Deel_Practical Machine Learning_Course Project

Jakob Deel

November 11, 2018

Executive Summary

Using data from wearable workout technology, this project builds and tests a machine learning algorithm to predict the level of quality with which an exercise was conducted ("classe"). The optimal model was a boosted generalized linear model, with an estimated very low out of sample error.

```
#sets working directory and removes unnecessary objects
setwd("C:/Users/jdeel/Documents/Training/JHU - Coursera/Practical Machine Learning/Cou
rse Project")
rm(list = ls())

library(AppliedPredictiveModeling)
library(caret)
```

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

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

```
library(ElemStatLearn)
library(pgmm)
library(rpart)
library(gbm)
```

```
## Loaded gbm 2.1.4
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```r
library(forecast)
library(e1071)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

Data Loading and Processing

The following data loads the provided training and test datasets and filters out the fields that are more than 75% missing data.

```r
#loads data
trainingraw = read.csv("pml-training.csv",na.strings=c("NA",""))
testingraw = read.csv("pml-testing.csv",na.strings=c("NA",""))

#filters out dimensions from training and test sets where most values are NA
trainNAs <- apply(trainingraw, 2, function(x) {
  sum(is.na(x))
})

mostNAstrain <- trainNAs>(0.75*dim(trainingraw[1]))

goodfieldstraining <- subset(names(mostNAstrain),mostNAstrain==FALSE)

training <- subset(trainingraw,select=goodfieldstraining)

testNAs <- apply(testingraw, 2, function(x) {
  sum(is.na(x))
})

mostNAstest <- testNAs>(0.75*dim(testingraw[1]))

goodfieldstest <- subset(names(mostNAstest),mostNAstest==FALSE)
testing <- testingraw[,goodfieldstest]
```

Model Building and Cross-Validation

The following data builds two models, a random forest and boosted model, and tests each using 3-fold cross validation to maximize accuracy. Though both models results in high accuracy, the boosted model is better and is therefore selected for use

```
#sets seed for reproducibility
set.seed(123)

#creates models using random forest and boosting, testing each using 3-fold cross vali
dation
# Define training control
train.control <- trainControl(method = "cv", number = 3)

# Train the random forest model
forestmodel <- train(classe~.,data=training,method="rf",
               trControl = train.control,na.action=na.omit)
# Summarize the results
print(forestmodel)
```

```
## Random Forest
##
## 19622 samples
##    59 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 13081, 13082, 13081
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9954643  0.9942628
##   41    0.9998981  0.9998711
##   81    0.9997961  0.9997422
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 41.
```

```
# Train the boosting model
boostedmodel <- train(classe~.,data=training,method="gbm",
                      trControl = train.control,na.action=na.omit)
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.4544
##      2        1.3252             nan     0.1000    0.3110
##      3        1.1324             nan     0.1000    0.2402
##      4        0.9841             nan     0.1000    0.1978
##      5        0.8625             nan     0.1000    0.1587
##      6        0.7635             nan     0.1000    0.1447
##      7        0.6750             nan     0.1000    0.1229
##      8        0.6001             nan     0.1000    0.1081
##      9        0.5345             nan     0.1000    0.0890
##     10        0.4788             nan     0.1000    0.0845
##     20        0.1648             nan     0.1000    0.0263
##     40        0.0236             nan     0.1000    0.0035
##     60        0.0039             nan     0.1000    0.0004
##     80        0.0009             nan     0.1000    0.0000
##    100        0.0003             nan     0.1000    0.0000
##    120        0.0001             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.7755
##      2        1.1448             nan     0.1000    0.4623
##      3        0.8699             nan     0.1000    0.3209
##      4        0.6790             nan     0.1000    0.2375
##      5        0.5377             nan     0.1000    0.1814
##      6        0.4296             nan     0.1000    0.1413
##      7        0.3454             nan     0.1000    0.1116
##      8        0.2789             nan     0.1000    0.0887
##      9        0.2258             nan     0.1000    0.0712
##     10        0.1833             nan     0.1000    0.0572
##     20        0.0241             nan     0.1000    0.0073
##     40        0.0007             nan     0.1000    0.0001
##     60        0.0000             nan     0.1000    0.0000
##     80        0.0000             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.7757
##      2        1.1447             nan     0.1000    0.4621
##      3        0.8700             nan     0.1000    0.3210
##      4        0.6790             nan     0.1000    0.2374
##      5        0.5377             nan     0.1000    0.1813
##      6        0.4297             nan     0.1000    0.1411
##      7        0.3455             nan     0.1000    0.1114
```

```
##      8        0.2789             nan     0.1000    0.0886
##      9        0.2259             nan     0.1000    0.0710
##     10        0.1835             nan     0.1000    0.0573
##     20        0.0242             nan     0.1000    0.0072
##     40        0.0005             nan     0.1000    0.0001
##     60        0.0000             nan     0.1000    0.0000
##     80        0.0000             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000   -0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.4532
##      2        1.3255             nan     0.1000    0.3098
##      3        1.1323             nan     0.1000    0.2418
##      4        0.9835             nan     0.1000    0.1977
##      5        0.8618             nan     0.1000    0.1581
##      6        0.7630             nan     0.1000    0.1443
##      7        0.6748             nan     0.1000    0.1224
##      8        0.6004             nan     0.1000    0.1081
##      9        0.5345             nan     0.1000    0.0893
##     10        0.4793             nan     0.1000    0.0849
##     20        0.1649             nan     0.1000    0.0262
##     40        0.0235             nan     0.1000    0.0032
##     60        0.0039             nan     0.1000    0.0006
##     80        0.0009             nan     0.1000    0.0001
##    100        0.0002             nan     0.1000    0.0000
##    120        0.0001             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.7769
##      2        1.1447             nan     0.1000    0.4614
##      3        0.8700             nan     0.1000    0.3211
##      4        0.6790             nan     0.1000    0.2377
##      5        0.5377             nan     0.1000    0.1813
##      6        0.4297             nan     0.1000    0.1414
##      7        0.3454             nan     0.1000    0.1115
##      8        0.2788             nan     0.1000    0.0887
##      9        0.2258             nan     0.1000    0.0711
##     10        0.1833             nan     0.1000    0.0571
##     20        0.0241             nan     0.1000    0.0073
##     40        0.0006             nan     0.1000    0.0002
##     60        0.0001             nan     0.1000    0.0000
##     80        0.0000             nan     0.1000   -0.0000
##    100        0.0000             nan     0.1000   -0.0000
##    120        0.0000             nan     0.1000   -0.0000
```

```
##    140        0.0000             nan     0.1000   -0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.7747
##      2        1.1447             nan     0.1000    0.4619
##      3        0.8699             nan     0.1000    0.3213
##      4        0.6789             nan     0.1000    0.2376
##      5        0.5377             nan     0.1000    0.1811
##      6        0.4297             nan     0.1000    0.1413
##      7        0.3454             nan     0.1000    0.1116
##      8        0.2789             nan     0.1000    0.0889
##      9        0.2258             nan     0.1000    0.0711
##     10        0.1833             nan     0.1000    0.0572
##     20        0.0241             nan     0.1000    0.0073
##     40        0.0005             nan     0.1000    0.0001
##     60        0.0000             nan     0.1000    0.0000
##     80        0.0000             nan     0.1000   -0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000   -0.0000
##    150        0.0000             nan     0.1000   -0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.4544
##      2        1.3253             nan     0.1000    0.3111
##      3        1.1319             nan     0.1000    0.2402
##      4        0.9832             nan     0.1000    0.1980
##      5        0.8615             nan     0.1000    0.1587
##      6        0.7629             nan     0.1000    0.1445
##      7        0.6746             nan     0.1000    0.1223
##      8        0.5998             nan     0.1000    0.1079
##      9        0.5340             nan     0.1000    0.0890
##     10        0.4787             nan     0.1000    0.0844
##     20        0.1658             nan     0.1000    0.0281
##     40        0.0236             nan     0.1000    0.0034
##     60        0.0039             nan     0.1000    0.0006
##     80        0.0008             nan     0.1000    0.0001
##    100        0.0002             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.7765
##      2        1.1446             nan     0.1000    0.4614
##      3        0.8698             nan     0.1000    0.3211
##      4        0.6789             nan     0.1000    0.2372
##      5        0.5376             nan     0.1000    0.1814
```

```
##      6        0.4295             nan     0.1000    0.1411
##      7        0.3452             nan     0.1000    0.1115
##      8        0.2787             nan     0.1000    0.0887
##      9        0.2257             nan     0.1000    0.0711
##     10        0.1832             nan     0.1000    0.0572
##     20        0.0240             nan     0.1000    0.0073
##     40        0.0004             nan     0.1000    0.0001
##     60        0.0000             nan     0.1000    0.0000
##     80        0.0000             nan     0.1000   -0.0000
##    100        0.0000             nan     0.1000   -0.0000
##    120        0.0000             nan     0.1000   -0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000   -0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.7765
##      2        1.1447             nan     0.1000    0.4603
##      3        0.8699             nan     0.1000    0.3212
##      4        0.6789             nan     0.1000    0.2372
##      5        0.5376             nan     0.1000    0.1809
##      6        0.4296             nan     0.1000    0.1413
##      7        0.3454             nan     0.1000    0.1113
##      8        0.2789             nan     0.1000    0.0887
##      9        0.2259             nan     0.1000    0.0710
##     10        0.1834             nan     0.1000    0.0573
##     20        0.0242             nan     0.1000    0.0072
##     40        0.0005             nan     0.1000    0.0001
##     60        0.0000             nan     0.1000    0.0000
##     80        0.0000             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000   -0.0000
##    120        0.0000             nan     0.1000   -0.0000
##    140        0.0000             nan     0.1000   -0.0000
##    150        0.0000             nan     0.1000   -0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.7777
##      2        1.1447             nan     0.1000    0.4619
##      3        0.8699             nan     0.1000    0.3210
##      4        0.6789             nan     0.1000    0.2375
##      5        0.5376             nan     0.1000    0.1813
##      6        0.4296             nan     0.1000    0.1411
##      7        0.3453             nan     0.1000    0.1114
##      8        0.2788             nan     0.1000    0.0888
##      9        0.2258             nan     0.1000    0.0710
##     10        0.1833             nan     0.1000    0.0572
##     20        0.0241             nan     0.1000    0.0073
##     40        0.0005             nan     0.1000    0.0001
##     50        0.0001             nan     0.1000    0.0000
```

```
# Summarize the results
print(boostedmodel)
```

```
## Stochastic Gradient Boosting
##
## 19622 samples
##    59 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 13080, 13082, 13082
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.9997961  0.9997422
##   1                  100      0.9997452  0.9996777
##   1                  150      0.9996942  0.9996132
##   2                   50      0.9997961  0.9997422
##   2                  100      0.9997961  0.9997422
##   2                  150      0.9997961  0.9997422
##   3                   50      0.9998981  0.9998711
##   3                  100      0.9998471  0.9998066
##   3                  150      0.9997961  0.9997422
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 50, interaction.depth
##  = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
# boosted has better accuracy in CV (0.99999 in optimal model vs. 0.9999 in optimal fo
rest model), so that's one is used
```

Out of Sample Error Estimation

The following code runs the model on a randomly-selected third of the training dataset and checks the accuracy to estimate out of sample error. Though not a perfect test, the fact that the accuracy is 100% means that we should expect very low out of sample error.

```
#runs boosted model on random third of training set to estimate out of sample error
oostrainsample <- training$classe %>%
  createDataPartition(p = 0.33, list = FALSE)

oostraining <- training[oostrainsample,]

oostraining$boostedpreds <- predict(boostedmodel,oostraining)
oostraining$boostedright <- oostraining$classe==oostraining$boostedpreds

#estimates out of sample error
mean(oostraining$boostedright)
```

```
## [1] 1
```