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 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]. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

The classe variable contains 5 different ways barbell lifts were performed correctly and incorrectly:

- 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

Objective

The goal of this project is to predict the manner in which people performed barbell lifts. This is the classe variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Loading the data

Packages used for analysis. This assumes the packages are already installed. Use the install.packages("") command if a package not installed yet.

```r
library(caret)
```

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

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

```r
library(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
```

```r
library(rpart)
library(rpart.plot)
library(RColorBrewer)
```

Load the data into R

```r
# The location where the training data is to be downloaded from
trainUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
# The location where the testing data is to be downloaded from
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# Reading/loading the training data
train_data <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
# Reading/loading the testing data in your working directory
test_data <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))

# Take a look at the Training data classe variable
summary(train_data$classe)
```

```
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

Partitioning the data for Cross-validation

The training data is split into two data sets, one for training the model and one for testing the performance of our model. The data is partitioned by the classe variable, which is the varible we will be predicting. The data is split into 60% for training and 40% for testing.

```r
inTrain <- createDataPartition(y=train_data$classe, p = 0.60, list=FALSE)
training <- train_data[inTrain,]
testing <- train_data[-inTrain,]

dim(training)
```

```
## [1] 11776   160
```

```r
dim(testing)
```

```
## [1] 7846  160
```

Data Processing

Drop the first 7 variables because these are made up of metadata that would cause the model to perform poorly.

```r
training <- training[,-c(1:7)]
```

Remove NearZeroVariance variables

```r
nzv <- nearZeroVar(training, saveMetrics=TRUE)
training <- training[, nzv$nzv==FALSE]
```

There are a lot of variables where most of the values are 'NA'. Drop variables that have 60% or more of the values as 'NA'.

```r
training_clean <- training
for(i in 1:length(training)) {
  if( sum( is.na( training[, i] ) ) /nrow(training) >= .6) {
    for(j in 1:length(training_clean)) {
      if( length( grep(names(training[i]), names(training_clean)[j]) ) == 1)  {
        training_clean <- training_clean[ , -j]
      }
    }
  }
}

# Set the new cleaned up dataset back to the old dataset name
training <- training_clean
```

Transform the test_data dataset

```r
# Get the column names in the training dataset
columns <- colnames(training)
# Drop the class variable
columns2 <- colnames(training[, -53])
# Subset the test data on the variables that are in the training data set
test_data <- test_data[columns2]
dim(test_data)
```

```
## [1] 20 52
```

Cross-Validation: Prediction with Random Forest

A Random Forest model is built on the training set. Then the results are evaluated on the test set

```r
set.seed(54321)
modFit <- randomForest(classe ~ ., data=training)
prediction <- predict(modFit, testing)
cm <- confusionMatrix(prediction, testing$classe)
print(cm)
```

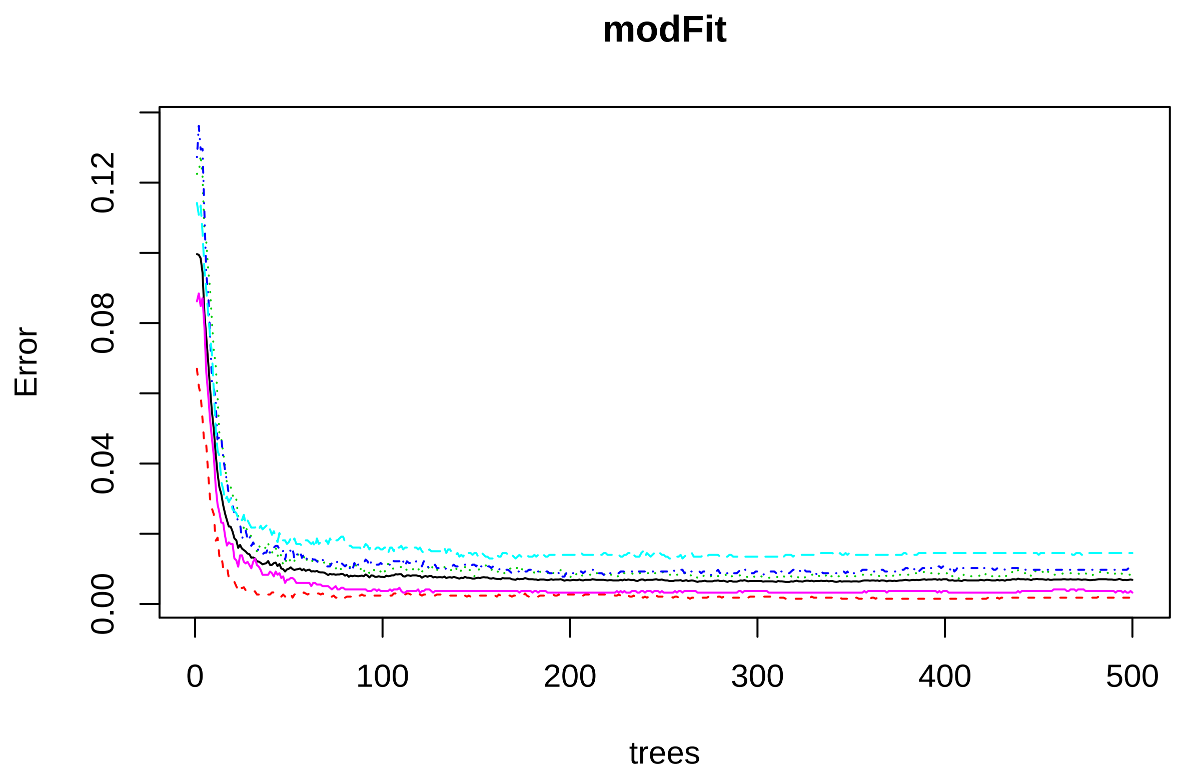
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2230    3    0    0    0
##          B    2 1515   11    1    0
##          C    0    0 1357   27    1
##          D    0    0    0 1258    2
##          E    0    0    0    0 1439
##
```

```
## Overall Statistics
##
##                Accuracy : 0.994
##                  95% CI : (0.992, 0.9956)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9924
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9980   0.9920   0.9782   0.9979
## Specificity            0.9995   0.9978   0.9957   0.9997   1.0000
## Pos Pred Value         0.9987   0.9908   0.9798   0.9984   1.0000
## Neg Pred Value         0.9996   0.9995   0.9983   0.9957   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1931   0.1730   0.1603   0.1834
## Detection Prevalence   0.2846   0.1949   0.1765   0.1606   0.1834
## Balanced Accuracy      0.9993   0.9979   0.9938   0.9890   0.9990
```

```
overall.accuracy <- round(cm$overall['Accuracy'] * 100, 2)
sam.err <- round(1 - cm$overall['Accuracy'],2)
```

The model is 99.40% accurate on the testing data partitioned from the training data. The expected out of sample error is roughly 0.01.

```
plot(modFit)
```



In the above figure, error rates of the model are plotted over 500 trees. The error rate is less than 0.04 for all 5 classe.

Cross-Validation: Prediction with a Decision Tree

```
set.seed(54321)
modFit2 <- rpart(classe ~ ., data=training, method="class")
prediction2 <- predict(modFit2, testing, type="class")
cm2 <- confusionMatrix(prediction2, testing$classe)
print(cm2)
```

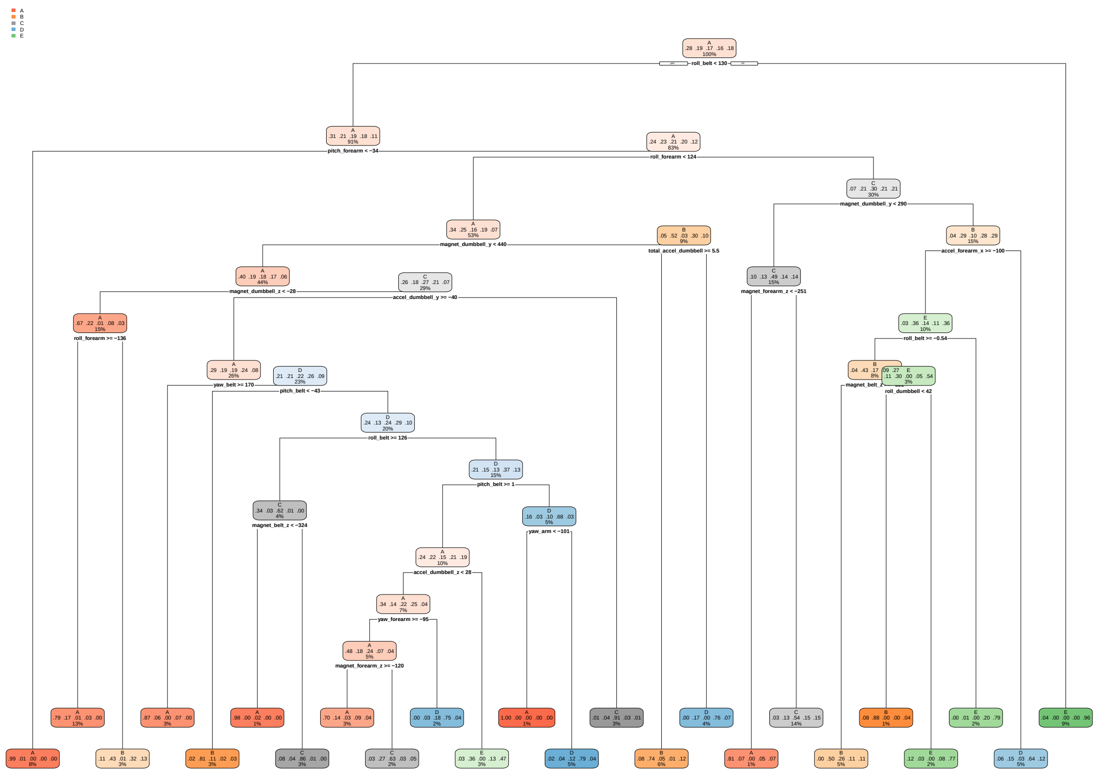
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2020  229   21   77   24
##          B   64  855  145  117  136
##          C   61  212 1122  203  156
##          D   22  114   75  813   84
##          E   65  108    5   76 1042
##
## Overall Statistics
##
##                Accuracy : 0.7459
##                  95% CI : (0.7361, 0.7555)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6779
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9050   0.5632   0.8202   0.6322   0.7226
## Specificity            0.9375   0.9270   0.9024   0.9550   0.9603
## Pos Pred Value         0.8520   0.6492   0.6397   0.7338   0.8040
## Neg Pred Value         0.9613   0.8985   0.9596   0.9298   0.9389
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2575   0.1090   0.1430   0.1036   0.1328
## Detection Prevalence   0.3022   0.1679   0.2236   0.1412   0.1652
## Balanced Accuracy      0.9212   0.7451   0.8613   0.7936   0.8415
```

```
overall.accuracy2 <- round(cm2$overall['Accuracy'] * 100, 2)
sam.err2 <- round(1 - cm2$overall['Accuracy'],2)
```

The model is 74.59% accurate on the testing data partitioned from the training data. The expected out of sample error is roughly 0.25.

Plot the decision tree model

```
rpart.plot(modFit2)
```



Prediction on the Test Data

The Random Forest model gave an accuracy of 99.40, which is much higher than the 74.59% accuracy from the Decision Tree. So we will use the Random Forest model to make the predictions on the test data to predict the way 20 participates performed the exercise.

```
final_prediction <- predict(modFit, test_data, type="class")
print(final_prediction)
```

```
##  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
```

Conclusions

There are many different machine learning algorithms. I chose to compare a Random Forest and Decision Tree model. For this data, the Random Forest proved to be a more accurate way to predict the manner in which the exercise was done.