Quantified Self Predictive Modeling - Machine Learning

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.

Objective

The objective of this project is to use the 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.

```
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
library(RCurl)
library(kernlab)
```

Data Loading

Loading the url for the training and testing data.

```
trainURL = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testURL = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

Reading the training and testing data into the respective data frames

```
trainingFile <- getURL(trainURL)
training <- read.csv(textConnection(trainingFile),na.strings=c("NA",""), header=T)
testingFile <- getURL(testURL)
testing <- read.csv(textConnection(testingFile),na.strings=c("NA",""), header=T)
```

Data Cleaning

The data contains many NAs. Removing the columns that contain NAs in both the training and testing set.

```
training = training[,colSums(is.na(training)) == 0]
testing = testing[,colSums(is.na(testing)) == 0]
```

Removing the variables with near zero variance.

```
nearzerovar = nearZeroVar(training, saveMetrics=TRUE)
training = training[,nearzerovar$nzv==FALSE]
nearzerovar = nearZeroVar(testing, saveMetrics=TRUE)
testing = testing[,nearzerovar$nzv==FALSE]
```

Viewing the training set after removing the NAs and variables with near zero variance.

```
str(training)
```

```
## 'data.frame':    19622 obs. of  59 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 13230842
32 1323084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2: int  788290 808298 820366 120339 196328 304277 368296 440
390 484323 484434 ...
##  $ cvtd_timestamp      : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9
9 9 9 ...
##  $ 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 1.45 ..
.
##  $ pitch_belt          : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ..
.
##  $ 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 ...
##  $ 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 -0.0
2 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 -308 ..
.
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ..
.
##  $ 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 -161 ..
.
##  $ total_accel_arm     : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ 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 -0.03 -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 -288 ..
.
##  $ 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 -124 ..
.
##  $ magnet_arm_x        : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ..
```

```
.
##  $ 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 ...
##  $ 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 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.0
2 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ..
.
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ..
.
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ..
.
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.
8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ..
.
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ..
.
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ..
.
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1
1 ...
```

The idea of the project is to asses the quality of the exercise performed. The columns that specify username , time stamp and window data can be removed.

```
training = training[,-c(1:6)]
testing = testing[,-c(1:6)]
```

Data Partitioning for fitting Machine Learning Algorithm.

```
inTrain = createDataPartition(y=training$classe,p=0.7,list = FALSE)
dataTrain = training[inTrain,]
dataTest = training[-inTrain,]
```

Training the Machine Learning algorithm with classification tree model. Using K fold cross validation. setting the value of k to 5 to minimise the number of computations performed on the training set.

```
train_control<- trainControl(method="cv", number=5)
modelCART<- train(classe ~., data=dataTrain, trControl=train_control, method="rpart")
```

Predicting the out of sample error using the testing data and viewing the confusion Matrix.

```
modelPredict = predict(modelCART,newdata = dataTest)
confMatCART = confusionMatrix(modelPredict,dataTest$classe)
paste("Out of Sample Accuracy - CART Model",round(confMatCART$overall['Accuracy'],4))
```

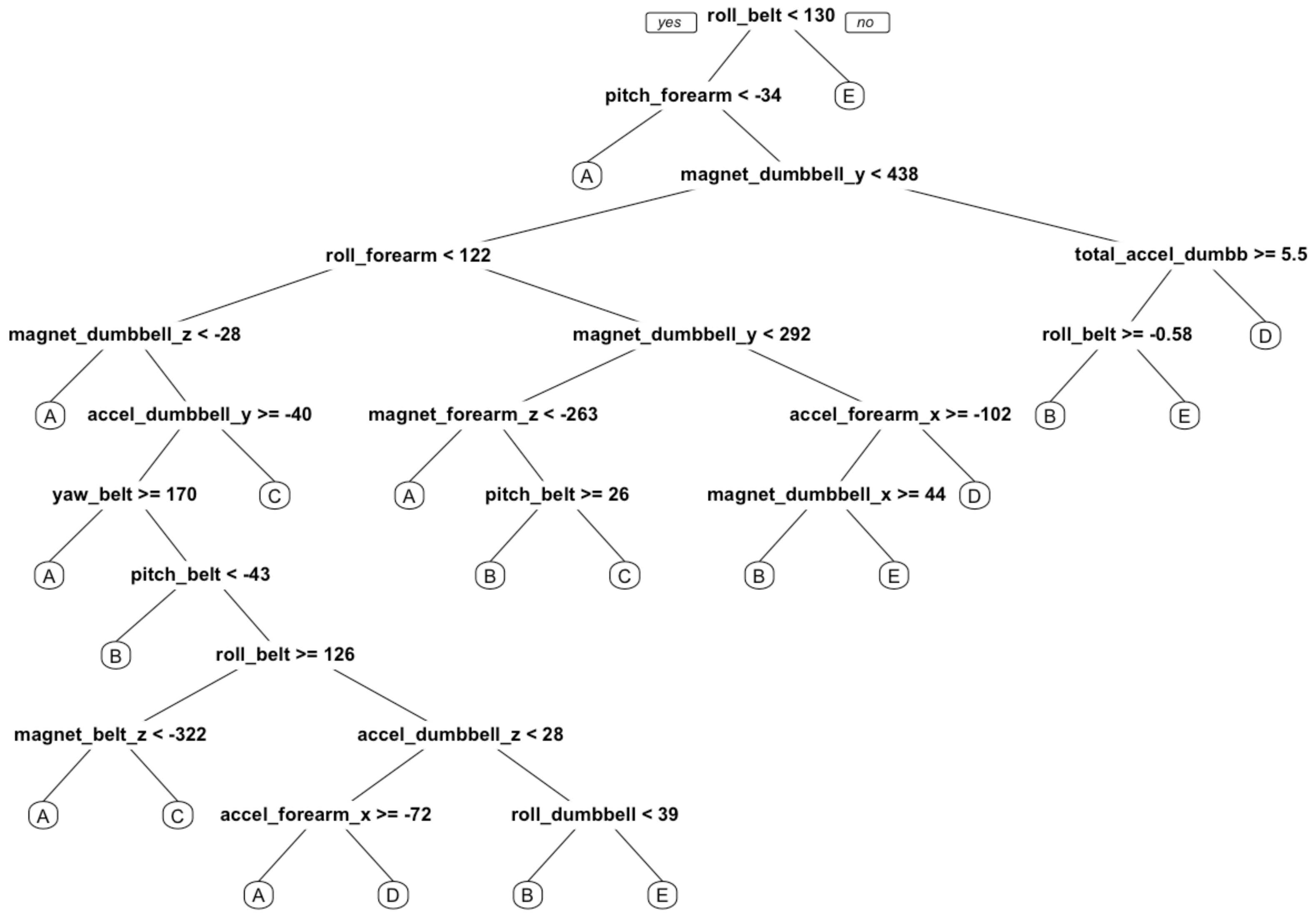
```
## [1] "Out of Sample Accuracy - CART Model 0.4879"
```

CART Model Results

This is not a good prediction approach. Hence need to go in for a better algorithm.

Classification Tree

```
modelClass = rpart(classe ~., data=dataTrain,method="class")
prp(modelClass)
```



```
predClass = predict(modelClass,newdata= dataTest,type = "class")
confMatClass = confusionMatrix(predClass,dataTest$classe)
paste("Out of Sample Accuracy - Classification Tree",round(confMatClass$overall['Accuracy'],4))
```

```
## [1] "Out of Sample Accuracy - Classification Tree 0.7108"
```

Random Forest

Fitting Random Forest Model and Calculating the out of sample accuracy

```
modelRF = randomForest(classe ~., data=dataTrain)
modelPredictRF = predict(modelRF,dataTest)
confMatRF = confusionMatrix(modelPredictRF,dataTest$classe)
paste("Out of Sample Accuracy - Random Forest",round(confMatRF$overall['Accuracy'],4)
)
```

```
## [1] "Out of Sample Accuracy - Random Forest 0.9952"
```

Predicting the final results on the test set

```
predict(modelRF,newdata= testing)
```

```
##  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
```

In the models fitted in the above analysis, Random Forest has performed better than the rest.