Human Activity Recognition (HAR)

HarvardX

Jose Ramon Hernandez Galan

18/11/2019

Introduction

This is a report showing the process and results for the creation of a model for human activity recognition (herafter HAR).

This is the capstone project of the Data Science course pursued by the author in HarvardX.

The main **goal** of this project is to use machine learning techniques in order to predict the human activity based on the data extracted from on-body sensors. We will compare our results to those obtained by the main **contributor** [1].

We will also try to observe if all 4 sensors are really necessary and if we can use less sensors in order to predict the activity.

Dataset

The dataset used in this project is the **HAR Dataset for benchmarking** [1]

The dataset includes measurements of **inertial sensors** attached to several people while doing normal activities during the day. It also includes data related to the person such as weight, height, etc.

A more detailed description of the dataset can be found **here**.

Analysis

In this section we will prepare the data to work with and explore some important characteristics of the dataset.

Data wrangling

The created har dataset has the following structure:

Name	Type	Description
user	Factor	w/ 4 levels “debora”,“jose_carlos”
gender	Factor	w/ 2 levels “Man”,“Woman”:
age	int	user age
how_tall_in_meters	num	height
weight	int	weight in kg
body_mass_index	num	body mass
x1	int	sensor 1 axis x in m/s2
y1	int	sensor 1 axis y in m/s2
z1	int	sensor 1 axis z in m/s2
x2	int	sensor 2 axis x in m/s2
y2	int	sensor 2 axis x in m/s2

Name	Type	Description
z2	int	sensor 2 axis x in m/s2
x3	int	sensor 3 axis x in m/s2
y3	int	sensor 3 axis x in m/s2
z3	int	sensor 3 axis x in m/s2
x4	int	sensor 4 axis x in m/s2
y4	int	sensor 4 axis x in m/s2
z4	int	sensor 4 axis x in m/s2
class	Factor	Activity type. w/ 5 levels

See the different activites listed in the dataset:

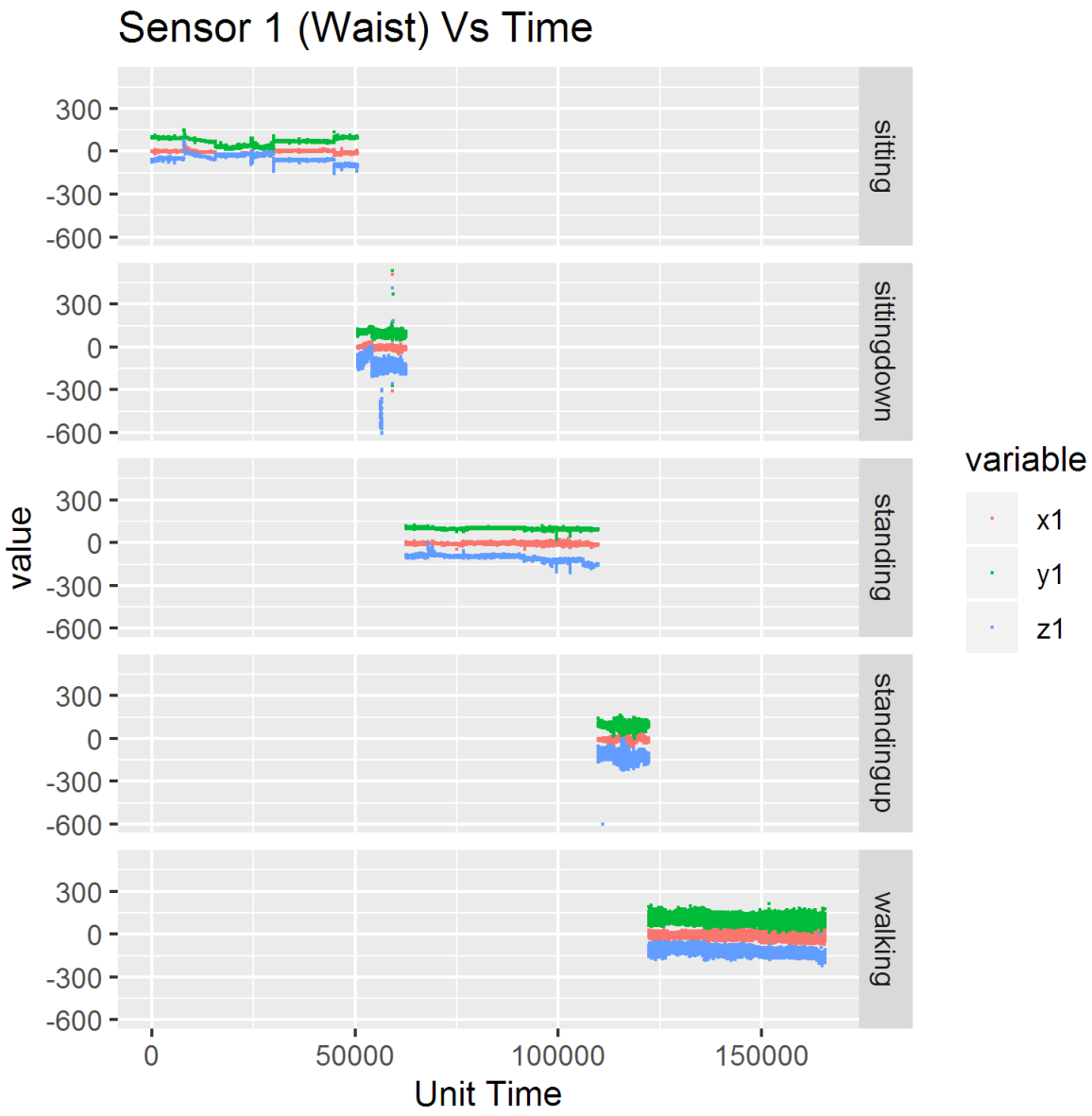
```
levels(har$class)
```

```
## [1] "sitting"     "sittingdown" "standing"    "standingup"  "walking"
```

These classes are associated to the values obtained by the sensors (1 to 4) expressed in m/s2. X, Y, Z are the values obtained for each axis.

Exploratory Data Analysis

In order to understand how the values were taken we will view the sampled data **across the time for sensor 1 values (x,y,z)**. The data was taken during 8 hours. During this time the sensed subject was doing different activites.



This is the sensor located at the waist of the subjects.

Note the **unit time** is not specified. We have included a non-scaled time unit for creating the chart.

Some additional **checkings** have been done in order to verify the consistency of data.

```
# Is there any NAs ?
nas <- apply(har, MARGIN = 2, function(x) any(is.na(x) | is.infinite(x)))
if (any(nas) == FALSE)
{
  cat("Great, no NAs or Infinite value in the dataset")
}else{
  cat("Attention, some NA or Infinite value was found in the dataset")
}
```

```
## Great, no NAs or Infinite value in the dataset
```

```r
# proportion of classes
har %>% group_by(class) %>% summarize(n = n(), p = n()/nrow(.)) %>% arrange(desc(p))
```

```
## # A tibble: 5 x 3
##   class           n        p
##   <fct>       <int>    <dbl>
## 1 sitting     50631 0.30568
## 2 standing    47370 0.28599
## 3 walking     43390 0.26196
## 4 standingup  12415 0.074955
## 5 sittingdown 11827 0.071405
```

```r
# check proportion of classes and users
har %>% group_by(class, user) %>% summarize(n = n(), p = n()/nrow(.)) %>% arrange(desc(p))
```

```
## # A tibble: 20 x 4
## # Groups:   class [5]
##    class       user            n         p
##    <fct>       <fct>       <int>     <dbl>
##  1 sitting     debora      15615 0.094275
##  2 sitting     wallace     14993 0.090519
##  3 standing    debora      14940 0.090199
##  4 standing    wallace     14467 0.087344
##  5 sitting     katia       14280 0.086215
##  6 standing    katia       14234 0.085937
##  7 walking     wallace     14037 0.084748
##  8 walking     debora      13622 0.082242
##  9 walking     katia       13556 0.081844
## 10 sitting     jose_carlos  5743 0.034673
## 11 standingup  wallace      4115 0.024844
## 12 sittingdown katia        4017 0.024252
## 13 standingup  debora       3853 0.023262
## 14 standing    jose_carlos  3729 0.022514
## 15 standingup  katia        3710 0.022399
## 16 sittingdown debora       3547 0.021415
## 17 sittingdown wallace      3486 0.021047
## 18 walking     jose_carlos  2175 0.013131
## 19 sittingdown jose_carlos   777 0.0046911
## 20 standingup  jose_carlos   737 0.0044496
```

Now, we are sure there are **not NA values** in our dataset. Also we are sure **all users (4)** have been measured in **all the different activities (5)**. Note the prevalance in some user/activity. This also can be observed in the time-based chart above.

Data set partitioning

We have splitted the har original dataset in several sets for training, tuning and validatiaon.

Dataset	Observations	Proportions	Description
har	165.633	1	Original
har_val	16.565	1/10 of har	Set for final validation

Dataset	Observations	Proportions	Description
har_set	149.068	9/10 of har	Set for model analysis
har_set_train	134.158	9/10 of har_set	Set for model training
har_set_test	16.565	1/10 of har_set	Set for model testing while optiomizing

Analysis approach

Since the outcome of the dataset is **multi-categorical** variable (class: sitting, sittingdown, standing, standingup, walking) with 5 possible values we have been focused on **classifcation algorithms** that support such variables. The algorithm chosen is the **decission trees** and its derivative random forest.

We will only focus on **features** provided by the **sensors** (x1..z4).

How results are measured

The results are measured by the **accuracy** when classifying the sensor observations.

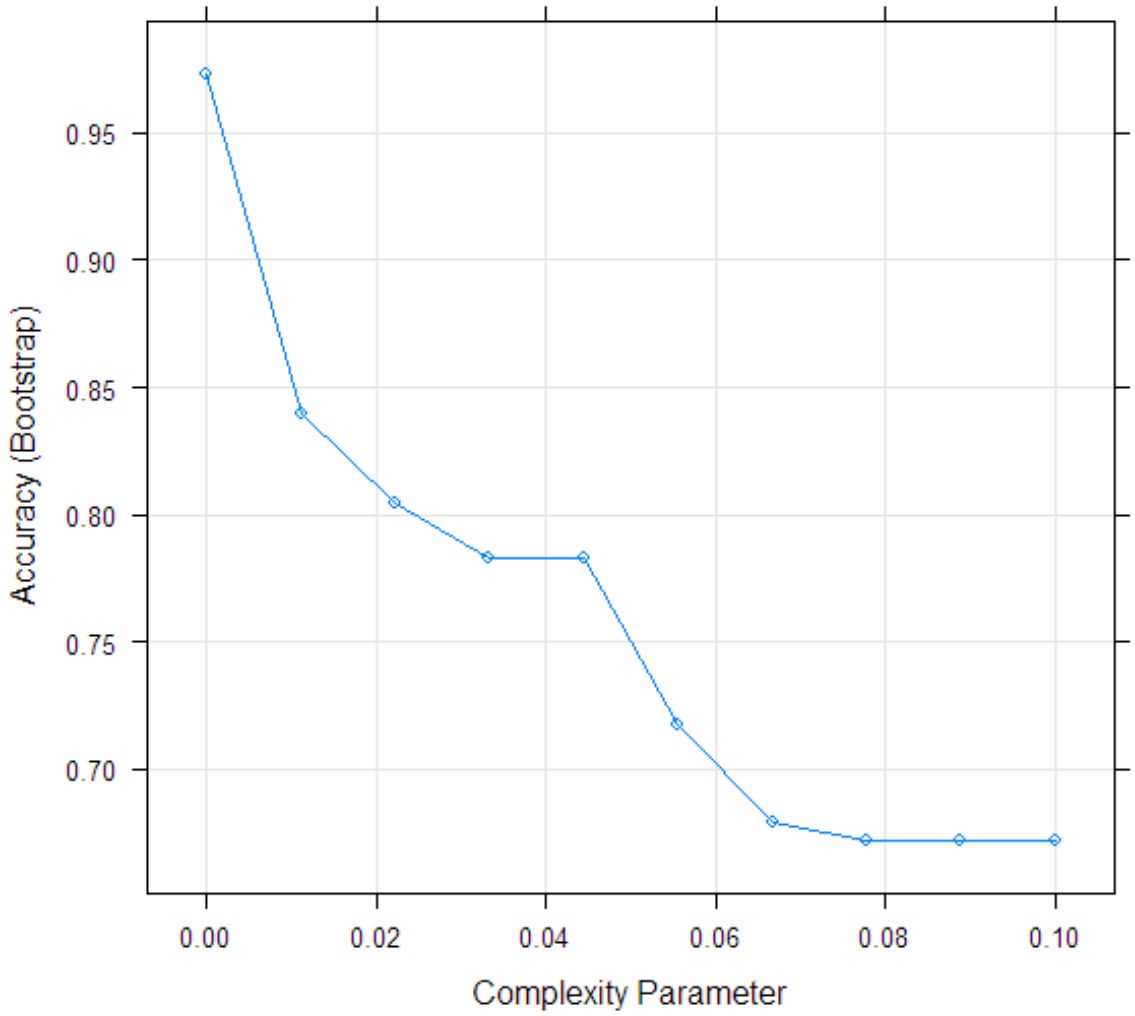
As final resut we will provide the confusion matrix so it can be compared to hte results in [1].

Model based on Classification Trees

Decision Trees

By using the functions found in the caret package we have created the model for the decision tree.

```
fit_part <- train(class ~ x1 + y1 + z1 + x2 + y2 + z2 + x3 + y3 + z3 + x4 + y4 + z4 ,
                  method = "rpart",
                  tuneGrid = data.frame(cp = seq(0.0, 0.1, len = 10)),
                  data = har_set_train)
plot(fit_part)
```

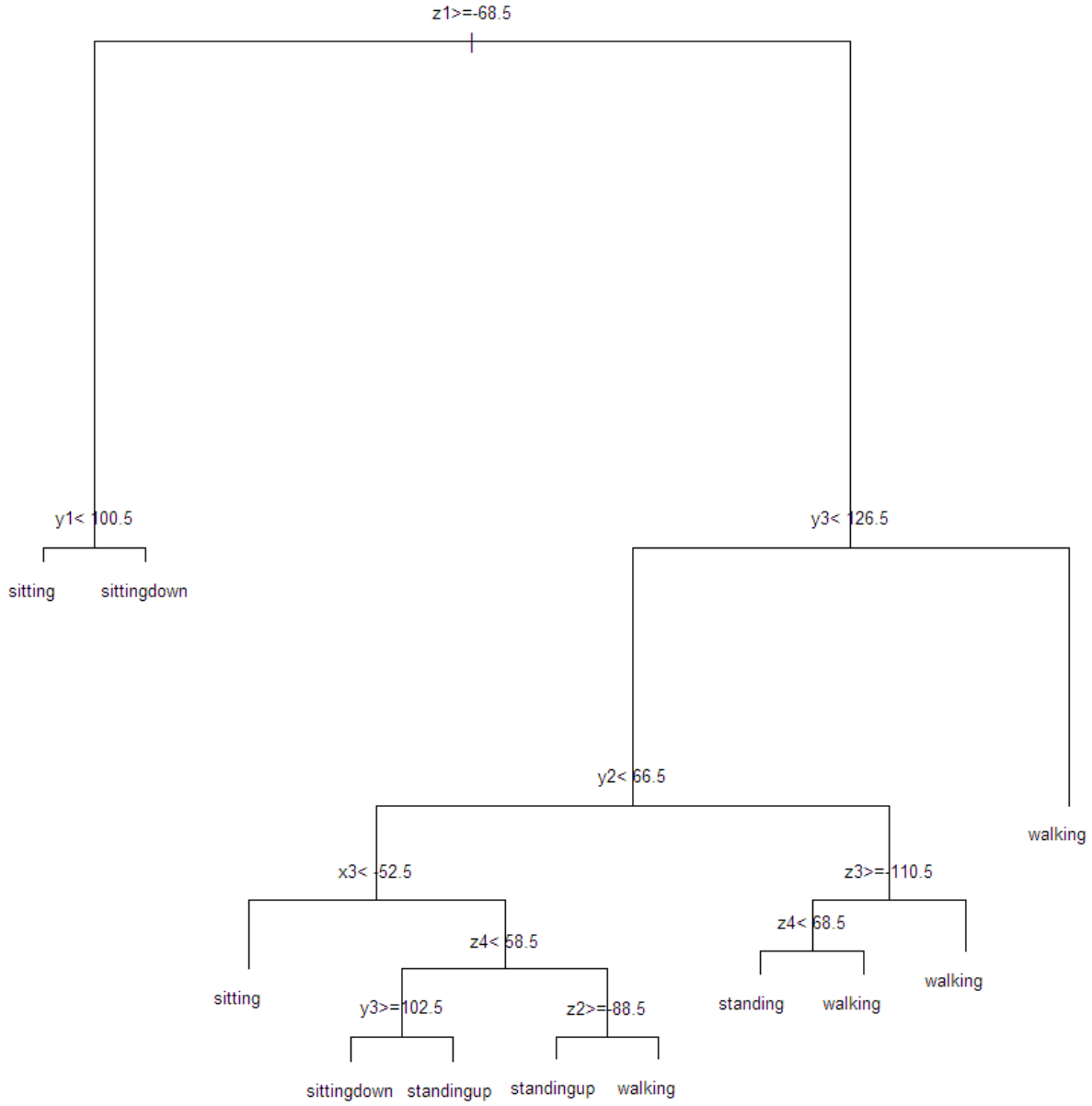


Since the created tree (the one for max accuracy) is pretty complex and dense (with 690 splits) we will ommit it. Below we will show a more simple tree as example.

When predicting over the har_set_test we obtain following accuracy:

```
## # A tibble: 1 x 4
##   METHOD                     TUNING           SENSORS ACCURACY
##   <chr>                      <chr>            <chr>      <dbl>
## 1 Classification tree(rpart) CP=0(690 splits) 1,2,3,4  0.97632
```

In order to view a real example of a tree we will create a model with few branches by **pruning** the original tree.



Note accuracy is lower with a higher Complexity Parameter (CP) as observed in the chart above. A higher CP means that lesser branches will be used to compose the tree.

```
## # A tibble: 3 x 4
##   METHOD                              TUNING             SENSORS ACCURACY
##   <chr>                               <chr>              <chr>      <dbl>
## 1 Classification tree(rpart)          CP=0(690 splits)   1,2,3,4  0.97632
## 2 Classification tree(rpart) pruned   CP=0.01(10 splits) 1,2,3,4  0.84997
## 3 Classification tree(Random Forest) Trees=50,mtry=3     1,2,3,4  0.99497
```

Random Forest

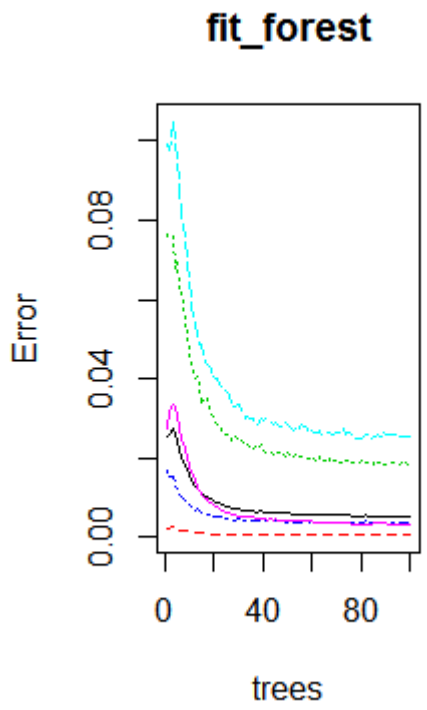
A more optimized algorithm based on classification trees is the random forest. This algorithm creates several trees with randomly selected features and the outcome is calculated by voting for the most probable outcome

across all trees.

```
set.seed(14)
mtry <- seq(1,12,1) # number of variables randomly selected for each tree

fit_forest <- randomForest(class ~ x1 + y1 + z1 + x2 + y2 + z2 + x3 + y3 + z3 + x4 + y4 + z4,
                           data = har_set_train,
                           ntree = 100,
                           do.trace = 10,
                           tuneGrid = data.frame(mtry = mtry))

plot(fit_forest)
```



We choose 50 trees as number of trees to grow and 3 as the number of features to be randomly selected for each tree (parameter called mtry).

```
fit_forest1$mtry
```

```
## [1] 3
```

Some train control is needed so as to speed up the run time of the model training. As we can see the accuracy **improves in 2%** with respect to a single tree algorithm.

```
## # A tibble: 3 x 4
##   METHOD                              TUNING             SENSORS ACCURACY
##   <chr>                               <chr>              <chr>      <dbl>
## 1 Classification tree(rpart)          CP=0(690 splits)   1,2,3,4  0.97632
## 2 Classification tree(rpart) pruned   CP=0.01(10 splits) 1,2,3,4  0.84997
## 3 Classification tree(Random Forest) Trees=50,mtry=3     1,2,3,4  0.99497
```

Random Forest (Sensor importance)

Let´s analyze how important is each feature in the model.

```
varImp(fit_forest)
```

```
## rf variable importance
##
##    Overall
## z1  100.00
## z2   89.65
## y2   84.98
## y3   72.84
## x4   47.95
## y1   44.35
## z4   43.09
## z3   29.86
## x2   28.78
## x3   13.41
## y4    6.83
## x1    0.00
```

According to the table above the importance of the sensors is ordered as follwos (from most to least): **S1 (waist)**, S3 (right angkle), S2(left thigh), S4(rigth upper arm).

Let´s observe how the model would perform in case of selecting the most important sensors.

```
## # A tibble: 3 x 4
##   METHOD                              TUNING          SENSORS ACCURACY
##   <chr>                               <chr>           <chr>      <dbl>
## 1 Classification tree(Random Forest) Trees=50,mtry=3 1,2,3,4  0.99497
## 2 Classification tree(Random Forest) Trees=50,mtry=3 1,2      0.96003
## 3 Classification tree(Random Forest) Trees=50,mtry=3 1,2,3    0.99081
```

Result with **2 sensors is too poor**, but if we use 3 sensors the results get close to values obatined with all 4 sensors.

Final Results

Find below the **confusion matrix** obtained from the prediction on the **har_val** set using the randm forest algorithm overa all 4 sensor values. Note this set is composed by 16.565 observations.

```
##              Reference
## Prediction    sitting sittingdown standing standingup walking
##   sitting        5059           1        0          0       0
##   sittingdown       2        1165        0          9       4
##   standing          0           2     4720         10       6
##   standingup        3           6        3       1205       4
##   walking           0           9       14         18    4325
```

We can observe how the accuracy is lower on those classes with lesser observations.

The **accuracy** results obatined on the validation set (har_val) across the different algorithms are shown below:

METHOD	TUNING	ACCURACY
1 Classification tree (rpart)	CP = 0 (690 splits)	0.97609
2 Classification tree (rpart)	CP = 0.01 (10 splits)	0.84618
3 Classification tree (Random Forest)	Trees = 50, mtry = 3	**0.99451**

Note all 4 sensors are used for this predictions.

Conclusions

- Modeling approach. As the outcome is a multi-categorical variable we have decided to use **classification algorithms**, which have provided pretty **high accurate results**.
- **Results** compared to the author. The accuracy reported by the main contributor [1] is 0.99414, which was predicted on the full har dataset. Our prediction, which was created on a subset of the har dataset (without overtraining), provided an accuracy of **0.99451**.
- Regarding to the **importance of the sensors**. We have observed how the sensor 1 (placed on waist) is the most important, the sensor 4 (placed on right upper arm) is the one that provides less important information. By omitting the use of this sensor we could get an accuracy close to 0.991.
- **Future work**. The features related to the user are not included in the described models. The **user effect** is something to be analysed. The testbench was used on the data extracted from the same subjects. But what would happen if we test our model on values from different users ? what about different **sensors**?

References

1. Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6_6

Read more: http://groupware.les.inf.puc-rio.br/har#sbia_paper_section#ixzz65cgnrXLU