Practical Machine Learning Course Project

Description

In this project, we use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The aim is to predict the manner in which the participants did the exercise. The 5 possible methods include:

- A - exactly according to the specification
- B - throwing the elbows to the front
- C - lifting the dumbbell only halfway
- D - lowering the dumbbell only halfway
- E - throwing the hips to the front

Data Set

Setting session, clearing space, loading necessary libraries

We first set the working directory, then clear the memory as a good practice before starting the analysis. After which, we load the necessary libraries for analysis.

```
setwd("~/Coursera/PracMachineLearning")
rm(list = ls())
library(knitr)
library(lattice)
library(ggplot2)
library(caret)
library(survival)
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
library(plyr)
library(corrplot)
library(rpart)
library(rpart.plot)
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
```

```
library(parallel)
library(splines)
library(gbm)
```

```
## Loaded gbm 2.1.1
```

Loading train and test data

Then we load the train and the test data by replacing all invalid fields as NA

```
Train_data <- read.csv("./pml-training.csv", na.strings = c("NA", "#DIV/0!",
    ""))
Test_data <- read.csv("./pml-testing.csv", na.strings = c("NA", "#DIV/0!", ""))
```

Cleaning data

Next, we explore the train and test data to get an idea about it using head, str and summary commands. The results produced using these commands are ignored here. It is inferred that there are 160 variables. The missing values, the first seven coulumns and the near zero variance variables are removed.

```
Train_data <- Train_data[, colSums(is.na(Train_data)) == 0]
Test_data <- Test_data[, colSums(is.na(Test_data)) == 0]
Train_data <- Train_data[, -c(1:7)]
Test_data <- Test_data[, -c(1:7)]
nzv <- nearZeroVar(Train_data, saveMetrics = T)
zero.var.ind <- sum(nzv$nzv)

if ((zero.var.ind > 0)) {
    Train_data <- Train_data[, nzv$nzv == F]
}
```

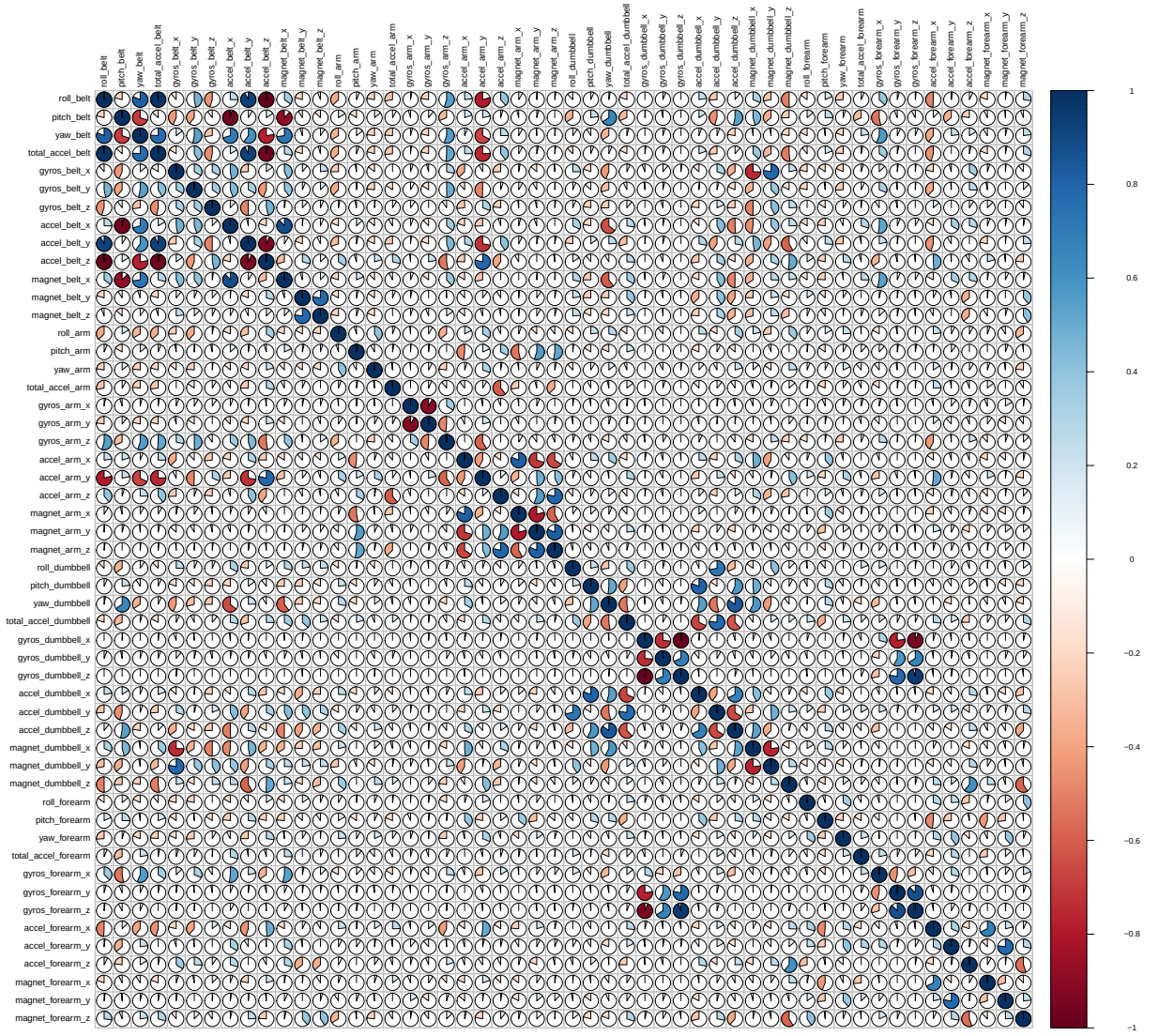
Splitting train set further and creating cross-validation set; a quick correlation analysis

We slice the train data further in to two parts for analysis using

```
set.seed(1234567)
Split_Train <- createDataPartition(Train_data$classe, p = 0.7, list = F)
Train_set <- Train_data[Split_Train, ]
Validate_set <- Train_data[-Split_Train, ]
```

The correlation analysis is performed. And, in the scale in the correlation plot, the further one approaches the end of this scale, the higher is the correlation. Note that, roll_belt is highly correlated to total_accel_belt and accel_belt_z. Similarly, gyros_dumbbell_z has a high correlation with gyros_dumbbell_x.

```
CorrelationMatrix <- cor(Train_set[, -53])
corrplot(CorrelationMatrix, method = "pie", tl.cex = 0.9, tl.col = "black")
```



Building machine learning model, Cross validation and estimating out of sample error

In this section we build two different models, namely Generalized Boosetd Regression Model and Random Forest Model on the train set extracted in the above section, and use them for cross validation. So, in each model the model is trained first and then a cross validation is done. The out of sample error must be small and is estimated using the rest of the probing sample. We expect it to be less than 3% or less.

```r
#Generalized Boosted Regression Model - Training set
gbm_model  <- train(classe ~ ., data=Train_set, method="gbm",
                    trControl = trainControl(method = "cv", number = 10), verbose = F)
pred_gbm <- predict(gbm_model, Train_set)
res_gbm <- confusionMatrix(pred_gbm, Train_set$classe)
confusionMatrix(pred_gbm, Train_set$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3863   72    0    3    5
##          B   32 2544   55   11   12
##          C    7   41 2315   60   14
##          D    3    1   22 2169   27
##          E    1    0    4    9 2467
##
## Overall Statistics
##
##                Accuracy : 0.9724
##                  95% CI : (0.9695, 0.9751)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9651
##  Mcnemar's Test P-Value : 6.036e-13
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9890   0.9571   0.9662   0.9631   0.9770
## Specificity            0.9919   0.9901   0.9892   0.9954   0.9988
## Pos Pred Value         0.9797   0.9586   0.9499   0.9761   0.9944
## Neg Pred Value         0.9956   0.9897   0.9928   0.9928   0.9948
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2812   0.1852   0.1685   0.1579   0.1796
## Detection Prevalence   0.2870   0.1932   0.1774   0.1618   0.1806
## Balanced Accuracy      0.9904   0.9736   0.9777   0.9793   0.9879
```

```r
# Cross validation
cv_gbm <- predict(gbm_model, Validate_set)
cv_res_gbm <- confusionMatrix(cv_gbm, Validate_set$classe)
confusionMatrix(cv_gbm, Validate_set$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1649   37    0    1    1
##          B   13 1069   41    3   16
##          C    5   30  970   33   13
##          D    5    2   14  922    8
##          E    2    1    1    5 1044
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9607
##                  95% CI : (0.9555, 0.9656)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9503
##  Mcnemar's Test P-Value : 6.498e-08
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9851   0.9385   0.9454   0.9564   0.9649
## Specificity            0.9907   0.9846   0.9833   0.9941   0.9981
## Pos Pred Value         0.9769   0.9361   0.9229   0.9695   0.9915
## Neg Pred Value         0.9940   0.9852   0.9884   0.9915   0.9921
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2802   0.1816   0.1648   0.1567   0.1774
## Detection Prevalence   0.2868   0.1941   0.1786   0.1616   0.1789
## Balanced Accuracy      0.9879   0.9616   0.9644   0.9753   0.9815
```

We do the same procedure as above, but adopt random forests in our training model this time. Finally, we compare both the models.

```
#Random Forest - Training set
rand_forest_model <- randomForest(classe ~. , data = Train_set, method = "class")
pred_rand_forest  <- predict(rand_forest_model, Train_set, type = "class")
res_rand_forest    <- confusionMatrix(pred_rand_forest, Train_set$classe)
confusionMatrix(pred_rand_forest, Train_set$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2658    0    0    0
##          C    0    0 2396    0    0
##          D    0    0    0 2252    0
##          E    0    0    0    0 2525
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     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             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```
# Cross validation
cv_rand_forest  <- predict(rand_forest_model, Validate_set, type = "class")
cv_res_rand_forest    <- confusionMatrix(cv_rand_forest, Validate_set$classe)
confusionMatrix(cv_rand_forest, Validate_set$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    4    0    0    0
##          B    0 1130    3    0    0
##          C    0    5 1021    7    0
##          D    0    0    2  957    1
##          E    0    0    0    0 1081
##
## Overall Statistics
##
##                Accuracy : 0.9963
##                  95% CI : (0.9943, 0.9977)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9953
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9921   0.9951   0.9927   0.9991
## Specificity            0.9991   0.9994   0.9975   0.9994   1.0000
## Pos Pred Value         0.9976   0.9974   0.9884   0.9969   1.0000
## Neg Pred Value         1.0000   0.9981   0.9990   0.9986   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1920   0.1735   0.1626   0.1837
## Detection Prevalence   0.2851   0.1925   0.1755   0.1631   0.1837
## Balanced Accuracy      0.9995   0.9957   0.9963   0.9961   0.9995
```

```
#Comparision
res_comparision <- data.frame(res_gbm$overall,res_rand_forest$overall)
res_comparision
```

```
##                res_gbm.overall res_rand_forest.overall
```

```
## Accuracy          9.724103e-01               1.0000000
## Kappa             9.650915e-01               1.0000000
## AccuracyLower     9.695327e-01               0.9997315
## AccuracyUpper     9.750853e-01               1.0000000
## AccuracyNull      2.843416e-01               0.2843416
## AccuracyPValue    0.000000e+00               0.0000000
## McnemarPValue     6.036299e-13                     NaN
```

We see that Random Forest performs better than generalized boosted regression model, since its accuracy, as seen in the comparision result above, is superior than the other model. Also, the estimated out of sample error is far less than expected.

Prediction

The random forest model is finally applied to predict the 20 samples from the remaining test data.

```
predict_test <- predict(rand_forest_model, newdata=Test_data)
predict_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
```