Prediction Assignment Writeup

Abe1985

Sunday, December 14, 2014

Introduction

This is an R Markdown document that describes my analysis and my code that predicts how well people do a certain exercise. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, the goal was to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E).This was documented by the the "classe" variable. For more details on the data and how it was collected see the Appendix.

The Model was obtained by random forest using cross validation. The in sample error rate is 0.004% and the estimated out of sample error is 0.02%.

Executive summary

This file describes how the prediction of the manner in which subjects did exercise was conducted and how well it works. In the dataset used, I used the following variables of the training set to predict it. The prediction model was used to predict 20 different test cases.

Data preperation

1st removing missing values

First, I read the training set into R and removed the missing values, that are represented by NAs:

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.2

## Loading required package: lattice
## Loading required package: ggplot2
```

```
trainraw <- read.csv("pml-training.csv")
colneeded <- c() #creates an empty vector that will contain all columns needed
k <- 0 #variable needed to count
#if the column contains more than 19 Na's it will be excluded in our count. Only the number
#of the columns with fewerer NA's will be stored
for(i in 1:length(names(trainraw))){
        if(sum(is.na(trainraw[i]))>19){next}
        else{ k <- k + 1
                colneeded[[k]] <- i}
        }
trains <- trainraw[,colneeded]
sum(is.na(trains)) #make sure no Nas are left
```

```
## [1] 0
```

The training set has 60 values left that contain values. Now we need to identify those variables that can be useful for predicting the effort with which the excersive was performed.

2nd remove near zero values

```
library(caret)
nsv <- nearZeroVar(trains, saveMetrics=FALSE)
#all numbers of the variables that are near zero are stored in nsv
trainset <- trains[,(-nsv)]
```

The commands above will delete all columns that contain near zero values, meaning variables that nearly remain the same over all subjects and are therefore useless for prediction.

3rd define factor variable

The outcome is represented by the "classe" variable. It can take the value of 6 different classes and is therefore a factor variable.

```
trainset$classe <- as.factor(trainset$classe)
```

4th split dataset

In order to evaluate the model and evaluate the error rate, I split the data in a testing and a training set:

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.2
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(444)
inTrain <- createDataPartition(y=trainset$classe, p=0.75, list=FALSE)
training <- trainset[inTrain,]
testing <- trainset[-inTrain,]
```

Prediction Model

Because with randomforest it is very likely to overfit the model it is important to use cross validation (cv). I used the trControl argument to include cv.

```
modelfit <- train(as.factor(classe) ~., method="rf", trControl = trainControl(method="cv", number=3), da
```

```
getTree(modelfit$finalModel,k=1)
```

```
##    left daughter right daughter split var split point status prediction
## 1              2              3        29   1.305e+02      1          0
## 2              4              5        69  -3.305e+01      1          0
## 3              6              7        31   1.595e+02      1          0
## 4              8              9        80   7.780e+02      1          0
```

```
## 5             10             11         1   5.580e+03      1          0
## 6              0              0         0   0.000e+00     -1          5
## 7             12             13         7   1.323e+09      1          0
## 8             14             15         1   6.024e+03      1          0
## 9              0              0         0   0.000e+00     -1          2
## 10             0              0         0   0.000e+00     -1          1
## 11            16             17         1   9.377e+03      1          0
## 12             0              0         0   0.000e+00     -1          1
## 13             0              0         0   0.000e+00     -1          5
## 14             0              0         0   0.000e+00     -1          1
## 15             0              0         0   0.000e+00     -1          2
## 16             0              0         0   0.000e+00     -1          2
## 17            18             19         1   1.280e+04      1          0
## 18             0              0         0   0.000e+00     -1          3
## 19            20             21         1   1.602e+04      1          0
## 20             0              0         0   0.000e+00     -1          4
## 21             0              0         0   0.000e+00     -1          5
```

Error rate

```
modelfit$finalModel
```

```
##
## 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: 41
##
##         OOB estimate of  error rate: 0.02%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4185    0    0    0    0   0.0000000
## B    1 2847    0    0    0   0.0003511
## C    0    1 2566    0    0   0.0003896
## D    0    0    0 2412    0   0.0000000
## E    0    0    0    1 2705   0.0003695
```

The in sample error is below 0.004%. I did a prediction on my testset:

```
pred <- predict(modelfit, newdata=testing)
table(pred,testing$classe)
```

```
##
## pred    A    B    C    D    E
##    A 1395    0    0    0    0
##    B    0  949    0    0    0
##    C    0    0  855    0    0
##    D    0    0    0  804    0
##    E    0    0    0    0  901
```

The model did a perpect fit, so that the sample error for my testset was 0.0%.

The out of sample error is estimated to be 0.02%. I guess that it might slighly larger due to overfitting.

Appendix

The test and the training set were provided from http://groupware.les.inf.puc-rio.br/har and are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.