Practical Machine Learning Project

Yvonne

April 10, 2017

Executive Summary

The objective of this study is 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 project may use any of the other variables to predict with. The report describes how a model is built, how to do cross validation and justification of a good prediction model to use for predicting 20 different test cases. The result is also presented at the end of the report.

Preparing Data Files

The working directory will be set before the analysis. The given data files are downloaded and data is cleaned from any blank value. The data was further processed to remove the non-relevant rows & columns, and convert some fields to numerical type for easier processing.

```
setwd("C:/DS/C08")

# Download data files
if(!file.exists("pml-training.csv")){
    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
        destfile = "pml-training.csv", method = "curl")
}
if(!file.exists("pml-testing.csv")){
    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
        destfile = "pml-testing.csv", method = "curl")
}

# Clean blank values in data
train <- read.csv("pml-training.csv", header = TRUE, na.strings=c("","NA", "#DIV/0!"))
test <- read.csv("pml-testing.csv", header = TRUE, na.strings=c("","NA", "#DIV/0!"))

# Identify columns for this study.
# Calculate the percentage of NA's for each column and then to check the error percentage
NAPercent <- round(colMeans(is.na(train)), 2)
table(NAPercent)
```

```
## NAPercent
##    0 0.98    1
##   60   94    6
```

From the result, there are only 60 variables with complete data. Thefore, these 60 variables will be used to build the prediction algorithm. To further massage the data the following steps were performed: 1. The 1st variable is dropped as it is the row index from the csv file. 2. Find index of the complete columns minus the first and then subset the data based on index 3. Identify and remove the columns that are irrelevant for prediction. 4. Convert the fields to numeric for easy processing.

```
# Find index of the complete columns minus the first
index <- which(NAPercent==0)[-1]
```

```r
# Subset the data
train <- train[, index]
test <- test[, index]

# Find data column that are irrelevant to prediction
head(train)
```

```
##   user_name raw_timestamp_part_1 raw_timestamp_part_2 cvtd_timestamp
## 1  carlitos           1323084231               788290 05-12-11 11:23
## 2  carlitos           1323084231               808298 05-12-11 11:23
## 3  carlitos           1323084231               820366 05-12-11 11:23
## 4  carlitos           1323084232               120339 05-12-11 11:23
## 5  carlitos           1323084232               196328 05-12-11 11:23
## 6  carlitos           1323084232               304277 05-12-11 11:23
##   new_window num_window roll_belt pitch_belt yaw_belt total_accel_belt
## 1         no         11      1.41       8.07    -94.4                3
## 2         no         11      1.41       8.07    -94.4                3
## 3         no         11      1.42       8.07    -94.4                3
## 4         no         12      1.48       8.05    -94.4                3
## 5         no         12      1.48       8.07    -94.4                3
## 6         no         12      1.45       8.06    -94.4                3
##   gyros_belt_x gyros_belt_y gyros_belt_z accel_belt_x accel_belt_y
## 1         0.00         0.00        -0.02          -21            4
## 2         0.02         0.00        -0.02          -22            4
## 3         0.00         0.00        -0.02          -20            5
## 4         0.02         0.00        -0.03          -22            3
## 5         0.02         0.02        -0.02          -21            2
## 6         0.02         0.00        -0.02          -21            4
##   accel_belt_z magnet_belt_x magnet_belt_y magnet_belt_z roll_arm
## 1           22            -3           599          -313     -128
## 2           22            -7           608          -311     -128
## 3           23            -2           600          -305     -128
## 4           21            -6           604          -310     -128
## 5           24            -6           600          -302     -128
## 6           21             0           603          -312     -128
##   pitch_arm yaw_arm total_accel_arm gyros_arm_x gyros_arm_y gyros_arm_z
## 1      22.5    -161              34        0.00        0.00       -0.02
## 2      22.5    -161              34        0.02       -0.02       -0.02
## 3      22.5    -161              34        0.02       -0.02       -0.02
## 4      22.1    -161              34        0.02       -0.03        0.02
## 5      22.1    -161              34        0.00       -0.03        0.00
## 6      22.0    -161              34        0.02       -0.03        0.00
##   accel_arm_x accel_arm_y accel_arm_z magnet_arm_x magnet_arm_y
## 1        -288         109        -123         -368          337
## 2        -290         110        -125         -369          337
## 3        -289         110        -126         -368          344
## 4        -289         111        -123         -372          344
## 5        -289         111        -123         -374          337
## 6        -289         111        -122         -369          342
##   magnet_arm_z roll_dumbbell pitch_dumbbell yaw_dumbbell
## 1          516      13.05217      -70.49400    -84.87394
## 2          513      13.13074      -70.63751    -84.71065
## 3          513      12.85075      -70.27812    -85.14078
## 4          512      13.43120      -70.39379    -84.87363
```

```
## 5          506      13.37872      -70.42856    -84.85306
## 6          513      13.38246      -70.81759    -84.46500
##   total_accel_dumbbell gyros_dumbbell_x gyros_dumbbell_y gyros_dumbbell_z
## 1                   37                0            -0.02             0.00
## 2                   37                0            -0.02             0.00
## 3                   37                0            -0.02             0.00
## 4                   37                0            -0.02            -0.02
## 5                   37                0            -0.02             0.00
## 6                   37                0            -0.02             0.00
##   accel_dumbbell_x accel_dumbbell_y accel_dumbbell_z magnet_dumbbell_x
## 1             -234               47             -271              -559
## 2             -233               47             -269              -555
## 3             -232               46             -270              -561
## 4             -232               48             -269              -552
## 5             -233               48             -270              -554
## 6             -234               48             -269              -558
##   magnet_dumbbell_y magnet_dumbbell_z roll_forearm pitch_forearm
## 1               293               -65         28.4         -63.9
## 2               296               -64         28.3         -63.9
## 3               298               -63         28.3         -63.9
## 4               303               -60         28.1         -63.9
## 5               292               -68         28.0         -63.9
## 6               294               -66         27.9         -63.9
##   yaw_forearm total_accel_forearm gyros_forearm_x gyros_forearm_y
## 1        -153                  36            0.03            0.00
## 2        -153                  36            0.02            0.00
## 3        -152                  36            0.03           -0.02
## 4        -152                  36            0.02           -0.02
## 5        -152                  36            0.02            0.00
## 6        -152                  36            0.02           -0.02
##   gyros_forearm_z accel_forearm_x accel_forearm_y accel_forearm_z
## 1           -0.02             192             203            -215
## 2           -0.02             192             203            -216
## 3            0.00             196             204            -213
## 4            0.00             189             206            -214
## 5           -0.02             189             206            -214
## 6           -0.03             193             203            -215
##   magnet_forearm_x magnet_forearm_y magnet_forearm_z classe
## 1              -17              654              476      A
## 2              -18              661              473      A
## 3              -18              658              469      A
## 4              -16              658              469      A
## 5              -17              655              473      A
## 6               -9              660              478      A
```

```r
# The first six columns looks like related to users and they are unlikely to predict the activity.
train <- train[, -(1:6)]
test <- test[, -(1:6)]

for(i in 1:(length(train)-1)){
    train[,i] <- as.numeric(train[,i])
    test[,i] <- as.numeric(test[,i])
}
```

Perform Data Cross Validation

The train data set is splitted into 2 sets: 80% of data is used to train the model and 20% of data is used to validate the model.

```
# Split train data set to 2
inTrain <- createDataPartition(y=train$classe,p=0.8, list=FALSE)
trainData <- train[inTrain,]
validation <- train[-inTrain,]

# Print the dimentions of the 3 data sets
rbind(trainData = dim(trainData), validation = dim(validation), test = dim(test))
```

```
##             [,1] [,2]
## trainData  15699   53
## validation  3923   53
## test          20   53
```

Generate Prediction Models

We plan to use 2 most widely-used & most accurate prediction algorithm to generate the model. We review the accuracy rate and out-of-bag (OOB) error rates returned by the models as estimates for a food model. The prediction model are generated using (1) Random Forest(RF) Algorithm & (2) Generalized Boosted Regression (GBM) Model . The RF model is computationally intensive, we will leverage the parallel processing using multiple cores through the doMC package

```
# Generate rf fit model
registerDoMC(cores = 8)
rfFit <- randomForest(classe~., data = trainData, method ="rf", prox = TRUE)

## Generate gbm model
gbmFit <- train(classe~., data = trainData, method ="gbm", verbose = FALSE)
```

```
## Loading required package: gbm

## Warning: package 'gbm' was built under R version 3.3.3

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: splines

## Loaded gbm 2.1.3
```

Predicting and validating model using both rf and gbm

```
# use rf model to predict on validation data set
rfFit
```

```
##
## Call:
```

```
##  randomForest(formula = classe ~ ., data = trainData, method = "rf",      prox = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.32%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4461    3    0    0    0 0.000672043
## B    8 3027    3    0    0 0.003620803
## C    0    8 2727    3    0 0.004017531
## D    0    0   16 2555    2 0.006995725
## E    0    0    1    7 2878 0.002772003
```

```
rfPred <- predict(rfFit, validation)
confusionMatrix(rfPred, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1114    1    0    0    0
##          B    1  758    2    0    0
##          C    0    0  682   10    0
##          D    0    0    0  633    1
##          E    1    0    0    0  720
##
## Overall Statistics
##
##                Accuracy : 0.9959
##                  95% CI : (0.9934, 0.9977)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9948
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9987   0.9971   0.9844   0.9986
## Specificity            0.9996   0.9991   0.9969   0.9997   0.9997
## Pos Pred Value         0.9991   0.9961   0.9855   0.9984   0.9986
## Neg Pred Value         0.9993   0.9997   0.9994   0.9970   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1932   0.1738   0.1614   0.1835
## Detection Prevalence   0.2842   0.1940   0.1764   0.1616   0.1838
## Balanced Accuracy      0.9989   0.9989   0.9970   0.9921   0.9992
```

```
# use gbm model to predict on validation data set
gbmFit
```

```
## Stochastic Gradient Boosting
##
## 15699 samples
```

```
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 15699, 15699, 15699, 15699, 15699, 15699, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7438094  0.6751491
##   1                  100      0.8125054  0.7626874
##   1                  150      0.8473524  0.8068713
##   2                   50      0.8484445  0.8080149
##   2                  100      0.9043558  0.8789690
##   2                  150      0.9291146  0.9103093
##   3                   50      0.8918599  0.8631053
##   3                  100      0.9395402  0.9235038
##   3                  150      0.9591056  0.9482661
##
## 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 = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
gbmPred <- predict(gbmFit, validation)
confusionMatrix(gbmPred, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1098   26    0    1    2
##          B   14  712   12    2    8
##          C    2   20  666   25    3
##          D    1    1    5  612    6
##          E    1    0    1    3  702
##
## Overall Statistics
##
##                Accuracy : 0.9661
##                  95% CI : (0.9599, 0.9715)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9571
##  Mcnemar's Test P-Value : 0.0004667
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9839   0.9381   0.9737   0.9518   0.9736
## Specificity            0.9897   0.9886   0.9846   0.9960   0.9984
## Pos Pred Value         0.9743   0.9519   0.9302   0.9792   0.9929
```

```
## Neg Pred Value         0.9936   0.9852   0.9944   0.9906   0.9941
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2799   0.1815   0.1698   0.1560   0.1789
## Detection Prevalence   0.2873   0.1907   0.1825   0.1593   0.1802
## Balanced Accuracy      0.9868   0.9633   0.9791   0.9739   0.9860
```

Result

RF MODEL shows that the model accuracy is 99.7% & the OOB estimate of error rate is 0.41%. GBM MODEL shows that the accuracy is 96.9% Therefore, we conclude that the RF model has higher accuracy percentage, it appears to be a better model and it will be used for subsequent predictions.

Predicting 20 given test data

Apply Random Forest model to test set

```r
predict(rfFit, test)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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 predictioin result was all correct using the RF model.