Project for Practical ML Class

Alvaro Ortiz

February 2, 2018

Introduction

The goal of this project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. We may use any of the other variables to predict with.

You can find more information about the data collected in this weblink
http://groupware.les.inf.puc-rio.br/har

Assignment

Before doing anything else, we download the csv files for the train and test set and load the data in R.

```
URL1 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
URL2 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(url=URL1, destfile="train.csv")
download.file(url=URL2, destfile="test.csv")

train <- read.csv("train.csv")
test <- read.csv("test.csv")
```

We take a first look into the train data set to see what kind of variables do we have.

```
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 1323084232 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 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 1.45 ...
```

```
##  $ 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 ...
##  $ 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 -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 ...
##  $ 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 -0.03 -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
...
##  $ 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]
```

As we can see, there are 160 vars in the dataset, let's see which are have variance near zero and therefore aren't going to be useful when it comes to use them to make predictions about the classes.

```
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(munsell)
library(e1071)
zeroVar <- nearZeroVar(train,saveMetrics=TRUE)
train <- train[,!zeroVar$nzv]
test <- test[,!zeroVar$nzv]
ncol(train)
```

```
## [1] 100
```

We have got rid of 60 vars.

Let's check for missing values as well

```
spl <- sort(sapply(train, function(x){sum(is.na(x))}), decreasing = TRUE) > 0
names(train[,spl])
```

```
##  [1] "X"                    "user_name"            "raw_timestamp_part_1"
##  [4] "raw_timestamp_part_2" "cvtd_timestamp"       "num_window"
##  [7] "roll_belt"            "pitch_belt"           "yaw_belt"
## [10] "total_accel_belt"     "max_roll_belt"        "max_picth_belt"
## [13] "min_roll_belt"        "min_pitch_belt"       "amplitude_roll_belt"
## [16] "amplitude_pitch_belt" "var_total_accel_belt" "avg_roll_belt"
## [19] "stddev_roll_belt"     "var_roll_belt"        "avg_pitch_belt"
## [22] "stddev_pitch_belt"    "var_pitch_belt"       "avg_yaw_belt"
## [25] "stddev_yaw_belt"      "var_yaw_belt"         "gyros_belt_x"
## [28] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [31] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [34] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [37] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [40] "var_accel_arm"        "gyros_arm_x"
```

so we have 41 vars with missing values. furthermore, most of the values for this vars are actually missing values, so we proceed to get rid of them.

```
spl <- colSums(is.na(train)) == 0
train <- train[,spl]
test <- test[,spl]
```

there other variables in the dataset that don't add any value to our prediction model, for instance, ID number of observations, name of the subject lifting weights, several timestamps (we aren't taking into account any time dependence)

```
train$X <- NULL
test$X <- NULL
train$user_name <- NULL
test$user_name <- NULL
train$raw_timestamp_part_1 <- NULL
test$raw_timestamp_part_1 <- NULL
train$raw_timestamp_part_2 <- NULL
test$raw_timestamp_part_2 <- NULL
train$cvtd_timestamp <- NULL
test$cvtd_timestamp <- NULL
train$num_window <- NULL
test$num_window <- NULL
```

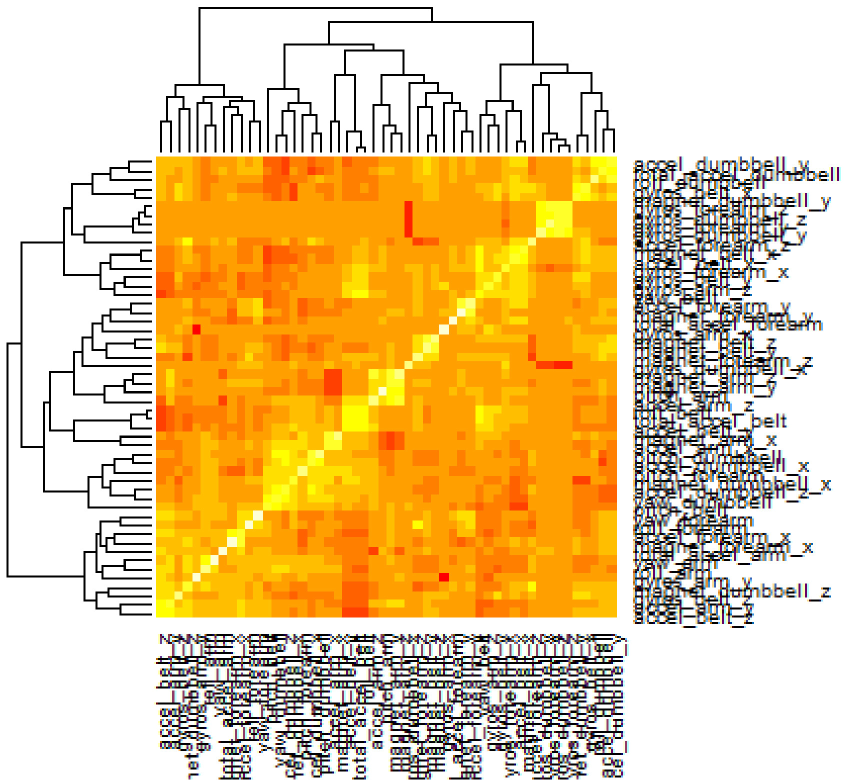
And now we end up with 52 vars plus the output, far less than the original 159 vars plus output.

Now we need to decide which kind of modeling we are going to use. As we don't want to take a look through exploratory analysis to the validation set it is time to split the train set into train and validation and keep this last one unused until the end.

```
inTrain <- createDataPartition(y=train$classe, p=0.7, list=FALSE)
train <- train[inTrain,]
validation <- train[-inTrain,]
```

we will know perform some exploratory analysis. for example we can check what is the correlation among the variables to see if we should go for a linear or nonlinear model.

```
corrM <- cor(train[,1:52])
heatmap(corrM)
```

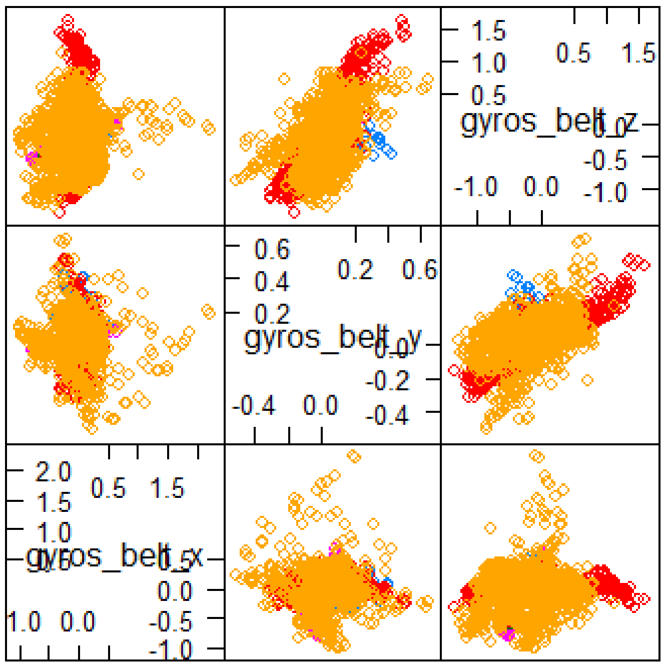


```
table(corrM > 0.8)

##
## FALSE  TRUE
##  2630    74
```

70 values are over 0.8 but we need to take into account that 52 belongs to the diagonal of the matrix and thus are trivial values. As there is not much correlation among different columns we are going to plot just a few set of variables to see how much of nonlinearity they have, and then we will proceed to fit a random forest if we confirm the previous point.

first we plot just the variables for the gyroscope in the belt area

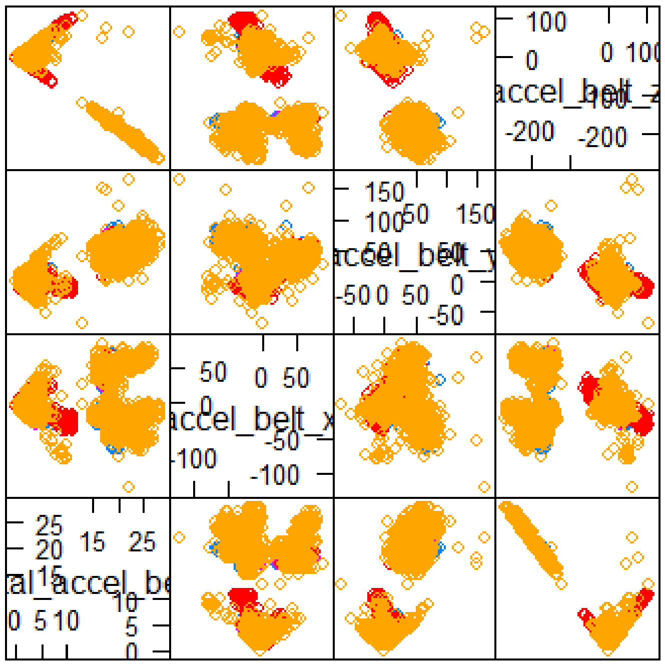
```
vars <- names(train[,grep("gyros_belt", names(train))])
featurePlot(x = train[,vars], y = train$classe, plot = "pairs")
```



Scatter Plot Matrix

and the accelerometer in the belt area.

```
vars <- names(train[,grep("accel_belt", names(train))])
featurePlot(x = train[,vars], y = train$classe, plot = "pairs")
```



Scatter Plot Matrix

the differnet colors represent the different classes that we want to predict. As we can see, a random forest seems a pretty good option in our case given the nonlinearity that our data exhibits.

```
set.seed(144)
fit <- train(classe ~ ., method = "rf", data = train, trControl =
trainControl(method = "cv", number = 3))
predValidation <- predict(fit, newdata=validation)
confusionMatrix(predValidation,validation$classe)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction    A    B    C    D    E
##          A 1141    0    0    0    0
##          B    0  829    0    0    0
##          C    0    0  725    0    0
##          D    0    0    0  674    0
##          E    0    0    0    0  754
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2767
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

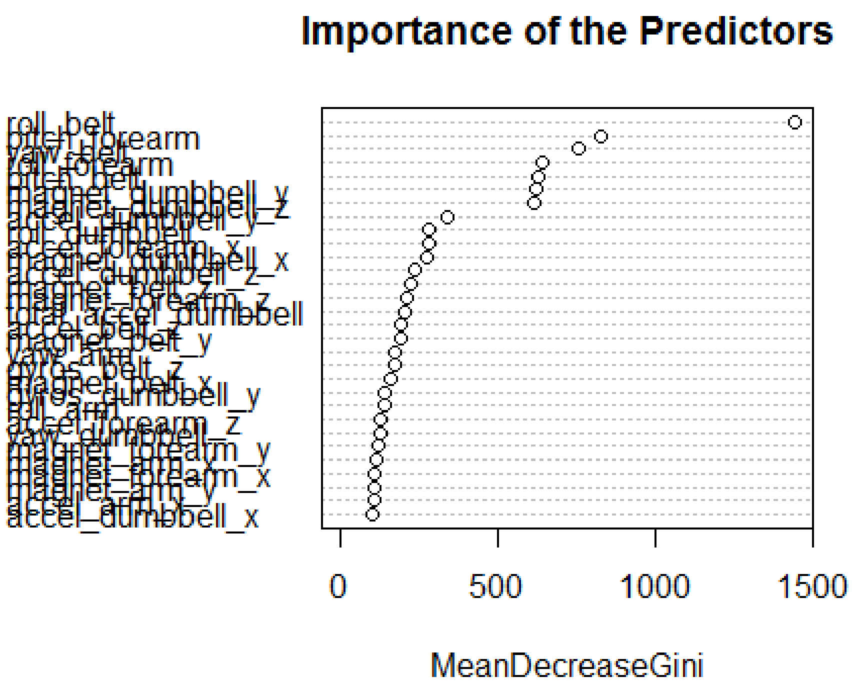
```
##                  Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2767   0.2011   0.1758   0.1635   0.1829
## Detection Rate         0.2767   0.2011   0.1758   0.1635   0.1829
## Detection Prevalence   0.2767   0.2011   0.1758   0.1635   0.1829
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Finally, we make a prediction over the test set. this is the one we will input in the quiz for this week on the coursera page

```
predTest <- predict(fit, newdata=test)
```

Lastly, and just to gain some insight on our data, we can take a look at what variables account for most of the infomration to predict our output

```
imp <- varImp(fit)$importance
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:ggplot2':
##
##     margin

varImpPlot(fit$finalModel, sort = TRUE, main = "Importance of the
Predictors")
```



Conclusions

We have an accuracy of 99% in an out of sample data. which give us an out of sample error of aproximately 0.1%

The prediction values for the test set have been exported so we can input them into the quiz section of the Course