Practical Machine Learning-project

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 relatively inexpensively.

In this project, the goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants (20 - 28 years old) who were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

The set of performances was 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: correct class_ exactly according to the specification (Class A), and 4 other classes corresponding to common mistakes including throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E).

The goal here is to predict the *"class"* with the help of other predictors. This project is a part of Coursera Practical Machine Learning Week 4 - Peer-graded Assignment: Prediction Assignment Writeup.

Data

Loading Data

ALl necessary data (training/testing) were downloaded and save into my local system: destop/PracticeML-data

```
setwd("~/Desktop/PracticeML-data")

d_train <- read.csv("~/Desktop/PracticeML-data/pml-training.csv", stringsAsFactors = F,na.strings = c("
d_test <- read.csv("~/Desktop/PracticeML-data/pml-testing.csv", stringsAsFactors = F,na.strings = c("",

dim(d_train)
```

```
## [1] 19622   160
```

```
dim(d_test)
```

```
## [1]  20 160
```

The training data set contains 19622 observations and 160 variables, while the testing data set contains 20 observations and 160 variables. The "classe" variable in the training set is the outcome to predict.

Data cleaning

```r
d_train<-d_train[,colSums(is.na(d_train)) == 0]
d_test <-d_test[,colSums(is.na(d_test)) == 0]

# Subset data
d_train   <-d_train[,-c(1:7)]
d_test <-d_test[,-c(1:7)]
dim(d_train)
```

```
## [1] 19622    53
```

```r
dim(d_test)
```

```
## [1] 20 53
```

Now, the cleaned training data set contains 19622 observations and 53 variables, while the testing data set contains 20 observations and 53 variables. The "classe" variable is still in the cleaned training set.

Slice data/Cross-validation

```r
# Set seed for reproducability
set.seed(7878)

# cross validation
crval <- caret::createDataPartition(d_train$classe, p = 0.8, list = F)
d_val <- d_train[-crval,]
d_train <- d_train[crval,]
dim(d_train); dim(d_val)
```

```
## [1] 15699    53
```

```
## [1] 3923   53
```

Modeling

ML Algorithm__ Decision tree

Predict with decision tree and output the confusion matrix. It seems like the result of the model is not ideal since the accuracy is 72.5%.

```r
# Fit model
modFitDT <- rpart(classe ~ ., data=d_train, method="class")
# Perform prediction
predictDT <- predict(modFitDT, d_val, type = "class")
table(d_val$classe, predictDT)
```

```
##    predictDT
##        A    B    C    D    E
##   A 1014   28   29   21   24
##   B  187  414   61   51   46
##   C   44   52  506   59   23
##   D   97   35   95  358   58
##   E   30   46   82   26  537
```

```r
confusionMatrix(table(d_val$classe, predictDT))
```

```
## Confusion Matrix and Statistics
##
##    predictDT
##        A    B    C    D    E
##   A 1014   28   29   21   24
##   B  187  414   61   51   46
##   C   44   52  506   59   23
##   D   97   35   95  358   58
##   E   30   46   82   26  537
##
## Overall Statistics
##
##                Accuracy : 0.7211
##                  95% CI : (0.7068, 0.7351)
##     No Information Rate : 0.3497
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6443
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7391   0.7200   0.6546  0.69515   0.7805
## Specificity            0.9600   0.8970   0.9435  0.91637   0.9431
## Pos Pred Value         0.9086   0.5455   0.7398  0.55677   0.7448
## Neg Pred Value         0.8725   0.9491   0.9176  0.95213   0.9528
## Prevalence             0.3497   0.1466   0.1970  0.13128   0.1754
## Detection Rate         0.2585   0.1055   0.1290  0.09126   0.1369
## Detection Prevalence   0.2845   0.1935   0.1744  0.16391   0.1838
## Balanced Accuracy      0.8495   0.8085   0.7990  0.80576   0.8618
```

ML Algorithm__ Random forest

```r
modFitRF <- randomForest(as.factor(classe) ~ ., data=d_train, method="class")
predictRF <- predict(modFitRF, d_val, type = "class")
```

Following confusion matrix shows the errors of the prediction algorithm.

```r
confusionMatrix(table(d_val$classe, predictRF))
```

```
## Confusion Matrix and Statistics
##
##    predictRF
##        A    B    C    D    E
##   A 1116    0    0    0    0
##   B    3  755    1    0    0
##   C    0    0  684    0    0
##   D    0    0   10  633    0
##   E    0    0    2    3  716
##
## Overall Statistics
##
##                Accuracy : 0.9952
##                  95% CI : (0.9924, 0.9971)
##     No Information Rate : 0.2852
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9939
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9973   1.0000   0.9813   0.9953   1.0000
## Specificity            1.0000   0.9987   1.0000   0.9970   0.9984
## Pos Pred Value         1.0000   0.9947   1.0000   0.9844   0.9931
## Neg Pred Value         0.9989   1.0000   0.9960   0.9991   1.0000
## Prevalence             0.2852   0.1925   0.1777   0.1621   0.1825
## Detection Rate         0.2845   0.1925   0.1744   0.1614   0.1825
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9987   0.9994   0.9907   0.9961   0.9992
```

So, the estimated accuracy of the model is 99.54%.

Predicting on Test dataset

```r
result <- predict(modFitRF, d_test[, -length(names(d_test))])
result
```

```
##  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
```

Conclusion

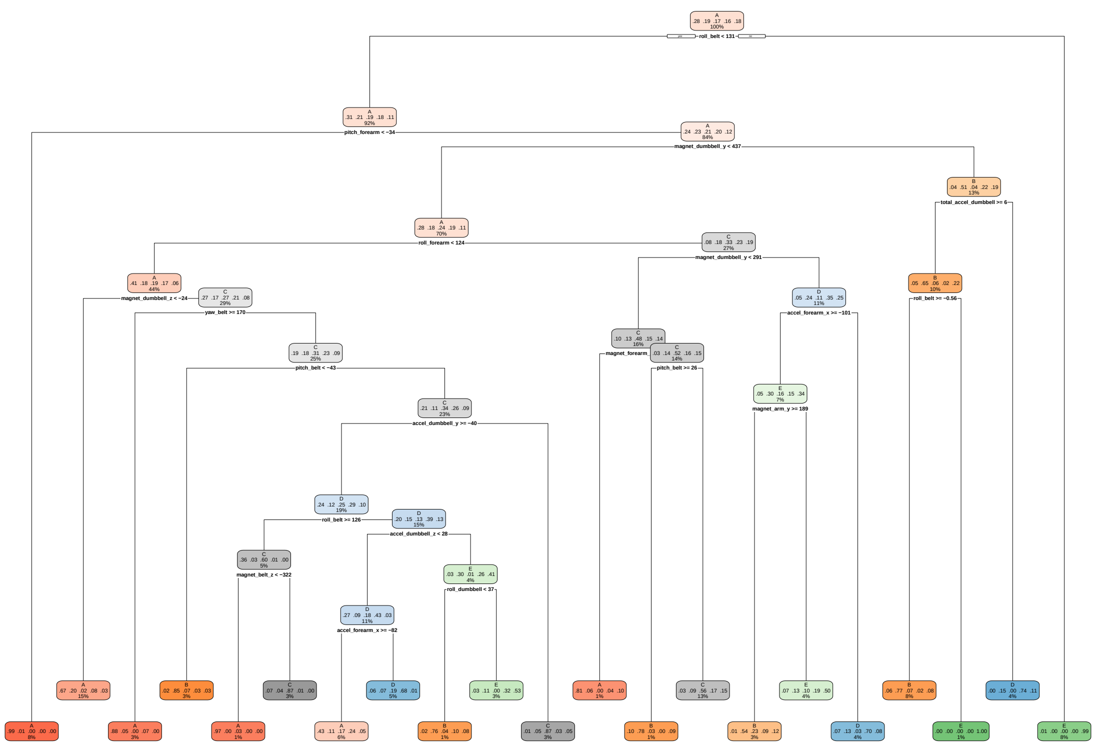
Result:

The confusion matrices show, that the Random Forest algorithm performens better than decision trees. The accuracy for the Random Forest model was 0.9954 (95% CI : (0.9928, 0.9973)) compared to 0.725 (95% CI : (0.7107, 0.7389)) for Decision Tree model. The random Forest model is choosen.

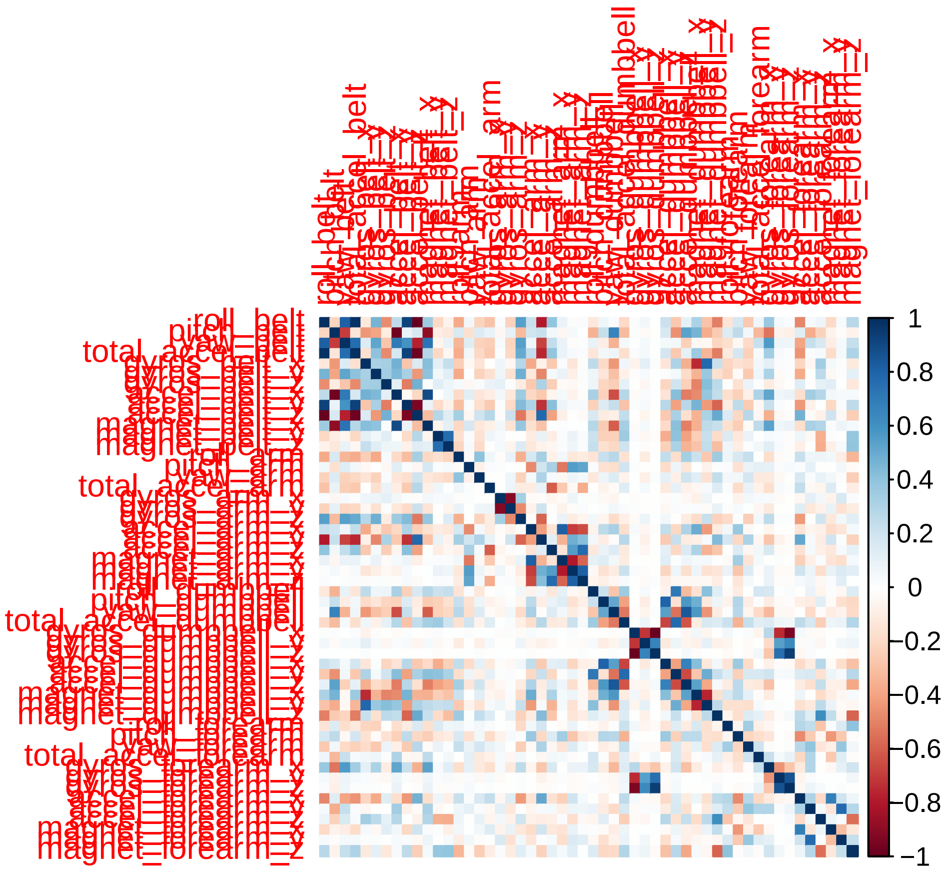
Appendix: Figures

```r
# Plot result_ Decision tree
rpart.plot(modFitDT)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



```r
# Plot Correlation Matrix
corrPlot <- cor(d_train[, -length(names(d_train))])
corrplot(corrPlot, method="color")
```


roll_belt
magnet_forearm_z
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1