PML Prediction Assignment Writeup

Pao Ying Heng

September 20, 2019

1. Introduction

The goal of this assignment is to predict the manner in which 6 participants performed barbell lifts. This report will describe how the machine learning algorithm is built, and which model was chosen based on cross validation and out-of-sample error. The algorithm is then used to predict 20 test cases, the results of which are submitted onto the Coursera Project Prediction Quiz for auto-grading.

2. Loading the Relevant Libraries

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.5.3
```

```
## Warning: package 'ggplot2' was built under R version 3.5.3
```

```
library(rpart)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.5.3
```

3. Getting & Cleaning the Data

```
#Set the download urls for the training and testing files
trainUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

#Reading the training and testing files
training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))
```

The first seven variables in both datasets are descriptive variables, so we will remove them as they are irrelevant to our ML algorithm.

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

The training dataset is then divided into two parts: `ptraining` for the training process (70% of data), and `ptesting` for validations (30% of data).

```r
set.seed(34334)
intrain <- createDataPartition(y=training$classe, p=0.7, list=FALSE)
ptraining <- training[intrain,]
ptesting <- training[-intrain,]
```

3.1 Cleaning the ptraining data

```r
#Identify near zero variables in the ptraining set
nearZeroVar(ptraining, saveMetrics=TRUE)
```

```
##                         freqRatio percentUnique zeroVar   nzv
## roll_belt                1.115689    8.28419597   FALSE FALSE
## pitch_belt               1.145038   12.26614253   FALSE FALSE
## yaw_belt                 1.039773   13.14697532   FALSE FALSE
## total_accel_belt         1.038609    0.21110868   FALSE FALSE
## kurtosis_roll_belt       1.000000    2.03101114   FALSE FALSE
## kurtosis_picth_belt      1.000000    1.70342870   FALSE FALSE
## kurtosis_yaw_belt        0.000000    0.00000000    TRUE  TRUE
## skewness_roll_belt       2.000000    2.03101114   FALSE FALSE
## skewness_roll_belt.1     1.000000    1.79078401   FALSE FALSE
## skewness_yaw_belt        0.000000    0.00000000    TRUE  TRUE
## max_roll_belt            1.125000    1.12833952   FALSE FALSE
## max_picth_belt           1.659091    0.13831259   FALSE FALSE
## max_yaw_belt             1.150000    0.41493776   FALSE FALSE
## min_roll_belt            1.125000    1.07738225   FALSE FALSE
## min_pitch_belt           2.179487    0.10919415   FALSE FALSE
## min_yaw_belt             1.150000    0.41493776   FALSE FALSE
## amplitude_roll_belt      1.080000    0.81531630   FALSE FALSE
## amplitude_pitch_belt     3.120000    0.08735532   FALSE FALSE
## amplitude_yaw_belt       0.000000    0.00727961    TRUE  TRUE
## var_total_accel_belt     1.413793    0.34942127   FALSE FALSE
## avg_roll_belt            1.090909    1.07738225   FALSE FALSE
## stddev_roll_belt         1.000000    0.40037854   FALSE FALSE
## var_roll_belt            1.722222    0.46589503   FALSE FALSE
## avg_pitch_belt           1.142857    1.24481328   FALSE FALSE
## stddev_pitch_belt        1.222222    0.26206595   FALSE FALSE
## var_pitch_belt           1.559322    0.36398049   FALSE FALSE
## avg_yaw_belt             1.142857    1.39768508   FALSE FALSE
## stddev_yaw_belt          1.631579    0.34214166   FALSE FALSE
## var_yaw_belt             1.548387    0.81531630   FALSE FALSE
## gyros_belt_x             1.009100    0.94634928   FALSE FALSE
## gyros_belt_y             1.113551    0.49501347   FALSE FALSE
## gyros_belt_z             1.071195    1.19385601   FALSE FALSE
## accel_belt_x             1.084746    1.17929679   FALSE FALSE
## accel_belt_y             1.062963    0.99002693   FALSE FALSE
## accel_belt_z             1.052970    2.08924802   FALSE FALSE
## magnet_belt_x            1.071730    2.20572177   FALSE FALSE
## magnet_belt_y            1.142191    2.08196841   FALSE FALSE
## magnet_belt_z            1.006042    3.18118949   FALSE FALSE
## roll_arm                47.400000   17.46378394   FALSE FALSE
## pitch_arm               79.000000   20.34650943   FALSE FALSE
```

```
## yaw_arm                33.857143   19.34192327   FALSE FALSE
## total_accel_arm         1.039746    0.48045425   FALSE FALSE
## var_accel_arm           5.500000    2.00917231   FALSE FALSE
## avg_roll_arm           55.000000    1.69614909   FALSE  TRUE
## stddev_roll_arm        55.000000    1.69614909   FALSE  TRUE
## var_roll_arm           55.000000    1.69614909   FALSE  TRUE
## avg_pitch_arm          55.000000    1.69614909   FALSE  TRUE
## stddev_pitch_arm       55.000000    1.69614909   FALSE  TRUE
## var_pitch_arm          55.000000    1.69614909   FALSE  TRUE
## avg_yaw_arm            55.000000    1.69614909   FALSE  TRUE
## stddev_yaw_arm         58.000000    1.67431026   FALSE  TRUE
## var_yaw_arm            58.000000    1.67431026   FALSE  TRUE
## gyros_arm_x             1.005333    4.62983184   FALSE FALSE
## gyros_arm_y             1.543909    2.67161680   FALSE FALSE
## gyros_arm_z             1.135359    1.75438596   FALSE FALSE
## accel_arm_x             1.094017    5.57618112   FALSE FALSE
## accel_arm_y             1.232394    3.84363398   FALSE FALSE
## accel_arm_z             1.170455    5.58346073   FALSE FALSE
## magnet_arm_x            1.016129    9.60180534   FALSE FALSE
## magnet_arm_y            1.135593    6.20222756   FALSE FALSE
## magnet_arm_z            1.064935    9.13591032   FALSE FALSE
## kurtosis_roll_arm       1.000000    1.68158987   FALSE FALSE
## kurtosis_picth_arm      1.000000    1.66703065   FALSE FALSE
## kurtosis_yaw_arm        1.000000    1.99461309   FALSE FALSE
## skewness_roll_arm       1.000000    1.68886948   FALSE FALSE
## skewness_pitch_arm      1.000000    1.66703065   FALSE FALSE
## skewness_yaw_arm        2.000000    2.00189270   FALSE FALSE
## max_roll_arm           18.333333    1.52143845   FALSE FALSE
## max_picth_arm          13.750000    1.41952391   FALSE FALSE
## max_yaw_arm             1.062500    0.33486205   FALSE FALSE
## min_roll_arm           27.500000    1.55783650   FALSE  TRUE
## min_pitch_arm          18.333333    1.55783650   FALSE FALSE
## min_yaw_arm             1.105263    0.26934556   FALSE FALSE
## amplitude_roll_arm     18.333333    1.59423455   FALSE FALSE
## amplitude_pitch_arm    14.500000    1.56511611   FALSE FALSE
## amplitude_yaw_arm       1.117647    0.35670088   FALSE FALSE
## roll_dumbbell           1.023529   86.76566936   FALSE FALSE
## pitch_dumbbell          2.448276   84.55994759   FALSE FALSE
## yaw_dumbbell            1.060976   86.24153745   FALSE FALSE
## kurtosis_roll_dumbbell  1.000000    2.03101114   FALSE FALSE
## kurtosis_picth_dumbbell 1.000000    2.05284997   FALSE FALSE
## kurtosis_yaw_dumbbell   0.000000    0.00000000    TRUE  TRUE
## skewness_roll_dumbbell  1.000000    2.05284997   FALSE FALSE
## skewness_pitch_dumbbell 1.000000    2.05284997   FALSE FALSE
## skewness_yaw_dumbbell   0.000000    0.00000000    TRUE  TRUE
## max_roll_dumbbell       1.333333    1.80534323   FALSE FALSE
## max_picth_dumbbell      1.000000    1.81262284   FALSE FALSE
## max_yaw_dumbbell        1.071429    0.45861542   FALSE FALSE
## min_roll_dumbbell       1.000000    1.77622479   FALSE FALSE
## min_pitch_dumbbell      1.000000    1.88541894   FALSE FALSE
## min_yaw_dumbbell        1.071429    0.45861542   FALSE FALSE
## amplitude_roll_dumbbell 14.000000   1.99461309   FALSE FALSE
## amplitude_pitch_dumbbell 7.000000   1.95093543   FALSE FALSE
## amplitude_yaw_dumbbell  0.000000    0.00727961    TRUE  TRUE
```

```
## total_accel_dumbbell      1.045596    0.30574361   FALSE FALSE
## var_accel_dumbbell        5.333333    1.95093543   FALSE FALSE
## avg_roll_dumbbell         1.000000    2.03101114   FALSE FALSE
## stddev_roll_dumbbell     14.000000    1.99461309   FALSE FALSE
## var_roll_dumbbell        14.000000    1.99461309   FALSE FALSE
## avg_pitch_dumbbell        1.000000    2.03101114   FALSE FALSE
## stddev_pitch_dumbbell    14.000000    1.99461309   FALSE FALSE
## var_pitch_dumbbell       14.000000    1.99461309   FALSE FALSE
## avg_yaw_dumbbell          1.000000    2.03101114   FALSE FALSE
## stddev_yaw_dumbbell      14.000000    1.99461309   FALSE FALSE
## var_yaw_dumbbell         14.000000    1.99461309   FALSE FALSE
## gyros_dumbbell_x          1.006993    1.71798792   FALSE FALSE
## gyros_dumbbell_y          1.228070    1.92909660   FALSE FALSE
## gyros_dumbbell_z          1.096618    1.45592196   FALSE FALSE
## accel_dumbbell_x          1.024390    2.94824197   FALSE FALSE
## accel_dumbbell_y          1.055556    3.31950207   FALSE FALSE
## accel_dumbbell_z          1.121387    2.89728471   FALSE FALSE
## magnet_dumbbell_x         1.056911    7.81830094   FALSE FALSE
## magnet_dumbbell_y         1.113821    6.01295771   FALSE FALSE
## magnet_dumbbell_z         1.046154    4.83366092   FALSE FALSE
## roll_forearm             11.252066   13.64926840   FALSE FALSE
## pitch_forearm            57.914894   18.94882434   FALSE FALSE
## yaw_forearm              15.465909   12.86307054   FALSE FALSE
## kurtosis_roll_forearm     1.000000    1.64519182   FALSE FALSE
## kurtosis_picth_forearm    1.000000    1.64519182   FALSE FALSE
## kurtosis_yaw_forearm      0.000000    0.00000000    TRUE  TRUE
## skewness_roll_forearm     1.000000    1.63791221   FALSE FALSE
## skewness_pitch_forearm    2.000000    1.63791221   FALSE FALSE
## skewness_yaw_forearm      0.000000    0.00000000    TRUE  TRUE
## max_roll_forearm         20.000000    1.42680352   FALSE  TRUE
## max_picth_forearm         3.157895    0.89539201   FALSE FALSE
## max_yaw_forearm           1.050000    0.27662517   FALSE FALSE
## min_roll_forearm         20.000000    1.45592196   FALSE  TRUE
## min_pitch_forearm         2.727273    0.96090850   FALSE FALSE
## min_yaw_forearm           1.050000    0.27662517   FALSE FALSE
## amplitude_roll_forearm   20.000000    1.53599767   FALSE  TRUE
## amplitude_pitch_forearm   4.066667    1.01186576   FALSE FALSE
## amplitude_yaw_forearm     0.000000    0.00727961    TRUE  TRUE
## total_accel_forearm       1.181055    0.49501347   FALSE FALSE
## var_accel_forearm         7.000000    2.04557036   FALSE FALSE
## avg_roll_forearm         30.000000    1.65247143   FALSE  TRUE
## stddev_roll_forearm      63.000000    1.63791221   FALSE  TRUE
## var_roll_forearm         63.000000    1.63791221   FALSE  TRUE
## avg_pitch_forearm        60.000000    1.65975104   FALSE  TRUE
## stddev_pitch_forearm     60.000000    1.65975104   FALSE  TRUE
## var_pitch_forearm        60.000000    1.65975104   FALSE  TRUE
## avg_yaw_forearm          60.000000    1.65975104   FALSE  TRUE
## stddev_yaw_forearm       61.000000    1.65247143   FALSE  TRUE
## var_yaw_forearm          61.000000    1.65247143   FALSE  TRUE
## gyros_forearm_x           1.086721    2.08924802   FALSE FALSE
## gyros_forearm_y           1.063241    5.25587828   FALSE FALSE
## gyros_forearm_z           1.133333    2.08924802   FALSE FALSE
## accel_forearm_x           1.095238    5.67081604   FALSE FALSE
## accel_forearm_y           1.265625    7.09033996   FALSE FALSE
```

```
## accel_forearm_z           1.067961    4.07658150   FALSE FALSE
## magnet_forearm_x          1.018182   10.62095072   FALSE FALSE
## magnet_forearm_y          1.407407   13.24888986   FALSE FALSE
## magnet_forearm_z          1.023256   11.76384946   FALSE FALSE
## classe                    1.469526    0.03639805   FALSE FALSE
```

```
#Now that we've identified the near zero variables, we're going to remove them from the ptraining set
nzv <- names(ptraining) %in% c("kurtosis_roll_belt", "kurtosis_picth_belt","kurtosis_yaw_belt", "skewnes

ptraining <- ptraining[!nzv]

#Now that we've removed near zero variables, we need to remove columns containing NA's
ptraining <- ptraining[ , colSums(is.na(ptraining)) == 0]
dim(ptraining)
```

```
## [1] 13737    53
```

We've cleaned the `ptraining` set and reduced the number of variables to 53.

3.2 Cleaning the `ptesting` data

```
#Identify near zero variables in the ptesting set and removing them
nearZeroVar(ptesting, saveMetrics=TRUE)
```

```
nzv2 <- names(ptesting) %in% c("kurtosis_roll_belt", "kurtosis_picth_belt","kurtosis_yaw_belt", "skewnes

ptesting <- ptesting[!nzv2]

#Remove columns containing NA's
ptesting <- ptesting[ , colSums(is.na(ptesting)) == 0]
dim(ptesting)
```

```
## [1] 5885   53
```

3.3 Cleaning the `testing` dataset

```
#Identify near zero variables in the testing dataset and removing them
nearZeroVar(testing, saveMetrics=TRUE)
```

```
nzv3 <- names(testing) %in% c("kurtosis_roll_belt", "kurtosis_picth_belt","kurtosis_yaw_belt", "skewness

testing <- testing[!nzv3]

#Remove columns containing NA's
testing <- testing[ , colSums(is.na(testing)) == 0]
dim(testing)
```

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

4. Prediction Models

4.1 Model 1: Regression Tree

```
modRT <- rpart(classe ~ ., data=ptraining, method="class")

#Cross Validation of the Regression Tree model
valRT <- predict(modRT, ptesting, type = "class")

confusionMatrix(valRT, ptesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1535  186   33   74   27
##          B   72  714   53   71   82
##          C   33   88  826  138  109
##          D   15   72   62  611   64
##          E   19   79   52   70  800
##
## Overall Statistics
##
##                Accuracy : 0.7623
##                  95% CI : (0.7512, 0.7731)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6981
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9170   0.6269   0.8051   0.6338   0.7394
## Specificity            0.9240   0.9414   0.9243   0.9567   0.9542
## Pos Pred Value         0.8275   0.7198   0.6918   0.7415   0.7843
## Neg Pred Value         0.9655   0.9131   0.9574   0.9303   0.9420
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2608   0.1213   0.1404   0.1038   0.1359
## Detection Prevalence   0.3152   0.1686   0.2029   0.1400   0.1733
## Balanced Accuracy      0.9205   0.7841   0.8647   0.7953   0.8468
```

```
##Calculating the out-of-sample error
1 - as.numeric(confusionMatrix(valRT, ptesting$classe)$overall[1])
```

```
## [1] 0.237723
```

The accuracy of the Regression Tree model in predicting is approximately 76%, while its out-of-sample error is approximately 25%, which aren't that great.

4.2 Model 2: Random Forest

```
modRF <- randomForest(classe ~. , data=ptraining)

#Cross Validation of the Random Forest model
valRF <- predict(modRF, ptesting)

confusionMatrix(valRF, ptesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    6    0    0    0
##          B    0 1133   10    0    0
##          C    0    0 1011    8    0
##          D    0    0    5  956    6
##          E    0    0    0    0 1076
##
## Overall Statistics
##
##                Accuracy : 0.9941
##                  95% CI : (0.9917, 0.9959)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9925
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9854   0.9917   0.9945
## Specificity            0.9986   0.9979   0.9984   0.9978   1.0000
## Pos Pred Value         0.9964   0.9913   0.9921   0.9886   1.0000
## Neg Pred Value         1.0000   0.9987   0.9969   0.9984   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1925   0.1718   0.1624   0.1828
## Detection Prevalence   0.2855   0.1942   0.1732   0.1643   0.1828
## Balanced Accuracy      0.9993   0.9963   0.9919   0.9947   0.9972
```

```
##Calculating the out-of-sample error
1 - as.numeric(confusionMatrix(valRF, ptesting$classe)$overall[1])
```

```
## [1] 0.005947324
```

The accuracy of the Random Forest model is 99.9%, while its out-of-sample error is 0.6%.

Our findings suggest that Model 2 (Random Forest) is superior to Model 1 (Regression Tree). Therefore, we will use this model to predict our test samples.

5. Applying the Selected Model to the `testing` dataset

Before we predict on test samples from the `testing` set, we need to check that its variables are identical to that of the trained algorithm (especially factor levels) or we may run into problems.

```
str(testing)
```

```
## 'data.frame':    20 obs. of  53 variables:
##  $ roll_belt           : num  123 1.02 0.87 125 1.35 -5.92 1.2 0.43 0.93 114 ...
##  $ pitch_belt          : num  27 4.87 1.82 -41.6 3.33 1.59 4.44 4.15 6.72 22.4 ...
##  $ yaw_belt            : num  -4.75 -88.9 -88.5 162 -88.6 -87.7 -87.3 -88.5 -93.7 -13.1 ...
##  $ total_accel_belt    : int  20 4 5 17 3 4 4 4 4 18 ...
##  $ gyros_belt_x        : num  -0.5 -0.06 0.05 0.11 0.03 0.1 -0.06 -0.18 0.1 0.14 ...
##  $ gyros_belt_y        : num  -0.02 -0.02 0.02 0.11 0.02 0.05 0 -0.02 0 0.11 ...
##  $ gyros_belt_z        : num  -0.46 -0.07 0.03 -0.16 0 -0.13 0 -0.03 -0.02 -0.16 ...
##  $ accel_belt_x        : int  -38 -13 1 46 -8 -11 -14 -10 -15 -25 ...
##  $ accel_belt_y        : int  69 11 -1 45 4 -16 2 -2 1 63 ...
##  $ accel_belt_z        : int  -179 39 49 -156 27 38 35 42 32 -158 ...
##  $ magnet_belt_x       : int  -13 43 29 169 33 31 50 39 -6 10 ...
##  $ magnet_belt_y       : int  581 636 631 608 566 638 622 635 600 601 ...
##  $ magnet_belt_z       : int  -382 -309 -312 -304 -418 -291 -315 -305 -302 -330 ...
##  $ roll_arm            : num  40.7 0 0 -109 76.1 0 0 0 -137 -82.4 ...
##  $ pitch_arm           : num  -27.8 0 0 55 2.76 0 0 0 11.2 -63.8 ...
##  $ yaw_arm             : num  178 0 0 -142 102 0 0 0 -167 -75.3 ...
##  $ total_accel_arm     : int  10 38 44 25 29 14 15 22 34 32 ...
##  $ gyros_arm_x         : num  -1.65 -1.17 2.1 0.22 -1.96 0.02 2.36 -3.71 0.03 0.26 ...
##  $ gyros_arm_y         : num  0.48 0.85 -1.36 -0.51 0.79 0.05 -1.01 1.85 -0.02 -0.5 ...
##  $ gyros_arm_z         : num  -0.18 -0.43 1.13 0.92 -0.54 -0.07 0.89 -0.69 -0.02 0.79 ...
##  $ accel_arm_x         : int  16 -290 -341 -238 -197 -26 99 -98 -287 -301 ...
##  $ accel_arm_y         : int  38 215 245 -57 200 130 79 175 111 -42 ...
##  $ accel_arm_z         : int  93 -90 -87 6 -30 -19 -67 -78 -122 -80 ...
##  $ magnet_arm_x        : int  -326 -325 -264 -173 -170 396 702 535 -367 -420 ...
##  $ magnet_arm_y        : int  385 447 474 257 275 176 15 215 335 294 ...
##  $ magnet_arm_z        : int  481 434 413 633 617 516 217 385 520 493 ...
##  $ roll_dumbbell       : num  -17.7 54.5 57.1 43.1 -101.4 ...
##  $ pitch_dumbbell      : num  25 -53.7 -51.4 -30 -53.4 ...
##  $ yaw_dumbbell        : num  126.2 -75.5 -75.2 -103.3 -14.2 ...
##  $ total_accel_dumbbell: int  9 31 29 18 4 29 29 29 3 2 ...
##  $ gyros_dumbbell_x    : num  0.64 0.34 0.39 0.1 0.29 -0.59 0.34 0.37 0.03 0.42 ...
##  $ gyros_dumbbell_y    : num  0.06 0.05 0.14 -0.02 -0.47 0.8 0.16 0.14 -0.21 0.51 ...
##  $ gyros_dumbbell_z    : num  -0.61 -0.71 -0.34 0.05 -0.46 1.1 -0.23 -0.39 -0.21 -0.03 ...
##  $ accel_dumbbell_x    : int  21 -153 -141 -51 -18 -138 -145 -140 0 -7 ...
##  $ accel_dumbbell_y    : int  -15 155 155 72 -30 166 150 159 25 -20 ...
##  $ accel_dumbbell_z    : int  81 -205 -196 -148 -5 -186 -190 -191 9 7 ...
##  $ magnet_dumbbell_x   : int  523 -502 -506 -576 -424 -543 -484 -515 -519 -531 ...
##  $ magnet_dumbbell_y   : int  -528 388 349 238 252 262 354 350 348 321 ...
##  $ magnet_dumbbell_z   : int  -56 -36 41 53 312 96 97 53 -32 -164 ...
##  $ roll_forearm        : num  141 109 131 0 -176 150 155 -161 15.5 13.2 ...
##  $ pitch_forearm       : num  49.3 -17.6 -32.6 0 -2.16 1.46 34.5 43.6 -63.5 19.4 ...
##  $ yaw_forearm         : num  156 106 93 0 -47.9 89.7 152 -89.5 -139 -105 ...
##  $ total_accel_forearm : int  33 39 34 43 24 43 32 47 36 24 ...
##  $ gyros_forearm_x     : num  0.74 1.12 0.18 1.38 -0.75 -0.88 -0.53 0.63 0.03 0.02 ...
##  $ gyros_forearm_y     : num  -3.34 -2.78 -0.79 0.69 3.1 4.26 1.8 -0.74 0.02 0.13 ...
##  $ gyros_forearm_z     : num  -0.59 -0.18 0.28 1.8 0.8 1.35 0.75 0.49 -0.02 -0.07 ...
```

```
##  $ accel_forearm_x     : int  -110 212 154 -92 131 230 -192 -151 195 -212 ...
##  $ accel_forearm_y     : int  267 297 271 406 -93 322 170 -331 204 98 ...
##  $ accel_forearm_z     : int  -149 -118 -129 -39 172 -144 -175 -282 -217 -7 ...
##  $ magnet_forearm_x    : int  -714 -237 -51 -233 375 -300 -678 -109 0 -403 ...
##  $ magnet_forearm_y    : int  419 791 698 783 -787 800 284 -619 652 723 ...
##  $ magnet_forearm_z    : int  617 873 783 521 91 884 585 -32 469 512 ...
##  $ problem_id          : int  1 2 3 4 5 6 7 8 9 10 ...
```

As we can see, the `testing` set has the variable `problem_id` (class: int) as opposed to the variable `classe` in the `training_set` (class: factor with 5 levels). We'll need to convert the `problem_id` class to a factor, and ensure that the levels are identical to that of the `training_set` (5 levels: A,B,C,D,E)

```
testing$problem_id <- as.factor(testing$problem_id)
testing$problem_id = factor(c(1:5))
levels(testing$problem_id) <- c("A", "B", "C", "D", "E")
```

Once this is done, we can then predict the test samples:

```
prediction <- predict(modRF, testing)
print(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
```

Conclusion

The selected model predicted all 20 test cases correctly (100% accuracy).