Practical Machine Learning Course Project

Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. 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 is 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.

Class A - exactly according to the specification; Class B - throwing the elbows to the front; Class C - lifting the dumbbell only halfway; Class D - lowering the dumbbell only halfway; Class E - throwing the hips to the front.

More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

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.

The following packages must be installed in advance: caret, e1071 and randomForest.

```r
library (caret)
library (e1071)
library(randomForest)
```

We download and read the training and test data.

```r
fileUrl_train = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
download.file(fileUrl_train, destfile="training_data.csv", method="curl")

fileUrl_test = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(fileUrl_test, destfile="testing_data.csv", method="curl")

train_data = read.csv("training_data.csv")
test_data = read.csv("testing_data.csv")
```

Cleaning data

We remove the first 6 columns which are unnecessary for the analysis:

```r
selected_train_data <- train_data[, c(7:160)]
```

We check for NAs in every column:

```r
colSums(is.na(selected_train_data))
```

We realize that some columns have 19216 NAs while others have none.

```r
NAs_per_column_train <- colSums(is.na(selected_train_data))
```

We remove these columns:

```r
cleaned_train_data <- selected_train_data[!NAs_per_column_train == 19216]
```

We also remove those columns with very few values. We identify them with the nearZeroVar function from the caret package.

```r
delete_columns_train<- nearZeroVar(cleaned_train_data)
cleaned_train_data <- cleaned_train_data[-delete_columns_train]
```

We clean the test data in the same way:

```r
selected_test_data <- test_data[, c(7:160)]
```

```r
colSums(is.na(selected_test_data))
```

There are columns with 20 NAs and others with none:

```r
NAs_per_columns_test <- colSums(is.na(selected_test_data))
cleaned_test_data <- selected_test_data[!NAs_per_columns_test == 20]
```

Model building

We create two subsets from the cleaned training dataset: one for training and the other one for testing the model:

```r
intrain = sample(1:dim(cleaned_train_data)[1], size=dim(cleaned_train_data)[1] * 0.8, replace=F)
train = cleaned_train_data[intrain,]
test = cleaned_train_data[-intrain,]
```

We will build our model using the Random Forest machine learning technique:

```r
rf <- randomForest(classe ~ ., data=train)
```

We calculate the in sample accuracy (prediction accuracy of our model on the training data set):

```r
training_pred <- predict(rf, train)
confusionMatrix(training_pred, train$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4518    0    0    0    0
##          B    0 3031    0    0    0
##          C    0    0 2713    0    0
##          D    0    0    0 2577    0
##          E    0    0    0    0 2858
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (1, 1)
##     No Information Rate : 0.288
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    1.000    1.000    1.000    1.000
## Specificity             1.000    1.000    1.000    1.000    1.000
## Pos Pred Value          1.000    1.000    1.000    1.000    1.000
## Neg Pred Value          1.000    1.000    1.000    1.000    1.000
## Prevalence              0.288    0.193    0.173    0.164    0.182
## Detection Rate          0.288    0.193    0.173    0.164    0.182
## Detection Prevalence    0.288    0.193    0.173    0.164    0.182
## Balanced Accuracy       1.000    1.000    1.000    1.000    1.000
```

Now we calculate the out of sample accuracy (prediction accuracy of our model on the testing data set):

```r
testing_pred <- predict(rf, test)
confusionMatrix(testing_pred, test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1062    1    0    0    0
##          B    0  765    4    0    0
##          C    0    0  705    4    0
##          D    0    0    0  635    1
##          E    0    0    0    0  748
##
## Overall Statistics
##
##                Accuracy : 0.997
##                  95% CI : (0.995, 0.999)
##     No Information Rate : 0.271
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.997
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    0.999    0.994    0.994    0.999
## Specificity             1.000    0.999    0.999    1.000    1.000
## Pos Pred Value          0.999    0.995    0.994    0.998    1.000
## Neg Pred Value          1.000    1.000    0.999    0.999    1.000
## Prevalence              0.271    0.195    0.181    0.163    0.191
## Detection Rate          0.271    0.195    0.180    0.162    0.191
## Detection Prevalence    0.271    0.196    0.181    0.162    0.191
## Balanced Accuracy       1.000    0.999    0.997    0.997    0.999
```

Finally, we apply the machine learning algorithm to each of the 20 provided test cases and we write the answers to files as specified in the assignment:

```
answers <- predict(rf, cleaned_test_data)
```

```
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(answers)
```

We get: B A B A A E D B A A B C B A E E A B B B