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, 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. More information is available from the website: (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here. The test data are available here.

The data for this project come from this source. Please cite them if you use this data to any further extent.

Goal

The goal of this project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. This is a report describing how I built the model, how I used cross validation, what I think the expected out of sample error is, and why I made the choices you did. I will also use my prediction model to predict 20 different test cases.

Building the Model

First, we set seed and load the data from the working directory and some packages that we'll use:

```
# Set seed
set.seed(101010)

# Load the training and testing data
pml_training <- read.csv("pml-training.csv", stringsAsFactors=FALSE)
pml_testing <- read.csv("pml-testing.csv", stringsAsFactors=FALSE)

# Load required packages
require(caret)
```

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

```
require(randomForest)
```

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

Remove classe field and zero variance, header, NA-ful and correlated predictors:

```
# Pick up fields that have near zero variance
zv_fields <- nearZeroVar(pml_training)

# Remove those fields from training and testing data
training_data <- pml_training[-zv_fields]
problem_data <- pml_testing[-zv_fields]

# Remove the header fields (index, user name, timestamps, window definitions)
header_fields <- 1:6
training_data <- training_data[-header_fields]
problem_data <- problem_data[-header_fields]

# Remove fields with NAs
na_fields <- which(sapply(training_data, function(x) sum(is.na(x)) > 0))
names(na_fields) <- rep(NULL, length(na_fields))
training_data <- training_data[-na_fields]
problem_data <- problem_data[-na_fields]

# Save off classe data
classe_data <- training_data$classe

# Remove classe data from training data
classe_field <- which(names(training_data) == "classe")
training_data <- training_data[-classe_field]
problem_data <- problem_data[-classe_field]

# Remove correlated fields
co_fields <- findCorrelation(cor(training_data), cutoff = 0.8)
training_data <- training_data[-co_fields]
problem_data <- problem_data[-co_fields]
```

Preprocess data and throw classe back on:

```
training_pre <- cbind(predict(preProcess(training_data,
                                         method = c("center", "scale", "knnImpute")),
                              training_data), classe_data)
names(training_pre)[dim(training_pre)[2]] <- "classe"
```

Create training and testing sets from the training data and preprocess training data:

```
# Partition pml_training into model training and testing data
in_training <- createDataPartition(y = classe_data, p = .8, list = FALSE)

# Pull in fully defined numeric fields (no username); we add classe back in later
training <- training_pre[in_training, ]
testing <- training_pre[-in_training, ]
```

Then create model from training data and test against testing data:

```r
  # Create model from training data
  pred_model <- randomForest(classe ~ ., data = training)

  # Test model with testing data
  testing_predictions <- predict(pred_model, testing)

  # Print expected error
  print(table(testing_predictions, testing$classe))
```

```
##
## testing_predictions    A    B    C    D    E
##                   A 1114    2    0    0    0
##                   B    1  756    7    0    0
##                   C    0    1  677    5    0
##                   D    0    0    0  638    1
##                   E    1    0    0    0  720
```

Make predictions:

```r
  problem_predictions <- predict(pred_model, problem_data)
```

Write predictions to file:

```r
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(problem_predictions)
```