Practical Machine Learning Course - Final Project

Jurgen Tas

12 Jun 2015

Introduction

Devices such as Jawbone Up, Nike FuelBand, and Fitbit can measure a large amount of personal excersise data. One thing that people regularly do is quantify how much of a particular activity they do. However, they rarely quantify how well they do it. The goal of this project is to predict the manner in which they did the exercise. To this end, we analyse data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The participants were asked to perform barbell lifts correctly and incorrectly in five different ways. For more information we refer to: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

Analysis

First, we import the raw data using:

```
# import raw data:
pml.training.raw = read.csv("pml-training.csv", header=TRUE, na.strings=c("NA","Na
N", '') )
pml.testing.raw = read.csv("pml-testing.csv", header=TRUE, na.strings=c("NA","NaN"
, " ") )
```

Inspection of this data reveals that there various columns containing only NA's or empty entries. Thus, we decide to remove these columns from both the data sets using:

```
# commands to clean the raw data:
pml.training = subset(pml.training.raw , select=colMeans(is.na(pml.training.raw))
== 0)
pml.testing = subset(pml.testing.raw , select=colMeans(is.na(pml.testing.raw)) ==
0)
```

The first seven columns of both resulting data sets do not contain accelerometer measurements or the classe outcome. So, we remove these columns as well.

```
# remove first 7 columns:
pml.training = pml.training[c(-1:-7)]
pml.testing = pml.testing[c(-1:-7)]
```

Next, we create a 30-70 partition of the pml.training data.

```
# load caret package:
library(caret)
```

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

```
# Set random seed:
set.seed(3456)
# create validation/train data sets:
inTrain = createDataPartition(pml.training$classe, p = .7, list = FALSE)
training = pml.training[inTrain,]
validation = pml.training[-inTrain,]
```

We apply the random forest algorithm to the training data set. This algorithm is one of the most accurate prediction algorithms. The training set is randomly split into 5 folds. We train the model on 4/5 of the data, and check its accuracy on the 1/5 of the data we left out. This procedure is repeated with each split of the data. After building the model, we test the predictive power on the validation set.

```
# load random forest package:
library(randomForest)
```

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

```
# Set random seed:
set.seed(6543)
# Use 5 folds for cross-validation:
trControl = trainControl(method = "cv", number = 5)
# apply random forest model:
modFit = train(classe ~ ., data=training, method="rf", trControl = trControl, prox
=TRUE, ntree=50)
# print model outcome:
print(modFit)
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 10990, 10989, 10990, 10990, 10989
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9895172  0.9867380  0.002906131  0.003677173
##   27    0.9902454  0.9876602  0.001009402  0.001276200
##   52    0.9853683  0.9814885  0.002640259  0.003341525
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
print(modFit$finalModel)
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 50, mtry = param$mtry, proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.92%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3891   11    2    1    1 0.003840246
## B   24 2617   14    2    1 0.015425132
## C    0   15 2372    9    0 0.010016694
## D    0    2   23 2225    2 0.011989343
## E    1    4    4   10 2506 0.007524752
```

```
# predict outcome for validation data set:
pred = predict(modFit,validation )
# print confusion matrix:
confusionMatrix(pred, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673   10    0    0    0
##          B    0 1125    0    0    0
##          C    1    4 1018   10    1
##          D    0    0    8  954    1
##          E    0    0    0    0 1080
##
## Overall Statistics
##
##                Accuracy : 0.9941
##                  95% CI : (0.9917, 0.9959)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9925
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9877   0.9922   0.9896   0.9982
## Specificity            0.9976   1.0000   0.9967   0.9982   1.0000
## Pos Pred Value         0.9941   1.0000   0.9845   0.9907   1.0000
## Neg Pred Value         0.9998   0.9971   0.9984   0.9980   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1912   0.1730   0.1621   0.1835
## Detection Prevalence   0.2860   0.1912   0.1757   0.1636   0.1835
## Balanced Accuracy      0.9985   0.9939   0.9945   0.9939   0.9991
```

The estimated out-of-sample accuracy of the model is 99.41%. Thus, the estimated out-of-sample error is 100% - 99.41% = 0.59%. The last step is to apply the model to each of the 20 test cases in the testing data set:

```
# predict outcome for test data set using model:
outcome = predict(modFit,pml.testing[,-53])
outcome
```

```
##  [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
```

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
pml_write_files(outcome)
```