Prediction assignment

1. Overview

This project uses machine learning algorithms to predict the maner the wearer of a human activity recognition device performed barbell lifts. It uses the Weight Lifting Exercises Dataset (Velloso et al., 2013). Two models are tested using decision tree and random forest.

2. Data

2.1 Summary

Data comes from the Weight Lifting Exercises Dataset. This dataset contains data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants that were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The "classe" variable refers to how the lift was performed. Its values stand for the following:

- 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 this website.

2.2 Exploratory data analysis and clean-up

First, we load and clean up the data from the URLs.

```
train_url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training <- read.csv(url(train_url), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(test_url), na.strings=c("NA","#DIV/0!",""))
```

Then we remove the columns with all NA values and the ones not relevant for our analysis (X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp, new_window, num_window).

```
training <- training[, colSums(is.na(training)) == 0]
training <- training[,-c(1:7)]
testing <- testing[, colSums(is.na(testing)) == 0]
testing <- testing[,-c(1:7)]
```

2.3 Partitioning data set for cross-validation

Cross-validation is performed by splitting the original training data set randomly without replacement into 2 subsets: TrainTraining (75% of the original set) and TestTraining (25%). The models will be fitted on the TrainTraining set, and tested on the TestTraining set. Once the most accurate model is established, it will be tested on the original testing data set.

```
set.seed(234)
require(caret)
inTrain <- createDataPartition(y=training$classe, p=0.75, list=FALSE)
trainTraining <- training[inTrain,]
testTraining <- training[-inTrain,]
```

2.4 Expected out-of-sample error

Accuracy is the proportion of correctly classified observations over the total sample in the testTraining dataset (the cross-validation data set). Expected accuracy refers to the accuracy anticipated in the testing data set (the out-of sample data set). The expected value of the out-of-sample error (the misclassification rate) corresponds to the expected number of missclassified observations in relation to the total number of observations in the testing data set, which is 1 minus the accuracy found in the cross-validation data set.

3. Prediction models

Two models will be tested using decision tree and random forest. The model with the highest accuracy will be chosen as the final model.

3.1 Decision tree

We fit a model using decision tree.

```
require(rpart)
require(e1071)
modFit_dt <- rpart(classe ~ ., data=trainTraining, method="class")
```

Now let's predict the in-sample-error.

```
pred_dt <- predict(modFit_dt, testTraining, type = "class")
cm_dt <- confusionMatrix(pred_dt, testTraining$classe)
cm_dt
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1292  222   67  140   39
##          B   34  494   60   28   62
##          C   27   67  645  126  107
##          D   22   74   57  446   53
##          E   20   92   26   64  640
##
## Overall Statistics
##
##                Accuracy : 0.7172
##                  95% CI : (0.7043, 0.7297)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6389
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9262   0.5205   0.7544  0.55473   0.7103
## Specificity            0.8666   0.9535   0.9192  0.94976   0.9495
## Pos Pred Value         0.7341   0.7286   0.6636  0.68405   0.7601
## Neg Pred Value         0.9672   0.8923   0.9466  0.91580   0.9357
## Prevalence             0.2845   0.1935   0.1743  0.16395   0.1837
```

```
## Detection Rate         0.2635   0.1007   0.1315  0.09095   0.1305
## Detection Prevalence   0.3589   0.1383   0.1982  0.13295   0.1717
## Balanced Accuracy      0.8964   0.7370   0.8368  0.75224   0.8299
```

3.2 Random forest

A model is fitted using random forest.

```
require(randomForest)
modFit_rf <- randomForest(classe ~. , data=trainTraining, na.action=na.omit)
```

The prediction of the in-sample-error is performed below.

```
pred_rf <- predict(modFit_rf, testTraining, type = "class")
cm_rf <- confusionMatrix(pred_rf, testTraining$classe)
cm_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    3    0    0    0
##          B    0  946    8    0    0
##          C    0    0  846   11    0
##          D    0    0    1  792    1
##          E    0    0    0    1  900
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9925, 0.9967)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9968   0.9895   0.9851   0.9989
## Specificity            0.9991   0.9980   0.9973   0.9995   0.9998
## Pos Pred Value         0.9979   0.9916   0.9872   0.9975   0.9989
## Neg Pred Value         1.0000   0.9992   0.9978   0.9971   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1929   0.1725   0.1615   0.1835
## Detection Prevalence   0.2851   0.1945   0.1748   0.1619   0.1837
## Balanced Accuracy      0.9996   0.9974   0.9934   0.9923   0.9993
```

3.3 Best prediction model

The Random Forest algorithm performed better than Decision Tree. The accuracy for the Random Forest model is 0.9949021 (95% CI: NA), whereas the accuracy for the Decision tree model is 0.7171697 (95% CI: NA). The expected out-of-sample error is estimated 0.2828303. Since the accuracy is very high in the cross-validation data, it is expected that the accuracy will be high as well in the test set.

4. Test model

Now we test the Random forest model agains the test set.

```
pred_final <- predict(modFit_rf, testing, type="class")
pred_final
```

```
##  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
```