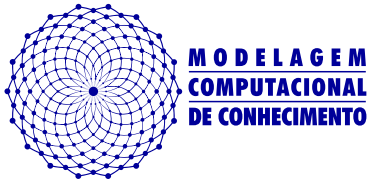


Dissertação de Mestrado

Avaliação de modelos de classificação automática de atividades diárias para dispositivos de baixo custo

Wylken dos Santos Machado
wylken.ufal@gmail.com

Orientadores:

Drª. Eliana Silva de Almeida
Dr. Andre Luiz Lins de Aquino

Maceió, Março de 2020

Wylken dos Santos Machado

Avaliação de modelos de classificação automática de atividades diárias para dispositivos de baixo custo

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-Graduação em Modelagem Computacional de Conhecimento do Instituto de Computação da Universidade Federal de Alagoas.

Orientadora: Drª. Eliana Silva de Almeida

Maceió, Março de 2020

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecária: Taciana Sousa dos Santos - CRB-4 - 2062

M149a Machado, Wylken dos Santos.
Avaliação de modelos de classificação automática de atividades diárias para dispositivos de baixo custo / Wylken dos Santos Machado. – 2020.
68 f. : il. ; figs. color.

Orientadora: Eliana Silva de Almeida.
Coorientador: André Luiz Lins de Aquino.
Dissertação (Mestrado em Modelagem Computacional do Conhecimento) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2020.

Bibliografia: 64-68.

1.Aprendizagem de máquina. 2. Atividades de vida diária. 3. Arcabouço computacional. 4. Algoritmos computacionais. 5. Modelagem computacional. I Título.

CDU: 004.8

Folha de Aprovação

Wylken dos Santos Machado

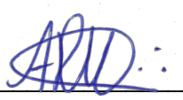
Avaliação de modelos de classificação automática de atividades diárias para dispositivos de baixo custo

Dissertação submetida ao corpo docente do Programa de Pós-Graduação em Modelagem Computacional de Conhecimento da Universidade Federal de Alagoas e aprovada em 06 de março de 2020.

Profa. Dra. Eliana Silva de Almeida
Faculdade de Medicina- UFAL
Orientadora

Prof. Dr. André Luiz Lins de Aquino
Instituto de Computação - UFAL
Coorientador

Banca Examinadora:



Prof. Dr. Aydano Pamponet Machado
Instituto de Computação - UFAL
Examinador Interno

Prof. Dr. Carlos Maurício Serodio Figueiredo
Escola Superior de Tecnologia - UEA
Examinador Externo

RESUMO

Neste trabalho foram utilizados algoritmos de aprendizado de máquina para identificar Atividades Diárias (ADLs) em três conjuntos de dados públicos, ARCMA, HMP e UMAFall. A ideia central é definir uma metodologia que seja capaz de avaliar um conjunto de métodos de aprendizado de máquina, em um cenário específico, com o objetivo de alcançar os melhores resultados na classificação automática de Atividades Diárias, com foco na simplificação da implantação e da estrutura necessária para a sua execução. Nos dias de hoje os estudos com melhores resultados utilizam dados de redes de sensores instalados tanto no ambiente quanto no corpo dos voluntários, esse tipo de abordagem torna a aplicação mais complexa e dificulta a sua implantação. Neste trabalho serão utilizados dados gerados apenas por um sensor inercial (acelerômetro ou magnetômetro, dependendo da base de dados), essa simplificação implica em uma menor complexidade tanto no desenvolvimento, quando na implantação, além de propiciar soluções mais baratas. Dessa forma , o objetivo deste trabalho é construir arcabouço computacional, que envolve parametrização, execução, simulação e validação, permitindo assim, encontrar modelo capaz de realizar a classificação de Atividades Diárias utilizando dispositivos de baixo custo. Para verificar o desempenho do modelo final foi utilizado o Raspberry Pi 3 B.

Para alcançar esse objetivo, a metodologia proposta realiza o tratamento dos dados, com a exclusão de entradas inválidas, balanceamento das bases, extração das características relevantes, avaliação dos algoritmos, busca da melhor janela de leitura e utilização de filtro limiar para melhorar a acurácia. Para mitigar os resultados obtidos utilizamos a validação cruzada estratificada para encontrar o F-score e Acurácia dos métodos. Foram avaliados os seguintes algoritmos: k-Nearest Neighbors, Naive Bayes, Support Vector Machine, Decision Tree, Random Forest e Extra-Trees. O algoritmo Extra-Trees apresentou os melhores resultados com uma acurácia final de 92.06%, 93.97% e 97.79% e um F-score de 91.29%, 92.76% e 97.41% para as bases ARCMA, HMP e UMAFall, respectivamente, em um cenário quem que seja tolerável a exclusão de duas atividades das bases de dados.

Utilizamos um filtro de decisão que escolhe se o registro deve ser descartado ou não, levando em consideração a tabela de probabilidade retornada por cada método, o que elevou a acurácia da classificação em 16.33%, 14.95% e 9.05% nas bases ARCMA, HMP e UMAFall. Toda modelagem aplicada neste trabalho é relevante para futuros estudos que tenham o objetivo de implementar uma aplicação real para realizar a detecção automática de ADLs, sendo essa portanto, a principal contribuição deixada por este trabalho.

Keywords: *Machine Learning, ADL, Activities of Daily Living, Activity Recognition*

ABSTRACT

In this work, machine learning algorithms were used to identify Activities of Daily Living (ADLs) in three public data sets, ARCMA, HMP and UMAFall. The central idea is to define a methodology that is capable of evaluating a set of machine learning methods, in a specific scenario, in order to achieve the best results in the automatic classification of Activities of Daily Living, considering the simplification of the implantation and the necessary structure for its execution.Nowadays the studies with better results use data from sensor networks installed both in the environment and in the body of the volunteers, this type of approach makes the application more complex and makes its implementation difficult.This work will use data generated only by an inertial sensor (accelerometer or magnetometer, depending on the database), this simplification implies less complexity in both development and implementation, in addition to providing cheaper solutions. Thus, the objective of this work is to build a computational framework, which involves parameterization, execution, simulation and validation, thus allowing to find a model capable of performing the classification of Activities of Daily Living using low cost devices. To check the performance of the final model, the Raspberry Pi 3 B was used.

To achieve this goal, the proposed methodology performs the data processing, excluding invalid entries, balancing the bases, extracting the relevant characteristics, evaluating the algorithms, search for the best reading window and using a threshold filter to improve accuracy. To mitigate the results obtained, we use stratified cross-validation to find the F-score and Accuracy of the methods. The following algorithms were evaluated: k-Nearest Neighbors, Naive Bayes, Support Vector Machine, Decision Tree, Random Forest, and Extra-Trees. The Extra-Trees algorithm presented the best results with a final accuracy of 92.06%, 93.97% e 97.79%, and 91.29%, 92.76% e 97.41% of F-score, for the ARCMA, HMP and UMAFall bases, respectively, in a scenario who would be tolerable to exclude two activities from the databases.

We use a decision filter that chooses whether the record should be discarded or not, taking into account the probability table returned by each method, this increased the accuracy of the classification by 16.33%, 14.95% and 9.05% on the ARCMA, HMP, and UMAFall bases. All modeling applied in this work is relevant for future studies that aim to implement a real application to perform automatic detection of ADLs, which is, therefore, the main contribution of this work.

Keywords: *Machine Learning, ADL, Activities of Daily Living, Activity Recognition.*

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus, que me deu saúde, força, coragem e uma família maravilhosa. A minha mãe Ednalva S.Machado, ao meu pai Cremilson M. da Silva e meu irmão Cleberson S. Machado, pelo apoio, incentivo, amor e carinho ao longo desta longa caminhada que é a vida. Obrigado também pela força para que eu pudesse subir mais esse degrau. A minha esposa Rafaela S. da Silva, companheira e amiga, por toda a atenção, amor e carinho que a mim foram dados, que de forma especial me deu força e coragem, me apoiando nos momentos de dificuldades. Ao Professor Dr. André Luiz Lins de Aquino e Professora Drª. Eliana S. Almeida, pela disponibilidade e atenção prestadas. Obrigado também pela ajuda nos momentos difíceis, pelos conselhos e inúmeras sugestões que foram importantes para a realização deste trabalho. E todos aqueles que de alguma forma estiveram e estão próximos de mim, fazendo esta vida valer cada vez mais a pena. Agradecemos também o apoio financeiro do CNPq, FAPEAL e FAPESP.

Wylken dos Santos Machado

LISTA DE FIGURAS

SUMÁRIO

1

INTRODUÇÃO

Em 6 de novembro de 2014 a Organização Mundial de Saúde (OMS) declarou que nas próximas décadas o número de pessoas com 60 anos ou mais, irá passar de 800 milhões para 2 bilhões até o ano de 2050 (Nations, 2014). Esse cenário acompanhado com desenvolvimento tecnológico, vem fomentando as pesquisas relativas a qualidade de vida dessas pessoas. Uma das áreas que abordam essa temática é o reconhecimento automático de atividades da vida diária, usualmente chamada de ADLs (Activity of Daily Living), que se utiliza de diversas técnicas computacionais para identificar as atividades realizadas por uma pessoa. Segundo Mlinac & Feng (2016), ADLs incluem as habilidades fundamentais necessárias para administrar as necessidades físicas básicas de uma pessoa, abrangendo as seguintes áreas: higiene pessoal, beleza, necessidades fisiológicas, deslocamento, alimentação etc.

O reconhecimento automático dessas atividades pode exercer um papel fundamental na qualidade de vida de pessoas que necessitam de cuidados especiais, contribuindo para melhorar o nível de segurança e saúde desses indivíduos, além de proporcionar economia de recursos financeiros, com substituição total ou parcial de serviços de cuidadores. Essa é uma área promissora e muitas aplicações podem surgir em um futuro próximo. Essa área possui diversos desafios, entre eles, está o desenvolvimento de sistemas que possam ser executados de forma viável em dispositivos baratos, de fácil acesso e com baixa complexidade de implementação.

A maioria dos trabalhos relacionados a este tema utiliza sensores inerciais instalados no corpo de voluntários para realizar a coleta dos dados. Em muitos deles, é formado uma rede de sensores para realizar esse trabalho, ou seja, sensores espalhados em mais de um local do corpo. Esse tipo de abordagem possui uma grande dificuldade de ser utilizado em ambientes reais, fora do laboratório, pois necessita de uma grande infraestrutura para ser implantado, além de provavelmente propiciar grande desconforto para as pessoas monitoradas, já que estas precisaram vestir roupas ou acessórios com uma grande quantidade de sensores espalhados pelo corpo.

Nesse trabalho apresentamos uma metodologia para avaliar um grupo de métodos de aprendizado de máquina, com o objetivo de alcançar os melhores resultados na classificação automática de Atividades Diárias, com foco na simplificação da implantação e da estrutura necessária para a sua execução, levando em consideração a qualidade da solução, a complexidade de implantação e o desempenho em dispositivos de baixo custo. Para tanto, serão utilizados dados gerados apenas por um sensor inercial (acelerômetro ou magnetômetro, dependendo da base de dados), essa simplificação possibilita a utilização de sensores de smartphones e smartwatches, o que implica em uma menor complexidade tanto no desenvolvimento, quando na implantação, além de propiciar soluções mais baratas e acessíveis.

Para alcançar esse objetivo realizamos o tratamento dos dados, com a exclusão de entradas inválidas e balanceamento das bases; extração automática das características relevantes; avaliação dos algoritmos; definição da melhor janela de leitura e identificação do limiar mais adequado para um filtro de decisão que escolhe se o registro deve ser descartado ou não, levando em consideração a tabela de probabilidade retornada na classificação. Para verificar o desempenho dos modelos foi utilizado um Raspberry Pi 3 B, um dispositivo computacional de baixo custo.

Com o objetivo de contribuir para a generalização dos resultados obtidos foram utilizados três conjuntos de dados públicos com registro de ADLs, são eles: ARCMA Casale et al. (2011*a*), HMP Bruno et al. (2013) e UMAFall Casilaria et al. (2017). Essas bases estão hospedadas no repositório da UCI Machine Learning (Dua & Graff, 2017), e possuem dados de sensores inerciais para algumas ADLs executadas por diferentes voluntários. O conjunto de métodos de aprendizado de máquina a serem avaliados em relação aos resultados obtidos na classificação de ADL's é constituído por: *k-Nearest Neighbors* (Cover & Hart, 1967), *Naive Bayes* (Lewis, 1998), *Support Vector Machine* (Vapnik & Cortes, 1995), *Árvore de Decisão* (Shalev-Shwartz & Ben-David, 2014), *Random Forest* (Breiman, 2001), *Extra-Trees* (Geurts et al., 2006) e Perceptron multicamadas (Pal & Mitra, 1992). Estes métodos foram escolhidos com base nos trabalhos relacionados descritos na Seção 2, os quais tem o objetivo de representar os principais métodos matemáticos e computacionais aplicados ao reconhecimento automático de atividades.

Para realizar a extração automática das características relevantes das bases de dados, foi utilizado o framework TSFRESH (Christ et al., 2018). Este Framework foi utilizado em Hooman et al. (2019) para detectar sinais precoces de doença de Parkinson. Tal ferramenta foi escolhida devido a sua popularização para realizar cálculos automáticos de características relevantes de séries temporais, uma ótima documentação e facilidade na sua utilização.

Foram obtidos bons resultados na classificação inicial, porém, levando em consideração a criticidade do contexto envolvido, é necessário elevar ao máximo a acurácia do modelo. Ao realizar uma classificação, os modelos de aprendizado de máquina geram uma matriz de probabilidade para verificar qual é a classe mais provável do novo registro, utilizamos esses dados para verificar o nível de certeza que o método possui ao realizar uma classificação, caso

esse nível esteja abaixo de um limiar definido, o registro é descartado. Tal abordagem elevou a acurácia em 16,33%, 14.95% e 9,05%, para as bases ARCMA, HMP e UMAFall. Os resultados serão apresentados de forma mais detalhada na Seção 4.

O primeiro passo no processo de avaliação foi excluir os dados inválidos de cada base (valores nulos e inconsistentes), em seguida, as características relevantes foram calculadas. Ao analisar os registros verificamos que todas as bases estavam desbalanceadas, ou seja, possuíam muitas entradas para algumas atividades e poucas para outras, dessa forma, realizamos o balanceamento utilizando o processo de subamostragem. O próximo passo foi verificar o desempenho retornada por cada método de aprendizado de máquina, tendo como base a acurácia, f-score, precisão e revocação, seguido da análise da melhor janela de leitura para as ADLs. Após o processo inicial, aplicamos o filtro baseado no nível de certeza para melhorar ao máximo os resultados do modelo encontrado. Em todas as etapas realizamos a validação dos modelos utilizando o método de validação cruzada estratificada. Com os experimentos realizados, em um cenário quem que seja tolerável a exclusão de duas atividades das bases de dados, foi alcançando uma acurácia total de 92.06%, 93.97% e 97.79% para as bases ARCMA, HMP e UMAFall, respectivamente. Esse resultados foram alcançados utilizando os dados gerados apenas por um sensor inercial (acelerômetro para as bases ARCMA e HMP, e magnetômetro para o UMAFall).

Os resultados para as bases UMAFall, ARCMA e HMP estão bem próximos dos apresentados por (Chetty & White, 2016) e (Barshan & Yüksek, 2014) que utilizam uma infraestrutura complexa, com diversos sensores espalhados pelo corpo dos voluntários. O tempo de execução foi compatível com o contexto apresentado, levando em média $0.4457s$ para realizar cada classificação. O esperado nas avaliações é encontrar um modelo preditivo capaz de realizar a detecção de ADLs utilizando dispositivos de baixo custo.

1.1 Objetivo

O objetivo geral do presente trabalho se resume em construir arcabouço computacional, que envolve parametrização, execução, simulação e validação, permitindo assim, encontrar modelo capaz de realizar a classificação da Atividades Diárias utilizando dispositivos de baixo custo.

O seguinte problema é abordado neste trabalho:

> *Como encontrar o melhor método de classificação, em um cenário específico, para realizar detecção automática de Atividades Diárias, levando em consideração a qualidade da solução, a complexidade de implantação e o desempenho em dispositivos de baixo custo?*

Como objetivo específico temos as seguintes metas:

- Encontrar melhor modelo preditivo para classificar Atividades Diárias (ADLs) utilizando dados gerados apenas por um único sensor inercial.
- Apresentar estratégias que melhorem a acurácia dos algoritmos utilizados.
- Testar desempenho do modelo encontrado em um dispositivo de baixo custo (Raspberry Pi 3 B).

Durante os estudos realizados, tivemos um artigo completo aceito em anais de evento nacional:

- Machado, W. S., Barros, P. H., Aquino, A. L. L. & Almeida, E. S. (2018), Avaliação de técnicas de inteligência computacional para identificação de atividades de vida diária, 10º Simpósio Brasileiro de Computação Ubíqua e Pervasiva (SBCUP) pp. 29–38.

No trabalho supracitado foi apresentado uma avaliação de desempenho dos métodos de aprendizado de máquina: K-Nearest Neighbors (kNN), Naive Bayes, Support Vector Machine (SVM), Decision Tree, Random Forest, Extra-Trees e Redes Neurais Recorrentes (RNN). Para realizar os testes foi utilizada a base de dados UMAFall, onde verificou-se que os melhores resultados foram alcançados com o algoritmo Extra-Trees, utilizando os dados gerados pelo sensor magnetômetro instalado na cintura dos voluntários.

Toda modelagem aplicada neste trabalho é relevante para futuros estudos que tenham o objetivo de implementar uma aplicação real para realizar a detecção automática de ADLs, sendo essa portanto, a principal contribuição deixada por essa dissertação. Modelos para detecção de ADLs que utilizam dispositivos de baixo custo, como é o caso de Anguita et al. (2012), são de suma importância para a popularização e democratização desse tipo de solução. Essa tecnologia pode viabilizar o desenvolvimento de aplicações reais com um amplo alcance populacional, incluindo indivíduos de classes econômicas cada vez mais baixas. Além dos benefícios da acessibilidade, aplicações nesses moldes podem ser executadas em ambientes que não possuem acesso 100% do tempo a rede, um diferencial em relação as soluções que necessitam de um ambiente em *cloud*, como é o caso de Chetty & White (2016).

De forma a contribuir ainda mais para a área de pesquisa e facilitar a reprodutibilidade dos estudos, foram disponibilizadas no repositório `https://github.com/SensorNet-UFAL/NewActivityRecognition` todas as codificações produzidas.

1.2 Visão Geral do Texto

Este trabalho está organizado nos seguinte capítulos:

Capítulo 2: fundamentação teórica, apresenta os conceitos relacionados as Atividades Diárias, os trabalhos relacionados e os desafios da área.

Capítulo 3: detalha a abordagem utilizada para resolução do problema, descrevendo os métodos de validação e de classificação, além da busca e descrição das bases de dados utilizadas, caracterização dos dados, seleção dos atributos, métodos de Aprendizado de Máquina, ambiente de desenvolvimento e processo de análise.

Capítulo 4: expõe os resultados alcançados e os dados geradas no estudo.

Capítulo 5: apresenta as considerações finais e os trabalhos futuros.

2

FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo serão apresentados os conceitos sobre as Atividades Diárias e reconhecimento automático. Em seguida temos uma breve síntese sobre os trabalhos relacionados, as estratégias de validação, as bases de dados utilizadas no experimento, o formato dos dados que serão utilizados e finalizamos com os métodos de aprendizado de máquina.

2.1 O que são ADLs

Segundo a Enciclopédia de Neuropsicologia Clínica (Troyer, 2011) Atividades Diárias (ADLs) são atividades de autocuidado importantes para manutenção da saúde e da independência de um indivíduo. Compreendem um amplo espectro de atividades, dentre elas: alimentação, higiêne pessoal, utilização de equipamentos comuns, atividades sociais, entre outros. Além da incapacidade gerada pelo envelhecimento natural do indivíduo, diversas doenças podem afetar a execução das ADLs, são elas: Doença de Alzheimer Clinic (2018), Doenças Reumáticas for Quality & in Health Care (2016), Parkinson Cheon et al. (2015) e diversas outras que causam inaptidão física e/ou demência.

O reconhecimento automático de ADLs consiste na identificação das atividades executadas por um indivíduo, utilizando dados gerados por sensores como entrada de um modelo preditivo. Tal ferramenta pode ajudar no monitoramento de pessoas que necessitam de acompanhamento constante, auxiliando os familiares e/ou cuidadores. Um sistema que implemente essa abordagem teria a capacidade de gerar relatórios e alertas sobre as atividades executadas, além de possibilitar o acompanhamento remoto.

Por exemplo, em pessoas com Alzheimer, doença que se caracteriza principalmente pela deterioração cognitiva e da memória, o monitoramento automático poderia identificar se a pessoa monitorada esqueceu de realizar alguma ADL importante para manutenção da sua saúde, tais como tomar banho ou alimentar-se. Em casos como este, um sistema automático poderia alertar o indivíduo e sua família sobre o ocorrido. Já para Doenças Reumáticas e

Parkinson um sistema automático de detecção de ADLs poderia identificar se o indivíduo está se movimentando com menos frequência, permanecendo muito tempo deitado ou sentado, o que poderia indicar progresso na evolução da doença, ou até mesmo o surgimento de outras enfermidades, como depressão por exemplo.

2.2 Trabalhos Relacionados

O reconhecimento de atividades diárias vem ganhando muita visibilidade nos últimos anos. Esse tema é amplamente estudado devido ao seu potencial para melhorar a qualidade de vida de indivíduos que necessitam de cuidados especiais.

Um dos primeiros trabalhos encontrados que aborda o problema de reconhecimento de ADLs é o descrito em Yamato et al. (1992). O referido artigo utiliza sequências de vídeo para tentar identificar ações humanas. Os experimentos foram executados para identificar 10 ações de três jogadores de tênis. Para realizar esse trabalho os autores utilizaram o método de Modelos Ocultos de Markov (HMM), chegando a uma acurácia de 90%. Em anos seguintes outros trabalhos como Starner et al. (1998) e Wilson & Bobick (1999) utilizaram técnicas semelhantes para identificar linguagem de sinais e gestos com as mãos. Soluções que utilizam vídeo para identificar ações humanas possuem dois principais desafios: o primeiro é a necessidade de um alto poder computacional para realizar o processamento das imagens, o segundo desafio se dá na manutenção da privacidade dos indivíduos monitorados, nesse ultimo, a segurança da informação tem um papel essencial.

Com o aprimoramento e miniaturização dos dispositivos eletrônicos, começaram a surgir trabalhos que utilizam sensores instalados no ambiente ou no próprio indivíduo. Tapia et al. (2004) utilizam sensores espalhados pelos cômodos de uma casa com o objetivo de monitorar as atividades realizadas pelos seus residentes. Nesse trabalho são utilizados pequenos sensores sem fio de baixo custo instalados em diversos itens na casa (portas, janelas, gavetas, forno microondas, geladeira, fogão, chuveiro, interruptores de luz, etc.), totalizando 77 sensores. Esse trabalho apresentou uma acurácia máxima de 89% para algumas ADLs.

Já em Casale et al. (2011*a*) é utilizado um dispositivo sem fio instalado no corpo dos voluntários para medir a aceleração do corpo nos três eixos (x, y e z), e identificar 5 atividades (caminhar, subir escadas, conversar com pessoas, ficar em pé e trabalhar no computador) de 15 indivíduos, obtendo uma acurácia de 94% utilizando a técnica Random Forest. Para melhorar os resultados foram aplicadas algumas técnicas de pré-processamento (filtro de alta e baixa frequência, windowing com overlapping e seleção das características mais relevantes). No mesmo ano também foi publicado o trabalho de Sagha et al. (2011) que utiliza dados de acelerômetro, giroscópio e sensores instalados no ambiente de teste, para identificar 4 atividades diárias (Ficar em pé, sentar, deitar e andar), utilizando as técnicas Nearest Centroid Classifier (NCC), Linear Discriminant Analysis (LDA) e Quadratic Discriminant Analysis

(QDA), esse trabalho não obteve resultados tão promissores quanto o anterior, alcançando uma acurácia máxima de 86%.

Já o trabalho de Anguita et al. (2012) utiliza acelerômetro e giroscópio de smartphones para tentar identificar as ações: andar, subir escada, descer escada, ficar em pé, sentar e deitar. Neste trabalho é utilizada a técnica Support Vector Machine (SVM) e foca na redução do custo computacional, já que todo o processamento é executado em um smartphone, dispositivo que possui baixo poder de computação. O estudo conseguiu reduzir o custo computacional do seu algoritmo de classificação, mantendo uma acurácia de 89,3%, chegando a 100% para a atividade deitar.

Após 2012 foram publicados os trabalho de Barshan & Yüksek (2014) e Bruno et al. (2013), ambos utilizando acelerômetros e giroscópios para coletar os dados. O primeiro possui na sua base de dados 19 atividades (sentar ficar em pé, deitar para trás, subir e descer escada, em pé no elevador, andar no elevador, caminhar no parque, caminhar na esteira, correr na esteira, andar de bicicleta, etc.), e utiliza diversos métodos de aprendizado supervisionado para verificar qual apresenta melhores resultados, são eles: Naive Bayes, Redes Neurais, J48, Random Forest, NB Tree e SVM, alcançando uma acurácia de 99,2% tanto utilizando Redes Neurais quando o SVM. O segundo utiliza técnicas de aprendizado supervisionado para classificar 14 atividades (escovar os dentes, pentear cabelo, subir escadas, descer escadas, andar, beber água, colocar água no copo, comer com garfo e faca, comer com colher, usar telefone, levantar da cama, deitar na cama, levantar da cadeira, sentar na cadeira). Para fazer o agrupamento das atividades são utilizadas dois atributos extraídos de cada amostra, distância de Mahalanobis e Dynamic Time Warping. Os resultados desse ultimo trabalho tiveram uma grande oscilação a depender da atividade classificada ficando entre 0% para a atividade sentar na cadeira, e 100% para a atividade beber água.

Em Khan et al. (2010) é realizado um estudo sobre a detecção de ADLs utilizando dados gerados por acelerômetro de um smartphone. No total 5 Atividades são classificadas (repouso, caminhar, subir escadas, descer escadas e correr), foi alcançada uma acurácia média de 96% utilizando Análise Discriminante de Kernel para o cálculo das características relevantes e Redes Neurais Artificiais para a classificação.

Banos et al. (2012) propõem método para extração de características relevantes de ADLs, para isso são calculadas 861 características e depois é aplicado um filtro baseado na discriminação e robustez do conjunto de dados para selecionar as mais relevantes. Para realizar a classificação são utilizados os métodos Decision Tree, Naive Bayes e SVM, a acurácia alcançada foi de 95% utilizando o SVM.

Chetty & White (2016) utilizam uma rede com diversos sensores sem fio para identificar 6 atividades (caminhar, subir escada, descer escada, sentar, ficar em pé e deitar), foi alcançado a acurácia máxima de 96% utilizando o Random Forest, para verificar o melhor modelo foram utilizados 6 métodos (Naive Bayes, K-Means, J48, Random Forest, Random Committee, Lazy IBk).

Trabalhos como os de Mahmoud et al. (2012) e Mahmoud et al. (2016), propõem técnicas para detecção de anomalia na execução de atividades diárias utilizando Lógica Fuzzy e distância de Hamming. A base de dados utilizada é formada por registros de sensores binários instalados em diversos pontos de uma casa (portas, torneiras, eletrodomésticos, etc).

A detecção automática de ADLs vem sendo amplamente estudada por diversos grupos de pesquisa, vários trabalhos apresentam uma boa acurácia nos modelos preditivos, como é o caso de Barshan & Yüksek (2014), Khan et al. (2010), Banos et al. (2012), Weiss et al. (2016) e Chetty & White (2016).

Os trabalhos de Khan et al. (2010), Banos et al. (2012) e Weiss et al. (2016), citados acima, não apresentam a matriz de confusão para que seja possível identificar o comportamento das classificações, bem como não é apresentado o gráfico de distribuição das ADLs utilizadas, ou seja, não é possível saber se a base está balanceada. Esse problema pode comprometer completamente as acurácias apresentadas já que, em um banco altamente desbalanceado, se o classificador sempre classificar as novas entradas como a classe mais comum dos dados, ele ainda terá uma grande taxa de precisão. Dessa forma não é possível afirmar que os resultados apresentados nesses trabalhos refletem de fato o desempenho real.

Já em Barshan & Yüksek (2014) e Chetty & White (2016), também citado acima, verifica-se que as bases utilizadas estão balanceadas porém os ambientes utilizados para coletar os dados são formados por redes de sensores, o que torna sua implantação em um ambiente real improvável. As Figuras 2.1, 2.2 e 2.3 apresentam um exemplo de como os dados foram coletados nesses trabalhos. Nestas figuras verificamos os diversos sensores utilizados na captura dos dados.

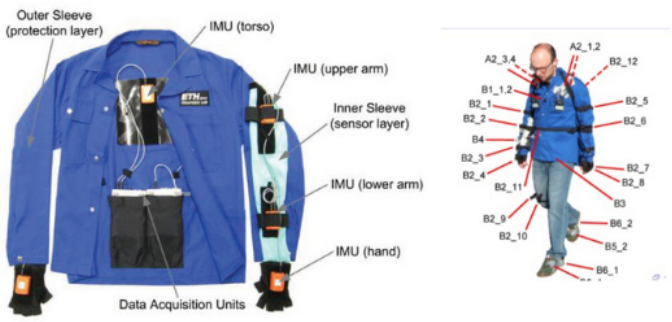


Figura 2.1: Rede de sensores utilizada em Chetty & White (2016), com acelerômetros espalhados pela jaqueta, calça e sapatos.

Neste cenário, o presente trabalho tem o objetivo de propor uma metodologia para avaliar um grupo de métodos de aprendizado de máquina (k-Nearest Neighbors, Naive Bayes, Support Vector Machine, Decision Tree, Random Forest, Extra-Trees e Perceptron multicamadas),

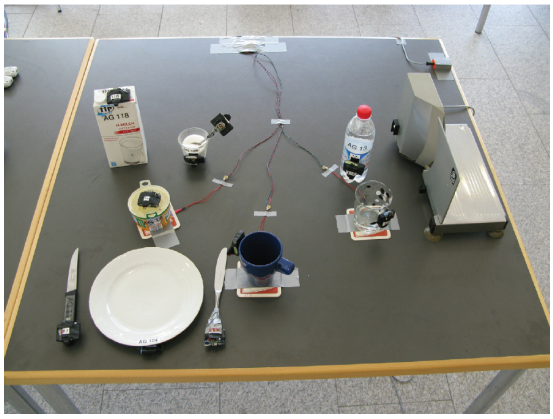


Figura 2.2: Rede de sensores utilizada em Chetty & White (2016), acelerômetros nos objetos utilizados.

Figura 2.3: Rede de sensores utilizada em Barshan & Yüksek (2014), total de 5 sensores acelerômetro/giroscópio.

com o objetivo de alcançar os melhores resultados na classificação automática de Atividades Diárias, com foco em diminuir a complexidade de implantação e execução, utilizando mecanismos que corroborem para a generalização dos resultados.

2.3 Abordagens Baseadas em Aprendizado de Máquina

Para resolver computacionalmente um determinado problema, é possível programar de forma direta, estabelecendo diretrizes e funções de entrada e saída, ou seja, codificar diretamente as regras do modelo desejado. Porém, codificar diretamente sistemas complexos, que necessitam de adaptabilidade e generalização, é uma tarefa muito difícil, ou até mesmo impossível. Uma boa alternativa é utilizar técnicas e ferramentas de Aprendizado de Máquina (do inglês, *Machine Learning*), que possibilita a criação de modelos sem que esses sejam explicitamente programados.

As técnicas de Aprendizado de Máquina utilizam dados em forma de exemplo para construir modelos analíticos, segundo Russell & Norvig (2009) Aprendizado de Máquina é uma subárea da inteligência artificial, que tem a capacidade de detectar e extrapolar padrões,

além de se adaptar a novas circunstâncias. Essas técnicas são utilizadas em diversos cenários, tais como: detecção de fraudes, sistemas de recomendação, mecanismos de busca, sistemas de vigilância em vídeo, reconhecimento de manuscrito, processamento de linguagem natural, segurança de TI, *bots* de serviço de cliente, detecção de anomalias, logística, veículos autônomos etc.

Existem duas principais formas de aprendizagem, a supervisionada e a não supervisionada, cujo as definições segundo Russell & Norvig (2009) seguem abaixo:

- **Supervisionada:** são utilizados exemplos de entrada e saída para modelar uma função que faz o mapeamento dessas entradas e saídas, ou seja, dado um conjunto de N pares de exemplos de entrada e saída $[(x_1, y_1), (x_1, y_1)], ..., (x_n, y_n)$, onde cada y_j foi gerado por uma função desconhecida $y = f(x)$, descobrir uma função h que se aproxime da função verdadeira f.

- **Não Supervisionada:** O modelo criado tem o objetivo de aprender padrões na entrada, embora não seja fornecido nenhum feedbak explícito. A tarefa mais comun de aprendizagem não supervisionada é o agrupamento: a detecção de grupos de exemplos potencialmente úteis.

Nesse trabalho são utilizados dados rotulados, ou seja, dados com exemplos de entrada e saída, e nesse cenário a abordagem de aprendizado supervisionado é a mais indicada.

No aprendizado supervisionado os dados de entrada são divididos em dois subconjuntos, o de treinamento e o de teste. O conjunto de treinamento é utilizado para aproximar o modelo da verdadeira função que gera as saídas de um determinado problema. Já o conjunto de teste é utilizado para verificar o desempenho do modelo criado. As estratégias chamadas de Holdout e Validação Cruzada são as principais utilizadas na literatura para dividir esses subconjuntos.

No método Holdout os dados são particionados em dois subconjuntos mutualmente exclusivos, onde normalmente o conjunto de treinamento possui 2/3 dos dados, e o de teste 1/3. Esse método de validação utiliza apenas uma parte dos dados para treinamento, o que pode comprometer a confiabilidade dos resultados, já que o conjunto escolhido pode ter a sorte, ou o azar, de conter os dados mais relevantes, ou não, para a construção do modelo (Kohavi, 1995).

Já na Validação Cruzada (*Cross-Validation*, em inglês), o conjunto de dados D é aleatoriamente dividido em k subconjuntos mutualmente exclusivos $(D_1, D_2, D_3, ..., D_k)$, de tamanhos aproximadamente iguais. O modelo é treinado e testado k vezes e, em cada ciclo $t \in \{1, 2, 3, ..., k\}$, o treinamento em $D \backslash D_t$ é realizado e testado com D_t. Na Validação Cruzada o resultado estimado é a média de todas as iterações (Kohavi, 1995). Esse método é amplamente utilizada na validação de modelos em trabalhos que utilizam aprendizado de máquina. Um dos seus problemas é que os subconjuntos são formados aleatoriamente sem levar em consideração sua representatividade, ou seja, não se sabe se uma determinada classe está em todos

os subconjuntos D_k, isso pode interferir tanto no treinamento quando no teste do modelo. Para suprir essa deficiência temos uma variação da Validação Cruzada, chamada de Validação Cruzada Estratificada (*Stratified Cross-Validation*, em inglês), onde todos os subconjuntos D_k contém aproximadamente a mesma proporção de classes do conjunto original D (Kohavi, 1995). Nesse trabalho utilizamos a estratégia da Validação Cruzada Estratificada.

Para avaliar o desempenho dos classificadores serão utilizadas 4 métricas amplamente utilizadas na literatura:

- **Acurácia:** é a probabilidade de classificar corretamente uma instância aleatória i, ex.: $AC = Pr(C(i) = y)$, para uma instância aleatória $(i, y) \in X$ (Russell & Norvig, 2009).
- **Precisão:** é o número de classificações corretas, dividido pelo total de classificações de uma determinada classe (Zhang & Zhang, 2009).
- **Revocação:** é o número de classificações corretas, dividido pelo total de ocorrências de uma determinada classe (Zhang & Zhang, 2009).
- **F-Score:** é a média ponderada entre precisão e revocação, representa a distorção entre a esses dois indicadores. Pode-se interpretar como uma medida de confiabilidade da acurácia (Zhang & Zhang, 2009).

2.4 Bases de Dados

Para realizar os estudos deste trabalho é necessário uma base de dados que contenha os registros dos movimentos de indivíduos e as atividades que estão sendo executadas naquele momento. Como a literatura já possui diversas bases com essas características, optamos por utilizar bases já existentes. Para isso, utilizamos o site da UCI (UC Irvine Machine Learning Repository) (Dua & Graff, 2017), uma coleção de bancos de dados mantida pela Universidade da Califórnia em Irvine, voltada especificamente para comunidade de estudiosos e pesquisadores da área de aprendizado de máquina.

Foi realizado uma pesquisa por bases que possuíssem o registro de ADL's na seção "View ALL Data Sets", e na filtragem por bases para classificação, foram encontradas o total de 12 bases relacionadas, das quais 3 (três) se destacaram por apresentar uma boa quantidade de ADL's, registros para mais de um indivíduo, boa documentação, e dados formados por acelerômetro. As três bases selecionadas foram: UMAFall (A Multisensor Dataset for the Research on Automatic Fall Detection) (Casilaria et al., 2017), ARCMA (Activity Recognition from a Single Chest-Mounted Accelerometer) (Casale et al., 2011*b*) e HMP (Public Dataset of Accelerometer Data for Human Motion Primitives Detection) (Bruno et al., 2013). Todas as bases possuem uma série temporal com o registro dos dados de acelerômetro e a atividade executada. As Figuras 2.4, 2.5 e 2.6 contêm amostra dos dados dessas bases, onde a abcissa

representa o valor retornado pelo sensor e a ordenada o tempo. Em cada linha temos os valores para um eixo, X, Y e Z. A seguir serão apresentadas as principais características de cada bases.

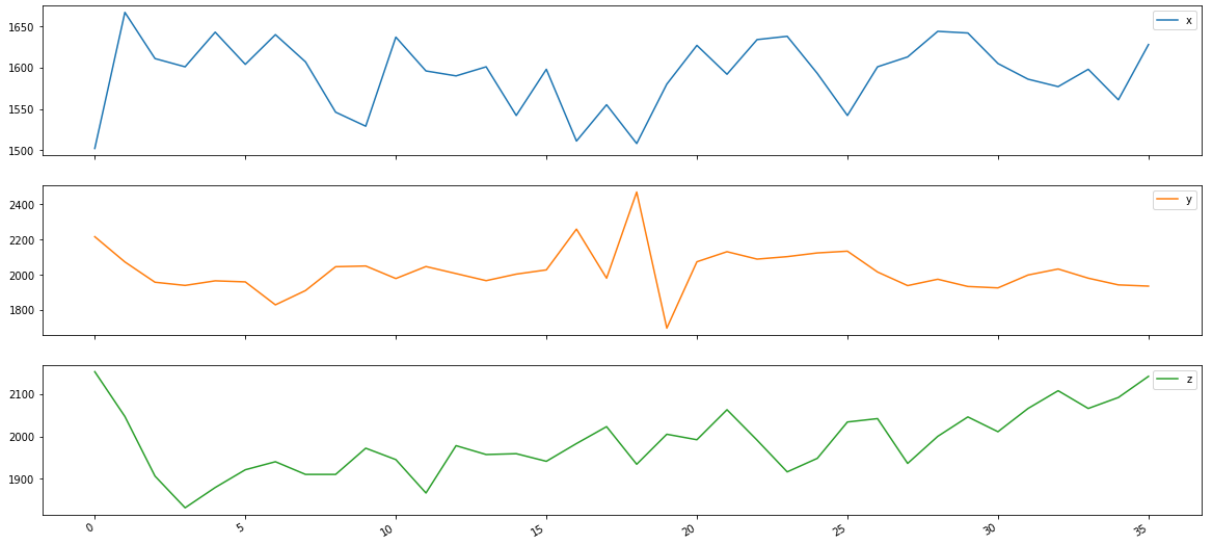


Figura 2.4: Amostragem para a atividade *Working at Computer* da base ARCMA.

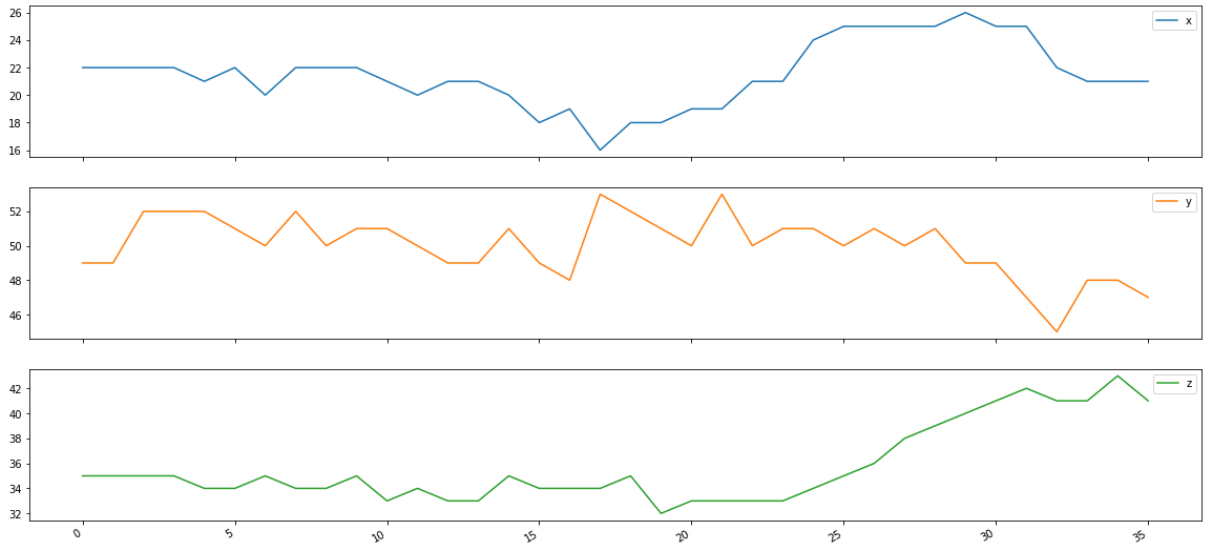


Figura 2.5: Amostragem para a atividade *Brush Teeth* da base HMP.

2.4.1 ARCMA (Activity Recognition from a Single Chest-Mounted Accelerometer)

A base de dados ARCMA (Casale et al., 2011*b*), possui dados de acelerômetro de atividades de 15 voluntários. Para fazer a coleta foi utilizado um dispositivo de baixo custo que possui

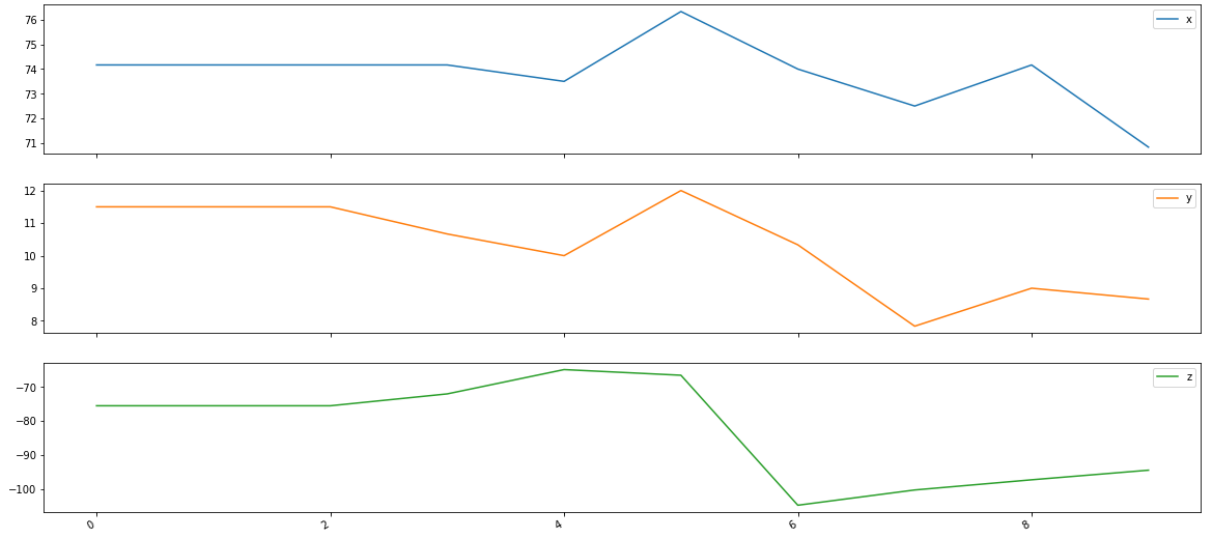


Figura 2.6: Amostragem para a atividade *Bending* da base UMAFall

autonomia de bateria de 4 horas, conexão *Bluetooth Low Energy* (para o envio dos dados) e uma taxa de amostragem de 52Hz. Esse dispositivo foi instalado no tórax dos voluntários, que executaram 7 atividades: "trabalhando no computador", "em pé andando e subindo escadas", "em pé", "caminhando", "subindo e descendo escadas", "caminhando e conversando com alguém", e "falando em pé". Embora a documentação da base ARCMA descreva 7 atividades, verificamos que seus registros possuem 8, dessa forma, a atividade 8 não tem um rótulo definido, dessa forma, não foi possível definir que tipo de atividade seria.

O grupo de voluntários foi formado com idades entre 27 e 35 anos. No processo de coleta, foi solicitado aos participantes, que informassem ao sistema qual atividade seria executada antes de começar o registro dos dados.

A base ARCMA consiste em 14 arquivos no formado CSV, onde cada linha contém um ID sequencial, aceleração em X, aceleração em Y, aceleração em Z e a atividade executada. Cada arquivo esta nomeado com o identificador do voluntário que executou as atividades. O Quadro 2.1 descreve as atividades diárias registradas por essa base e o total de registros para cada ADL. O Quadro 2.2 contém a quantidade de registros para cada voluntário por atividade.

2.4.2 UMAFall (A Multisensor Dataset for the Research on Automatic Fall Detection)

O conjunto de dados UMAFall (Casilaria et al., 2017) contém registros de uma rede de sensores instalada no corpo de voluntários e utiliza três tipo de sensores: acelerômetro, giroscópio e magnetômetro. Esta base de dados possui registro de oito tipo de ADL's, são elas: "agachamento", "pequenos saltos", "corrida leve", "deitando e se levantando de uma cama", "sentando e levantando de uma cadeira", "caminhada normal", "descendo escadas"e

ID da ADL	Descrição	Qtd. de Registros
1	Trabalhando no computador	608.667
2	Em pé, andando e subindo escadas	47.878
3	Em pé	216.737
4	Caminhando	357.064
5	Subindo e descendo escadas	51.498
6	Caminhando e conversando com alguém	47.770
7	Falando em pé	593.563
0	Atividade Indefinida	3.719
	Total de Registros	**1.926.896**

Quadro 2.1: Descrição e Quantidade de Registros das ADLs da Base ARCMA

Voluntário\Atividade	0	1	2	3	4	5	6	7
1	1	33.677	928	11.179	26.860	3.191	2.917	83.748
2	1	44.050	3.435	23.596	22.149	3.890	7.449	22.231
3	1	54.170	2.330	8.775	17.750	3.225	1.049	17.151
4	1	48.750	4.340	11.730	30.130	3.650	1.200	14.901
5	1	18.280	1.650	8.220	17.650	3.400	1.201	17.249
6	1	52.875	505	12.785	29.315	3.620	1.500	15.500
7	1	51.600	4.680	7.910	17.510	3.300	1.000	17.500
8	270	44.150	3.490	23.475	22.175	3.910	7.100	33.431
9	1	41.675	5.475	4.293	25.900	2.657	1.400	20.940
10	1	31.540	4.810	21.405	23.300	3.745	1.910	35.490
11	1	30.980	5.370	13.420	26.750	30.00	2.880	77.600
12	231	44.040	3.460	23.495	22.040	3.935	6.989	36.711
13	1	32.750	3.600	10.519	26.770	2.960	2.700	83.701
14	206	44.040	3.485	24.605	22.035	3.900	5.980	33.750
15	3.001	36.090	320	11.330	26.730	3.115	2.495	83.660

Quadro 2.2: Registros por Voluntário por Atividade da Base ARCMA

"subindo escadas". Além das ADL's o UMAFall também possui três tipos de queda: "queda para frente", "queda para traz"e "queda para o lado". Os dados foram coletados de 17 voluntários, 10 homens e 7 mulheres, entre 14 e 55 anos.

A rede utilizada consiste de um smartphone android colocado no bolso direito dos voluntários, usado como nó central, e quatro dispositivos sem fio posicionados em diferentes partes do corpo: tórax, pulso, tornozelo e cintura, como mostra a Figura 2.7.

Durante a coleta dos dados foram utilizados dois modelos de smartphones, Samsung S5 e LG G4, ambos integrados com acelerômetro e conectividade Bluetooth. Cada dispositivo sem fio consiste em um SensorTag CC2650 Multi-Standard da Texas Instruments, que possui interface Bluetooth Low Energy, acelerômetro tri-axial, giroscópio tri-axial e magnetômetro. Cada dispositivo utiliza uma bateria de lítio CR2032 3v e possui uma taxa de amostragem de 20Hz.

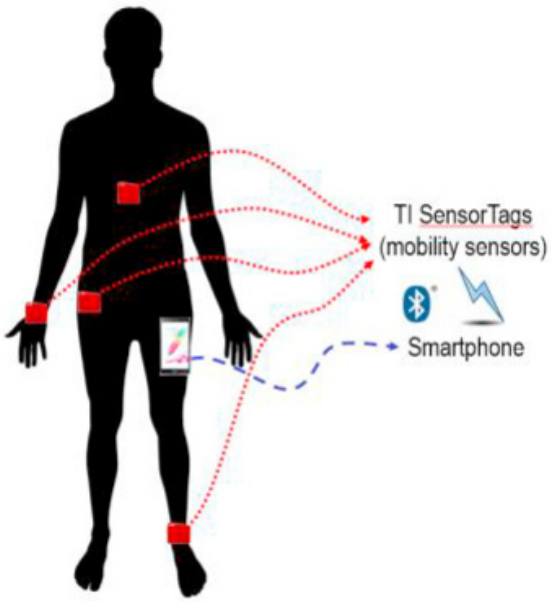


Figura 2.7: Disposição dos sensores no corpo dos voluntários. Fonte: Casilaria et al. (2017)

Originalmente o UMAFall possui 531 arquivos CSV, onde cada um é nomeado de forma a identificar o voluntário, a atividade e a data. Dentro de cada arquivo encontram-se os dados, *timestamp*, número da amostra, eixo X, eixo Y, eixo Z, tipo do sensor (acelerômetro, giroscópio e magnetômetro) e local do sensor (bolso, cintura, tornozelo, pulso e tórax).

Para todos os testes com o UMAFall foram utilizados os dados gerados pelo sensor magnetômetro instalado na cintura dos voluntários, essa escolha se deu com base nos resultados obtidos em (Machado et al., 2018), que apresentou melhor acurácia para classificação de ADL's utilizando essa configuração. O Quadro 2.3 descreve as atividades diárias registradas por essa base e o total de registros para cada ADL. O Quadro 2.4 contém a quantidade de registros para cada voluntário por atividade.

ADL\Voluntários	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Bending	19717	19703	26274	19703	19713	6567	16807	19730	19696
Hopping	19719	19704	19708	19711	19703	0	17094	17158	19712
Jogging	19712	19713	19711	19717	0	0	0	0	19704
LyingDown	19733	19703	19693	32843	19727	0	22152	17100	19707
Sitting	19715	19722	19696	19715	19717	13130	16776	17141	19111
Walking	19700	19710	19732	26283	19724	6575	16876	16894	19711
GoDownstairs	0	0	0	0	0	0	17123	11393	0
GoUpstairs	0	0	0	0	0	0	16973	11450	0
forwardFall	52571	38393	39402	32638	0	13112	0	0	39044
backwardFall	39425	19713	39406	39439	39414	13153	0	0	39443

lateralFall	39421	19685	38881	39446	0	13146	0	0	39433
Atividades\ Voluntários	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	
Bending	17156	22868	25925	6569	13122	6558	51538	16905	
Hopping	16852	17089	19415	6571	14	13126	45681	16961	
Jogging	0	0	0	6565	0	6566	45301	16778	
LyingDown	16919	16968	19656	6562	0	13143	51472	16554	
Sitting	16962	17153	18878	6573	13139	13147	51561	16901	
Walking	16850	16961	19215	13108	6562	13130	45839	16995	
GoDownstairs	17155	11308	26290	0	0	0	34078	0	
GoUpstairs	17073	11460	19149	0	0	0	39679	0	
forwardFall	0	0	19467	26141	19054	19667	115811	34029	
backwardFall	0	0	19598	26265	13131	26257	111259	34050	
lateralFall	0	0	18969	26293	6562	25698	98214	33833	

Quadro 2.4: Registros por Voluntário por Atividade da Base UMAFall

2.4.3 HMP (Public Dataset of Accelerometer Data for Human Motion Primitives Detection)

O HMP (Bruno et al., 2013) é uma coleção pública de dados coletados a partir de um acelerômetro tri-axial fixado no pulso direito dos voluntários, que possui transmissão via cabo USB. Possui o registro de 14 ADL's para 16 voluntários, e tem uma taxa de amostragem de 32Hz. As atividades rotuladas são: "escovando os dentes com uma escova de dentes", "subir vários degraus de uma escada", "pentear o cabelo com uma escova", "descer vários degraus de uma escada", "pegar um copo, beber e colocar de volta na mesa", "comer usando garfo e faca", "comer algo usando uma colher", "levantar de uma cama", "deitar em uma cama", "derramar um liquido de um copo em uma mesa", "sentar em uma cadeira", "levantar-se de uma cadeira", "fazer uma chamada telefônica utilizando um aparelho fixo", "dar vários passos". Os dados estão em formato *txt*, nomeados com o formato: [tempo]-[atividade]-[voluntario].txt, ex: "*Accelerometer-2011-03-24-10-24-39-climb_stairs-f1.txt*". Os dados são formados pelo registro de atividades de 11 homens e 5 mulheres, entre 19 e 81 anos.A A Quadro 2.5 descreve as atividades diárias registradas por essa base e o total de registros para cada ADL. A Quadro 2.6 contém a quantidade de registros para cada voluntário por atividade.

Atividades\Voluntários	**f1**	**f2**	**f3**	**f4**	**f5**	**m1**	**m2**	**m3**
brush_teeth	23609	0	0	0	0	3199	3021	0
climb_stairs	17166	1313	1876	0	0	2567	3932	967

comb_hair	15175	1282	0	3117	0	3211	719	0
descend_stairs	12961	0	0	0	0	1294	1120	0
drink_glass	19563	5983	1695	3890	0	7256	2547	4882
eat_meat	31236	0	0	0	0	0	0	0
eat_soup	6683	0	0	0	0	0	0	0
getup_bed	7788	3908	5603	7896	4184	4135	4668	3287
liedown_bed	6882	0	0	1353	0	2133	1078	0
pour_water	15295	4239	1456	11085	0	4733	408	5447
sitdown_chair	12876	697	445	5868	0	3010	3200	542
standup_chair	12109	1522	344	5629	0	3276	3295	1135
use_telephone	12758	0	0	0	0	1082	1385	0
walk	29244	1854	3051	0	0	15757	6978	2798
Atividades\Voluntários	**m4**	**m5**	**m6**	**m7**	**m8**	**m9**	**m10**	**m11**
brush_teeth	0	0	0	0	0	0	0	0
climb_stairs	1748	5829	6026	3834	0	0	0	0
comb_hair	0	0	0	0	0	0	0	0
descend_stairs	0	0	0	0	0	0	0	0
drink_glass	724	0	0	0	646	1006	0	0
eat_meat	0	0	0	0	0	0	0	0
eat_soup	0	0	0	0	0	0	0	0
getup_bed	5718	0	0	0	0	0	3814	0
liedown_bed	0	0	0	0	0	0	0	0
pour_water	1869	0	0	0	0	1120	0	3321
sitdown_chair	305	0	0	0	332	861	0	0
standup_chair	197	0	0	0	366	904	0	0
use_telephone	0	0	0	0	0	0	0	0
walk	3056	11302	10906	10708	0	0	0	0

Quadro 2.6: Registros por Voluntário por Atividade da Base HMP

2.5 Caracterização dos Dados e Formação dos Atributos

As bases mencionadas na Seção 2.4 contêm valores obtidos pela utilização de sensores do tipo acelerômetro, com exceção do UMAFall, que também possui dados para outros tipos de sensores (magnetômetro e Gisroscópio).

Basicamente, os acelerômetros convertem movimento mecânico em sinais elétricos, São dispositivos eletromecânicos que medem a força de aceleração causada por gravidade ou

ID da ADL	Descrição	
Bending	Agachamento	328.551
Hopping	Pequenos saltos	288.204
Jogging	Corrida leve	173.767
LyingDown	Deitando e se levantando de uma cama	311.932
Sitting	Sentando e levantando de uma cadeira	319.037
Walking	Caminhada normal	313.865
GoDownstairs	Descendo escadas	117.347
GoUpstairs	Subindo escadas	115.784
forwardFall	Queda para frente	449.329
backwardFall	Queda para traz	460.553
lateralFall	Queda para o lado	399.581
	Total de Registros	**3.277.950**

Quadro 2.3: Descrição das ADLs da Base UMAFall

movimento. Existem diversos tipos deferentes de acelerômetros, temos os sensores piezoelétricos, piezoresistivos, capacitivos, efeito Hall, magnetoresistivos e de temperatura. Os piezoelétricos, piezoresistivos e capacitivos são os mais comuns em dispositivos comerciais (Minyoung, 2015). A Tabela 2.7 apresenta cada tipo de acelerômetro.

O sensor magnetômetro, utilizado pela base UMAFall, utiliza as propriedades de um sensor de efeito Hall para detecta o campo magnético da terra ao longo de três eixos (x, y e z). Esse sensor produz tensão proporcional a força da polaridade do campo magnético ao longo de cada eixo. A tensão detectada é convertida em sinal digital, que pode ser utilizado para diversas aplicações.

Os avanços tecnológicos, que proporcionam baixo consumo de energia e miniaturização dos componentes, vem tornando esses dispositivos cada vez mais acessíveis para diversos tipos de aplicações, aumentando a precisão e permitindo a detecção de movimentos complexos. A Figura 2.8 mostra um exemplo de miniaturização desses dispositivos, e a Figura 2.9 apresenta os eixos de leitura x, y e z dos acelerômetros.

2.5.1 Seleção dos Atributos

Definir o conjunto de atributos que será utilizado na criação de um modelo preditivo é um dos passos mais importantes para o sucesso de uma estratégia de Aprendizado de Máquina. Neste trabalho, a ferramenta TSFRESH (Christ et al., 2018) foi utilizada para encontrar o conjunto de atributos que melhor descreve as ADLs utilizadas.

O TSFRESH[1] é um framework feito em Python que calcula diversas características de séries

[1]Os detalhes de todos os atributos utilizados pelo TSFRESH podem ser consultados em: https://tsfresh.readthedocs.io/en/latest/api/tsfresh.feature_extraction.html.

ID da ADL	Descrição	Qtd. de Registros
brush_teeth	Escovando os dentes com uma escova de dentes.	29.829
climb_stairs	Subir vários degraus de uma escada.	45.258
comb_hair	Pentear o cabelo com uma escova.	23.504
descend_stairs	Descer vários degraus de uma escada.	15.375
drink_glass	Pegar um copo, beber e colocar de volta na mesa.	48.192
eat_meat	Comer usando garfo e faca.	31.236
eat_soup	Comer algo usando uma colher.	6.683
getup_bed	Levantar de uma cama.	51.001
liedown_bed	Deitar em uma cama	11.446
pour_water	Derramar um liquido de um como em uma mesa	48.973
sitdown_chair	Sentar em uma cadeira	28.136
standup_chair	Levantar-se de uma cadeira	28.777
use_telephone	Fazer uma chamada telefônica utilizando um aparelho fixo.	15.225
walk	Dar vários passos.	95.654
	Total de Registros	**479.289**

Quadro 2.5: Descrição e Quantidade de Registros das ADLs da Base HMP

temporais, contendo métodos que avaliam a importância dessas características utilizando teste de hipótese. Nesse estudo o TSFRESH tem como parâmetro de entrada as sequências das atividades executadas, e como saída as características relevantes desse conjunto de dados.

O processo de extração e seleção das características é realizado em três etapas:

- **Extração**: são extraídos todos os atributos da série utilizando formulas pré-estabelecida pelo framework, formando vetores de atributos.
- **Teste de significância**: cada vetor de atributo é avaliado individualmente e independentemente em relação a sua significância para classificação, o resultado é um vetor de valor-p, quantificando a importância de cada atributo. Uma característica x é relevante para uma classificação y se x e y não são estatisticamente independentes.
- **Teste múltiplo**: o vetor de valor-p é avaliado conforme os procedimentos estabelecidos em (Benjamini & Yekutieli, 2001), para decidir quais atributos serão mantidos. Para isso, é utilizado o FDR (Taxa de Descoberta Falsa), uma média do erro estatístico acumulado, com base no vetor de valor-p.

2.6 Métodos de Aprendizado de Máquina

Este trabalho utiliza 07 (sete) diferentes métodos de aprendizado de máquina para verificar qual seria o mais adequado na classificação das ADL's. Todos apresentam uma abordagem de aprendizado supervisionado e foram escolhidas com base nos trabalhos relacionados descritos na Seção 2.2, os quais tem o objetivo de representar os principais métodos matemáticos

Tipo de Sensor	Características
Piezoelétrico	Constituído de cristais piezoelétricos que sofrem uma força de aceleração, o que resulta em uma carga elétrica proporcional a essa aceleração.
Piezoresistivo	Sua resistência muda de acordo com a aceleração aplicada.
Capacitivo	Sua capacitância muda de acordo com a aceleração aplicada.
Acelerômetro de Efeito Hall	Utilizando as propriedades do efeito Hall, o sinal de aceleração é formado com a alteração do campo magnético.
Magnetoresistivo	A resistência do sensor muda de acordo com a presença de campo magnético.
Temperatura	O sinal de aceleração é obtido por um conjunto de fonte de calor e termoresistores.

Quadro 2.7: Tipos de acelerômetros

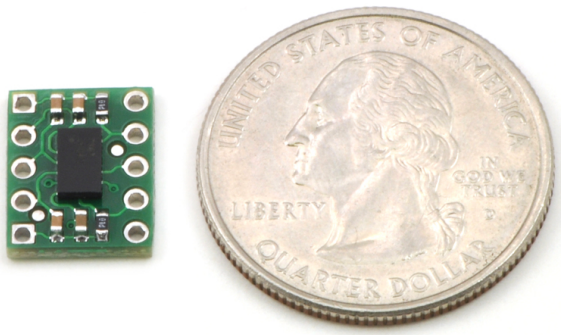


Figura 2.8: Sensor MMA7361LC da Freescale, dimensão de 1 cm × 1.2 cm e consumo de 400 µA.

Fonte: https://www.pololu.com/product/1246

e computacionais aplicados ao reconhecimento automático de atividades. Os métodos utilizados foram: k-Nearest Neighbors, Naive Bayes, Support Vector Machine, Árvore de Decisão, Random Forest, Extra-Trees e Perceptron multicamadas, que são descritos a seguir.

2.6.1 k-Nearest Neighbors

O k-Nearest Neighbors (k-NN) (Cover & Hart, 1967) é um algoritmo de classificação baseado em instâncias, que necessita de uma base de dados rotulada para realizar o treinamento do classificador e identificar novos elementos.

Esse método calcula as distâncias entre um elemento desconhecido e os elementos do conjunto de treinamento, será atribuído ao novo elemento a classe majoritária dos k elementos mais próximos, a Figura 2.10 representa visualmente a ideai.

Vários métodos são utilizados para calcular a distância, sendo o mais utilizado a distância Euclidiana (Mulak & Talhar, 2015) (Equação 2.1). Para realizar a classificação o k-NN percorre

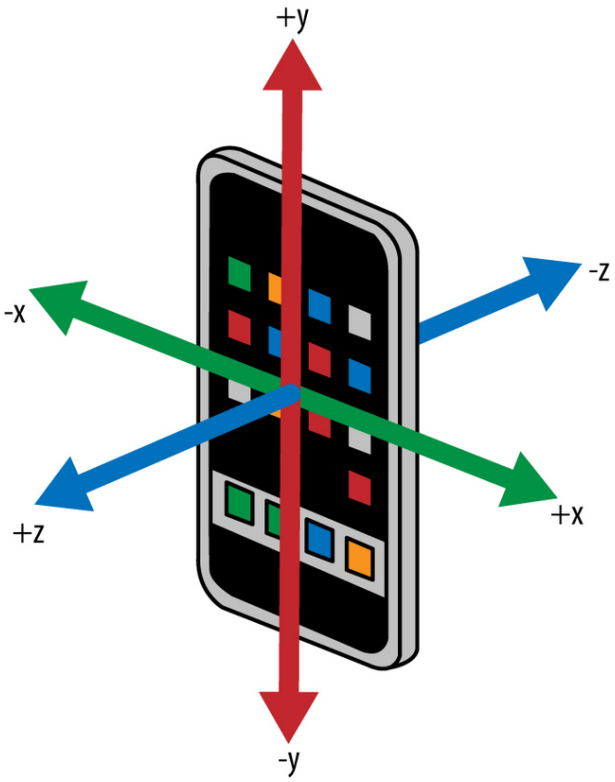


Figura 2.9: Eixos de leitura dos acelerômetros e magnetômetros.

Fonte: https://steemit.com/technology/@mike11/accelerometer-in-a-smartphone-how-it-works

todo o conjunto e calcula a distância d entre x (novo elemento) e as observações no conjunto de treinamento. Após, escolher os k elementos mais próximos à x e formar o conjunto A, estima-se a probabilidade condicional para classe em A utilizando a Equação 2.2.

$$d(x,x') = \sqrt[2]{(x_1 - x_1')^2 + (x_2 - x_2')^2 + ... + (x_n - x_n')^2} \tag{2.1}$$

$$p(y = j|X = x) = \frac{1}{k} \sum_{i \in A} I(y_i = j) \tag{2.2}$$

2.6.2 Naive Bayes

O algoritmo Naive Bayes (Lewis, 1998), utiliza uma abordagem probabilística baseada no Teorema de Bayes,

$$p(c|x) = \frac{p(x|c)p(c)}{p(c)}. \tag{2.3}$$

O método Naive Bayes assume que todas as características são condicionalmente independentes entre si. Ou seja, a presença de uma característica não está relacionada a nenhuma outra, por isso da palavra "Naive (ingênuo)". Um exemplo seria uma fruta classificada como banana por ser alongada, amarela e com peso de 100g. Mesmo que essas características

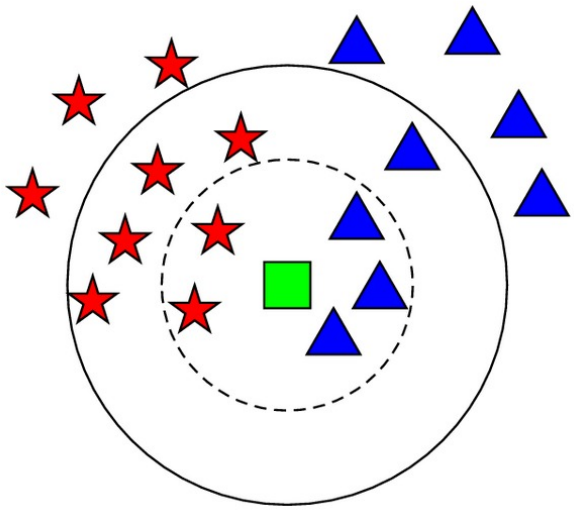


Figura 2.10: Exemplo de classificação do algoritmo k-NN.

Fonte: (Jian et al., 2014)

dependam umas das outras para existirem, elas contribuem de forma independente para a probabilidade do fruto ser uma banana.

Dado um elemento $x = x1,x2,...,xn$ com n atributos, o Naive Bayes prediz a probabilidade de x pertencente a classe C_k, utilizando a probabilidade calculada pelo Teorema de Bayes,

$$(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)} = \frac{p(x_1,x_2,...,x_n|C_k)p(C_k)}{p(x_1,x_2,...,x_n)}, \tag{2.4}$$

onde C é o conjunto com as k classes possíveis. Após calcular as probabilidades de x pertencente a cada uma das k classes, é atribuído a classe C_k com maior probabilidade ao elemento x.

O Naive Bayes tem sido utilizado obtendo bons resultados em diversas situações no mundo real, principalmente em sistemas de recomendação, análise de perfil, classificação de documentos e identificação de spam. Geralmente esse método necessita de uma base de treinamento relativamente pequena, e pode ser extremamente rápido se comparado com métodos mais sofisticados, sendo muito utilizado em problemas de previsão em tempo real.

Um dos pontos fracos do Naive Bayes é justamente sua suposição de independência. Na vida real, é difícil encontrar problemas que sejam completamente independentes.

2.6.3 Support Vector Machine (SVM)

Esse método foi criado por Corinna Cortes e Vladimir Vapnik em (Vapnik & Cortes, 1995). O SVM pode ser utilizado tanto para regressão, quanto para classificação. Para realizar a classificação o algoritmo encontra o hiperplano que melhor separa as classes do conjunto de treinamento, ou seja, que maximiza a distância entre as diferentes classes do conjunto.

A Figura 2.11 demonstra de forma visual o conceito. A Figura 2.11(A) apresenta diversas retas que separam as duas classes, porém, a reta que melhor separa as classes está na Figura 2.11(B). No exemplo ilustrativo da Figura 2.11 o conjunto de treinamento possui apenas duas características (x_1, x_2), e assim, o hiperplano se torna uma reta. Caso o conjunto tenha três características, o hiperplano seria um plano com duas dimensão, e assim por diante.

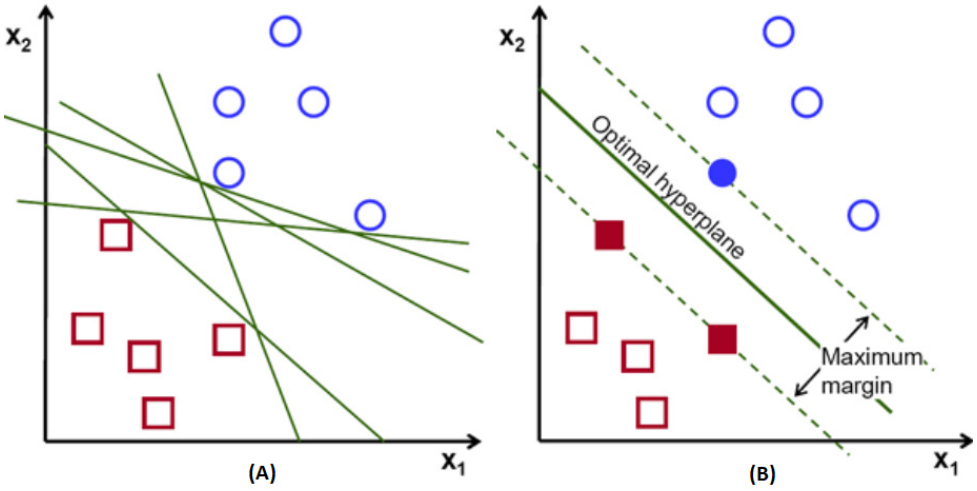


Figura 2.11: Procura do melhor hiperplano pelo SVM.

Fonte: (Drakos, 2012)

Os elementos mais próximos do hiperplano são chamados de vetores de suporte. Encontrar esses vetores é a principal tarefa na fase de treinamento. A classificação dos novos elementos é realizada utilizando apenas os vetores de suporte, após encontrar o(s) hiperplano(s) todo o resto dos dados são descartados, o que faz com que esse método seja bastante leve e eficiente para realizar classificações, já que utiliza apenas alguns vetores de suporte, e não todo o conjunto.

Em conjuntos não separáveis linearmente, o SVM utiliza funções matemáticas chamadas de kernels que transformam esse conjunto em outro linearmente separável. Normalmente são utilizados os kernels: linear, polinomial, radial e sigmoide. Para construir um hiperplano ótimo, o SVM aplica um algoritmo de treinamento iterativo, utilizado para maximizar a margem entre esse hiperplano e os elementos das diferentes classes.

Suponha que temos um conjunto de dados $[(x_1,y_1),(x_2,y_2),...,(x_m,y_m)]$, com classes linearmente separáveis, onde $y_i = -1$ para x_i da Classe 0 e $y_i = 1$ para x_i da Classe 1. A reta que separa essas duas classes pode ser definida como $\vec{w} \cdot \vec{x} + b = 0$, onde $\vec{w}$ e $\vec{x}$ são dois vetores dimensionais. Definimos o vetor negativo $\overrightarrow{x_n}$ em que todas as entradas são da Classe 0, e o vetor positivo $\overrightarrow{x_p}$ onde todas as entradas são da Classe 1. Agora podemos definir os limites inferior e superior da nossa reta: $\vec{w} \cdot \overrightarrow{x+n} + b = -1$ para Classe 0, e $\vec{w} \cdot \overrightarrow{x+n} + b = 1$ para Classe 1. A distância entre esses dois limites é dada por $\frac{2}{||\vec{w}||}$, e a margem M é dada por $\frac{1}{||\vec{w}||}$. Para

maximizar M devemos minimizar $||\vec{w}||$. O SVM utilizara técnicas para resolver problemas de otimização com o objetivo de minimizar $||\vec{w}||$. Esse é um exemplo em um ambiente com duas dimensões, a mesma técnica pode ser utilizada para N dimensões.

O SVM funciona muito bem em conjunto de dados que possuem uma margem de separação clara entre as classes, e tem grande eficiência em espaços com grande dimensionalidade, porém, quando o conjunto de dados é grande, ele pode requerer muito tempo na fase de treinamento.

2.6.4 Árvore de Decisão

A Árvore de Decisão pode ser utilizada para tarefas de classificação de dados categóricos e contínuos.

Segundo (Shalev-Shwartz & Ben-David, 2014), uma Árvore de Decisão é um preditor $h : x \rightarrow y$ que prediz a classe associada a uma instância x percorrendo uma estrutura de dados em árvore, dos seus nós raízes até um nó folha, a Figura 2.12 trás uma representação gráfica do algoritmo.

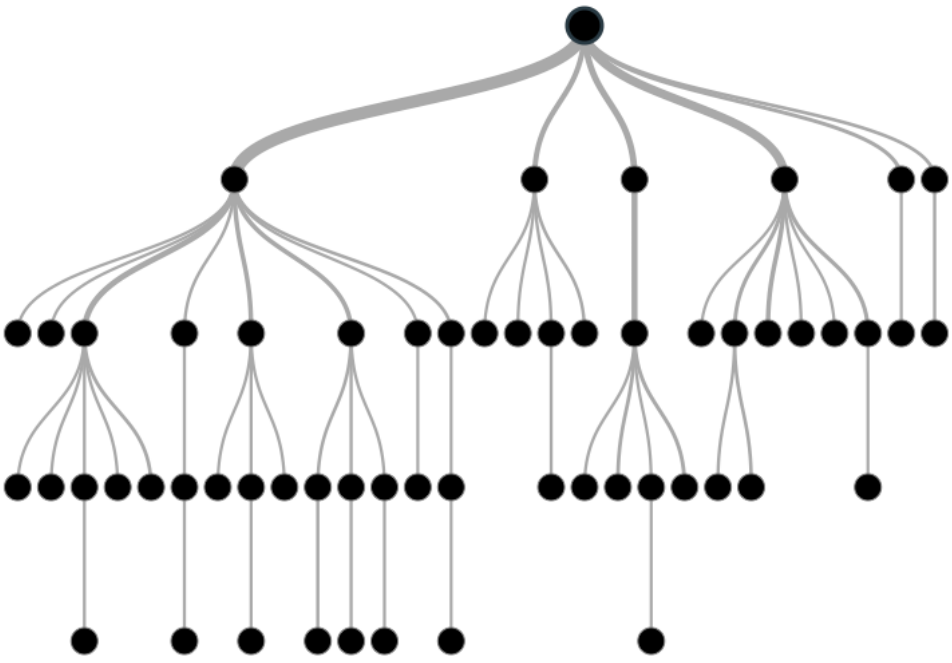


Figura 2.12: Estrutura de uma Árvore de Decisão.

Fonte: (Vidhya, 2012)

Uma Árvore de Decisão é composta por vários nós, as folhas contêm as classes de interesse, e os nós intermediários representam as características que definem as classes. Esse

classificador é de fácil entendimento e interpretação, sendo essa, uma das vantagens em utilizá-lo.

Para obter bons resultados se faz necessário escolher métricas adequadas para selecionar as características que irão formar os nós. Geralmente é utilizado o Ganho de Informação que usa o conceito de entropia,

$$h(x) = \sum_{i=1}^{k} p_i \log_2(p_i), \tag{2.5}$$

onde x é o atributo, k é a quantidade de classes do conjunto de dados e p_i é a probabilidade de cada uma das classes acontecer para um dado atributo. Para o cálculo do Ganho de Informação utilizamos a função,

$$ig(x) = h(classe) - h(x). \tag{2.6}$$

Também precisamos saber quando parar a construção da árvore, pois uma Árvore maior do que o necessário pode levar a overfitting (sobreajuste, onde não é possível a generalização). Uma das formas mais utilizadas de prever essa parada é definir a profundidade máxima do modelo, ou seja, o comprimento do caminho mais longo de uma raiz à uma folha.

2.6.5 Random Forest

O Random Forest (Breiman, 2001), pertence ao grupo de algoritmos denominados de "Ensemble", cujo o principal objetivo é combinar o resultado de um conjunto de classificadores. Para isso, o Random Forest cria diversas Árvores de Decisão, e para cada árvore, k atributos são escolhidos aleatoriamente para formá-la. Após a escolha dos atributos pelo algoritmo, as árvores são construídas da forma tradicional, utilizando o ganho de informação. Sua estrutura permite executar o algoritmo de forma distribuída, tornando-se eficiente para problemas de classificação em grandes bases de dados. A acurácia tende a aumentar de acordo com o número de árvores utilizadas, a Figura 2.13 apresenta de forma gráfica o funcionamento do algoritmo.

2.6.6 Extra-Trees

O algoritmo Extra-Trees (Geurts et al., 2006) é bastante similar ao Random Forest, a diferença entre eles é que o Extra-Trees adiciona mais uma camada de aleatoriedade para montar as árvores, ou seja, o algoritmo utiliza uma estratégia aleatória para montar os nós em vez de utilizar o ganho de informação ou outras métricas. O aumento da aleatoriedade leva a algumas vantagens como: diminuição do viés e melhor desempenho em problemas complexos de alta dimensionalidade.

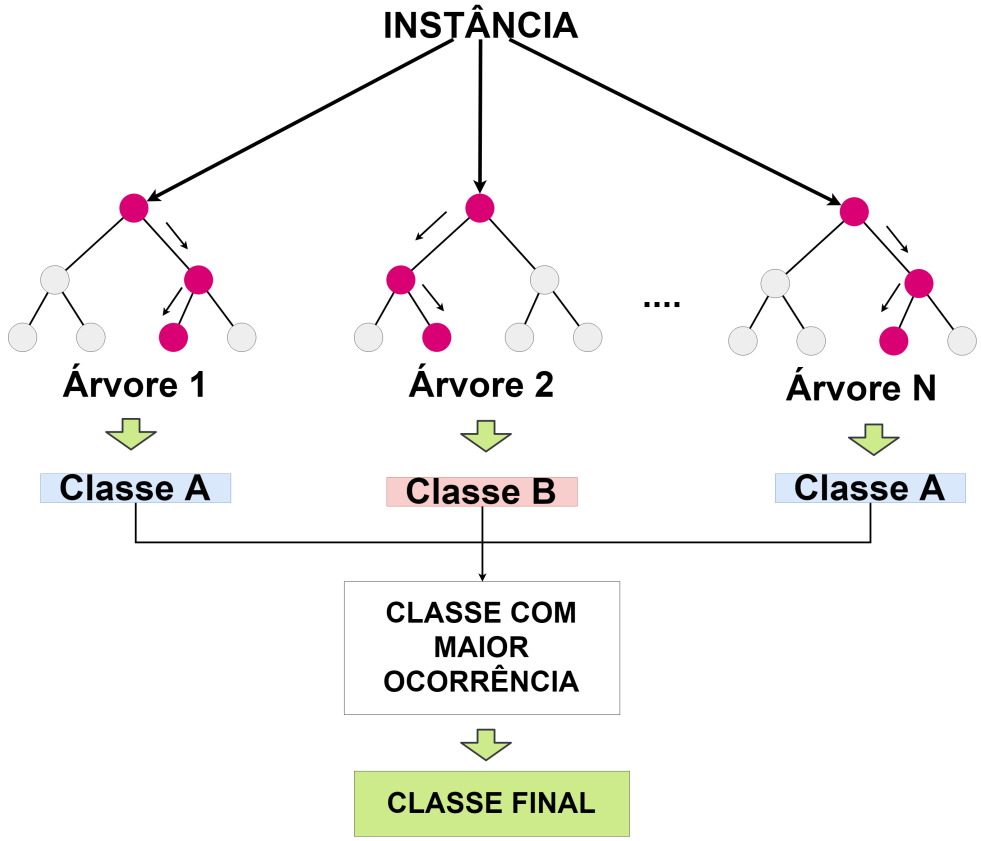


Figura 2.13: Funcionamento do Random Forest.

2.6.7 Perceptron Multicamadas

Perceptron Multicamadas (em inglês *Multilayer Perceptron - MLP*) é um algoritmo de aprendizado de máquina do grupo das Redes Neurais Artificiais. É uma Rede Neural formada por camada de entrada, cadas ocultas (mais de uma) e camada de saída, que pode conter mais de um neurônio. A Figura 2.14, apresenta a estrutura básica de um MLP.

Cada camada da rede tem uma função específica, a camada mais à esquerda, conhecida como camada de entrada, consiste em um conjunto de neurônios que representam os recursos de entrada. Cada neurônio na camada oculta transforma os valores da camada anterior com um somatório linear ponderado, seguido por uma função de ativação. A camada de saída recebe os valores da última camada oculta e os transforma em valores de saída. Esse algoritmo é utilizado principalmente em problemas não separáveis linearmente.

A função de ativação, mencionada anteriormente, é a função responsável por definir a ativação de saída do neurônio em termos do seu nível de ativação interna. Normalmente o sinal de ativação do neurônio pertence ao intervalo (0, 1) ou (-1, 1). As principais funções utilizadas para a ativação são:

- Função de Heaviside: também conhecida como Função Degrau, é uma função que apresenta valor zero quando seu argumento é negativo, e valor unitário quando o

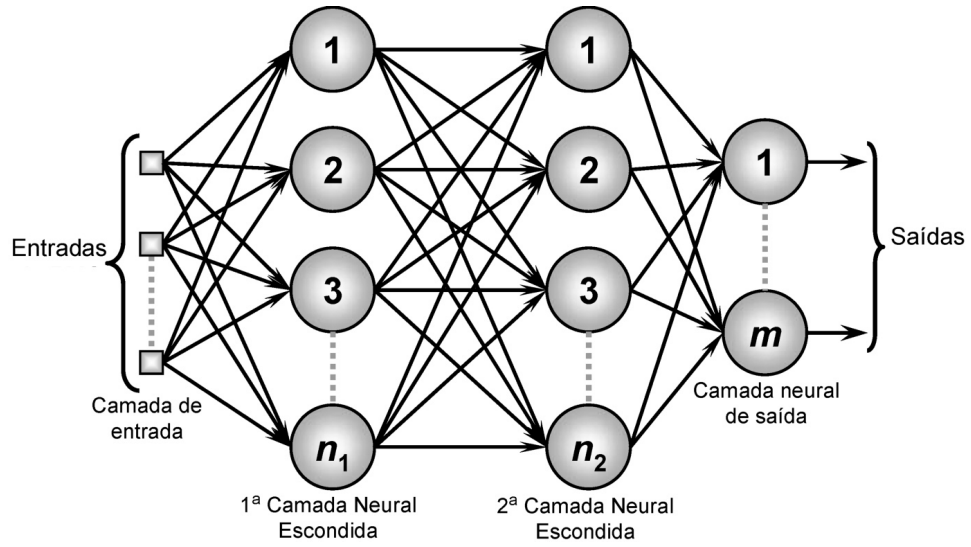


Figura 2.14: Estrutura do Perceptron Multicamadas.

argumento é positivo. É definida pela função,

$$u(t) = \begin{cases} 1, t \geq 0 \\ 0, t < 0 \end{cases} \tag{2.7}$$

- Função Sigmoid: é uma função de ativação amplamente utilizada e definida por,

$$f(x) = \frac{1}{1+e^{-x}}. \tag{2.8}$$

  Esta é uma função suave e continuamente diferenciável. Sua maior vantagem é que ela não é linear, ou seja, um MLP que utiliza essa função de ativação, pode ser aplicado em problemas que não são linearmente separáveis.

- Função ReLU: é a função de ativação mais utilizada atualmente, também se trata de uma função não linear que é definida por,

$$f(x) = max(0,x). \tag{2.9}$$

  A principal vantagem de usar a função ReLU sobre outras funções de ativação, é que ela não ativa todos os neurônios ao mesmo tempo. Isso significa que, ao mesmo tempo, apenas alguns neurônios são ativados, tornando a rede esparsa, eficiente e com menos necessidade de computação.

O algoritmo Perceptron Multicamadas vem sendo amplamente utilizado, junto com os conceitos de Deep Learning, para o desenvolvimento de diversas aplicações, sistemas de busca, tradução, indicação, reconhecimento de padrões, entre outros. Eles são reconhecidos pela sua capacidade de resolver problemas estocásticos, que muitas vezes permite obter soluções aproximadas para problemas extremamente complexos.

3

MODELOS DE CLASSIFICAÇÃO AUTOMÁTICA DE ATIVIDADES DIÁRIAS

Nesse capítulo serão apresentados o ambiente de desenvolvimento e toda metodologia utilizada.

3.1 Ambiente de Desenvolvimento e Ferramentas Utilizadas

Para implementar todos os testes e análises utilizamos a linguagem de programação Python versão 3.6.4 (Python, 2017), em conjunto com a plataforma de ciência de dados Anaconda (Anaconda, 2016), o ambiente de desenvolvimento científico Spyder versão 3.3.3 (Raybaut & Córdoba, 2009). Também utilizamos as bibliotecas listadas abaixo.

- Numpy (Walt et al., 2011): possibilita a manipulação de lista de dados de grandes proporções e múltiplas dimensões, criação de funções lineares, transformada de Fourier, geração de números aleatórios, operações matemáticas com matrizes (soma, subtração, multiplicação, escalar, etc), entre diversas outras funcionalidades.

- IPython (Pérez & Granger, 2007): é um interpretador Python, que fornece suporte interativo à visualização de dados, possibilita a depuração do código, salvar e carregar o ambiente de trabalho, fácil de utilizar e possui uma boa performance.

- Matplotlib (Hunter, 2007): é uma biblioteca para realizar plots em 2D, que produz gráficos de excelente qualidade e em vários formatos. Essa ferramenta pode gerar gráficos, histogramas, espectros de potência, gráficos de barras, gráficos de erros, gráficos de dispersão etc. O Matplotlib fornece uma API orientada a objetos de fácil entendimento e utilização.

- Pandas (McKinney, 2010): é uma biblioteca escrita em Python que oferece estruturas e operações para manipular tabelas numéricas e séries temporais. Entre as principais funções estão:
  - Criação da estrutura de dados DataFrame, uma tabela que aceita qualquer tipo de variável e possibilita diversas funções matemáticas e semânticas;
  - Operação com bases de dados e arquivos;
  - Cálculos estatísticos como: média, mediana, valor máximo, valor mínimo, desvio padrão etc.
  - Remove dados inconsistentes com base em valores ausentes e diversos outros critérios;
  - Visualização de dados com integração ao Matplotlib, possibilitando a construção de gráficos de barra, linhas, histogramas, bolhas, entre outros.
- Scikit-learn (Pedregosa et al., 2011): Uma das Bibliotecas de Aprendizado de Máquina mais utilizada atualmente, fornece diversos métodos de aprendizado supervisionado e não supervisionado. Possui compatibilidade com Numpy, Matplotlib e Pandas. O Scikit-learn fornece ferramentas de regressão, classificação, agrupamento, redução de dimensionalidade, seleção de modelos (testes de comparação e validação) e pre-processamento.

Como ambiente de execução utilizamos um Servidor Virtual Ubuntu/Linux com 32GB de memória RAM, CPU Intel com 8 núcleos de 1.6Ghz. Para verificar se a solução proposta teria compatibilidade com dispositivos de baixo poder de processamento, utilizamos o dispositivo Raspberry Pi 3 modelo B, processador Quad Core 1.2GHz Broadcom BCM2837 64bit, 1GB de memória RAM, 32GB de armazenamento via MicroSD e sistema operacional Raspibian versão 4.19.

3.2 Metodologia

Após definir o objeto de estudo, base de dados, ferramentas de validação, extrator de características e o conjunto de métodos de aprendizado de máquina que serão testados, vamos apresentar a metodologia utilizada para avaliação dos métodos de aprendizado de máquina, com o objetivo de realizar a classificação automáticas das Atividades Diárias. A Figura 3.1 apresenta um quadro resumo da metodologia.

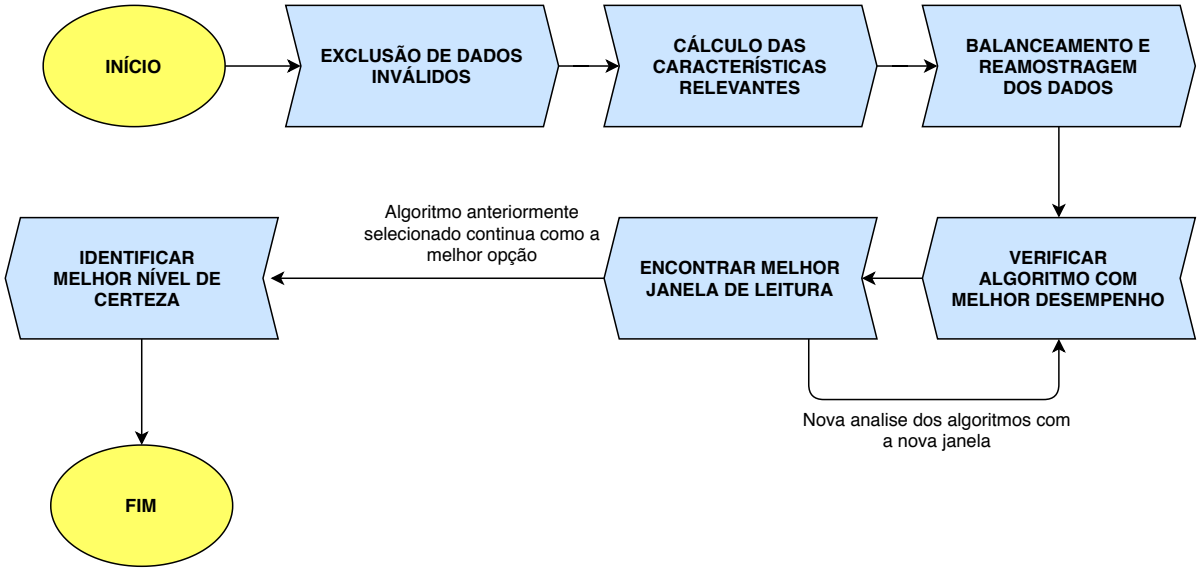


Figura 3.1: Metodologia utilizada na análise dos algoritmos.

Nas Seções a seguir serão detalhadas as etapas descrita na Figura 3.1:

3.2.1 Exclusão dos dados inválidos

O ponto de partida de todo processo de aprendizado de máquina é eliminar valores inconsistentes, nulos ou inválidos dos dados provenientes das bases de dados utilizadas. Para isso percorremos todos os registros das bases ARCMa, HMP e UMAFall em busca dessas ocorrências.

3.2.2 Cálculo das características relevantes

Definir o conjunto de características que melhor descreve as classes de um conjunto de dados é outro ponto crítico em um processo de aprendizado de máquina. Para encontrar as características mais relevantes em cada base de dados (ARCMa, HMP e UMAFall), utilizamos a blibloteca TSFRESH, já apresentada na Seção 2.5.1.

Para realizar o cálculo desse conjunto precisamos definir o tamanho das janelas de leitura, ou seja, o tamanho do conjunto de dados que sera utilizado para calcular cada característica. Como não temos conhecimento do valor ideal nesse contexto, definimos inicialmente como sendo a metade da taxa de amostragem de cada base de dados. Esse valor foi escolhido como ponto de partida da análise, mais a frente será realizada uma busca do tamanho de janela mais adequado para cada base de dados.

Foi utilizado a função *extract_relevant_features*, da biblioteca TSFRESH, para realizar o cálculo das características mais relevantes, conforme descrito na Seção 2.5.1. Essa função recebe como parâmetro uma lista de séries temporais e suas respectivas classes. Após realizar

o processamento, o método retorna uma matriz com todas as características extraídas. A Tabela 3.1 apresenta um exemplo de retorno desta função.

ID	TIME	F_x	F_y	F_z	T_x	T_y	T_z
1	0	12	-22	6	1	2	-10
1	1	-6	5	3	-2	4	7
2	2	4	7	-1	1	2	8
...	...	...	...	...	...	...	...

Quadro 3.1: Exemplo da matriz retornada pela função *extract_relevant_features*. Nas colunas temos as características relevantes, e nas linhas as séries temporais. A coluna "ID"contém a classe que a série temporal representa.

3.2.3 Balanceamento e re-amostragem dos dados

Um dos passos mais importantes na análise da acurácia de algoritmos de aprendizado de máquina, é garantir que a base de dados está devidamente balanceada. Isso evitará equívocos nos resultados obtidos, já que em bases muito desequilibradas, mesmo se o classificador apenas predizer as classes mais comuns, o modelo terá um bom desempenho.

Para evitar esse problema utilizamos o método de subamostragem, onde a quantidade de registro de cada classe se limita a quantidade da classe minoritária. Ou seja, basta reduzir o tamanho da amostra de cada classe para que todas tenham a mesma quantidade de registros. O tamanho mínimo limitado para a classe minoritária foi de 40 amostras. Caso a classe tenha menos que esse valor, ela será excluída dos dados. A Figura 3.2 exibe o funcionamento da subamostragem.

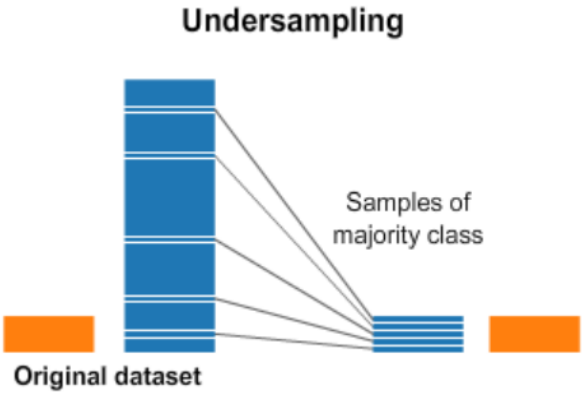


Figura 3.2: Processo de subamostragem.

Fonte: https://www.kaggle.com/

3.2.4 Algoritmo com melhor desempenho

Nessa etapa vamos verificar qual método de aprendizado de máquina, dentro do conjunto estudado, irá apresentar o melhor desempenho. Para mensurar o desempenho serão utilizadas 4 métricas amplamente aplicadas no mundo do aprendizado de máquinas, são elas: F-score, Acurácia, Precisão e Revocação. Como forma de validação será utilizado o método de Validação Cruzada Estratificada. Tanto das métricas de avaliação, quanto o método de validação foram descritos anteriormente na Seção 2.3. Com base nos trabalhos relacionados e para melhorar o tempo gasto no processo de validação, utilizamos $k = 5$ na Validação Cruzada Estratificada.

Outra característica importante que é verificada nessa etapa é o tempo que cada método necessita para realizar a classificação dos registros, essa informação é importante para realizar comparações e verificar se o método consegue ser executado em dispositivos com baixo poder de processamento, para isso, utilizamos um Servidor Virtual Ubuntu/Linux com 32GB de memória RAM, CPU Intel com 8 núcleos de 1.6Ghz, e em um dispositivo Raspberry Pi 3 modelo B, processador Quad Core 1.2GHz Broadcom BCM2837 64bit, 1GB de memória RAM, 32GB de armazenamento via MicroSD, que possui baixo poder de processamento, comparado com o servidor inicialmente utilizado.

Cada método de aprendizado de máquina teve seus parâmetros padrões definidos pelo Scikit-learn, ligeiramente modificados afim de otimizar a sua execução e melhorar os resultados obtidos.

3.2.5 Melhor janela de leitura

Após identificar o melhor método de Aprendizado de Máquina, utilizando o tamanho de janela inicial, metade da taxa de amostragem de cada base de dados, será iniciado uma busca para verificar qual é o tamanho de janela mais apropriado nesse contexto. Para isso, utilizamos um algoritmo iterativo que verifica o desempenho do algoritmo de aprendizado de maquina encontrado na etapa anterior, utilizando diferentes tamanhos de janelas, entre 5 a 100, com passo 5.

Após encontrar a janela que retorna o melhor resultado, voltamos a etapa anterior para encontrar novamente o algoritmo de aprendizado de máquina com melhor resultado, porém, utilizando o novo tamanho de janela encontrado. Essa iteração entre encontrar o melhor algoritmo e melhora janela se dá até que o algoritmo que apresente o mesmo resultado seja o mesmo utilizado para encontrar a melhor janela, como podemos verificar na Figura 3.1.

3.2.6 Identificar o melhor nível de certeza

Nesse ponto queremos elevar ao máximo os resultados do algoritmo selecionado nas etapas anteriores, para isso, será utilizado o nível de certeza que esse método retorna ao

realizar uma classificação.

Para obter esse nível de certeza foi utilizada a função *predict_proba* da biblioteca *Scikit-learn*(Pedregosa et al., 2011), que retorna a matriz de probabilidade da classificação para cada método de aprendizado de máquina utilizado, a Figura 3.3 apresenta o modelo de saída desse método.

Bending	Hopping	Jogging	LyingDown	Sitting	Walking	backwardFall	forwardFall	lateralFall
0.062	0.03	0.039	0.05	0.114	0.111	0.076	0.415	0.103
0.007	0.495	0.246	0.009	0.005	0.113	0.019	0.08	0.026
0.094	0.262	0.004	0.057	0.192	0.029	0.258	0.04	0.064
0.015	0.004	0.003	0.141	0.056	0	0.599	0.163	0.019
0.037	0.018	0.012	0.198	0.152	0.007	0.421	0.121	0.034
0.689	0.046	0.019	0.015	0.036	0.009	0.087	0.06	0.039
0.055	0.149	0.35	0.027	0.038	0.128	0.058	0.078	0.117
0.06	0.052	0.005	0.088	0.032	0.07	0.63	0.021	0.042
0.058	0.011	0.02	0.022	0.014	0.006	0.075	0.072	0.722
0.046	0.092	0.098	0.07	0.088	0.059	0.273	0.136	0.138
0.081	0.095	0.024	0.198	0.081	0.255	0.095	0.075	0.096
0.118	0.09	0.009	0.055	0.082	0.008	0.298	0.082	0.258
0.115	0.079	0.027	0.118	0.052	0.013	0.388	0.049	0.159
0.112	0.092	0.106	0.105	0.207	0.059	0.116	0.084	0.119
0.006	0.005	0.024	0.031	0.014	0.01	0.893	0.006	0.011
0	0.001	0	0.051	0.012	0	0.916	0.013	0.007
0.044	0.044	0.054	0.048	0.117	0.036	0.08	0.236	0.341
0.045	0.018	0.014	0.547	0.124	0.068	0.103	0.016	0.065
0.016	0.515	0.151	0.032	0.019	0.07	0.085	0.067	0.045
0.852	0.022	0.002	0.009	0.038	0.002	0.04	0.025	0.01
0.008	0.163	0.419	0.02	0.012	0.054	0.218	0.05	0.056

Figura 3.3: Matriz de probabilidade para a base UMAFall.

Com base nessa tabela, foi estabelecido um filtro limiar que exclui as classificações que possuem nível de certeza menor ou igual a um limiar estabelecido, e assim, os registros que geram "dúvida"na classificação são desconsiderados

Para encontrar o limiar que melhor se adapta a esse cenário utilizamos novamente um algoritmo iterativo que verificar os níveis de certeza entre 5% e 95%. Nessa verificação será apresentado o desempenho e a quantidade de registros excluídos para cada limiar.

Também apresentamos a matriz de confusão antes e depois de aplicar o nível de certeza, o que ajuda a verificar o comportamento da classificação e a quanto esse filtro colabora com a precisão do modelo final.

Após todas as etapas executadas, se pretende obter dados relevantes que corroborem para a escolha do melhor conjunto de parâmetros e métodos, dentre os utilizados nesse estudo, para realizar a classificação automática de atividades diárias, levando em consideração sua compatibilidade com dispositivos que possuem baixo poder de processamento.

4

RESULTADOS

Nesse capítulo serão apresentados as análises e discussões dos resultados obtidos.

Com a finalidade de melhorar a didática do texto iremos apresentar os resultados seguindo os passos definidos na Seção 3.2.

4.1 Exclusão de dados inválidos

Verificar a consistência dos dados é imprescindível em um processo de análise. Tal procedimento evita possíveis erros de processamento. Neste trabalho foi verificado a existência de registros nulos e o tipo de dado presente em cada atributo, a existência de valores não numéricos nos atributos das coordenadas x, y e z, e a existência de valores não correspondentes aos atributos que identificam os voluntários e as atividades. Após realizar toda verificação, não foi encontrado nenhuma inconsistência em nenhuma base de dados utilizada. Possivelmente as bases ARCMA, HMP e UMAFall já passaram por processo semelhante antes de serem hospedadas no site da UCI Machine Learning.

4.2 Cálculo das Características Relevantes

Após carregar os dados provenientes das bases utilizadas, as séries temporais com o tamanho de janela definido na Seção 3.2 foram criadas e passadas como parâmetro da função *extract_relevant_features* da biblioteca TSFRESH (Christ et al., 2018). O Quadro 4.1 apresenta o total de características extraídas para cada base de dados[2]. Nesse quadro é possível verificar que cada base obteve uma quantidade diferente de características relevantes, essa diferença ocorre devido aos critérios de seleção de características da biblioteca TSFRESH e

[2]Lista detalhada com todas as características extraídas pode ser consultada em https://github.com/SensorNet-UFAL/NewActivityRecognition/blob/master/workspace/relevant_features_list.txt

das peculiaridades das base, a seleção das características relevantes foi descrita na Seção 2.5.1 [3].

Base	Quantidade de Características Extraídas
ARCMA	784
HMP	862
UMAFall	595

Quadro 4.1: Quantidade de características extraídas para cada base de dados.

4.3 Balanceamento e re-amostragem da base de dados

Ao analisar os registros das bases estudadas, ARCMA, HMP e UMAFall, verificou-se que elas estão desbalanceadas, ou seja, algumas ADLs estão com muitos registros e outras quase sem nenhum. As Figuras 4.1, 4.2 e 4.3 apresentam a distribuição das atividades em cada base.

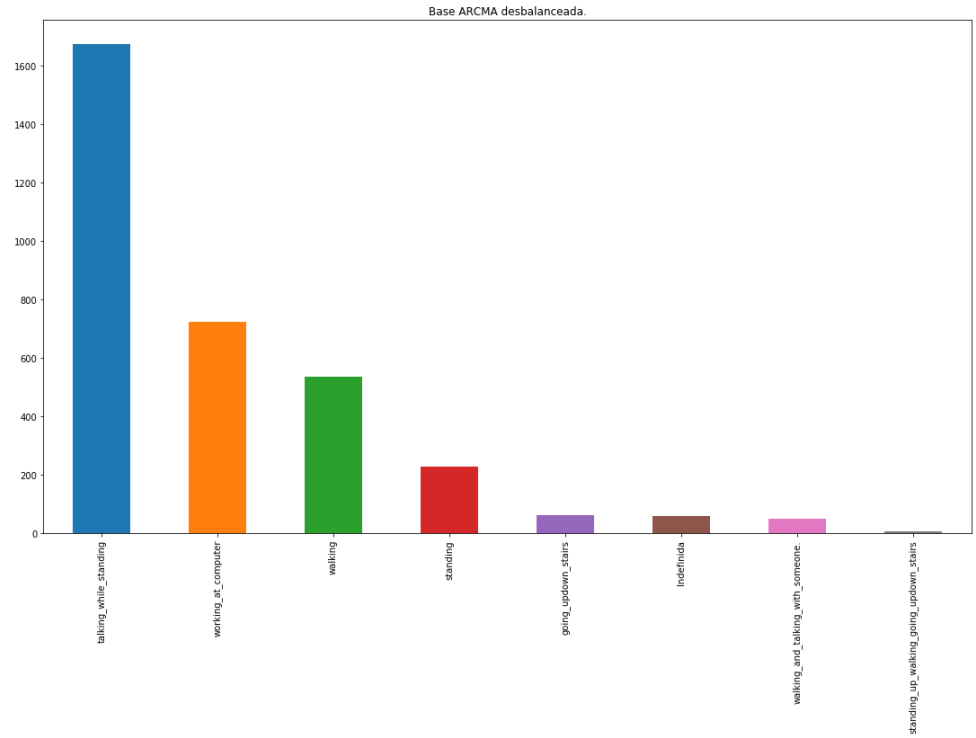


Figura 4.1: Distribuição original das atividades da base ARCMA para o voluntário 1.

[3]Mais detalhes sobre a extração das características pelo TSFRESH podem ser consultados no endereço https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html

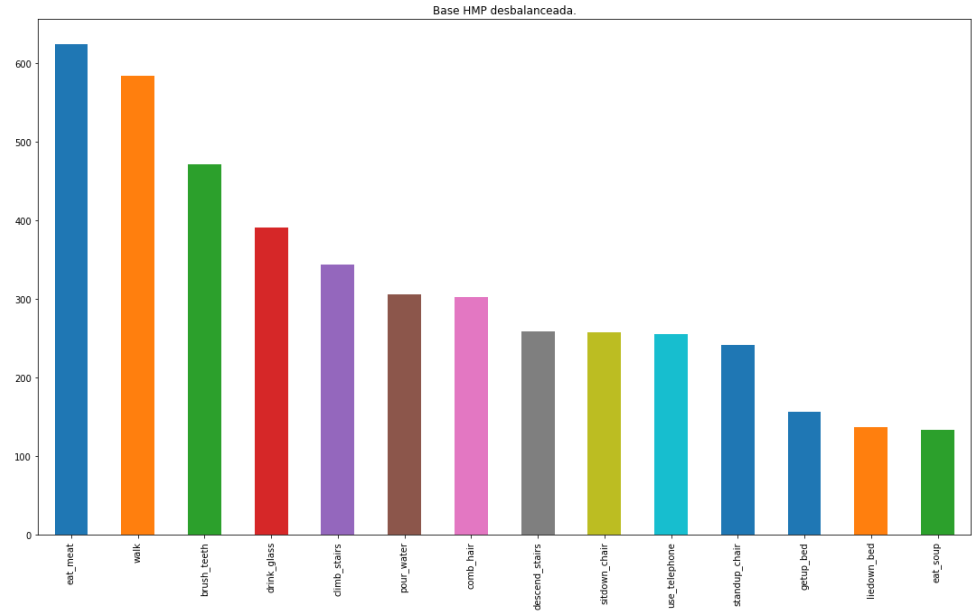


Figura 4.2: Distribuição original das atividades da base HMP para o voluntário F1.

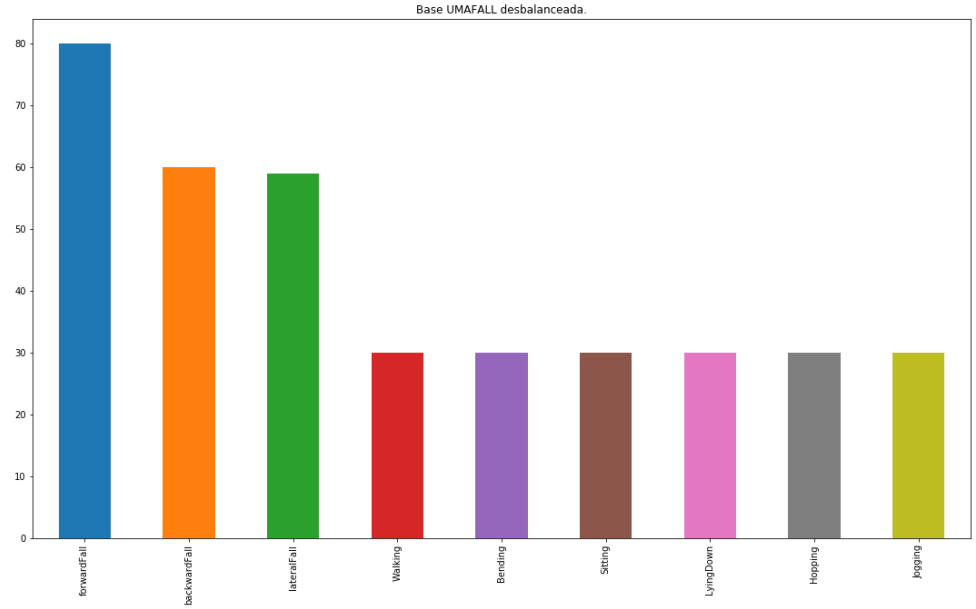


Figura 4.3: Distribuição original das atividades da base UMAFall para o voluntário 1.

Nesse caso precisamos realizar o balanceamento de todas as bases. Vamos tomar como exemplo a distribuição da Figura 4.1, podemos notar que mesmo se um modelo preditivo classificar todos os novos registros como *talking_while_standing*, ele continuará com uma boa acurácia, já que essa classe contém mais de 50% do total de registros da base.

Dessa forma realizamos a subamostragem para balancear as ADLs das bases de dados, como descrito no Seção 3.2.3. As Figuras 4.4, 4.5 e 4.6 contêm uma amostra das novas distribuições após o balanceamento.

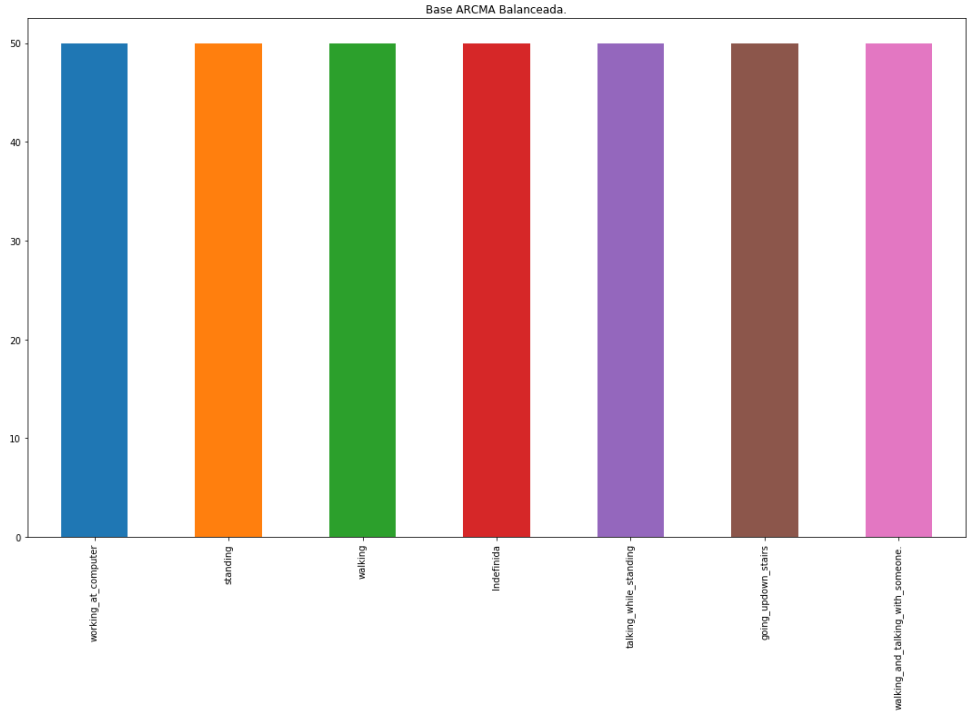


Figura 4.4: Distribuição balanceada das atividades da base ARCMA para o voluntário 1.

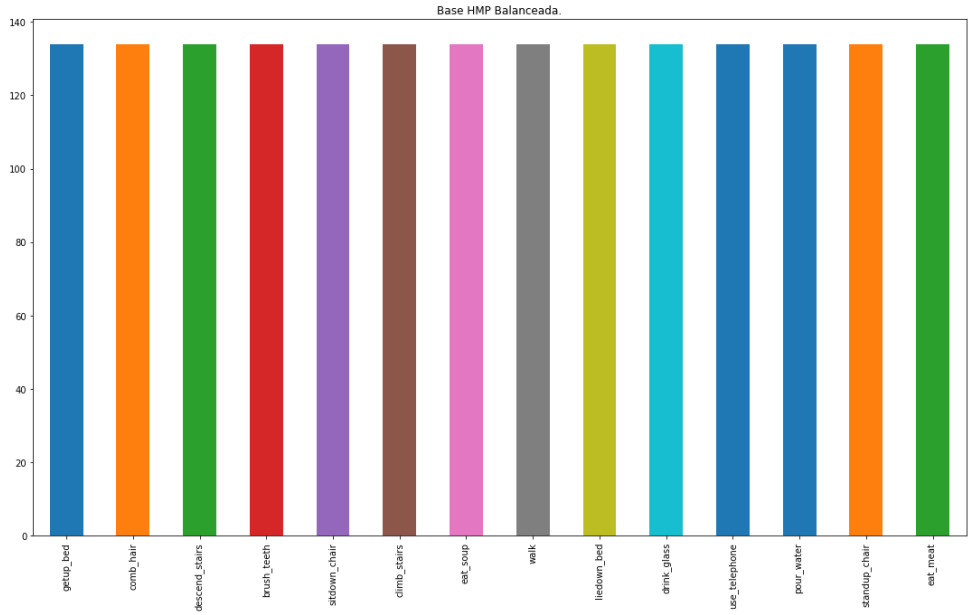


Figura 4.5: Distribuição balanceada das atividades da base HMP para o voluntário F1.

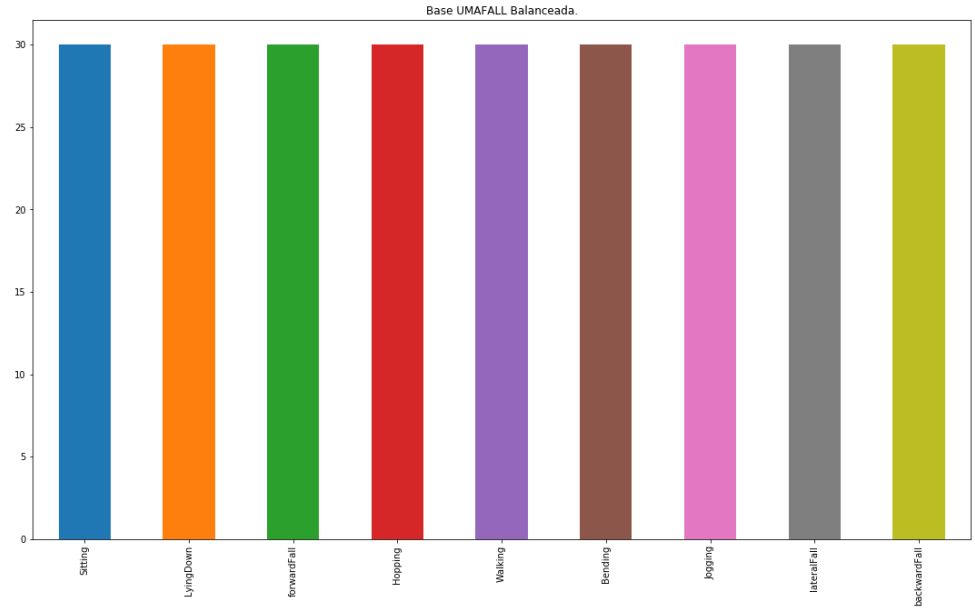


Figura 4.6: Distribuição balanceada das atividades da base UMAFall para o voluntário 1.

4.4 Melhor Método de Aprendizado de Máquina - 1º Iteração

Após realizar o balanceamento dos dados, o próximo passo foi verificar qual dos métodos de aprendizado de máquina, dentre os abordados nesse trabalho, apresentam o melhor desempenho utilizando o tamanho de janela inicial definido nas Seções 3.2.2 e 3.2.5. O Quadro 4.2 contém as configurações dos métodos, onde foram utilizados os parâmetros padrões definidos pela biblioteca Scikit-learn, ligeiramente modificados afim de otimizar a sua execução e melhorar os resultados obtidos.

Para realizar as classificações foi utilizado um Servidor Virtual Ubuntu/Linux com 32GB de memória RAM, CPU Intel com 8 núcleos de 1.6Ghz, e um dispositivo Raspberry Pi 3 modelo B, processador Quad Core 1.2GHz Broadcom BCM2837 64bit, 1GB de memória RAM, 32GB de armazenamento via MicroSD e sistema operacional Raspibian versão 4.19. Nesse trabalho consideramos o Raspberry Pi um dispositivo com baixo poder de processamento, desa forma, ele foi utilizado para verificar se os métodos conseguem ser executados nesse tipo de dispositivo.

Nos Quadros 4.3 a 4.5, temos os resultados dos métodos utilizando o tamanho de janela inicial, na primeira iteração.

Método	Parâmetros
k-NN	K = 5
Naive Bayes	-
Árvore de Decisão	Critério = "Gini impurity"
SVM	C=1; kernel = rbf
Random Forest	Nº Árvores=1000, Critério = "Gini impurity"
Extra-Trees	Nº Árvores=1000
Perceptron Multicamadas	activation="relu", hidden_layer_size=500, max_iter=100.000

Quadro 4.2: Configuração dos métodos utilizados.

Método	F-score	Precisão	Revocação	Acurácia	Tempo Servidor Virtual	Tempo Raspberry Pi
Random Forest	0.86	87.00%	86.59%	86.59%	0.00533s	0.4319s
SVM	0.47	47.70%	53.26%	53.26%	0.00015s	0.0064s
Extra-Trees	0.88	89.21%	89.21%	88.74%	0.00538s	0.4391s
Árvore de Decisão	0.78	79.76%	78,97%	78.97%	0.00007s	0.0053s
k-NN	0.75	76.53%	76.30%	76.30%	0.00025s	0.0085s
Naive Bayes	0.56	59.45%	59.32%	59.32%	0.00009s	0.0069s
Perceptron Multicamadas	0.04	4.35%	15.05%	15.05%	0.0045s	

Quadro 4.3: UMAFall - 1º iteração

Método	F-score	Precisão	Revocação	Acurácia	Tempo Servidor Virtual	Tempo Raspberry Pi
Random Forest	0.71	72.79%	72.79%	71.58%	0.00138s	0.4495s
SVM	0.22	24.21%	31.35%	31.35%	0.00005s	0.0071s
Extra-Trees	0.73	74.44%	73.74%	73.74%	0.00145s	0.4486s

Árvore de Decisão	0.58	60.13%	59.30%	59.30%	0.00003s	0.0054s
k-NN	0.63	64.85%	64.20%	64.20%	0.00003s	0.0090s
Naive Bayes	0.47	50.53%	51.69%	51.69%	0.00840s	0.0084s
Perceptron Multicamadas	0.03	02.14%	14.59%	14.59%	0.00443s	

Quadro 4.4: ARCMA - 1º iteração

Método	**F-score**	**Precisão**	**Revocação**	**Acurácia**	**Tempo Servidor Virtual**	**Tempo Raspberry Pi**
Random Forest	0.67	69.20%	68.26%	69.26%	0.00180s	0.4491s
SVM	0.17	19.36%	25.10%	54.10%	0.00116s	0.0115s
Extra-Trees	0.68	70.25%	69.51%	69.51%	0.00192s	0.4496s
Árvore de Decisão	0.55	56.21%	55.38%	55.38%	0.00004s	0.0056s
k-NN	0.58	60.04%	59.01%	59.01%	0.00023s	0.0106s
Naive Bayes	0.17	19.29%	25.26%	25.26%	0.00007s	0.0119s
Perceptron Multicamadas	0.03	01.98%	13.24%	13+24%	0.00473s	

Quadro 4.5: HMP - 1º iteração

Verificando os Quadros 4.3 a 4.5, constata-se que o método Extra-Trees foi o método que apresentou o melhor resultado em todas as métricas e bases utilizadas. A tempo descrito nas duas ultimas colunas de cada quadro, é o tempo equivalente para que os métodos executem uma única classificação, cada coluna representa um ambiente de execução.

4.5 Melhor Tamanho de Janela com *Extra-Trees*

Após identificar que o método *Extra-Trees* apresenta os melhores resultados utilizando o tamanho de janela inicial, foi utilizado um algoritmo iterativo para verificar o desempenho do *Extra-Trees* em diferentes tamanhos de janela entre 10 a 100, com passo 10. Os Quadros 4.6 à 4.8, apresentam o desempenhos do *Extra-Trees* utilizando diferentes tamanhos de janela.

Janela	**F-score**	**Acurácia**	**Precisão**	**Revocação**
10	0.83	83.49%	85.24%	83.49%

20	0.81	81.72%	82.60%	81.72%
30	0.81	82.06%	83.65%	82.06%
40	0.81	81.70%	83.21%	81.70%

Quadro 4.6: UMAFall - Desempenho por tamanho de janela.

Janela	**F-score**	**Acurácia**	**Precisão**	**Revocação**
10	0.81	84.05%	82.81%	84.05%
20	0.82	84.65%	83.52%	84.65%
30	0.82	84.34%	83.59%	84.34%
40	0.83	84;98%	84.25%	84.98%
50	0.82	84.67%	83.17%	84.67%
60	0.82	84.56%	83.54%	84.56%
70	0.82	84.55%	83.92%	84.55%
80	0.83	85.12%	83.84%	85.12%
90	0.81	84.07%	82.57%	84.07%
100	0.83	85.06%	84.23%	85.06%

Quadro 4.7: ARCMA - Desempenho por tamanho de janela.

Janela	**F-score**	**Acurácia**	**Precisão**	**Revocação**
10	0.61	62.33%	62.56%	62.33%
20	0.65	65.81%	66.23%	65.81%
30	0.68	68.62%	69.43%	68.62%
40	0.70	70.70%	71.43%	70.70%
50	0.71	72.26%	73.23%	72.26%
60	0.73	73.62%	74.36%	73.62%
70	0.74	75.02%	76.54%	75.02%
80	0.76	76.53%	77.37%	76.53%
90	0.76	77.31%	78.55%	77.31%
100	0.76	77.21%	77.98%	77.21%

Quadro 4.8: HMP - Desempenho por tamanho de janela.

Nessa primeira iteração verificamos que os melhores resultados ocorreram nas janelas de tamanho 10, 40 e 90, para as bases UMAFall, ARCMA e HMP, respectivamente. Devido a baixa taxa de amostragem da base UMAFall, $20Hz$, o Quadro 4.6 possui registro apenas até a janela de tamanho 40, já que a base não possui registros suficientes para realizar os testes utilizando

janelas maiores.

Após otimizar o tamanho da janela para *Extra-Trees*, será realizado novo teste para verificar se esse algoritmo continua como a melhor opção para os novos valores de janela encontrados, conforme a metodologia apresentada na Seção 3.2.

4.6 Melhor Método de Aprendizado de Máquina - 2º Iteração

Nessa etapa será verificado novamente qual método de aprendizado de máquina apresenta o melhor desempenho, porém, agora utilizando os tamanhos de janela de leitura encontrados na etapa anterior. Essa nova busca não será executada para a base UMAFall, uma vez que o melhor tamanho de janela encontrado foi o mesmo utilizado na 1º busca dos algoritmos, Seção 4.4. Os Quadros 4.9 e 4.10 apresentam os novos resultados para as bases ARCMA e HMP.

Método	**F-score**	**Acurácia**	**Precisão**	**Revocação**	**Tempo Servidor Virtual**	**Tempo Raspberry Pi**
Perceptron Multicamadas	0.04	15.37%	2.39%	15.37%	0.00443s	
Extra-Trees	0.74	74.73%	75.61%	74.73%	0.00145s	0.4486s
k-NN	0.65	65.07%	66.27%	65.07%	0.00003s	0.0090s
Naive Bayes	0.55	58.48%	58.93%	58.48%	0.00840s	0.0084s
Random Forest	0.74	74.30%	75.30%	74.30%	0.00138s	0.4495s
Árvore de Decisão	0.62	62.51%	63.52%	62.51%	0.00003s	0.0054s
SVM	0.26	35.03%	26.42%	35.03%	0.00005s	0.0071s

Quadro 4.9: ARCMA - 2º iteração

Método	**F-score**	**Acurácia**	**Precisão**	**Revocação**	**Tempo Servidor Virtual**	**Tempo Raspberry Pi**
Perceptron Multicamadas	0.13	26.98%	8.96%	26.98%	0.00473s	
Extra-Trees	0.79	79.02%	80.85%	79.02%	0.00192s	0.4496s
k-NN	0.64	64.74%	66.03%	64.74%	0.00023s	0.0106s
Naive Bayes	0.62	63.16%	65.50%	63.16%	0.00007s	0.0119s
Random Forest	0.78	77.67%	79.67%	77.67%	0.00180s	0.4491s

Árvore de Decisão	0.64	64.27%	65.38%	64.27%	0.00004s	0.0056s
SVM	0.55	57.60%	62.61%	57.60%	0.00116s	0.0115s

Quadro 4.10: HMP - 2º iteração

Analisando os Quadros 4.9 e 4.10 verifica-se que o método *Extra-Trees* continua apresentando o melhor desempenho para todas as bases, dessa forma, a iteração pode ser finalizada. Também é possível constatar o tempo de execução de cada algoritmos, e assim nota-se que o *Extra-Trees* executa as classificações em menos de 0.5s, o que consideramos um tempo satisfatório para uma aplicação de reconhecimento automático de atividades diárias.

4.7 Filtro Linear - Utilizando Nível de Certeza

Os Quadros 4.11 à 4.13 e apresentam o desempenho obtido com o método *Extra-Trees* utilizando diferentes níveis de certeza.

Utilizando as regras estabelecidas na Seção 3.2 os valores de 65, 65 e 55 foram escolhidos para os limiares das bases ARCMA, HMP e UMAFall respectivamente, os quais alcançaram acurácias de 94.80%, 97.20% e 96.17%.

Limiar	**F-score**	**Acurácia**	**Precisão**	**Revocação**	**Atividades Descartadas**
5.00%	80.55%	81.59%	82.28%	81.59%	0
10.00%	80.55%	81.59%	82.28%	81.59%	0
15.00%	80.55%	81.59%	82.28%	81.59%	0
20.00%	80.61%	81.65%	82.36%	81.65%	0
25.00%	80.98%	82.01%	82.76%	82.01%	0
30.00%	81.84%	82.85%	83.63%	82.85%	0
35.00%	83.36%	84.35%	85.12%	84.35%	0
40.00%	85.28%	86.17%	87.02%	86.17%	0
45.00%	87.44%	88.36%	88.85%	88.36%	0
50.00%	89.31%	90.26%	90.55%	90.26%	0
55.00%	91.18%	92.05%	92.33%	92.05%	0
60.00%	92.36%	93.18%	93.49%	93.18%	0
65.00%	93.53%	94.36%	94.41%	94.36%	0
70.00%	94.34%	95.05%	95.25%	95.05%	1
75.00%	94.92%	95.61%	95.40%	95.61%	1
80.00%	95.98%	96.51%	96.58%	96.51%	1
85.00%	96.49%	96.93%	97.14%	96.93%	2

90.00%	97.41%	97.79%	97.75%	97.79%	2
95.00%	98.44%	98.77%	98.52%	98.77%	3

Quadro 4.11: UMAFall - desempenho por Limiar

Limiar	**F-score**	**Acurácia**	**Precisão**	**Revocação**	**Atividades Descartadas**
5.00%	76.91%	78.35%	78.61%	78.35%	0
10.00%	76.91%	78.35%	78.61%	78.35%	0
15.00%	76.91%	78.35%	78.61%	78.35%	0
20.00%	76.91%	78.35%	78.61%	78.35%	0
25.00%	76.97%	78.41%	78.69%	78.41%	0
30.00%	77.30%	78.73%	79.05%	78.73%	0
35.00%	78.07%	79.52%	79.77%	79.52%	0
40.00%	79.18%	80.65%	80.63%	80.65%	0
45.00%	80.68%	82.09%	82.17%	82.09%	0
50.00%	82.03%	83.42%	83.32%	83.42%	0
55.00%	83.55%	84.94%	84.68%	84.94%	0
60.00%	85.09%	86.39%	86.08%	86.39%	1
65.00%	86.63%	87.78%	87.65%	87.78%	1
70.00%	88.04%	89.07%	89.13%	89.07%	1
75.00%	89.16%	90.07%	90.39%	90.07%	1
80.00%	90.22%	91.08%	91.70%	91.08%	2
85.00%	91.29%	92.06%	93.00%	92.06%	2
90.00%	92.33%	93.12%	94.51%	93.12%	3
95.00%	92.52%	93.39%	94.14%	93.39%	4

Quadro 4.12: ARCMA - desempenho por Limiar

Limiar	**F-score**	**Acurácia**	**Precisão**	**Revocação**	**Atividades Descartadas**
5.00%	72.81%	75.01%	73.68%	75.01%	0
10.00%	72.81%	75.01%	73.68%	75.01%	0
15.00%	72.92%	75.11%	73.78%	75.11%	0
20.00%	73.73%	75.87%	74.66%	75.87%	0
25.00%	75.47%	77.51%	76.18%	77.51%	0
30.00%	77.77%	79.78%	78.16%	79.78%	0
35.00%	80.84%	82.62%	81.21%	82.62%	0
40.00%	83.77%	85.40%	83.85%	85.40%	0
45.00%	86.55%	87.97%	86.85%	87.97%	1

50.00%	89.36%	90.66%	89.80%	90.66%	1
55.00%	90.83%	92.29%	90.44%	92.29%	2
60.00%	92.76%	93.97%	92.57%	93.97%	2
65.00%	94.40%	95.44%	94.22%	95.44%	3
70.00%	96.33%	97.23%	95.75%	97.23%	3
75.00%	96.92%	97.73%	96.38%	97.73%	4
80.00%	98.08%	98.60%	97.80%	98.60%	5
85.00%	99.11%	99.38%	98.90%	99.38%	6
90.00%	99.02%	99.30%	98.82%	99.30%	6
95.00%	99.90%	99.94%	99.87%	99.94%	7

Quadro 4.13: HMP - desempenho por Limiar

A última coluna da cada Quadro representa a quantidade de atividades que foram excluídas dos registros das bases, ou seja, que deixaram de existir após filtro ser aplicado. Como é de se esperar nota-se que, ao aumentar o nível de certeza, uma parte dos dados é descartada pelo método e o desempenho do algoritmo melhora. Isso ocorre porque alguns dos novos registros classificados não possuem um nível de certeza de classificação maior que o limiar estabelecido, fazendo com que sejam desconsiderados. Uma das vantagens desse método é que, dependendo da necessidade da aplicação, podemos aumentar o nível de certeza para que o método classifique apenas as entradas que possuem um certo grau de certeza.

Com o objetivo de verificar com maior clareza o comportamento do modelo obtido foram construídas amostras das matrizes de confusão de cada base, antes e depois de se aplicar o nível de certeza como filtro. Para isso, foi escolhido um cenário onde seria tolerável o descarte de até 2 atividades de cada base de dados, dessa forma, foram escolhidos os limiares de 90%, 85% e 60%, o que representou um ganho de acurácia de 9,05%, 16,33% e 14.95% para as bases UMAFall, ARCMA e HMP. As Figuras 4.7, 4.8, 4.9, 4.10, 4.11 e 4.12 apresentam essas matrizes de confusão, baseada na acurácia obtida, onde é possível verificar que o nível de certeza melhora o desempenho do modelo, ao custo do descarte de alguns registros.

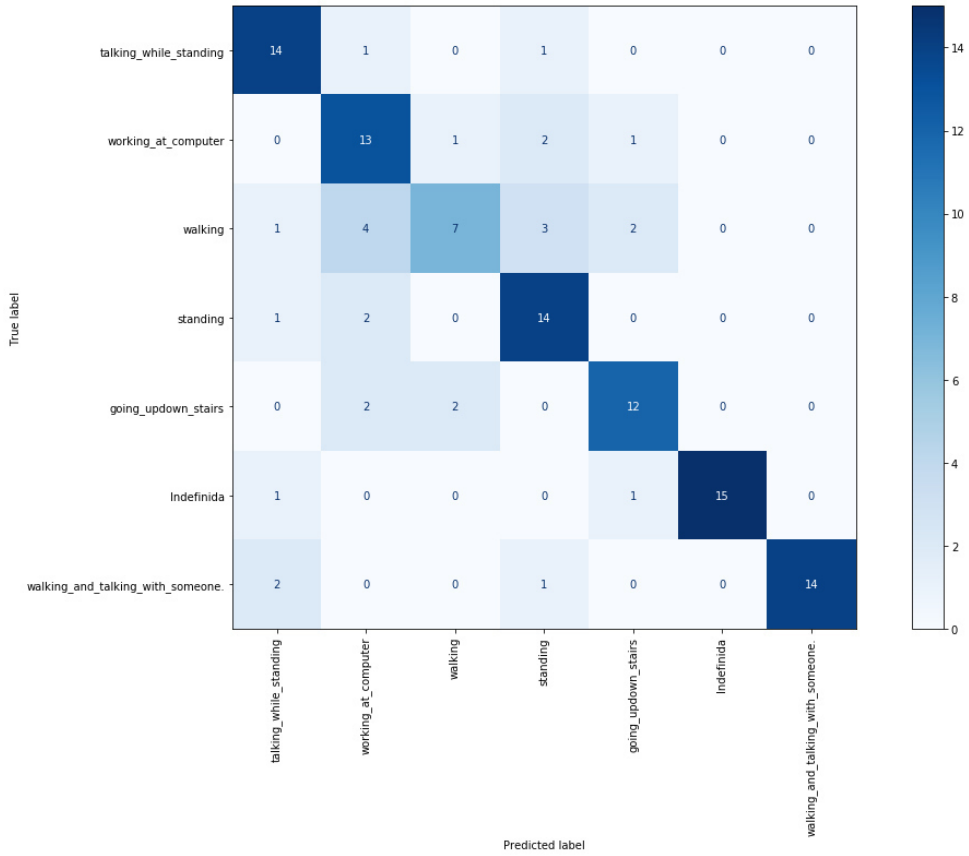


Figura 4.7: Amostra da matriz de confusão para base ARCMA, voluntário 1, com acurácia total de 76.06%.

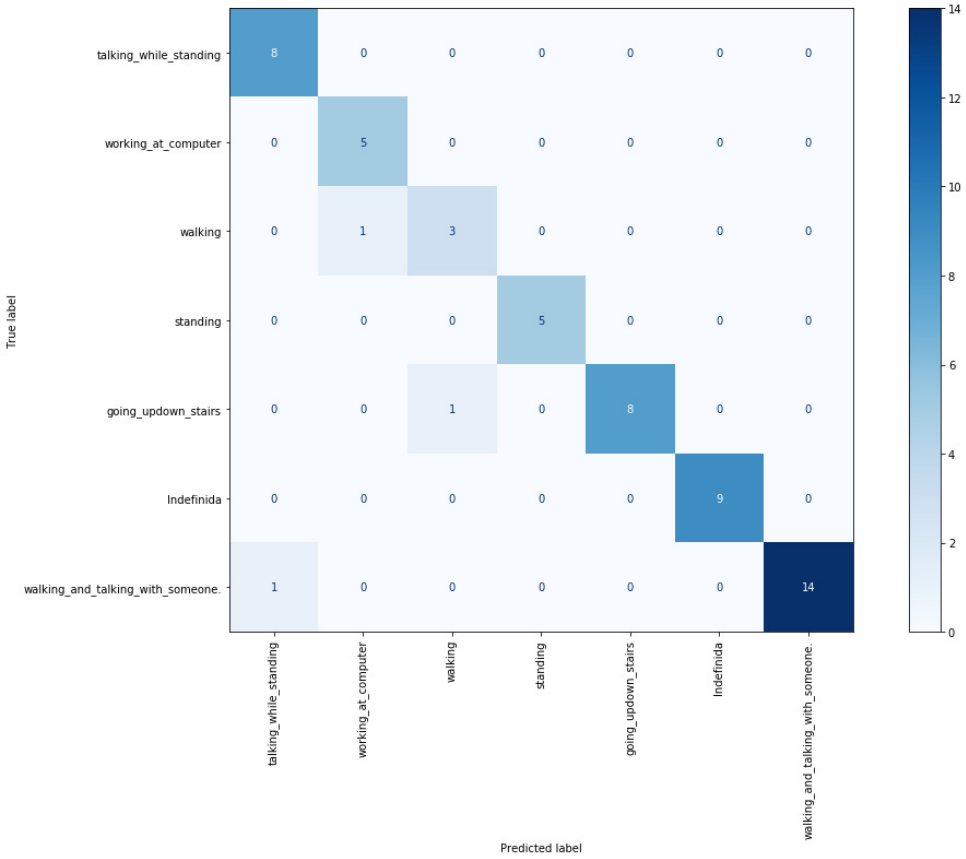


Figura 4.8: Amostra da matriz de confusão para base ARCMA, voluntário 1, após aplicar o nível de certeza de 65%, acurácia total de 94.54%.

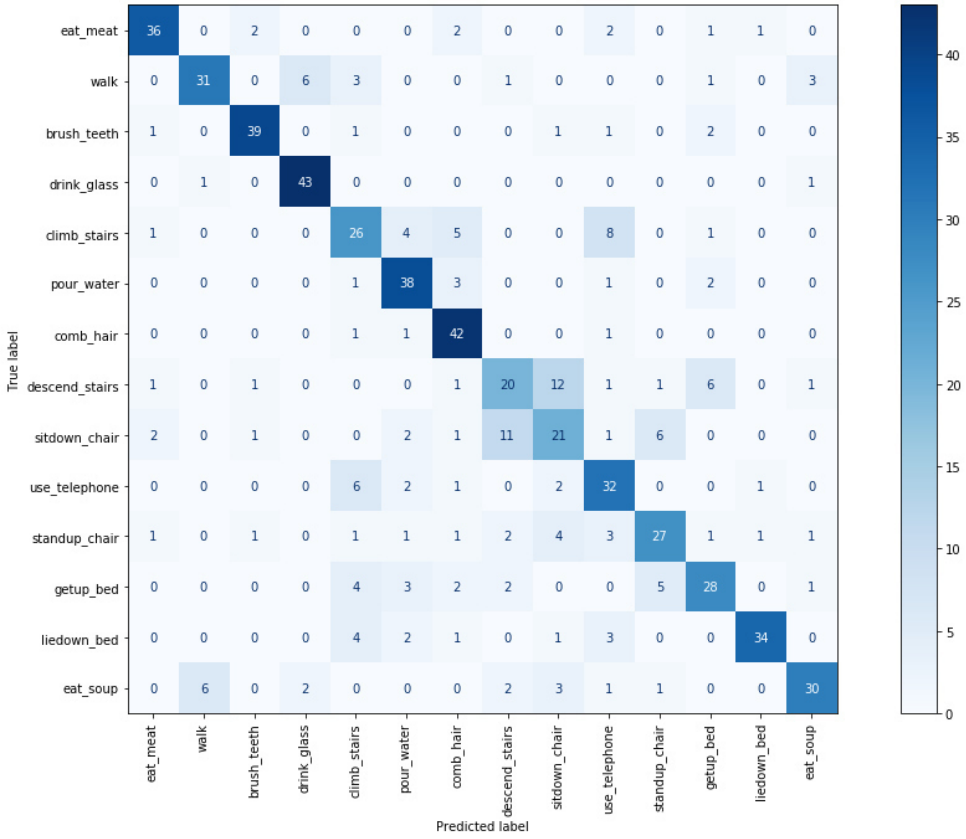


Figura 4.9: Amostra da matriz de confusão para base HMP, voluntário F1, com acurácia total de 71.40%.

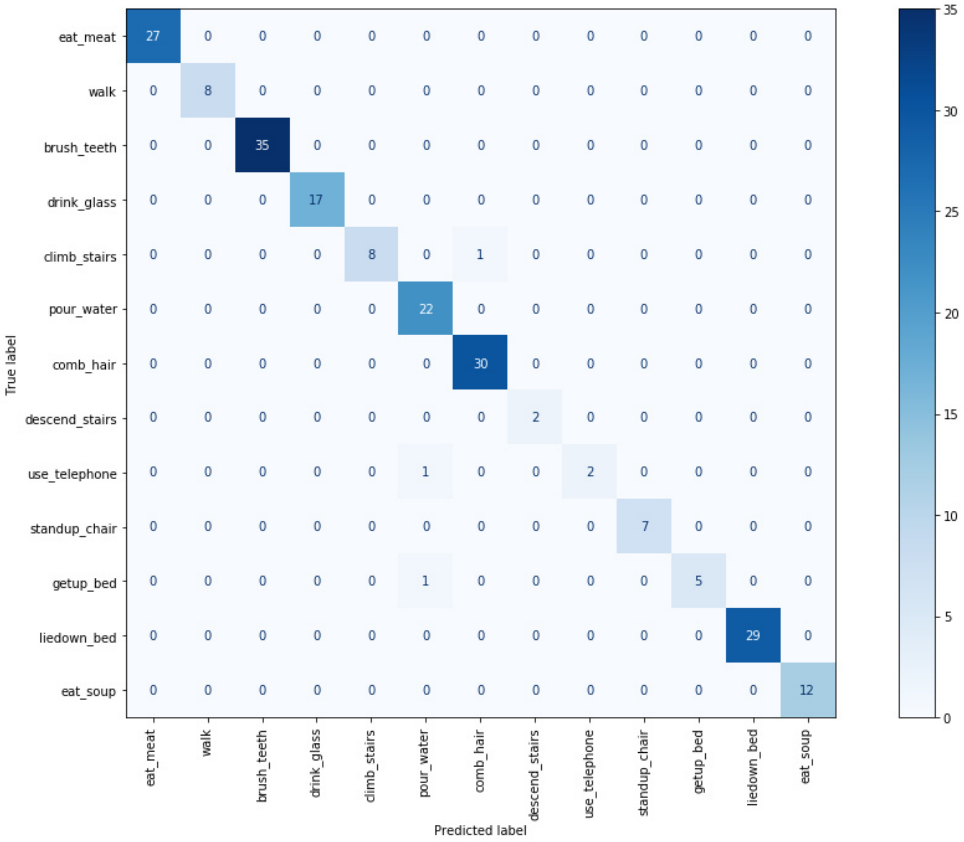


Figura 4.10: Amostra da matriz de confusão para base HMP, voluntário F1, após aplicar o nível de certeza de 65%, acurácia total de 98.55%.

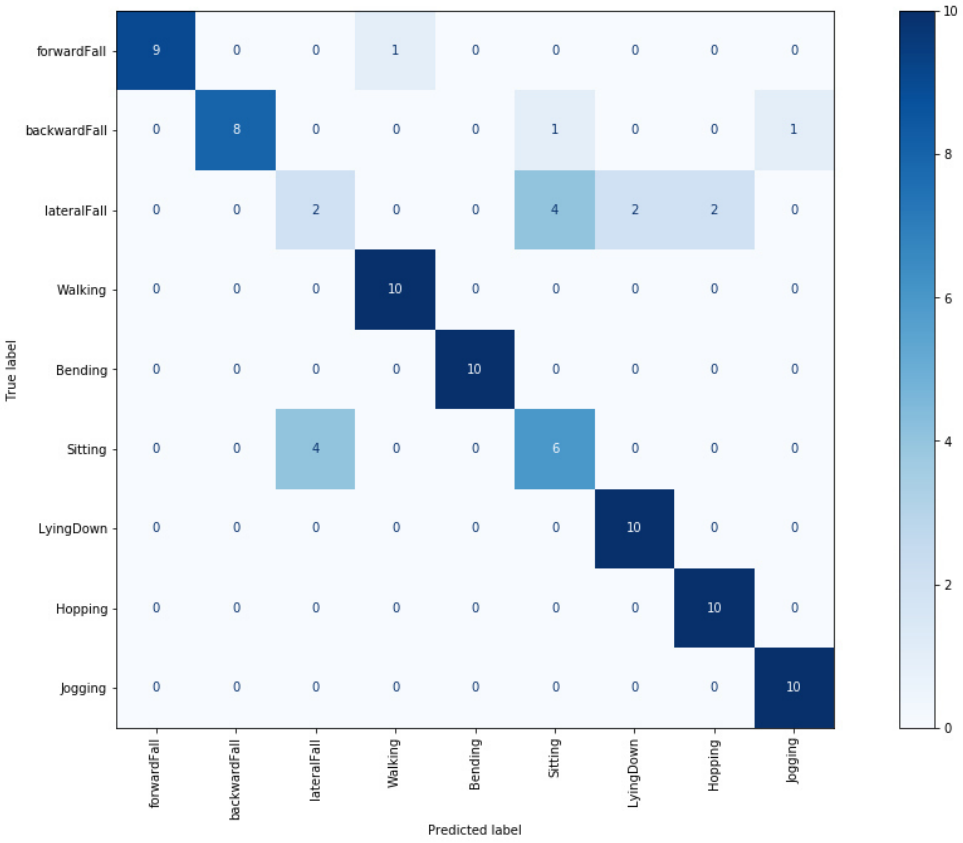


Figura 4.11: Amostra da matriz de confusão para base UMAFall, voluntário 1, com acurácia total de 83.33%.

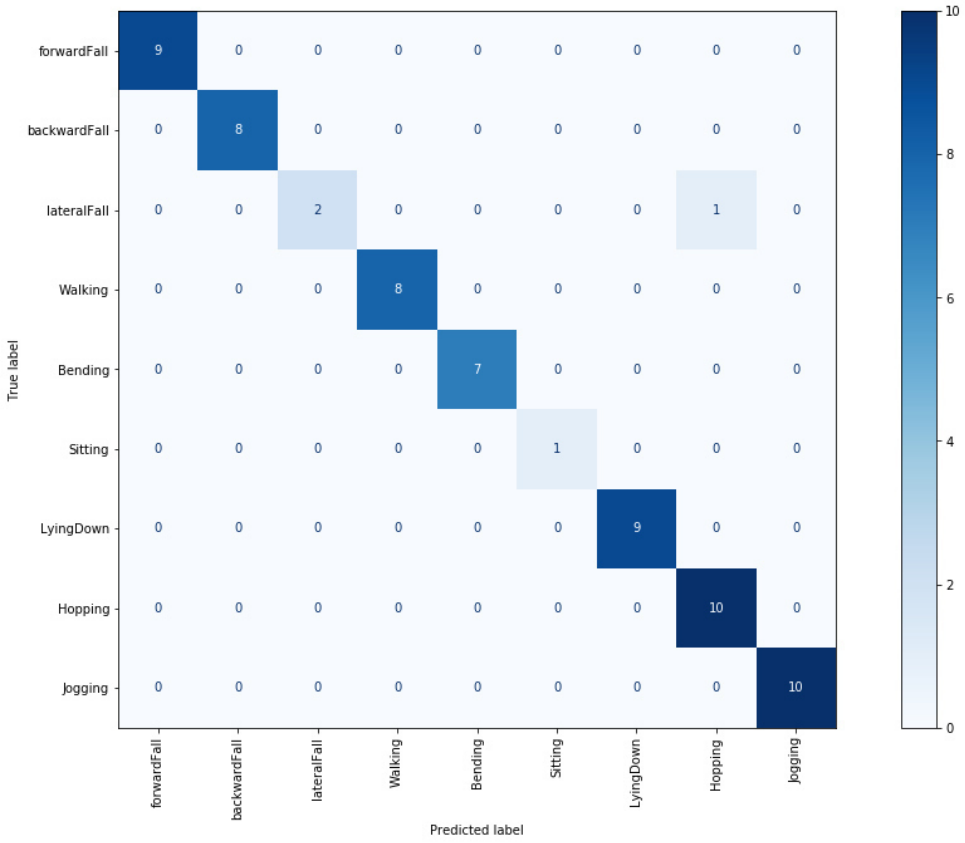


Figura 4.12: Amostra da matriz de confusão para base UMAFall, voluntário 1, após aplicar o nível de certeza de 55%, acurácia total de 98.46%.

Na base ARCMA verificamos pela Figura 4.7 que a ADL que mais se confunde com as outras é a *"Caminhar"*, confundida com outras 4 atividades, principalmente com a ADL *"Trabalhando no Computador"*. Essa confusão é fortemente reduzida ao se utilizar o nível de certeza para filtrar os dados, como visto na Figura 4.8. Da mesma forma na base HMP, Figura 4.9, a ADL *"Descer escadas"* é frequentemente confundida como *"Sentar na cadeira"*. Essa confusão pode ocorrer porque nas duas atividades o indivíduo está se deslocando para baixo, porém ao aplicar o nível de certeza essa confusão some totalmente, conforme Figura 4.10. Nesse exemplo é possível verificar que as ADLs que mais se confundem são as mais descartadas após o filtro.

Na matriz de confusão para a base UMAFall 4.11 verificamos poucos erros. O que mais se destaca é a classificação da ADL *"Queda para o lado"* que algumas vezes é confundido com *"Deitar"*, e vice e versa. Essa confusão parece ser coerente, já que esses movimentos são realmente parecidos, e se diferem talvez pelo contexto e pela velocidade do movimento. Porém, ao aplicar o nível de certeza, verifica-se que esse tipo de erro não ocorre mais, Figura 4.12.

Os trabalhos relacionados que apresentaram a melhor acurácia foram os de Barshan & Yüksek (2014) e Chetty & White (2016), com acurácias de 99.2% e 96.00%. Para um cenário em que seria tolerável o descarte de até 2 atividades, as bases UMAFall, ARCMA e HMP tiveram resultados bem próximos a esses trabalhos. O UMAFall chegou a ultrapassar a acurácia

apresentada em Chetty & White (2016), porém, os estudos citados acima utilizaram dados gerados por uma rede complexa com vários sensores espalhados pelo corpo dos voluntários, o que faz esse tipo de implementação ser muito difícil de ser utilizada em um ambiente real, o que pode inviabilizar essa solução. Por outro lado, o modelo encontrado nesse trabalho utiliza apenas um sensor inercial para realizar as classificações, e mesmo assim, alcança bons resultados tando na acurácia quando no F-score, o que sugere que ao mesmo tempo que o método mantém uma boa frequência de acertos, também é preciso, ou seja, classifica de forma correta as ocorrências que realmente são daquela classe. Essa simplificação torna a implantação desse modelo menos complexa, necessitando apenas de um dispositivo que realize as leituras, o qual poderia ser um smartphone colocado na cintura (para replicar os experimentos com a base UMAFall), ou um smartwatch no pulso dos voluntários (como foi feito para base HMP). Todos esses indícios indicam que o objetivo final deste trabalho foi alcançado.

Por fim, vale destacar que cada base exigiu diferentes configurações para atingir o máximo de acurácia, entre essas configurações estão: os atributos extraídos, o tamanho da janela de leitura e o limiar utilizado pelo filtro. Tais diferenças ocorrem devido as características específicas de cada base, taxa de amostragem, atividades registradas e a forma que os voluntários realizaram essas atividades. Dessa forma, a metodologia proposta para encontrar modelos para classificação de Atividades Diárias pode se adapta facilmente a qualquer outra base de dados, já que os atributos mais relevantes das séries temporais são extraídos automaticamente pelo TSFRESH, e a seleção do tamanho da janela e do limiar pode ser realizada de forma automatizada.

5

CONCLUSÕES

Esse trabalho apresentou uma metodologia para avaliar métodos de aprendizado de máquina para realizar a classificação de Atividades Diárias presentes em três bases de dados públicas, ARCMA, HMP e UMAFall. Dentre as principais contribuições deste trabalho temos:

- Proposta de uma metodologia de avaliação de métodos de aprendizado de máquina com o objetivo de realizar classificação automática de Atividades Diárias;
- Proposta de um filtro baseado no nível de certeza para melhorar os resultados dos métodos.
- Definição de um modelo que obteve bons resultados, e a acurácia bem próxima aos trabalhos relacionados, porém, utilizando dados colhidos de forma mais simples, utilizando apenas um sensor inercial.
- Ao executar as classificações em um dispositivo Raspberry Pi 3, verificamos que o modelo final é compatível com dispositivos que possuem baixo poder de processamento.

Com os estudos realizados identificamos que o *Extra-Trees* é uma boa escolha, com relação aos resultados da classificação e ao tempo de processamento, para realizar a classificação de atividades utilizando o conjunto de atributos calculados pela biblioteca TSFRESH e uma base de dados nos moldes das bases UMAFall, ARCMA e HMP. Também conseguimos bom valores de F-score e acurácia na classificação para essas bases.

Como trabalhos futuros temos:

- Utilizar o conhecimento adquirido em uma aplicação de tempo real, monitorando as atividades do dia-a-dia de diversos voluntários;
- Facilitar a aquisição do conjunto de treinamento personalizado para a pessoa monitorada;

- Implementar filtro para descartar registros que não pertencem as ADLs registradas no modelo, criando assim, um mecanismo capaz de ser utilizado em um ambiente real, exposto a diversos tipos de ruídos;
- Criação de arquitetura para possibilitar o monitoramento remoto do paciente por um cuidador;
- Utilizar séries temporais para prever antecipadamente a ADL que será executada pelo paciente, e dessa forma, interagir com ambientes inteligentes, possibilitando que esses se adaptem as ações que serão executadas.

REFERÊNCIAS BIBLIOGRÁFICAS

Anaconda (2016), 'Anaconda software distribution'. Available at https://www.anaconda.com/distribution/.

Anguita, D., Ghio, A., Oneto, L., Parra, X. & Reyes-Ortiz, J. L. (2012), 'Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine', *Ambient Assisted Living and Home Care* pp. 216–223. URL https://link.springer.com/chapter/10.1007/978-3-642-35395-6_30.

Banos, O., Damas, M., Pomares, H., Prieto, A. & Rojas, I. (2012), 'Daily living activity recognition based on statistical feature quality group selection', *Expert Systems with Applications - Elsevier* **39**, 8013–8021.

Barshan, B. & Yüksek, M. C. (2014), 'Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units', *The Computer Journal* **57**, 1649–1667. URL https://ieeexplore.ieee.org/document/8130901.

Benjamini, Y. & Yekutieli, D. (2001), 'The control of the false discovery rate in multiple testing under dependency', *Ann Statist* **29**, 1165–1188.

Breiman, L. (2001), 'Random forests', *Machine Learning* pp. 5–32.

Bruno, B., Mastrogiovanni, F., Sgorbissa, A., Vernazza, T. & Zaccaria, R. (2013), 'Analysis of human behavior recognition algorithms based on acceleration data', *IEEE International Conference on Robotics and Automation* pp. 1602–1607. URL https://ieeexplore.ieee.org/document/6630784.

Casale, P., Pujol, O. & Radeva, P. (2011*a*), 'Human activity recognition from accelerometer data using a wearable device', *Proceedings of the 5th Iberian conference on Pattern recognition and image analysis* **5**, 289–296. URL https://pdfs.semanticscholar.org/ed62/ea6986fcdefa42acbc46a1a8b7e42b11ce55.pdf.

Casale, P., Pujol, O. & Radeva, P. (2011*b*), 'Human activity recognition from accelerometer data using a wearable device', www.researchgate.net. URL https://www.researchgate.net/publication/221258784_Human_Activity_Recognition_from_Accelerometer_Data_Using_a_Wearable_Device.

Casilaria, E., Santoyo-Ramón, J. A. & Cano-García, J. M. (2017), 'Umafall: A multisensor dataset for the research on automaticfall detection', *The 14th International Conference on Mobile Systems and Pervasive Computing* pp. 32–29.

Cheon, S. M., Park, K. W. & Kim, J. W. (2015), 'Identification of daily activity impairments in the diagnosis of parkinson disease dementia', *Cognitive and behavioral neurology : official journal of the Society for Behavioral and Cognitive Neurology* p. 261–266.

Chetty, G. & White, M. (2016), 'Body sensor networks for human activity recognition', *2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN)* pp. 660 – 665. URL https://ieeexplore.ieee.org/document/7566779.

Christ, M., Braun, N., Neuffer, J. & Kempa-Liehr, A. W. (2018), 'Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package)', *Neurocomputing - Elsevier*.

Clinic, C. (2018), 'Alzheimer's disease: Assisting your loved one with activities of daily living', Cleveland Clinic. URL https://my.clevelandclinic.org/health/articles/9589-alzheimers-disease-assisting-with-your-loved-ones-activities-of-daily-living Accessed 17/10/2018.

Cover, T. & Hart, P. (1967), 'Nearest neighbor pattern classification', *IEEE Transactions on Information Theory* pp. 21–27. URL https://ieeexplore.ieee.org/document/1053964.

Drakos, G. (2012), 'Support vector machine vs logistic regression', Towards Data Science. URL https://towardsdatascience.com/support-vector-machine-vs-logistic-regression-94cc2975433f.

Dua, D. & Graff, C. (2017), 'UCI machine learning repository'. URL http://archive.ics.uci.edu/ml.

for Quality, I. & in Health Care, E. (2016), 'Everyday life with rheumatoid arthritis', NCBI. URL https://www.ncbi.nlm.nih.gov/books/NBK384458/, Accessed 10/09/2018.

Geurts, P., Ernst, D. & Wehenkel, L. (2006), 'Extremely randomized trees', *Machine Learning* pp. 3–42.

Hooman, O. M. J., Oldfield, J. & Nicolaou, M. A. (2019), 'Detecting early parkinson's disease from keystroke dynamics using the tensor-train decomposition', *European Signal Processing Conference*. URL https://ieeexplore.ieee.org/document/8902562.

Hunter, J. D. (2007), 'Matplotlib: A 2d graphics environment', *Computing in Science & Engineering* **9**(3), 90–95. URL https://aip.scitation.org/doi/abs/10.1109/MCSE.2007.55.

Jian, W., Zhiming, C., Victor, S., Yujie, S. & Pengpeng, Z. (2014), 'Mixed pattern matching-based traffic abnormal behavior recognition', *The Scientific World Journal* **2014**, 834013.

Khan, A. M., Lee, Y. K., Lee, S. Y. & Kim, T. S. (2010), 'Human activity recognition via an accelerometer-enabled-smartphone using kernel discriminant analysis', *5th International Conference on Future Information Technology*.

Kohavi, R. (1995), 'A study of cross-validation and bootstrap for accuracy estimation and model selection', *Appears in the International Joint Conference on Articial Intelligence (IJCAI)*.

Lewis, D. D. (1998), 'Naive (bayes) at forty: The independence assumption in information retrieval', *European Conference on Machine Learning* pp. 4–15. URL https://link.springer.com/chapter/10.1007/BFb0026666.

Machado, W. S., Barros, P. H., Aquino, A. L. L. & Almeida, E. S. (2018), 'Avaliação de técnicas de inteligência computacional para identificação de atividades de vida diária', *10º Simpósio Brasileiro de Computação Ubíqua e Pervasiva (SBCUP)* pp. 29–38.

Mahmoud, S., Lotfi, A. & Langensiepen, C. (2012), 'User activities outlier detection system using principal component analysis and fuzzy rule-based system', *ACM International Conference Proceeding Series* pp. 1–8.

Mahmoud, S., Lotfi, A. & Langensiepen, C. (2016), 'User activities outliers detection; integration of statistical and computational intelligence techniques', *Computational Intelligence* **32**, 49–71.

McKinney, W. (2010), *Data Structures for Statistical Computing in Python*.

Minyoung, S. (2015), *Electronic Textiles, 12 - Wearable sensors for athletes*, Elsevier.

Mlinac, M. E. & Feng, M. C. (2016), 'User activities outliers detection; integration of statistical and computational intelligence techniques', *Clinical Neuropsychology* **32**, 506–516.

Mulak, P. & Talhar, N. (2015), 'Analysis of distance measures using k-nearest neighbor algorithm on kdd dataset', *International Journal of Science and Research (IJSR)* pp. 2101–2104. URL https://www.ijsr.net/archive/v4i7/SUB156942.pdf.

Nations, U. (2014), 'Ageing well must be global priority', warns un health agency in new study', UN News. URL https://news.un.org/en/story/2014/11/483012#.VFyq6_nF-z4.

Pal, S. K. & Mitra, S. (1992), 'Multilayer perceptron, fuzzy sets, and classification', *IEEE Transactions on Neural Networks* **3**(5), 683–697.

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

Python (2017), 'Python language reference, version 3.6.4'. Available at https://docs.python.org/release/3.6.4/.

Pérez, F. & Granger, B. E. (2007), 'Ipython: A system for interactive scientific computing', *Computing in Science & Engineering* **9**(3), 21–29. URL https://aip.scitation.org/doi/abs/10.1109/MCSE.2007.53.

Raybaut, P. & Córdoba, C. (2009), 'Spyder project contributors'. Available at https://github.com/spyder-ide/spyder.

Russell, S. & Norvig, P. (2009), *Artificial Intelligence: A Modern Approach*, 3rd ed., Prentice Hall Press, Upper Saddle River, NJ, USA.

Sagha, H., Digumarti, S. T., Millán, J. R., Chavarriaga, R., Calatroni, A., Roggen, D. & Troster, G. (2011), 'Benchmarking classification techniques using the opportunity human activity dataset', *IEEE International Conference on Systems, Man, and Cybernetics* pp. 36–40. URL https://ieeexplore.ieee.org/document/6083628.

Shalev-Shwartz, S. & Ben-David, S. (2014), *Understanding Machine Learning:From Theory to Algorithms*, Cambridge University Press.

Starner, T., Weaver, J. & Pentland, A. (1998), 'Real-time american sign language recognition using desk and wearable computer based video', *Pattern Analysis and Machine Intelligence IEEE Transactions* **20**, 1371–1375. URL https://ieeexplore.ieee.org/document/735811.

Tapia, E. M., Intille, S. S. & Larson, K. (2004), 'Activity recognition in the home using simple and ubiquitous sensors', *Pervasive Computing, Proceedings* **3001**, 158–175. URL http://courses.media.mit.edu/2004fall/mas622j/04.projects/home/TapiaIntilleLarson04.pdf.

Troyer, A. K. (2011), *Activities of Daily Living (ADL)*, Springer New York, New York, NY, pp. 28–30. URL https://doi.org/10.1007/978-0-387-79948-3_1077.

Vapnik, V. & Cortes, C. (1995), 'Support-vector networks', *Machine Learning* pp. 273–297. URL http://image.diku.dk/imagecanon/material/cortes_vapnik95.pdf.

Vidhya, A. C. T. (2012), 'A complete tutorial on tree based modeling from scratch (in r & python)', Analytics Vidhya. URL https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/.

Walt, S., Colbert, S. C. & Varoquaux, G. (2011), 'The numpy array: A structure for efficient numerical computation', *Computing in Science & Engineering* **13**(2), 22–30. URL https://aip.scitation.org/doi/abs/10.1109/MCSE.2011.37.

Weiss, G. M., Timko, J. L., Gallagher, C. M., Yoneda, K. & Schreiber, A. J. (2016), 'Smartwatch-based activity recognition : A machine learning approach', *2016 IEEE-EMBS International Conference on Biomedical and Health Informatics* pp. 426–429.

Wilson, A. & Bobick, A. (1999), 'Parametric hidden markov models for gesture recognition', *Pattern Analysis and Machine Intelligence IEEE Transactions* **21**, 884–900. URL https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=790429.

Yamato, J., Ohya, J. & Ishii, K. (1992), 'Recognizing human action in time-sequential images using hidden markov model', *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* pp. 379–385. URL https://www.cs.sfu.ca/~mori/courses/cmpt888/summer10/papers/yamato_cvpr92.pdf.

Zhang, E. & Zhang, Y. (2009), *Encyclopedia of Database Systems*, Springer US, Boston, MA. URL https://doi.org/10.1007/978-0-387-39940-9_480.

Este trabalho foi redigido em LaTeX utilizando uma modificação do estilo IC-UFAL. As referências bibliográficas foram preparadas no JabRef e administradas pelo BibTeX com o estilo LaCCAN. O texto utiliza fonte Fourier-GUT*enberg* e os elementos matemáticos a família tipográfica Euler Virtual Math, ambas em corpo de 12 pontos.