Project

PRACTICAL MACHINE LEARNING COURSE PROJECT

20th February, 2015

SYNOPSIS

This report attempted to predict how well participants perform personal activity using devices such as Jawbone Up, Nike FuelBand, and Fitbit in order to improve their health.

Source of data

Data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants wre collected by asking the participants to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Source of training data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

Source of testing data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Prepare reproducible results

```
# load the packages
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

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

```
set.seed(12345) # set the seed
```

Process the data

To reduce the noise, empty columns were removed from the 2 sets of data. The first 7 columns in each data set were also removed as they were not considered to be relevant to participants' performance.

The cleaned training data was partitioned for model building.

```
# get training data
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))

# get testing data
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))

# remove empty columns and irrelevant columns from training and testing data
training.complete <- training[colnames(training[colSums(is.na(training)) == 0])[-(1:7)]]
testing.complete <- testing[colnames(testing[colSums(is.na(testing)) == 0])[-(1:7)]]

# check whether the data schema of the 2 data sets are the same
all.equal(training.complete[1:length(training.complete)-1], training.complete[1:length(testing.complete
```

```
## [1] TRUE
```

```
# partition training data
i.patition.training <- createDataPartition(y=training.complete$classe, p=0.6, list=FALSE )
training.complete.training <- training.complete[i.patition.training,]
training.complete.testing <- training.complete[-i.patition.training,]
```

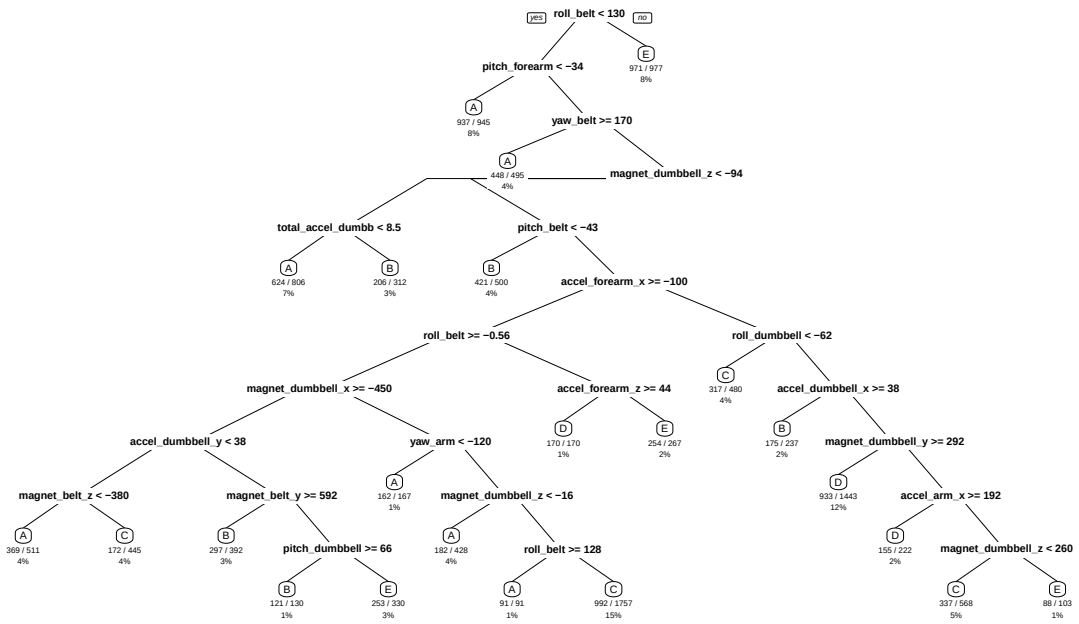
Build the models

Models were built using decision tree and random forest method.

```
# List any data if the variance is near to zero
Check.var <- nearZeroVar(training.complete, saveMetrics=TRUE)
Check.var[Check.var$nzv!=FALSE,]
```

```
## [1] freqRatio     percentUnique zeroVar       nzv
## <0 rows> (or 0-length row.names)
```

```
# Model with decision tree
dtree.model <- rpart(classe ~ ., data=training.complete.training, method="class")
rpart.plot(dtree.model, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```




```
#   Alternative plot for decision tree
## fancyRpartPlot(dtree.model, main="Classification Tree")

# Model with random forest
rforest.model <- randomForest(classe ~. , data=training.complete.training, method="class")
```

Cross validation

The models were tested with the processed testing data set.

```
# Test the decision tree model
dtree.prediction <- predict(dtree.model, training.complete.testing , type = "class")
dtree.cm <- confusionMatrix(dtree.prediction, training.complete.testing$classe)
dtree.cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1879  260   30   69   66
##          B   56  759   88   34   54
##          C  105  340 1226  354  234
##          D  155  132   23  807   57
##          E   37   27    1   22 1031
```

```
##
## Overall Statistics
##
##                Accuracy : 0.7267
##                  95% CI : (0.7167, 0.7366)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6546
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8418  0.50000   0.8962   0.6275   0.7150
## Specificity            0.9243  0.96334   0.8405   0.9441   0.9864
## Pos Pred Value         0.8155  0.76589   0.5427   0.6874   0.9222
## Neg Pred Value         0.9363  0.88928   0.9746   0.9282   0.9389
## Prevalence             0.2845  0.19347   0.1744   0.1639   0.1838
## Detection Rate         0.2395  0.09674   0.1563   0.1029   0.1314
## Detection Prevalence   0.2937  0.12631   0.2879   0.1496   0.1425
## Balanced Accuracy      0.8831  0.73167   0.8684   0.7858   0.8507
```

```
# Test the random forest model
rforest.prediction <- predict(rforest.model, training.complete.testing, type = "class")
rforest.cm <- confusionMatrix(rforest.prediction, training.complete.testing$classe)
rforest.cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2228    9    0    0    0
##          B    4 1504    5    0    0
##          C    0    5 1362   15    3
##          D    0    0    1 1269    4
##          E    0    0    0    2 1435
##
## Overall Statistics
##
##                Accuracy : 0.9939
##                  95% CI : (0.9919, 0.9955)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9923
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9908   0.9956   0.9868   0.9951
## Specificity            0.9984   0.9986   0.9964   0.9992   0.9997
## Pos Pred Value         0.9960   0.9941   0.9834   0.9961   0.9986
```

```
## Neg Pred Value         0.9993   0.9978   0.9991   0.9974   0.9989
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1917   0.1736   0.1617   0.1829
## Detection Prevalence   0.2851   0.1928   0.1765   0.1624   0.1832
## Balanced Accuracy      0.9983   0.9947   0.9960   0.9930   0.9974
```

Out of sample error

The out of sample error was expected to be smaller with the random forest method. 40% of the training data was used to estimate the error, which was expected to be 3% at maximum.

```
# highlight the results
Decision_Tree<-c(dtree.cm$overall[1],1-dtree.cm$overall[1])
Random_forest<-c(rforest.cm$overall[1],1-rforest.cm$overall[1])
results<-rbind(Decision_Tree,Random_forest, deparse.level = 1)
colnames(results)<-c("Accuracy","Sample Error")
results
```

```
##                Accuracy Sample Error
## Decision_Tree 0.7267397  0.273260260
## Random_forest 0.9938822  0.006117767
```

The outcome is satisfactory.

Choose the model

The test showed the random forest model is more accurate.

```
# get the answer
answers <- predict(rforest.model, newdata=testing.complete )
```

```
# get the answer text files
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
pml_write_files(answers)
```

Conclusion

A model was built to evaluate the performance of doing a particular activity. With the measurements as listed in the data set, the performances can be preditced and classified into 5 classes. The error of the prediction model is acceptable.