PML Project - Prediction of Exercise Manner

Suren G

Friday, January 23, 2015

Introduction

Availability of devices such as Jawbone Up, Nike FuelBand, and Fitbit allows to capture personal activity of physical movements relatively inexpensively. In this analysis, goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and predict the manner of exercise of the participants. The participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

Downloadable copies of the training and test datasets are available at the following links:

Training Datafile URL (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

Test Datafile URL (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Data Source Acknowledgment

The training and test datasets were made available by Groupware@LES (mailto:Groupware@LES) organization.

GroupwareLES URL (http://groupware.les.inf.puc-rio.br/har)

Assumptions

1. Examination of Training and Test case datasets showed they do not have values available for all of the prediction variables. Also the variables which are entirely “NA or blanks or null strings or #DIV/0”" are not exactly common between train and test datasets. For the purpose of this project the variables which have one of the listed values will be excluded, as they should not have any impact on the prediction variable **classe** and will add noise to predict. The list of variables to exclude was driven by the training population, irrespective of the test population, as the training set has more observations and cross validated for model fitness.
2. The training/test HAR data populations contain relevant prediction variables of three categories which are pertinent to “arm”, “belt” or “dumbbell” sensors. Any variables other than the sensor specific ones are excluded (i.e, th first seven variables: “X”, “user_name”, “raw_timestamp_part_1”, “raw_timestamp_part_2”, “cvtd_timestamp”, “new_window”, “num_window”).

```r
#Load requisite libraries
library(knitr, warn.conflicts = FALSE, quietly=TRUE)
library(plyr, warn.conflicts = FALSE, quietly=TRUE)
library(dplyr, warn.conflicts = FALSE, quietly=TRUE)
library(caret, warn.conflicts = FALSE, quietly=TRUE)
library(randomForest, warn.conflicts = FALSE, quietly=TRUE)
library(gbm, warn.conflicts = FALSE, quietly=TRUE)
library(doParallel, warn.conflicts = FALSE, quietly=TRUE)

#Download the requisite train and test data files
opts_chunk$set(echo = TRUE, results = 'asis', fig.path = "./figures/", cache = TRUE)
trainFile <- file.path(".", "pml-training.csv")
download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", trainFile, quiet
 = TRUE)
trainDf <- read.csv(trainFile, header = TRUE, na.strings = c("#DIV/0!", "", "NA", " "))



#Include only variables which have content by excluding exclusive NA values
# and other non sensor variables (e.g. X, user_name etc)
trainDf <- trainDf[ , -c(1:7)]
includeVar <- names(trainDf[ , !(sapply(trainDf, function(x) sum(is.na(x))))])
trainDf <- trainDf[ , c(includeVar)]

testFile <- file.path(".", "pml-testing.csv")
download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", testFile, quiet =
 TRUE)
testDf <- read.csv(testFile, header = TRUE, na.strings = c("#DIV/0!", "")) [ , c(includeVar[1:(length
(includeVar)-1)], "problem_id")]
```

With the above logic the populations have been adjusted to only relevant variables (52) to be used for prediction.

After exclusion of noise predictors the dimensions of downloaded Train and Test sets were pruned as follows respectively:

Train Set Downloaded Population from Data Source (number of observations, variables): 19622, 53

Test Set Downloaded Population from Data Source (number of observations, variables): 20, 53

Prediction Study Design

1. Separate the Training dataset into two separate populations for purposes of Training(60%), Validation/Testing(40%)

2. For prediction purposes fit multiple models/methods. Fit models “gbm”, “rpart” and RandomForest

   (Use RandomForest package function to fit model, as it gave higher speed than ‘rf’ model in caret package)

   - Fit each model against a subset of the training population (60% observations)
   - Validate each model with a subset of the remaining observations (40% observations) created as test set
   - Preprocess with Principal Components Analysis so that a weighted combination of predictors will be taken
   - Prepare PCA variables against the train and test populations and fit a RandomForest model with the PCA
   - Compare difference in the Confidence Interval accuracy to guage best fit model
3. Use the model with highest accuracy, to fit full training population and use it to score 20 test cases

---

```
#Partition the training dataset
partitionIndex <- createDataPartition(trainDf$classe, p = 0.60, list = FALSE)
trainPartition <- trainDf[partitionIndex,]
testPartition <- trainDf[-partitionIndex,]
predictVarCount <- ncol(trainPartition) - 1

#Fit gbm model
modelFit <- train(classe ~., data = trainPartition, method = "gbm", verbose = FALSE)
predictOutcome <- predict(modelFit, newdata = testPartition)
cmGbm <- confusionMatrix(predictOutcome, testPartition$classe)
cmGbm$overall
```

```
  Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
```

9.604894e-01 9.500093e-01 9.559408e-01 9.646918e-01 2.844762e-01 AccuracyPValue McnemarPValue 0.000000e+00 1.893379e-07

```
cmGbm
```

Confusion Matrix and Statistics

```
      Reference
```

Prediction A B C D E A 2198 48 0 1 5 B 20 1429 52 4 11 C 8 35 1292 39 13 D 6 3 20 1229 25 E 0 3 4 13 1388

Overall Statistics

```
          Accuracy : 0.9605
            95% CI : (0.9559, 0.9647)
No Information Rate : 0.2845
P-Value [Acc > NIR] : < 2.2e-16

             Kappa : 0.95
```

Mcnemar’s Test P-Value : 1.893e-07

Statistics by Class:

```
                 Class: A Class: B Class: C Class: D Class: E
```

Sensitivity 0.9848 0.9414 0.9444 0.9557 0.9626 Specificity 0.9904 0.9863 0.9853 0.9918 0.9969 Pos Pred Value 0.9760 0.9426 0.9315 0.9579 0.9858 Neg Pred Value 0.9939 0.9859 0.9882 0.9913 0.9916 Prevalence 0.2845 0.1935 0.1744 0.1639 0.1838 Detection Rate 0.2801 0.1821 0.1647 0.1566 0.1769 Detection Prevalence 0.2870 0.1932 0.1768 0.1635 0.1795 Balanced Accuracy 0.9876 0.9638 0.9649 0.9737 0.9797

```
#Fit rpart model
modelFit <- train(classe ~., data = trainPartition, method = "rpart")
predictOutcome <- predict(modelFit, newdata = testPartition)
cmRpart <- confusionMatrix(predictOutcome, testPartition$classe)
cmRpart$overall
```

```
  Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
 0.4922253      0.3362589      0.4811040      0.5033524      0.2844762
```

AccuracyPValue McnemarPValue 0.0000000 NaN

```
cmRpart
```

Confusion Matrix and Statistics

```
      Reference
```

Prediction A B C D E A 2025 644 647 580 195 B 31 500 48 216 190 C 169 374 673 490 393 D 0 0 0 0 0 E 7 0 0 0 664

Overall Statistics

```
          Accuracy : 0.4922
            95% CI : (0.4811, 0.5034)
No Information Rate : 0.2845
P-Value [Acc > NIR] : < 2.2e-16

             Kappa : 0.3363
```

Mcnemar’s Test P-Value : NA

Statistics by Class:

```
                Class: A Class: B Class: C Class: D Class: E
```

Sensitivity 0.9073 0.32938 0.49196 0.0000 0.46047 Specificity 0.6320 0.92336 0.77987 1.0000 0.99891 Pos Pred Value 0.4950 0.50761 0.32063 NaN 0.98957 Neg Pred Value 0.9449 0.85163 0.87907 0.8361 0.89157 Prevalence 0.2845 0.19347 0.17436 0.1639 0.18379 Detection Rate 0.2581 0.06373 0.08578 0.0000 0.08463 Detection Prevalence 0.5214 0.12554 0.26752 0.0000 0.08552 Balanced Accuracy 0.7696 0.62637 0.63591 0.5000 0.72969

```
#Fit RandomForest model
fitRfModel <- randomForest(trainPartition$classe ~ ., data = trainPartition, importance = TRUE)
predictRfOutcome <- predict(fitRfModel, testPartition)
cmRf <- confusionMatrix(predictRfOutcome, testPartition$classe)
cmRf$overall
```

```
  Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
 0.9943920      0.9929053      0.9924788      0.9959224      0.2844762
```

AccuracyPValue McnemarPValue 0.0000000 NaN

```
cmRf
```

Confusion Matrix and Statistics

```
      Reference
```

Prediction A B C D E A 2232 12 0 0 0 B 0 1506 11 0 0 C 0 0 1355 8 4 D 0 0 2 1278 7 E 0 0 0 0 1431

Overall Statistics

```
          Accuracy : 0.9944
            95% CI : (0.9925, 0.9959)
No Information Rate : 0.2845
P-Value [Acc > NIR] : < 2.2e-16

             Kappa : 0.9929
```

Mcnemar’s Test P-Value : NA

Statistics by Class:

```
                 Class: A Class: B Class: C Class: D Class: E
```

Sensitivity 1.0000 0.9921 0.9905 0.9938 0.9924 Specificity 0.9979 0.9983 0.9981 0.9986 1.0000 Pos Pred Value 0.9947 0.9927 0.9912 0.9930 1.0000 Neg Pred Value 1.0000 0.9981 0.9980 0.9988 0.9983 Prevalence 0.2845 0.1935 0.1744 0.1639 0.1838 Detection Rate 0.2845 0.1919 0.1727 0.1629 0.1824 Detection Prevalence 0.2860 0.1933 0.1742 0.1640 0.1824 Balanced Accuracy 0.9989 0.9952 0.9943 0.9962 0.9962

```
# Random Forests with PCA prep data
pcaPrep <- preProcess(trainPartition[ , 1:predictVarCount], method = "pca")
pcaTrainOutcome <- predict(pcaPrep, trainPartition[ , 1:predictVarCount])
pcaTrainOutcome$classe <- trainPartition$classe
pcaTestOutcome <- predict(pcaPrep, testPartition[ , 1:predictVarCount])
pcaTestOutcome$classe <- testPartition$classe
fitPcaRfModel <- randomForest(pcaTrainOutcome$classe ~ ., data = pcaTrainOutcome, importance = TRUE)
pcaRfOutcome <- predict(fitPcaRfModel, pcaTestOutcome)
cmRfPca <- confusionMatrix(pcaRfOutcome, pcaTestOutcome$classe)
cmRfPca$overall
```

```
  Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
```

9.718328e-01 9.643548e-01 9.679282e-01 9.753813e-01 2.844762e-01 AccuracyPValue McnemarPValue 0.000000e+00 2.652049e-10

```
cmRfPca
```

Confusion Matrix and Statistics

```
      Reference
```

Prediction A B C D E A 2215 32 5 5 5 B 5 1463 29 3 6 C 8 15 1315 51 11 D 3 3 16 1227 15 E 1 5 3 0 1405

Overall Statistics

```
           Accuracy : 0.9718
             95% CI : (0.9679, 0.9754)
No Information Rate : 0.2845
P-Value [Acc > NIR] : < 2.2e-16

              Kappa : 0.9644
```

Mcnemar’s Test P-Value : 2.652e-10

Statistics by Class:

```
                     Class: A Class: B Class: C Class: D Class: E
```

Sensitivity 0.9924 0.9638 0.9613 0.9541 0.9743 Specificity 0.9916 0.9932 0.9869 0.9944 0.9986 Pos Pred Value 0.9792 0.9714 0.9393 0.9707 0.9936 Neg Pred Value 0.9970 0.9913 0.9918 0.9910 0.9942 Prevalence 0.2845 0.1935 0.1744 0.1639 0.1838 Detection Rate 0.2823 0.1865 0.1676 0.1564 0.1791 Detection Prevalence 0.2883 0.1919 0.1784 0.1611 0.1802 Balanced Accuracy 0.9920 0.9785 0.9741 0.9742 0.9865

All the above models were fitted and validated with the two sub-populations (train & test) created from the overall training population (19622 observations) provided from the data source.

The dimensions of Train and Test sub-populations created out of Training population are as follows respectively:

Train Set sub-population (observations, number of variables): 11776, 53

Test Set sub-population (observations, number of variables): 7846, 53

Model Choice for Test Cases Prediction

As noted from above Confusion Matrix results of each model, the RandomForest model has been found to be fitting with highest accuracy. Thus fit a RandomForest model against the complete data population of 19622 rows from the full training dataset and use it for predicting the 20 test cases, which need to be part of the assignment of the project

---

```
#Fit a RandomForest Model with the full training data set and use it for scoring with the test data t
o predict classe
#Use the code provided in the project to write 20 text files for each prediction case
fitRfModel <- randomForest(trainDf$classe ~ ., data = trainDf[, 1:predictVarCount], importance = TRUE
)
answers <- predict(fitRfModel, testDf)

#Print answers for 20 test cases
answers
```

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

```
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)
```

Inference

- RandomForest fit model (Accuracy : 0.992) gave slightly higher accuracy than the gbm fit model (Accuracy : 0.96). The accuracy loss is approximately 0.03%
- RandomForest fit model against PCA preprocessed dataset gave slightly lower accuracy (Accuracy : 0.97) compared to above (Accuracy : 0.992). The accuracy loss is approximately 0.02%
- The rpart method to fit the model gave the lowest accuracy out of the different models used
- RandomForest model is the best algorithm which offered the highest accuracy
- RandomForest model did not overfit and cross validation checks only yielded very low accuracy difference among RandomForest vs gbm vs RandomForest with PCA