08 Practical Machine Learning - Project Assignment.

Author: vkoretsky | Date: 10-25-2014

Executive summary:

The project goal is to build a prediction model on a set of Human Activity Recognition (HAR) data. The data comes from the following source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

Dataset description:

HAR training data set: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv) HAR testing data set: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv) Data set consists of various measurements (3-D acceleration at various points) on subjects performing various types of activities (walking, standing, standing up, sitting, sitting down). The outcome variable is classe. Because alorigthm model selection is the main point of this exercise, I omitted downloading of data files from this script (on purpose).

Exploratory Data Analysis:

The training dataset is comprised of 160 columns with 19622 observations. (** Appendix A1 **) There is a fair amount of columns with a high proportion of either NA or empty values. I removed these columns. The trigger I used was if more than 20% of observations in a column were NA values or if more than 80% of the column values were empty. However, the actual NA/empty ratios were in the 97% range, so these cutoffs turned out to be symbolic.

Model Selection Strategy:

I attempted to plot availabe predictor variables against the outcome. The resulting plots were non-informative (every variable had values roughly in the same range accross most of the outcome categories). Example plot can be seen in ** Appendix A2 **. This dataset is not nice and clear-cut as our simple lecture examples and I was not able to come up with any single variable that could be used as a reliable and intuitive predictor. I split the training data set into training and testing sub-sets (60/40). The training sub-set was further split into a training and testing sub-sub-sets (25/75). I started with a tree model, which produced un-remarkable results (30-55% accuracy, depending on column options I chose). At that point, I decided to try random forest model. Use of this model necessitated removal of non-numeric columns.

Model Tuning:

Due to random forest being processing-intensive, I performed model tuning on a training sub-sub-set (15% of total training data set). Computation took ~9min. The resulting model was based on 27 predictors. No pre-processing was done. Cross-validation was performed via 25 repetitions of bootstrapping. You can see model tuning parameters in ** Appendix A3 **. I then tried to predict the values on the same set used to tune the model. The model fit the data perfectly (an example of overfitting). I then fit the model to the 60% of the training data set. The overall accuracy results were encouraging (98.07% accuracy). I then predicted the outcome on the remaining 40% of the training data set (set apart as test data) using this model, which produced 98.18% accuracy. I therefore, estimate out-of-sample error to be ~2%, based on these observations. Please see** Appendix A4 ** for these details.

Conclusion:

Overall, random forest model was highly effective in using the available data to provide a very accurate (by beginner standards) prediction. This model was able to predict 18/20 outcomes in the TEST data set (I actually got 20/20, but on my final version something changed and I could not reproduce that). The drawback of random forest approach (to me) is difficulty in inferring or explaining the results. I am not even sure how to identify the 27 out of 54 columns that were used in the final model.

Appendix:

A1:

```
set.seed(56789)
dfRaw <- read.table(trainingFilename, sep=",", header=T, na.strings="NA"); df <- dfRaw
dfTest <- read.table(testingFilename, sep=",", header=T, na.strings="NA")
paste("Training data set has rows, columns: ", toString(dim(dfRaw)))
```

```
## [1] "Training data set has rows, columns:  19622, 160"
```

```
naColumns <- clean_data(dfRaw)
# Remove columns with high NA/empty ratio:
df <- df[,-naColumns]
dfTest <- dfTest[,-naColumns]
paste("Trainig data set after removing NA/empty columns has", ncol(df), "columns.")
```

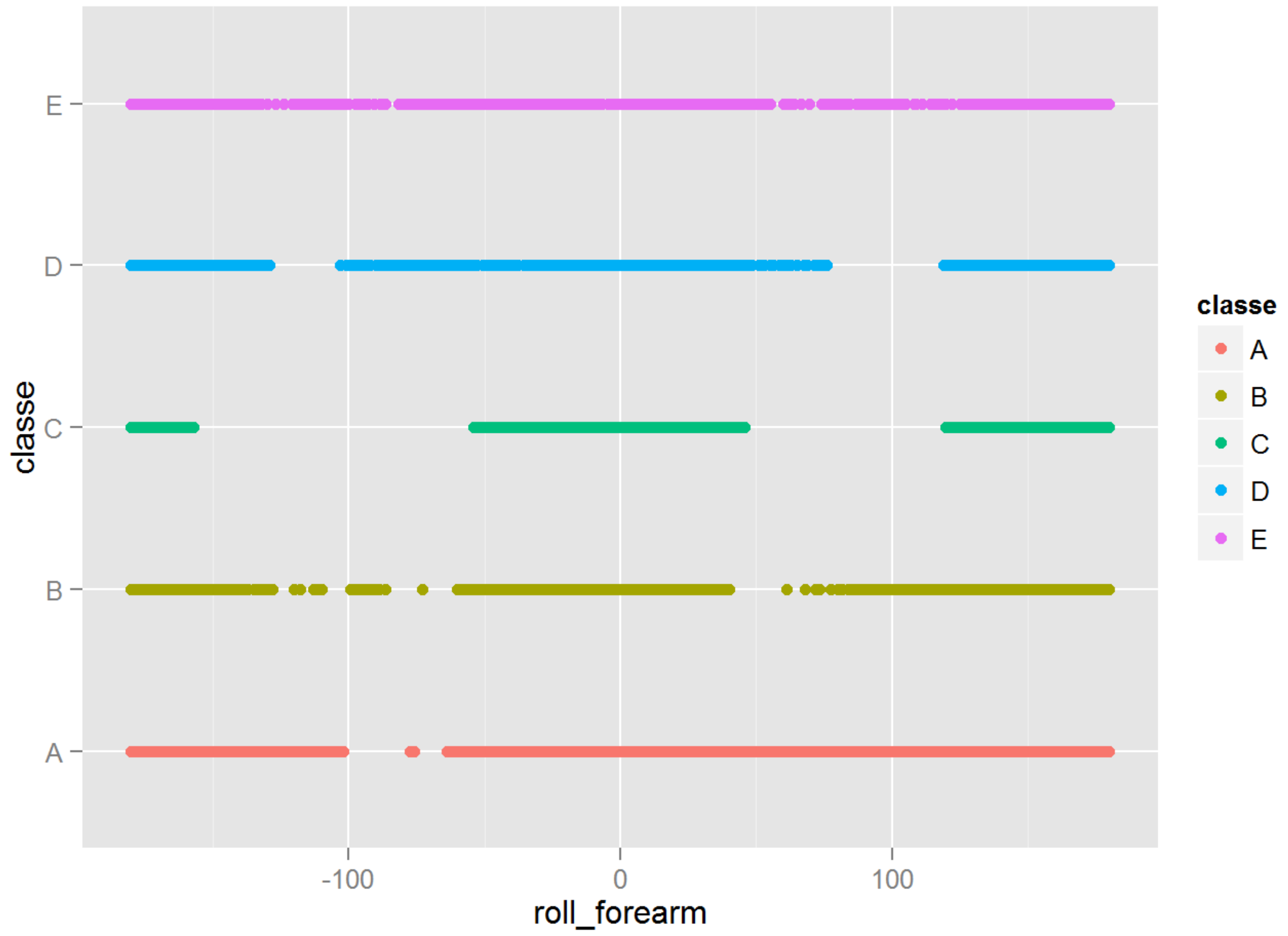
```
## [1] "Trainig data set after removing NA/empty columns has 60 columns."
```

```
# Remove non-numeric columns:
df <- df[,7:60]
dfTest <- dfTest[,7:60]
paste("Trainig data set after removing non-numeric columns has", ncol(df), "columns.")
```

```
## [1] "Trainig data set after removing non-numeric columns has 54 columns."
```

A2:

```
library(caret)
qplot(roll_forearm, classe, colour=classe, data=df)
```



```
# Split df into training and testing data sets:
inTrain <- createDataPartition(y=df$classe, p=0.6, list=F);
training <- df[inTrain,]
testing <- df[-inTrain,]
dim(training)
```

```
## [1] 11776    54
```

```
# Split training data set into a small and a large training set:
smallInTrain <- createDataPartition(y=training$classe, p=0.25, list=F);
smallTraining <- training[smallInTrain,];
largeTraining <- training[-smallInTrain,];
dim(smallTraining)
```

```
## [1] 2946   54
```

```
modFitTree <- train(classe ~ ., method="rpart", data=smallTraining)
```

```
## Loading required package: rpart
```

```
confusionMatrix(smallTraining$classe, predict(modFitTree, smallTraining))
```

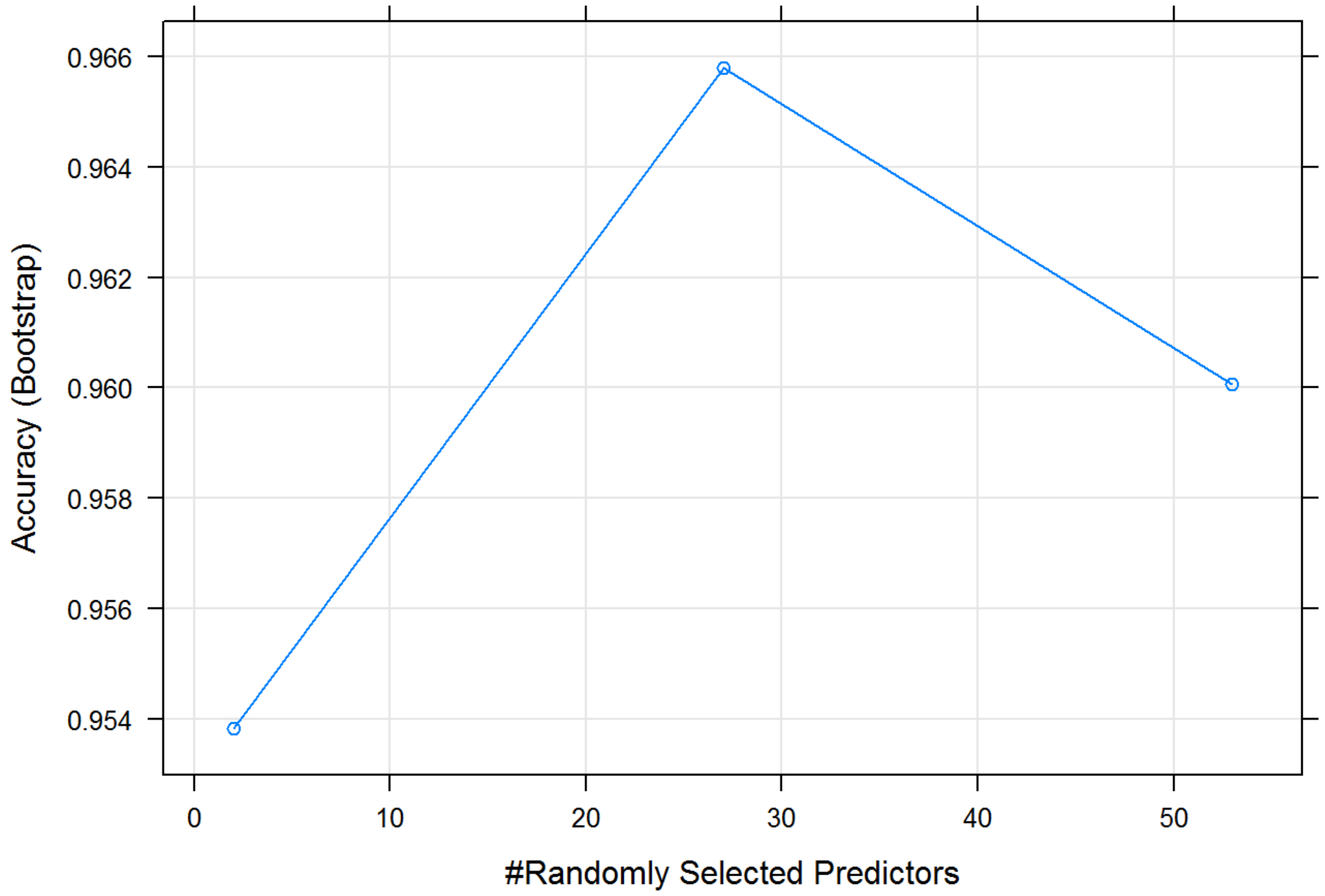
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 712  26  97   0   2
##          B 126 236 208   0   0
##          C  47  40 427   0   0
##          D 141 116 226   0   0
##          E 126 110  87   0 219
##
## Overall Statistics
##
##                Accuracy : 0.541
##                  95% CI : (0.523, 0.559)
##     No Information Rate : 0.391
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.411
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.618   0.4470    0.409       NA   0.9910
## Specificity             0.930   0.8619    0.954    0.836   0.8815
## Pos Pred Value          0.851   0.4140    0.831       NA   0.4041
## Neg Pred Value          0.791   0.8771    0.746       NA   0.9992
## Prevalence              0.391   0.1792    0.355    0.000   0.0750
## Detection Rate          0.242   0.0801    0.145    0.000   0.0743
## Detection Prevalence    0.284   0.1935    0.174    0.164   0.1840
## Balanced Accuracy       0.774   0.6544    0.681       NA   0.9362
```

A3:

```
modFitForest <- train(classe ~ ., method="rf", data=smallTraining)
```

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

```
plot(modFitForest)
```



A4:

```
confusionMatrix(smallTraining$classe, predict(modFitForest, smallTraining))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   A   B   C   D   E
##          A 837   0   0   0   0
##          B   0 570   0   0   0
##          C   0   0 514   0   0
##          D   0   0   0 483   0
##          E   0   0   0   0 542
## 
## Overall Statistics
## 
##                Accuracy : 1
##                  95% CI : (0.999, 1)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
## 
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    1.000    1.000    1.000    1.000
## Specificity             1.000    1.000    1.000    1.000    1.000
## Pos Pred Value          1.000    1.000    1.000    1.000    1.000
## Neg Pred Value          1.000    1.000    1.000    1.000    1.000
## Prevalence              0.284    0.193    0.174    0.164    0.184
## Detection Rate          0.284    0.193    0.174    0.164    0.184
## Detection Prevalence    0.284    0.193    0.174    0.164    0.184
## Balanced Accuracy       1.000    1.000    1.000    1.000    1.000
```

```
confusionMatrix(largeTraining$classe, predict(modFitForest, largeTraining))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2507    0    0    1    3
##          B   39 1645   21    0    4
##          C    0   22 1515    3    0
##          D    0    1   37 1403    6
##          E    0    7    2   21 1593
##
## Overall Statistics
##
##                Accuracy : 0.981
##                  95% CI : (0.978, 0.984)
##     No Information Rate : 0.288
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.976
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.985    0.982    0.962    0.982    0.992
## Specificity             0.999    0.991    0.997    0.994    0.996
## Pos Pred Value          0.998    0.963    0.984    0.970    0.982
## Neg Pred Value          0.994    0.996    0.992    0.997    0.998
## Prevalence              0.288    0.190    0.178    0.162    0.182
## Detection Rate          0.284    0.186    0.172    0.159    0.180
## Detection Prevalence    0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy       0.992    0.987    0.979    0.988    0.994
```

```
confusionMatrix(testing$classe, predict(modFitForest, testing))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2228    0    1    0    3
##          B   28 1473   14    0    3
##          C    0   31 1333    4    0
##          D    0    3   24 1252    7
##          E    2    9    0   14 1417
##
## Overall Statistics
##
##                Accuracy : 0.982
##                  95% CI : (0.979, 0.985)
##     No Information Rate : 0.288
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.977
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.987    0.972    0.972    0.986    0.991
## Specificity             0.999    0.993    0.995    0.995    0.996
## Pos Pred Value          0.998    0.970    0.974    0.974    0.983
## Neg Pred Value          0.995    0.993    0.994    0.997    0.998
## Prevalence              0.288    0.193    0.175    0.162    0.182
## Detection Rate          0.284    0.188    0.170    0.160    0.181
## Detection Prevalence    0.284    0.193    0.174    0.164    0.184
## Balanced Accuracy       0.993    0.982    0.983    0.990    0.994
```

```
# Generate predictions for the TEST dataset:
pred <- predict(modFitForest, dfTest)
paste("Test dataset predictions:", toString(pred))
```

```
## [1] "Test dataset predictions: B, A, A, A, A, E, D, D, A, A, B, C, B, A, E, E, A, B, B, B"
```