

Prediction Assignment

Prabhjot Singh

September 30, 2016

Introduction

Bakground

The goal of this project is to create a machine-learning algorithm that can correctly identify the quality of barbell lifts by using data from belt, forearm, arm, and dumbbell monitors. There are five classifications of this exercise, one method is the correct form of the exercise while the other four are common mistakes: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E)

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).

Input Data

The training data for this project are available here: <https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv> (<https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv>)

The test data are available here: <https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv> (<https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv>)

Load and Clean the Data

```
library(caret)
library(gbm)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(randomForest)
library(knitr)
```

```
set.seed(7575)

trainingDataUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testingDataUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

training <- read.csv(url(trainingDataUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testingDataUrl), na.strings=c("NA","#DIV/0!",""))

#Partioning the training set into two (60-40 ratio)

inTrain <- createDataPartition(training$classe, p=0.6, list=FALSE)
trainingData <- training[inTrain, ]
testingData <- training[-inTrain, ]
```

```
dim(trainingData);
dim(testingData)
```

```
#Clean the data
```

```
nzv <- nearZeroVar(trainingData, saveMetrics=TRUE)
trainingData <- trainingData[,nzv$nzv==FALSE]
```

```
nzv<- nearZeroVar(testingData,saveMetrics=TRUE)
testingData <- testingData[,nzv$nzv==FALSE]
```

```
#Remove the first column of the trainingData data set
trainingData <- trainingData[c(-1)]
```

```
#Clean variables with more than 60% NA
```

```
cleanTrainingData <- trainingData
for(i in 1:length(trainingData)) {
  if( sum( is.na( trainingData[, i] ) ) /nrow(trainingData) >= .7) {
    for(j in 1:length(cleanTrainingData)) {
      if( length( grep(names(trainingData[i]), names(cleanTrainingData)[j]) ) == 1) {
        cleanTrainingData <- cleanTrainingData[ , -j]
      }
    }
  }
}
```

```
# Set back to the original variable name
```

```
trainingData <- cleanTrainingData
rm(cleanTrainingData)
```

```
#Transform the testingData and testing data sets
```

```
cleanData1 <- colnames(trainingData)
cleanData2 <- colnames(trainingData[, -58]) # remove the classe column
testingData <- testingData[cleanData1] # allow only variables in testingData that are al
so in trainingData
testing <- testing[cleanData2] # allow only variables in testing that are also in tr
ainingData
```

```
dim(testingData)
dim(testing)
```

```
#Coerce the data into the same type
```

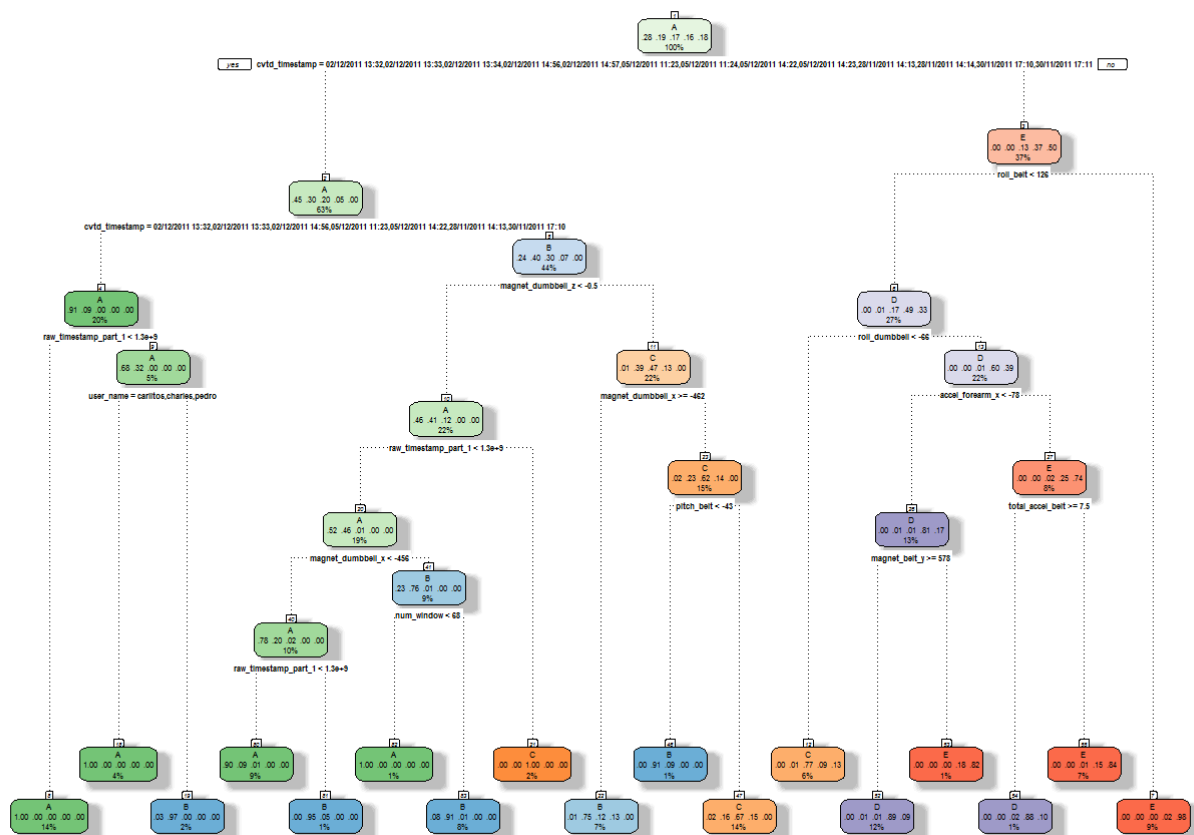
```
for (i in 1:length(testing) ) {
  for(j in 1:length(trainingData)) {
    if( length( grep(names(trainingData[i]), names(testing)[j]) ) == 1) {
      class(testing[j]) <- class(trainingData[i])
    }
  }
}
```

```
# To get the same class between testing and trainingData
```

```
testing <- rbind(trainingData[2, -58] , testing)
testing <- testing[-1,]
```

Prediction using Decision Tree

```
set.seed(7575)
modFitA1 <- rpart(classe ~ ., data=trainingData, method="class")
fancyRpartPlot(modFitA1)
```



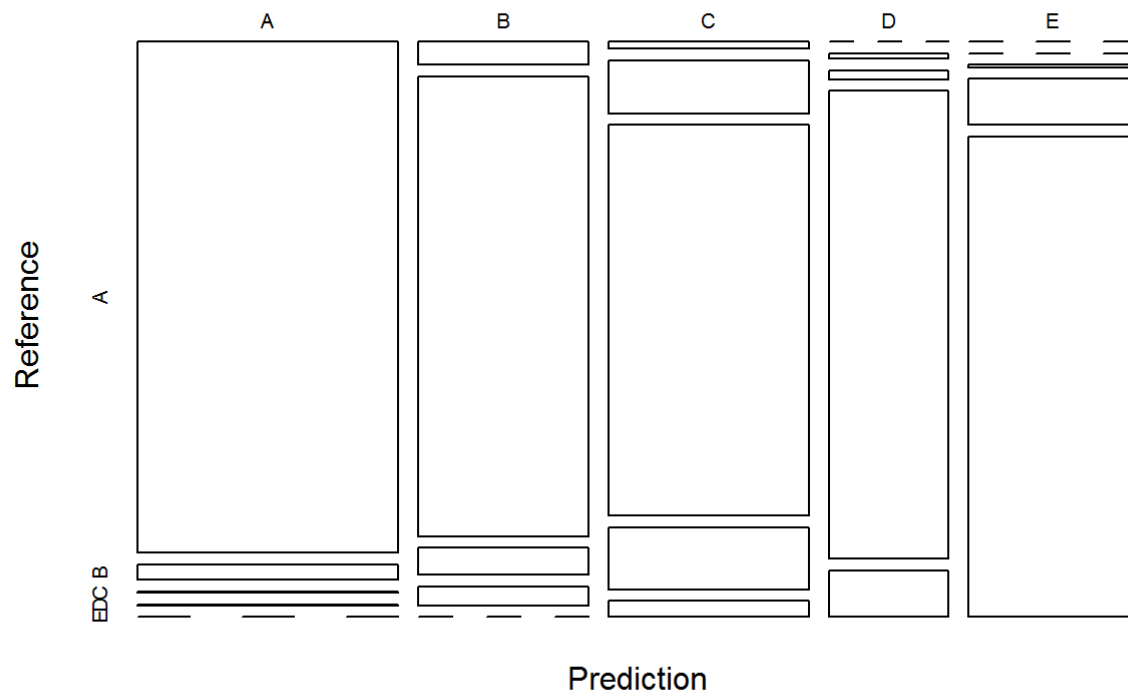
Rattle 2016-Sep-30 08:27:54 admin

```
predictionsA1 <- predict(modFitA1, testingData, type = "class")
cmtree <- confusionMatrix(predictionsA1, testingData$classe)
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##           A 2147   65    6    4    0
##           B   63 1269   75   53    0
##           C   22  173 1263  200   52
##           D    0   11   17  905   90
##           E    0    0    7  124 1300
##
## Overall Statistics
##
##           Accuracy : 0.8774
##           95% CI : (0.8699, 0.8846)
##           No Information Rate : 0.2845
##           P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.8449
##           McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##           Class: A Class: B Class: C Class: D Class: E
## Sensitivity      0.9619  0.8360  0.9232  0.7037  0.9015
## Specificity      0.9866  0.9698  0.9310  0.9820  0.9795
## Pos Pred Value   0.9662  0.8692  0.7386  0.8847  0.9085
## Neg Pred Value   0.9849  0.9610  0.9829  0.9442  0.9779
## Prevalence       0.2845  0.1935  0.1744  0.1639  0.1838
## Detection Rate   0.2736  0.1617  0.1610  0.1153  0.1657
## Detection Prevalence 0.2832  0.1861  0.2179  0.1304  0.1824
## Balanced Accuracy 0.9743  0.9029  0.9271  0.8429  0.9405
```

```
plot(cmtree$table, col = cmtree$byClass, main = paste("Decision Tree Confusion Matrix: Accuracy
=", round(cmtree$overall['Accuracy'], 4)))
```

Decision Tree Confusion Matrix: Accuracy = 0.8774



Prediction using Random Forest

```
set.seed(7575)
modFitB1 <- randomForest(classe ~ ., data=trainingData)
predictionB1 <- predict(modFitB1, testingData, type = "class")
cmrf <- confusionMatrix(predictionB1, testingData$classe)
cmrf
```

Confusion Matrix and Statistics

##

Reference

Prediction A B C D E

A 2231 0 0 0 0

B 1 1518 2 0 0

C 0 0 1366 1 0

D 0 0 0 1285 3

E 0 0 0 0 1439

##

Overall Statistics

##

Accuracy : 0.9991

95% CI : (0.9982, 0.9996)

No Information Rate : 0.2845

P-Value [Acc > NIR] : < 2.2e-16

##

Kappa : 0.9989

McNemar's Test P-Value : NA

##

Statistics by Class:

##

Class: A Class: B Class: C Class: D Class: E

Sensitivity 0.9996 1.0000 0.9985 0.9992 0.9979

Specificity 1.0000 0.9995 0.9998 0.9995 1.0000

Pos Pred Value 1.0000 0.9980 0.9993 0.9977 1.0000

Neg Pred Value 0.9998 1.0000 0.9997 0.9998 0.9995

Prevalence 0.2845 0.1935 0.1744 0.1639 0.1838

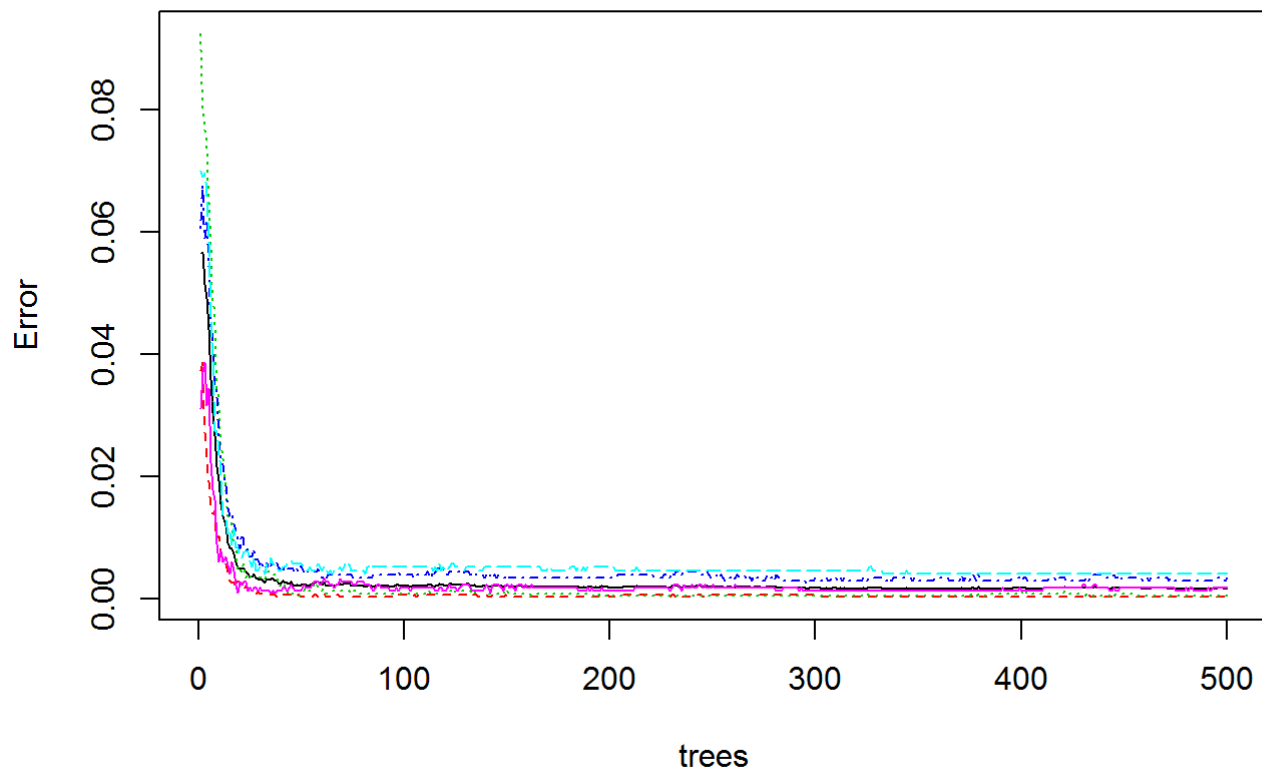
Detection Rate 0.2843 0.1935 0.1741 0.1638 0.1834

Detection Prevalence 0.2843 0.1939 0.1742 0.1642 0.1834

Balanced Accuracy 0.9998 0.9998 0.9992 0.9994 0.9990

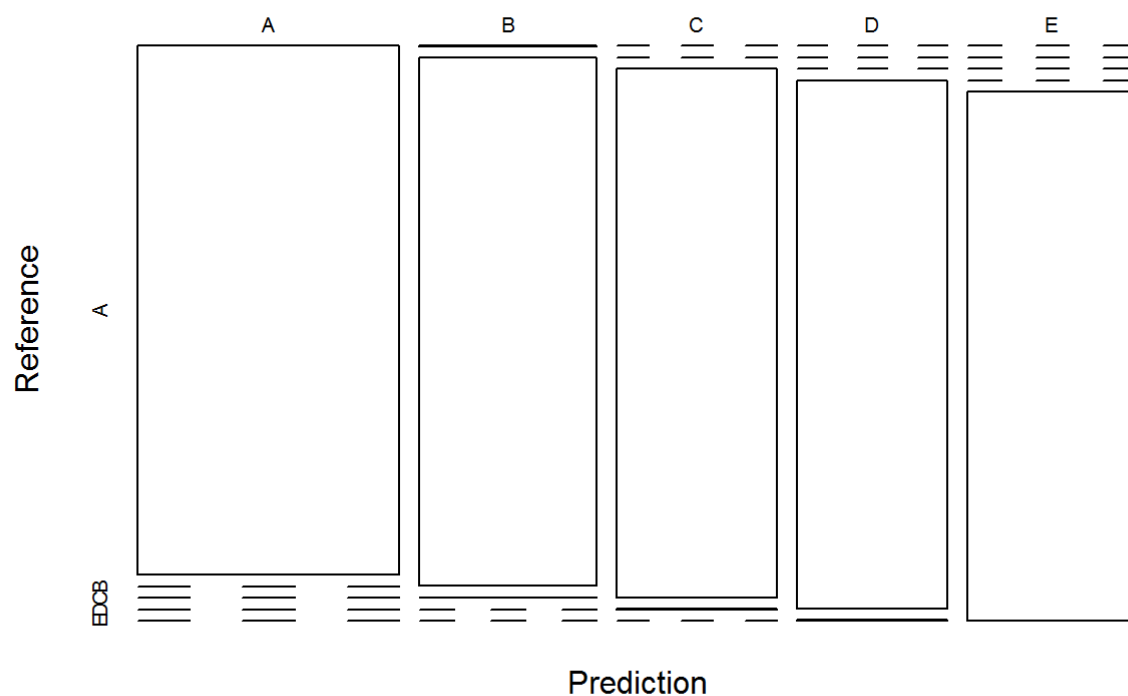
plot(modFitB1)

modFitB1



```
plot(cmrfr$table, col = cmtree$byClass, main = paste("Random Forest Confusion Matrix: Accuracy = ", round(cmrfr$overall['Accuracy'], 4)))
```


Random Forest Confusion Matrix: Accuracy = 0.9991



Prediction using Generalized Boosted Regression

```
set.seed(7575)
fitControl <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
gbmFit1 <- train(classe ~ ., data=trainingData, method = "gbm", trControl = fitControl, verbose
= FALSE)
gbmFinMod1 <- gbmFit1$finalModel

gbmPredTest <- predict(gbmFit1, newdata=testingData)

gbmAccuracyTest <- confusionMatrix(gbmPredTest, testingData$classe)
gbmAccuracyTest
```

Confusion Matrix and Statistics

##

Reference

Prediction A B C D E

A 2229 1 0 0 0

B 3 1512 1 0 0

C 0 3 1360 3 0

D 0 2 7 1281 4

E 0 0 0 2 1438

##

Overall Statistics

##

Accuracy : 0.9967

95% CI : (0.9951, 0.9978)

No Information Rate : 0.2845

P-Value [Acc > NIR] : < 2.2e-16

##

Kappa : 0.9958

McNemar's Test P-Value : NA

##

Statistics by Class:

##

Class: A Class: B Class: C Class: D Class: E

Sensitivity 0.9987 0.9960 0.9942 0.9961 0.9972

Specificity 0.9998 0.9994 0.9991 0.9980 0.9997

Pos Pred Value 0.9996 0.9974 0.9956 0.9900 0.9986

Neg Pred Value 0.9995 0.9991 0.9988 0.9992 0.9994

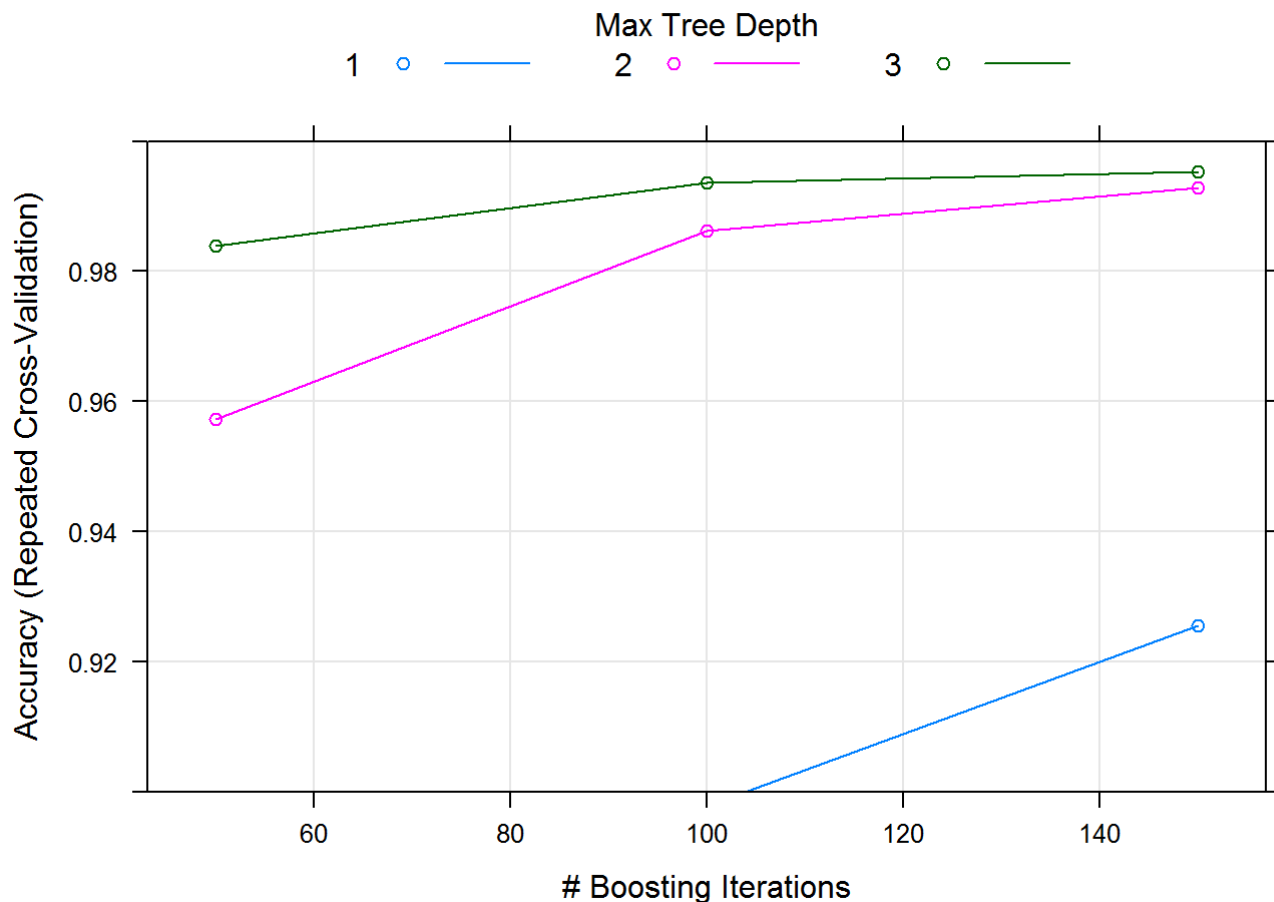
Prevalence 0.2845 0.1935 0.1744 0.1639 0.1838

Detection Rate 0.2841 0.1927 0.1733 0.1633 0.1833

Detection Prevalence 0.2842 0.1932 0.1741 0.1649 0.1835

Balanced Accuracy 0.9992 0.9977 0.9966 0.9971 0.9985

plot(gbmFit1, ylim=c(0.9, 1))



Predicting Results on the Test Data

```
predictionB2 <- predict(modFitB1, testing, type = "class")
predictionB2
```

```
##  2 31  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
##  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

Random Forest and Boosted Regression were superior models for prediction of exercise quality compared to part. The nominal categories were dependent on various variables and the interaction between them. The RF, BR model had over 99% accuracy and fitted well to other subsamples of the data. However, the algorithm may not have as high of accuracy on other samples, particularly ones with different subjects.

Overall, it is interesting to see how monitors are affected by the quality of an exercise and are able to predict the error made which is an important indicator for health and fitness as it is not just the quantity of exercise that can be collected and analyzed but also the quality.