Assignment Machine Learning

ThierryFauret

26 avril 2017

1. Executive Summary

This analyze treats data for Human Activity Recognition (HAR). We have a dataset with 5 classes (sitting-down, standing-up, standing, walking, and sitting). Our goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and build a model predicting the class from this recorded data.

We have tested 4 methodologies (lda, rpart, unsupervised model, random forest). The best fiiting has been obtained with the random forest model. With this model, the prediction on the 20 data of the testing data are : [1] B A B A A E D B A A B C B A E E A B B B

2. Context

This analyze treats data for Human Activity Recognition (HAR). We have a dataset with 5 classes (sitting-down, standing-up, standing, walking, and sitting). Our goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and build a model predicting the class from this recorded data.

For more information, see the publication:

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. Cited by 2 (Google Scholar)

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz4fNsaOa5R (http://groupware.les.inf.puc-rio.br /har#ixzz4fNsaOa5R)

3. Getting and cleaning data / Exploratory data analysis

Two files are available: pml-training.csv : training data pml-testing.csv ; testing data on which will be applied a model in order to give the predictions

```
library(caret)
library(rattle)
library(ggplot2)
library(GGally)
library(dplyr)
library(RANN)

training<-read.csv("pml-training.csv",sep=",")
testing<-read.csv("pml-testing.csv",sep=",")

nb<-dim(training)
```

By an exploratory analysis of the data base, we can notice that several parameters contains a high percentage of NA. We propose two methods to treat this : - use an algorithm to impute data - remove this parameters

Firstly we will fill the missing data. knnImpute will be used to fill the missing data for training data set. For testing data set, the median values calculated with the training data have been put to replace the missing data:

```
for (i in 8:nb[2]-1) {
  training[,i]<-as.numeric(training[,i])
}




for (i in 8:nb[2]-1) {
  testing[,i]<-as.numeric(testing[,i])
}

preProcValues <- preProcess(training[,8:nb[2]], method = c("knnImpute"))
training_imp <- predict(preProcValues, training[,8:nb[2]])


for (i in 8:nb[2]-1){
  if (is.na(testing[1,i])){
    testing[1:20,i]<-rep(median(training_imp[,i-7]),20)
  }else{
    testing[,i]<-scale(testing[,i])
  }
}
```

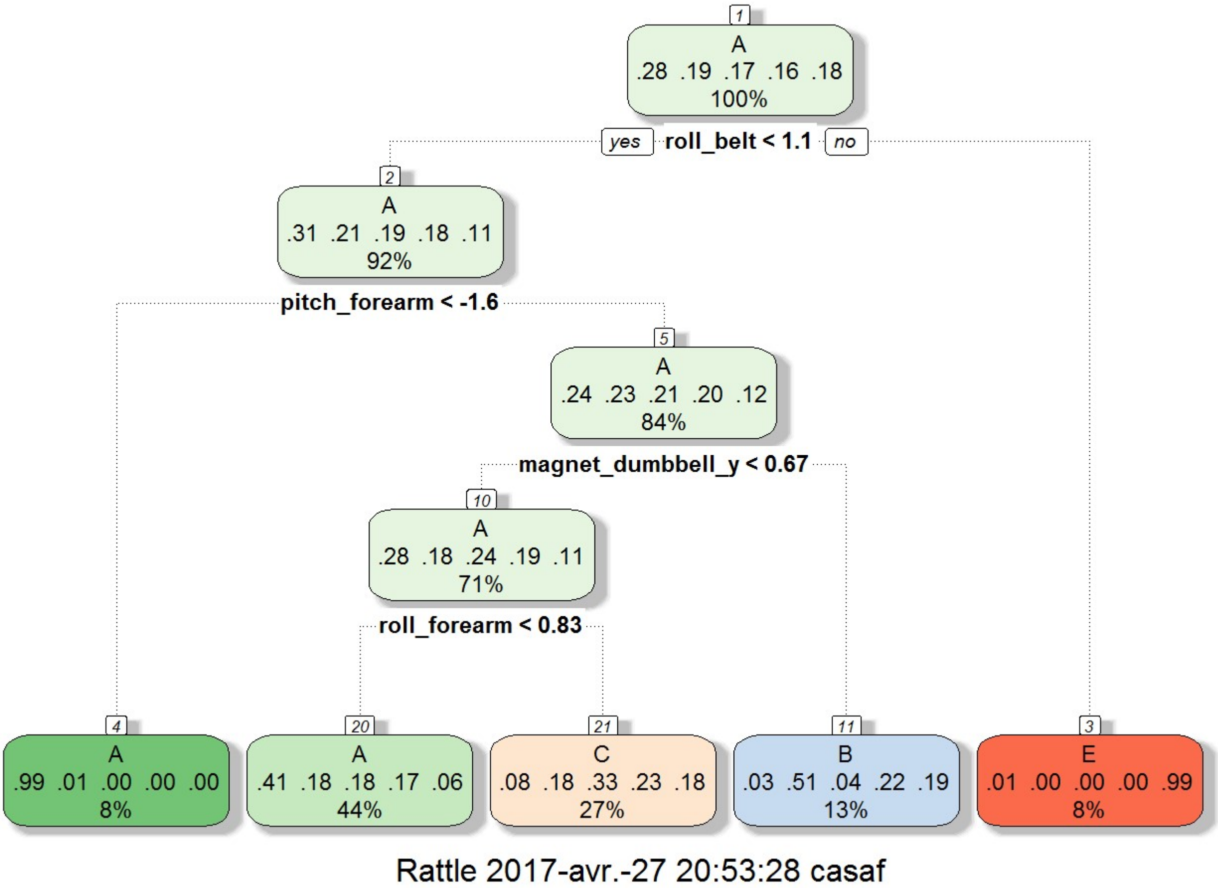
4. Models fitting

rpart model

```
mod.rpart<-train(classe~.,data=training_imp,method="rpart")
print(confusionMatrix(training_imp$classe,predict(mod.rpart,training_imp)))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5080   81  405    0   14
##          B 1581 1286  930    0    0
##          C 1587  108 1727    0    0
##          D 1449  568 1199    0    0
##          E  524  486  966    0 1631
##
## Overall Statistics
##
##                Accuracy : 0.4956
##                  95% CI : (0.4885, 0.5026)
##     No Information Rate : 0.5209
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3407
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4970  0.50850  0.33040       NA  0.99149
## Specificity            0.9468  0.85310  0.88225   0.8361  0.89008
## Pos Pred Value         0.9104  0.33869  0.50468       NA  0.45218
## Neg Pred Value         0.6339  0.92145  0.78395       NA  0.99913
## Prevalence             0.5209  0.12889  0.26638   0.0000  0.08383
## Detection Rate         0.2589  0.06554  0.08801   0.0000  0.08312
## Detection Prevalence   0.2844  0.19351  0.17440   0.1639  0.18382
## Balanced Accuracy      0.7219  0.68080  0.60633       NA  0.94079
```

```
fancyRpartPlot(mod.rpart$finalModel)
```



This model is not satisfactory because it does not predict any classe D (there is 3216 cases of the classe D in the training data set).

Linear discriminant model

Secondly the LDA model is tested:

```
mod.lda<-train(classe~.,data=training_imp,method="lda",preProcess="pca")
print(confusionMatrix(training_imp$classe,predict(mod.lda,training_imp)))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3641  427  626  752  134
##          B  837 1613  647  417  283
##          C  887  336 1778  295  126
##          D  251  478  492 1632  363
##          E  338  724  492  350 1703
##
## Overall Statistics
##
##                Accuracy : 0.5283
##                  95% CI : (0.5213, 0.5353)
##     No Information Rate : 0.3034
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4025
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6115   0.4508  0.44064  0.47359  0.65274
## Specificity            0.8581   0.8639  0.89453  0.90208  0.88809
## Pos Pred Value         0.6525   0.4248  0.51958  0.50746  0.47214
## Neg Pred Value         0.8353   0.8758  0.86068  0.88943  0.94343
## Prevalence             0.3034   0.1823  0.20564  0.17562  0.13296
## Detection Rate         0.1856   0.0822  0.09061  0.08317  0.08679
## Detection Prevalence   0.2844   0.1935  0.17440  0.16390  0.18382
## Balanced Accuracy      0.7348   0.6573  0.66759  0.68783  0.77041
```

The accuracy is equals to 52,83%. It is better than rpart model but it is not very good.

Random forest has been tested but I meet memory problem with this data. The option choosen had been to remove from the data bae the parameters with an high percentage of NA value. It is treated after.

New Data treatment

```
training_col<-matrix(nrow=19622)
training_col<-as.data.frame(training_col)


k<-0
for (i in 1:152){
  x<-training[,7+i]
  if(sum(is.na(x))/length(x)<0.75){
    k<-k+1
    training_col[,k]<-training[,7+i]
    colnames(training_col)[k]<-colnames(training)[7+i]
  }
}

quantile(abs(cor(training_col)),probs=c(0.85,0.90,0.95))
```

```
##       85%       90%       95% 
## 0.7194513 0.7859464 0.8824169
```

```
# the results confirms that some parameters are correlated

training_col[,k+1]<-training[,160]
colnames(training_col)[k+1]<-colnames(training)[160]
```

Unsupervised model

We try to fit an unsupervised model

```
titi<-kmeans(subset(training_col,select=-classe),centers=5)
training_col$clusters<-as.factor(titi$cluster)
table(training_col$classe,training_col$clusters)
```

```
##    
##        1    2    3    4    5
##   A  640  731  587 3045  577
##   B  487 1223  654  726  707
##   C  490 1053  256 1312  311
##   D  469 1071  704  642  330
##   E  444 1233  728  601  601
```

The results are not good. The five cluster are significantly different from the classes. Consequently we do not analyse more the possibility.

Random Forest model

Now we calculate a random forest model.

```
set.seed(62433)
training_col<-subset(training_col,select=-clusters)
mod.rf<-train(classe~.,data=training_col,method="rf", allowParallel = TRUE)
print(confusionMatrix(training_col$classe,predict(mod.rf,training_col)))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5580    0    0    0    0
##          B    0 3797    0    0    0
##          C    0    0 3422    0    0
##          D    0    0    0 3216    0
##          E    0    0    0    0 3607
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2844   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

On the training data, the data are very well fitted.

The significant parameters are :

```
varImp(mod.rf, useModel=TRUE)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 85)
##
##                      Overall
## roll_belt             100.00
## pitch_forearm          59.18
## yaw_belt               55.49
## pitch_belt             44.47
## magnet_dumbbell_z      43.09
## magnet_dumbbell_y      43.01
## roll_forearm           41.72
## accel_dumbbell_y       23.21
## magnet_dumbbell_x      18.39
## roll_dumbbell          18.28
## accel_forearm_x        18.05
## magnet_belt_z          16.05
## accel_dumbbell_z       15.07
## magnet_forearm_z       14.93
## total_accel_dumbbell   14.60
## accel_belt_z           13.77
## magnet_belt_y          12.95
## gyros_belt_z           12.02
## yaw_arm                11.95
## magnet_belt_x          10.92
```

Now we will calculate the prediction based on the testing data base. For that, we have to treat the testing data base (fill the missing data and remowe unuseful parameters).

```
testing<-read.csv("pml-testing.csv",sep=",")

nb<-dim(training)

for (i in 8:nb[2]-1) {
  testing[,i]<-as.numeric(testing[,i])
}

testing_col<-matrix(nrow=20)
testing_col<-as.data.frame(testing_col)

k<-0
for (i in 1:152){
  x<-training[,7+i]
  if(sum(is.na(x))/length(x)<0.75){
    k<-k+1
    testing_col[,k]<-testing[,7+i]
    colnames(testing_col)[k]<-colnames(testing)[7+i]
  }
}



for (i in 1:85){
  if (is.na(testing_col[1,i])){
    testing_col[1:20,i]<-rep(median(training_col[,i]),20)
  }
}

print(predict(mod.rf,newdata=testing_col))
```

```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

5. Conclusion

We have tested 4 methodologies (lda, rpart, unsupervised model, random forest). The best fiiting has been obtained with the random forest model. With this model, the prediction on the 20 data of the testing data are : predict(mod.rf,newdata=testing_col) [1] B A B A A E D B A A B C B A E E A B B B