Practical Machine Learning Project

Introduction

The introduction of wearable devices fitted with sensors capable of capturing human motion, has made possible to quantify the amount of physical activity. The objective of this project is to use measurements from sensors placed in the forearm, arm, belt and dumbbell of a group of subjects who were asked to perform a set of 10 repetitions of "Unilateral Dumbbell Biceps Curls" in order to classify the quality of the repetitions.

Data

The data consists of measurements obtained with accelerometers and gyroscopes mounted on the belt, arm, and forearm of the participants in the study, as well as in the dumbbells used during the experiment. The subjects were asked to perform 10 repetitions of "Unilateral Dumbbell Biceps Curls" in five different forms:

- According to specifications (A).
- Throwing the elbows to the front (B).
- Lifting the elbow only halfway (C).
- Lowering the dumbbell only halfway (D).
- Throwing the hips to the front (E).

We will start by loading the training and testing sets:

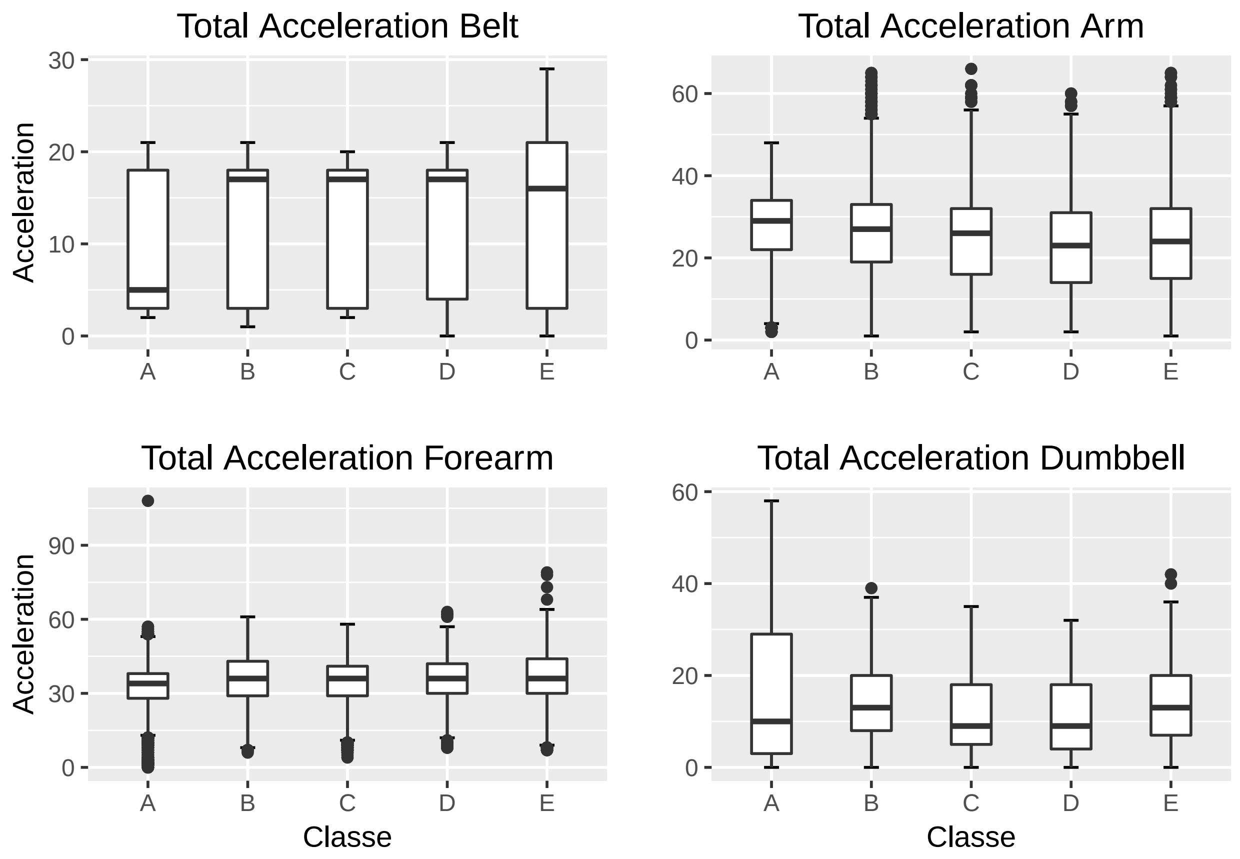
```r
# Load training and testing sets
df.train <- read.csv("pml-training.csv", header = TRUE, sep = ",")
df.test <- read.csv("pml-testing.csv", header = TRUE, sep = ",")
```

Next, we will remove those columns which are not the measurements from the sensors:

```r
# Preprocessing (leave only raw sensor measurements in both datasets)
df.train <- df.train[,c(8:11,37:49,60:68,84:86,102,113:124,140,151:160)]
df.test <- df.test[,c(8:11,37:49,60:68,84:86,102,113:124,140,151:160)]
```

Exploratory Data Analysis

Now we are ready to start looking at our data. We will examine just a few variables in this report (for the sake of brevity), these are: *total arm acceleration*, *total belt acceleration*, *total forearm acceleration*, and *total dumbbell acceleration*. We will now compare them to each other using some summary statistics in the form of boxplots.



It seems that some variables show a stronger difference between the correct repetition (A) and any of the incorrect ones (B,C,D,E), as for example in the case of "Total Acceleration Belt". In other cases, there is no clear difference, as for example in "Total Acceleration Forearm".

Model Fitting

We will fit the data using three techniques: *Bagging*, *Random Forests*, and *Boosting*.

Bagging

Boostrap aggregation (bagging) as applied to decision trees consists of generating **M** decision trees using **M** bootraped training sets, and then averaging the predictions resulting from all the trees. Averaging all the predictions automatically reduces the variance and the risk of overfitting. Let's now fit our model using this technique:

```
# bagging
bg <- randomForest(classe ~., data = df.train, ntree = 300, mtry = 52)
```

```

Call:
 randomForest(formula = classe ~ ., data = df.train, ntree = 300,      mtry = 52)
               Type of random forest: classification
                     Number of trees: 300
No. of variables tried at each split: 52
```

```
        OOB estimate of  error rate: 0.95%
Confusion matrix:
     A    B    C    D    E class.error
A 5560   11    6    0    3 0.003584229
B   37 3742   13    3    2 0.014485120
C    8   21 3376   17    0 0.013442431
D    4    4   32 3172    4 0.013681592
E    0    7    6    9 3585 0.006099251
```

In the case of bagging, the *Out-Of-Bag (OOB)* estimation of the error can be used as a replacement for cross-validation. For the i^{th} observation, the error is obtained using the trees in which such observation was not used during training. The package *randomForest* calculates this automatically for us.

Random Forests

Random forests is similar to bagging except that at each split a different subset of all the predictors is considered. The objective of this is to *decorrelate* the resulting trees from each other. This has proven to improve results over bagging.

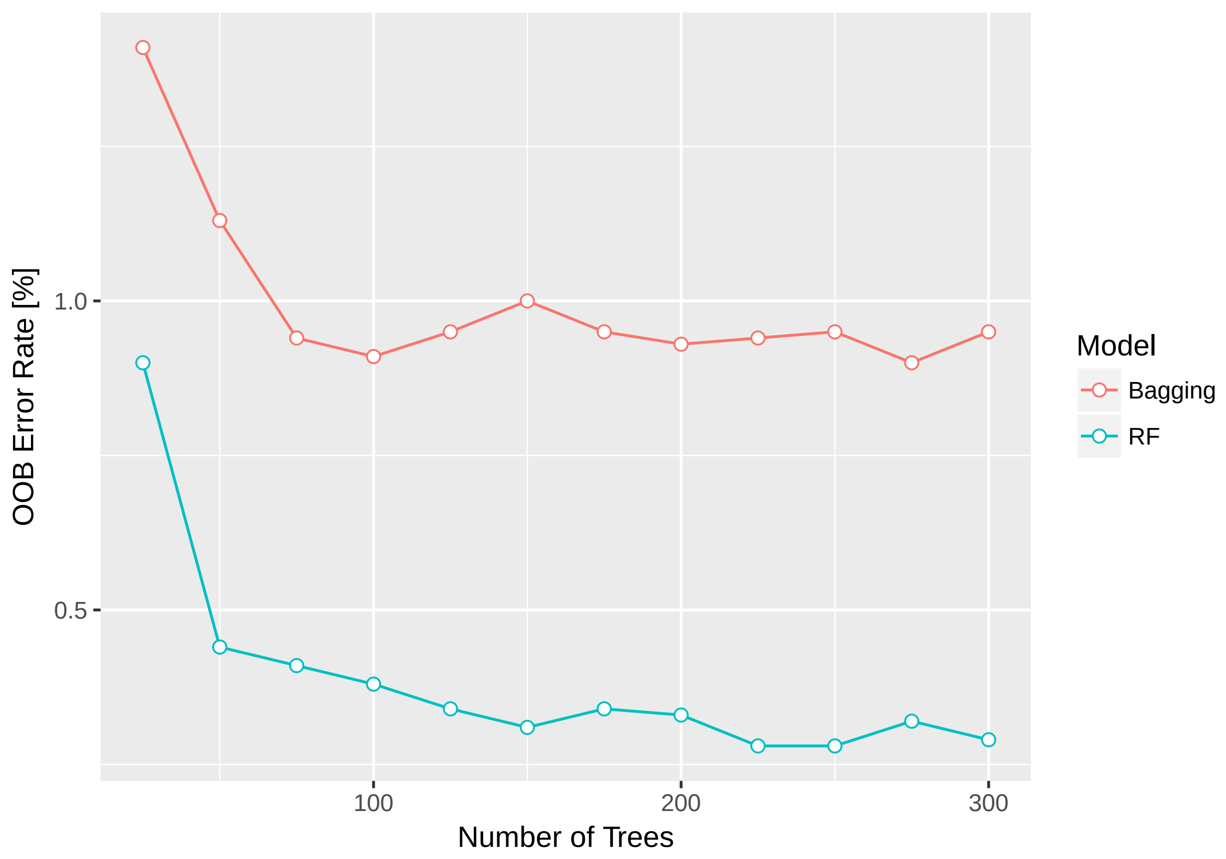
```r
# random forest
rf <- randomForest(classe ~ ., data = df.train, ntree = 300)
```

```

Call:
 randomForest(formula = classe ~ ., data = df.train, ntree = 300)
               Type of random forest: classification
                     Number of trees: 300
No. of variables tried at each split: 7

        OOB estimate of  error rate: 0.3%
Confusion matrix:
     A    B    C    D    E  class.error
A 5578    2    0    0    0 0.0003584229
B   14 3780    3    0    0 0.0044772189
C    0   11 3409    2    0 0.0037989480
D    0    0   20 3194    2 0.0068407960
E    0    0    0    4 3603 0.0011089548
```

Just like in the case of bagging, the number of trees selected must be large enough for the OOB error to have flattened. In our case 300 trees seems sufficient.



Boosting

Unlike bagging, in which the trees are grown independently from each other, boosting grows trees sequentially, using information from the previous tree to update the model. The update is accomplished by fitting the residuals instead of the actual observations.

```
bo <- gbm(classe ~ ., data = df.train, distribution="multinomial", n.trees=150,
           interaction.depth=10, cv.folds = 5, shrinkage = 0.02)
```

```
gbm(formula = classe ~ ., distribution = "multinomial", data = df.train,
    n.trees = 150, interaction.depth = 10, shrinkage = 0.02,
    cv.folds = 5)
A gradient boosted model with multinomial loss function.
150 iterations were performed.
The best cross-validation iteration was 150.
There were 52 predictors of which 42 had non-zero influence.
```

In the case of boosted regression trees, it is possible to overfit if the number of trees if **M** is too large, so in this case we need to use cross-validation to select the number of trees **M**. Usually the number of folds for cross-validation is 5-10.

Predictions on Test Data

We will now look at the predictions obtained from every model. For *Bagging* we have:

```r
# prediction on test data
pred.bg <- predict(bg, newdata = df.test)
```

```
 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
```

For *Random Forest* we have:

```r
# prediction on test data
pred.rf <- predict(rf, newdata = df.test)
```

```
 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
```

Finally for *Boosting* we have:

```r
# prediction on test data
pred.bo <- predict(bo, newdata = df.test, type = "response", n.trees = 150)
```

In the case of boosting, further processing is needed since the result of the prediction is a *probability matrix*, which assigns to each obervation the probability of belonging to each one of the five classes.

```r
p.pred.bo <- apply(pred.bo, 1, which.max)
colnames(pred.bo)[p.pred.bo]
```

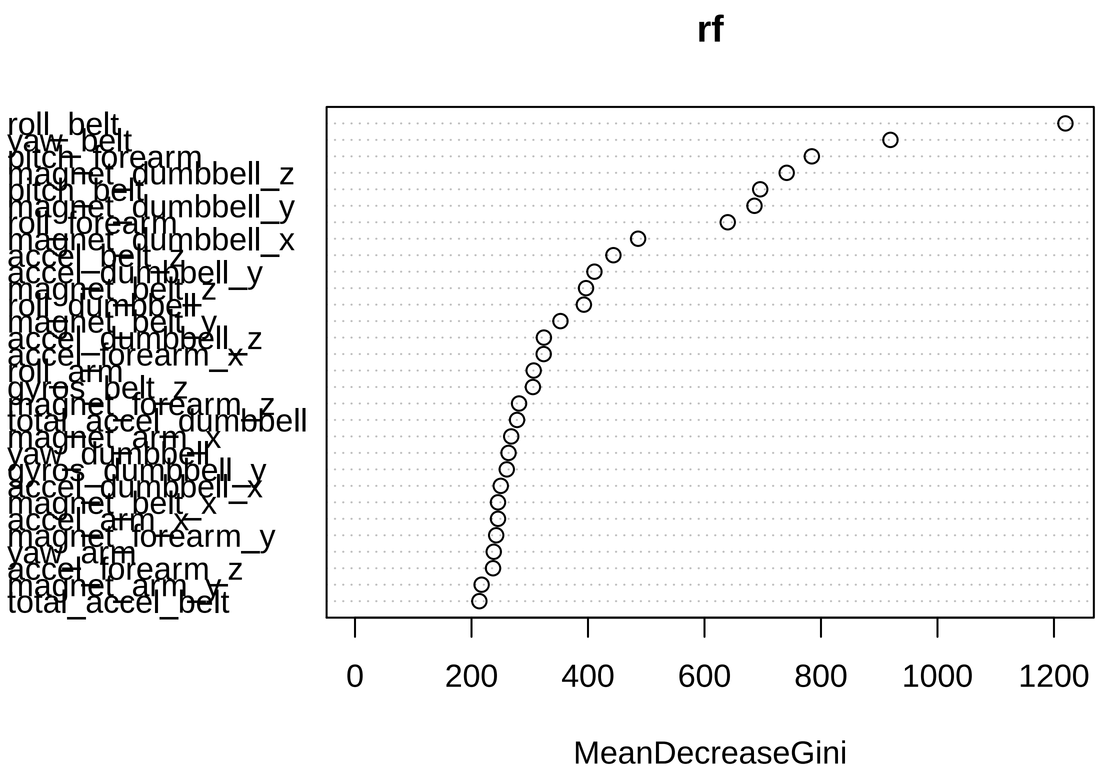
```
 [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A"
[18] "B" "B" "B"
```

We can see that all three models make the same predictions.

Variable Importance

The improvement in accuracy obtained with any of the methods decribed above (over that of single decision tree), comes at the expense of interpretability. A remedy to this problem can be obtained by calculating the *variable importance* of each predictor according to the average reduction in the *Gini Index* (in the case of classification) by splits over such predictor. Let's now plot the variable importance using our random forest model:

```r
varImpPlot(rf)
```



It is clear from the plot, that the two most important variables in predicting the correct class of the repetitions are *roll_belt* and *yaw_belt*.