HumanActivityRecognition

Abiyu Giday

November 22, 2015

# Motivation:

When it comes to sports exercise, quantifying repetition of a particular activity is easier than measuring quality or how well the an exercise is performed. Paying attention to both frequency and quality will result in optimal results. However, measuring quality of physical activity is a subtle but important aspect that differs from one person to the other. Here we will discuss how machine Machine Learning (ML) algorithm is utilized to recognize exercise patterns for six individuals in the experiment and accurately predict which person performed which 1 of 5 quality grade exercises. On this case, after considering SVM and NNET algorithms, a Random Forest (RF) algorithm resulted in 98% level of accuracy after it was trained with a Human Activity Recognition (HAR) dataset comes from [here.](http://groupware.les.inf.puc-rio.br/har) Caret package’s ‘Train’ function was used to preprocess, train and tune the model for the human activity recognition dataset. Principal Component Analysis (PCA) function was used to in the preprocessing stage to separate the signal from the noise and select optimal features based on finding new set of multivariate variables that represent as much of the variability. PCA needed 25 features to capture 95% of the variance. Because the dataset contains 19,622 observations and 160 features, a fairly large data size for personal computers, the ‘foreach’ and ‘doMC’ packages were used to configure 8 core processors to cut the amount of time it took to train the RF algorithm from 3+ hours to 10 minutes.

# The Experiment:

The HAR dataset was collected from researchers at XYZ. To measure quality, they chose 6 individuals from the ages of 21-49 and have them lift a dumbbell in five qualitatively different manners. One of the repetition was performed accurately, rated A, the other four were variations of less than optimal way of performing the exercise (rated B-E). Four sensors, similar to once found in smartphones, were strapped to each person, on the belt, bicep, arm and palms. This sensors then sent data while lifting the dumbbell.

# Machine Learning Steps

The following steps detail the data collection and exploration steps, followed by how parallel computing was configured and used to minimize the amount to time it took to process the data. We then discuss the steps to train, tune and evaluate the data.

## Data Collection:

Here are all of 160 features in the original data set. There are two downloads. One for the training datd and the other os the test data.

setwd("~/Documents/Data-Science/DataScienceSpecialization/MachineLearning/Project1")  
#rm(list=ls())   
#load relevant libraryies  
library(dplyr)

##   
## Attaching package: 'dplyr'  
##   
## The following objects are masked from 'package:stats':  
##   
## filter, lag  
##   
## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(caret)

## Loading required package: lattice  
## Loading required package: ggplot2

library(randomForest)

## randomForest 4.6-12  
## Type rfNews() to see new features/changes/bug fixes.  
##   
## Attaching package: 'randomForest'  
##   
## The following object is masked from 'package:dplyr':  
##   
## combine

library(rattle)

## Rattle: A free graphical interface for data mining with R.  
## Version 4.0.0 Copyright (c) 2006-2015 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

library(pROC)

## Type 'citation("pROC")' for a citation.  
##   
## Attaching package: 'pROC'  
##   
## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

library(evtree)

## Loading required package: partykit  
## Loading required package: grid

library(foreach)  
library(doMC)

## Loading required package: iterators  
## Loading required package: parallel

registerDoMC(cores = 8)  
  
  
#Download training data   
if(!file.exists("./data")){dir.create("./data")}  
fileUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"  
download.file(fileUrl, destfile = "./data/pml-training.csv", method = "curl")  
  
trainData <- read.csv("./data/pml-training.csv")  
  
#Structure of the dataset  
#str(trainData)  
names(trainData)

## [1] "X" "user\_name"   
## [3] "raw\_timestamp\_part\_1" "raw\_timestamp\_part\_2"   
## [5] "cvtd\_timestamp" "new\_window"   
## [7] "num\_window" "roll\_belt"   
## [9] "pitch\_belt" "yaw\_belt"   
## [11] "total\_accel\_belt" "kurtosis\_roll\_belt"   
## [13] "kurtosis\_picth\_belt" "kurtosis\_yaw\_belt"   
## [15] "skewness\_roll\_belt" "skewness\_roll\_belt.1"   
## [17] "skewness\_yaw\_belt" "max\_roll\_belt"   
## [19] "max\_picth\_belt" "max\_yaw\_belt"   
## [21] "min\_roll\_belt" "min\_pitch\_belt"   
## [23] "min\_yaw\_belt" "amplitude\_roll\_belt"   
## [25] "amplitude\_pitch\_belt" "amplitude\_yaw\_belt"   
## [27] "var\_total\_accel\_belt" "avg\_roll\_belt"   
## [29] "stddev\_roll\_belt" "var\_roll\_belt"   
## [31] "avg\_pitch\_belt" "stddev\_pitch\_belt"   
## [33] "var\_pitch\_belt" "avg\_yaw\_belt"   
## [35] "stddev\_yaw\_belt" "var\_yaw\_belt"   
## [37] "gyros\_belt\_x" "gyros\_belt\_y"   
## [39] "gyros\_belt\_z" "accel\_belt\_x"   
## [41] "accel\_belt\_y" "accel\_belt\_z"   
## [43] "magnet\_belt\_x" "magnet\_belt\_y"   
## [45] "magnet\_belt\_z" "roll\_arm"   
## [47] "pitch\_arm" "yaw\_arm"   
## [49] "total\_accel\_arm" "var\_accel\_arm"   
## [51] "avg\_roll\_arm" "stddev\_roll\_arm"   
## [53] "var\_roll\_arm" "avg\_pitch\_arm"   
## [55] "stddev\_pitch\_arm" "var\_pitch\_arm"   
## [57] "avg\_yaw\_arm" "stddev\_yaw\_arm"   
## [59] "var\_yaw\_arm" "gyros\_arm\_x"   
## [61] "gyros\_arm\_y" "gyros\_arm\_z"   
## [63] "accel\_arm\_x" "accel\_arm\_y"   
## [65] "accel\_arm\_z" "magnet\_arm\_x"   
## [67] "magnet\_arm\_y" "magnet\_arm\_z"   
## [69] "kurtosis\_roll\_arm" "kurtosis\_picth\_arm"   
## [71] "kurtosis\_yaw\_arm" "skewness\_roll\_arm"   
## [73] "skewness\_pitch\_arm" "skewness\_yaw\_arm"   
## [75] "max\_roll\_arm" "max\_picth\_arm"   
## [77] "max\_yaw\_arm" "min\_roll\_arm"   
## [79] "min\_pitch\_arm" "min\_yaw\_arm"   
## [81] "amplitude\_roll\_arm" "amplitude\_pitch\_arm"   
## [83] "amplitude\_yaw\_arm" "roll\_dumbbell"   
## [85] "pitch\_dumbbell" "yaw\_dumbbell"   
## [87] "kurtosis\_roll\_dumbbell" "kurtosis\_picth\_dumbbell"   
## [89] "kurtosis\_yaw\_dumbbell" "skewness\_roll\_dumbbell"   
## [91] "skewness\_pitch\_dumbbell" "skewness\_yaw\_dumbbell"   
## [93] "max\_roll\_dumbbell" "max\_picth\_dumbbell"   
## [95] "max\_yaw\_dumbbell" "min\_roll\_dumbbell"   
## [97] "min\_pitch\_dumbbell" "min\_yaw\_dumbbell"   
## [99] "amplitude\_roll\_dumbbell" "amplitude\_pitch\_dumbbell"  
## [101] "amplitude\_yaw\_dumbbell" "total\_accel\_dumbbell"   
## [103] "var\_accel\_dumbbell" "avg\_roll\_dumbbell"   
## [105] "stddev\_roll\_dumbbell" "var\_roll\_dumbbell"   
## [107] "avg\_pitch\_dumbbell" "stddev\_pitch\_dumbbell"   
## [109] "var\_pitch\_dumbbell" "avg\_yaw\_dumbbell"   
## [111] "stddev\_yaw\_dumbbell" "var\_yaw\_dumbbell"   
## [113] "gyros\_dumbbell\_x" "gyros\_dumbbell\_y"   
## [115] "gyros\_dumbbell\_z" "accel\_dumbbell\_x"   
## [117] "accel\_dumbbell\_y" "accel\_dumbbell\_z"   
## [119] "magnet\_dumbbell\_x" "magnet\_dumbbell\_y"   
## [121] "magnet\_dumbbell\_z" "roll\_forearm"   
## [123] "pitch\_forearm" "yaw\_forearm"   
## [125] "kurtosis\_roll\_forearm" "kurtosis\_picth\_forearm"   
## [127] "kurtosis\_yaw\_forearm" "skewness\_roll\_forearm"   
## [129] "skewness\_pitch\_forearm" "skewness\_yaw\_forearm"   
## [131] "max\_roll\_forearm" "max\_picth\_forearm"   
## [133] "max\_yaw\_forearm" "min\_roll\_forearm"   
## [135] "min\_pitch\_forearm" "min\_yaw\_forearm"   
## [137] "amplitude\_roll\_forearm" "amplitude\_pitch\_forearm"   
## [139] "amplitude\_yaw\_forearm" "total\_accel\_forearm"   
## [141] "var\_accel\_forearm" "avg\_roll\_forearm"   
## [143] "stddev\_roll\_forearm" "var\_roll\_forearm"   
## [145] "avg\_pitch\_forearm" "stddev\_pitch\_forearm"   
## [147] "var\_pitch\_forearm" "avg\_yaw\_forearm"   
## [149] "stddev\_yaw\_forearm" "var\_yaw\_forearm"   
## [151] "gyros\_forearm\_x" "gyros\_forearm\_y"   
## [153] "gyros\_forearm\_z" "accel\_forearm\_x"   
## [155] "accel\_forearm\_y" "accel\_forearm\_z"   
## [157] "magnet\_forearm\_x" "magnet\_forearm\_y"   
## [159] "magnet\_forearm\_z" "classe"

#Download test data   
if(!file.exists("./data")){dir.create("./data")}  
fileUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"  
download.file(fileUrl, destfile = "./data/pml-testing.csv", method = "curl")  
  
testData <- read.csv("./data/pml-testing.csv")

## Data Exploration:

In the following syrps we will tidy the data with dplyr package. Variables not used as measurment and contain only missing values are removed resulting in just 53 features.

#Remove variables that contain only NA  
library(dplyr)  
df1 <- trainData  
df2 <- df1 %>% select(-matches("^kurtosis|^skewness|^max|^min|^amplitude|^var|^stddev|^avg"))  
  
#Remove variables that do not measure performance & reorder data table with classe feature appearing on column 1  
df3 <- df2[,-c(1:7)]   
df3 <- df3[, c(53,1:52)]   
  
# Structur of the new dataset  
str(df3)

## 'data.frame': 19622 obs. of 53 variables:  
## $ classe : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ 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 -0.02 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 -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 -0.02 -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 ...

#Verify if missing values are in the dataset  
sum(is.na(df2))

## [1] 0

## Parallel Computing:

Here we set the number of cores for parallel computing to 8. This is enabled with the 'doMC' packages. Being able to parallely train the data cut the amount of time by about 90%.

#Register the number of cores to be used.  
library(doMC)  
registerDoMC(cores = 8)

## Spliting the data into training and validation:

The training data is split using the createDataParition function from the caret package. The split was 75/25. 75% is for training the data and the 25% is used to validate the trained model.

library(caret)  
#Split the training data into training and validation  
set.seed(1582)  
inTrain <- createDataPartition(y = df3$classe, p = 0.75, list = FALSE)  
train22 <- df3[inTrain,]  
valdt22 <- df3[-inTrain,]  
  
#check if missing value exist  
dim(train22)

## [1] 14718 53

sum(is.na(train22))

## [1] 0

dim(valdt22)

## [1] 4904 53

sum(is.na(valdt22))

## [1] 0

#PreProcessing

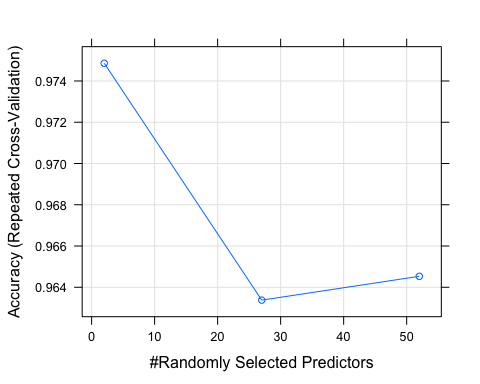
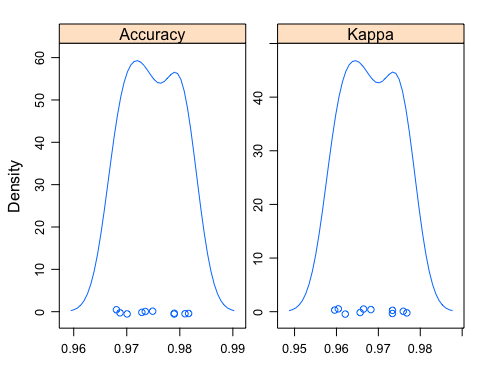
## Tuning and training the Model

Because it takes long time to train the data when number of folds or number of resampling iteration is set to 10. On this case we left it to a default. In the training data Random Forest (rf) is used as method, and Principal Component Analysis (PCA) is turned on for preprocessing, centering and scaling of the data is also excplicityly set. The system.time function is also used to time how long it takes to process the data.

fitControl <- trainControl(   
 method = "repeatedcv",  
 #number = 10,  
 #repeats = 10,  
 allowParallel = TRUE)  
  
set.seed(222)  
system.time(modFit1 <- train(classe ~.,  
 data = train22,  
 method = "rf",  
 preProcess=c("pca","center","scale"),  
 trControl=fitControl))

## user system elapsed   
## 1202.429 21.980 234.705

### Plots for the model

The following set of plots show Model vs Accuracy, the Accuracy and kappa no0rmal curves. And the  

### Variable Importance

Of the 53 variables used in the training data, preprocessing with PCA identfied **25 predictors that can explain 95%** of the variance.

## Created from 14718 samples and 52 variables  
##   
## Pre-processing:  
## - centered (52)  
## - ignored (0)  
## - principal component signal extraction (52)  
## - scaled (52)  
##   
## PCA needed 25 components to capture 95 percent of the variance

## rf variable importance  
##   
## only 20 most important variables shown (out of 25)  
##   
## Overall  
## PC8 100.00  
## PC14 96.89  
## PC12 90.66  
## PC1 88.69  
## PC5 72.30  
## PC3 69.42  
## PC15 66.35  
## PC9 62.69  
## PC6 61.13  
## PC2 59.73  
## PC7 49.27  
## PC16 48.58  
## PC22 47.00  
## PC17 45.48  
## PC4 41.69  
## PC13 41.04  
## PC10 40.64  
## PC25 40.08  
## PC21 39.85  
## PC20 35.80

#### Variable importance plot

#### Scatter plot of selected important variables

The scatter plot uses the top 4 variables and their relations to each person that participated in the experiment.

# Scatterplot Matrix with Ellipses  
featurePlot(x = df2[, c(8,10,20,45)],  
 y = df2$user\_name,  
 plot = "ellipse",  
 ## Add a key at the top  
 auto.key = list(columns = 3))

# Model Evaluation:

To evaluate the models the **confustionMatrix** function on the valdation data that shows about 98% accuracy and 0.97% error rate. P value is also less also shows that is is my less than 2.2e-16 < 0.05. With the 95% confidence interval between (0.9719, 0.9806). Sensetivity and specificity for the model are in the ranges of 94-99% for all the qualities that are predicted by the moedl. The Kappy value is also at 97% which are all lead to a very high accuracy predition.

modFit1$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: 2  
##   
## OOB estimate of error rate: 2.28%  
## Confusion matrix:  
## A B C D E class.error  
## A 4159 10 11 3 2 0.006212664  
## B 46 2766 29 2 5 0.028792135  
## C 2 31 2506 27 1 0.023763148  
## D 5 2 103 2297 5 0.047678275  
## E 1 15 17 18 2655 0.018847007

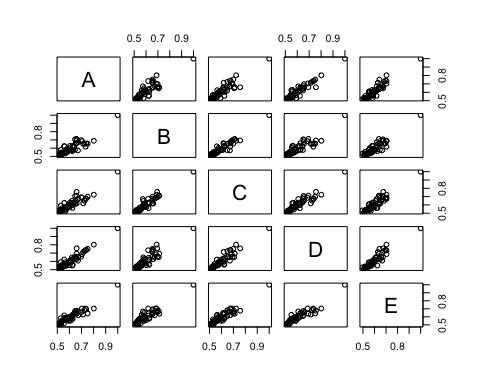
confusionMatrix(valdt22$classe,predict(modFit1,valdt22))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1390 1 1 2 1  
## B 21 918 9 0 1  
## C 1 18 828 7 1  
## D 1 1 39 761 2  
## E 0 0 5 4 892  
##   
## Overall Statistics  
##   
## Accuracy : 0.9765   
## 95% CI : (0.9719, 0.9806)  
## No Information Rate : 0.2881   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9703   
## Mcnemar's Test P-Value : 2.548e-07   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9837 0.9787 0.9388 0.9832 0.9944  
## Specificity 0.9986 0.9922 0.9933 0.9896 0.9978  
## Pos Pred Value 0.9964 0.9673 0.9684 0.9465 0.9900  
## Neg Pred Value 0.9934 0.9949 0.9867 0.9968 0.9988  
## Prevalence 0.2881 0.1913 0.1799 0.1578 0.1829  
## Detection Rate 0.2834 0.1872 0.1688 0.1552 0.1819  
## Detection Prevalence 0.2845 0.1935 0.1743 0.1639 0.1837  
## Balanced Accuracy 0.9911 0.9854 0.9660 0.9864 0.9961

# area under the ROC curve for each predictor   
RocImportance <- filterVarImp(x=train22[,-ncol(train22)], y = train22$classe)  
head(RocImportance)

## A B C D E  
## classe 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000  
## roll\_belt 0.6391397 0.6214129 0.6276103 0.6434238 0.6434238  
## pitch\_belt 0.5432035 0.5268634 0.5390042 0.5448612 0.5448612  
## yaw\_belt 0.5625353 0.5601188 0.5546821 0.5492554 0.5625353  
## total\_accel\_belt 0.5680607 0.5385448 0.5279744 0.5435384 0.5680607  
## gyros\_belt\_x 0.5291272 0.5291272 0.5107615 0.5099477 0.5107700

plot(RocImportance)



#Testing the model against the testData  
data.frame(TestData = testData$user\_name, Predicted = predict(modFit1, testData))

## TestData Predicted  
## 1 pedro B  
## 2 jeremy A  
## 3 jeremy C  
## 4 adelmo A  
## 5 eurico A  
## 6 jeremy E  
## 7 jeremy D  
## 8 jeremy B  
## 9 carlitos A  
## 10 charles A  
## 11 carlitos A  
## 12 jeremy C  
## 13 eurico B  
## 14 jeremy A  
## 15 jeremy E  
## 16 eurico E  
## 17 pedro A  
## 18 carlitos B  
## 19 pedro B  
## 20 eurico B

# Take Away:

The objective of this excise had been to create a model that will accurately recognize patterns and predict the manner (quality) in which the participants of the experiment exercised from the collected dataset. Utilizing the Random Forest algorithm, a model was generated with Caret packages trControl function, and preprocessed with PCA that select optimal number of high valued variables. Although the the accu resulted in 99.25% pattern recognition and prediction accuracy.

# Reference

\_Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements Read more: <http://groupware.les.inf.puc-rio.br/har#ixzz3sEPIq2gc_>

\_Max Kuhn. PhD Pfizer Global R&D, <http://topepo.github.io/caret/index.html_>

\_Jeff Leek, PhD - Practical Machine Learning lectures. <https://www.coursera.org/course/predmachlearn_>

\_Wikepedia: <https://en.wikipedia.org/wiki/Machine_learning_>

\_StackOverflow: <http://stackoverflow.com/_>