Prediction-Assignment-Writeup

2022-10-23

Data

Data Cleaning and Preparation

```
trainUrl <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-tr
aining.csv'
testUrl <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-tes
ting.csv'
train_in <- read.csv(trainUrl, header=T)
validation <- read.csv(testUrl, header=T)
```

Data Partitioning

```
library(ggplot2)
library(recipes)

## 载入需要的程辑包：dplyr

##
## 载入程辑包：'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

##
## 载入程辑包：'recipes'

## The following object is masked from 'package:stats':
##
##     step

library(caret)

## 载入需要的程辑包：lattice

set.seed(111)
training_sample <- createDataPartition(y=train_in$classe, p=0.7, list=F
ALSE)
training <- train_in[training_sample, ]
testing <- train_in[-training_sample, ]
```

Identification on Non-Zero Data

```
all_zero_colnames <- sapply(names(validation), function(x) all(is.na(va
lidation[,x])==TRUE))
nznames <- names(all_zero_colnames)[all_zero_colnames==FALSE]
nznames <- nznames[-(1:7)]
nznames <- nznames[1:(length(nznames)-1)]
print(nznames)

##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"

##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"

##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"

## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"

## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"

## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"

## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"

## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"

## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"

## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbe
ll"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"

## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"

## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z
"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"

## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"

## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"

## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"

## [52] "magnet_forearm_z"
```

Model building

The three model types that will be tested are:

1. Decision trees with CART (rpart)

2. Stochastic gradient boosting trees (gbm)

3. Random forest decision trees (rf)

```
library(gbm)

## Loaded gbm 2.1.8.1

library(randomForest)

## randomForest 4.7-1.1

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

##
## 载入程辑包：'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

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

Cross validation

```
fitControl <- trainControl(method='cv', number = 3)

model_cart <- train(
  classe ~ .,
  data=training[, c('classe', nznames)],
  trControl=fitControl,
  method='rpart'
)
save(model_cart, file='./ModelFitCART.RData')
model_gbm <- train(
  classe ~ .,
  data=training[, c('classe', nznames)],
  trControl=fitControl,
  method='gbm'
)

## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1333
##      2        1.5244             nan     0.1000    0.0846
##      3        1.4678             nan     0.1000    0.0641
##      4        1.4257             nan     0.1000    0.0532
##      5        1.3913             nan     0.1000    0.0420
##      6        1.3625             nan     0.1000    0.0424
##      7        1.3339             nan     0.1000    0.0400
##      8        1.3089             nan     0.1000    0.0383
##      9        1.2839             nan     0.1000    0.0328
```

```
##     10        1.2633             nan     0.1000    0.0308
##     20        1.1074             nan     0.1000    0.0158
##     40        0.9345             nan     0.1000    0.0093
##     60        0.8251             nan     0.1000    0.0058
##     80        0.7455             nan     0.1000    0.0044
##    100        0.6805             nan     0.1000    0.0046
##    120        0.6293             nan     0.1000    0.0029
##    140        0.5830             nan     0.1000    0.0024
##    150        0.5628             nan     0.1000    0.0019
## 
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1746
##      2        1.4928             nan     0.1000    0.1326
##      3        1.4093             nan     0.1000    0.1040
##      4        1.3426             nan     0.1000    0.0816
##      5        1.2884             nan     0.1000    0.0775
##      6        1.2375             nan     0.1000    0.0622
##      7        1.1986             nan     0.1000    0.0515
##      8        1.1649             nan     0.1000    0.0536
##      9        1.1311             nan     0.1000    0.0463
##     10        1.1002             nan     0.1000    0.0433
##     20        0.8947             nan     0.1000    0.0222
##     40        0.6754             nan     0.1000    0.0118
##     60        0.5473             nan     0.1000    0.0062
##     80        0.4601             nan     0.1000    0.0048
##    100        0.3934             nan     0.1000    0.0042
##    120        0.3396             nan     0.1000    0.0020
##    140        0.2984             nan     0.1000    0.0011
##    150        0.2816             nan     0.1000    0.0022
## 
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2278
##      2        1.4623             nan     0.1000    0.1595
##      3        1.3609             nan     0.1000    0.1247
##      4        1.2811             nan     0.1000    0.1065
##      5        1.2150             nan     0.1000    0.0860
##      6        1.1596             nan     0.1000    0.0893
##      7        1.1042             nan     0.1000    0.0592
##      8        1.0648             nan     0.1000    0.0556
##      9        1.0285             nan     0.1000    0.0587
##     10        0.9923             nan     0.1000    0.0468
##     20        0.7532             nan     0.1000    0.0220
##     40        0.5210             nan     0.1000    0.0105
##     60        0.3947             nan     0.1000    0.0072
##     80        0.3150             nan     0.1000    0.0035
##    100        0.2594             nan     0.1000    0.0037
##    120        0.2165             nan     0.1000    0.0017
##    140        0.1825             nan     0.1000    0.0020
##    150        0.1691             nan     0.1000    0.0012
## 
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1288
##      2        1.5240             nan     0.1000    0.0865
##      3        1.4676             nan     0.1000    0.0682
##      4        1.4229             nan     0.1000    0.0540
##      5        1.3878             nan     0.1000    0.0482
##      6        1.3564             nan     0.1000    0.0385
##      7        1.3306             nan     0.1000    0.0409
##      8        1.3046             nan     0.1000    0.0359
##      9        1.2819             nan     0.1000    0.0322
##     10        1.2590             nan     0.1000    0.0302
##     20        1.1050             nan     0.1000    0.0173
##     40        0.9322             nan     0.1000    0.0096
##     60        0.8259             nan     0.1000    0.0063
##     80        0.7463             nan     0.1000    0.0039
##    100        0.6804             nan     0.1000    0.0033
##    120        0.6287             nan     0.1000    0.0038
##    140        0.5842             nan     0.1000    0.0020
##    150        0.5662             nan     0.1000    0.0026
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1796
##      2        1.4922             nan     0.1000    0.1324
##      3        1.4063             nan     0.1000    0.0993
##      4        1.3407             nan     0.1000    0.0800
##      5        1.2883             nan     0.1000    0.0709
##      6        1.2418             nan     0.1000    0.0674
##      7        1.1989             nan     0.1000    0.0630
##      8        1.1594             nan     0.1000    0.0578
##      9        1.1240             nan     0.1000    0.0445
##     10        1.0954             nan     0.1000    0.0432
##     20        0.8905             nan     0.1000    0.0186
##     40        0.6774             nan     0.1000    0.0133
##     60        0.5478             nan     0.1000    0.0085
##     80        0.4585             nan     0.1000    0.0050
##    100        0.3902             nan     0.1000    0.0026
##    120        0.3429             nan     0.1000    0.0027
##    140        0.3011             nan     0.1000    0.0032
##    150        0.2828             nan     0.1000    0.0018
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2271
##      2        1.4627             nan     0.1000    0.1623
##      3        1.3608             nan     0.1000    0.1215
##      4        1.2823             nan     0.1000    0.1107
##      5        1.2125             nan     0.1000    0.0815
##      6        1.1593             nan     0.1000    0.0720
##      7        1.1132             nan     0.1000    0.0751
##      8        1.0658             nan     0.1000    0.0687
##      9        1.0228             nan     0.1000    0.0575
```

```
##     10        0.9865             nan     0.1000    0.0468
##     20        0.7561             nan     0.1000    0.0218
##     40        0.5252             nan     0.1000    0.0113
##     60        0.4037             nan     0.1000    0.0051
##     80        0.3239             nan     0.1000    0.0038
##    100        0.2651             nan     0.1000    0.0030
##    120        0.2217             nan     0.1000    0.0040
##    140        0.1867             nan     0.1000    0.0017
##    150        0.1725             nan     0.1000    0.0018
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1283
##      2        1.5246             nan     0.1000    0.0840
##      3        1.4672             nan     0.1000    0.0652
##      4        1.4234             nan     0.1000    0.0524
##      5        1.3888             nan     0.1000    0.0429
##      6        1.3605             nan     0.1000    0.0480
##      7        1.3304             nan     0.1000    0.0363
##      8        1.3065             nan     0.1000    0.0362
##      9        1.2809             nan     0.1000    0.0319
##     10        1.2596             nan     0.1000    0.0299
##     20        1.1053             nan     0.1000    0.0185
##     40        0.9341             nan     0.1000    0.0063
##     60        0.8258             nan     0.1000    0.0050
##     80        0.7443             nan     0.1000    0.0055
##    100        0.6826             nan     0.1000    0.0045
##    120        0.6316             nan     0.1000    0.0031
##    140        0.5879             nan     0.1000    0.0018
##    150        0.5676             nan     0.1000    0.0014
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1816
##      2        1.4892             nan     0.1000    0.1316
##      3        1.4043             nan     0.1000    0.1001
##      4        1.3404             nan     0.1000    0.0882
##      5        1.2842             nan     0.1000    0.0809
##      6        1.2308             nan     0.1000    0.0582
##      7        1.1930             nan     0.1000    0.0615
##      8        1.1541             nan     0.1000    0.0484
##      9        1.1217             nan     0.1000    0.0486
##     10        1.0907             nan     0.1000    0.0392
##     20        0.8880             nan     0.1000    0.0190
##     40        0.6805             nan     0.1000    0.0176
##     60        0.5496             nan     0.1000    0.0048
##     80        0.4613             nan     0.1000    0.0048
##    100        0.3976             nan     0.1000    0.0030
##    120        0.3477             nan     0.1000    0.0034
##    140        0.3040             nan     0.1000    0.0027
##    150        0.2875             nan     0.1000    0.0011
##
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2324
##      2        1.4610             nan     0.1000    0.1550
##      3        1.3625             nan     0.1000    0.1218
##      4        1.2834             nan     0.1000    0.1107
##      5        1.2136             nan     0.1000    0.0956
##      6        1.1529             nan     0.1000    0.0722
##      7        1.1058             nan     0.1000    0.0660
##      8        1.0635             nan     0.1000    0.0638
##      9        1.0221             nan     0.1000    0.0582
##     10        0.9850             nan     0.1000    0.0457
##     20        0.7525             nan     0.1000    0.0234
##     40        0.5266             nan     0.1000    0.0090
##     60        0.4001             nan     0.1000    0.0067
##     80        0.3165             nan     0.1000    0.0039
##    100        0.2597             nan     0.1000    0.0016
##    120        0.2168             nan     0.1000    0.0020
##    140        0.1834             nan     0.1000    0.0017
##    150        0.1687             nan     0.1000    0.0013
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2309
##      2        1.4636             nan     0.1000    0.1613
##      3        1.3619             nan     0.1000    0.1241
##      4        1.2833             nan     0.1000    0.1066
##      5        1.2159             nan     0.1000    0.0935
##      6        1.1581             nan     0.1000    0.0748
##      7        1.1118             nan     0.1000    0.0795
##      8        1.0630             nan     0.1000    0.0584
##      9        1.0265             nan     0.1000    0.0512
##     10        0.9940             nan     0.1000    0.0523
##     20        0.7580             nan     0.1000    0.0288
##     40        0.5384             nan     0.1000    0.0181
##     60        0.4094             nan     0.1000    0.0076
##     80        0.3286             nan     0.1000    0.0068
##    100        0.2685             nan     0.1000    0.0033
##    120        0.2252             nan     0.1000    0.0019
##    140        0.1919             nan     0.1000    0.0015
##    150        0.1786             nan     0.1000    0.0020

save(model_gbm, file='./ModelFitGBM.RData')
model_rf <- train(
  classe ~ .,
  data=training[, c('classe', nznames)],
  trControl=fitControl,
  method='rf',
  ntree=100
)
save(model_rf, file='./ModelFitRF.RData')
```

Model Assessment (Out of sample error)

```
predCART <- predict(model_cart, newdata=testing)
cmCART <- confusionMatrix(predCART, as.factor(testing$classe))
predGBM <- predict(model_gbm, newdata=testing)
cmGBM <- confusionMatrix(predGBM, as.factor(testing$classe))
predRF <- predict(model_rf, newdata=testing)
cmRF <- confusionMatrix(predRF, as.factor(testing$classe))
AccuracyResults <- data.frame(
  Model = c('CART', 'GBM', 'RF'),
  Accuracy = rbind(cmCART$overall[1], cmGBM$overall[1], cmRF$overall[1])
)
print(AccuracyResults)
```

```
##   Model  Accuracy
## 1  CART 0.5029737
## 2   GBM 0.9593883
## 3    RF 0.9949023
```

```
print(cmRF)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    5    0    0    0
##          B    0 1131    6    0    0
##          C    0    3 1018   10    0
##          D    0    0    2  954    4
##          E    0    0    0    0 1078
## 
## Overall Statistics
## 
##                Accuracy : 0.9949
##                  95% CI : (0.9927, 0.9966)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9936
## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9930   0.9922   0.9896   0.9963
## Specificity            0.9988   0.9987   0.9973   0.9988   1.0000
## Pos Pred Value         0.9970   0.9947   0.9874   0.9937   1.0000
## Neg Pred Value         1.0000   0.9983   0.9984   0.9980   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1922   0.1730   0.1621   0.1832
```

```
## Detection Prevalence   0.2853   0.1932   0.1752   0.1631   0.1832
## Balanced Accuracy      0.9994   0.9959   0.9948   0.9942   0.9982
```

Based on an assessment of these 3 model fits and out-of-sample results, it looks like both gradient boosting and random forests outperform the CART model, with random forests being slightly more accurate.

Prediction

The Random Forest model was applied to predict 20 different test cases.

```r
predRF_Test <- predict(model_rf, newdata=validation)
predRF_Test

##  [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
```

Conclusion

The Random Forest classification model in combination with a couple of simple data preprocessing procedures (such as removing irrelevant data columns and standardizing) is turned out to be a great approach to predict the manner in which people did the exercise, using the given data from accelerometers on the belt, forearm, arm, and dumbell of participants.