Practical Machine Learning Course Project

SP

March 5, 2019

Disclaimer: I understand that we were expected to limit the length of the assignment to 2000 words. I am not sure if I have been able to stay within limit but I found it necessary to explain some of the transformations required in order to develop the models and make the predictions.

Introduction

With a variety of fitness tracking devices available on the market, it is now possible to track a large number of variables for activity. One thing that has been studied is the quantity,i.e. how much a particular activity is done. However there isn't much done to track the quality of an activity. This project aims to explore this aspect of fitness tracking by looking at the data available here http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har)

Exploratory Data Analysis

This analysis looks at a weight liting dataset where six male participants with little weight lifting experience performed bicep curl for one set with 10 repititions. The participants were guided by an experienced instructor and had to perform bicep curls in 5 different fashions or classes. These classes are:
A - performing the exercise according to specification
B - throwing the elbows to the front
C - lifting the dumbell only halfway
D - lowering the dumbell only halfway
E - throwing the hips to the front

Their activities were tracked using four sensors mounted on the arm, forearm, belt and the dumbell. These activities manifest as variables in the dataset. Our goal is to predict the class from these variables. The data has already been split into a training and a test set.

We now begin by loading the data and looking at its attributes.

```
##setwd ("set the working directory accordingly")

train <- read.csv("pml-training.csv")
test <- read.csv("pml-testing.csv")

str(train)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2
2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 13230
84232 1323084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296
440390 484323 484434 ...
##  $ 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 ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.4
5 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.1
7 ...
##  $ 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 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y            : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02
-0.02 0 ...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y            : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z            : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x           : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y           : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z           : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -30
8 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -12
8 ...
##  $ pitch_arm               : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.
6 ...
##  $ yaw_arm                 : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -16
1 ...
##  $ 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 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.0
3 -0.03 ...
##  $ gyros_arm_z             : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x             : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -28
8 ...
##  $ accel_arm_y             : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z             : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -12
4 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -37
6 ...
##  $ magnet_arm_y            : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z            : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ 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 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell          : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell            : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ 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]
```

We see that there are a bunch of columns that have NAs in them. We could also argue that there are rows that have NAs in them. However if we remove the NAs based on the rows, then we only have 406 observations to work with but if we remove columns that are mostly NAs, then we still get 19622 observations and we have meaningful predictors that we can use in our model.

Now on closer observation we also observe that some of the factor variables in the training dataset contain over 50 levels. Since the tree and randomforest methods that we intend to use here, do not accept such a high number of levels for the factor variables, we need to convert these variables to numeric. We will get to this later. First let us treat the NAs.

```
train_1 <- na.omit(train)

train_a <- train[, colSums(is.na(train)) == 0]

ncol_train_a <- ncol(train_a)
```

We now have **93 variables** in the train set after removing columns that mostly contain NAs.

We must however also consider the variables that can be used in the test set so that when we train our model, it has those variables available in the test set to later on make those predictions. We start this by again removing columns from the test set that mostly contain NAs.

```
options(warn = -1)

test_a <- test[, colSums(is.na(test)) == 0]

ncol_test_a <- ncol(test_a)
```

We now notice that we have **60 variables** in the test set after removing columns that contain NAs.

We now proceed to convert factor variables with over 50 levels to numeric in order to use the training algorithms later on.

```
for (i in 1: ncol(train_a)){
        indx <- sapply(train_a, is.factor)
        lev <- sapply(train_a[,sapply(train_a, is.factor)], nlevels)
        if (nlevels(train_a[, i]) >40){
                train_a[i] <- lapply(train_a[i], function(x) as.numeric(as.character
(x)))
        }
}
```

We also have to reconcile the training and test sets so that we use the same 60 variables to build our training model.

```
train_c <- train_a[, names(train_a) %in% names(test_a)]

train_c$classe <- train_a$classe
```

The dataframe train_c does not have the “classe” variable that we require to make the predictions since this variable does not appear in the test_a data frame. So we need to add this column back to train_c which is what the last line above does.

We now have a clean training data set that we can use to build our model.

Training the model using classification trees

Here we will explore the training dataset and try to fit a model that can predict classe. We start by fitting a classification tree.

```
library(tree)

tree.train <- tree(classe~.-X -cvtd_timestamp, data = train_c)

plot(tree.train)
text(tree.train, pretty = 0)
```

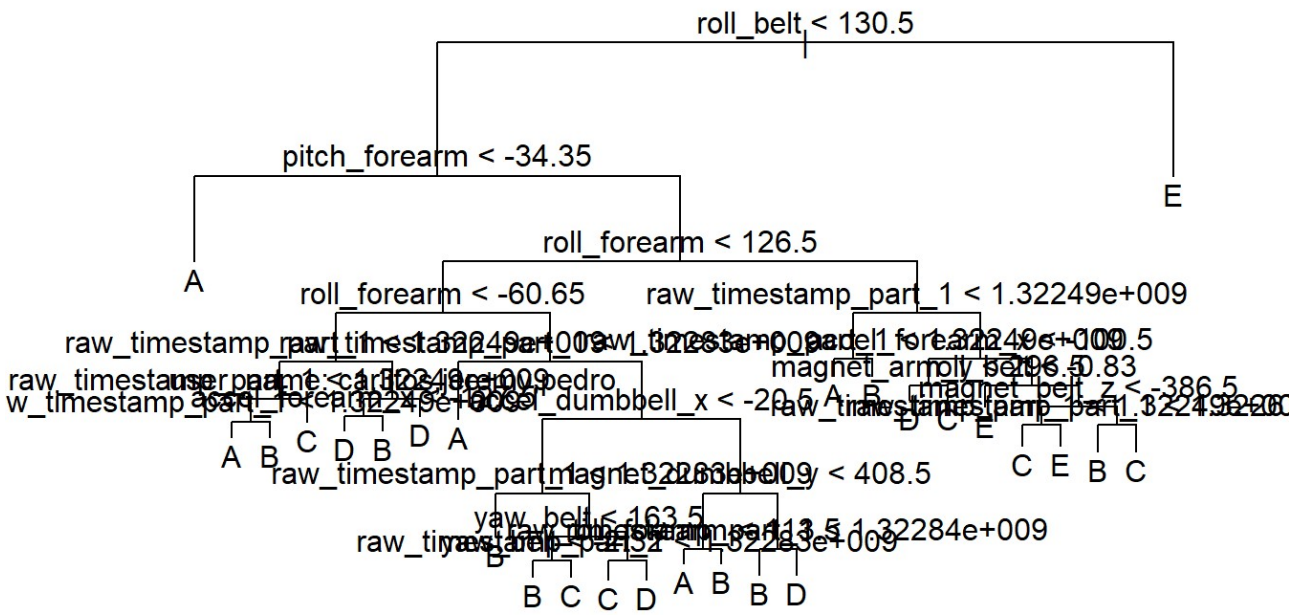

Figure 1: Classification tree fit to the training data. We can see the first three most important predictors for classe.

```
summary(tree.train)
```

```
##
## Classification tree:
## tree(formula = classe ~ . - X - cvtd_timestamp, data = train_c)
## Variables actually used in tree construction:
##  [1] "roll_belt"            "pitch_forearm"        "roll_forearm"
##  [4] "raw_timestamp_part_1" "user_name"            "accel_forearm_x"
##  [7] "accel_dumbbell_x"     "yaw_belt"             "magnet_dumbbell_y"
## [10] "magnet_arm_y"         "magnet_belt_z"
## Number of terminal nodes:  27
## Residual mean deviance:  0.9551 = 18720 / 19600
## Misclassification error rate: 0.1715 = 3366 / 19622
```

We see from the summary above that the misclassification error rate for the classification tree model is 17% and the number of terminal nodes used are 27.

We can also use cross-validation to train the model as shown below

```
cv.train <- cv.tree(tree.train, FUN = prune.misclass)

cv.train
```

```
## $size
##  [1] 27 25 24 23 22 21 20 19 18 17 16 15 14 12 11  9  8  4  2  1
##
## $dev
##  [1]  8086  8186  8273  8331  8571  8571  8865  8831  8827  8827  8719
## [12]  8854  8971  8971  8971  9534  9534 12440 12440 14042
##
## $k
##  [1]    -Inf   95.00  111.00  148.00  201.00  206.00  243.00  255.00
##  [9]  257.00  259.00  277.00  286.00  299.00  310.00  321.00  404.50
## [17]  546.00  667.75  680.00 1617.00
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

We can see from above that the lowest error rate (noted by the dev variable) occurs for the case where we have 27 terminal nodes. This would most likely give us a model very similar to what we had before doing cross-validation since we had 27 terminal nodes in that case as well.

Let us now proceed to build a prediction for class.

```
tree.pred <- predict(tree.train, test_a, type = "class")

tree.pred
```

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

Using these predictions on the quiz, we note that we only get a 50% accuracy. Let us now try to fit a random forest model.

Training the model using random forests

We now move on to develop a random forest model. But we first need to make sure that we have the same number of levels for the factor variables in both the training and test datasets. This is a necessary step since if we try to predict without setting the same levels on all factor variables, the random forest algorithm gives us an error.

```
levels(test_a$new_window) <- levels(train_c$new_window)
levels(test_a$cvtd_timestamp) <- levels(train_c$cvtd_timestamp)

library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
rf.train <- randomForest(classe~.-X -cvtd_timestamp, data = train_c,importance = TRUE)

summary(rf.train)
```

```
##                 Length Class  Mode
## call               4  -none- call
## type               1  -none- character
## predicted      19622  factor numeric
## err.rate        3000  -none- numeric
## confusion         30  -none- numeric
## votes          98110  matrix numeric
## oob.times      19622  -none- numeric
## classes            5  -none- character
## importance       399  -none- numeric
## importanceSD     342  -none- numeric
## localImportance    0  -none- NULL
## proximity          0  -none- NULL
## ntree              1  -none- numeric
## mtry               1  -none- numeric
## forest            14  -none- list
## y              19622  factor numeric
## test               0  -none- NULL
## inbag              0  -none- NULL
## terms              3  terms  call
```

We can now look at the importance of the variables in the random forest model.

```
varImpPlot(rf.train)
```

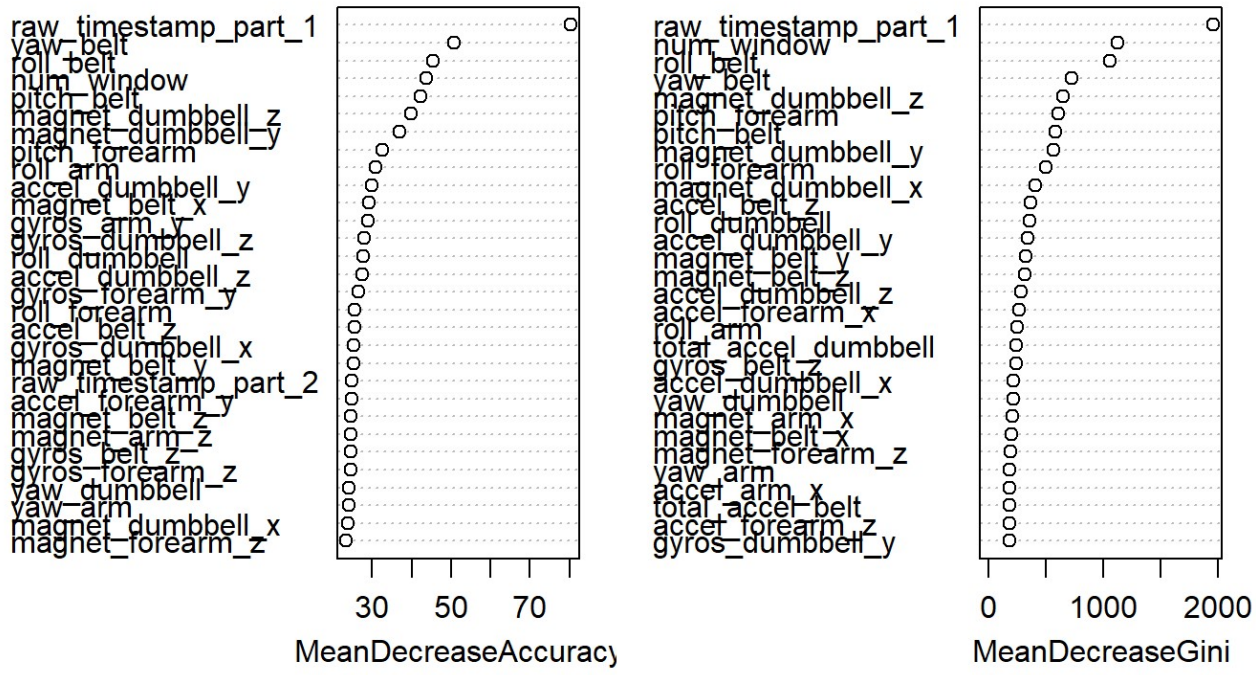

Figure 2: Plot showing the variable importance for the training data

The first plot shows the mean decrease in accuracy of predictions on the out of bag samples when a given variable is excluded. We can see that the variable raw_timestamp_part_1 has a pretty significant influence on this measure. The second plot shows the decrease in node impurity due to a split over that variable averaged across all the trees.

We can now finally fit the model to the test data to see how the predictions work out.

```
rf.pred <- predict(rf.train, test_a)

rf.pred
```

```
##  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
```

Using these predictions on the quiz we see that we are able to get a 100% accuracy indicating that the model has worked to a satisfactory level.

Note: I'd like to thank the authors of the paper below for allowing us to use this dataset.
Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.