Practical Machine Learning - Prediction Assignment

Farah M

13/12/2020

Overview

The goal of the project is to to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they did the exercise. This is the "classe" variable in the training set. This report will outline:

1. How the model is built.
2. How cross validation is used.
3. What is the expected out of sample error.
4. Rationale for the choices made.

Load libraries

```
library(tidyverse)
library(caret)
library(randomForest)
library(xgboost)
library(corrplot)
```

Getting Data

```
# URLs for the training and testing data
training_url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# data directory and files
data_dir = "./data"
training_file = "pml-training.csv"
test_file = "pml-test.csv"

# if directory does not exist, create new
if (!file.exists(data_dir)) {
  dir.create(data_dir)
}

# if files does not exist, download the files
if (!file.exists(file.path(data_dir, training_file))) {
  download.file(training_url, destfile=file.path(data_dir, training_file))
}
if (!file.exists(file.path(data_dir, test_file))) {
  download.file(test_url, destfile=file.path(data_dir, test_file))
}

# load the CSV files as data.frame
train <- read_csv(file.path(data_dir, training_file), na=c("", "NA", "NULL", "#DIV/0!"))
test <- read_csv(file.path(data_dir, test_file), na=c("", "NA", "NULL", "#DIV/0!"))
dim(train)
```

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

```
dim(test)
```

```
## [1]  20 160
```

Data pre-processing

In the preprocessing of the data, we will:

- Remove predictors containg missing values.
- Remove "zero- and near zero- variance predictors".
- Remove columns that will not be useful for the model.

```
# remove predictors with NA and missing values
cleanTrain <- train %>%
  select(which(colMeans(is.na(.)) == 0))

# remove "zero- and near zero- variance predictors"
cleanTrain <- cleanTrain %>%
  select(-nearZeroVar(cleanTrain))

# remove the columns that will not be useful, such as user_name and timestamps
cleanTrain <- cleanTrain[, -(1:5)]
dim(cleanTrain)
```

```
## [1] 19622    54
```

After the preprocessing, we're left with 48 predictors (exclude "classe" and index columns) for our model training.

Data Splitting

We will split the training set into a training set and a validation set, in order for us to estimate the out-of-sample error.

```
# split data to create trainset and testset
set.seed(2468)

inTrain  <- createDataPartition(cleanTrain$classe, p=0.8, list=FALSE)

trainSet <- cleanTrain[inTrain,]
validationSet <- cleanTrain[-inTrain,]
dim(trainSet)
```

```
## [1] 15699    54
```

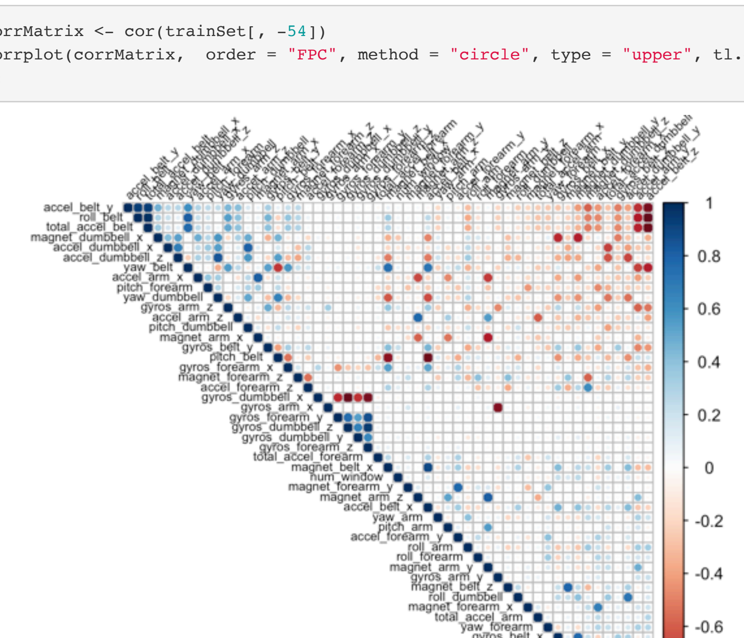
```
dim(validationSet)
```

```
## [1] 3923   54
```

Data Exploration

After removing the prdeictors that we'll not be using, we can have a look at the correlation between the remaining predictors before moving on to deciding on the models.

```
corrMatrix <- cor(trainSet[, -54])
corrplot(corrMatrix,  order = "FPC", method = "circle", type = "upper", tl.cex = 0.6, tl.col ="black", tl.srt = 4
5)
```



We can observe that the data set is largely uncorrelated. The higher the number (denoted by the blue color), the higher the correlation.

Model Training

The models that we will try are:

- Random Forest
- eXtreme Gradient Boosting

Both models are known to be quite accurate and are widely used methods for prediction. Another plus is that the correlations observed above will not adversely affect the effectiveness of the model.

Before training the models, we'll configure parallel processing to speed up the process.

```
library(parallel)
library(doParallel)
cluster <- makeCluster(detectCores() - 1, setup_strategy="sequential") # convention to leave 1 core for OS
registerDoParallel(cluster)
```

1. Random Forest

Here we will do a 5-fold cross validation resampling and let the model run up to 500 trees.

```
# model fit
rfControl <- trainControl(method="cv", number=5, allowParallel = TRUE)

set.seed(2468)
rfModel <- train(classe~., data=trainSet, method="rf", trControl=rfControl)
rfModel$finalModel
```

```
## 
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry) 
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
## 
##         OOB estimate of  error rate: 0.13%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4463    0    0    0    1 0.0002240143
## B    1 3034    2    1    0 0.0013166557
## C    0    4 2734    0    0 0.0014609204
## D    0    0    8 2564    1 0.0034978624
## E    0    0    0    3 2883 0.0010395010
```

```
# prediction
rfPredict <- predict(rfModel, newdata=validationSet)
rfConfMatrix <- confusionMatrix(rfPredict, as.factor(validationSet$classe))
rfConfMatrix
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1115    2    0    0    0
##          B    1  755    1    0    0
##          C    0    2  683    2    0
##          D    0    0    0  641    0
##          E    0    0    0    0  721
## 
## Overall Statistics
##                                          
##                Accuracy : 0.998          
##                  95% CI : (0.996, 0.9991)
##     No Information Rate : 0.2845         
##     P-Value [Acc > NIR] : < 2.2e-16      
##                                          
##                   Kappa : 0.9974         
##                                          
##  Mcnemar's Test P-Value : NA             
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9947   0.9985   0.9969   1.0000
## Specificity            0.9993   0.9994   0.9988   1.0000   1.0000
## Pos Pred Value         0.9982   0.9974   0.9942   1.0000   1.0000
## Neg Pred Value         0.9996   0.9987   0.9997   0.9994   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1925   0.1741   0.1634   0.1838
## Detection Prevalence   0.2847   0.1930   0.1751   0.1634   0.1838
## Balanced Accuracy      0.9992   0.9970   0.9987   0.9984   1.0000
```

The Random Forest method performed very well, and has low out-of-sample errors.

2. eXtreme Gradient Boosting

Again, we will do a 5-fold cross validation resampling.

```
# model fit
xgbControl <- trainControl(method="cv", number=5, allowParallel = TRUE)

set.seed(2468)
xgbModel <- train(classe~., data=trainSet, method="xgbTree", trControl=xgbControl)
xgbModel
```

```
## eXtreme Gradient Boosting 
## 
## 15699 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E' 
## 
## No pre-processing
## Resampling: Cross-Validated (5 fold) 
## Summary of sample sizes: 12561, 12557, 12558, 12560, 12560 
## Resampling results across tuning parameters:
## 
##   eta  max_depth  colsample_bytree  subsample  nrounds  Accuracy   Kappa    
##   0.3  1          0.6               0.50        50      0.8137454  0.7639077
##   0.3  1          0.6               0.50       100      0.8857900  0.8554230
##   0.3  1          0.6               0.50       150      0.9165558  0.8944118
##   0.3  1          0.6               0.75        50      0.8138120  0.7638497
##   0.3  1          0.6               0.75       100      0.8815232  0.8500407
##   0.3  1          0.6               0.75       150      0.9163648  0.8941729
##   0.3  1          0.6               1.00        50      0.8156587  0.7662711
##   0.3  1          0.6               1.00       100      0.8829240  0.8517916
##   0.3  1          0.6               1.00       150      0.9140714  0.8912615
##   0.3  1          0.8               0.50        50      0.8143850  0.7646440
##   0.3  1          0.8               0.50       100      0.8845165  0.8537850
##   0.3  1          0.8               0.50       150      0.9170022  0.8949608
##   0.3  1          0.8               0.75        50      0.8159780  0.7666893
##   0.3  1          0.8               0.75       100      0.8837517  0.8528512
##   0.3  1          0.8               0.75       150      0.9163011  0.8940797
##   0.3  1          0.8               1.00        50      0.8136838  0.7637419
##   0.3  1          0.8               1.00       100      0.8830516  0.8519425
##   0.3  1          0.8               1.00       150      0.9159824  0.8936798
##   0.3  2          0.6               0.50        50      0.9505705  0.9374626
##   0.3  2          0.6               0.50       100      0.9866238  0.9830807
##   0.3  2          0.6               0.50       150      0.9952867  0.9940381
##   0.3  2          0.6               0.75        50      0.9532468  0.9408527
##   0.3  2          0.6               0.75       100      0.9882802  0.9851753
##   0.3  2          0.6               0.75       150      0.9957960  0.9946825
##   0.3  2          0.6               1.00        50      0.9518451  0.9390851
##   0.3  2          0.6               1.00       100      0.9882793  0.9851740
##   0.3  2          0.6               1.00       150      0.9962421  0.9952467
##   0.3  2          0.8               0.50        50      0.9533103  0.9409325
##   0.3  2          0.8               0.50       100      0.9874517  0.9841273
##   0.3  2          0.8               0.50       150      0.9957327  0.9946023
##   0.3  2          0.8               0.75        50      0.9548383  0.9428737
##   0.3  2          0.8               0.75       100      0.9890441  0.9861419
##   0.3  2          0.8               0.75       150      0.9963055  0.9953271
##   0.3  2          0.8               1.00        50      0.9556027  0.9438319
##   0.3  2          0.8               1.00       100      0.9891081  0.9862233
##   0.3  2          0.8               1.00       150      0.9961144  0.9950851
##   0.3  3          0.6               0.50        50      0.9893628  0.9865454
##   0.3  3          0.6               0.50       100      0.9982805  0.9978252
##   0.3  3          0.6               0.50       150      0.9988536  0.9985499
##   0.3  3          0.6               0.75        50      0.9896171  0.9868662
##   0.3  3          0.6               0.75       100      0.9984715  0.9980667
##   0.3  3          0.6               0.75       150      0.9990447  0.9987918
##   0.3  3          0.6               1.00        50      0.9899356  0.9872698
##   0.3  3          0.6               1.00       100      0.9982168  0.9977445
##   0.3  3          0.6               1.00       150      0.9991721  0.9989529
##   0.3  3          0.8               0.50        50      0.9896177  0.9868675
##   0.3  3          0.8               0.50       100      0.9980256  0.9975027
##   0.3  3          0.8               0.50       150      0.9988537  0.9985501
##   0.3  3          0.8               0.75        50      0.9903822  0.9878343
##   0.3  3          0.8               0.75       100      0.9986627  0.9983085
##   0.3  3          0.8               0.75       150      0.9989174  0.9986307
##   0.3  3          0.8               1.00        50      0.9914644  0.9892033
##   0.3  3          0.8               1.00       100      0.9985352  0.9981473
##   0.3  3          0.8               1.00       150      0.9991721  0.9989528
##   0.4  1          0.6               0.50        50      0.8462979  0.8053178
##   0.4  1          0.6               0.50       100      0.9090407  0.8848808
##   0.4  1          0.6               0.50       150      0.9378951  0.9214096
##   0.4  1          0.6               0.75        50      0.8440670  0.8024811
##   0.4  1          0.6               0.75       100      0.9084028  0.8840893
##   0.4  1          0.6               0.75       150      0.9347101  0.9173969
##   0.4  1          0.6               1.00        50      0.8489718  0.8086527
##   0.4  1          0.6               1.00       100      0.9060464  0.8811046
##   0.4  1          0.6               1.00       150      0.9352197  0.9180371
##   0.4  1          0.8               0.50        50      0.8434315  0.8016840
##   0.4  1          0.8               0.50       100      0.9109513  0.8873216
##   0.4  1          0.8               0.50       150      0.9371303  0.9204555
##   0.4  1          0.8               0.75        50      0.8461704  0.8051391
##   0.4  1          0.8               0.75       100      0.9079572  0.8835195
##   0.4  1          0.8               0.75       150      0.9368120  0.9200456
##   0.4  1          0.8               1.00        50      0.8489099  0.8086010
##   0.4  1          0.8               1.00       100      0.9070651  0.8823924
##   0.4  1          0.8               1.00       150      0.9365576  0.9197295
##   0.4  2          0.6               0.50        50      0.9694883  0.9614044
##   0.4  2          0.6               0.50       100      0.9927382  0.9908143
##   0.4  2          0.6               0.50       150      0.9976433  0.9970191
##   0.4  2          0.6               0.75        50      0.9705076  0.9626932
##   0.4  2          0.6               0.75       100      0.9949044  0.9935548
##   0.4  2          0.6               0.75       150      0.9980894  0.9975834
##   0.4  2          0.6               1.00        50      0.9736921  0.9667236
##   0.4  2          0.6               1.00       100      0.9950316  0.9937153
##   0.4  2          0.6               1.00       150      0.9983441  0.9979056
##   0.4  2          0.8               0.50        50      0.9700626  0.9621295
##   0.4  2          0.8               0.50       100      0.9940766  0.9925076
##   0.4  2          0.8               0.50       150      0.9979620  0.9974224
##   0.4  2          0.8               0.75        50      0.9723544  0.9650298
##   0.4  2          0.8               0.75       100      0.9940125  0.9924265
##   0.4  2          0.8               0.75       150      0.9982167  0.9977445
##   0.4  2          0.8               1.00        50      0.9756052  0.9691400
##   0.4  2          0.8               1.00       100      0.9948406  0.9934740
##   0.4  2          0.8               1.00       150      0.9982805  0.9978251
##   0.4  3          0.6               0.50        50      0.9952862  0.9940375
##   0.4  3          0.6               0.50       100      0.9989173  0.9986306
##   0.4  3          0.6               0.50       150      0.9992358  0.9990334
##   0.4  3          0.6               0.75        50      0.9950315  0.9937153
##   0.4  3          0.6               0.75       100      0.9989811  0.9987112
##   0.4  3          0.6               0.75       150      0.9991085  0.9988724
##   0.4  3          0.6               1.00        50      0.9954139  0.9941991
##   0.4  3          0.6               1.00       100      0.9988536  0.9985500
##   0.4  3          0.6               1.00       150      0.9992357  0.9990334
##   0.4  3          0.8               0.50        50      0.9957323  0.9946020
##   0.4  3          0.8               0.50       100      0.9985990  0.9982280
##   0.4  3          0.8               0.50       150      0.9991084  0.9988723
##   0.4  3          0.8               0.75        50      0.9956685  0.9945211
##   0.4  3          0.8               0.75       100      0.9987263  0.9983889
##   0.4  3          0.8               0.75       150      0.9992358  0.9990334
##   0.4  3          0.8               1.00        50      0.9962419  0.9952466
##   0.4  3          0.8               1.00       100      0.9989810  0.9987112
##   0.4  3          0.8               1.00       150      0.9992358  0.9990334
## 
## Tuning parameter 'gamma' was held constant at a value of 0
## Tuning
##  parameter 'min_child_weight' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were nrounds = 150, max_depth = 3, eta
##  = 0.4, gamma = 0, colsample_bytree = 0.8, min_child_weight = 1 and subsample
##  = 1.
```

```
# prediction
xgbPredict <- predict(xgbModel, newdata=validationSet)
xgbConfMatrix <- confusionMatrix(xgbPredict, as.factor(validationSet$classe))
xgbConfMatrix
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    0    0    0    0
##          B    0  759    0    0    0
##          C    0    0  684    0    0
##          D    0    0    0  643    0
##          E    0    0    0    0  721
## 
## Overall Statistics
##                                      
##                Accuracy : 1          
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2845     
##     P-Value [Acc > NIR] : < 2.2e-16  
##                                      
##                   Kappa : 1          
##                                      
##  Mcnemar's Test P-Value : NA         
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

The XGB model also performed very well.

De-register the parallel processing cluster

```
stopCluster(cluster)
registerDoSEQ()
```

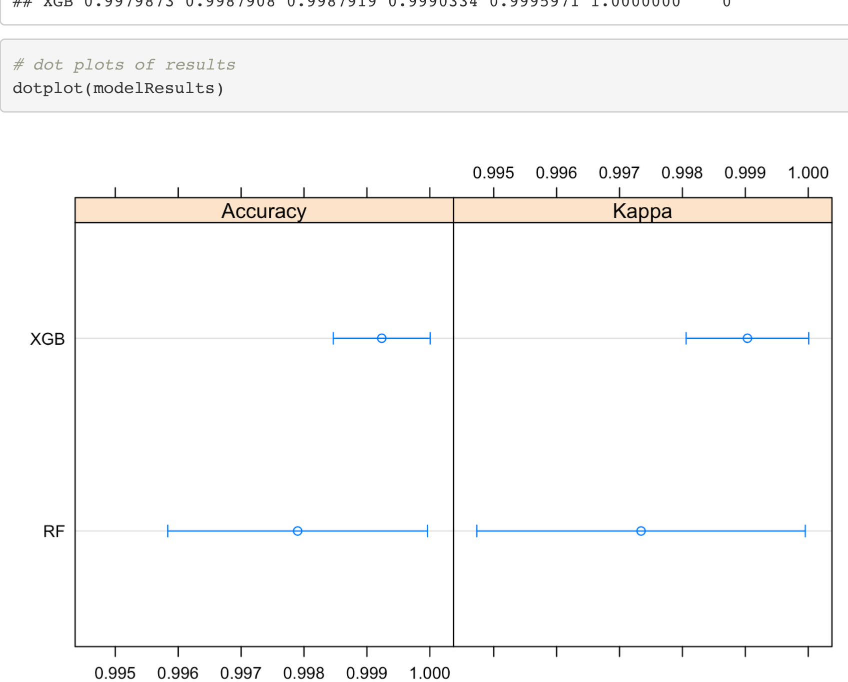
Comparing the models

```
# collect resamples
modelResults <- resamples(list(RF=rfModel, XGB=xgbModel))

# summarize the distributions
summary(modelResults)
```

```
## 
## Call:
## summary.resamples(object = modelResults)
## 
## Models: RF, XGB 
## Number of resamples: 5 
## 
## Accuracy 
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## RF  0.9952260 0.9974514 0.9984082 0.9978985 0.9990440 0.9993629    0
## XGB 0.9984087 0.9990440 0.9990449 0.9992358 0.9996814 1.0000000    0
## 
## Kappa 
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## RF  0.9939619 0.9967766 0.9979865 0.9973420 0.9987906 0.9991941    0
## XGB 0.9979873 0.9987908 0.9987919 0.9990334 0.9995971 1.0000000    0
```

```
# dot plots of results
dotplot(modelResults)
```



From the summary, we observe that both models performed similarly well. The XGB model performed marginally better than the Random Forest model.

Model Results on Test Data

Next we run both model on the test data.

```
testPrediction <- predict(rfModel, test)
testPrediction
```

```
##  [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
```

```
testPrediction <- predict(xgbModel, test)
testPrediction
```

```
##  [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
```

Note that both models produce the same prediction outcomes.