Practical Machine Learning Project

MiRo

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. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it.

In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked 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).

Data Preprocessing

```r
packages <- c("data.table", "ggplot2", "dplyr", "scales", "caret", "randomForest", "corrplot", "rpart",
sapply(packages, library, character.only=TRUE, quietly=TRUE)
```

Download the Data

```r
trainUrl <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
trainFile <- "./pml-training.csv"
testFile  <- "./pml-testing.csv"
if (!file.exists(trainFile)) {
  download.file(trainUrl, destfile=trainFile, method="curl")
}
if (!file.exists(testFile)) {
  download.file(testUrl, destfile=testFile, method="curl")
}

trainRaw <- read.csv("./pml-training.csv")
testRaw <- read.csv("./pml-testing.csv")
```

Training set: 19622 observations and 160 variables, Testing set: 20 observations and 160 variables.

Prepare data

We will clean the data and leave only meaning variables.

```r
sum(complete.cases(trainRaw))
```

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

```
nearzero <- nearZeroVar(trainRaw, saveMetrics = TRUE)
train <- trainRaw[, !nearzero$nzv]

rem <- sapply(colnames(train), function(x) if (sum(is.na(train[, x])) > 0.30*nrow(train)) { return(TRUE
train <- train[, !rem]
train <- train[, -(1:6)]

trainRaw <- trainRaw[, colSums(is.na(trainRaw)) == 0]
testRaw <- testRaw[, colSums(is.na(testRaw)) == 0]

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

Dimensions reduced from 60 to 53. We don't want to use particular time/date.

Slice the data

Spliting the cleaned training set into a pure training data set (70%) and a validation data set (30%).

```
set.seed(10001)
inTrain <- createDataPartition(trainCleaned$classe, p=0.70, list=F)
trainData <- trainCleaned[inTrain, ]
testData <- trainCleaned[-inTrain, ]
```

Correlation

```
corr <- findCorrelation(cor(train[, -53]), cutoff=0.8)
names(train)[corr]
```

```
##  [1] "accel_belt_z"     "roll_belt"        "accel_belt_y"
##  [4] "accel_dumbbell_z" "accel_belt_x"     "pitch_belt"
##  [7] "accel_arm_x"      "accel_dumbbell_x" "magnet_arm_y"
## [10] "gyros_arm_y"      "gyros_forearm_z"  "gyros_dumbbell_x"
```

Many variables are highly correlated. PCA will be used in the pre-processing.

Applying ML algorithm

Random Forest algorithm will be used to predict the results. It automatically selects important variables and is robust to correlated covariates. **5-fold cross validation** will be used when applying the algorithm.

```
controlRf <- trainControl(method="cv", number=5, preProcOptions="pca", allowParallel=TRUE)
modelRf <- train(classe ~ ., data=trainData, method="rf", trControl=controlRf, ntree=200)
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: 10990, 10989, 10989, 10991, 10989
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9900997  0.9874743  0.003759171  0.004756127
##   27    0.9920648  0.9899616  0.001774433  0.002245312
##   52    0.9870416  0.9836058  0.002807343  0.003554778
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Performance estimation of the model on the validation data set.

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B   17 1116    5    1    0
##          C    0    5 1014    7    0
##          D    0    0    6  957    1
##          E    0    1    4    1 1076
##
## Overall Statistics
##
##                Accuracy : 0.9918
##                  95% CI : (0.9892, 0.994)
##     No Information Rate : 0.2873
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9897
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9899   0.9947   0.9854   0.9907   0.9991
## Specificity            1.0000   0.9952   0.9975   0.9986   0.9988
## Pos Pred Value         1.0000   0.9798   0.9883   0.9927   0.9945
## Neg Pred Value         0.9960   0.9987   0.9969   0.9982   0.9998
## Prevalence              0.2873   0.1907   0.1749   0.1641   0.1830
## Detection Rate          0.2845   0.1896   0.1723   0.1626   0.1828
## Detection Prevalence    0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy       0.9950   0.9949   0.9915   0.9946   0.9989
```

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

So, the estimated accuracy of the model is 0.9918437, 0.9896801.

Predicting for Test Data Set

Validate our model with test dataset dowloaded above.

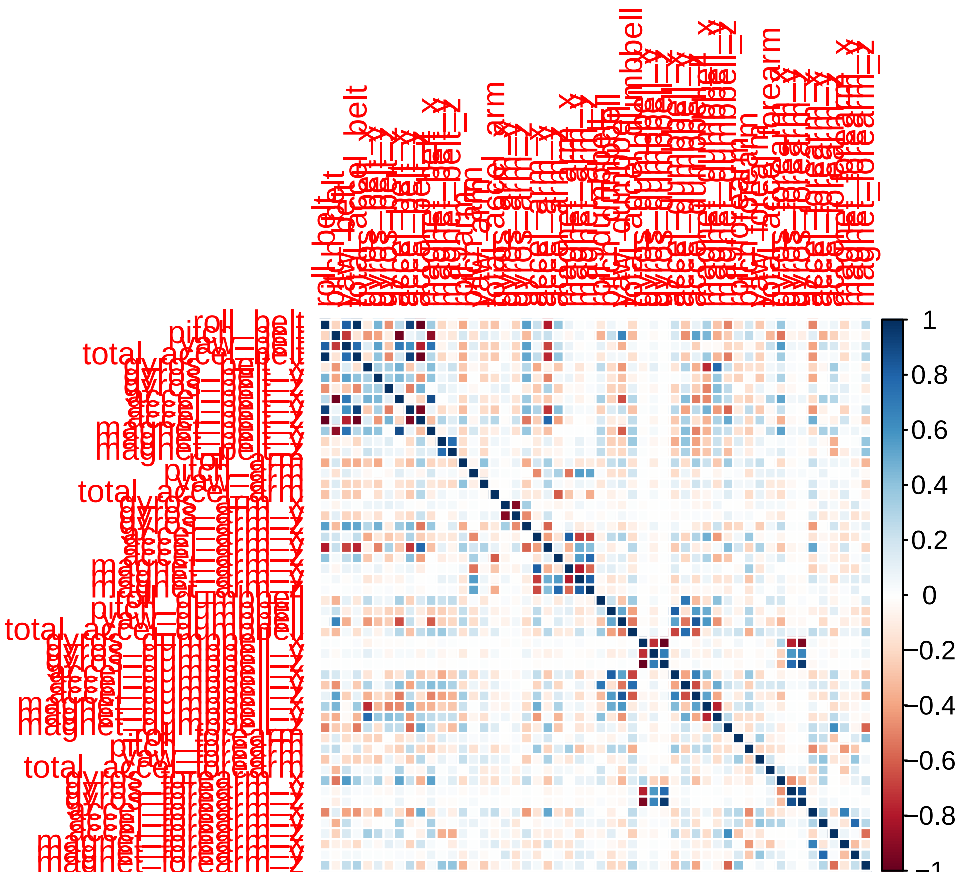
```
result <- predict(modelRf, testCleaned[, -length(names(testCleaned))])
result
```

```
##  [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
```

Appendix: Figures

1. Correlation Matrix Visualization

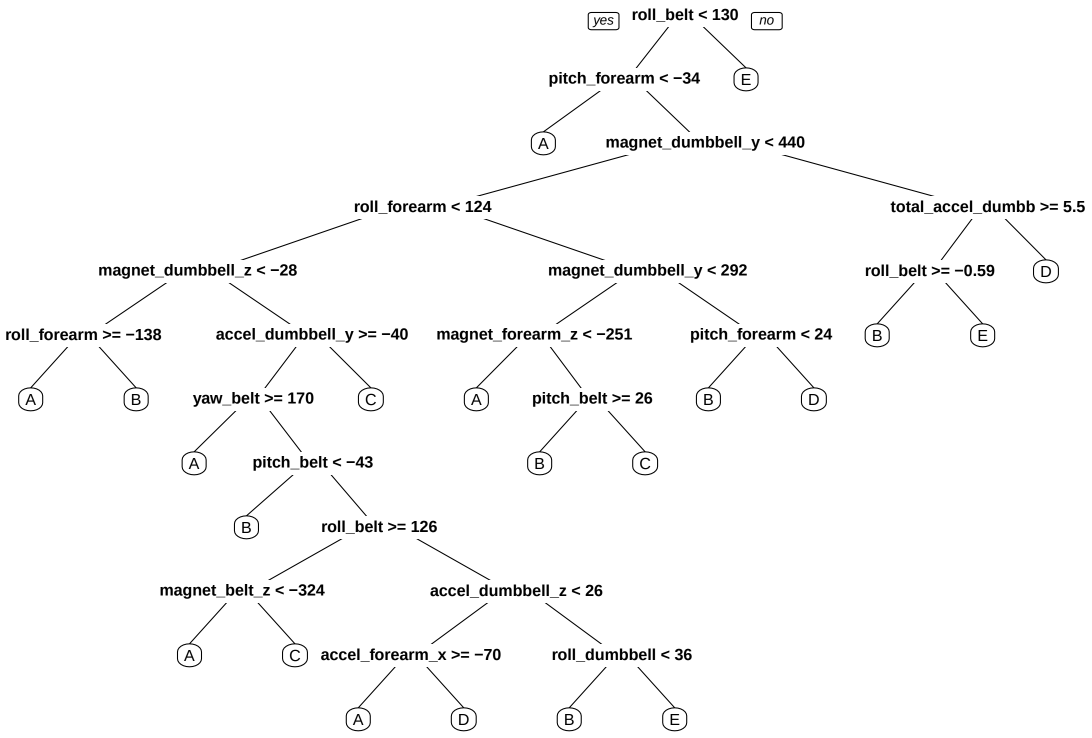
```
corrPlot <- cor(trainData[, -length(names(trainData))])
corrplot(corrPlot, method="color")
```



2. Decision Tree Visual-

ization

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


yes
roll_belt < 130
no
pitch_forearm < −34
E
A
magnet_dumbbell_y < 440
roll_forearm < 124
total_accel_dumbb >= 5.5
magnet_dumbbell_z < −28
magnet_dumbbell_y < 292
roll_belt >= −0.59
D
roll_forearm >= −138
accel_dumbbell_y >= −40
magnet_forearm_z < −251
pitch_forearm < 24
B
E
A
B
yaw_belt >= 170
C
A
pitch_belt >= 26
B
D
A
pitch_belt < −43
B
C
B
roll_belt >= 126
magnet_belt_z < −324
accel_dumbbell_z < 26
A
C
accel_forearm_x >= −70
roll_dumbbell < 36
A
D
B
E