Prediction Assignment Writeup

Dumenko Mikhail

24/07/2017

Background

Human activity recognition research has traditionally focused on discriminating between different activities. However, the “how (well)” investigation has only received little attention so far, even though it potentially provides useful information for a large variety of applications,such as sports training (http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har))). For the prediction of how welll individuals performed the assigned exercise six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). This report aims to use machine learning algoritmhs to predict the class of exercise the individuals was performing by using meaurements available from devices such as Jawbone Up, Nike FuelBand, and Fitbit.

Loading and cleaning the data

```
library(caret)
library(rpart)
library(randomForest)
library(e1071)
library(ggplot2)
```

Two data set were available a training set (pml-training.csv) and a test set (pmltesting. csv) for which 20 individuals without any classification for the class of exercise was available.

```
setwd("C:/Users/dumenko/Desktop/coursera/8 - Practical Machine Learning")
```

The data contains mostly numerical features. However, many of them contain nonstandard coded missing values. In addition to the standard NA, there are also empty strings “”, and error expressions “#DIV/0!”. All variables with at least one “NA” were excluded from the analysis.

```
dataTrain<-read.csv("pml-training.csv", header=T, na.strings=c("NA", "#DIV/0!"))
dataTest<-read.csv("pml-testing.csv", header=T, na.string=c("NA", "#DIV/0!"))

dim(dataTrain)
```

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

There are 19622 observation in traning dataset, including 160 variables. The last column is the target variable classe with values A, B, C, D, and E.

```
table(dataTrain$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

Variables related to time and user information were excluded for a total of 51 variables and 19622 class measurements. Several variables are not direcly related to the target variable classe, also removed (“x”, “user_name”, all the time related variables, etc.) Same variables were mainteined in the test dataset to be used for predicting the 20 test cases provided.

```
noNA_dataTrain<-dataTrain[, apply(dataTrain, 2, function(x) !any(is.na(x)))]
dim(noNA_dataTrain)
```

```
## [1] 19622    60
```

```
## User information, time and undefined
clean_Train<-noNA_dataTrain[,-c(1:8)]
dim(clean_Train)
```

```
## [1] 19622    52
```

```
## Validation dataset
clean_Test<-dataTest[,names(clean_Train[,-52])]
dim(clean_Test)
```

```
## [1] 20 51
```

Data Prediction and Modelling

The cleaned dataset was subset in order to generate a test set independent from the 20 cases provided set. Partitioning was performed to obtain a 75% training set and a 25% test set.

```
inTrain <- createDataPartition(y=clean_Train$classe, p=0.75,list=F)
training <-clean_Train[inTrain,]
test <- clean_Train[-inTrain,]
```

Algorithm which will be used for the predictive model here is Random Forest.

```
set.seed(2017)
setting <- trainControl(method="cv", number=5, allowParallel=T, verbose=T)
fit_rf <- train(classe~.,data=training, method="rf", trControl=setting, verbose=F)
```

```
## + Fold1: mtry= 2
## - Fold1: mtry= 2
## + Fold1: mtry=26
## - Fold1: mtry=26
## + Fold1: mtry=51
## - Fold1: mtry=51
## + Fold2: mtry= 2
## - Fold2: mtry= 2
## + Fold2: mtry=26
## - Fold2: mtry=26
## + Fold2: mtry=51
## - Fold2: mtry=51
## + Fold3: mtry= 2
## - Fold3: mtry= 2
## + Fold3: mtry=26
## - Fold3: mtry=26
## + Fold3: mtry=51
## - Fold3: mtry=51
## + Fold4: mtry= 2
## - Fold4: mtry= 2
## + Fold4: mtry=26
## - Fold4: mtry=26
## + Fold4: mtry=51
## - Fold4: mtry=51
## + Fold5: mtry= 2
## - Fold5: mtry= 2
## + Fold5: mtry=26
## - Fold5: mtry=26
## + Fold5: mtry=51
## - Fold5: mtry=51
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 26 on full training set
```

```
fit_rf
```

```
## Random Forest
##
## 14718 samples
##    51 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11775, 11774, 11775, 11775, 11773
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9908276  0.9883960
##   26    0.9915068  0.9892555
##   51    0.9893323  0.9865055
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 26.
```

```
pred_rf <- predict(fit_rf, newdata=test)
confusionMatrix(pred_rf, test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1393    3    0    0    0
##          B    2  946    3    0    0
##          C    0    0  851    8    0
##          D    0    0    1  796    4
##          E    0    0    0    0  897
##
## Overall Statistics
##
##                Accuracy : 0.9957
##                  95% CI : (0.9935, 0.9973)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9946
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9986   0.9968   0.9953   0.9900   0.9956
## Specificity            0.9991   0.9987   0.9980   0.9988   1.0000
## Pos Pred Value         0.9979   0.9947   0.9907   0.9938   1.0000
## Neg Pred Value         0.9994   0.9992   0.9990   0.9981   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2841   0.1929   0.1735   0.1623   0.1829
## Detection Prevalence   0.2847   0.1939   0.1752   0.1633   0.1829
## Balanced Accuracy      0.9989   0.9978   0.9967   0.9944   0.9978
```

Random forest trees were generated for the training dataset using cross-validation. Then the generated algorithm was examnined under the partitioned training set to examine the accuracy and estimated error of prediction. By using 51 predictors for five classes using cross-validation at a 5-fold an accuracy of 99.12% with a 95% CI [0.988-0.994] was achieved accompanied by a Kappa value of 0.99.

```
pred_20 <- predict(fit_rf, newdata=clean_Test)
pred_20
```

```
##  [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
```

A boosting algorithm was also run to confirm and be able to compare predictions. The boosting approach presented less accuracy (96%). However, when the predictions for the 20 test cases were compared match was same for both ran algorimths.

```
setting2 <- trainControl(method="cv", number=5, allowParallel=T, verbose=T)
fit_gbm <- train(classe~.,data=training, method="gbm", trControl=setting2, verbose=F)
```

```
## Loading required package: gbm
```

```
## Warning: package 'gbm' was built under R version 3.4.1
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.3
```

```
## Loading required package: plyr
```

```
## + Fold1: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold1: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold1: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold1: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold1: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold1: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold2: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold2: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold2: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold2: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold2: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold2: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold3: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold3: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold3: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold3: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold3: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold3: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold4: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold4: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold4: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold4: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold4: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold4: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold5: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold5: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold5: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold5: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold5: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold5: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## Aggregating results
## Selecting tuning parameters
## Fitting n.trees = 150, interaction.depth = 3, shrinkage = 0.1, n.minobsinnode = 10 on
full training set
```

```
fit_gbm$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 51 predictors of which 42 had non-zero influence.
```

```
class(fit_gbm)
```

```
## [1] "train"         "train.formula"
```

```
pred_gmb <- predict(fit_gbm, newdata=test)
confusionMatrix(pred_gmb, test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1373   21    1    0    1
##          B   13  903   28    2   14
##          C    7   24  818   24    9
##          D    2    0    5  770   16
##          E    0    1    3    8  861
##
## Overall Statistics
##
##                Accuracy : 0.9635
##                  95% CI : (0.9579, 0.9686)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9538
##  Mcnemar's Test P-Value : 1.096e-05
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9842   0.9515   0.9567   0.9577   0.9556
## Specificity            0.9934   0.9856   0.9842   0.9944   0.9970
## Pos Pred Value         0.9835   0.9406   0.9274   0.9710   0.9863
## Neg Pred Value         0.9937   0.9883   0.9908   0.9917   0.9901
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2800   0.1841   0.1668   0.1570   0.1756
## Detection Prevalence   0.2847   0.1958   0.1799   0.1617   0.1780
## Balanced Accuracy      0.9888   0.9686   0.9705   0.9761   0.9763
```

```
pred_20_gbm <- predict(fit_gbm, newdata=clean_Test)
pred_20_gbm
```

```
##  [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
```