An Analysis of the Weight Lifting Exercises Dataset

Peer Assessment Project for Practical Machine Learning Course

Author: Monika Traple

Introduction

The goal of the project is to create the machine learning model enabling to predict the manner, in which people do their sport activities. The model is based on the data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The data for this project come from the source: linked phrase.

Requaring and reading in the data

```
#Training data
trainURL<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
download.file(trainURL, destfile = "~/Desktop/pml-training.csv", method = "curl")

#At the initial reading in of the data the following characters were identified as NA, ie. "NA", "", "D

training<-read.csv("~/Desktop/pml-training.csv", na.strings = c("NA", "", "DIV/0!"))


#Testing data
testURL<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(testURL, destfile = "~/Desktop/pml-testing.csv", method = "curl")
testing<-read.csv("~/Desktop/pml-testing.csv", na.strings = c("NA", "", "DIV/0!"))
```

Cleaning dataset and selecting variables for further analysis

- Removing the variables used only either to identify the record (variable "X") or the participant (variable "user_name"), i.e. the variables, which contain no information on the sport activity.

```
training<-training[, -(1:2)] #in training dataset
```

- Dealing with missing values

```
#Calculating the proportion of missing values in each variable
check_mv<-sapply(training,
        function(x){colSums(is.na(as.matrix(x)))/nrow(training)})
#Removing variables containing more than 80% of missing values
training<-training[, which(check_mv<0.8)]
```

- Removing near zero covariates

```
library(caret)
nzv<-nearZeroVar(training, saveMetrics = TRUE)
names_nzv<-row.names(nzv[which(nzv$nzv==TRUE), ])
#removing from training set
training<-training[, -which(names(training) %in% names_nzv)]
```

Sampling for cross-validation

```
set.seed(1234)
inTrain<-createDataPartition(training$classe, p=0.6, list=FALSE)
training<-training[inTrain, ]
validation<-training[-inTrain, ]
```

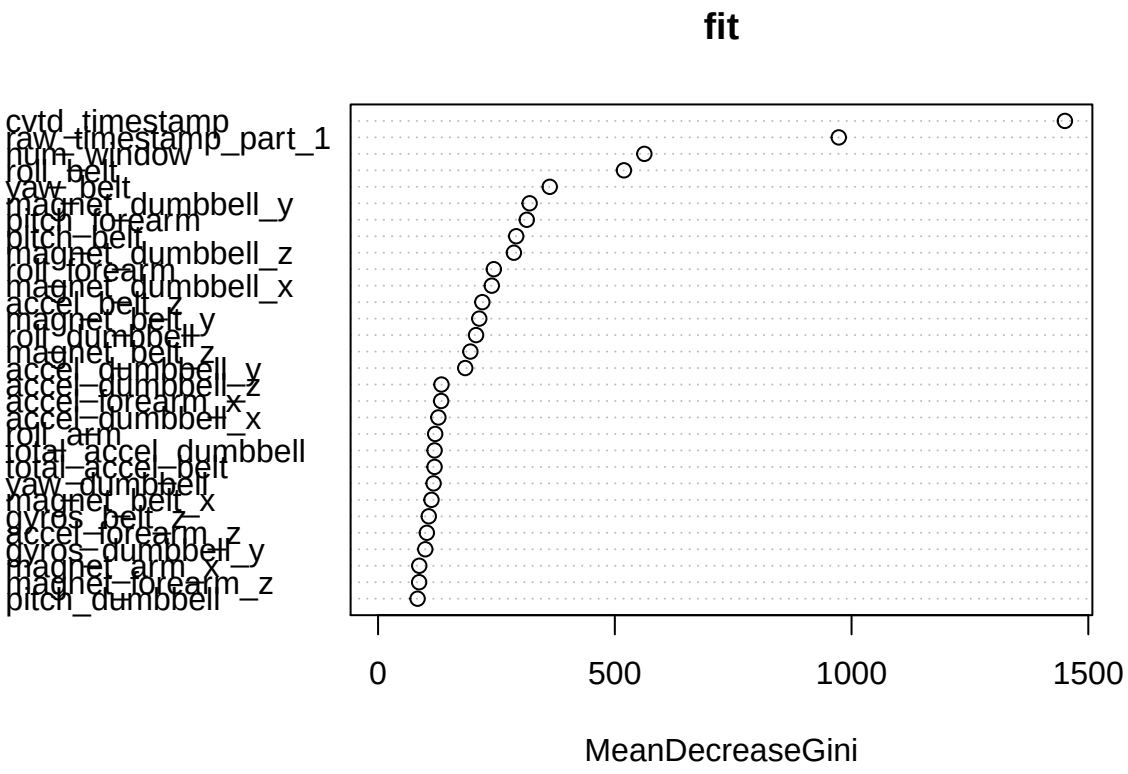
Fitting the model

As the dataset consists of variables of many type the random forest method for modeling was chosen. This method do not require any specific preprocessing.

```
library(randomForest)
fit<-randomForest(classe~., data=training)
```

The model shows following importance of the variables included:

```
varImpPlot(fit)
```



Evaluating the model

- Firstly we can check the prediction ability of the model on the training dataset itself:

```
confusionMatrix(training$classe, predict(fit, training))$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    0    0    0
```

```
##          C    0    0 2054    0    0
##          D    0    0    0 1930    0
##          E    0    0    0    0 2165
```

```
confusionMatrix(training$classe, predict(fit, training))$overall[1]
```

```
## Accuracy
##        1
```

As it shows above the model has a perfect accuracy on the training set.

- Secondly we can check the accuracy of the model on the validation dataset.

Modifying validation dataset to encompass exactly the same variables as in training dataset.

```
neededvariables<-names(training)
validation<-validation[, which(names(validation) %in% neededvariables)]
```

Checking the accuracy on the validation dataset

```
pred<-predict(fit, validation)
confusionMatrix(validation$classe, pred)$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1346    0    0    0    0
##          B    0  904    0    0    0
##          C    0    0  814    0    0
##          D    0    0    0  774    0
##          E    0    0    0    0  854
```

```
confusionMatrix(validation$classe, pred)$overall[1]
```

```
## Accuracy
##        1
```

The model shows also 100% accuracy on validation dataset.

Predicting on the testing dataset

Modifying testing dataset to encompass exactly the same variables as in training dataset and the same factor levels in factor variable

```
testing<-testing[, which(names(testing) %in% neededvariables)]
levels(testing$cvtd_timestamp) <- levels(training$cvtd_timestamp)
```

Making a prediction

```
predict(fit, testing)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```