Practical Machine Learning (Project)

TD

Monday, February 16, 2015

Introduction

In this project, we will use the data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to find patterns in their behavior. 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).

Synopsis

In this project, two methods were used to fit our training data set. Cross validation were applied. Based on the model evaluation, it reveals that model using the method of random forest is better than the model using the method of 'rpart' in terms of accuracy of prediction. Finally, the random forest model were applied to predict the new test data set (Test_final).

Load data sets

```
#setwd("E:\\Coursera\\Practical Machine Learning\\project")
#UrlTrain <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
#UrlTest <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
#download.file (UrlTrain, destfile = "training.csv")
#download.file (UrlTest, destfile = "testing.csv")

setwd("E:\\Coursera\\Practical Machine Learning\\project")
setTrain<-read.table(file="training.csv", header = TRUE, sep = ",")
setTest<-read.table(file="testing.csv", header = TRUE, sep = ",")
```

Preprocess data sets

Keep variables that satisfying the following criteria:

1. variables that indicate measures from accelerometers on the belt, forearm, arm, and dumbell
2. no missing values
3. no near zero variables

```
dim(setTrain);dim(setTest)
```

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

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

```r
#select the variables that relates to
#the data from accelerometers on the belt, forearm, arm, and dumbell
library(dplyr)
Train<-select(setTrain, contains("accel"),raw_timestamp_part_1,raw_timestamp_part_2, classe)
names(Train)
```

```
##  [1] "total_accel_belt"     "var_total_accel_belt" "accel_belt_x"
##  [4] "accel_belt_y"         "accel_belt_z"         "total_accel_arm"
##  [7] "var_accel_arm"        "accel_arm_x"          "accel_arm_y"
## [10] "accel_arm_z"          "total_accel_dumbbell" "var_accel_dumbbell"
## [13] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [16] "total_accel_forearm"  "var_accel_forearm"    "accel_forearm_x"
## [19] "accel_forearm_y"      "accel_forearm_z"      "raw_timestamp_part_1"
## [22] "raw_timestamp_part_2" "classe"
```

```r
Test<-select(setTest, contains("accel"),raw_timestamp_part_1,raw_timestamp_part_2)
names(Test)
```

```
##  [1] "total_accel_belt"     "var_total_accel_belt" "accel_belt_x"
##  [4] "accel_belt_y"         "accel_belt_z"         "total_accel_arm"
##  [7] "var_accel_arm"        "accel_arm_x"          "accel_arm_y"
## [10] "accel_arm_z"          "total_accel_dumbbell" "var_accel_dumbbell"
## [13] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [16] "total_accel_forearm"  "var_accel_forearm"    "accel_forearm_x"
## [19] "accel_forearm_y"      "accel_forearm_z"      "raw_timestamp_part_1"
## [22] "raw_timestamp_part_2"
```

```r
#check missing data and keep complete columns
VarMtest<-which(colSums(is.na(Test))>0)
Test_final<-Test[,-c(VarMtest)]
include<- names(Train) %in% c("classe", names(Test[,-c(VarMtest)]))
Train_final <- Train[include]

#check near zero variables
library(caret)
nsv <- nearZeroVar(Train_final, saveMetrics=TRUE)
nsv
```

```
##                      freqRatio percentUnique zeroVar   nzv
## total_accel_belt      1.063160     0.1477933   FALSE FALSE
## accel_belt_x          1.055412     0.8357966   FALSE FALSE
## accel_belt_y          1.113725     0.7287738   FALSE FALSE
## accel_belt_z          1.078767     1.5237998   FALSE FALSE
## total_accel_arm       1.024526     0.3363572   FALSE FALSE
## accel_arm_x           1.017341     3.9598410   FALSE FALSE
## accel_arm_y           1.140187     2.7367241   FALSE FALSE
## accel_arm_z           1.128000     4.0362858   FALSE FALSE
## total_accel_dumbbell  1.072634     0.2191418   FALSE FALSE
## accel_dumbbell_x      1.018018     2.1659362   FALSE FALSE
## accel_dumbbell_y      1.053061     2.3748853   FALSE FALSE
## accel_dumbbell_z      1.133333     2.0894914   FALSE FALSE
## total_accel_forearm   1.128928     0.3567424   FALSE FALSE
```

```
## accel_forearm_x       1.126437     4.0464784   FALSE FALSE
## accel_forearm_y       1.059406     5.1116094   FALSE FALSE
## accel_forearm_z       1.006250     2.9558659   FALSE FALSE
## raw_timestamp_part_1  1.000000     4.2656202   FALSE FALSE
## raw_timestamp_part_2  1.000000    85.5315462   FALSE FALSE
## classe                1.469581     0.0254816   FALSE FALSE
```

```
#double check
dim(Train_final);dim(Test_final)
```

```
## [1] 19622    19
```

```
## [1] 20 18
```

```
sapply(Train_final, function(x) sum(is.na(x) | x == ""))
```

```
##     total_accel_belt         accel_belt_x         accel_belt_y
##                    0                    0                    0
##         accel_belt_z      total_accel_arm          accel_arm_x
##                    0                    0                    0
##          accel_arm_y          accel_arm_z total_accel_dumbbell
##                    0                    0                    0
##     accel_dumbbell_x     accel_dumbbell_y     accel_dumbbell_z
##                    0                    0                    0
##  total_accel_forearm      accel_forearm_x      accel_forearm_y
##                    0                    0                    0
##      accel_forearm_z raw_timestamp_part_1 raw_timestamp_part_2
##                    0                    0                    0
##               classe
##                    0
```

```
sum(!complete.cases(Train_final))
```

```
## [1] 0
```

```
sapply(Test_final, function(x) sum(is.na(x) | x == ""))
```

```
##     total_accel_belt         accel_belt_x         accel_belt_y
##                    0                    0                    0
##         accel_belt_z      total_accel_arm          accel_arm_x
##                    0                    0                    0
##          accel_arm_y          accel_arm_z total_accel_dumbbell
##                    0                    0                    0
##     accel_dumbbell_x     accel_dumbbell_y     accel_dumbbell_z
##                    0                    0                    0
##  total_accel_forearm      accel_forearm_x      accel_forearm_y
##                    0                    0                    0
##      accel_forearm_z raw_timestamp_part_1 raw_timestamp_part_2
##                    0                    0                    0
```

```
sum(!complete.cases(Test_final))
```

```
## [1] 0
```

```
set.seed(1968)
library(caret)
inTrain = createDataPartition(y=Train_final$classe, p=0.6, list=FALSE)
training = Train_final[inTrain,]
testing = Train_final[-inTrain,]
dim(training);dim(testing)
```

Split the 'Train_final' set into 'training' set(60%) and 'testing' set(40%)

```
## [1] 11776    19
```

```
## [1] 7846   19
```

```
summary(training$classe)
```

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

Fit Models

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

model<-train(classe~., method="rpart", data=training)
#print(model,digits=3)
#print(model$finalModel, digits=3)
```

Model1: Predicting with trees using 'rpart'

```
library(randomForest)
model2<- randomForest(classe~., data=training,proximity=TRUE)
#print(model2,digits=3)
```

Model2: Predicting with random forest

Model evaluation

```
predictions<-predict(model, newdata=testing)
confusionMatrix(predictions, testing$classe)
```

Model1 (method="rpart")

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1125  150  175   59   83
##          B  619  883  561  163  428
##          C    0    0  208    0    0
##          D  485  485  424 1064  307
##          E    3    0    0    0  624
##
## Overall Statistics
##
##                Accuracy : 0.4976
##                  95% CI : (0.4865, 0.5087)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3718
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5040   0.5817  0.15205   0.8274  0.43273
## Specificity            0.9168   0.7201  1.00000   0.7407  0.99953
## Pos Pred Value         0.7067   0.3327  1.00000   0.3848  0.99522
## Neg Pred Value         0.8230   0.8777  0.84813   0.9563  0.88669
## Prevalence             0.2845   0.1935  0.17436   0.1639  0.18379
## Detection Rate         0.1434   0.1125  0.02651   0.1356  0.07953
## Detection Prevalence   0.2029   0.3383  0.02651   0.3524  0.07991
## Balanced Accuracy      0.7104   0.6509  0.57602   0.7840  0.71613
```

```
####Model2 (method="randomForest")
predictions2<-predict(model2, newdata=testing, type="class")
confusionMatrix(predictions2, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    3    0    0    0
##          B    1 1513    0    0    0
##          C    0    2 1364    3    0
##          D    0    0    4 1283    2
##          E    0    0    0    0 1440
##
```

```
## Overall Statistics
##
##                Accuracy : 0.9981
##                  95% CI : (0.9968, 0.9989)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9976
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9967   0.9971   0.9977   0.9986
## Specificity            0.9995   0.9998   0.9992   0.9991   1.0000
## Pos Pred Value         0.9987   0.9993   0.9963   0.9953   1.0000
## Neg Pred Value         0.9998   0.9992   0.9994   0.9995   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1928   0.1738   0.1635   0.1835
## Detection Prevalence   0.2847   0.1930   0.1745   0.1643   0.1835
## Balanced Accuracy      0.9995   0.9983   0.9982   0.9984   0.9993
```

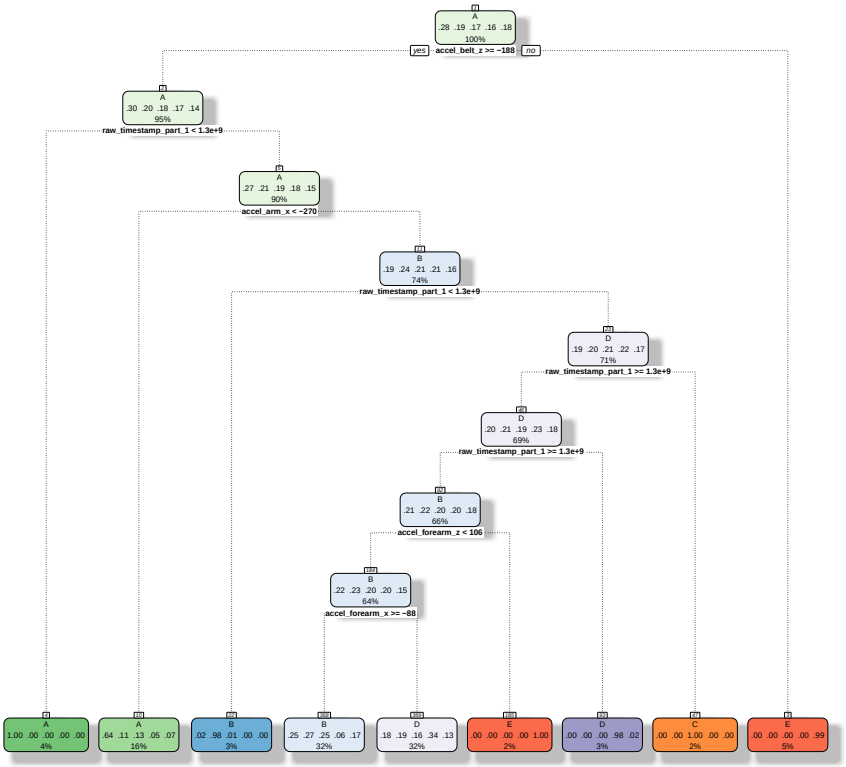
Prediction on the 'Test_final' set

```
predict_new<-predict(model2, newdata=Test_final, type="class")
predict_new
```

```
##  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
```

Appendix: Tree plot for the final model that using the method of 'rpart'

```
fancyRpartPlot(model$finalModel)
```



Rattle 2015−Feb−16 13:38:53 gridA