Practical Machine Learning Course Project

Miguel Prata

18/05/2017

Summary

The goal of this project is to analyze data collected from personal activity trackers in order to identify the performed activity using machine learning techniques. Subjects were asked to perform a weight lifting exercise while wearing sensors on the forearm, arm, and belt. The dumbell also had a sensor. The test subjects performed the exercise in five different ways:

- 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).

The original data was collected by E. Velloso et al., more information can be found in http://groupware.les.inf.puc-rio.br/har#ixzz4hQJaWpGI (http://groupware.les.inf.puc-rio.br/har#ixzz4hQJaWpGI).

To speed up the analysis parallel processing was enabled, according to instructions from https://github.com/lgreski/datasciencectacontent/blob/master/markdown/pml-randomForestPerformance.md (https://github.com/lgreski/datasciencectacontent/blob/master/markdown/pml-randomForestPerformance.md).

Data Cleaning

After an initial inspection it became evident that the dataset required some cleaning and columns filled interely with NA values or “” were removed. After splitting the dataset (60% training, 40% testing) the best practices for data pre processing were followed:

- Low variability of the predictors was checked for;
- Highly correlated predictors were identified and removed;

```
library(parallel);
library(doParallel);
library(caret);
library(dplyr);
#Setup parallel processing
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)
#Configure working directory
#setwd('C:/Users/mprata/Desktop/8_Practical_Machine_Learning/project')
setwd('/Users/miguelprata/Dropbox/Minhas/Coursera/Data Science Specialization/8 - P
ractical Machine Learning/Projecto')
#Load data set
data <- tbl_df(read.csv("pml-training.csv",stringsAsFactors=FALSE))
#Remove first seven columns because they are junk - QED
hold <- 1:7;
data <- select(data,-hold)
#Identify columns CONSISTING SOLELY of missing data:
#Columns with NA or "" are flagged  with logical value TRUE and removed
MISS <- sapply(data, function (x) any(is.na(x) | x == ""))
holder <- !MISS
myVariables <- names(MISS)[holder]
data2 <- select(data,one_of(myVariables))
#Convert "classe" column to factor
data2$classe <- as.factor(data2$classe)
#Split data into training and testing sets (60/40)
set.seed(1991)
trainIndex <- createDataPartition(data2$classe,
p = .6, list = FALSE,times = 1)
dataTrain <- data2[trainIndex,]
dataTest  <- data2[-trainIndex,]
#Check for low variability features (there are no low variability features)
variability <- nearZeroVar(data2)
#Check for highly correlated features
descrCorr <-  cor(select(dataTrain,-classe))
highCorr <- findCorrelation(descrCorr, 0.90)
#Highly correlated features found
#Remove these features from the datasets
dataTrain <- dataTrain[, -highCorr]
dataTest  <- dataTest [, -highCorr]
```

This effectivly reduces the number of predictors in the dataset from 152 to 47.

Analysis

10-fold cross validation was setup and used in conjunction with:

- trees;
- random forests;
- bagging;

The trained models were applied to the test sets and its' accuracy assessed with the confusion matrix statistics.

```
#Configure trainControl object to perform cross validation with 10 folds
fitControl <- trainControl(method = "cv",
number = 10, allowParallel = TRUE)
#Train tree algorithm
system.time(tree <- train(classe ~ ., data = dataTrain,
            method = "rpart", trControl = fitControl))
```

```
##    user  system elapsed
##   4.154   0.122  13.536
```

```
#Train random forest algorithm
system.time(rf <- train(classe ~ ., data = dataTrain,
            method = "rf", trControl = fitControl))
```

```
##    user  system elapsed
##  46.747   1.240 772.893
```

```
#Train boosting algorithm
system.time(gbm <- train(classe ~ ., data = dataTrain,
            method = "gbm", trControl = fitControl))
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2127
##      2        1.4786             nan     0.1000    0.1500
##      3        1.3851             nan     0.1000    0.1291
##      4        1.3063             nan     0.1000    0.0956
##      5        1.2475             nan     0.1000    0.0859
##      6        1.1945             nan     0.1000    0.0718
##      7        1.1481             nan     0.1000    0.0703
##      8        1.1033             nan     0.1000    0.0600
##      9        1.0670             nan     0.1000    0.0519
##     10        1.0342             nan     0.1000    0.0503
##     20        0.7999             nan     0.1000    0.0237
##     40        0.5502             nan     0.1000    0.0148
##     60        0.4128             nan     0.1000    0.0063
##     80        0.3278             nan     0.1000    0.0074
##    100        0.2668             nan     0.1000    0.0040
##    120        0.2236             nan     0.1000    0.0023
##    140        0.1928             nan     0.1000    0.0022
##    150        0.1785             nan     0.1000    0.0014
```

```
##    user  system elapsed
##  23.510   0.797 227.810
```

```
#Predict on test dataset
pred_tree <- predict(tree,dataTest)
pred_rf <- predict(rf,dataTest)
pred_gbm <- predict (gbm, dataTest)
```

```
#Generate Confusion Matrixes
confusionMatrix(dataTest$classe,pred_tree)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2020   34  136   41    1
##          B  603  513  344   57    1
##          C  598   44  585  140    1
##          D  492  232  140  323   99
##          E  338  263  329   57  455
##
## Overall Statistics
##
##                Accuracy : 0.4966
##                  95% CI : (0.4854, 0.5077)
##     No Information Rate : 0.5163
##     P-Value [Acc > NIR] : 0.9998
##
##                   Kappa : 0.343
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4986  0.47238  0.38136  0.52265  0.81688
## Specificity            0.9441  0.85133  0.87595  0.86677  0.86459
## Pos Pred Value         0.9050  0.33794  0.42763  0.25117  0.31553
## Neg Pred Value         0.6382  0.90945  0.85350  0.95503  0.98407
## Prevalence             0.5163  0.13841  0.19551  0.07877  0.07099
## Detection Rate         0.2575  0.06538  0.07456  0.04117  0.05799
## Detection Prevalence   0.2845  0.19347  0.17436  0.16391  0.18379
## Balanced Accuracy      0.7214  0.66185  0.62865  0.69471  0.84073
```

```
confusionMatrix(dataTest$classe,pred_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2225    5    0    0    2
##          B   15 1490   11    0    2
##          C    0    9 1351    8    0
##          D    0    1   17 1268    0
##          E    0    0    6    4 1432
##
## Overall Statistics
##
##                Accuracy : 0.9898
##                  95% CI : (0.9873, 0.9919)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9871
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9933   0.9900   0.9755   0.9906   0.9972
## Specificity            0.9988   0.9956   0.9974   0.9973   0.9984
## Pos Pred Value         0.9969   0.9816   0.9876   0.9860   0.9931
## Neg Pred Value         0.9973   0.9976   0.9948   0.9982   0.9994
## Prevalence             0.2855   0.1918   0.1765   0.1631   0.1830
## Detection Rate         0.2836   0.1899   0.1722   0.1616   0.1825
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9960   0.9928   0.9864   0.9939   0.9978
```

```
confusionMatrix(dataTest$classe,pred_gbm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2194   25    8    2    3
##          B   38 1429   47    1    3
##          C    0   62 1274   27    5
##          D    0    2   39 1236    9
##          E    6   14   20   26 1376
##
## Overall Statistics
##
##                Accuracy : 0.957
##                  95% CI : (0.9523, 0.9614)
##     No Information Rate : 0.2852
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9457
##  Mcnemar's Test P-Value : 5.775e-06
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9803   0.9328   0.9179   0.9567   0.9857
## Specificity            0.9932   0.9859   0.9854   0.9924   0.9898
## Pos Pred Value         0.9830   0.9414   0.9313   0.9611   0.9542
## Neg Pred Value         0.9922   0.9837   0.9824   0.9915   0.9969
## Prevalence             0.2852   0.1953   0.1769   0.1647   0.1779
## Detection Rate         0.2796   0.1821   0.1624   0.1575   0.1754
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9868   0.9593   0.9517   0.9745   0.9877
```

Conclusion

Using random forests yielded the best results (estimated error rate smaller than 1%), as such this method was used to answer the Prediction Quiz.

```
dataQ <- tbl_df(read.csv("pml-testing.csv",stringsAsFactors=FALSE))
results <- predict(rf,dataQ)
results
```

```
##  [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
```