Correct Exercise Procedure Execution Prediction

dboiani

Practical Machine Learning Peer Assessment

Synopsis: 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 (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Prepare the environment

Install the packages and load the required libraries

```
#Loading and preprocessing the data
#install.packages("caret")
#install.packages("rpart")
#install.packages("randomForest")
#install.packages("rattle")

library(caret)
```

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

```
library(rpart)
library(randomForest)
```

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

```
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.
```

Set the seed for RNG (reproducibility)

```
set.seed(12115) #used current date
```

Load the data

Retrieve and load the data (i.e. read.csv()).

A visual review of the data found NA, DIV/0!, and blanks contained in the csv file. They were replaced with NA when the data was imported.

```
#Retrieve and load the data (i.e. read.csv())

#The training data for this project are available here:
#http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
#The test data are available here:
#http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv
#source:http://groupware.les.inf.puc-rio.br/har

#save the Urls
trainSource <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testSource <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# read the source data
#visually reviewed the data to cleanup NA, DIV/0!, blank data on import
#retrieve the data from the url and store locally

#download.file(trainSource, destfile = "pml-testing.csv")
#download.file(testSource, destfile = "pml-training.csv")

#load the data into data frames
trainData <- read.csv(url(trainSource), na.strings=c("NA","#DIV/0!",""))
#31 seconds
testData <- read.csv(url(testSource), na.strings=c("NA","#DIV/0!",""))
```

The dimensions of the data frames prior to any processing.

```
dim(trainData)
```

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

```
dim(testData)
```

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

Data Processing Section

Goal: Keep the data columns of Interest as predictors for this analysis.

It has already been noted that a visual inspection of the trainData and testData found NA data. Since the testData was available, it was assumed that the columns that had all NA values would not be used as predictors to determine the outcome 'classe', so they were identified and removed from both datasets. If the accuracy of the resulting model had been determined to be unacceptable, this approach would have been reversed and applied instead to the NA columns identified by the trainData dataset.

```
#programmatically collect data on all columns that have all NA values in the testData dataset
id_NA_Cols <- sapply(testData,function(x)any(is.na(x)))
#100 columns were identified

#remove the those 'NA' columns from both datasets
trainData <- trainData[,!(id_NA_Cols)]
testData <- testData[,!(id_NA_Cols)]
```

The visual inspection of the data also identified the attributes "X" and "problem_id" as probable row numbers, so "X" was removed from both datasets, and "problem_id" from the trainData dataset. A few other columns seemed to be unlikely predictor candidates and were later removed, resulting in a decrease in model accuracy. For this report they were not removed. Those columns deserving further study are: user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp, new_window, and num_window.

```
#exclusive of the "classe" outcome to be predicted, remove X from both datasets and problem_id
from the trainData set.
trainData <- trainData[, -1] #X the first column
testData <- testData[, -1] #X the first column
testData <- testData[, -length(colnames(testData))] #problem_id the last column
```

The dimensions of the data frames post processing.

```
dim(trainData)
```

```
## [1] 19622    59
```

```
dim(testData)
```

```
## [1] 20 58
```

The command “nearZeroVar(trainData,saveMetrics=TRUE)” executed against the original “trainData” identified many of the same columns already removed above. The additional column identified, but not removed for this report was: new_window. If the the accuracy of the best model had been deemed unacceptable, this column would have also been removed.

The goal of the project was to predict the manner in which the exercise was completed. This is the “classe” variable in the trainData set. The trainData was partitioned into training (workingTraining) and testing (workingTesting) subsets.

```
#The goal of the project is to predict the manner in which they did the exercise.
#This is the "classe" variable in the training set.
#Partition the trainData into training and testing subsets
#a 60/40 (training/test) partition is recommended for medium datasets. 70/30 and 75/25 were als
o used in the class slides

#library(caret)
inTrain <- createDataPartition(y=trainData$classe, p=0.6, list=FALSE)
workingTraining <- trainData[inTrain,]
workingTesting <- trainData[-inTrain,]
```

The dimensions of the training data frames.

```
dim(workingTraining)
```

```
## [1] 11776    59
```

```
dim(workingTesting)
```

```
## [1] 7846   59
```

Model Building

As a learning exercise every model presented in class was examined. Of course some of the models were not appropriate for this problem. In other cases the selected model could not be completed because of hardware or time limitations. These failures and additional research led to alternatives such as the rpart() and randomForest() functions used to build the models below.

Example of model that did not complete in a reasonable amount of time:
modelrf1 <- train(classe ~ ., data=workingTraining,method=“rf”,prox=TRUE)

This model did not complete after several hours and was stopped.
Entering “warnings()” listed:

In eval(expr, envir, enclos) : model fit failed for Resample06: mtry= 2 Error : cannot allocate vector of size 1.0 Gb

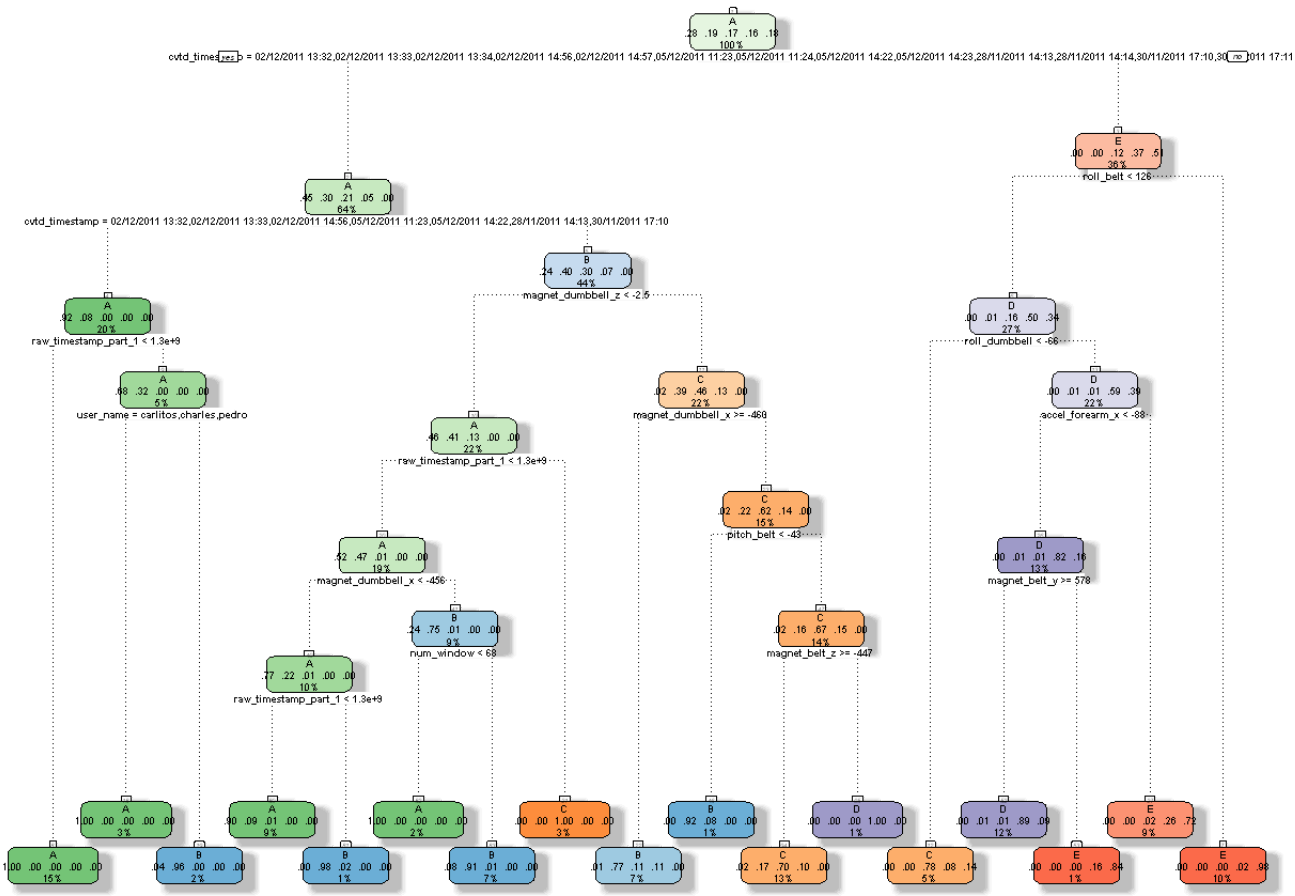
Decision Tree

Using “rpart()” - recursive partitioning for classification,regression and survival trees. The resulting model yielded an accuracy of 87.09%. The out of sample error is expected to be 12.91%.

```
#library(rpart)
modelrpart2 <- rpart(classe ~ ., data=workingTraining, method="class")
```

Plot of the decision tree.

```
#modelrpart2
#plot(modelrpart2)
#text(modelrpart2, use.n=FALSE, all=FALSE, cex=.5)
fancyRpartPlot(modelrpart2) #prettiest
```



Rattle 2015-Jan-23 16:58:30 Family

Prediction and Confusion Matrix

```
#predict
predictrpart2 <- predict(modelrpart2, workingTesting, type="class")

#confusion matrix
confusionMatrix(predictrpart2, workingTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2150   64    9    3    0
##          B   59 1242   71   61    0
##          C   23  203 1265  143   56
##          D    0    9   13  861   71
##          E    0    0   10  218 1315
##
## Overall Statistics
##
##                Accuracy : 0.8709
##                  95% CI : (0.8633, 0.8782)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8366
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9633   0.8182   0.9247   0.6695   0.9119
## Specificity            0.9865   0.9698   0.9344   0.9858   0.9644
## Pos Pred Value         0.9659   0.8667   0.7485   0.9025   0.8522
## Neg Pred Value         0.9854   0.9570   0.9833   0.9383   0.9799
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2740   0.1583   0.1612   0.1097   0.1676
## Detection Prevalence   0.2837   0.1826   0.2154   0.1216   0.1967
## Balanced Accuracy      0.9749   0.8940   0.9296   0.8277   0.9382
```

```
#87% Accuracy
```

Random Forests

Using “randomForest()”. The resulting model yielded an accuracy of 99.81%. The out of sample error is expected to be .19%. Note: ‘train(classe ~ ., data=workingTraing, method=“rf”)’, failed and was not pursued.

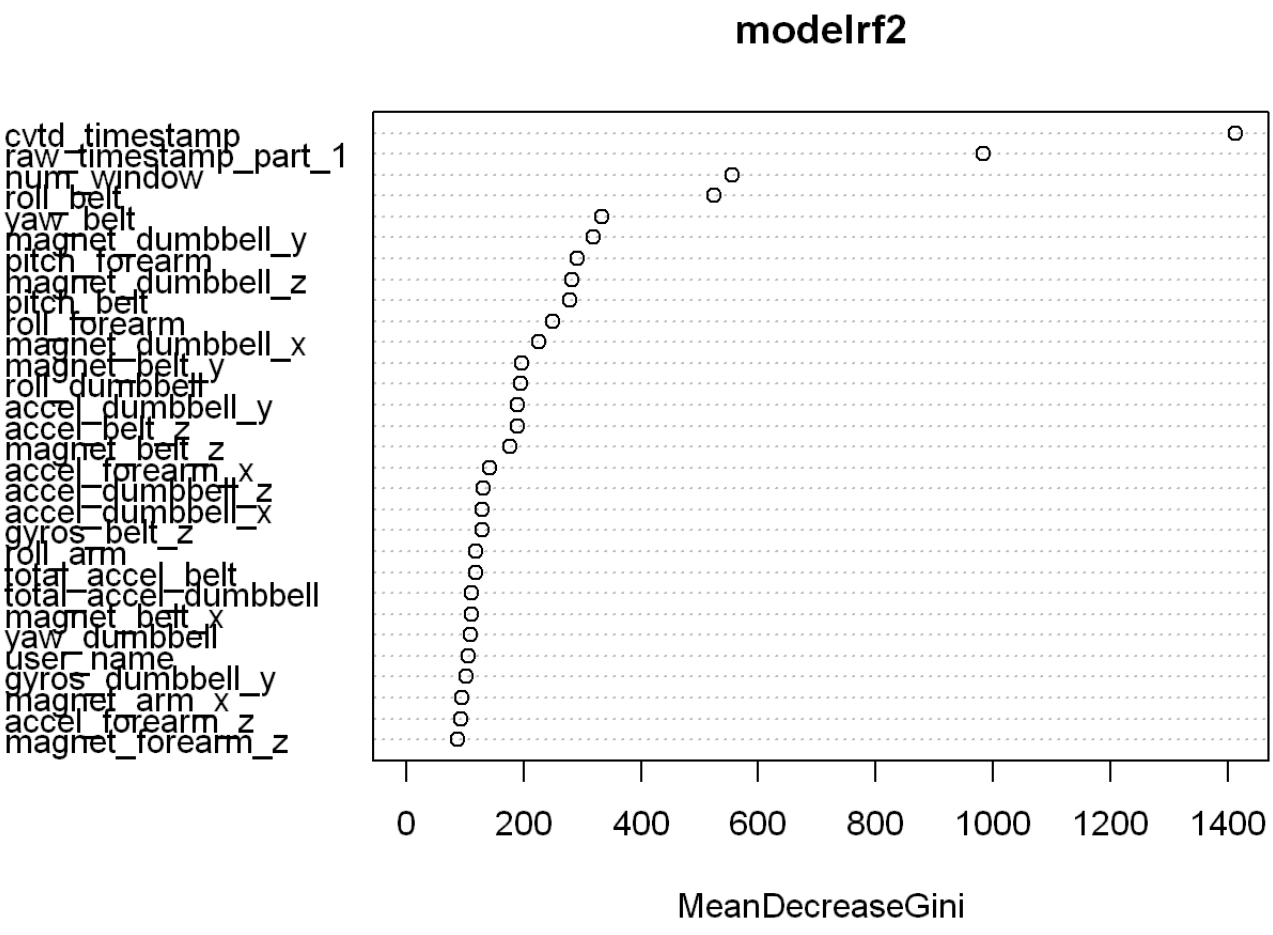
```
#library(randomForest)
modelrf2 <- randomForest(classe ~ ., data=workingTraining)
```

```
print(modelrf2)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = workingTraining)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.13%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3347    1    0    0    0 0.0002986858
## B    2 2277    0    0    0 0.0008775779
## C    0    4 2049    1    0 0.0024342746
## D    0    0    5 1924    1 0.0031088083
## E    0    0    0    1 2164 0.0004618938
```

Importance was examined and plotted. As expected attributes such as ‘cvtd_timestamp’ and ‘raw_timestamp_part_1’ were insignificant in determining the outcome.

```
#importance(modelrf2)
varImpPlot(modelrf2) #cool
```



Prediction and Confusion Matrix

```
#predict
predictrf2 <- predict(modelrf2, workingTesting, type="class")

#confusion matrix
confusionMatrix(predictrf2, workingTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    4    0    0    0
##          B    0 1514    2    0    0
##          C    0    0 1360    3    0
##          D    0    0    6 1283    0
##          E    0    0    0    0 1442
##
## 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            1.0000   0.9974   0.9942   0.9977   1.0000
## Specificity            0.9993   0.9997   0.9995   0.9991   1.0000
## Pos Pred Value         0.9982   0.9987   0.9978   0.9953   1.0000
## Neg Pred Value         1.0000   0.9994   0.9988   0.9995   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1930   0.1733   0.1635   0.1838
## Detection Prevalence   0.2850   0.1932   0.1737   0.1643   0.1838
## Balanced Accuracy      0.9996   0.9985   0.9968   0.9984   1.0000
```

```
#99.8%
```

Apply The Models

The models were applied to the testData. The randomForest() model's (modelrf2) results were submitted for this project and the 20 predictions were successful.

Final model1 using rpart()

```
finalPredictionrpart <- predict(modelrpart2, testData, type="class")
finalPredictionrpart
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  C  A  A  E  D  C  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```

Final model2 using randomForest()

The predict() function unexpectedly produced an error.

predict(modelrf2, testData, type="class")

Error in predict.randomForest(modelrf2, testData, type = "class") : Type of predictors in new data do not match that of the training data. sapply(WorkingTraining, class) & sapply(testData, class)

sapply(workingtraing, class) and sapply(testData, class) indicated differences in data type for some of the same named columns in the two data frames. Several methods were identified to change the data in the testData set to match that of the testingData sets. This code corrects the data types and is simple to read.

```
newFrame <- head(workingTraining,1)
newFrame <- newFrame[, -length(colnames(newFrame))]
fixedtestData <- rbind(newFrame, testData)
fixedtestData <- fixedtestData[-1,]
```

Continuing with the corrected data types.

```
finalPredictionrf2 <- predict(modelrf2, fixedtestData, type="class")
finalPredictionrf2
```

```
## 22  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
##  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
```

Out of Sample Error Discussion

The randomForest model was expected to yield an accuracy of 99.81%, therefore an out of sample error rate of approximately .19% was expected. The model did better than expected against the testingData dataset yielding 100% accuracy. The rpart model was expected to yield an accuracy of 87.09%, therefore an out of sample error rate of approximately 12.91% was expected. This model also did better than expected against

the testingData dataset yielding 90% accuracy. The testingData dataset consisted of 20 observations. Had the testingData dataset been larger, it is likely that the actual accuracy in both cases would have been closer to the calculated accuracies and perhaps lower.

Prediction Assignment Submission

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) } }

setwd(“J:/Practical Machine Learning/Project/answers”)

answers <- finalPredictionrf2 pml_write_files(answers)