Comparação entre modelos de Aprendizado de Máquina aplicados ao reconhecimento atividades humanas e mudanças de postura com dados de sensores de celular

Venicius G. R. Junior[1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
Av. Gal. Milton Tavares de Souza, s/nº, São Domingos – Niterói – RJ – Brazil – 24210-346

veniciusgrjr@gmail.com

Resumo. Esse trabalho analisa e compara o desempenho de algoritmos de Aprendizado de Máquina aplicados a uma base de dados de sensores acelerômetro e giroscópio, a fim de reconhecer atividades e transições entre atividades humanas. Os seguintes algoritmos foram empregados: *Voting Classifier, Logistic Regression, Extra Trees Classifier, Suport Vector Classifier, Random Forest Classifier, Naive Bayes* com Distribuição de Bernoulli, Naive Bayes com Distribuição Gaussiana, *Decision Tree Classifier, Ada Boost Classifier* combinado com *Suport Vector Classifier* e *Ada Boost Classifier*. Como medida de desempenho, esse estudo utilizou a acurácia, tempo de execução, erro médio absoluto nas etapas de treinamento e teste e testes em múltiplas sub bases de dados, geradas a partir da base principal, em um processo de *cross-validation* com *KFolds*, para identificar possíveis sub e super ajustes.

1. Introdução

Os modelos apresentados nesse trabalho foram desenvolvidos utilizando uma base de dados disponibilizada pela Universidade da California, Irvine (UCI). Essa base de dados foi construída por meio de um experimento com um grupo de 30 voluntários com idades entre 19 e 48 anos. Nesse experimento, os voluntários desempenharam um conjunto de atividades compostas por seis atividades básicas, três em repouso: em pé, sentado e deitado; e, três em movimento: caminhando, caminhando descendo escadas e caminhando subindo escadas.

O experimento também considerou transições entre as atividades estáticas: passando da posição em pé para sentado, da posição sentado para em pé, da posição sentado para deitado, da posição deitado para sentado, da posição em pé para deitado e da posição deitado para em pé. Todos os participantes estavam usando um celular no quadril durante a execução do experimento. Foram capturados dados de aceleração linear de sensores acelerômetro em três dimensões e velocidades angulares de sensores giroscópio, também capturadas em três eixos, a uma taxa constante de 50Hz.

Todo o experimento foi gravado em vídeo para tornar a tarefa de rotular os dados possível de ser realizada manualmente. Para o experimento, e aplicação dos modelos preditivos, a base de dados foi particionada em dois subconjuntos, sendo 70% dos dados utilizados para treinamento e 30% dos dados utilizados para teste.

Os sinais dos sensores (acelerômetro e giroscópio) foram pré-processados pela aplicação de filtros antirruído. O sinal do sensor de aceleração, responsável por aceleração gravitacional e componentes dos movimentos corporais, foi separado usando um filtro passa baixas. Para cada instância de dado, foram calculados 561 atributos com base em propriedades físicas e estatísticas dos movimentos, mais informações sobre esses atributos podem ser encontradas no arquivo features_info.txt, disponível em [UCI 2015]. Essa versão da base disponibiliza os dados originais dos sinais inerciais capturados dos sensores dos celulares, além dos dados pré-processados. Esse fato viabiliza testes com os dados originais. Além disso, novos rótulos foram introduzidos, adicionando as transições entre diferentes posturas, dados que não estavam disponíveis em uma primeira versão da base, também disponível em [UCI 2015].

2. Referencial Teórico e Revisão da Literatura

No cenário acadêmico atual, o reconhecimento da atividade humana se tornou um campo de pesquisa importante devido a suas contribuições com o objetivo de melhorar a qualidade de vida das pessoas: Computação Ubíqua, Inteligência Ambiental, Computação Pervasiva e Tecnologias Assistivas segundo [Chen et al. 2012], [Cook e Das 2012] e [Campbell e Choudhury 2012]. Essas áreas usam sistemas de reconhecimento de atividades como um instrumento que fornece informações sobre o comportamento e as ações das pessoas Segundo [Clarkson 2002].

Atualmente, existem muitos aplicativos em que os sistemas de reconhecimento de atividade humana são utilizados, por exemplo, no monitoramento contínuo de pacientes com problemas motores para diagnóstico de saúde e adaptação de medicamentos. Esse tipo de abordagem combina a captura de sinais de sensores de ambientes e sensores usados pelos usuários com o processamento por meio de algoritmos de aprendizado de máquina para classificação como mostrado por [Avci et al. 2010].

Vários sistemas de reconhecimento de atividade humana estão sendo propostos e pesquisados, incluindo áreas de locomoção, de atividades da vida diária, transporte, esportes e segurança, como abordado por [Nham et al. 2008] e [Tapia et al. 2004]. Esses sistemas são categorizados em relação à sua duração e complexidade, e as suas atividades são categorizadas em três grupos principais: eventos curtos, atividades básicas e atividades complexas. O primeiro grupo é composto por atividades de curta duração (da ordem de segundos), como transições posturais (por exemplo, sentar-se) e gestos corporais segundo [Mannini e Sabatini 2010]. Atividades básicas são caracterizadas por uma duração mais longa e podem ser dinâmicas ou estáticas (por exemplo, corrida, leitura) segundo [Bao e Intille 2004]. O último grupo, atividades complexas, é composto por progressões de atividades mais simples acima mencionadas e envolvem aspectos como interação com objetos e outras pessoas (por exemplo, praticar esportes, atividades sociais) segundo [Aggarwal e Ryoo 2011]. O presente trabalho aplica algoritmos abordando as duas primeiras categorias: eventos curtos e atividades básicas.

Sensores de ambientes e sensores que os usuários vestem têm sido ativamente explorados por trabalhos relacionados ao reconhecimento de atividades humanas. Câmeras de vídeo, microfones, sensores GPSs e sensores para medir a proximidade, movimento corporal e sinais vitais são apenas alguns exemplos. A pesquisa atual sobre sensores de ambiente tem se concentrado principalmente em câmeras de vídeo devido à

facilidade de recuperação de informações visuais do ambiente. Câmeras também foram combinadas com sensores acelerômetros e microfones e introduzidas recentemente em tecnologias vestíveis para novas aplicações onipresentes segundo [Behera et al. 2015].

Após a capturas de movimentos, realizada por uma, ou pela combinação de mais de uma das formas mencionadas nesse estudo, os dados coletados são analisados. Esses dados podem passar por processos de pré-processamento, por processos de exploração e adequação, para que finalmente sejam desenvolvidos modelos que possibilitem a identificação de determinadas atividades, tomando como base dados rotulados previamente. O estudo atual, aproveitou as primeiras etapas de aquisição de dados, de observação do experimento e de rotulação dos dados adquiridos de um estudo realizado por pesquisadores da Universidade da California, Irvine (UCI) [UCI 2015]. Com as etapas de realização de experimento e aquisição de dados consideradas realizadas, esse estudo focou na análise dos dados e na aplicação de modelos e algoritmos da área de Aprendizado de Máquina.

Os algoritmos *Random Forest* (RF) [Breiman 2001] e *Extremely Randomized Trees* (ERT) [Geurts et al. 2006] são algoritmos de classificação muito eficientes com base em árvores de decisão e fornecem estimativas da importância dos atributos, sendo indicado para a classificação de recursos. Esses algoritmos foram testados e aplicados sobre os dados adquiridos. Ao final do processo, os resultados produzidos foram comparados. O *Random Forest* gera várias árvores de decisão, amostrando aleatoriamente instâncias de treinamento do conjunto de dados e selecionando aleatoriamente m recursos de cada amostra, onde M é o número total de recursos por instância e m <M. A ramificação das árvores é realizada encontrando-se a melhor divisão entre os recursos m em cada nó. No processo de classificação, cada árvore vota na classe e a classe majoritária é escolhida. Por outro lado, a ERT escolhe os pontos de divisão de recursos aleatoriamente. Isso aumenta a velocidade do treinamento porque o número de cálculos por nó diminui. A velocidade do ERT em relação ao RF pode ser comprovada nesse experimento. Ambos algoritmos fornecem excelente desempenho de classificação e podem treinar modelos em conjuntos de dados muito grandes. O valor do parâmetro m usado no RF e ERT foi o padrão por sua implementação na biblioteca scikit-learn [Pedregosa et al. 2011]. Não foi observado qualquer ganho significativo ajustando este parâmetro.

O algoritmo de Regressão Logística (LR) [Hosmer et al. 2013] também foi utilizado nesse estudo, pois é simples e rápido. Além disso, o LR fornece fácil interpretação dos modelos, fornece uma estimativa da importância dos atributos e, de forma simples, é possível utilizar recursos de computação paralela, o que aumenta sua velocidade de execução.

O algoritmo *Support Vector Machine* (SVM) *Classifier* [Cortes e Vapnik 1995] também foi usado. Esse modelo é composto por um conjunto de métodos de aprendizado supervisionado usados para classificação, regressão e detecção de outliers segundo [Cortes e Vapnik 1995]. É um classificador muito poderoso, mas para conjuntos de dados maiores, é necessário muito tempo para a construção de modelos, fator que em determinados casos é decisivo. Foi usado nesse estudo, pois, ainda segundo [Cortes e Vapnik 1995], esse algoritmo é muito eficiente em espaços com número de dimensões elevado. O ajuste de parâmetros de SVMs é recomendado, pois aumenta significativamente a precisão da classificação e reduz o *overfitting* segundo [Lameski et al. 2015].

O algoritmo *Naive Bayes* (NB) [Rish 2001] também foi usado nesse estudo. Foram utilizadas funções de distribuição Gaussiana e de Bernoulli para os modelos testados. Os algoritmos *Naive Bayes* são um conjunto de algoritmos de aprendizado supervisionado com base na aplicação do teorema de *Bayes* com a suposição "ingênua" de independência condicional entre cada par de instância de dado, dado o valor da variável da classe. Apesar de suas suposições aparentemente simplificadas, os classificadores *Naive Bayes* têm obtido bom desempenho em muitas situações do mundo real, como classificação de documentos e filtragem de spam. Eles exigem uma pequena quantidade de dados de treinamento para estimar os parâmetros necessários segundo [Pedregosa et al. 2011].

Esse estudo também utilizou o classificador *Adaptive Boosting (AdaBoost)* [Freund e Schapire 1995]. Esse algoritmo é um meta-estimador que começa ajustando um classificador no conjunto de dados original e depois ajusta cópias adicionais do classificador no mesmo conjunto de dados, mas ajustando os pesos das instâncias classificadas incorretamente, para que os classificadores subsequentes se concentrem mais em casos difíceis, ainda segundo [Freund e Schapire 1995]. Nos últimos anos, os algoritmos de reforço (*boosting)* se tornaram mais populares nas competições de ciência de dados ou aprendizado de máquina. A maior parte dos vencedores dessas competições usam algoritmos de reforço para obter melhor desempenho para seu modelo. Os algoritmos de reforço combinam vários modelos com um rendimento não muito elevado para criar modelos de alta precisão. Algoritmos como AdaBoost, Gradient Boosting e XGBoost, são algoritmos de aprendizado de máquina amplamente utilizados para vencer as competições de ciência de dados segundo [Nielsen 2016].

Além desses algoritmos mencionados, esse estudo também utilizou um algoritmo de arvore de decisão (*Decision Tree - DT*) [Breiman et al. 1984]. Esse algoritmo é considerado um método de aprendizado supervisionado não paramétrico usado para classificação e regressão. O objetivo é criar um modelo para prever o valor de uma variável de destino, aprendendo regras simples de decisão inferidas a partir dos atributos dos dados. Esse algoritmo foi escolhido para esse estudo, pois arvores de decisão são simples de se interpretar e de se entender, não precisam de muita preparação nos dados, o custo computacional é da ordem de logaritmo de n, é capaz de utilizar dados numéricos e categóricos, além disso, árvores podem ser visualizadas segundo [Pedregosa et al. 2011].

3. Metodologia da Análise Experimental

Primeiramente, foi escolhida a linguagem de programação Python em sua versão 3 para a manipulação dos dados e execução dos modelos preditivos. Para facilitar a edição do código fonte e a visualização de seus resultados após execução, optou-se por utilizar a ferramenta Ipython Notebook, disponível no pacote de programas para Windows chamado Anaconda. Todas as análises foram feitas utilizando um computador com o sistema operacional Windows, processador Intel core i5 de 5ª geração com velocidade de 2,40 GHz, 2 núcleos físicos, 4 núcleos virtuais, 16 GB de memória RAM DDR3.

Foi realizada uma análise exploratória da base de dados utilizada nesse experimento, para encontrar possíveis pontos de falha, dados faltando, para observar o formato de cada atributo e para adequar a base de dados a cada algoritmo que irá utilizá-la. Os dados da base utilizada nesse experimento foram disponibilizados pela Universidade da

California, Irvine (UCI), e faziam parte de um estudo para identificar diferentes tipos de postura e diferentes tipos de transição entre posturas, ou posições em que os participantes da pesquisa se encontravam.

Após essa etapa de exploração e adequação dos dados, diversos modelos da área de Aprendizado de Máquina foram aplicados. Todos os modelos aplicados podem ser vistos na tabela 1. Durante a aplicação de cada modelo, seus parâmetros foram testados e ajustados, com a finalidade de encontrar o conjunto de parâmetros mais adequado para que o modelo obtivesse boa performance com a sua proposta de classificação. Todos os códigos podem ser encontrados em https://github.com/veniciusgrjr/hapt-ml

Em outro momento, com os resultados encontrados pelos algoritmos, realizou-se uma comparação entre os valores de resultados obtidos, para encontrar o modelo que mais se adequa aos dados e às previsões desejadas.

4. Resultados Obtidos

O algoritmo *Decision Tree Classifier* apresentou desempenho razoável. A simplicidade do algoritmo está de acordo com a performance do modelo preditivo gerado, com 80,20% de acerto na classificação, ocupando a 10ª posição em comparação com os algoritmos utilizados nesse trabalho, como é possível observar na tabela 1. Quanto ao tempo de execução, 5,71 segundos, foi um algoritmo que não se destacou, porém para aplicações em tempo real esse valor seria considerado ruim. Analisando os erros no treinamento e no teste, 0,000 e 0,313, respectivamente, observa-se um erro baixo nos dados de treinamento, indicando ajuste elevado, talvez excessivo, e um pouco de *overfitting* se compararmos o erro no teste com o no treinamento. Esse valor não é considerado um *overfitting* elevado.

O classificador *Naive Bayes* apresentou desempenho razoável em comparação com os outros algoritmos utilizados nesse trabalho, ocupando 9ª posição utilizando a distribuição de Bernoulli e ocupando a 11ª posição utilizando a distribuição Gaussiana, de acordo com os resultados mostrados na tabela 1. Um dos fatores mais importante desse algoritmo é a sua velocidade de execução, como pode ser observado na tabela 1, ocupando as primeiras posições quando analisado em relação à velocidade de execução. Analisando os erros no treinamento e no teste, utilizando a distribuição de Bernoulli e a distribuição Gaussiana, 0,237 e 0,366, 0,393 e 0,45, respectivamente, observa-se um pouco de *underfitting* e um pouco de *overfitting* se compararmos o erro no teste com o no treinamento. Nessa modelo, o que mais se destaca é o *underfitting*, expressando resultados pouco confiáveis.

O algoritmo *Random Forest Classifier* obteve também um desempenho razoável com 90,48% de porcentagem de acerto na classificação, utilizando seus processos de votação e aprendizado com “florestas aleatórias”. Quanto a velocidade de execução seu desempenho também não se destacou entre os mais rápidos, levando 6,52 segundos para realizar sua classificação. A característica de lentidão de alguns algoritmos dificulta sua aplicação em modelos preditivos em tempo real. Um ponto positivo para o uso desse algoritmo nesse trabalho foi sua versatilidade, pois pode receber entradas binárias, categóricas ou numéricas segundo [Pedregosa et al. 2011]. Analisando os erros no treinamento e no teste, 0,000 e 0,080, respectivamente, observa-se um erro baixo nos dados de treinamento, indicando ajuste elevado, talvez excessivo, e um pouco de *overfitting* se compararmos o erro no teste com o no treinamento. Essa diferença entre

os erros não é considerada um *overfitting* elevado. No trabalho, foram feitos testes com *cross-validation* utilizando *KFold*, dividindo os dados em 20 *folds*, e realmente não apresenta *overfitting* elevado.

O algoritmo *Extremely Randomized Trees*, mesmo tendo semelhanças com o algoritmo *Random Forest*, apresentou desempenho um pouco mais elevado, atingindo a taxa de previsão correta de 92,28%, ocupando a 6ª posição na classificação por performance. Assim como o *Random Forest*, esse algoritmo utiliza certas decisões de forma aleatória, porém apresentou velocidade de execução (2,87 s) maior do que a velocidade de execução do *Random Forest*. Analisando os erros no treinamento e no teste, 0,000 e 0,124, respectivamente, observa-se um erro baixo nos dados de treinamento, indicando ajuste elevado, talvez excessivo, e um pouco de *overfitting* se compararmos o erro no teste com o no treinamento. Essa diferença entre os erros não é considerada um *overfitting* elevado. No trabalho, foram feitos testes com *cross-validation* utilizando *KFold*, dividindo os dados em 20 *folds*, e realmente não apresenta *overfitting* elevado.

Embora esse seja um problema de classificação e não de regressão, este trabalho optou por utilizar o algoritmo *Logistic Regression, com certas transformações nos dados para que fosse possível*. Esse algoritmo obteve um desempenho acima da média, atingindo a taxa de acerto de 94,52%. Em relação ao tempo de execução, seu desempenho não atingiu níveis de destaque de acordo com os algoritmos utilizados nesse trabalho. Analisando os erros no treinamento e no teste, 0,014 e 0,096, respectivamente, observa-se um erro baixo nos dados de treinamento, e um pouco de *overfitting* se compararmos o erro no teste com o erro no treinamento. Essa diferença entre os erros não é considerada um *overfitting* elevado.

O algoritmo *Support Vector Machine* obteve desempenho razoável, atingindo a taxa de acerto de 91,80%, ficando classificado na 7ª posição em relação aos outros algoritmos. Quanto à velocidade de execução, seu desempenho não se destacou, atingindo a 16,61 segundos. Esse algoritmo ofereceria grandes latências se fosse usado em sistemas em tempo real. Analisando os erros no treinamento e no teste, 0,090 e 0,141, respectivamente, observa-se um erro baixo nos dados de treinamento, e um pouco de *overfitting* se compararmos o erro no teste com o no treinamento. Essa diferença entre os erros não é considerada um *overfitting* significativo.

O algoritmo *AdaBoostClassifier* não obteve desempenho significativo quando comparado ao desempenho de outros algoritmos abordados nesse trabalho. Seu desempenho foi de 51,96%, utilizando o estimador base padrão, ocupando a 13ª posição. O desempenho desse algoritmo aumentou quando o *Suport Vector Classifier* foi utilizado como estimador base, obtendo um desempenho de 69,48% e ocupando a 12ª posição, porém o seu tempo de execução cresceu muito com essa combinação, chegando a 25587,35 segundos. Analisando os erros no treinamento e no teste, 1,630 e 1,774, respectivamente, observa-se um caso de *underfitting.*

Os algoritmos que obtiveram melhor desempenho nesse trabalho foram os do tipo *Voting Classifier*, que são gerados pela combinação de outros algoritmos, com seus respectivos pesos. Essa alternativa de combinar múltiplos algoritmos em um processo de votação em busca de aumento na acurácia é utilizada com frequência em competições de análise de dados. Esse aumento de performance por meio da combinação de algoritmos pôde ser observado nesse trabalho também, de acordo com a tabela 1. A combinação que obteve melhor resultado foi a composta pelos algoritmos

Logistic Regression com peso 5, *Support Vector Classifier* com peso 1 e *Extra Trees Classifier* com peso 3, atingindo o desempenho de 95,00% de acerto. Essa combinação de modelos ganha na acurácia, porém perde no tempo elevado de execução, não sendo indicada para aplicações em tempo real. Analisando os erros no treinamento e no teste, 0,004 e 0,060, respectivamente, observa-se um erro baixo nos dados de treinamento, indicando ajuste elevado, talvez excessivo. Essa diferença entre os erros não é considerada um *overfitting* significativo. No trabalho, foram feitos testes com *cross-validation* utilizando *KFold*, dividindo os dados em 20 *folds*, em um dos *folds* foram encontrados 0,004 e 0,186, para os erros de treinamento e teste, respectivamente, mostrando um *overfitting* pequeno para essa iteração.

Tabela 1. Performance dos algoritmos analisados

Classi-ficação	Algoritmo	Performance (%)	Tempo de execussão (Segundos)	Erro no treinamento	Erro no teste
1º	VotingClassifier com: LogisticRegression(peso 5), SVC(peso 1), ExtraTreesClassifier(peso 3)	95,00	153,84	0,004	0,060
2º	VotingClassifier com: LogisticRegression(peso 3), SVC(peso 1), ExtraTreesClassifier(peso 2)	94,90	149,94	0,003	0,062
3º	VotingClassifier com: LogisticRegression(peso 1), SVC(peso 1), ExtraTreesClassifier(peso 1)	94,84	153,96	0,004	0,058
4º	Logistic Regression	94,52	11,56	0,014	0.096
5º	VotingClassifier com: DecisionTreeClassifier, LogisticRegression, GaussianNB, RandomForestClassifier, SVC, ExtraTreesClassifier	92,97	171,20	0,010	0,065
6º	Extremely Randomized Trees	92,28	2,98	0,000	0,124
7º	Support Vector Machine	91,80	16,64	0,090	0,141
8º	RandomForestClassifier	90,48	6,52	0,000	0,080
9º	Naive Bayes Distribuição de Bernoulli	82,41	0,16	0,237	0,366
10º	Decision Tree Classifier	80,20	5,71	0,000	0,313
11º	Naive Bayes Distribuição	74,73	0,54	0,393	0,456

	Gaussiana				
12º	AdaBoostClassifier + Suport Vector Classifier	69,48	25587,35	---	---
13º	AdaBoostClassifier	51,96	29,36	1,630	1,774

5. Conclusões e Trabalhos Futuros

Esse trabalho analisou o desempenho de diversos algoritmos de Aprendizado de Máquina aplicados a uma base de dados de sensores acelerômetro e giroscópio, dados mencionados nesse estudo. O algoritmo que obteve melhor desempenho foi o *Voting Classifier*, que foi gerado pela combinação dos seguintes algoritmos: *Logistic Regressiom* com peso 5, *Support Vector Classifier* com peso 1 e *Extra Trees Classifier* com peso 3, atingindo o desempenho de 95,00% de acerto, com baixa taxa de *overfitting*, porém com tempo de execução um pouco elevado, 153,84 segundos. Para aplicações em tempo real, o algoritmo mais indicado seria o Naive Bayes com Distribuição de Bernoulli, com tempo de execução de 0,16 segundos e acurácia de 82,41 segundos.

Como sugestão de trabalhos futuros, seria interessante repetir o experimento com um grupo de pessoas diferentes, nas mesmas condições do inicial, para que uma outra base fosse disponibilizada para testes, para novos ajustes de hiper parâmetros dos modelos usados e para identificação de sub e super ajuste nos modelos testados.

Referências

Boulic, R. and Renault, O. (1991) “3D Hierarchies for Animation”, In: New Trends in Animation and Visualization, Edited by Nadia Magnenat-Thalmann and Daniel Thalmann, John Wiley & Sons ltd., England.

Dyer, S., Martin, J. and Zulauf, J. (1995) “Motion Capture White Paper”, http://reality.sgi.com/employees/jam_sb/mocap/MoCapWP_v2.0.html, December.

Holton, M. and Alexander, S. (1995) “Soft Cellular Modeling: A Technique for the Simulation of Non-rigid Materials”, Computer Graphics: Developments in Virtual Environments, R. A. Earnshaw and J. A. Vince, England, Academic Press Ltd., p. 449-460.

Knuth, D. E. (1984), The TeXbook, Addison Wesley, 15th edition.

Smith, A. and Jones, B. (1999). On the complexity of computing. In *Advances in Computer Science*, pages 555–566. Publishing Press.

Jorge L. Reyes-Ortiz, Luca Oneto, Xavier Parra, and Davide Anguita. 2016. Transition-aware human activity recognition using smartphones. Neurocomputing 171(C), 754--767.

L. Chen, J. Hoey, C. Nugent, D. Cook, Z. Yu, Sensor-based activity recognition, IEEE Transactions. on Systems, Man, and Cybernetics, Part C: Applications and Reviews 42 (2012) 790–808.

D. J. Cook, S. K. Das, Pervasive computing at scale: Transforming the state of the art, Pervasive. Mobile Computing 8 (2012) 22–35.

A. Campbell, T. Choudhury, From smart to cognitive phones, IEEE Pervasive Computing (2012).

B. P. Clarkson, Life patterns: structure from wearable sensors, Ph.D. thesis, Massachusetts Institute of Technology (2002).

A. Avci, S. Bosch, M. Marin-Perianu, R. Marin-Perianu, P. Havinga, Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey, in: International Conference on Architecture of Computing Systems, 2010.

B. Nham, K. Siangliulue, S. Yeung, Predicting mode of transport from iphone accelerometer data, Tech. rep., Tech. report, Stanford Univ (2008).

E. Tapia, S. Intille, K. Larson, Activity recognition in the home using simple and ubiquitous sensors, in: Pervasive Computing, 2004.

A. Mannini, A. M. Sabatini, Machine learning methods for classifying human physical activity from on-body accelerometers, Sensors 10 (2010) 1154–1175.

L. Bao, S. Intille, Activity recognition from user-annotated acceleration data, in: Pervasive Computing, 2004.

J. Aggarwal, M. S. Ryoo, Human activity analysis: A review, ACM Computing Surveys 43 (2011) 16.

UCI, https://archive.ics.uci.edu/ml/datasets/SmartphoneBased+Recognition+of+Human+Activities+and+Postural+Transitions (2015), acessado em 06/06/2020.

A. Behera, D. Hogg, A. Cohn, Egocentric activity monitoring and recovery, in: Asian Conference on Computer Vision, 2013.

L. Breiman, ‘‘Random forests,’’ Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.

P. Geurts, D. Ernst, and L. Wehenkel, ‘‘Extremely randomized trees,’’ Mach. Learn., vol. 63, no. 1, pp. 3–42, 2006.

F. Pedregosa et al., ‘‘Scikit-learn: Machine learning in Python,’’ J. Mach. Learn. Res., vol. 12, pp. 2825–2830, Oct. 2011.

D. W. Hosmer, Jr., S. Lemeshow, and R. X. Sturdivant, Applied Logistic Regression, vol. 398. Hoboken, NJ, USA: Wiley, 2013.

C. Cortes and V. Vapnik, ‘‘Support-vector networks,’’ Mach. Learn., vol. 20, no. 3, pp. 273–297, 1995.

P. Lameski, E. Zdravevski, R. Mingov, and A. Kulakov, ‘‘SVM parameter tuning with grid search and its impact on reduction of model over-fitting,’’ in Rough Sets, Fuzzy Sets, Data Mining, Granular Computing. Cham, Switzerland: Springer, 2015, pp. 464–474.

I. Rish, ‘‘An empirical study of the naive Bayes classifier,’’ in Proc. IJCAI Workshop Empirical Methods Artif. Intell., vol. 3. Menlo Park, CA, USA, 2001, no. 22, pp. 41–46.

Y. Freund, R. Schapire, “A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting”, 1995.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and Regression Trees. Wadsworth, Belmont, CA, 1984.

D. Nielsen, "Tree Boosting With XGBoost Why Does XGBoost Win 'Every' Machine Learning Competition?", 2016.