Prediction Assignment Writeup

mpmartins1970

2016/08/12

Executive Summary

In this report, data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants are used to quantify how well they did each particular activity. The goal of this report is to predict the manner in which they did the exercise, describing how the model was built, the expected out of sample error and why the choices were made. The predict model is used to predict 20 different test cases. Using exploratory data analysis and machine learning theory, the findings showed that Random Forest has the best accuracy for this testing dataset.

Exploratory Data Analysis

Loading the training and test datasets that previously have been downloaded to working directory in local machine. The datasets have 160 variables.

```
# Libraries
library(caret)
```

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

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

```
# Load the training and testing dataset
trainingDS <- read.csv("./pml-training.csv", na.strings = c("NA","#DIV/0!"))
testingDS <- read.csv("./pml-testing.csv", na.strings = c("NA","#DIV/0!"))
```

Removing all the variables containing NA values, the near zero variance (NVZ) variables and id/timestamp variables.

```
# Cleaning data
trainingDS <- trainingDS[,colSums(is.na(trainingDS)) == 0]
trainingDS <- trainingDS[,-nearZeroVar(trainingDS)]
trainingDS <- trainingDS[,-c(1:7)]
```

Splitting data into a 70% training data set and a 30% testing data set to estimate the out of sample error of the predictor.

```
# Splitting data
inTrain <- createDataPartition(y = trainingDS$classe, p = 0.70,list = F)
training <- trainingDS[inTrain,]
testing <- trainingDS[-inTrain,]
```

Prediction Model Building

The problem to be resolved is a classification one, so, using random forest, the out of sample error should be small. Random forest is used for the training dataset using cross-validation.

```
# Making report reproducible
set.seed(1970)

# Fitting Random Forest model
trc5 <- trainControl(method = "cv", number = 5, allowParallel = TRUE, verbose = TRUE)
modelRF <- train(classe~.,data = training, method = "rf", trControl = trc5, verbose = FALSE)
```

```
## Loading required package: 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
```

```
## + Fold1: mtry= 2
## - Fold1: mtry= 2
## + Fold1: mtry=26
## - Fold1: mtry=26
## + Fold1: mtry=51
## - Fold1: mtry=51
## + Fold2: mtry= 2
## - Fold2: mtry= 2
## + Fold2: mtry=26
## - Fold2: mtry=26
## + Fold2: mtry=51
## - Fold2: mtry=51
## + Fold3: mtry= 2
## - Fold3: mtry= 2
## + Fold3: mtry=26
## - Fold3: mtry=26
## + Fold3: mtry=51
## - Fold3: mtry=51
## + Fold4: mtry= 2
## - Fold4: mtry= 2
## + Fold4: mtry=26
## - Fold4: mtry=26
## + Fold4: mtry=51
## - Fold4: mtry=51
## + Fold5: mtry= 2
## - Fold5: mtry= 2
## + Fold5: mtry=26
```

```
## - Fold5: mtry=26
## + Fold5: mtry=51
## - Fold5: mtry=51
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 26 on full training set
```

```
modelRF$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, verbose = FALSE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 26
##
##         OOB estimate of  error rate: 0.81%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3901    4    0    0    1 0.001280082
## B   24 2623    9    0    2 0.013167795
## C    0   21 2368    7    0 0.011686144
## D    0    0   30 2219    3 0.014653641
## E    0    0    2    8 2515 0.003960396
```

In sequence, the fitted model generated is examined with the testing sample from the partitioned training dataset to evaluate the accuracy and estimated error of prediction.

```
# Predicting with the Random Forest Model
predictionRF <- predict(modelRF, testing)
confMatrixRF <- confusionMatrix(predictionRF, testing$classe)
confMatrixRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    8    0    0    0
##          B    0 1130    4    0    0
##          C    1    1 1017   10    4
##          D    0    0    5  953    0
##          E    0    0    0    1 1078
##
## Overall Statistics
##
##                Accuracy : 0.9942
##                  95% CI : (0.9919, 0.996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9927
##  Mcnemar's Test P-Value : NA
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9921   0.9912   0.9886   0.9963
## Specificity            0.9981   0.9992   0.9967   0.9990   0.9998
## Pos Pred Value         0.9952   0.9965   0.9845   0.9948   0.9991
## Neg Pred Value         0.9998   0.9981   0.9981   0.9978   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1920   0.1728   0.1619   0.1832
## Detection Prevalence   0.2856   0.1927   0.1755   0.1628   0.1833
## Balanced Accuracy      0.9988   0.9956   0.9940   0.9938   0.9980
```

The accuracy of the modeling method above is 99,2% with a small out of sample error. As a consequence, it could be expected almost all of the submitted test cases will be correct.

Predicting with the Testing Data

Applying the Random Forest model to predict the 20 test cases provided (testing dataset) as shown below.

```
# Predicting using pml-testing.csv data
predictTestingDS <- predict(modelRF, newdata = testingDS)
predictTestingDS
```

```
##  [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
```

Conclusions

From these data, after validation in the prediction quizz, the goal was accomplished since all the submitted test cases were correct.
