, Francisca e Gabriela, meus tios Cláudio e Joana, minha prima Beatriz, quero agradecer pelo constante suporte e preocupação com meu futuro.

Agradeço também a todo o corpo docente da UFMA, em especial ao Prof. Mário, que além de ter sido um professor excelente, também me deu todo o suporte necessário para a elaboração deste trabalho.

Agradeço também a todos do Colégio Santo Expedito, afinal todo meu estudo pré-universidade foi junto a eles. Muito obrigado por todas as experiências proporcionadas.

Quero agradecer aos meus amigos Bruno, Márcio, Rosa, Carlos José, Pedro, Isabelle, Bianca, Francisco e Tárcio por todos os risos e apoio proporcionados. Aos amigos que conheci no curso, Alexsandro, Daniela, Polyana, Jéssica, Ramon, Júlia, Leonardo, Robherson, André e Gabriel Rezende, obrigado por suportarem as dificuldades do estudo junto a mim e também por tornarem a experiência mais divertida.

Agradeço também a meus amigos de Brasília, Gabriel, Ricardo, Tiago e João Pedro. Posso não vê-los pessoalmente, mas isso não os torna menos importantes. Obrigado por todo o suporte e anos de diversão.

Agradeço a todos que de alguma forma colaboraram para a realização desta tão sonhada (e temida) conquista.

"We all make choices, but in the end, our choices make us."
(KEN LEVINE ; BIOSHOCK, 2007)

RESUMO

Este trabalho propõe uma arquitetura baseada em nuvem com a finalidade de realizar inferências de atividades físicas. O sistema pode ser decomposto em um aplicativo para dispositivos móveis e um sistema alocado em uma nuvem. O objetivo de tal distribuição é reduzir a carga de processamento do dispositivo móvel ao máximo, pois dependendo do hardware, o mesmo pode não ser adequado para processamento intensivo de dados. Portanto o aplicativo é responsável por somente efetuar a coleta de dados sensores de movimento e enviá-los à nuvem. O sistema alocado na nuvem é responsável por todo o processamento e manipulação restante dos dados recebidos. Primeiramente os dados são pré-processados com o objetivo de simplificar e direcionar os cálculos que serão realizados nas etapas posteriores. Estes dados são então utilizados por um classificador da biblioteca Weka para treinamento do mesmo e para a realização posterior de inferências da atividade física sendo desempenhada pelo usuário do aplicativo. A eficácia do sistema foi avaliada ao final do projeto com o auxílio de voluntários e se demonstrou satisfatória.

Palavras-chave: Dispositivos Móveis, Computação em Nuvem, Modelos Preditivos, Atividades Físicas, Weka.

ABSTRACT

This work proposes a cloud-based architecture for the purpose of making inferences of physical activities. The system can be decomposed into a mobile application and a system allocated in a cloud. The purpose of such distribution is to reduce the processing load of the mobile device to the maximum, because depending on the hardware, it may not be suitable for intensive data processing. Therefore the application is responsible for only collecting motion sensor data and sending it to the cloud. The system allocated in the cloud is responsible for all processing and remaining handling of the received data. First, the data is preprocessed in order to simplify and direct the calculations that will be performed in later stages. These data are then used by a Weka library classifier for its training and for the subsequent realization of inferences of physical activity being performed by the application user. The effectiveness of the system was assessed at the end of the project with the help of volunteers and was considered satisfactory.

Key words: Mobile Devices, Cloud Computing, Predictive Models, Physical Activities, Weka.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface* (Interface de programação de aplicativos)
HTTP	*HyperText Transfer Protocol* (Protocolo de Transferência de Hipertexto)
IaaS	*Infrastructure as a Service* (Infraestrutura como Serviço)
PaaS	*Platform as a Service* (Plataforma como Serviço)
RMS	*Root Mean Square* (Raiz Quadrada da Média dos Quadrados)
SaaS	*Sofware as a Service* (*Software* como Serviço)
SGBD	*Data Base Management System* (Sistemas de Gestão de Base de Dados)
SQL	*Structured Query Language* (Linguagem de Consulta Estruturada)

SUMÁRIO

1 INTRODUÇÃO

A pesquisa em *Ambient Assisted Living* é um campo multidisciplinar emergente que tem como objetivo criar um ecossistema de diversos tipos de sensores, computadores, dispositivos móveis e aplicativos para o monitoramento de saúde pessoal (SUN et al., 2009). Antes, estas aplicações computacionais eram encontradas apenas em grandes centros hospitalares, mas os avanços mais recentes na tecnologia permitiram uma maior acessibilidade a estes sistemas, por meio de dispositivos mais compactos, baratos e com maior poder de processamento. Consequentemente, as metodologias de atendimento ao paciente vem passando por mudanças, possibilitando serviços individuais, personalizados e flexíveis. A partir deste contexto, este trabalho tem como um de seus focos realizar uma contribuição para a área, propondo uma arquitetura de sistema computacional que auxiliará profissionais da área médica no monitoramento e tratamento de seus pacientes.

Também é foco deste trabalho contribuir para a área da computação em nuvem. Este campo de estudo tem ganhado muita atenção nos últimos anos por fornecer serviços e recursos computacionais de fácil acesso e manipulação. Este modelo computacional é caracterizado pela escalabilidade, pelo provisionamento dinâmico de recurso sob demanda, pelo modelo de *utility computing*, onde a cobrança é feita de acordo com a quantidade de recursos usados e pela visão única do sistema, apesar de internamente toda a distribuição de recursos estar sob responsabilidade da empresa provedora do serviço (MELL; GRANCE et al., 2011). Levando-se em consideração estas características e o fato de que a arquitetura apresentada se trata de um sistema destinado a dispositivos móveis, é do interesse deste trabalho utilizar a computação em nuvem para uma maior eficiência de processamento de dados, já estes dispositivos podem muitas vezes ter baixo poder computacional.

1.1 Trabalhos Relacionados

Em (RIBEIRO FILHO, 2015) foi apresentado um sistema móvel voltado para o acompanhamento de pacientes portadores de doenças crônicas. O aplicativo tinha como objetivo reconhecer as atividades feitas pelo usuário, assim como também medir a intensidade em tempo real considerando diferentes níveis de mobilidade. O estudo realizado em (MORAES NETO, 2016) buscou dar continuidade a este trabalho, ao realizar experimentos com diversos algoritmos de classificação, com o objetivo de aprimorar as técnicas de predição de atividades físicas.

Em (FANG; LIANG; CHIU, 2012) foi apresentado um protótipo de sistema móvel com a capacidade de detectar quedas, voltado especialmente para idosos. O sistema proposto possuía três componentes: a captura de dados do acelerômetro embutido no dispositivo móvel,

o aprendizado do relacionamento entre o comportamento de queda e os dados coletados e um sistema de alerta para contatos pré-configurados. Em (YI et al., 2014) foi apresentado uma arquitetura de sistema que, além de detectar quedas, poderia realizar diagnósticos de saúde por meio de sensores adicionais ligados via *bluetooth* para realizar a coleta de dados fisiológicos, tais como temperatura corporal, posição geográfica, eletrocardiografia e postura corporal.

1.2 Objetivos

São objetivos deste trabalho:

1.2.1 Geral

Este trabalho tem como objetivo propor uma arquitetura baseada em nuvem que coleta dados de sensores de movimento com o fim de inferir qual atividade física o usuário está realizando.

1.2.2 Específicos

- Utilizar a plataforma Android para o desenvolvimento de sistemas
- Utilizar técnicas de processamento de dados de sensores de movimento
- Utilizar processamento em nuvem e implantar sistemas nesta arquitetura
- Utilizar a biblioteca Weka e seus algoritmos de aprendizado de máquina
- Desenvolver protótipo de sistema de inferência de atividades físicas
- Demonstrar efetividade da arquitetura proposta

1.3 Estrutura do Trabalho

Este trabalho está organizado em cinco capítulos com suas respectivas seções. A divisão procura esclarecer todos os conceitos utilizados ao longo do trabalho, bem como todas as etapas do processo de seu desenvolvimento. O capítulo 2 apresenta uma síntese de toda a teoria relacionada ao trabalho, bem como também introduz conceitualmente as tecnologias utilizadas.

O capítulo 3 apresenta toda a metodologia utilizada na arquitetura aqui proposta. São demonstradas todas as etapas de desenvolvimento, com amplo detalhamento na forma como as tecnologias descritas no capítulo 2 foram aplicadas.

O capítulo 4 apresenta uma metodologia para verificação da efetividade da arquitetura e seus resultados. Ao final, são realizadas considerações finais quanto a desempenho deste trabalho e como se deseja aprimorá-lo no futuro.

2 TECNOLOGIAS UTILIZADAS

Neste capítulo são introduzidos conceitos e tecnologias que são importantes para a construção e compreensão deste trabalho. Inicialmente são apresentadas técnicas para a captura de dados de sensores de movimento na plataforma Android, em seguida é discutido o funcionamento do classificador utilizado e os conceitos relacionados a este. Ao final são introduzidos alguns conceitos sobre computação em nuvem e a plataforma adotada para esta proposta.

2.1 Plataforma Android

O Android é uma pilha de software para dispositivos móveis que inclui um sistema operacional, um *middleware* e um conjunto de aplicações chave. A popularidade da plataforma se deve à dois fatores: a natureza de código aberto e seu modelo arquitetural.

Por ser um projeto de código aberto, é possível analisá-lo e compreendê-lo, o que permite o aperfeiçoamento da plataforma com novas funcionalidades e a adaptação para novo *hardware*, além da correção de erros.

O seu modelo arquitetural se baseia no *kernel* Linux, o que permite tomar vantagem de todo o conhecimento e recursos oferecidos pelo mesmo. Além disso, as aplicações Android são escritas na linguagem de programação Java e executadas em uma máquina virtual otimizada para a plataforma móvel (GANDHEWAR; SHEIKH, 2010).

2.1.1 Explicações para escolha de sensores

Esta proposta de arquitetura utiliza dois sensores de movimento para realizar a captura de dados do usuário: um acelerômetro e um giroscópio. Estes dois sensores estão disponíveis em uma ampla quantidade de dispositivos móveis Android e podem ser facilmente utilizados, como será visto na próxima seção.

O acelerômetro é um dispositivo utilizado para medir a aceleração própria de um objeto ou sistema. Este dispositivo é amplamente utilizado na indústria e na ciência, como por exemplo na detecção e monitoramento de vibrações em componentes estruturais ou na área da saúde pode ser usado para caracterizar a inclinação de membros de um paciente.

O giroscópio consiste essencialmente em um ou mais círculos articulados que se opõem a qualquer movimento que tente alterar suas orientações. Usando este princípio, o dispositivo é capaz de detectar e mensurar rotações ao longo dos eixos do sistema. Em *softwares* de realidade aumentada, o dispositivo pode auxiliar a manter objetos virtuais visualmente fixos no espaço físico do mundo real.

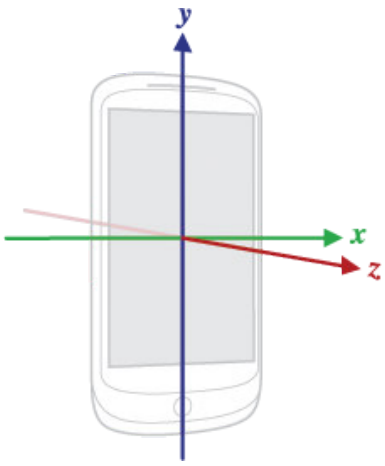
Como visto em (RIBEIRO FILHO, 2015), estes sensores podem ser utilizados para mensurar o movimento do usuário e a partir disto realizar inferências sobre a atividade física sendo desempenhada. Também foi realizada uma análise em (MORAES NETO, 2016), com o objetivo de avaliar o desempenho de diversos algoritmos de classificação quando estes tentam predizer a atividade física sendo realizada pelo usuário.

2.1.2 Sensores na plataforma Android

A maioria dos dispositivos Android possui sensores embutidos capazes de medir movimento, orientação espacial e várias outras condições de ambiente. Estes sensores fornecem dados não tratados com alta precisão e acurácia e são úteis para monitorar movimentação e posicionamento tridimensional e mudanças no ambiente próximo ao dispositivo.

A arquitetura apresentada neste trabalho utiliza dois sensores: acelerômetro e giroscópio. Estes sensores de movimento medem a força de aceleração e rotação ao longo dos três eixos cartesianos, como apresentado na Figura 1. (ANDROID DEVELOPERS, 2011)

Figura 1 – Eixos cartesianos no dispositivo móvel



Fonte: (ANDROID DEVELOPERS, 2011)

2.1.2.1 *Framework* de sensores Android

O acesso aos sensores presentes em um dispositivo móvel Android e a coleta de dados são realizados através do *framework* de sensores Android. Este *framework* tem como base as seguintes classes:

- SensorManager
- Sensor
- SensorEvent

- SensorEventListener

A classe "SensorManager" é utilizada para criar uma instância de um serviço de sensor. A mesma fornece métodos para acessar e listar sensores disponíveis no dispositivo móvel, permitindo também realizar e cancelar registros de *listeners* de sensores.

A classe "Sensor" é utilizada para criar instâncias de sensores e verificar suas capacidades. Estes sensores, ao identificar mudanças em seus registros, irão gerar objetos da classe "SensorEvent". Este objeto fornece informações sobre o sensor que sofreu mudanças e seus dados não tratados (no caso dos sensores utilizados nesta arquitetura, são gerados três valores referentes ao movimento nos eixos cartesianos).

A classe "SensorEventListener" se trata de uma interface que fornece métodos para receber notificações de novos objetos "SensorEvent", auxiliando na coleta de seus dados. (ANDROID DEVELOPERS, 2011)

2.2 Aprendizado de Máquina no Weka

Nesta seção são apresentados os principais conceitos referentes ao aprendizado de máquina e alguns exemplos de sua aplicação prática. Também é discutido o funcionamento da plataforma Weka, bem como sua biblioteca e o algoritmo de classificação utilizado pela arquitetura.

2.2.1 Definição de aprendizado de máquina

Segundo Alpaydin (2010), aprendizado de máquina é a programação de computadores para otimizar um critério de performance utilizando dados de exemplo ou experiência já existente. É criado um modelo definido por parâmetros, e o aprendizado é a execução de um programa de computador para otimizar estes parâmetros utilizando dados de treino.

O modelo pode ser preditivo para realizar predições no futuro, descritivo para ganhar conhecimento a partir dos dados, ou ambos. Este campo de estudo utiliza teorias da estatística para construir modelos matemáticos, já que a tarefa principal é realizar inferências a partir de uma amostra.

2.2.2 Utilidades

Alguns exemplos de aplicações práticas para aprendizado de máquina, segundo Alpaydin (2010), incluem:

- Uma rede de varejo pode utilizar aprendizado de máquina para realizar análise de compras

de clientes, dessa forma encontrando associações entre produtos e elaborando recomendações apropriadas para cada cliente;

- Um banco pode realizar análise de crédito de seus clientes utilizando dados como renda, economias, profissão, idade e histórico financeiro para classificá-los entre baixo risco e alto risco;

- Uma concessionária pode reunir dados que afetam o preço de um carro, como marca, ano, capacidade do motor e quilometragem para avaliar o valor de compra ou venda por meio de técnicas de regressão.

2.2.3 Plataforma Weka

Weka é um *software* destinado ao aprendizado de máquina que foi criada com o objetivo de auxiliar na aplicação de técnicas de aprendizado de máquina para uma variedade de problemas do mundo real. Diferente de outros projetos da mesma área, a ênfase é fornecer um ambiente funcional ao especialista do domínio em vez do especialista em aprendizado de máquina.

O *software* fornece um conjunto de ferramentas interativas para manipulação de dados, visualização de resultados e acesso a bancos de dados para complementar as ferramentas básicas de aprendizado de máquina.

2.2.3.1 Biblioteca do Weka

Além da interface interativa, a plataforma também fornece uma API que pode ser utilizada para automatizar todos os recursos do software. Entre os componentes mais importantes estão o mecanismo de instâncias e o classificador.

Instância é a abstração utilizada pelo Weka para representar cada caso a ser classificado ou utilizado para treinamento. Para a obtenção de um conjunto de instâncias, a plataforma suporta dois métodos de aquisição de dados: leitura de arquivo ou acesso a banco de dados. O primeiro método utiliza arquivos formatados como o formato “CSV” ou “ARFF”. O segundo utiliza um *driver* JDBC especificado para consumir uma tabela e gerar um *dataset*.

O processo de construção de um classificador é simples quando já se possui um *dataset* completo. Para isto, é criado um objeto da classe referente ao algoritmo de classificação que se deseja utilizar e então o *dataset* é passado ao classificador para leitura e treinamento.

A classificação de instâncias é realizada a partir da obtenção de um *dataset* não rotulado. A partir disto, o classificador irá ler cada uma das instâncias e calcular seu rótulo apropriado (de acordo com o conhecimento gerado pelo treinamento).

2.2.4 Algoritmo Random Forest

Em (MORAES NETO, 2016), foi realizado um experimento de diversos algoritmos no *software* Weka com o objetivo de avaliar o desempenho destes ao realizar classificações de atividades físicas. Os dados de caracterização de movimento dos usuários foram obtidos através de sensores de dispositivos móveis com poucas diferenças de metodologia em relação a arquitetura apresentada neste trabalho.

Ao fim da avaliação em (MORAES NETO, 2016), foi verificado que o algoritmo de classificação "Random Forest" obteve um grande destaque por apresentar uma ótima acurácia. Devido a este fator, e também o interesse de dar continuidade ao trabalho, este classificador foi adotado para a realização da mesma tarefa na arquitetura aqui proposta.

Segundo Ho (1995), o algoritmo "Random Forest" é um método de aprendizado conjunto para classificação e regressão que opera construindo múltiplas árvores de decisão no momento do treinamento e gerando a classe a partir da média das classes resultantes das árvores individuais.

2.2.4.1 Árvores de decisão

Como já explicado anteriormente, o algoritmo "Random Forest" funciona a partir da criação de múltiplas árvores de decisão. Antes de continuar a discussão sobre o algoritmo, é importante relatar os principais conceitos sobre a técnica de árvore de decisão para identificar suas qualidades e deficiências.

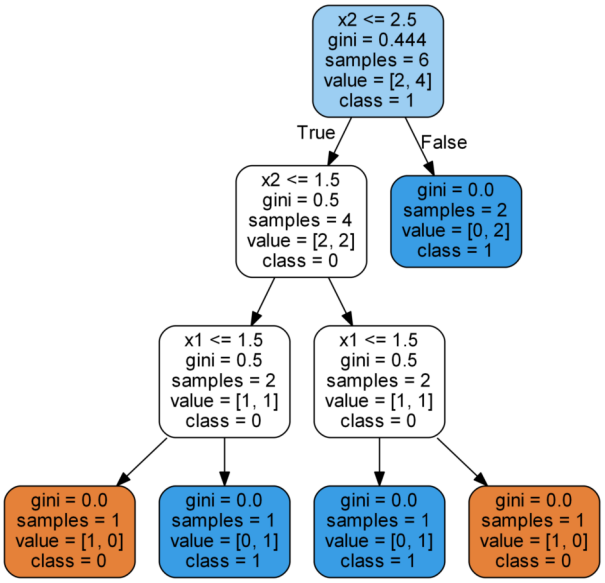
Árvore de decisão é um algoritmo de aprendizado supervisionado. Seu principal uso é criar um modelo de treinamento que pode ser utilizado para predizer uma classe ou valor a partir de um conjunto de variáveis por meio de criação de regras de decisão inferidas a partir de dados prévios (dados de treinamento). (QUINLAN, 1987)

Como pode ser visto na Figura 2, cada nó representa uma decisão a ser tomada. O processo de inferência da classe de cada registro consiste em avaliar o atributo que está em questão em cada um dos nós. No caso deste exemplo, o nó raiz questiona se o atributo "x2" é menor ou igual a "2.5". Caso a resposta seja positiva, o registro será passado para o nó filho da esquerda, caso contrário, para a direita. Este processo se repete até um nó folha ser atingido, o que determinará a classe do registro analisado.

O processo de criação da árvore envolve alguns conceitos estatísticos que serão discutidos posteriormente. Cada nó carrega informações adicionais: o número de amostras alocadas no nó, a quantidade de amostras de cada classe e também um rótulo de classe. Inicialmente toda as amostras serão alocadas no nó raiz, neste exemplo: seis amostras, sendo duas da classe "0" e quatro da classe "1".

A classe de cada nó é determinada pela classe que possuir o maior número de representantes. No caso do nó raiz, esta classe é "1". Após o preenchimento destas informações, é realizada uma decisão baseada em um atributo presente nas amostras. Semelhante ao processo de

Figura 2 – Exemplo de árvore de decisão



Fonte: (KOEHRSEN, 2018)

inferência, as amostras são avaliadas e movimentadas para um nó filho adequado. Este processo se repete até todos os nós possuírem representantes de somente uma classe ou ao atingirem um limite de profundidade estabelecido.

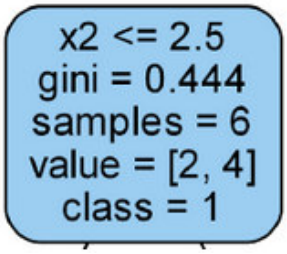
O processo de escolha do atributo a ser analisado em cada um dos nós é feito a partir de uma métrica chamada "Índice de Gini". Esta métrica calcula a probabilidade de que uma amostra escolhida aleatoriamente em um nó seja rotulada incorretamente se fosse rotulada pela distribuição de amostras no nó. (KOEHRSEN, 2018)

O cálculo do Índice de Gini em um nó utilizando um determinado atributo como critério de decisão é descrito na fórmula 2.1:

$$I_G = 1 - \sum_{i=1}^{J} (p_i)^2 \tag{2.1}$$

- A variável "J" é determinada pelo número de classes presentes nas amostras;
- A variável "i" se refere a uma das classes;
- "p_i" se trata de uma relação entre o número de amostras da classe "i" e o número total de amostras presentes no nó.

Figura 3 – Nó a ser avaliado pelo Índice de Gini



Fonte: (KOEHRSEN, 2018)

Aplicando a fórmula 2.1 no nó exibido na Figura 3, é obtido:

$I_G = 1 - (\frac{2}{6}^2 + \frac{4}{6}^2)$

$I_G = 0.444$

O objetivo ao utilizar o Índice de Gini é avaliar o resultado utilizando cada um dos atributos como um método de decisão no nó. O atributo que exibir o nó com o menor Índice de Gini será então utilizado. Este processo se repetirá a cada novo nó filho gerado, até que o índice se torne "0" (o que coincide com a característica do nó folha possuir amostras somente da mesma classe).

Ao utilizar esta técnica, a árvore de decisão se torna extremamente eficiente ao avaliar amostras de treinamento. Entretanto, a adaptação aos dados de treinamento se torna tão alta que a mesma perde a capacidade de generalização, ocorrendo *overfitting*. Um dos objetivos de um modelo de aprendizado de máquina é ser capaz de generalizar bem o suficiente para avaliar dados não previamente vistos, portanto uma árvore de decisão simples se torna ineficiente para usos práticos.

2.2.4.2 Funcionamento do algoritmo Random Forest

Como já introduzido anteriormente, o algoritmo "Random Forest" utiliza múltiplas árvores de decisão e realiza uma média das predições de classe destas árvores para inferir uma classe somente ao final. Esta configuração, além de outras técnicas de aleatoriedade, é utilizada para solucionar o problema de *overfitting* mencionado na seção anterior.

De acordo com Breiman (1999), este algoritmo utiliza duas técnicas de aleatoriedade ao longo de suas amostras de treinamento:

- Seleção aleatória de amostras
- Seleção aleatória de características

As árvores de decisão criadas pelo algoritmo não possuem acesso a toda a base de treinamento. Para cada árvore é fornecida somente uma parcela aleatória das amostras com

reposição, o que significa que uma mesma amostra pode ser selecionada múltiplas vezes para uma mesma árvore. Este conceito é conhecido como *bootstrapping*.

As árvores também não possuem acesso a todas as características das amostras. Cada árvore irá trabalhar com apenas um número limitado de características aleatórias e deverá realizar a seleção dos melhores atributos (de acordo o Índice de Gini) apenas com estas informações. O número de características a serem cedidas para cada árvore costuma ser a raiz quadrada do número de total.

2.3 Computação em Nuvem

Nesta seção são apresentados os principais conceitos referentes à computação em nuvem, como características, modelos de serviço, modelos de implantação e por fim é apresentado o provedor de nuvem utilizado nesta arquitetura.

2.3.1 Definição

Mell, Grance et al. (2011) definem a computação em nuvem como um modelo para permitir acesso de rede onipresente, conveniente e sob demanda a um conjunto compartilhado de recursos de computação configuráveis (por exemplo, redes, servidores, armazenamento, aplicativos e serviços) que podem ser rapidamente provisionados e liberados com esforço mínimo de gerenciamento ou interação com o provedor de serviços.

2.3.2 Características

De acordo com Mell, Grance et al. (2011), existem cinco características principais na computação em nuvem:

- Serviços sob demanda: Um consumidor pode requerer recursos computacionais, como tempo de servidor e armazenamento em rede, automaticamente quando necessário sem a necessidade de interação humana;
- Amplo acesso à rede: Os recursos oferecidos estão disponíveis na rede e podem ser acessados por qualquer dispositivo e mecanismo padrão;
- Agrupamento de recursos: Os recursos computacionais do provedor são distribuídos e agrupados de forma a servir múltiplos consumidores. Recursos físicos e virtuais são alocados dinamicamente de acordo com a demanda do consumidor. O usuário não possui conhecimento exato da localização dos recursos que está utilizando;
- Elasticidade rápida: A capacidade dos recursos pode ser aumentada ou diminuída elasticamente. Este dimensionamento ocorre de acordo com a demanda e não é necessária a intervenção do usuário;

- Serviço mensurável: Serviços de nuvem automaticamente controlam e otimizam o uso de recursos. O nível de uso de cada recurso pode ser monitorado, controlado e reportado, fornecendo transparência para o provedor e o consumidor.

2.3.3 Modelos de Serviço

Segundo Mell, Grance et al. (2011), existem três modelos básicos de serviços de computação em nuvem:

- *Software* como serviço (SaaS): O serviço fornecido ao usuário é composto pelo uso de aplicações em uma infraestrutura de nuvem. Esta aplicações estão acessíveis a partir de vários dispositivos e meios. O consumidor não possui controle algum sobre a infraestrutura ou o funcionamento interno das aplicações;

- Plataforma como serviço (PaaS): É fornecida ao usuário a capacidade de implantar na infraestrutura de nuvem suas próprias aplicações utilizando linguagem de programação, bibliotecas, serviços e ferramentas suportados pelo provedor. O usuário não possui controle sobre elementos da infraestrutura, como rede, servidores, sistemas operacionais ou armazenamento;

- Infraestrutura como serviço (IaaS): É fornecida ao usuário a capacidade de gerenciar processamento, armazenamento, rede e outros recursos computacionais. É possível implantar *softwares* livremente, como sistemas operacionais ou aplicações.

2.3.4 Modelos de Implantação

Em relação a forma de disponibilização dos serviços de nuvem, em (MELL; GRANCE et al., 2011) é descrita a existência de quatro modelos de implantação:

- Nuvem privada: A infraestrutura de nuvem é provisionada para uso exclusivo por uma única organização. Pode existir dentro ou fora das instalações, além de ser gerenciada pela organização ou por terceiros;

- Nuvem comunitária: A infraestrutura de nuvem é provisionada para uso exclusivo por um conjunto de organizações que têm interesses compartilhados. Pode existir dentro ou fora das instalações e é administrada por uma ou mais das organizações da comunidade ou por terceiros;

- Nuvem pública: A infraestrutura de nuvem é disponibilizada para uso público geral. Pode ser administrada por uma organização comercial, acadêmica ou governamental;

- Nuvem híbrida: A infraestrutura de nuvem é uma composição de dois ou mais modelos de implantação que permanecem como entidades distintas, mas unidas por tecnologia padronizada ou proprietária que permite a portabilidade de dados e aplicativos.

2.3.5 Provedores de Serviço

O provedor de serviço de nuvem é responsável por fornecer a plataforma, infraestrutura, aplicativos e serviços de armazenamento baseados em nuvem. Cada provedor possui um acervo distinto de recursos fornecidos, além de também disponibilizá-los utilizando diferentes padrões. Este trabalho utiliza a plataforma Heroku, que será apresentada a seguir.

2.3.5.1 Heroku

Heroku é uma plataforma de nuvem como serviço (PaaS) com suporte a diversas linguagens de programação. A plataforma fornece ao cliente um ambiente pronto para a implantação de aplicações com a necessidade de mínimas alterações devido a particularidades do serviço. Semelhantemente a outros serviços de nuvem PaaS, a implantação ocorre por meio de *commits* em repositórios remotos.

A plataforma Heroku usa o modelo de contêiner para executar e dimensionar todos os aplicativos nela implantados. Estes contêineres são abstrações criadas com o objetivo de minimizar o esforço do desenvolvedor, ao agrupar o código e suas dependências em um ambiente de execução isolado que fornece computação, memória, sistema operacional e um sistema de arquivos temporário.

Os contêineres usados na Heroku são chamados de “dynos”. “Dynos” são contêineres Linux isolados e virtualizados projetados para executar código com base em um comando especificado pelo usuário. Uma aplicação pode ser dimensionada para qualquer número especificado de “dynos” com base em sua demanda de recursos.

A plataforma fornece três tipos destes contêineres:

- *Web Dynos*: São “dynos” com a capacidade de receber e atender requisições *web*;
- *Worker Dynos*: São responsáveis por executar tarefas que demandam alto processamento;
- *One-Off Dynos*: São ambientes criados para realizar apenas uma tarefa e logo serem “destruídos”.

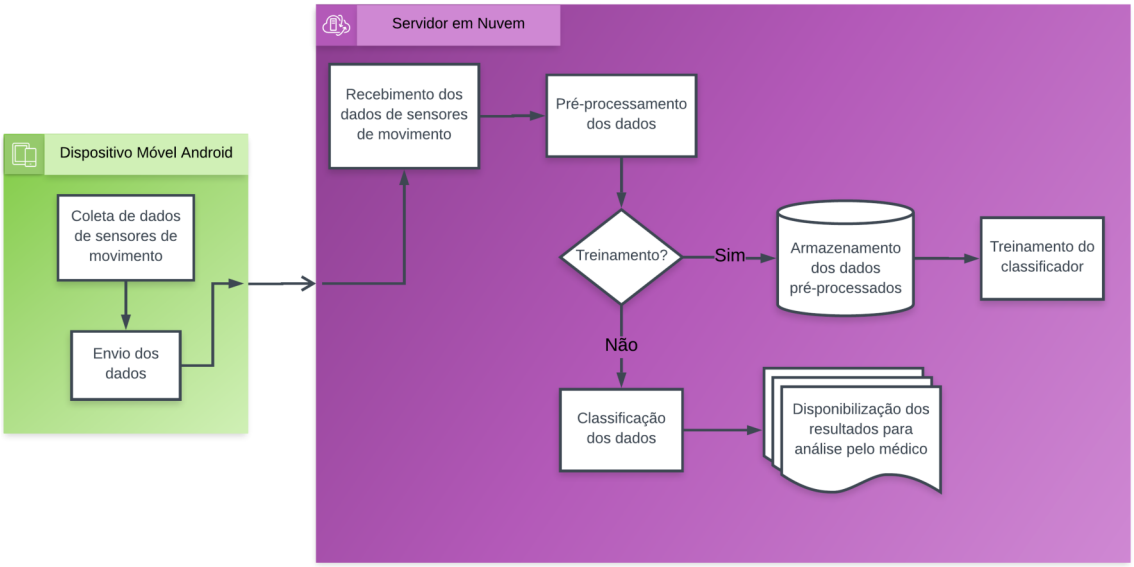
Uma aplicação comum geralmente utiliza apenas um “web dyno” para comunicação externa e tarefas de baixo nível de processamento e opcionalmente alguns “worker dynos” caso necessário. “One-off dynos” são comumente utilizados para tarefas de execução única como leitura de *logs*.

3 ARQUITETURA DO SISTEMA

Neste capítulo são apresentados os métodos desenvolvidos para a implementação da arquitetura proposta neste trabalho. Serão descritos todos os procedimentos realizados em cada etapa da metodologia, além de descrever cada camada da arquitetura. Como visto na Figura 4, o sistema é distribuído entre o dispositivo móvel e o servidor (nuvem). Sendo o dispositivo móvel responsável por realizar a coleta de dados de sensores de movimento e o servidor por realizar o processamento dos dados, assim como também treinar um classificador e utilizá-lo para realizar inferências a partir de novos dados.

O protótipo de sistema foi desenvolvido na linguagem de programação Java, além de também utilizar o ambiente de desenvolvimento Android. É importante destacar que este trabalho se restringe a trabalhar apenas com a plataforma Android, não descrevendo as alterações necessárias na arquitetura para o funcionamento com outros sistemas operacionais móveis.

Figura 4 – Arquitetura do sistema



Fonte: Autor.

3.1 Coleta e Gerenciamento de Dados

Esta primeira camada está alocada no aplicativo do dispositivo móvel e é responsável por realizar toda a coleta dos dados gerados pelos sensores de movimento. Além de também realizar o gerenciamento destes dados para o uso posterior na próxima camada.

3.1.1 Classe SensorData

O *framework* de sensores Android gera, como dados principais, valores dispostos nos eixos X, Y e Z. Para realizar o gerenciamento e armazenamento destes dados, foi criada a classe "SensorData", conforme o Quadro 1. As instâncias dessa classe tem por objetivo manter informações de cada evento gerado pelo *framework*. Essas informações consistem nos dados de aceleração ou velocidade angular para cada eixo e também o tipo de sensor que gerou o evento.

Quadro 1 – A classe SensorData

Atributo	**Valor**
tipo	Tipo de sensor utilizado
x	Valor gerado no eixo X
y	Valor gerado no eixo Y
z	Valor gerado no eixo Z

Fonte: Autor.

3.1.2 Método de coleta

A coleta de dados de sensores de movimento se inicia através da interação do usuário com o aplicativo. O processo de inicialização envolve a criação de uma contagem regressiva e também o registro de *listeners* para o acelerômetro e o giroscópio com a especificação de *delay* mínimo através do *framework* de sensores Android. Dessa forma, estes passam a registrar os dados sempre que for detectado movimentação no dispositivo.

Cada evento detectado pelos *listeners* será analisado para que os dados possam ser registrados. Na análise, é descoberto o tipo de sensor que o gerou o evento, bem como também os valores de aceleração ou velocidade angular. Estes dados são então armazenados em um objeto da classe "SensorData".

Os *listeners* permanecem ligados até o fim da contagem regressiva previamente estabelecida. Este método de encerramento foi escolhido, pois esperar pela interação do usuário para encerrar o processo requer a realização de movimentos não referentes à atividade física sendo desempenhada.

3.2 Envio de Dados

O módulo de envio dos dados é iniciado com o fim da contagem regressiva. Ao iniciar, é verificado qual tipo de requisição o usuário selecionou: treinamento ou inferência. Caso o usuário deseje realizar um treinamento, também é verificado qual atividade física foi informada no início da coleta.

3.2.1 Protocolo HTTPS

Toda a comunicação entre o aplicativo Android e o servidor em nuvem é realizada através do protocolo HTTPS. Este método foi escolhido por se tratar do padrão atual de comunicação *web* e por oferecer maior segurança e confiabilidade, como explicado em (CLOUDFLARE, 2018). Métodos alternativos como o *Socket* não possuem um padrão exato e são mais suscetíveis à ataques externos.

Também é importante ressaltar que a plataforma Heroku não fornece suporte para outros métodos de comunicação *web*. A plataforma utiliza uma camada de roteamento que recebe todas as requisições HTTP e as redireciona para o endereço ou sistema adequado. O uso desta camada adicional é obrigatório, já que os sistemas hospedados não possuem comunicação direta com a internet.

3.2.2 Leiaute da mensagem

Toda a formatação da mensagem HTTP é feita através da classe "EnvioHTTP". Esta classe recebe em seu construtor:

- O nome do usuário;
- O tipo de operação (treinamento ou inferência);
- A atividade desempenhada pelo usuário (caso seja um treinamento);
- A lista de todas os objetos da classe "SensorData" criados durante a coleta.

Por se tratar de uma requisição HTTP de grande conteúdo, é utilizado o método "POST" para o envio. Cada um dos dados passados ao construtor compõe a requisição como um argumento no formato exibido no Quadro 2:

Quadro 2 – Estrutura da requisição

Argumento	**Valor**
Nome	Nome do usuário
Tipo	Tipo de operação
Atividade	Atividade física realizada
SensorsData	String com lista de dados

Fonte: Autor.

A lista de instâncias da classe "SensorData" não pode ser enviada diretamente na requisição HTTP, portanto as instâncias são reescritas no formato de uma String parametrizada, como exemplificado na Figura 5.

Figura 5 – Exemplo de requisição HTTP com dados de treinamento

```
HTML Form URL Encoded: application/x-www-form-urlencoded
  Form item: "Nome" = "Antonio"
      Key: Nome
      Value: Antonio
  Form item: "Tipo" = "Treinamento"
      Key: Tipo
      Value: Treinamento
  Form item: "Atividade" = "Andando"
      Key: Atividade
      Value: Andando
  Form item: "SensorsData" = "TYPE_ACCELEROMETER;0.12;0.15;0.0005<SD>TYPE_GYROSCOPE;0.5;-1.2;-0.000232"
      Key: SensorsData
      Value: TYPE_ACCELEROMETER;0.12;0.15;0.0005<SD>TYPE_GYROSCOPE;0.5;-1.2;-0.000232
```

Fonte: Autor.

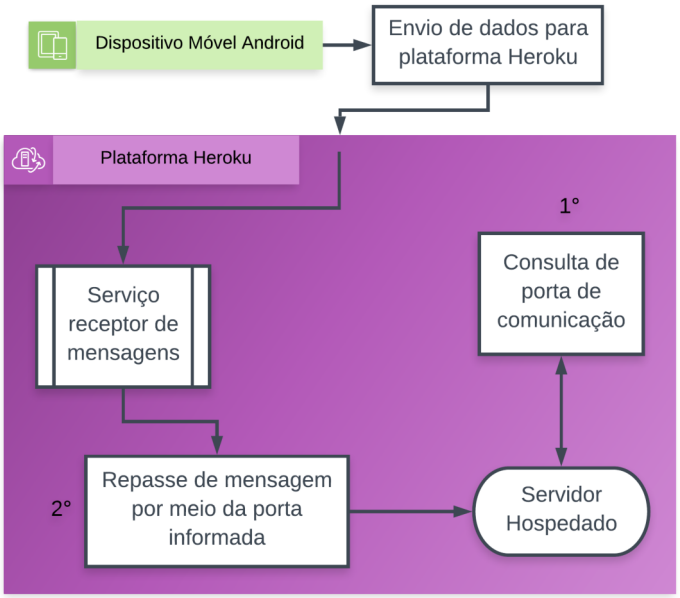
Assim que a composição da requisição HTTP for completada, a mesma é enviada para o endereço *web* fornecido pela plataforma Heroku. Isto conclui o módulo de envio de dados e também o trabalho efetuado pelo aplicativo no dispositivo móvel.

3.3 Recebimento de Dados

A partir desta seção, todas as camadas descritas estão alocadas em um serviço de nuvem.

3.3.1 Modelo de servidor

Figura 6 – Modelo de servidor



Fonte: Autor.

Todos os dados enviados pelo aplicativo Android são direcionados ao endereço *web*: “https://activity-inferencer.herokuapp.com/echoPost”. O domínio “activity-inferencer.herokuapp” é fornecido pelo serviço de nuvem Heroku e é a principal forma de comunicação com o servidor.

O modelo de nuvem da Heroku não permite contato direto com os sistemas nela hospedados. Portanto, a comunicação não é realizada diretamente pela porta 80 destinada ao protocolo HTTP. Em vez disto, é utilizado um serviço auxiliar da plataforma Heroku que fornece uma porta temporária para o servidor hospedado, como representado na Figura 6.

Assim, o servidor deverá utilizar a variável de ambiente "PORT" exclusiva da plataforma Heroku para descobrir qual porta deve utilizar para comunicação. Todas as requisições enviadas ao endereço web fornecido são primeiramente recebidas por uma camada externa da plataforma. Esta camada irá então repassar a requisição ao servidor hospedado através da porta informada na variável de ambiente.

3.3.2 Interpretação da mensagem

Como discutido na seção 3.2.2, o aplicativo móvel irá efetuar uma requisição HTTP utilizando o método "POST". A camada de recebimento de dados é responsável por interpretar a mensagem contida nesta requisição com o objetivo de extrair os dados do usuário, bem como também todos os dados gerados pelos sensores de movimento.

Primeiramente são extraídos os argumentos presentes na requisição HTTP: o nome do usuário, o tipo de operação (treinamento ou inferência), os dados gerados pelos sensores e a atividade física realizada (caso tenha sido informada).

Os dados gerados pelos sensores estão no formato de uma *String* parametrizada. Esta *String* é decomposta para realizar a reconstrução das instâncias da classe "SensorData" que foram criadas no aplicativo.

3.4 Pré-processamento

Esta camada é responsável por realizar o pré-processamento de todos os dados presentes nos objetos da classe "SensorData". Após estas operações, as múltiplas tuplas de dados de "SensorData" passarão a ser representadas por apenas uma tupla.

3.4.1 Motivação

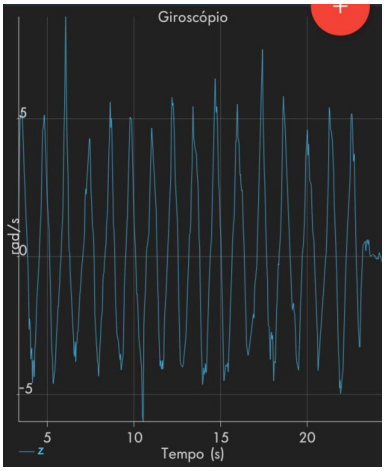
Cada caso de treinamento ou inferência gera múltiplos dados que descrevem toda a movimentação efetuada pelo usuário no formato de objetos "SensorData". Como já discutido na seção 2.2.3, os classificadores presentes na biblioteca Weka consideram cada tupla de dados como um caso individual a ser classificado.

Portanto não é adequado utilizar diretamente múltiplas tuplas de dados para a descrição de cada caso. Por este motivo, esta camada utilizará técnicas matemáticas para reduzir as múltiplas tuplas de cada caso para apenas uma.

3.4.2 Seleção de Picos

Os dados de movimentação em cada eixo gerados pelo giroscópio, quando representados em função do tempo, possuem um formato semelhante ao da Figura 7. A curva representada tem o formato de um senoide que oscila de maneira estável entre valores positivos e negativos. Devido à estabilidade da curva, é possível simplificar os dados selecionando apenas os valores presentes nos picos superiores e inferiores da curva.

Figura 7 – Exemplo de captura por meio do giroscópio



Fonte: Autor.

A seleção dos picos superiores e inferiores da curva é feita por meio do algoritmo descrito na Figura 8 a seguir:

Figura 8 – Algoritmo de seleção de picos

```
   Entrada: Lista de dados gerados pelo giroscópio
   Saída: Lista com picos superiores e inferiores
1  Elemento atual = Início da lista;
2  Próximo elemento = Elemento atual + 1;
3  enquanto Posição do próximo elemento < Tamanho da lista de entrada faça
4      se Procurando por pico superior então
5          se Elemento atual > Próximo elemento então
6              Adiciona elemento atual na lista de saída;
7              Passa a procurar por pico inferior;
8          senão
9              Vai para próxima posição na lista;
10         fim
11     senão
12         se Elemento atual < Próximo elemento então
13             Adiciona elemento atual na lista de saída;
14             Passa a procurar por pico superior;
15         senão
16             Vai para próxima posição na lista;
17         fim
18     fim
19 fim
```

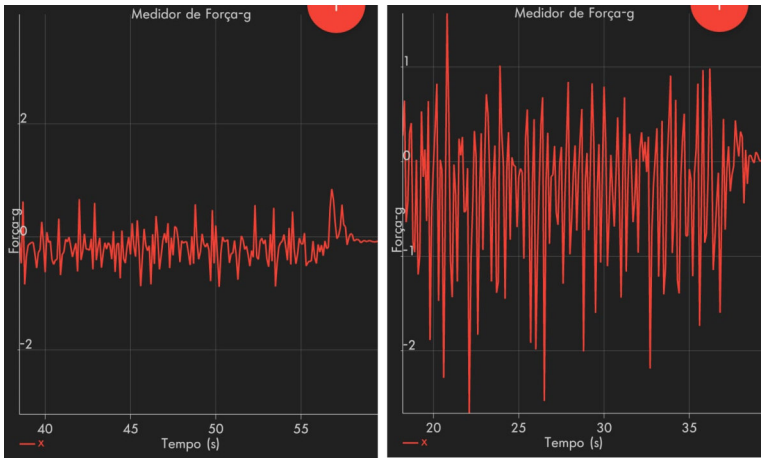
Fonte: Autor.

Após esta operação, os objetos “SensorData” gerados pelo giroscópio que não foram selecionados pelo algoritmo são descartados por não contribuírem de maneira significativa para a representação da atividade física.

3.4.3 Média Aritmética

Os dados gerados pelo acelerômetro para cada uma das atividades físicas possuem valores dentro de um certo intervalo. Isto é evidenciado ao comparar dados gerados a partir de cada atividade física, sendo possível verificar que os intervalos são geralmente bem distintos, como representado no exemplo da Figura 9.

Figura 9 – Comparação de captura de usuário andando (à esquerda) e usuário correndo (à direita)



Fonte: Autor.

Considerando-se estas características, é possível realizar a média aritmética nos dados de cada eixo físico. Desta forma os dados de cada eixo serão representados por valores simplificados em apenas uma tupla.

3.4.4 Raiz Quadrada da Média dos Quadrados

Como já explicado, o giroscópio gera dados com uma distribuição semelhante a um senoide. Estes dados são pré-processados pelo algoritmo de Seleção de Picos e como resultado são obtidos diversos valores próximos, mas simétricos. Se a média aritmética comum for aplicada a este resultado, serão obtidos apenas valores próximos de zero.

A RMS é utilizada como forma de contornar este problema, já que esta realiza uma média que desconsidera o sinal dos valores. Desta forma é possível uma obter uma simplificação dos dados gerados pelo giroscópio sem descaracterizá-los.

3.5 Armazenamento

Todo o armazenamento de dados persistente do sistema é feito através de um banco de dados SQL. Este banco possui duas tabelas principais: a de dados de treinamento e a de resultados de inferências.

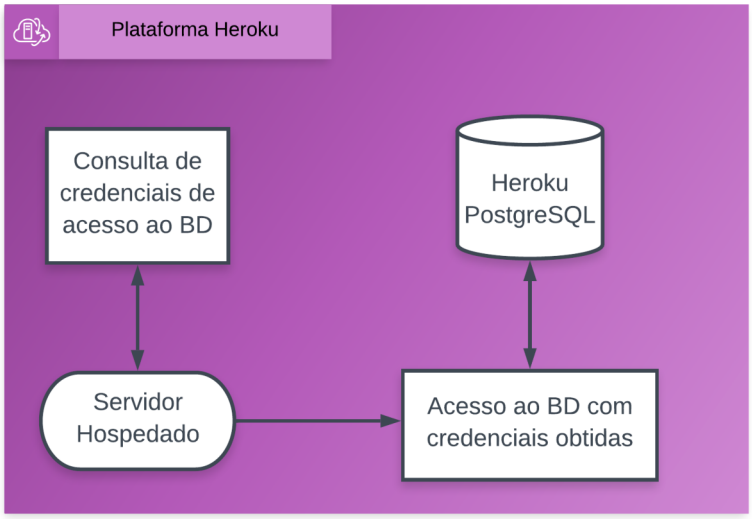
3.5.1 Heroku PostgreSQL

O PostgreSQL é um SGBD objeto-relacional de código aberto desenvolvido majoritariamente na linguagem de programação C. Este banco possui recursos avançados tais como: consultas complexas, chaves estrangeiras, gatilhos, visões e integridade transacional.

O banco de dados PostgreSQL utilizado no sistema é fornecido pela plataforma Heroku como um *add-on* (recurso adicional) e está ligado à aplicação por meio de uma API própria da plataforma. Assim como a porta de comunicação citada em 3.3.1, as credenciais de acesso ao banco de dados não são decisão do desenvolvedor.

É possível verificar as credenciais de acesso ao banco de dados através do painel de controle de aplicações fornecido pela Heroku, entretanto não é possível utilizá-las permanentemente já que estas são rotacionadas periodicamente pela plataforma. Por este motivo, não é recomendado utilizá-las diretamente no código da aplicação.

Figura 10 – Acesso ao banco de dados



Fonte: Autor.

A Heroku recomenda que todo acesso ao banco de dados deve primeiramente consultar as credenciais de acesso através de variáveis de ambiente, conforme a Figura 10. Desta forma, a aplicação pode estar sempre com os dados de acesso atualizados.

3.5.2 Base de Treinamento

A camada de pré-processamento, quando estiver trabalhando em dados de treinamento, irá gerar um conjunto de dados que será utilizado para o treinamento do algoritmo de classificação de atividades físicas. Como será explicado em 3.6, os dados de cada um dos casos de treinamento precisam estar disponíveis simultaneamente para o algoritmo. Portanto, é necessário armazenar, além do conhecimento para a realização de inferências, os dados de todos os casos de treinamento.

Para o tratamento desta característica, foi criado um banco de dados responsável por armazenar os dados pré-processados de todos os casos de treinamento. Os dados estão dispostos no formato da Figura 11:

Figura 11 – Estrutura da base de treinamento

Treinamento	
id	Chave primária
nome	Nome do usuário que realizou o treinamento
atividade	Atividade realizada pelo usuário
mediaaccelx	Média de aceleração no eixo X
mediaaccely	Média de aceleração no eixo Y
mediaaccelz	Média de aceleração no eixo Z
rmsaccelx	RMS de aceleração no eixo X
rmsaccely	RMS de aceleração no eixo Y
rmsaccelz	RMS de aceleração no eixo Z
mediagyrox	Média de taxa de rotação no eixo X
mediagyroy	Média de taxa de rotação no eixo Y
mediagyroz	Média de taxa de rotação no eixo Z
rmsgyrox	RMS de taxa de rotação no eixo X
rmsgyroy	RMS de taxa de rotação no eixo Y
rmsgyroz	RMS de taxa de rotação no eixo Z

Fonte: Autor.

3.6 Treinamento do Algoritmo de Inferência

Como já explicado, o aplicativo Android, quando está no modo de treinamento, irá enviar os dados gerados pelos sensores de movimento para o servidor na nuvem e após o pré-processamento estes dados serão armazenados em um banco de dados. Este processo é completamente automatizado no aplicativo e no sistema hospedado na nuvem. Entretanto, o módulo de treinamento do classificador não é iniciado para cada caso de treinamento de forma automatizada.

O processo de treinamento pode ser demorado, portanto optou-se que o mesmo seja iniciado manualmente depois do recolhimento de uma grande quantidade de dados novos. Sempre que o módulo de treinamento é iniciado, ele irá utilizar todos os dados presentes na base de treinamento, não havendo uma forma de complementar o modelo de inferência apenas com os dados novos.

Este módulo realiza o treinamento do algoritmo em duas etapas, sendo elas a recuperação dos dados na base de treinamento e o treinamento e armazenamento do conhecimento do classificador. Estas etapas tem como base a utilização de recursos da biblioteca Weka.

3.6.1 Recuperação dos dados

Como os formatos de dados utilizados em um banco de dados e na linguagem Java podem ser diferentes, é necessário especificar para a biblioteca Weka como ela deve interpretar os dados de cada coluna no banco. Esta especificação consiste em associar cada formato de dados no banco com um formato básico da linguagem Java.

Como já explicado anteriormente, a base de treinamento do classificador está armazenada em um banco de dados SQL. A biblioteca Weka é utilizada para a recuperação e adaptação destes dados para uma estrutura de dados própria da biblioteca.

Para acessar o banco, a biblioteca precisa de suas credenciais de acesso. Como a arquitetura utiliza o Heroku PostgreSQL, é necessário verificá-las através da variável de ambiente referente ao banco de dados. Assim que estas informações são obtidas, é então realizada uma consulta ao banco de dados através da classe “InstanceQuery”. Esta consulta irá retornar todas as tuplas existentes na base de treinamento.

O retorno da consulta é armazenado em um objeto da classe “Instances” para uso na próxima etapa. Antes de realizar o processo de treinamento do classificador, são eliminadas das tuplas as colunas de chave primária e nome do usuário já que estas não tem utilidade ao classificador, apenas atrapalhando o algoritmo de treinamento.

3.6.2 Treinamento do classificador

O treinamento é realizado a partir do instanciamento da classe “RandomForest”, sendo esta classe uma estrutura de dados que representa um classificador que utiliza o algoritmo “RandomForest”. Após o instanciamento, os dados da base de treinamento presentes no objeto de classe “Instances” são então utilizados para a construção do classificador.

Após o treinamento, o modelo de inferência está armazenado somente em memória no objeto “RandomForest”. Como a persistência dos dados presentes neste modelo é necessária para o funcionamento mais eficiente do sistema, os mesmos são escritos em um arquivo.

3.7 Inferência

O processo de inferência se inicia com o envio de dados de sensores de movimento a partir do aplicativo. Assim como no treinamento, os dados serão recebidos pelo servidor em nuvem e serão otimizados pelas mesmas técnicas de pré-processamento.

Ao fim desta etapa, este conjunto de dados é enviado para o módulo de predição. Este módulo, durante sua inicialização, irá utilizar o arquivo mencionado em 3.6.2 para reconstruir o modelo de inferência. Os dados são então classificados por este modelo, gerando um rótulo de classe. Ao final, este rótulo é enviado de volta ao usuário, informando a predicação da atividade física realizada. Opcionalmente, é possível armazenar os dados desta inferência em um banco de dados para visualização futura.

4 RESULTADOS

Neste capítulo é descrito o método de verificação utilizado para avaliar a eficiência do sistema construído e seus resultados. Esta avaliação é fundamental, pois é necessário validar a capacidade do sistema realizar inferências corretas sobre a atividade física atual do paciente. Ao final do capítulo, é realizada uma discussão sobre os resultados apresentados.

4.1 Metodologia de Verificação

O principal objetivo desta avaliação é mensurar o desempenho das inferências realizadas para cada atividade física. Para tal, é calculada a taxa de acerto por meio da fórmula 4.1:

$$Taxa\ de\ acerto = \frac{Número\ de\ acertos}{Total\ de\ amostras} \tag{4.1}$$

Também são calculadas a média, mediana e desvio padrão para as taxas de acerto de cada atividade física.

Os dados utilizados nesta avaliação foram obtidos através de 15 voluntários durante a realização das seguintes atividades: andar, correr, estar em pé, estar sentado e estar deitado. Os voluntários possuíam idade entre 18 e 30 anos e não tinham nenhuma deficiência física que prejudique a locomoção.

A captura dos dados de sensores de movimento foi realizada através do aplicado Android mencionado em 3.1. O aplicativo foi utilizado em diversos modelos de *smartphones* com sistema operacional Android e todos estes possuíam acelerômetro e giroscópio embutidos. Durante a captura, os dispositivos estavam posicionados na coxa dos voluntários por meio de bolsos laterais ou frontais e a execução de cada atividade física durou 15 segundos.

Os dados capturados foram alocados em um banco de dados, assim como mencionado no capítulo 3.5.2. O *software* Weka foi então utilizado para aplicar o algoritmo “Random Forest” sobre esta base de treinamento, realizando os testes por meio do método de validação cruzada (AMAZON, 2017).

4.2 Resultados apresentados

Os primeiros resultados foram obtidos ao utilizar todos os atributos disponíveis: média aritmética e RMS sobre os dados gerados pelo acelerômetro e giroscópio. A taxa de acertos deste primeiro experimento pode ser vista na Tabela 1 e sua matriz de confusão na Tabela 2.

Tabela 1 – Taxa de acerto para cada atividade física ao utilizar todos os atributos

Atividade Física	**Taxa de Acerto (%)**
Parado em pé	86,67
Sentado	58,82
Deitado	75
Andando	84,21
Correndo	87,5
Média	78,44
Mediana	84,21
Desvio padrão	12,04

Fonte: Autor.

Tabela 2 – Matriz de confusão ao utilizar todos os atributos

Classificado como:	**Parado em pé**	**Sentado**	**Deitado**	**Andando**	**Correndo**
Parado em pé	**13**	0	0	2	0
Sentado	1	**10**	6	0	0
Deitado	0	3	**12**	1	0
Andando	2	0	0	**16**	1
Correndo	0	0	0	2	**14**

Fonte: Autor.

Como já explicado em 3.4.4, os dados gerados pelo giroscópio são descaracterizados ao utilizar a média aritmética como técnica de pré-processamento. Portanto, como um segundo experimento, os atributos obtidos por esta técnica foram removidos da verificação. A taxa de acertos resultante pode ser vista na Tabela 3 e sua matriz confusão na Tabela 4.

Tabela 3 – Taxa de acerto para cada atividade física ao utilizar somente RMS

Atividade Física	**Taxa de Acerto (%)**
Parado em pé	93,33
Sentado	70,59
Deitado	75
Andando	89,47
Correndo	81,25
Média	81,93
Mediana	81,25
Desvio padrão	9,54

Fonte: Autor.

Tabela 4 – Matriz de confusão ao utilizar somente RMS

Classificado como:	**Parado em pé**	**Sentado**	**Deitado**	**Andando**	**Correndo**
Parado em pé	**14**	0	0	1	0
Sentado	1	**12**	4	0	0
Deitado	0	3	**12**	1	0
Andando	0	0	0	**17**	2
Correndo	0	0	0	3	**13**

Fonte: Autor.

É notável que o classificador possui as menores taxas de acerto nas atividades “Sentado” e “Deitado” e além disso as atividades são frequentemente confundidas entre si. Este problema ocorre, pois a posição do *smartphone* ao realizar a captura de dados é muito semelhante entre as duas atividades. Pensando nesta característica, foi realizado um terceiro experimento, onde as atividades “Sentado” e “Deitado” foram unificadas na atividade “Em repouso”.

Tabela 5 – Taxa de acerto para cada atividade física ao utilizar classe “Em repouso”

Atividade Física	**Taxa de Acerto (%)**
Parado em pé	86,67
Em repouso	100
Andando	84,21
Correndo	87,5
Média	89,6
Mediana	86,5
Desvio padrão	7,08

Fonte: Autor.

Tabela 6 – Matriz de confusão ao utilizar classe “Em repouso”

Classificado como:	**Parado em pé**	**Em repouso**	**Andando**	**Correndo**
Parado em pé	**13**	0	2	0
Em repouso	0	**33**	0	0
Andando	0	0	**16**	3
Correndo	0	0	2	**14**

Fonte: Autor.

4.3 Considerações finais

O modelo de treinamento utilizado no primeiro experimento, apesar de possuir uma boa taxa de acerto média, é extremamente falho. Como visto na Tabela 1, a atividade “Sentado” foi corretamente classificada somente 58,82% das vezes. Este problema ocorre devido a técnica utilizada para a captura de dados. Durante o uso do aplicativo para a coleta de dados, não houve uma padronização na orientação espacial do *smartphone* utilizado. Consequentemente, atividades

físicas da mesma classe podem ter atributos de valores próximos, mas simétricos, o que torna o treinamento do classificador menos eficiente.

No segundo experimento, os atributos obtidos por médias aritméticas foram desconsiderados. Como a média obtida por RMS desconsidera o sinal dos dados, o problema verificado no primeiro experimento foi solucionado. Pode ser visto na Tabela 3 que houve uma melhoria na classificação de quatro das atividades físicas, incluindo a "Sentado". Pode-se considerar que as taxas de acerto para cada atividade física no segundo experimento estão dentro do aceitável. Entretanto, como já mencionado, o classificador tem certa dificuldade em diferenciar as atividades "Sentado" e "Deitado".

As taxas de acerto exibidas na Tabela 5 demonstram que o modelo utilizado no terceiro experimento é bem mais eficiente que os dois anteriores. Por meio dos resultados obtidos, é possível afirmar que a arquitetura teve um desempenho satisfatório e que o objetivo deste trabalho foi cumprido.

Apesar da unificação das classes "Sentado" e "Deitado" causar uma perda de ganho de informação sobre o estado atual do paciente, acredita-se que a classe "Em repouso" é suficiente, já que o objetivo do sistema é dar ao profissional da área médica uma ferramenta para avaliar o grau de risco atual do paciente.

5 CONCLUSÃO

Este trabalho apresentou uma proposta de sistema com atuação na área de *Ambient Assisted Living*. Este sistema tem como foco o acompanhamento de pacientes com doenças crônicas, permitindo o reconhecimento de suas atividades físicas por meio de um dispositivo móvel. Também foi objetivo deste trabalho contribuir para área de computação em nuvem, ao apresentar uma arquitetura de sistema dividida em dois ambientes: um dispositivo móvel Android e um servidor em nuvem. Tal modelo de implantação teve como objetivo aumentar a eficiência, ao considerar que *smartphones* podem muitas vezes ter baixo poder computacional e autonomia de bateria.

Ao fim do trabalho foi apresentada uma metodologia de avaliação para determinar a eficiência do sistema construído, baseado no seu desempenho ao realizar predições sobre a atividade física atual do usuário. Esta avaliação demonstrou que, apesar de haver um certo problema para diferenciar atividades semelhantes, é possível obter taxas de acerto satisfatórias ao realizar alterações no método de treinamento inicialmente proposto. Com este resultado, pode ser dito que este trabalho teve êxito e que o sistema proposto pode ser útil como uma ferramenta adicional para os profissionais da área médica que desejam tem um acompanhamento maior sobre seus pacientes.

5.1 Trabalhos Futuros

Planeja-se como trabalho futuro a implantação do sistema desenvolvido em dispositivos móveis além de *smartphones*, tais como *smartwatches*, visto que estes aparelhos são mais adequados para o porte constante durante o dia a dia e para a prática de atividades físicas.

Também é sugerido como um estudo futuro, a ampliação do número de sensores ligados ao paciente, como por exemplo sensores de temperatura, frequência cardíaca, taxa respiratória ou glicemia. Tais dispositivos podem ser encontrados em modelos *wireless* ou já embutidos em outros aparelhos móveis e podem contribuir para o cuidado do paciente, ao ampliar o monitoramento feito pelo profissional da área médica ou ao serem utilizados por classificadores para inferir o estado atual do paciente.

REFERÊNCIAS

ALPAYDIN, E. *Introduction to Machine Learning*. 2nd. ed. [S.l.]: The MIT Press, 2010. ISBN 026201243X, 9780262012430. Citado na página 18.

AMAZON. *Validação cruzada*. 2017. Disponível em: <https://docs.aws.amazon.com/pt_br/machine-learning/latest/dg/cross-validation.html>. Acesso em: 05 jun. 2019. Citado na página 37.

ANDROID DEVELOPERS. *Sensors Overview*. 2011. Disponível em: <https://developer.android.com/guide/topics/sensors/sensors_overview>. Acesso em: 15 mar. 2019. Citado 2 vezes nas páginas 17 e 18.

BREIMAN, L. Random forests. *UC Berkeley TR567*, 1999. Citado na página 22.

CLOUDFLARE. *Why Use HTTPS?* 2018. Disponível em: <https://www.cloudflare.com/learning/ssl/why-use-https/>. Acesso em: 29 mar. 2019. Citado na página 28.

FANG, S.-H.; LIANG, Y.-C.; CHIU, K.-M. Developing a mobile phone-based fall detection system on android platform. In: IEEE. *2012 Computing, Communications and Applications Conference*. [S.l.], 2012. p. 143–146. Citado na página 14.

GANDHEWAR, N.; SHEIKH, R. Google android: An emerging software platform for mobile devices. *International Journal on Computer Science and Engineering*, v. 1, n. 1, p. 12–17, 2010. Citado na página 16.

HO, T. K. Random decision forests. In: IEEE. *Proceedings of 3rd international conference on document analysis and recognition*. [S.l.], 1995. v. 1, p. 278–282. Citado na página 20.

KOEHRSEN, W. *An Implementation and Explanation of the Random Forest in Python*. 2018. Disponível em: <https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76>. Acesso em: 18 jun. 2019. Citado 2 vezes nas páginas 21 e 22.

MELL, P.; GRANCE, T. et al. The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, 2011. Citado 3 vezes nas páginas 14, 23 e 24.

MORAES NETO, A. de J. Ambiente pervasivo para acompanhamento personalizado de portadores de doenças crônicas. São Luís - MA, 2016. Citado 3 vezes nas páginas 14, 17 e 20.

QUINLAN, J. R. Simplifying decision trees. *International journal of man-machine studies*, Elsevier, v. 27, n. 3, p. 221–234, 1987. Citado na página 20.

RIBEIRO FILHO, J. D. P. *MHARS: Um Sistema Pervasivo de Reconhecimento de Atividades para Ambientes de Assistência à Autonomia no Domicílio*. Dissertação (Mestrado) — Universidade Federal do Maranhão, São Luís - MA, 2015. Citado 2 vezes nas páginas 14 e 17.

SUN, H. et al. Promises and challenges of ambient assisted living systems. In: IEEE. *2009 Sixth International Conference on Information Technology: New Generations*. [S.l.], 2009. p. 1201–1207. Citado na página 14.

YI, W.-J. et al. Wearable sensor data fusion for remote health assessment and fall detection. In: IEEE. *IEEE International Conference on Electro/Information Technology*. [S.l.], 2014. p. 303–307. Citado na página 15.