eindproject

Cynthia de Nijs

20 mei 2017

library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(corrplot)  
library(rpart)  
library(rpart)  
library(randomForest)

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(parallel)  
library(doParallel)

## Loading required package: foreach

## Loading required package: iterators

library(ggplot2)  
library(lattice)

### project course 8

class-variabele voorspellen schrijf een rapport over hoe je het model hebt gebouwd hoe je cross validation hebt gebruikt wat je denkt dat de out of sample error is waarom heb je de keuzes gemaakt die je hebt gemaakt.

Data

The training data for this project are available here:

<https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv>

The test data are available here:

<https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv>

The data for this project come from this source: <http://groupware.les.inf.puc-rio.br/har>. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

What you should submit

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Peer Review Portion

Your submission for the Peer Review portion should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis. Please constrain the text of the writeup to < 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders :-).

Course Project Prediction Quiz Portion

Apply your machine learning algorithm to the 20 test cases available in the test data above and submit your predictions in appropriate format to the Course Project Prediction Quiz for automated grading.

Reproducibility

Due to security concerns with the exchange of R code, your code will not be run during the evaluation by your classmates. Please be sure that if they download the repo, they will be able to view the compiled HTML version of your analysis.

# read the data and replace "NA", "#DIV/0!", "" by NA (160 columns)

training <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"), na.strings=c("NA","#DIV/0!", ""))  
testing <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"), na.strings=c("NA","#DIV/0!", ""))

# cleaning data

remove NA-columns (60 columns left) remove columns that will not explain the outcome variable: X, user\_name, raw\_timestamp\_part\_1, raw\_timestamp\_part\_2, cvdt\_timestamp, new\_window, num\_window (53 columns left)

training <- training[,colSums(is.na(training))==0]  
testing <- testing[,colSums(is.na(testing))==0]  
  
training <- training[,-c(1:7)]  
testing <- testing[,-c(1:7)]

# split the data

I split the trainingsset in a testset and a trainingset (train). The dataset isn't big and isn't small so the trainingsset contains 60% of the data and the testset 40%.

set.seed(123)  
inTrain <- createDataPartition(y=training$classe, p = 0.60, list=FALSE)  
train <- training[inTrain,]  
test <- training[-inTrain,]  
dim(train)

## [1] 11776 53

dim(test)

## [1] 7846 53

# explore the data

balance in outcomes? yes!

table(train$classe)

##   
## A B C D E   
## 3348 2279 2054 1930 2165

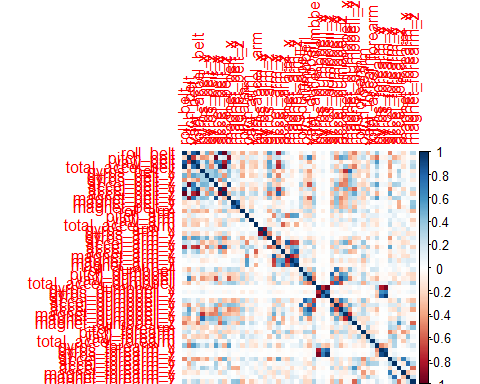
near zero variable; there are no near zero var. left

nzv <- nearZeroVar(train, saveMetrics=TRUE)  
nzv

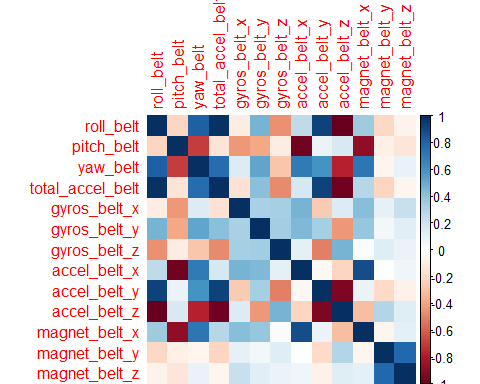
## freqRatio percentUnique zeroVar nzv  
## roll\_belt 1.099812 8.49184783 FALSE FALSE  
## pitch\_belt 1.196078 13.88417120 FALSE FALSE  
## yaw\_belt 1.049536 14.60597826 FALSE FALSE  
## total\_accel\_belt 1.062878 0.22927989 FALSE FALSE  
## gyros\_belt\_x 1.071871 1.08695652 FALSE FALSE  
## gyros\_belt\_y 1.140443 0.55197011 FALSE FALSE  
## gyros\_belt\_z 1.043315 1.35869565 FALSE FALSE  
## accel\_belt\_x 1.098684 1.35020380 FALSE FALSE  
## accel\_belt\_y 1.105946 1.12941576 FALSE FALSE  
## accel\_belt\_z 1.090038 2.43716033 FALSE FALSE  
## magnet\_belt\_x 1.076555 2.54755435 FALSE FALSE  
## magnet\_belt\_y 1.069588 2.39470109 FALSE FALSE  
## magnet\_belt\_z 1.003413 3.63451087 FALSE FALSE  
## roll\_arm 49.926829 19.45482337 FALSE FALSE  
## pitch\_arm 75.814815 22.60529891 FALSE FALSE  
## yaw\_arm 29.242857 21.45040761 FALSE FALSE  
## total\_accel\_arm 1.093458 0.55197011 FALSE FALSE  
## gyros\_arm\_x 1.119205 5.23097826 FALSE FALSE  
## gyros\_arm\_y 1.512987 3.05706522 FALSE FALSE  
## gyros\_arm\_z 1.162500 1.98709239 FALSE FALSE  
## accel\_arm\_x 1.009524 6.45380435 FALSE FALSE  
## accel\_arm\_y 1.171642 4.40726902 FALSE FALSE  
## accel\_arm\_z 1.052632 6.44531250 FALSE FALSE  
## magnet\_arm\_x 1.019231 11.11582880 FALSE FALSE  
## magnet\_arm\_y 1.055556 7.14164402 FALSE FALSE  
## magnet\_arm\_z 1.125000 10.54687500 FALSE FALSE  
## roll\_dumbbell 1.219178 87.21127717 FALSE FALSE  
## pitch\_dumbbell 2.134831 85.02887228 FALSE FALSE  
## yaw\_dumbbell 1.219178 86.65930707 FALSE FALSE  
## total\_accel\_dumbbell 1.102689 0.36514946 FALSE FALSE  
## gyros\_dumbbell\_x 1.013661 1.93614130 FALSE FALSE  
## gyros\_dumbbell\_y 1.249300 2.25033967 FALSE FALSE  
## gyros\_dumbbell\_z 1.002740 1.64741848 FALSE FALSE  
## accel\_dumbbell\_x 1.036458 3.33729620 FALSE FALSE  
## accel\_dumbbell\_y 1.006757 3.81283967 FALSE FALSE  
## accel\_dumbbell\_z 1.036810 3.35427989 FALSE FALSE  
## magnet\_dumbbell\_x 1.097087 8.71263587 FALSE FALSE  
## magnet\_dumbbell\_y 1.278846 6.83593750 FALSE FALSE  
## magnet\_dumbbell\_z 1.052632 5.53668478 FALSE FALSE  
## roll\_forearm 11.085308 14.70788043 FALSE FALSE  
## pitch\_forearm 66.828571 21.05129076 FALSE FALSE  
## yaw\_forearm 15.904762 14.22384511 FALSE FALSE  
## total\_accel\_forearm 1.135099 0.56046196 FALSE FALSE  
## gyros\_forearm\_x 1.060317 2.33525815 FALSE FALSE  
## gyros\_forearm\_y 1.053333 6.05468750 FALSE FALSE  
## gyros\_forearm\_z 1.090909 2.37771739 FALSE FALSE  
## accel\_forearm\_x 1.034483 6.59816576 FALSE FALSE  
## accel\_forearm\_y 1.080645 8.15217391 FALSE FALSE  
## accel\_forearm\_z 1.032609 4.61107337 FALSE FALSE  
## magnet\_forearm\_x 1.000000 12.07540761 FALSE FALSE  
## magnet\_forearm\_y 1.142857 15.18342391 FALSE FALSE  
## magnet\_forearm\_z 1.027027 13.44259511 FALSE FALSE  
## classe 1.469065 0.04245924 FALSE FALSE

how about correlations/multicollinearity? : a lot of blue -> multicollinearity

corrplot(cor(train[,-53]), method="color")

 let's have a look at the first 13 rows

corrplot(cor(train[,c(1:13)]), method="color")



# make a model

classification model: Tree. this model is not accurate (was aspected because of multicollinearity)

modFitT <- train(classe ~ ., method = "rpart", data=train)  
finModT <- modFitT$finalModel  
print(modFitT)

## CART   
##   
## 11776 samples  
## 52 predictor  
## 5 classes: 'A', 'B', 'C', 'D', 'E'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.03998576 0.5024092 0.3534124  
## 0.04224015 0.4757915 0.3104877  
## 0.11722829 0.3238448 0.0598947  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.03998576.

# classification model: random forest

let's try a random forest-model. This model can handle multicollinearity.

short discription: the random forest-model creates a lot of trees. Each tree is build on a different sample of the data (bootstrapping). A each node of the tree is using a random set of m variables that may contribute to the split. The trees are voted to predict an outcome.

Because it's very slow we use clusters.

The chosen resampling method is cross validation (10-fold).

cv 5-fold: accuracy 0,99 almost the same as 10-fold. for k=10 the bias-variance balance is a little bit beter. also cv 30-fold has almost the same accuracy The model seems to be stable. I use k=10.

set.seed(123)  
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS  
registerDoParallel(cluster)  
  
  
fitControl <- trainControl(method = "cv",  
 number = 10,  
 allowParallel = TRUE)  
  
modFitRf <- train(classe ~ ., method = "rf", data=train, trControl=fitControl)  
  
stopCluster(cluster)  
registerDoSEQ()

confusionmatrix:

finModRf <- modFitRf$finalModel  
finModRf

##   
## Call:  
## randomForest(x = x, y = y, mtry = param$mtry)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 0.93%  
## Confusion matrix:  
## A B C D E class.error  
## A 3343 4 0 0 1 0.001493429  
## B 16 2256 7 0 0 0.010092146  
## C 0 30 2018 6 0 0.017526777  
## D 1 0 38 1889 2 0.021243523  
## E 0 0 1 4 2160 0.002309469

rf performs good and is accurate: 0,9907

print(modFitRf)

## Random Forest   
##   
## 11776 samples  
## 52 predictor  
## 5 classes: 'A', 'B', 'C', 'D', 'E'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 10599, 10599, 10598, 10598, 10599, 10598, ...   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.9906586 0.9881822  
## 27 0.9904039 0.9878619  
## 52 0.9846296 0.9805559  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 2.

# out of sample error

the accuracy of the testset is 0,9916 (see the statistics) the out of sample error is 1-0,9916 =0,0084 the accuracy of the testset is (higer than)almost) the same as the accuracy of the train-set

pred <- predict(modFitRf, test)  
confusionMatrix(pred, test$classe)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 2228 11 0 0 0  
## B 4 1503 14 0 0  
## C 0 4 1354 26 2  
## D 0 0 0 1259 4  
## E 0 0 0 1 1436  
##   
## Overall Statistics  
##   
## Accuracy : 0.9916   
## 95% CI : (0.9893, 0.9935)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9894   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9982 0.9901 0.9898 0.9790 0.9958  
## Specificity 0.9980 0.9972 0.9951 0.9994 0.9998  
## Pos Pred Value 0.9951 0.9882 0.9769 0.9968 0.9993  
## Neg Pred Value 0.9993 0.9976 0.9978 0.9959 0.9991  
## Prevalence 0.2845 0.1935 0.1744 0.1639 0.1838  
## Detection Rate 0.2840 0.1916 0.1726 0.1605 0.1830  
## Detection Prevalence 0.2854 0.1939 0.1767 0.1610 0.1832  
## Balanced Accuracy 0.9981 0.9936 0.9924 0.9892 0.9978

# the most important variables

varImp(modFitRf)

## rf variable importance  
##   
## only 20 most important variables shown (out of 52)  
##   
## Overall  
## roll\_belt 100.00  
## yaw\_belt 83.78  
## magnet\_dumbbell\_z 71.57  
## magnet\_dumbbell\_y 66.96  
## pitch\_belt 63.12  
## pitch\_forearm 60.77  
## roll\_forearm 55.62  
## magnet\_dumbbell\_x 54.85  
## accel\_belt\_z 51.07  
## magnet\_belt\_z 47.16  
## roll\_dumbbell 46.54  
## accel\_dumbbell\_y 46.09  
## magnet\_belt\_y 44.91  
## accel\_dumbbell\_z 38.68  
## roll\_arm 35.20  
## accel\_forearm\_x 33.96  
## gyros\_dumbbell\_y 31.58  
## yaw\_dumbbell 31.52  
## accel\_dumbbell\_x 31.38  
## magnet\_arm\_x 30.85

# now we predict on the testingset

predT <- predict(modFitRf, testing)  
predT

## [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