

PH125_9 Choose Your Own Capstone Project

Mario De Toma

May 26, 2019

Introduction

Objectives

Motivation) This project has been conducted as part of the Data Science Professional Certification path provided by HarvardX, an online learning initiative of Harvard University through edX. In particular this is the second data science project to submit for PH125.9x Course denominated “Data Science: Capstone”. The name of the project, Choose Your Own, is due to the fact that the dataset under analysis could be chosen by the learner from public available datasets.

I chose the Fall Detection dataset from the curated list of datasets at the following link https://www.kaggle.com/annavictoria/ml-friendly-public-datasets?utm_medium=email&utm_source=intercom&utm_campaign=data+projects+onboarding, a repository indicated by Course Staff.

Project objective) As per course project introduction the project aim is to apply machine learning techniques that go beyond standard linear regression. In particular the task of this project is multi class classification i.e. where the outcome variable is categorical with more than 2 classes. Specifically the problem statement is related to predict the type of activity among 6 different activities of daily living (ADLs) on the basis of monitored medical measures obtained through sensors worn by elder people as recorded in the dataset analysed.

Research question) The research question can be stated as: is it possible to predict 6 activities of daily living (ADLs) including Standing , Walking, Sitting, Falling, Cramps, Running from few predictors monitoring health status?

This project is not intended to solve the problem of “Fall Detection”, as the name of the dataset could suggest, which could have been settled by binarysing the ADL information as Falling vs all the remaining activities.

Furthermore note that no causal inference claim can be raised after this study that focus only on supervised learning.

Dataset) The Fall detection dataset of Chinese hospitals of old age patients [1] is hosted by kaggle. It reports 16382 observations containing the ADL label and related 6 numerical predictors measuring the medical characteristics monitored.

Background

Starting point for conducting this study was the supervised leaning process as decribed by Professor Rafael Irizarry in the PH125_8 edX course on Machine Learning and in his book: Introduction to Data Science [2]. In particular this study was conducted using the R package ‘caret’ as the framework for machine learning [3].

Overview and outline

The study demonstrates that ADLs can be predicted and that the reachable accuracy of prediction depends on the model chosen.

The rest of this report is articulated in the following sections:

- *Methods*: in this section the dataset is explored in order to find useful insight, then the design of the study is explained and different models are proposed. Finally the modeling is described in details.
- *Results*: this section shows actual results achieved and compares diverse models evaluation. Furthermore, in order to explain the local behavior of the best model, the ceteris paribus analysis procedure is shown
- *Conclusions*: the section summarizes achievement, it discusses the project validity and it indicates potential model improvements
- *Reproducibility*: this section provides information related to the reproducibility of the analysis including computation considerations, HW and SW stack used.

Methods

In order to answer the research question posed in the introduction section, the dataset Fall Detection has been analyzed and then the machine learning experiments conducted using different machine learning techniques adequate for the multi class classification task.

Exploratory Data Analysis

Fall detection dataset has been downloaded from kaggle, put on my github and then loaded into R and partitioned such that 70% of the observations belong to the training set (fall_train) and 30% the test set (fall_test). The caret::createDataPartition function has been used in order to maintain the class distribution between train and test set so that the main assumption of every prediction study is respected: train, test and future “production” data all come from the same data generating process producing the same distribution.

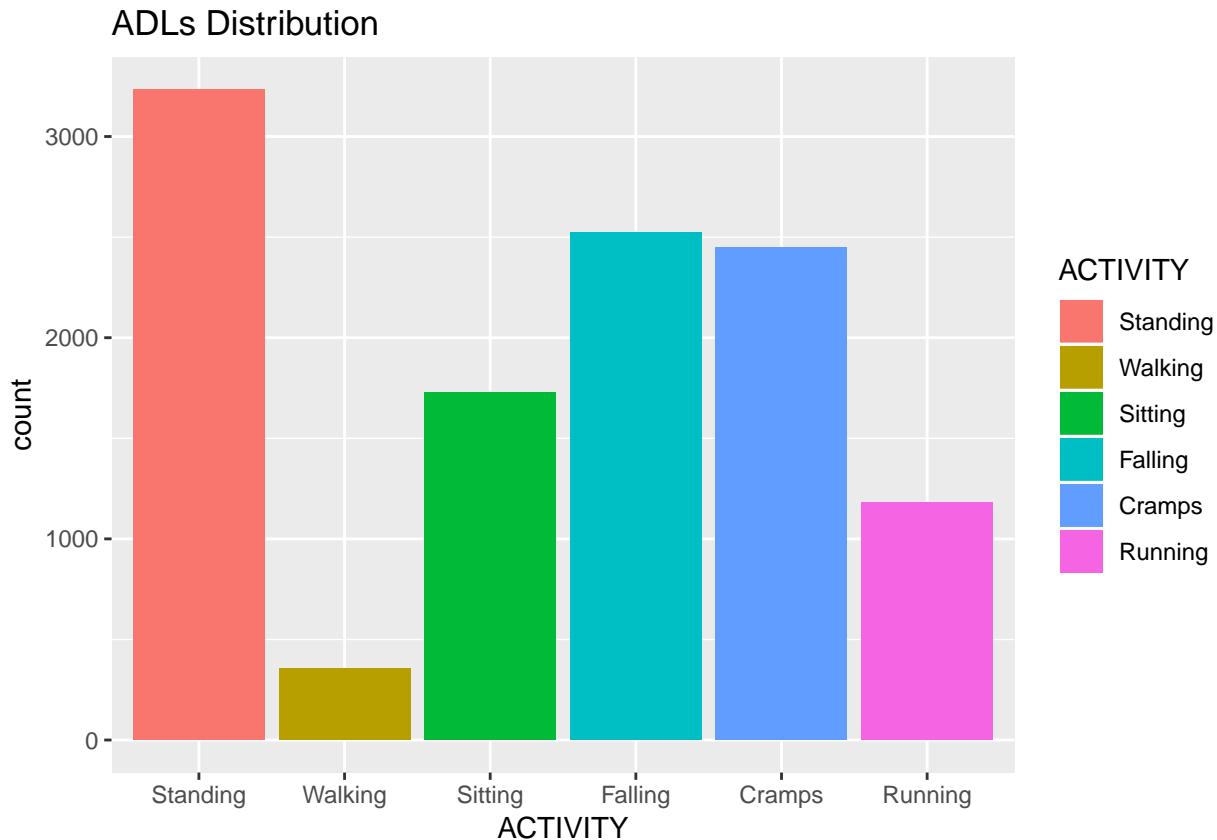
Fall Detection dataset contains the following variables:

```
## Observations: 11,468
## Variables: 7
## $ ACTIVITY    <dbl> 3, 2, 2, 4, 4, 5, 3, 4, 5, 0, 4, 3, 3, 0, 0, 3, 2, ...
## $ TIME        <dbl> 4722.92, 4059.12, 4773.56, 8271.27, 7102.16, 7015....
## $ SL          <dbl> 4019.64, 2191.03, 2787.99, 9545.98, 14148.80, 7336...
## $ EEG          <dbl> -1600.000, -1146.080, -1263.380, -2848.930, -2381....
## $ BP           <dbl> 13, 20, 46, 26, 85, 22, 35, 82, 61, 59, 44, 16, 53...
## $ HR           <dbl> 79, 54, 67, 138, 120, 95, 157, 315, 214, 104, 156, ...
## $ CIRCLUATION <dbl> 317, 165, 224, 554, 809, 427, 1519, 5844, 1469, 65...
```

The outcome variable ACTIVITY is numeric. In order to enable classification it has been converted to a factor with the following levels: Standing, Walking, Sitting, Falling, Cramps, Running.

ADL Distribution

The ADL classes are not evenly distributed. In particular Walking ADL has few observations. This could make our multi class classification task harder.

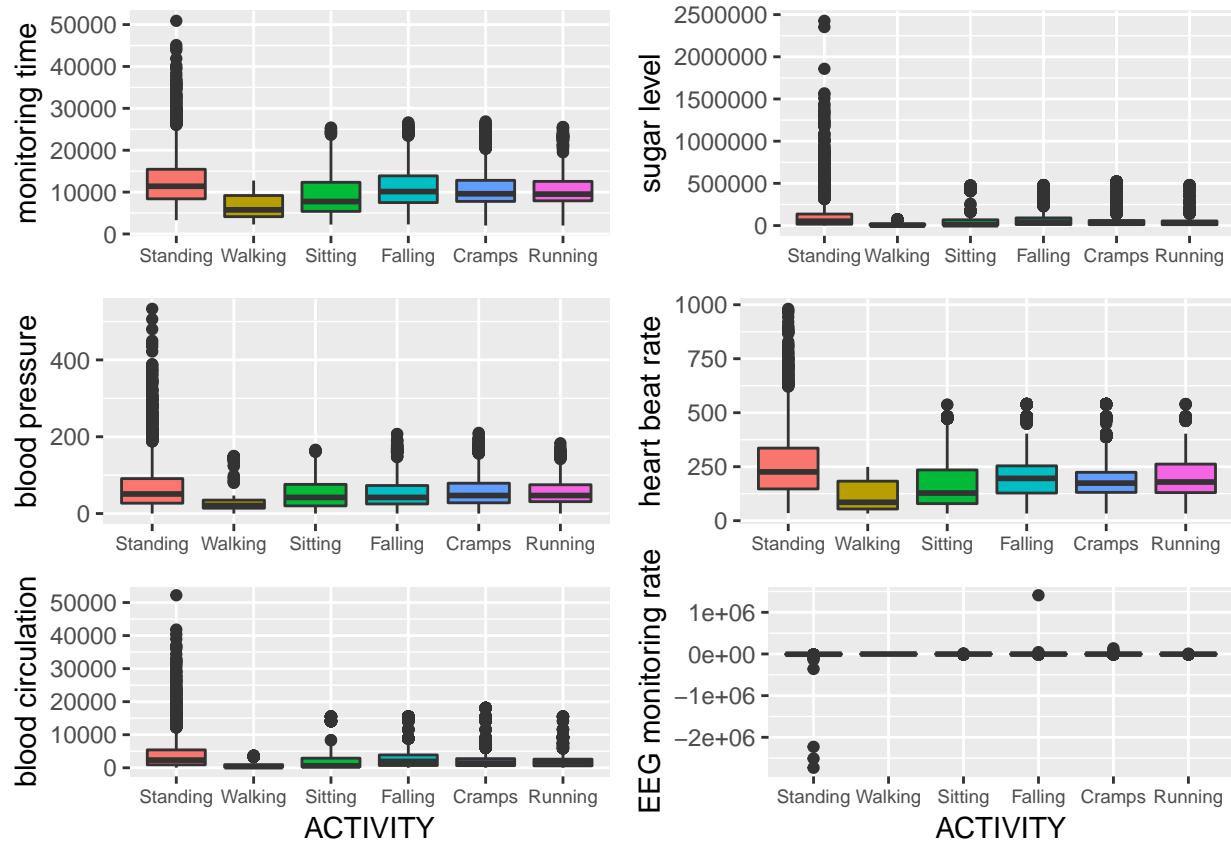


Predictors discriminative power

Predictors for ADL in Fall Detection dataset are all numeric and they specifically are:

- TIME monitoring time
- SL sugar level
- EEG monitoring rate
- BP blood pressure
- HR Heart beat rate
- CIRCLUATION Blood circulation (the typo in the original dataset has been kept)

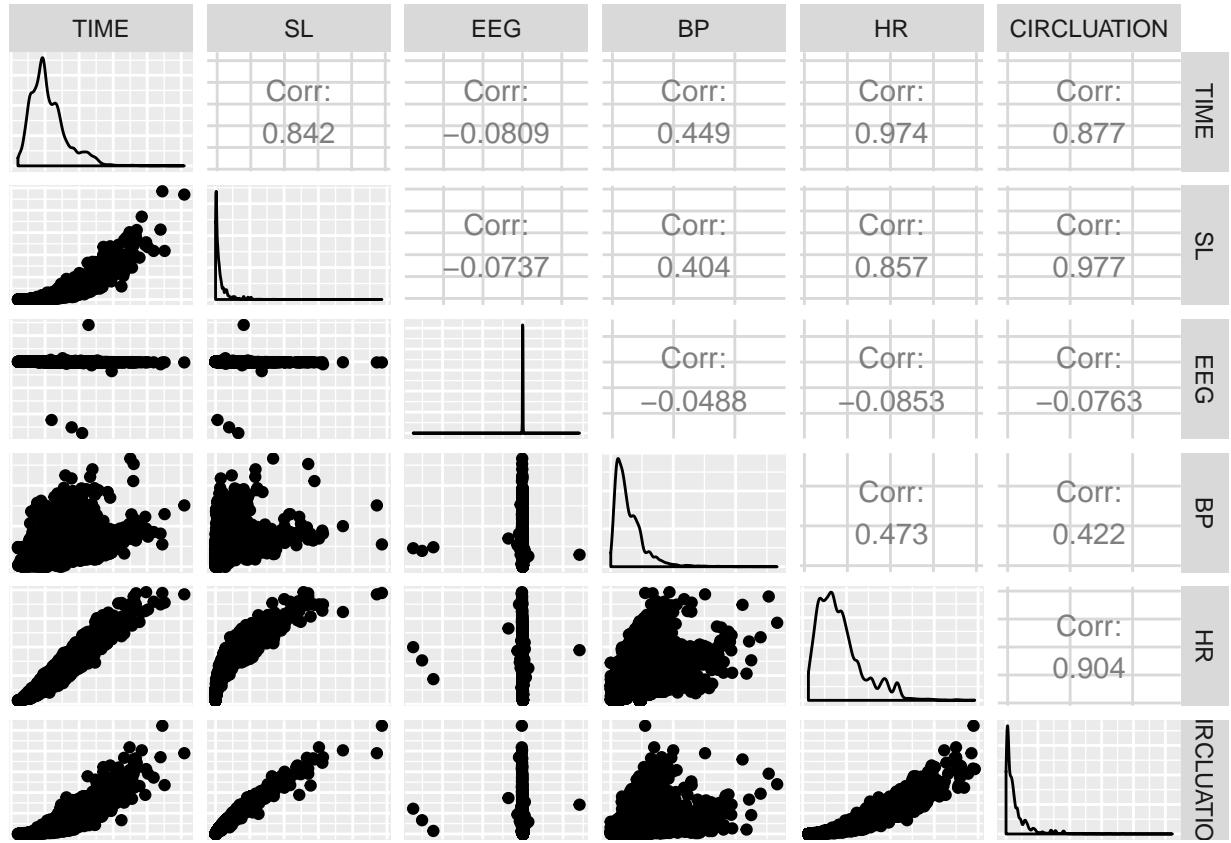
In order to check single predictor capacity to discriminate among classes, the boxplot by class for each predictor has been drawn.



Considering the predictor by class distributions overlapping, the discriminative power of single predictors does not seem to be enough to classify ADL with a good accuracy.

Predictors correlation

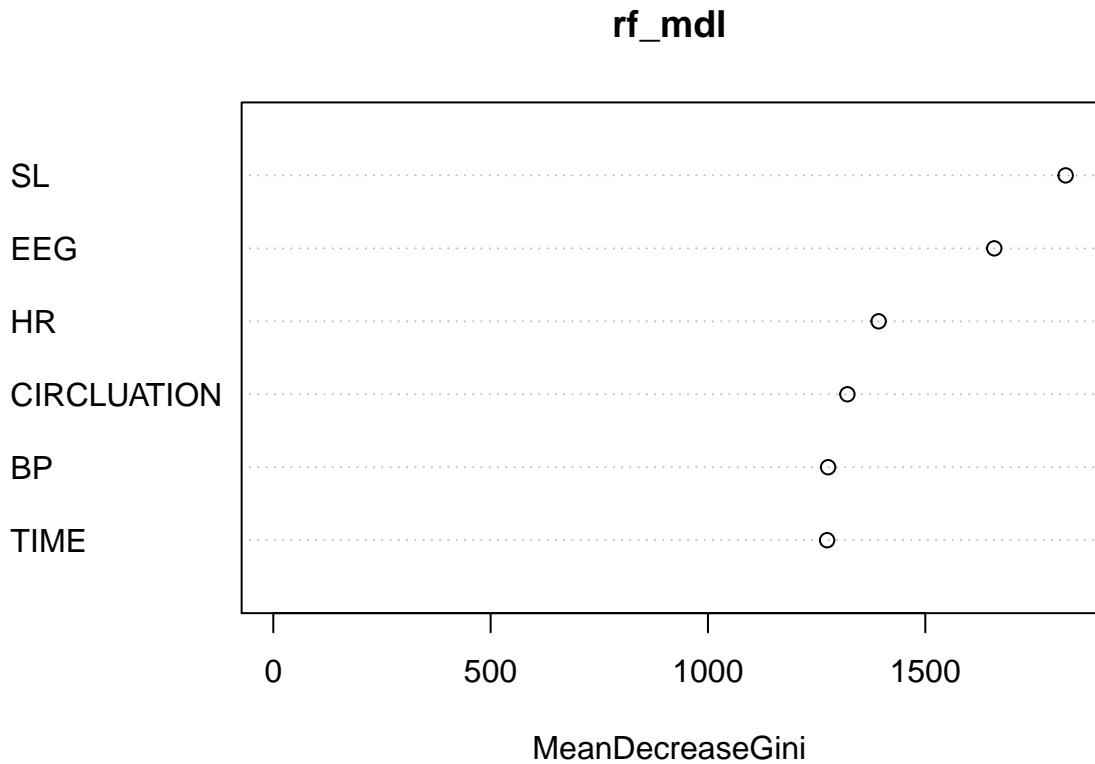
Further element of analysis is related to the high level of correlation among predictors.



In particular SL (sugar level) and CIRCLUATION (blood circulation) with a correlation of 0.977 and TIME (monitoring time) and HR (heart beat rate) with a correlation of 0.974 are almost colinear. Also HR and CIRCULATION (0.904), HR with SL (0.857) and TIME with CIRCLUATION (0.877) correlations are really high.

Feature importance

The feature importance has been therefore investigated through the random forest algorithm which provide as a side outcome the importance of a feature in discriminating one class from the other. In other words, for random forest algorithm it is easy to compute how much each variable is contributing to the classification decision.



From the plot, SL results the most important predictor followed by EEG but all 6 predictors are of great help in the classification attempt since each contributes to decrease the GINI index. This index measures the decrease in node impurities from splitting on the variable, averaged over all trees.

Proposed models

In order to accomplish the multi class classification task the following models have been tried:

- multinomial
- k Nearest Neighbors
- random forest

All the models are natural choices for multi class classification.

Study design

The study has been conducted using the training set, fall_train, for training the model and tuning hyper-parameters through cross validation while the final accuracy has been evaluated on the test set, fall_test.

The test dataset has not been used in any former phase of the study and therefore it could simulate new data and it allowed to evaluate the capacity of the model to generalize .

Accuracy metric has been used to evaluate and compare different models. For each model the confusion matrix has been produced in order to evaluate which class is harder to identify.

Cross validation has been configured in caret machine learning framework with 5 folds (80% for training, 20% for validation).

```
ctrl <- trainControl(method = 'cv', number = 5, p = .8)
```

Cross validation lead to long computation times because in the defined study design the model has to be trained and validated 5 times. Therefore cross validation computation has been parallelized making use of doParallel package [4] and of the multithread architecture of the HW used for this project.

Multinomial Logistic Regression

Multinomial logistic regression is a classification method that generalizes logistic regression to multiclass problems. The probability to belong to a particular class given the predictors is formulated as:

$$P(y = k|x^{(i)}, \theta) = \frac{\exp(\theta^{(k)T}x^{(i)})}{\sum_{j=1}^k \exp(\theta^{(j)T}x^{(i)})}$$

where k is the class to be predicted, $x^{(i)}$ the features vector for observation $^{(i)}$ and θ the coefficient vector learned by the learner algorithm for each class.

It is also called softmax regression.

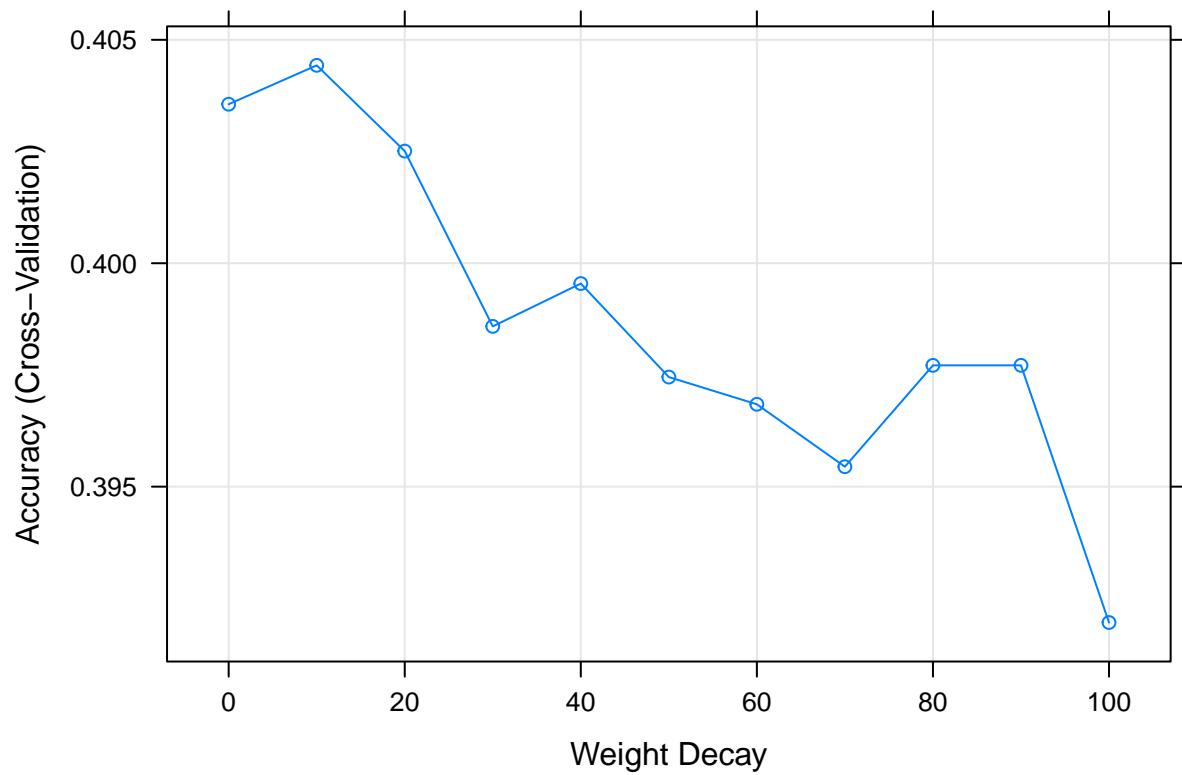
The multinomial model is implemented in R via neural networks by ‘nnet’ package [5]. Under the ‘caret’ framework it is possible to tune the following parameter:

```
##      model parameter      label forReg forClass probModel
## 1 multinom      decay Weight Decay FALSE      TRUE      TRUE
```

The tuning parameter Weight Decay is specific to neural networks and it helps the optimization process avoiding over-fitting. The training process has been performed in two steps, the first went through a wide range of values for the hyperparameter

```
tunegrid <- data.frame(decay = seq(from = 0, to = 100, by = 10))
```

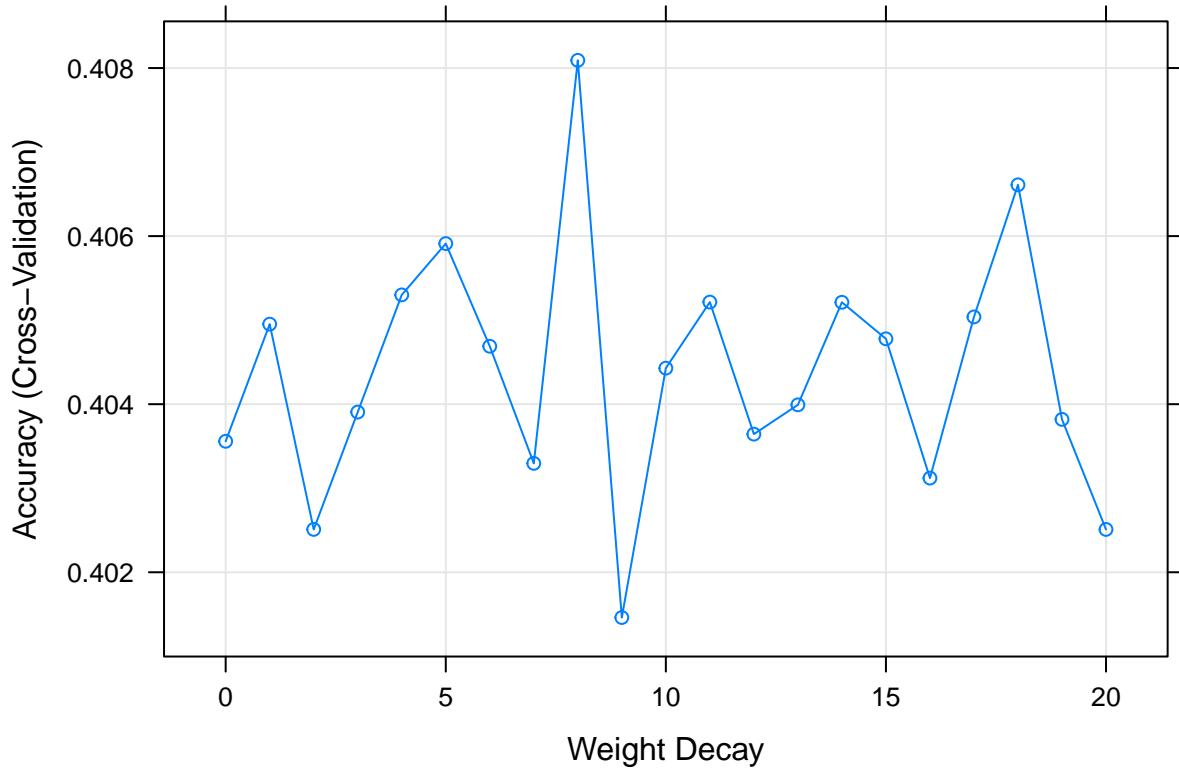
The resulting cross-validation plot identify the range where the best model can be found.



Then the training effort has been restricted to the following range:

```
tunegrid <- data.frame(decay = seq(from = 0, to = 20, by = 1))
```

The resulting cross-validation plot identify the best model.



After cross-validation training through the defined tune grid the best model found has the following tuning parameter:

```
##     decay
## 9      8
```

After evaluating the accuracy of the model prediction to the test unseen data, the confusion matrix and the accuracy overall score is displayed.

```
##           Reference
## Prediction Standing Walking Sitting Falling Cramps Running
##   Standing      1008     45    199    297    337    169
##   Walking        0       0      0       0       0       0
##   Sitting       41     79    221     91     88     59
##   Falling       241     9    214    482    394    166
##   Cramps        85    13    141    194    229    112
##   Running        0       0      0       0       0       0

## [1] "model test accuracy:  0.39479"
```

As per this results, the multinomial model can be discarded. In some sense the bad accuracy results were expected since predictors are correlated and interconnected while multinomial model, as a type of generalized linear model, does not take into account interactions if not expressly defined in the model. Furthermore we can see from the confusion matrix that Walking and Running ADLs are never predicted probably because they have few occurrences in the dataset in respect of other ADLs.

k Nearest Neighbour

k Nearest Neighbors is a non-parametric classification method that make use of distance (or similarity) measures. In particular for numerical predictors the euclidean distance is used. Euclidean distance is the length of the segment connecting 2 data points in the predictor space and it is defined as:

$$d(\vec{x}^{(i)}, \vec{x}^{(j)}) = \sqrt{(x_1^{(i)} - x_1^{(j)})^2 + (x_2^{(i)} - x_2^{(j)})^2 + \dots + (x_p^{(i)} - x_p^{(j)})^2}$$

The kNN algorithm stored all the data and classify new data points in relation of majority of k nearest (as per euclidean distance) points class.

The k Nearest Neighbors model is implemented in R by ‘e1071’ package [6]. Under the ‘caret’ machine learning framework it is possible to tune the following hyperparameter:

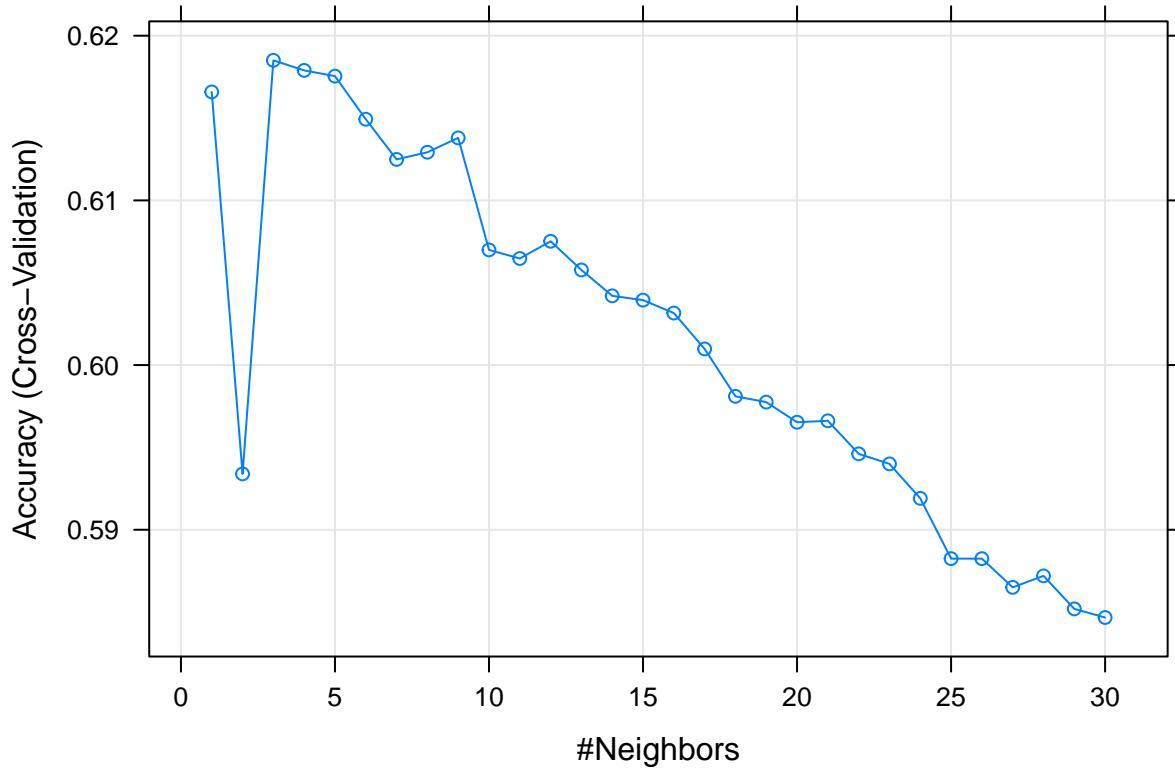
```
##   model parameter      label forReg forClass probModel
## 1   knn            k #Neighbors    TRUE     TRUE     TRUE
```

As tuning parameter k, number of nearest neighbors to consider, increases the decision boundary gets more smooth. k can be thought as a mean of regularizaion. The training process will go through the following value for hyperparameters

```
tunegrid <- data.frame(k= seq(1, 30, 1))
```

Given that kNN algorithm is based on distance/similarity measures, data needs to be scaled (by dividing by respective standard deviation) and centered (by subctracing the mean) before traning in order to avoid that predictors with largest numerical range mask the effect of other predictors.

The resulting cross-validation plot identify the best model.



After cross-validation training through the defined tune grid the best model found has the following tuning parameter:

```
##     k
## 3 3
```

After evaluating the accuracy of the model prediction to the test unseen data, the confusion matrix and the accuracy overall score is displayed.

```
##          Reference
## Prediction Standing Walking Sitting Falling Cramps Running
##   Standing      911      4     12     19     48     48
##   Walking       13    107     28      8      7      2
##   Sitting       52     31    525    162     57     24
##   Falling      136      2    149    663    194     54
##   Cramps       165      0     45    179    615    123
##   Running       98      2     16     33    127    255
## [1] "model test accuracy:  0.62597"
```

kNN model succeeded in classifying all 6 ADLs and it gets a good accuracy score considering that we have 6 class. Even Walking despite of class small numerosity is predicted with good accuracy. The best model is obtained with a quite small k, indicating a complex model so more subject to overfitting.

Random Forest

Random Forest algorithm builds multiple decision trees and merges them together to get a more accurate and stable prediction reducing variance and avoiding overfitting in respect of the single decision tree. Random forest improves the predictive performance of decision tree through bagging, averaging models learned on multiple bootstrapped samples from the original dataset, and randomly selecting the predictors among which identify the one for partitioning data so that purest node are created at each split.

The Random Forest model is implemented in R by ‘randomForest’ package [7]. Under the ‘caret’ framework it is possible to tune the following hyperparameter:

```
##   model parameter           label forReg forClass probModel
## 1    rf      mtry #Randomly Selected Predictors  TRUE     TRUE     TRUE
```

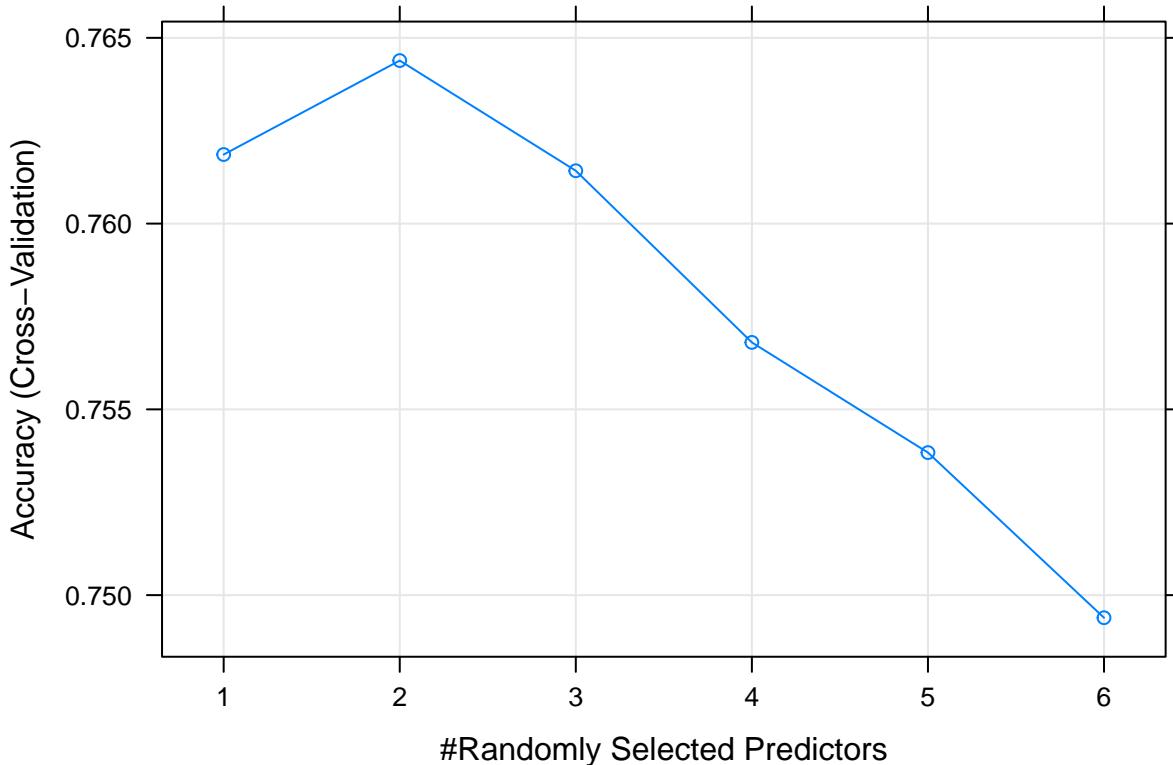
mtry set the number of variables randomly sampled as candidates at each split.

The training process went through the following value for hyperparameters

```
tunegrid <- data.frame(mtry= seq(1, 6, 1))
```

Note that setting mtry equal to the numbers of predictors is only bagging not exactly random forest (trees are not decorrelated because all trees use all the predictors) and it is expected to obtain a worse accuracy in cross validation.

The resulting cross-validation plot identify the best model.



After cross-validation training through the defined tune grid the best model found has the following tuning parameter:

```

##     mtry
## 2      2

After evaluating the accuracy of the model prediction to the test unseen data, the confusion matrix and the accuracy overall score is displayed.

##          Reference
## Prediction Standing Walking Sitting Falling Cramps Running
##   Standing     1330      0       7       8      22      28
##   Walking        0     120      20       3       4       3
##   Sitting        5      26     586     123      43      15
##   Falling       14      0    141     773     178      41
##   Cramps        19      0      19     145     702     125
##   Running        7      0       2      12      99     294

## [1] "model test accuracy:  0.77432"

```

Random Forest model succeeded in classifying all 6 ADLs and it gets a more than good accuracy score considering that we have 6 class. Even Walking despite of class small numerosity is predicted with good accuracy. The best model is obtained with a small mtry.

Results

The following table showed the results achieved for all models.

method	accuracy
multinom	0.3947904
knn	0.6259666
rf	0.7743183

Multinomial model tends to perform badly on the Fall Detection dataset because predictors are heavily correlated and multinomial model is a generalised linear model .

Better accuracy performance can be achieved with kNN a memory based algorithm but with a small number k of neighbors revealing an intrinsic complexity and therefore a potential tendency to overfit.

Random Forest model predicts with a good accuracy all 6 ADLs because trees are able to better understand interactions among predictors.

Interpretation

RandomForest is a black-box model with low interpretability even if considering the variable importance side result. The interpretability issue related to machine learning black-box methods adoption has become relevant and a subject of discussion in the data science community. In this regard it is possible to mention the provoking titled paper “Why should I trust you? ...” [8] and the equally provoking book “Weapons of Math Destruction ...” [9].

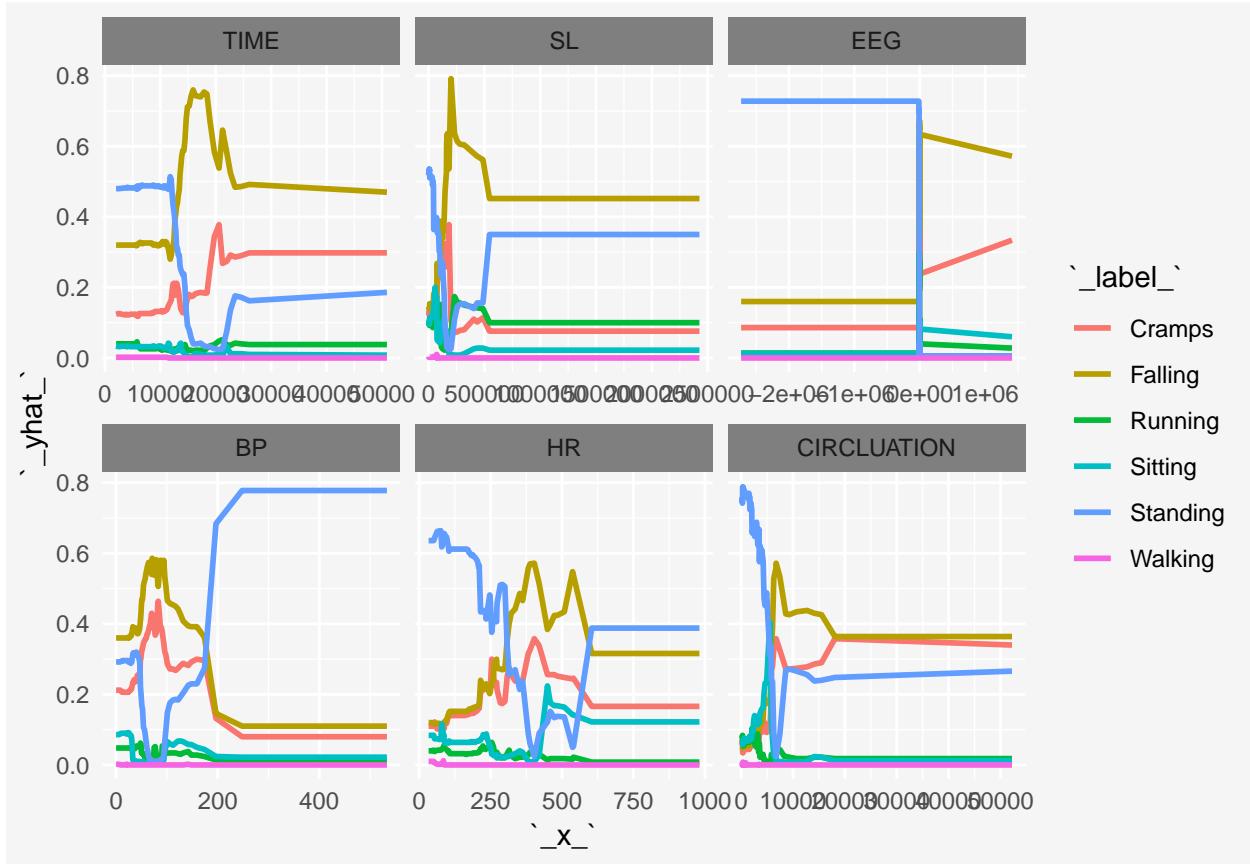
So in order to provide explanation about the model result that could be validated by domain experts, a “ceteris paribus” analysis has been performed. The concept behind this kind of analysis is very simple: one or more single predictions of interest for domain experts is taken under consideration and then varying

only one predictor at a time keeping all other variable unchanged (this is the meaning of latin word ‘ceteris paribus’: all else unchanged) what the model have learned can be assessed.

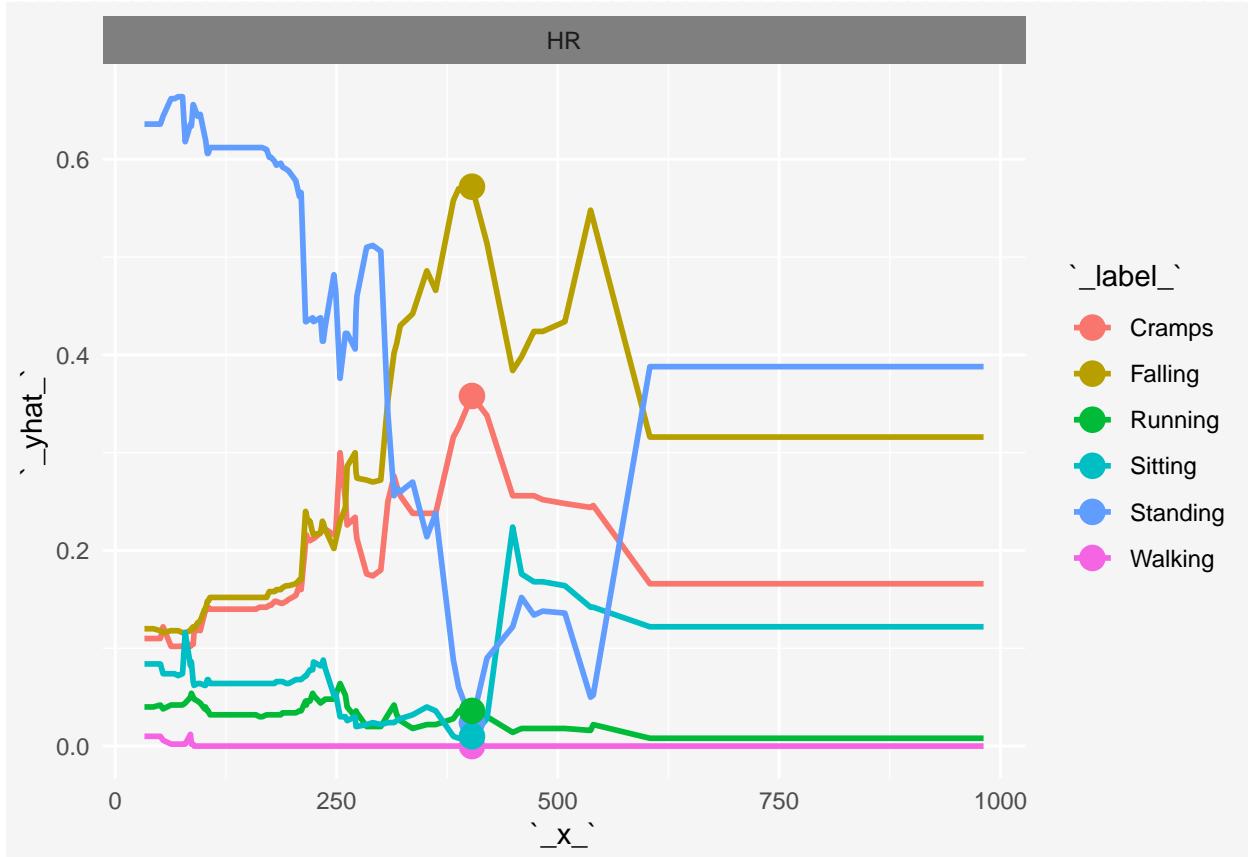
As an example the following prediction has been selected:

ACTIVITY	TIME	SL	EEG	BP	HR	CIRCLUATION
Falling	20368.8	193489	-4900	93	403	6746

The analysis for multiclass classification has been performed following the procedure documented at https://pbiecek.github.io/DALEX_docs/5-5-cetParLocalMulticlass.html#cetParLocalMulticlass for using “cerisParibus” package [10].



In order to understand how the resulting plot should be analysed it is useful to focus on only one variable plot: HR for example.



At the selected prediction the probability of “Falling” is way higher than for other ADLs. There is a range varying only HR in which the prediction remains “Falling” even if predicted probabilities change. Decreasing HR Heart Beat Rate does not change the prediction until below about 300 heart beat where the most probable ADLs become “Standing”. Also increasing HR till about 350 the model mantains the prediction at “Falling”. Without any domain expertise it is impossible to say if physiology theory can support the result learned by our model, but this is why generally Data Science Team includes domain experts together with Data Scientists.

Conclusions

Going back to our research question, it is possible to state that ADLs can be predicted from basic health measures with an accuracy over 77%. This means that the best model guesses the right ADL among 6 more than 3 times over 4. It is a remarkable result.

Validity

Results can be considered valid because of this 3 main reasons:

- a training / validation / test study design has been followed consistently for all models
- test and training/validation set contains thousands of observations
- all used machine learning techniques are consolidated

Furthermore ceteris paribus analysis presented could represent a feasable way to validate the model consistency with domain theory.

Limitations

This project is a data science project in the context of supervised learning focused on the study of prediction.

Therefore the following 2 general limitations apply:

- from the results cannot be inferred anything about causation;
- the results validity depends on the quality of the data collected and contained in the dataset under study. Any sampling or measurement bias could be reflected in the results.

Model improvements

Future research should look at evaluating different machine learning techniques such as implementing a deep neural network with enough hidden layers in order to better learn the interactions between predictors.

Another possibility for increasing the accuracy of the prediction could be stacking: an ensemble method that builds a classification model at an upper level in regards of the studied models using prediction of the lower levels model as predictors for the upper level model.

In the contest of an actual scientific study other explanatory tool could be investigated and used so that domain experts can validate what the model have learned.

Implications for my course of study

This project helped me in improving and consolidating:

- my understanding of the data science research methodology;
- the ability to communicate data science results in a reproducible report;
- and the expert use of statistical computation tools.

Reproduciblty

R script and rmarkdown files are available for review on public github repository:

👉 https://github.com/mdt-ds/PH125.9_cyo_fallDetection .

R script is intended to be reproducible so:

- all package loading is checked for package installation.
- dataset is provided together with the code and loaded into R from my GitHub.
- seed for random number generation has been set to guarantee reproducible results wherever it is needed
- number of cores for parallel computation has not been hardcoded but retrieved with parallel::detectCores() function
- furthermore in order to facilitate reproducibility, HW and SW used for this project have been reported below.

HW

The computation has been performed on my laptop:

```
## [1] "Machine:      AMD Ryzen 5 PRO 2500U w/ Radeon Vega Mobile Gfx"  
## [1] "Num cores:    4"  
## [1] "Num threads: 8"  
## [1] "RAM:          8GB"
```

SW

The software stack is shown below launching sessionInfo() R function.

```
R version 3.5.3 (2019-03-11)  
Platform: x86_64-w64-mingw32/x64 (64-bit)  
Running under: Windows 10 x64 (build 16299)  
  
Matrix products: default  
  
locale:  
[1] LC_COLLATE=Italian_Italy.1252  LC_CTYPE=Italian_Italy.1252  
[3] LC_MONETARY=Italian_Italy.1252 LC_NUMERIC=C  
[5] LC_TIME=Italian_Italy.1252  
  
attached base packages:  
[1] parallel  stats      graphics   grDevices  utils      datasets  methods  
[8] base  
  
other attached packages:  
[1] benchmarkme_1.0.0     ceterisParibus_0.3.1  gower_0.2.0  
[4] DALEX_0.4            doParallel_1.0.14    iterators_1.0.10  
[7] foreach_1.4.4        randomForest_4.6-14  GGally_1.4.0  
[10] gridExtra_2.3       caret_6.0-81       lattice_0.20-38  
[13]forcats_0.4.0       stringr_1.4.0      dplyr_0.8.0.1  
[16] purrr_0.3.2        readr_1.3.1       tidyverse_1.2.1  
[19] tibble_2.1.1        ggplot2_3.1.0      tidyverse_1.2.1  
  
loaded via a namespace (and not attached):  
[1] httr_1.4.0           jsonlite_1.6       splines_3.5.3  
[4] prodlm_2018.04.18   modelr_0.1.4      assertthat_0.2.1  
[7] highr_0.8            stats4_3.5.3      cellranger_1.1.0  
[10] yaml_2.2.0          ipred_0.9-8       pillar_1.3.1  
[13] backports_1.1.3    glue_1.3.1        digest_0.6.18  
[16] RColorBrewer_1.1-2 rvest_0.3.2      colorspace_1.4-1  
[19] recipes_0.1.5     htmltools_0.3.6   Matrix_1.2-15  
[22] plyr_1.8.4          timeDate_3043.102 pkgconfig_2.0.2  
[25] broom_0.5.1         haven_2.1.0       scales_1.0.0  
[28] lava_1.6.5          generics_0.0.2    withr_2.1.2  
[31] nnet_7.3-12         lazyeval_0.2.2    cli_1.1.0
```

```

[34] survival_2.43-3      magrittr_1.5          crayon_1.3.4
[37] readxl_1.3.1        evaluate_0.13       fansi_0.4.0
[40] nlme_3.1-137        MASS_7.3-51.1       xml2_1.2.0
[43] class_7.3-15         tools_3.5.3          data.table_1.12.0
[46] hms_0.4.2           munsell_0.5.0       compiler_3.5.3
[49] e1071_1.7-1          rlang_0.3.2          grid_3.5.3
[52] rstudioapi_0.10     labeling_0.3         rmarkdown_1.12
[55] gtable_0.3.0         ModelMetrics_1.2.2 codetools_0.2-16
[58] reshape_0.8.8         curl_3.3            benchmarkmeData_1.0.1
[61] reshape2_1.4.3        R6_2.4.0             lubridate_1.7.4
[64] knitr_1.22            utf8_1.1.4          stringi_1.4.3
[67] Rcpp_1.0.1            rpart_4.1-13        tidyselect_0.2.5
[70] xfun_0.5

```

Computation time

Sourcing the script containing all the analysis on my laptop configured as above took around 5 minutes to complete.

Acknowledgments

I gratefully aknowledge the efforts of Professor Rafael Irizarry and all HarvardX Course Staff for teaching this learning path towards a deeper understanding of Data Science.

References

- [1] Özdemir, Ahmet Turan, and Billur Barshan. “Detecting Falls with Wearable Sensors Using Machine Learning Techniques.” *Sensors* (Basel, Switzerland) 14.6 (2014): 10691–10708. PMC. Web. 23 Apr. 2017.
- [2] Rafael Irizarry (2018). Introduction to Data Science. Data Analysis and Prediction Algorithms with R. Chapters 30, 31, 32 and 33 <https://rafalab.github.io/dsbook/>
- [3] Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang, Can Candan and Tyler Hunt. (2018). caret: Classification and Regression Training. R package version 6.0-81. <https://CRAN.R-project.org/package=caret>
- [4] Microsoft Corporation and Steve Weston (2018). doParallel: Foreach Parallel Adaptor for the ‘parallel’ Package. R package version 1.0.14. <https://CRAN.R-project.org/package=doParalle>
- [5] Venables, W. N. & Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth Edition. Springer, New York. ISBN 0-387-95457-0
- [6] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel and Friedrich Leisch (2019). e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R package version 1.7-1. <https://CRAN.R-project.org/package=e1071>
- [7] A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. *R News* 2(3), 18–22.
- [8] Ribeiro, M.T., Singh, S., & Guestrin, C. (2016). “Why Should I Trust You?”: Explaining the Predictions of Any Classifier. HLT-NAACL Demos.
- [9] O’NEIL, C. (2016). Weapons of math destruction: how big data increases inequality and threatens democracy.

[10] Przemyslaw Biecek (2019). ceterisParibus: Ceteris Paribus Profiles. R package version 0.3.1. <https://CRAN.R-project.org/package=ceterisParibus>

