Practical Machine Learning 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, your goal will be 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.

The goal of project is to predict the manner in which they did the exercise, using 20 different test cases (pml-testing.csv).

Data loading and preparation

First step is loading the data and have a look on its structure and values. (not printed here):

```
train = read.csv("pml-training.csv")
test = read.csv("pml-testing.csv")

head(train)
summary(train)
str(train)
```

Based on the first view, the next data preparation issues were identified:

- string variables, with "", "#DIV/0!" values
- variables including mostly null values
- 1-7 cols not relevant data for a general model (obs. id, name of participants, etc.)

Applied data preparation process to manage the issues:

- replace "#DIV/0!" values with "NA" values
- remove all columns with "NA" values (including impacted variables in previos step)
- remove the 1-7 columns with not desired content

```
trainProc = as.data.frame(lapply(train, function(x) if(is.character(x)|is.factor(x)) gsub("#DIV/0!", NA
trainValid = trainProc[, colSums(is.na(trainProc)) == 0]
trainValid = trainValid[,8:60]

summary(trainValid)
```

As I did not use any transformations which impacted the values of the remaining variables, modification of test set is not needed.

Modeling

Load used libraries:

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.1
```

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

```
## Warning: package 'ggplot2' was built under R version 3.2.1
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.2
```

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

Split the original "trainValid" dataset to creat subsets for learning and testing:

```r
set.seed(123)
split = createDataPartition(trainValid$classe, p=0.70, list=FALSE)
trainLearn = trainValid[split, ]
testLearn = trainValid[-split, ]
```

Create random forest model, using cross-validation:

```r
set.seed(123)
numFolds = trainControl(method="cv", number = 3)
model = train(classe ~ ., data=trainLearn, method="rf", trControl=numFolds, ntree=100, nodesize=25)
```

To manage potential overfitting riks, cross-validation (3 fold) and limited nodesize (25) were applied. The numer of trees was limited (100) as well to reach reasonable modeling time.

Let's have a look on the model:

```r
model
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9157, 9160, 9157
## Resampling results across tuning parameters:
```

```
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9645485  0.9551313  0.003479010  0.004407765
##   27    0.9791798  0.9736621  0.002128973  0.002698303
##   52    0.9692805  0.9611411  0.002901434  0.003667669
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

To check the out-of-sample error, let's use the separated part of the dataset ("testLearn"):

```
predict = predict(model, testLearn)
```

The results are:

```
confusionMatrix(testLearn$classe, predict)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1669    4    1    0    0
##          B   11 1114    7    7    0
##          C    0   16 1007    3    0
##          D    0    0   28  935    1
##          E    0    0    2    3 1077
##
## Overall Statistics
##
##                Accuracy : 0.9859
##                  95% CI : (0.9825, 0.9888)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9822
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9935   0.9824   0.9636   0.9863   0.9991
## Specificity            0.9988   0.9947   0.9961   0.9941   0.9990
## Pos Pred Value         0.9970   0.9781   0.9815   0.9699   0.9954
## Neg Pred Value         0.9974   0.9958   0.9922   0.9974   0.9998
## Prevalence             0.2855   0.1927   0.1776   0.1611   0.1832
## Detection Rate         0.2836   0.1893   0.1711   0.1589   0.1830
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9961   0.9886   0.9799   0.9902   0.9990
```

The accuracy is 0.9859, the out-of-sample error is 0.0141.

Prediction for provided cases (20)

Prediction using the given test cases and the created model:

```
predict20 = predict(model, test)
```

The results are:

```
predict20
```

```
##  [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
```

Thank you for your time! :)