Machine Learning Course Project

Sinopsis

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. The goal of this project is to predict the manner in which people did the exercise. This is the "classe" variable in the training set.

Data preprocessing

Loading data

Load necessary libraries

```
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
```

Load training and testing data sets

```
train_data <- read.csv("pml-training.csv")
test_data <- read.csv("pml-testing.csv")
```

Checking dimensions of both sets

```
dim(train_data)
```

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

```
dim(test_data)
```

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

Cleaning and partitioning data

Remove near zero variance variables and get rid off personal information which should not be included in model prediction

```
nearzero <- nearZeroVar(train_data, saveMetrics = TRUE)
train_data <- train_data[, !nearzero$nzv]
train_data <- train_data[ , -c(1:6)]
```

Create particions of train data. I will split up train data in 2 parts: 60% for train data and 40 % for test data

```
inTrain <- createDataPartition(y=train_data$classe, p=0.6, list=FALSE)
train <- train_data[inTrain, ]
test  <- train_data[-inTrain, ]
```

Let's look on the data

```
table(train$classe)
```

```
##
##    A    B    C    D    E
## 3348 2279 2054 1930 2165
```

As we can see the each type of exercises has the same order of magnitude. Level A is the most frequent with more than 3000 and level D is less frequent with around 1900.

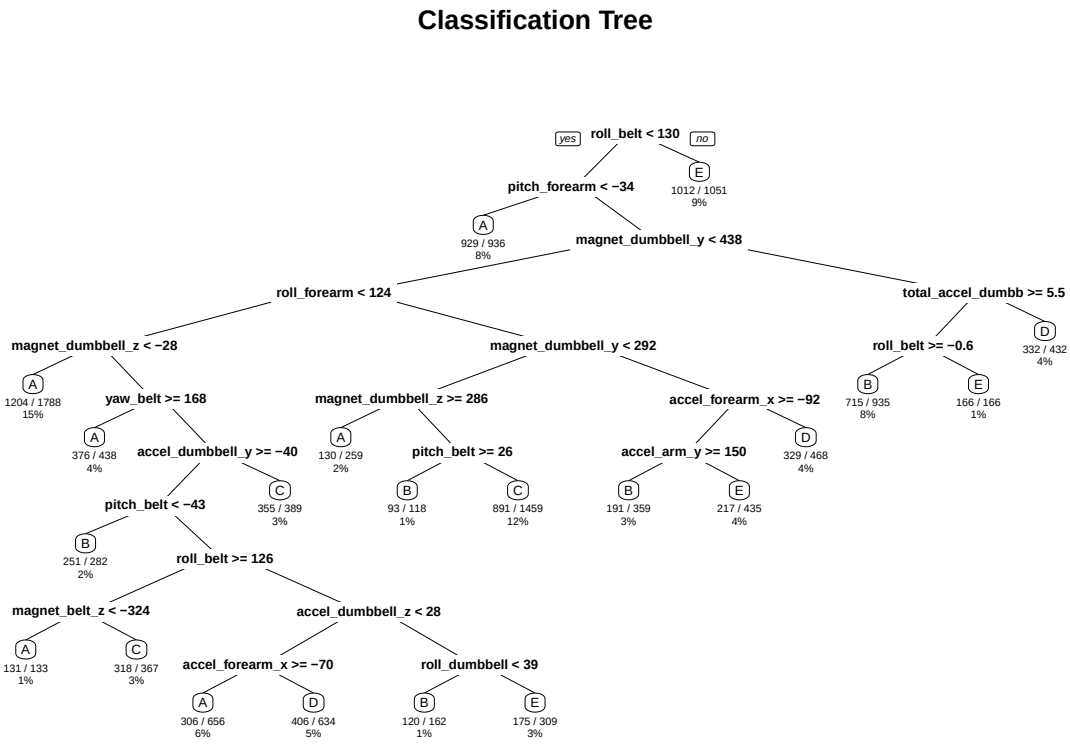
Prediction with trees

Let's use trees model for data prediction

```
modelTree <- rpart(classe ~ ., data=train, method="class")
```

And now build classification tree based on the estimated model:

```
rpart.plot(modelTree, extra = 102, under = TRUE, faclen = 0, main = "Classification Tree")
```



Then we will apply our model on the test part of the training data set:

```
predictionTree <- predict(modelTree, test, type = "class")
```

And then build confusion matrix:

```
confusionMatrix(predictionTree, test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2004  369  102  216   83
##          B   60  863  111   45   99
##          C   49  102 1016  185  138
##          D   55  118   91  705   61
##          E   64   66   48  135 1061
##
## Overall Statistics
##
##                Accuracy : 0.72
##                  95% CI : (0.7099, 0.7299)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6426
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8978   0.5685   0.7427  0.54821   0.7358
## Specificity            0.8628   0.9502   0.9268  0.95046   0.9511
## Pos Pred Value         0.7224   0.7326   0.6819  0.68447   0.7722
## Neg Pred Value         0.9550   0.9018   0.9446  0.91476   0.9411
## Prevalence             0.2845   0.1935   0.1744  0.16391   0.1838
## Detection Rate         0.2554   0.1100   0.1295  0.08985   0.1352
## Detection Prevalence   0.3536   0.1501   0.1899  0.13128   0.1751
## Balanced Accuracy      0.8803   0.7594   0.8348  0.74933   0.8435
```

Prediction with Random Forests

Apply Random Forests model on the trainung dataset

```
modelFitRF <- randomForest(classe ~. , data=train, method="class", na.action=na.roughfix)
```

Use created model for predictions on the test dataset and build confusion matrix:

```
predictionsRF <- predict(modelFitRF, test, type = "class")
confusionMatrix(predictionsRF, test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E
##          A 45  0  0  0  0
##          B  0 34  0  0  0
```

```
##          C  0  1 24  0  0
##          D  0  0  0 30  0
##          E  0  0  0  0 29
##
## Overall Statistics
##
##                Accuracy : 0.9939
##                  95% CI : (0.9663, 0.9998)
##     No Information Rate : 0.2761
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9922
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9714   1.0000    1.000   1.0000
## Specificity            1.0000   1.0000   0.9928    1.000   1.0000
## Pos Pred Value         1.0000   1.0000   0.9600    1.000   1.0000
## Neg Pred Value         1.0000   0.9922   1.0000    1.000   1.0000
## Prevalence             0.2761   0.2147   0.1472    0.184   0.1779
## Detection Rate         0.2761   0.2086   0.1472    0.184   0.1779
## Detection Prevalence   0.2761   0.2086   0.1534    0.184   0.1779
## Balanced Accuracy      1.0000   0.9857   0.9964    1.000   1.0000
```

Conclusions

Based on the investigations above I will select `RANDOM FORESTS` prediction model. As it has much more better accurancy `0.99` (with `95% CONFIDENCE INTERVAL 0.96, 0.9998`) comparing with accurancy `0.74` (with `95% CONFIDENCE INTERVAL 0.7325, 0.7519`) for `PREDICTION TREE` prediction model.

Appendix

Generate files for submission

```r
predictions_testing <- predict(modelFitRF, test_data, type = "class")

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(predictions_testing)
```