Prediction Assignment Writeup

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 (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Project Aim

The aim of your project is to predict the way in which the exercise was done. The "classe" variable will be used from the training dataset. The project will describe how the model was built, how cross validation was done, what the expected out of sample error is, and why choices we made. The prediction model will be used to predict 20 different test cases.

Load Packages

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.3
```

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

```
## Warning: package 'lattice' was built under R version 3.1.3
```

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

```
library(kernlab)
library(randomForest)
```

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

```r
library(rpart)
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.1.3
```

```
## Loading required package: survival
## Loading required package: splines
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: parallel
## Loaded gbm 2.1.1
```

Read the datas

```r
training_data<- read.csv("pml-training.csv", na.strings=c("NA",""))
test_data<-read.csv("pml-testing.csv", na.strings=c("NA",""))
names(training_data)
```

```
##   [1] "X"                        "user_name"
##   [3] "raw_timestamp_part_1"     "raw_timestamp_part_2"
##   [5] "cvtd_timestamp"           "new_window"
##   [7] "num_window"               "roll_belt"
##   [9] "pitch_belt"               "yaw_belt"
##  [11] "total_accel_belt"         "kurtosis_roll_belt"
##  [13] "kurtosis_picth_belt"      "kurtosis_yaw_belt"
##  [15] "skewness_roll_belt"       "skewness_roll_belt.1"
##  [17] "skewness_yaw_belt"        "max_roll_belt"
##  [19] "max_picth_belt"           "max_yaw_belt"
##  [21] "min_roll_belt"            "min_pitch_belt"
##  [23] "min_yaw_belt"             "amplitude_roll_belt"
##  [25] "amplitude_pitch_belt"     "amplitude_yaw_belt"
##  [27] "var_total_accel_belt"     "avg_roll_belt"
##  [29] "stddev_roll_belt"         "var_roll_belt"
##  [31] "avg_pitch_belt"           "stddev_pitch_belt"
##  [33] "var_pitch_belt"           "avg_yaw_belt"
##  [35] "stddev_yaw_belt"          "var_yaw_belt"
##  [37] "gyros_belt_x"             "gyros_belt_y"
##  [39] "gyros_belt_z"             "accel_belt_x"
##  [41] "accel_belt_y"             "accel_belt_z"
##  [43] "magnet_belt_x"            "magnet_belt_y"
##  [45] "magnet_belt_z"            "roll_arm"
```

```
##  [47] "pitch_arm"                "yaw_arm"
##  [49] "total_accel_arm"          "var_accel_arm"
##  [51] "avg_roll_arm"             "stddev_roll_arm"
##  [53] "var_roll_arm"             "avg_pitch_arm"
##  [55] "stddev_pitch_arm"         "var_pitch_arm"
##  [57] "avg_yaw_arm"              "stddev_yaw_arm"
##  [59] "var_yaw_arm"              "gyros_arm_x"
##  [61] "gyros_arm_y"              "gyros_arm_z"
##  [63] "accel_arm_x"              "accel_arm_y"
##  [65] "accel_arm_z"              "magnet_arm_x"
##  [67] "magnet_arm_y"             "magnet_arm_z"
##  [69] "kurtosis_roll_arm"        "kurtosis_picth_arm"
##  [71] "kurtosis_yaw_arm"         "skewness_roll_arm"
##  [73] "skewness_pitch_arm"       "skewness_yaw_arm"
##  [75] "max_roll_arm"             "max_picth_arm"
##  [77] "max_yaw_arm"              "min_roll_arm"
##  [79] "min_pitch_arm"            "min_yaw_arm"
##  [81] "amplitude_roll_arm"       "amplitude_pitch_arm"
##  [83] "amplitude_yaw_arm"        "roll_dumbbell"
##  [85] "pitch_dumbbell"           "yaw_dumbbell"
##  [87] "kurtosis_roll_dumbbell"   "kurtosis_picth_dumbbell"
##  [89] "kurtosis_yaw_dumbbell"    "skewness_roll_dumbbell"
##  [91] "skewness_pitch_dumbbell"  "skewness_yaw_dumbbell"
##  [93] "max_roll_dumbbell"        "max_picth_dumbbell"
##  [95] "max_yaw_dumbbell"         "min_roll_dumbbell"
##  [97] "min_pitch_dumbbell"       "min_yaw_dumbbell"
##  [99] "amplitude_roll_dumbbell"  "amplitude_pitch_dumbbell"
## [101] "amplitude_yaw_dumbbell"   "total_accel_dumbbell"
## [103] "var_accel_dumbbell"       "avg_roll_dumbbell"
## [105] "stddev_roll_dumbbell"     "var_roll_dumbbell"
## [107] "avg_pitch_dumbbell"       "stddev_pitch_dumbbell"
## [109] "var_pitch_dumbbell"       "avg_yaw_dumbbell"
## [111] "stddev_yaw_dumbbell"      "var_yaw_dumbbell"
## [113] "gyros_dumbbell_x"         "gyros_dumbbell_y"
## [115] "gyros_dumbbell_z"         "accel_dumbbell_x"
## [117] "accel_dumbbell_y"         "accel_dumbbell_z"
## [119] "magnet_dumbbell_x"        "magnet_dumbbell_y"
## [121] "magnet_dumbbell_z"        "roll_forearm"
## [123] "pitch_forearm"            "yaw_forearm"
## [125] "kurtosis_roll_forearm"    "kurtosis_picth_forearm"
## [127] "kurtosis_yaw_forearm"     "skewness_roll_forearm"
## [129] "skewness_pitch_forearm"   "skewness_yaw_forearm"
## [131] "max_roll_forearm"         "max_picth_forearm"
## [133] "max_yaw_forearm"          "min_roll_forearm"
## [135] "min_pitch_forearm"        "min_yaw_forearm"
## [137] "amplitude_roll_forearm"   "amplitude_pitch_forearm"
## [139] "amplitude_yaw_forearm"    "total_accel_forearm"
## [141] "var_accel_forearm"        "avg_roll_forearm"
## [143] "stddev_roll_forearm"      "var_roll_forearm"
## [145] "avg_pitch_forearm"        "stddev_pitch_forearm"
## [147] "var_pitch_forearm"        "avg_yaw_forearm"
```

```
## [149] "stddev_yaw_forearm"       "var_yaw_forearm"
## [151] "gyros_forearm_x"          "gyros_forearm_y"
## [153] "gyros_forearm_z"          "accel_forearm_x"
## [155] "accel_forearm_y"          "accel_forearm_z"
## [157] "magnet_forearm_x"         "magnet_forearm_y"
## [159] "magnet_forearm_z"         "classe"
```

```
dim(training_data)
```

```
## [1] 19622   160
```

```
dim(test_data)
```

```
## [1]  20 160
```

As seen, the number of rows for the training dataset is 19622 and there are 160 columns. The testing dataset has 20 rows and 160 columns.

Tidy the dataset

We will proceed on to tidy the dataset as it is too huge. There seems to many columns with alot of NAs. We will also remove some unwanted columns.

```
training_data <- training_data[, colSums(is.na(training_data)) == 0]
test_data<-test_data[, colSums(is.na(test_data)) == 0]
training_data <-training_data[,-c(1:7)]
test_data <-test_data[,-c(1:7)]
dim(training_data)
```

```
## [1] 19622    53
```

```
dim(test_data)
```

```
## [1] 20 53
```

Now, the number of rows for the training dataset is 19622 and there are 56 columns. The testing dataset has 20 rows and 56 columns.This is easier to work with.

Split the dataset for Cross Validdation

```
set.seed(12222)
inTrain <-createDataPartition(training_data$classe,p=0.75,list=F)
training<-training_data[inTrain,]
testing<-training_data[-inTrain,]
dim(training)
```

```
## [1] 14718    53
```

Fitting of model

We will first use the random forest method to fit the model

```
Modelfit1 <- randomForest(classe ~. , data=training, method="class")
prediction1<-predict(Modelfit1,testing,type="class")
confusionMatrix(prediction1,testing$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    5    0    0    0
##          B    0  942    5    0    0
##          C    0    2  848    7    0
##          D    0    0    2  797    0
##          E    0    0    0    0  901
## 
## Overall Statistics
## 
##                Accuracy : 0.9957
##                  95% CI : (0.9935, 0.9973)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9946
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9926   0.9918   0.9913   1.0000
## Specificity            0.9986   0.9987   0.9978   0.9995   1.0000
## Pos Pred Value         0.9964   0.9947   0.9895   0.9975   1.0000
## Neg Pred Value         1.0000   0.9982   0.9983   0.9983   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1921   0.1729   0.1625   0.1837
## Detection Prevalence   0.2855   0.1931   0.1748   0.1629   0.1837
## Balanced Accuracy      0.9993   0.9957   0.9948   0.9954   1.0000
```

Next we will use Decision Trees

```
Modelfit2<- rpart(classe ~ ., data=training, method="class")
prediction2<-predict(Modelfit2,testing,type="class")
confusionMatrix(prediction2,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1283  208   19   95   32
##          B   28  525   72   29   57
##          C   36   96  681  128  110
##          D   21   80   63  498   52
##          E   27   40   20   54  650
##
## Overall Statistics
##
##                Accuracy : 0.7416
##                  95% CI : (0.7291, 0.7538)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6713
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9197   0.5532   0.7965   0.6194   0.7214
## Specificity            0.8991   0.9530   0.9086   0.9473   0.9648
## Pos Pred Value         0.7838   0.7384   0.6480   0.6975   0.8217
## Neg Pred Value         0.9657   0.8989   0.9548   0.9270   0.9390
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2616   0.1071   0.1389   0.1015   0.1325
## Detection Prevalence   0.3338   0.1450   0.2143   0.1456   0.1613
## Balanced Accuracy      0.9094   0.7531   0.8526   0.7834   0.8431
```

Next we will use Boosting (I had problems running the code for this)

From the results we can see that the Random forest gives more accurate results than Decision trees. As seen from results, the accuracy for the Random Forest is 0.9957 but the accuracy for Decision Trees is 0.7416. Hence we will choose the Random Forest. The expected out of sample error is 0.0043.

Lastly, we will predict based on the original Testing dataset. The method we will use is the Random Forest algorithm.

```
predictionsub<-predict(Modelfit1,test_data,type="class")
predictionsub
```

```
##  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
```

Submission

We will now use this code to produce the 20 files for submission

```
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(predictionsub)
```