Preciction Assignment: Activity Recognition

DDThomas

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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Steps for this predictive analysis:

1. Download and Split Data
2. Exploratory Data Analysis
3. Prediction Model Comparison
4. Test Set Performance

Getting the Data Ready

Both training and testing datasets are downloaded. The training dataset is then broken down into its own training and testing sets.

```r
library(caret)
trainURL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testURL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training <- read.csv(url(trainURL))
testing <- read.csv(url(testURL))
label <- createDataPartition(training$classe, p = 0.7, list = FALSE)
train <- training[label, ]
test <- training[-label, ]
str(train)
```

```
## 'data.frame':    13737 obs. of  160 variables:
##  $ X                       : int  1 4 5 6 8 9 12 14 15 17 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084232 1323084232 1323084232 1323084232 1323084232 
##  $ raw_timestamp_part_2    : int  788290 120339 196328 304277 440390 484323 528316 576390 604281 692
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  11 12 12 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.48 1.48 1.45 1.42 1.43 1.43 1.42 1.45 1.51 ...
##  $ pitch_belt              : num  8.07 8.05 8.07 8.06 8.13 8.16 8.18 8.21 8.2 8.12 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : Factor w/ 397 levels "","-0.016850",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_belt     : Factor w/ 317 levels "","-0.021887",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt      : Factor w/ 395 levels "","-0.003095",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt.1    : Factor w/ 338 levels "","-0.005928",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ skewness_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : Factor w/ 4 levels "","#DIV/0!","0.00",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ var_total_accel_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_belt_x            : num  0 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0 0 ...
##  $ gyros_belt_y            : num  0 0 0.02 0 0 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 0 -0.02 ...
##  $ accel_belt_x            : int  -21 -22 -21 -21 -22 -20 -22 -22 -21 -21 ...
##  $ accel_belt_y            : int  4 3 2 4 4 2 2 4 2 4 ...
##  $ accel_belt_z            : int  22 21 24 21 21 24 23 21 22 22 ...
##  $ magnet_belt_x           : int  -3 -6 -6 0 -2 1 -2 -8 -1 -6 ...
##  $ magnet_belt_y           : int  599 604 600 603 603 602 602 598 597 598 ...
##  $ magnet_belt_z           : int  -313 -310 -302 -312 -313 -312 -319 -310 -310 -317 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -129 -129 ...
##  $ pitch_arm               : num  22.5 22.1 22.1 22 21.8 21.7 21.5 21.4 21.4 21.3 ...
##  $ yaw_arm                 : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm         : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_arm_x             : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  0 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 0 0 0 ...
##  $ gyros_arm_z             : num  -0.02 0.02 0 0 0 -0.02 0 -0.03 -0.03 -0.02 ...
##  $ accel_arm_x             : int  -288 -289 -289 -289 -289 -288 -288 -288 -289 -289 ...
##  $ accel_arm_y             : int  109 111 111 111 111 109 111 111 111 110 ...
##  $ accel_arm_z             : int  -123 -123 -123 -122 -124 -122 -123 -124 -124 -122 ...
##  $ magnet_arm_x            : int  -368 -372 -374 -369 -372 -369 -363 -371 -374 -371 ...
##  $ magnet_arm_y            : int  337 344 337 342 338 341 343 331 342 337 ...
##  $ magnet_arm_z            : int  516 512 506 513 510 518 520 523 510 512 ...
##  $ kurtosis_roll_arm       : Factor w/ 330 levels "","-0.02438",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_arm      : Factor w/ 328 levels "","-0.00484",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ kurtosis_yaw_arm        : Factor w/ 395 levels "","-0.01548",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_arm       : Factor w/ 331 levels "","-0.00051",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_arm      : Factor w/ 328 levels "","-0.00184",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_arm        : Factor w/ 395 levels "","-0.00311",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ roll_dumbbell           : num  13.1 13.4 13.4 13.4 12.8 ...
##  $ pitch_dumbbell          : num  -70.5 -70.4 -70.4 -70.8 -70.3 ...
##  $ yaw_dumbbell            : num  -84.9 -84.9 -84.9 -84.5 -85.1 ...
##  $ kurtosis_roll_dumbbell  : Factor w/ 398 levels "","-0.0035","-0.0073",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_dumbbell : Factor w/ 401 levels "","-0.0163","-0.0233",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_dumbbell  : Factor w/ 401 levels "","-0.0082","-0.0096",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_dumbbell : Factor w/ 402 levels "","-0.0053","-0.0084",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

Data Cleaning

The structure above shows that we probably have too many variables. Some variables have an excessive number of NAs and need to be excluded. Other variables lack variance and should be removed.

```r
NZV <- nearZeroVar(train)
train <- train[ ,-NZV]
test <- test[ ,-NZV]
label <- apply(train, 2, function(x) mean(is.na(x))) > 0.95
train <- train[, -which(label, label == FALSE)]
test <- test[, -which(label, label == FALSE)]
train <- train[ , -(1:5)]
test <- test[ , -(1:5)]
```

We started with 160 variables and have reduced those to only 54.

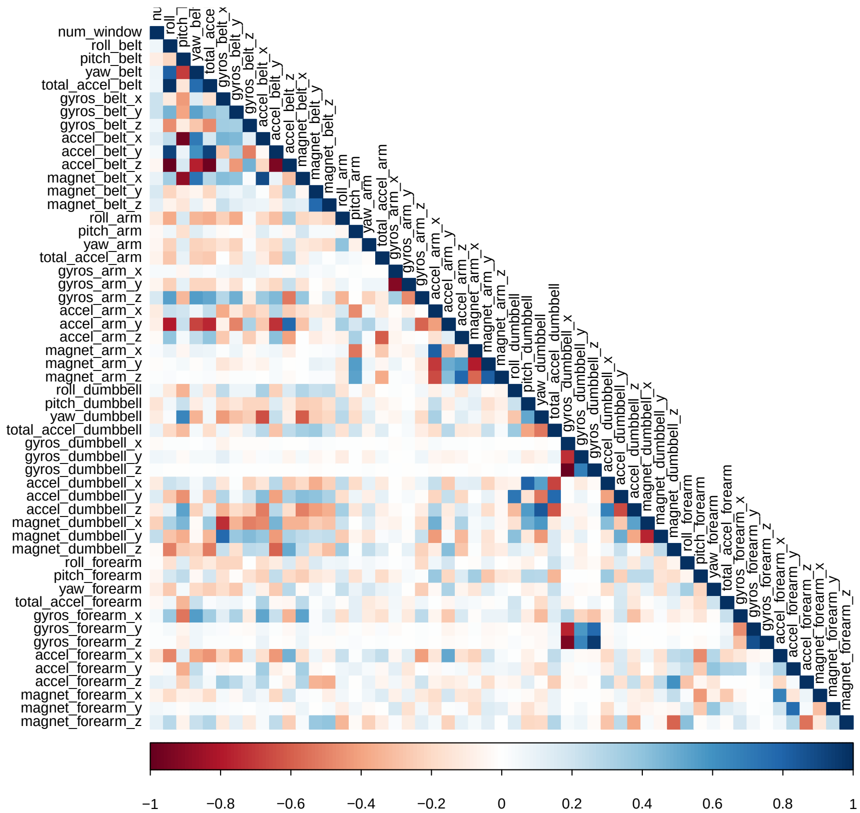
Exploratory Analysis

The correlation plot below will begin to show us relationships in the data.

```r
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
corrMat <- cor(train[,-54])
corrplot(corrMat, method = "color", type = "lower", tl.cex = 0.8, tl.col = rgb(0,0,0))
```



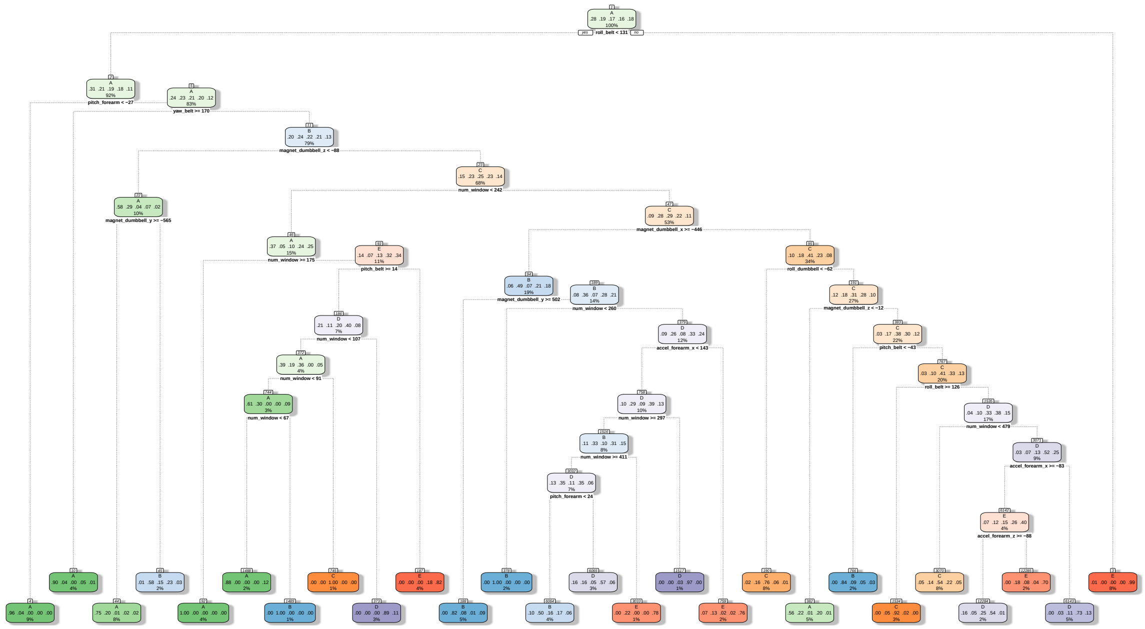
Darker color indicates a stronger relationship; red=negative, blue=positive.

Prediction Models

Of the available methods we will use Decision Tree, Random Forest and Generalized Boosted Model.

Decision Tree

```r
library(rpart)
library(rpart.plot)
library(rattle)
set.seed(13908)
modelDT <- rpart(classe ~ ., data = train, method = "class")
fancyRpartPlot(modelDT)
```



Rattle 2019−May−11 17:42:34 David Thomas

```r
predictDT <- predict(modelDT, test, type = "class")
confMatDT <- confusionMatrix(predictDT, test$classe)
confMatDT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1542  200    8   90   32
##          B   29  724  103   79   39
##          C   33  146  824  128   24
##          D   58   26   70  612   71
##          E   12   43   21   55  916
##
## Overall Statistics
##
##                Accuracy : 0.7847
##                  95% CI : (0.774, 0.7952)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7265
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9211   0.6356   0.8031   0.6349   0.8466
## Specificity            0.9216   0.9473   0.9319   0.9543   0.9727
## Pos Pred Value         0.8237   0.7433   0.7134   0.7312   0.8749
## Neg Pred Value         0.9671   0.9155   0.9573   0.9303   0.9657
```

```
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2620   0.1230   0.1400   0.1040   0.1556
## Detection Prevalence   0.3181   0.1655   0.1963   0.1422   0.1779
## Balanced Accuracy      0.9214   0.7915   0.8675   0.7946   0.9097
```

Random Forest

```r
library(caret)
set.seed(13908)
control <- trainControl(method = "cv", number = 3, verboseIter=FALSE)
modelRF <- train(classe ~ ., data = train, method = "rf", trControl = control)
modelRF$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.23%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    0    0    0    1 0.0002560164
## B   10 2646    2    0    0 0.0045146727
## C    0    5 2391    0    0 0.0020868114
## D    0    0    5 2246    1 0.0026642984
## E    0    1    0    6 2518 0.0027722772
```

```r
predictRF <- predict(modelRF, test)
confMatRF <- confusionMatrix(predictRF, test$classe)
confMatRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    5    0    0    0
##          B    0 1133    2    0    0
##          C    0    1 1024    2    0
##          D    0    0    0  962    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9983
##                  95% CI : (0.9969, 0.9992)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9979
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9981   0.9979   1.0000
## Specificity            0.9988   0.9996   0.9994   1.0000   1.0000
## Pos Pred Value         0.9970   0.9982   0.9971   1.0000   1.0000
## Neg Pred Value         1.0000   0.9987   0.9996   0.9996   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1925   0.1740   0.1635   0.1839
## Detection Prevalence   0.2853   0.1929   0.1745   0.1635   0.1839
## Balanced Accuracy      0.9994   0.9972   0.9987   0.9990   1.0000
```

Generalized Boosted Model

```
library(caret)
library(gbm)
set.seed(13908)
control <- trainControl(method = "repeatedcv", number = 5, repeats = 1, verboseIter = FALSE)
modelGBM <- train(classe ~ ., data = train, trControl = control, method = "gbm", verbose = FALSE)
modelGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

```
predictGBM <- predict(modelGBM, test)
confMatGBM <- confusionMatrix(predictGBM, test$classe)
confMatGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674   12    0    0    0
##          B    0 1115   13    6    1
##          C    0   11 1003   10    3
##          D    0    1   10  946    4
##          E    0    0    0    2 1074
##
## Overall Statistics
##
##                Accuracy : 0.9876
##                  95% CI : (0.9844, 0.9903)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9843
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9789   0.9776   0.9813   0.9926
## Specificity            0.9972   0.9958   0.9951   0.9970   0.9996
## Pos Pred Value         0.9929   0.9824   0.9766   0.9844   0.9981
```

```
## Neg Pred Value         1.0000   0.9949   0.9953   0.9963   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1895   0.1704   0.1607   0.1825
## Detection Prevalence   0.2865   0.1929   0.1745   0.1633   0.1828
## Balanced Accuracy      0.9986   0.9874   0.9863   0.9891   0.9961
```

Of the three methods used, the Random Forest Model gave the highest accuracy, 99.75%.

Predicting Test Set Output

```
predictRF <- predict(modelRF, testing)
predictRF
```

```
##  [1] B A A A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```