Practical Machine Learning Project

Antonio Lloris Amor

1 Introduction

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity to predict the manner in which they did the exercise ("classe" variable in the training set). 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.

In this project, our goal will be to use data from four sensors placed in: belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

In this report we are going to describe how we built our model, how we used cross validation, what our expected out of sample error is, and why we made the choices we did. We will also use our prediction model to predict 20 different test cases.

2 Preprocesing

2.1 Data loading and columns pruning

The fisrt step in our study is to load the data.

```
rawData <- read.csv("./OriginalData/pml-training.csv", header = TRUE)
```

Our dataset has many columns but we are interested only in columns related with "roll", "pitch", "yaw" and "total_accel" from sensors. Why this columns and not other, this columns are derivated from x, y and z components from the "magnet", "accel" and "gyros" information of the sensors. There are other columns in original data set but they are derivated from "roll", "pitch", "yaw" and "total_accel" and these columns have many NA's.

```
prunedCol <- colnames(rawData)[grep("^roll_|^pitch_|^yaw_|^total_accel_",
                                    colnames(rawData))]
studyData <- subset(rawData, select = c("classe",prunedCol))
summary(studyData)
```

```
##  classe     roll_belt        pitch_belt          yaw_belt
##  A:5580   Min.   :-28.90   Min.   :-55.8000   Min.   :-180.00
##  B:3797   1st Qu.:  1.10   1st Qu.:  1.7600   1st Qu.: -88.30
##  C:3422   Median :113.00   Median :  5.2800   Median : -13.00
##  D:3216   Mean   : 64.41   Mean   :  0.3053   Mean   : -11.21
##  E:3607   3rd Qu.:123.00   3rd Qu.: 14.9000   3rd Qu.:  12.90
##           Max.   :162.00   Max.   : 60.3000   Max.   : 179.00
##  total_accel_belt    roll_arm         pitch_arm          yaw_arm
##  Min.   : 0.00    Min.   :-180.00   Min.   :-88.800   Min.   :-180.0000
##  1st Qu.: 3.00    1st Qu.: -31.77   1st Qu.:-25.900   1st Qu.: -43.1000
##  Median :17.00    Median :   0.00   Median :  0.000   Median :   0.0000
##  Mean   :11.31    Mean   :  17.83   Mean   : -4.612   Mean   :  -0.6188
##  3rd Qu.:18.00    3rd Qu.:  77.30   3rd Qu.: 11.200   3rd Qu.:  45.8750
```

```
##  Max.   :29.00    Max.   : 180.00   Max.   : 88.500   Max.   : 180.0000
##  total_accel_arm roll_dumbbell     pitch_dumbbell     yaw_dumbbell
##  Min.   : 1.00   Min.   :-153.71   Min.   :-149.59   Min.   :-150.871
##  1st Qu.:17.00   1st Qu.: -18.49   1st Qu.: -40.89   1st Qu.: -77.644
##  Median :27.00   Median :  48.17   Median : -20.96   Median :  -3.324
##  Mean   :25.51   Mean   :  23.84   Mean   : -10.78   Mean   :   1.674
##  3rd Qu.:33.00   3rd Qu.:  67.61   3rd Qu.:  17.50   3rd Qu.:  79.643
##  Max.   :66.00   Max.   : 153.55   Max.   : 149.40   Max.   : 154.952
##  total_accel_dumbbell  roll_forearm       pitch_forearm
##  Min.   : 0.00        Min.   :-180.0000   Min.   :-72.50
##  1st Qu.: 4.00        1st Qu.:  -0.7375   1st Qu.:  0.00
##  Median :10.00        Median :  21.7000   Median :  9.24
##  Mean   :13.72        Mean   :  33.8265   Mean   : 10.71
##  3rd Qu.:19.00        3rd Qu.: 140.0000   3rd Qu.: 28.40
##  Max.   :58.00        Max.   : 180.0000   Max.   : 89.80
##   yaw_forearm      total_accel_forearm
##  Min.   :-180.00   Min.   :  0.00
##  1st Qu.: -68.60   1st Qu.: 29.00
##  Median :   0.00   Median : 36.00
##  Mean   :  19.21   Mean   : 34.72
##  3rd Qu.: 110.00   3rd Qu.: 41.00
##  Max.   : 180.00   Max.   :108.00
```

The columns used in our study are: classe, roll_belt, pitch_belt, yaw_belt, total_accel_belt, roll_arm, pitch_arm, yaw_arm, total_accel_arm, roll_dumbbell, pitch_dumbbell, yaw_dumbbell, total_accel_dumbbell, roll_forearm, pitch_forearm, yaw_forearm, total_accel_forearm

2.2 Get training and test sets

For cross validation I going to divide it in two portions with the same size. Each portion will be divided in a training and a test set (60% / 40%).

```
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(rpart)
library(lattice)
library(ggplot2)
library(caret)
library(corrplot)

set.seed(260668)

portions <- createDataPartition(y = studyData$classe,
                                p = 0.5,
                                list = FALSE)
Data1 <- studyData[portions,]
Data2 <- studyData[-portions,]

set.seed(260668)
```

```
inTrain <- createDataPartition(y = Data1$classe,
                               p = 0.6,
                               list = FALSE)
trainingData1 <- Data1[inTrain,]
testingData1 <- Data1[-inTrain,]

set.seed(260668)

inTrain <- createDataPartition(y = Data2$classe,
                               p = 0.6,
                               list = FALSE)
trainingData2 <- Data2[inTrain,]
testingData2 <- Data2[-inTrain,]
```

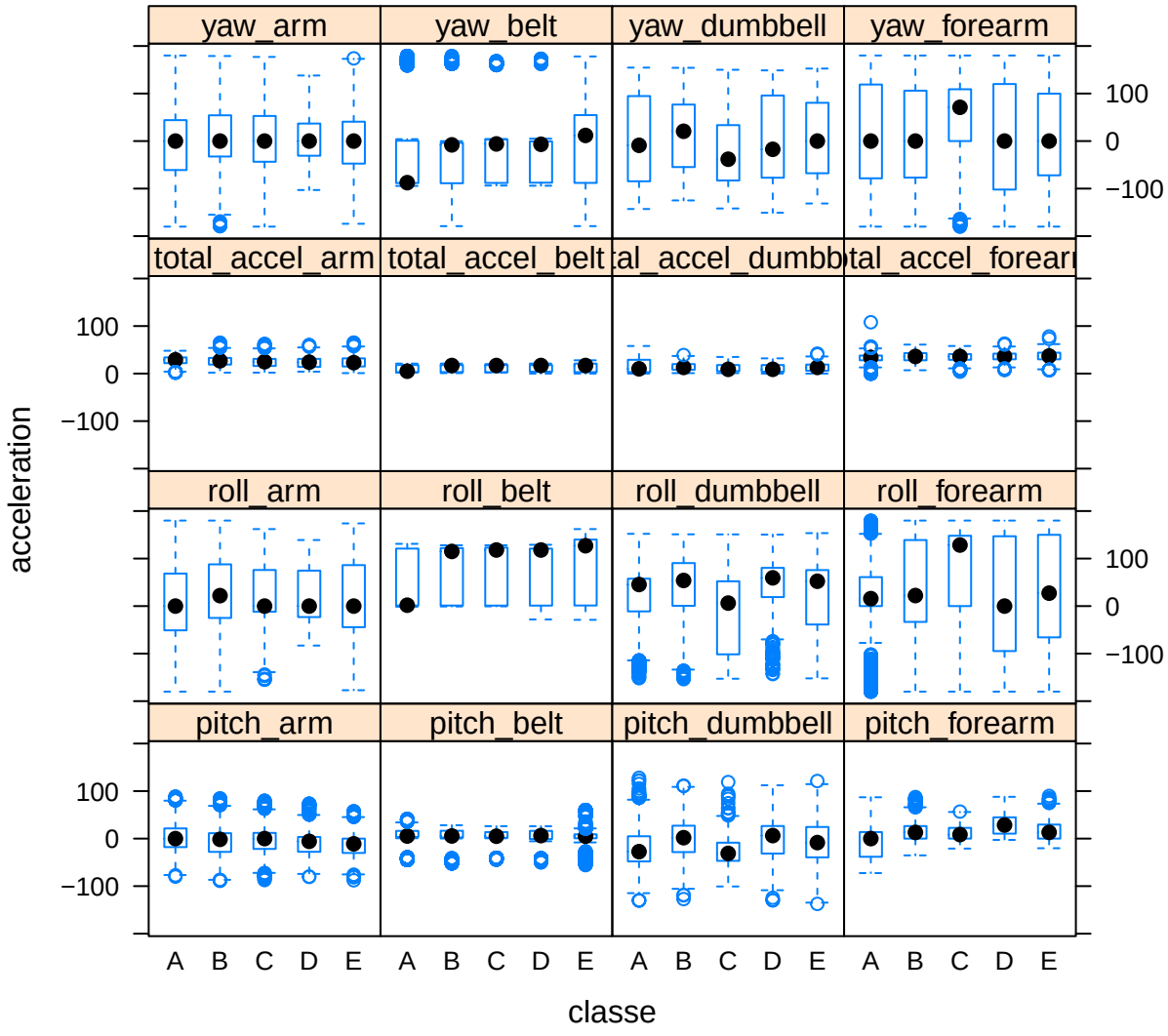
We have two traning sets with 5.889 and 5.887 and two test sets with 3.923 and 3.923 observations.

2.3 Plotting predictors

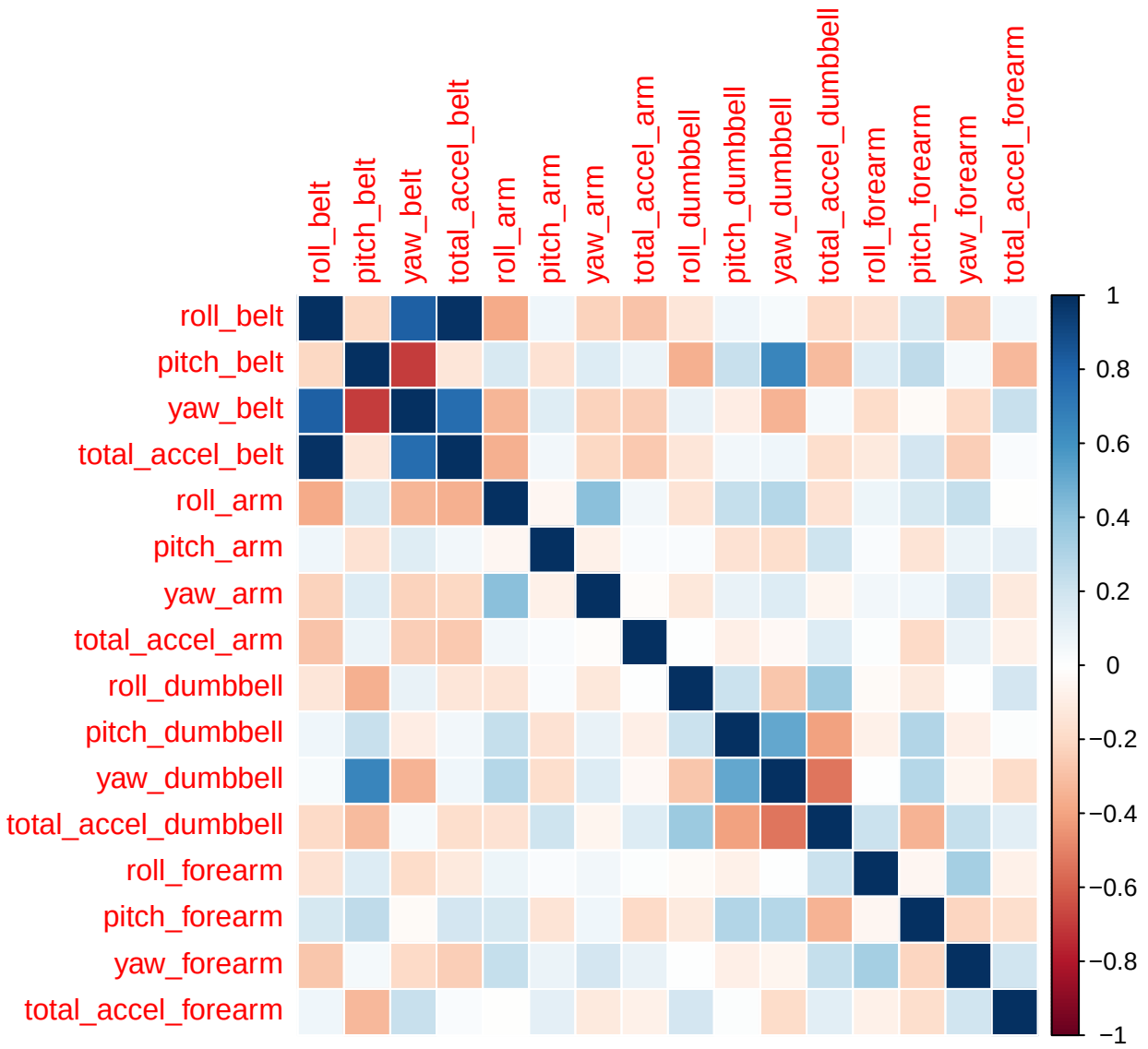
In the following graph we can see that there isn't a unique predictor that can classify the training data.

```
trainingDataU <- rbind(trainingData1,trainingData2)

featurePlot (x = trainingDataU[,-1], y = trainingDataU$classe,
             plot = "box",
             labels = c("classe", "acceleration"))
```




```
corrplot(cor(trainingDataU[,-1]), method = "color")
```



The previous graph show us that there are correlations between the predictors.

3 Prediction

Now I going to use the *traininData*1 and *traininData*2 datasets for training process with different method. The training process include a cross validation step.

3.1 Classification Tree

3.1.1 Training

```
set.seed(260668)

modelFit1 <- train(classe ~., data = trainingData1,
```

```
                  method = "rpart",
                  trControl = trainControl(method = "cv", number = 4))
modelFit1
```

```
## CART
##
## 5889 samples
##   16 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
##
## Summary of sample sizes: 4416, 4417, 4416, 4418
##
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa       Accuracy SD  Kappa SD
##   0.03368921  0.5158661  0.38281804  0.02493363   0.03395737
##   0.04329775  0.4179267  0.21773200  0.06178628   0.11198375
##   0.11720047  0.3233221  0.05970543  0.04521353   0.06904997
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03368921.
```

```
set.seed(260668)

modelFit2 <- train(classe ~., data = trainingData2,
                  method = "rpart",
                  trControl = trainControl(method = "cv", number = 4))
modelFit2
```

```
## CART
##
## 5887 samples
##   16 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
##
## Summary of sample sizes: 4414, 4417, 4414, 4416
##
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa       Accuracy SD  Kappa SD
##   0.04035129  0.4223369  0.22308721  0.06733448   0.11678891
##   0.05174460  0.3949745  0.17469238  0.06261363   0.10694905
##   0.11606931  0.3220600  0.05774784  0.04353458   0.06674853
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.04035129.
```

We can see that the values for *Accuracy* are very pour.

3.1.2 Testing

```
set.seed(260668)

predictions1 <- predict(modelFit1, newdata=testingData1)
confusionMatrix(predictions1, testingData1$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 798 272 146 179  99
##          B  59 289  42 126 174
##          C 186 166 469 165 114
##          D  70  32  27 173  28
##          E   3   0   0   0 306
##
## Overall Statistics
##
##                Accuracy : 0.5187
##                  95% CI : (0.503, 0.5345)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3834
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7151  0.38076   0.6857  0.26905  0.42441
## Specificity            0.7520  0.87326   0.8052  0.95213  0.99906
## Pos Pred Value         0.5341  0.41884   0.4264  0.52424  0.99029
## Neg Pred Value         0.8691  0.85462   0.9238  0.86919  0.88517
## Prevalence             0.2845  0.19347   0.1744  0.16391  0.18379
## Detection Rate         0.2034  0.07367   0.1196  0.04410  0.07800
## Detection Prevalence   0.3808  0.17589   0.2804  0.08412  0.07877
## Balanced Accuracy      0.7336  0.62701   0.7454  0.61059  0.71174
```

```
set.seed(260668)

predictions2 <- predict(modelFit2, newdata=testingData2)
confusionMatrix(predictions2, testingData2$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 959 350 305 225  78
##          B  65 215  30 178  42
##          C  87 194 349 240 265
##          D   0   0   0   0   0
```

```
##          E   5   0   0   0 336
##
## Overall Statistics
##
##                Accuracy : 0.4739
##                  95% CI : (0.4581, 0.4896)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3153
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8593   0.2833  0.51023   0.0000  0.46602
## Specificity            0.6587   0.9004  0.75733   1.0000  0.99844
## Pos Pred Value         0.5003   0.4057  0.30749      NaN  0.98534
## Neg Pred Value         0.9217   0.8397  0.87984   0.8361  0.89252
## Prevalence             0.2845   0.1935  0.17436   0.1639  0.18379
## Detection Rate         0.2445   0.0548  0.08896   0.0000  0.08565
## Detection Prevalence   0.4887   0.1351  0.28932   0.0000  0.08692
## Balanced Accuracy      0.7590   0.5919  0.63378   0.5000  0.73223
```

3.2 Random forest

3.2.1 Training

```
set.seed(260668)

modelFit3 <- train(classe ~., data = trainingData1,
                   method = "rf",
                   trControl = trainControl(method = "cv", number = 4))
modelFit3
```

```
## Random Forest
##
## 5889 samples
##   16 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
##
## Summary of sample sizes: 4416, 4417, 4416, 4418
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9653611  0.9561781  0.007325281  0.009272088
##    9    0.9702842  0.9624194  0.003851995  0.004872157
##   16    0.9631516  0.9534091  0.003975251  0.005016614
```

```
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 9.
```

```
set.seed(260668)

modelFit4 <- train(classe ~., data = trainingData2,
                   method = "rf",
                   trControl = trainControl(method = "cv", number = 4))
modelFit4
```

```
## Random Forest
##
## 5887 samples
##   16 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
##
## Summary of sample sizes: 4414, 4417, 4414, 4416
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9706062  0.9628218  0.009204451  0.011637822
##    9    0.9695911  0.9615365  0.004046331  0.005121274
##   16    0.9619481  0.9518811  0.003403157  0.004312700
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

We can see that the values for *Accuracy* are very pour.

3.2.2 Testing

```
set.seed(260668)

predictions3 <- predict(modelFit3, newdata=testingData1)
confusionMatrix(predictions3, testingData1$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1103   12    1    0    3
##          B    6  723   11    3    9
##          C    0   21  663   11   10
##          D    4    3    9  627    2
##          E    3    0    0    2  697
##
```

```
## Overall Statistics
##
##                Accuracy : 0.972
##                  95% CI : (0.9663, 0.9769)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9645
##  Mcnemar's Test P-Value : 0.001104
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9884   0.9526   0.9693   0.9751   0.9667
## Specificity            0.9943   0.9908   0.9870   0.9945   0.9984
## Pos Pred Value         0.9857   0.9614   0.9404   0.9721   0.9929
## Neg Pred Value         0.9954   0.9886   0.9935   0.9951   0.9925
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2812   0.1843   0.1690   0.1598   0.1777
## Detection Prevalence   0.2852   0.1917   0.1797   0.1644   0.1789
## Balanced Accuracy      0.9913   0.9717   0.9782   0.9848   0.9826
```

```
set.seed(260668)

predictions4 <- predict(modelFit4, newdata=testingData2)
confusionMatrix(predictions4, testingData2$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1107   30    0    1    1
##          B    3  717    9    1    8
##          C    5   10  668   13    2
##          D    0    2    7  626    3
##          E    1    0    0    2  707
##
## Overall Statistics
##
##                Accuracy : 0.975
##                  95% CI : (0.9696, 0.9797)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9684
##  Mcnemar's Test P-Value : 1.396e-05
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9919   0.9447   0.9766   0.9736   0.9806
## Specificity            0.9886   0.9934   0.9907   0.9963   0.9991
## Pos Pred Value         0.9719   0.9715   0.9570   0.9812   0.9958
## Neg Pred Value         0.9968   0.9868   0.9950   0.9948   0.9956
```

```
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2822   0.1828   0.1703   0.1596   0.1802
## Detection Prevalence   0.2903   0.1881   0.1779   0.1626   0.1810
## Balanced Accuracy      0.9903   0.9690   0.9837   0.9850   0.9898
```

3.3 In/Out sample error

3.3.1 In sample error

Method	Training Set #1	Training Set #2	Mean
Classification Tree	0.4841339	0.5776631	0.5308985
Random forest	0.0297158	0.0293938	0.0295548

3.3.2 Out sample error

Method	Testing Set #1	Testing Set #2	Mean
Classification Tree	0.4813	0.5261	0.5037
Random forest	0.028	0.025	0.0265

3.3.3 Conclusion

Out sample error < In sample error, our model don't have overfitting.

4 Validation

Now we are goint to use "$pml - testing.csv$" file for validate our model.

```
rawData <- read.csv("./OriginalData/pml-testing.csv", header = TRUE)
```

```
prunedCol <- colnames(rawData)[grep("^roll_|^pitch_|^yaw_|^total_accel_",
                                    colnames(rawData))]
validationData <- subset(rawData, select = c(prunedCol))
```

```
set.seed(260668)

predictions4 <- predict(modelFit4, newdata=validationData)
predictions4
```

```
##  [1] 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
```