Practical Machine Learning Project Report

Dheeraj Snngh

Synopsis of the project

According to documentation available on website, Using data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants who were asked to perform barbell lifts correctly and incorrectly in 5 different ways, we will attempt to predict the way, they did the exercise.

In this project, we will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they did the exercise.

Loading the packages

```
library(data.table)
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
library(corrplot)
library(ggplot2)
```

Download the Data

```
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", "training.csv")
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", "testing.csv")
TrainDataRaw <- read.csv("training.csv")
TestDataRaw <- read.csv("testing.csv")
dim(TrainDataRaw)
```

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

```
dim(TestDataRaw)
```

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

Using data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants who were asked to perform barbell lifts correctly and incorrectly in 5 different ways, we will predict the manner in which they did the exercise. We have recieved above details from document provided on the website.

Data Processing

In this step, we will clean the data and get rid of observations with missing values as well as some meaningless variables.

We see that both the test and training data sets have the same column dimensions, with only the last column differing in name. For our training data set the last column is the "classe" variable, which is the variable that predicts the manner in which the participants do excercise. From the dataset documentation, we get that five different fashions of activity are: exactly according to the specification (Class A), 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). For our testing set the last column is a problem id.

In the plot, we can see that most activities are classified in class "A", which is performing the activity exactly as specified.

```
sum(complete.cases(TrainDataRaw))
```

```
## [1] 406
```

In first, we remove columns that contain NA missing values.

```
TrainDataRaw <- TrainDataRaw[, colSums(is.na(TrainDataRaw)) == 0]
TestDataRaw <- TestDataRaw[, colSums(is.na(TestDataRaw)) == 0]
```

Next, we get rid of some columns that do not contribute much to the accelerometer measurements.

```
classe <- TrainDataRaw$classe
trainRemove <- grepl("^X|timestamp|window", names(TrainDataRaw))
TrainDataRaw <- TrainDataRaw[, !trainRemove]
trainCleaned <- TrainDataRaw[, sapply(TrainDataRaw, is.numeric)]
trainCleaned$classe <- classe
testRemove <- grepl("^X|timestamp|window", names(TestDataRaw))
TestDataRaw <- TestDataRaw[, !testRemove]
testCleaned <- TestDataRaw[, sapply(TestDataRaw, is.numeric)]
```

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.

Creating Training and validation dataset

Then, we can split, using caret pachage, the cleaned training set into a training data set (70%) and a validation data set (30%). We will use the validation data set to conduct cross validation in future steps.

```
set.seed(22519) # For reproducibile purpose
inTrain <- createDataPartition(trainCleaned$classe, p=0.70, list=F)
trainData <- trainCleaned[inTrain, ]
testData <- trainCleaned[-inTrain, ]
```

Data Modeling and Cross validation

We fit a predictive model for activity recognition using **Random Forest** algorithm because it automatically selects important variables and is robust to correlated covariates & outliers in general. We will use **5-fold cross validation** when applying the algorithm.

```
controlRf <- trainControl(method="cv", 5)
modelRf <- train(classe ~ ., data=trainData, method="rf", trControl=controlRf, ntree=250)
modelRf
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10991, 10988, 10989, 10991
## Resampling results across tuning parameters:
```

```
##
##   mtry  Accuracy   Kappa
##    2    0.9909729  0.9885802
##   27    0.9914091  0.9891325
##   52    0.9849311  0.9809363
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

Then, we calculating the performance of the model on the validation data set.

```
predictRf <- predict(modelRf, testData)
confusionMatrix(testData$classe, predictRf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    0    0    0    1
##          B    6 1129    4    0    0
##          C    0    0 1021    5    0
##          D    0    0   14  949    1
##          E    0    0    0    6 1076
##
## Overall Statistics
##
##                Accuracy : 0.9937
##                  95% CI : (0.9913, 0.9956)
##     No Information Rate : 0.2853
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.992
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   1.0000   0.9827   0.9885   0.9981
## Specificity            0.9998   0.9979   0.9990   0.9970   0.9988
## Pos Pred Value         0.9994   0.9912   0.9951   0.9844   0.9945
## Neg Pred Value         0.9986   1.0000   0.9963   0.9978   0.9996
## Prevalence             0.2853   0.1918   0.1766   0.1631   0.1832
## Detection Rate         0.2843   0.1918   0.1735   0.1613   0.1828
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9981   0.9989   0.9908   0.9927   0.9984
```

Calculating Out of Sample Error

We know the accuracy from above, so let's calculate the sample error below.

```
accuracy <- postResample(predictRf, testData$classe)
accuracy
```

```
##  Accuracy     Kappa
```

```
## 0.9935429 0.9918320
```

```
oose <- 1 - as.numeric(confusionMatrix(testData$classe, predictRf)$overall[1])
oose
```

```
## [1] 0.006457094
```

The out of sample error is always larger that the sample error, so we can easily estimate that the out of sample error will be larger than 0.6%.So, the estimated accuracy of the model is 99.42% and the estimated out-of-sample error is 0.58%.

Prediction of Test Data Set

Now, we apply the model to the original testing data set downloaded from the data source. We remove the `problem_id` column first.

```
result <- predict(modelRf, testCleaned[, -length(names(testCleaned))])
result
```

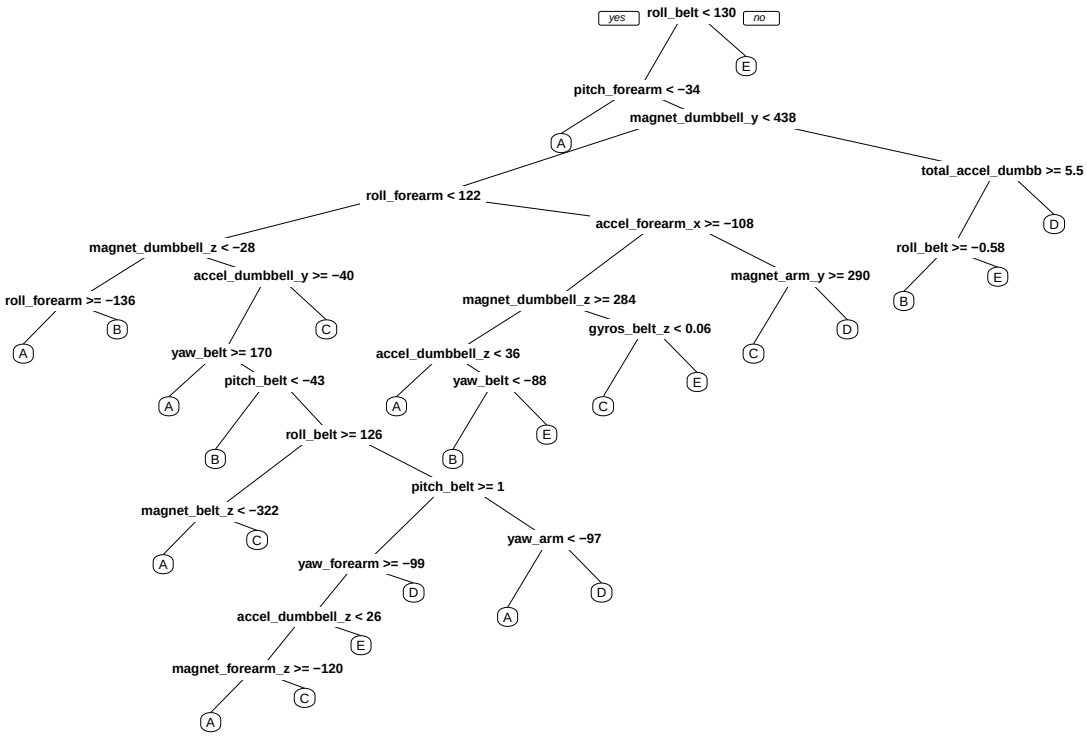
```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

The resulting predictions for the 20 test values were:

Appendix: Figures

1. Decision Tree Visualization

```
treeModel <- rpart(classe ~ ., data=trainData, method="class")
prp(treeModel) # fast plot
```


yes
roll_belt < 130
no
E
pitch_forearm < −34
magnet_dumbbell_y < 438
A
total_accel_dumbb >= 5.5
roll_forearm < 122
accel_forearm_x >= −108
D
magnet_dumbbell_z < −28
roll_belt >= −0.58
accel_dumbbell_y >= −40
magnet_arm_y >= 290
E
roll_forearm >= −136
magnet_dumbbell_z >= 284
B
gyros_belt_z < 0.06
A
B
C
yaw_belt >= 170
accel_dumbbell_z < 36
C
D
pitch_belt < −43
yaw_belt < −88
A
A
E
C
E
roll_belt >= 126
B
B
pitch_belt >= 1
magnet_belt_z < −322
yaw_arm < −97
A
C
yaw_forearm >= −99
A
D
D
accel_dumbbell_z < 26
E
magnet_forearm_z >= −120
C
A