Practical Machine Learning Project

Sam Mengistu

May 14, 2018

Practical Machine Learning - Peer Assessment

1. Data Overview

Three machine learning algorithms were implemented to predict "how well" people perform weight lifting activities using accelerometer acquired data sets. The accelerometers were placed on the belt, forearm, arm, and dumbbell of six male health participants during weight lifting exercises using a relatively light dumbbell (1.25kg). The participants were between 20-28 years old and had little weight lifting experience.

The six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). Various variables pertinent to this analysis were collected and made available as training and testing datasets.

Training data sets are available here while testing data set can be found here. Additional details about the project data can be also found here.

2. Objective

The objective of this analysis was to predict the quality of weight lifting activities by six wearers of accelerometer with little weight lifting experience using data sets acquired by accelerometers placed at different locations.

3. Installing Required Packages and Libraries

The following packages and libraries may be required to perform data preprocessing and model building.

```
library(caret)
library(randomForest)
library(corrplot)
library(rpart)
library(RColorBrewer)
library(rattle)
library(htmltools)
```

4. Data Acquisition and Preprocessing

4.1. *Downloading data*

The training and testing datasets were downloaded from the links provided above. A working directory was set first inside which a folder named "PRassignment" was created to store the downloaded datasets.

```
setwd("/home/gis/EPA/GitHub/TSTools/ML/PracticalMachineLearning")
```

```
# creating a new project directory
if (!file.exists("./Assignment_Data")) {
dir.create("./Assignment_Data")
}
```

```
# downloading training and testing datasets
trainUrl <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(trainUrl, destfile ="./Assignment_Data/trainingData.csv")
download.file(testUrl, destfile ="./Assignment_Data/testingData.csv")
```

4.2. Loading data

Downloaded datasets (both of them in a .csv format) were then loaded to R environment for subsequent preprocessing and analysis.

```
# loading training and testing datasets
trainData <- read.csv("./Assignment_Data/trainingData.csv", header =T, sep=",",na.strings=c("NA",""))
testData <- read.csv("./Assignment_Data/testingData.csv", header =T, sep=",", na.strings=c("NA",""))
```

4.3. Cleaning data

Data cleaning may include removing unwanted features (columns) from the datasets, identifying less variable or less meaningful predictors, and dealing with NA values, including imputing missing data. First, unwanted variable columns were removed from both the training and testing datasets followed by variables with close to zero variance. It is always difficult to deal with missing data. The question of how many missing values is enough to keep or remove the variable for subsequent modeling analysis is dependent on a several factors. In this exercise, the columns with more than 60 percent missing values were dropped while imputation was planned for variables with less than 60 percent missing observations to avoid problems in training models.

```
# removing unwanted variable or feature columns
unwanted.cols <-c("X", "user_name","raw_timestamp_part_1","raw_timestamp_part_2","cvtd_timestamp","num_
wanted.cols<-!names(trainData)%in%unwanted.cols
training.data<- trainData [, wanted.cols]
testing.data<- testData[, wanted.cols]
```

```
# removing unimportant predictors or variables
nsv <- nearZeroVar(training.data,saveMetrics=TRUE)
training.data.nzv <- training.data[,!nsv$nzv]
testing.data.nzv <- testing.data[,!nsv$nzv]
```

```
# removing variables with more than 60% NA values
na.lessthan60p <- which(colSums(is.na(training.data.nzv))<0.6*nrow(training.data.nzv))
training.data.nzvCln<-  training.data.nzv[ ,na.lessthan60p]
testing.data.nzvCln <- testing.data.nzv [,na.lessthan60p]
testing <- testing.data.nzvCln
```

4.4. Partitioning data

After cleaning, the downloaded training dataset was partitioned in two categories to create a training set (70% of the data) for model training and a validating set (with the remaining 30%) for model validation. The trained model, after being validated, was used for prediction using cleaned testing dataset.

```
# partitioning data to training and validating datasets
inTrain = createDataPartition(training.data.nzvCln$classe, p=0.70, list=FALSE)
training = training.data.nzvCln[inTrain,]
validating = training.data.nzvCln[-inTrain,]
```

No NA values in all the datasets suggesting no requirement of imputation.

```
sum(colSums(is.na(training)))
```

```
## [1] 0
```

```
sum(colSums(is.na(validating)))
```

```
## [1] 0
```

```
sum(colSums(is.na(testing)))
```

```
## [1] 0
```

5. Correlation Analysis

The training data with the exclusion of the classe feature was subjected to correlation analysis to evaluate how predictor variables are correlated to each other. The correlation graph is included in the pdf version of this report (not included in the HTML file to limit the size of this HTML document below 1MB, beyond which rendered HTML files can only be viewed as raw HTML even if the file is pusehd to the gh-pages branch). To come up with more compact predictors, the data can be also subjected to a Principal Component Analysis (PCA) based pre-processing. According to the correlation plot, fewer variables with higher correlations are observed, therefore, the PCA based preprocessing has been skipped for this exercise.

```
# performing correlation analysis
corAnalysis <- cor(training [, -53])
corrplot(corAnalysis, order = "hclust", method = "circle", type = "lower",  tl.cex = 0.52, cl.cex = 1, c
```

6. Modeling

Three machine learning algorithms were applied to generate models that fit to the training dataset. These include Random Forests, Decision Tree and Generalized Boosted Model. A seed value was declared (set.seed(1345)) to aid reproducibility of analysis outcomes. The fitted models were then tested on validating and testing datasets. A Confusion Metrix between the observed and predicted classe values for the validating data sets was revealed at the end of each analysis to aid better visualization of the accuracy of modeling results.

6.1. Random Forest

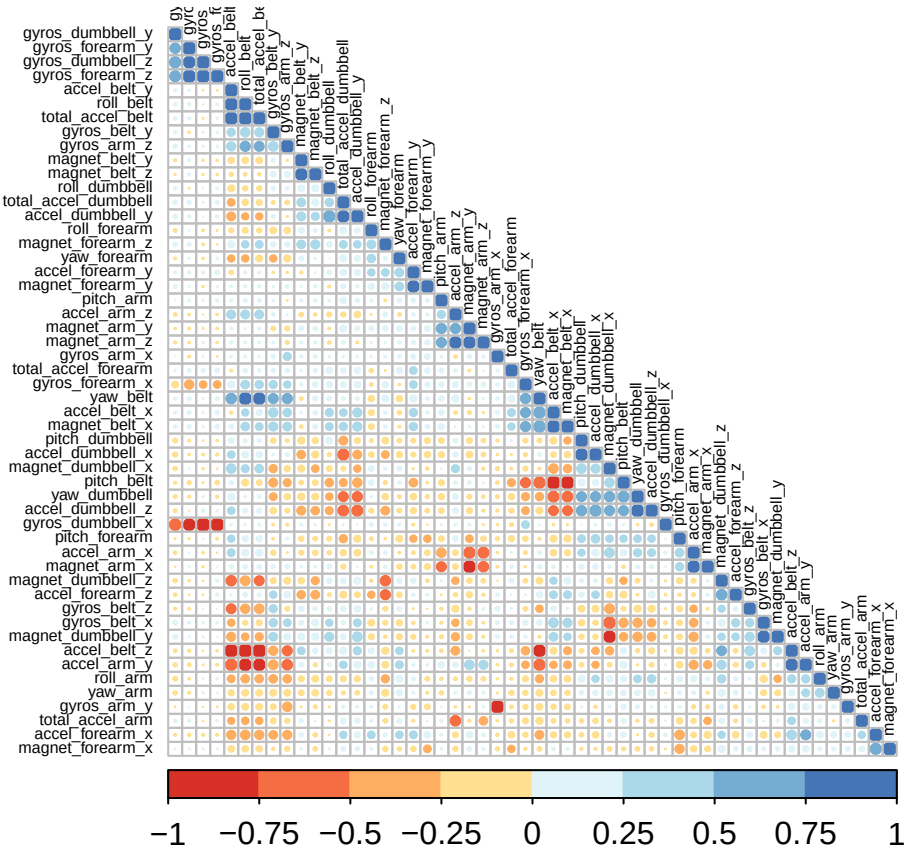

Figure 1: Correlation analysis of training data

```r
set.seed(1345)
# model training
rfMod.fit <- train(classe ~ ., method = "rf", data = training, importance = T, trControl = trainControl
rfMod.fit
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 10304, 10301, 10303, 10303
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9903182  0.9877520
##   27    0.9901724  0.9875677
##   52    0.9834027  0.9790044
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```r
# model validation
rfValid.pred <- predict(rfMod.fit, newdata=validating)

confusionMatrix(rfValid.pred,validating$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    1 1137    2    0    0
##          C    0    1 1023    4    0
##          D    0    0    1  960    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9983
##                  95% CI : (0.9969, 0.9992)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9979
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            0.9994   0.9982   0.9971   0.9959   1.0000
## Specificity            0.9998   0.9994   0.9990   0.9998   1.0000
## Pos Pred Value         0.9994   0.9974   0.9951   0.9990   1.0000
## Neg Pred Value         0.9998   0.9996   0.9994   0.9992   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1932   0.1738   0.1631   0.1839
## Detection Prevalence   0.2845   0.1937   0.1747   0.1633   0.1839
## Balanced Accuracy      0.9996   0.9988   0.9980   0.9978   1.0000
```

6.2. Decision Tree

```
# model training
dtreeCtrl <- trainControl(method = "repeatedcv", number = 10, repeats = 4)
set.seed(1345)
rpartMod.fit <- train(classe ~ ., method = "rpart", data = training,  trControl=dtreeCtrl, tuneLength =

rpartMod.fit
```

```
## CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 4 times)
## Summary of sample sizes: 12363, 12366, 12363, 12362, 12362, 12364, ...
## Resampling results across tuning parameters:
##
##   cp           Accuracy   Kappa
##   0.009968467  0.6806823  0.5981343
##   0.012918320  0.6675587  0.5818140
##   0.019224901  0.6163390  0.5140086
##   0.020954125  0.6018682  0.4952385
##   0.021361001  0.5988300  0.4913424
##   0.029803682  0.5272313  0.3917286
##   0.030210558  0.5221920  0.3849541
##   0.042315126  0.5024319  0.3562891
##   0.050503509  0.3855059  0.1620039
##   0.114128776  0.3373569  0.0810896
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.009968467.
```

```
# model validation
rpartValid.pred <- predict(rpartMod.fit, newdata=validating)
confusionMatrix(rpartValid.pred,validating$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction    A    B    C    D    E
##          A 1367  209   16   39   31
##          B   41  576  120   83  127
##          C   75  117  718  147   95
##          D  174  237  172  671  214
##          E   17    0    0   24  615
##
## Overall Statistics
##
##                Accuracy : 0.6707
##                  95% CI : (0.6585, 0.6827)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5847
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8166  0.50571   0.6998   0.6961   0.5684
## Specificity            0.9299  0.92183   0.9107   0.8380   0.9915
## Pos Pred Value         0.8225  0.60824   0.6233   0.4571   0.9375
## Neg Pred Value         0.9273  0.88599   0.9349   0.9337   0.9107
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2323  0.09788   0.1220   0.1140   0.1045
## Detection Prevalence   0.2824  0.16092   0.1958   0.2494   0.1115
## Balanced Accuracy      0.8733  0.71377   0.8052   0.7670   0.7799
```

6.3. Generalized Boosted Model

```
# model training
set.seed(1345)
controlGBM <- trainControl(method = "repeatedcv", number = 10, repeats = 4)
gbmMod.fit <- train(classe~., data=training, method = "gbm", trControl = controlGBM, verbose = FALSE)

gbmMod.fit
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 4 times)
## Summary of sample sizes: 12363, 12366, 12363, 12362, 12362, 12364, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7537875  0.6879354
##   1                  100      0.8225790  0.7755041
```

```
##   1                  150      0.8546643  0.8161376
##   2                   50      0.8568838  0.8187418
##   2                  100      0.9054033  0.8803076
##   2                  150      0.9303173  0.9118349
##   3                   50      0.8969412  0.8695546
##   3                  100      0.9416191  0.9261367
##   3                  150      0.9613106  0.9510524
##
## 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.
```

```
# model validation
gbmValid.pred <- predict(gbmMod.fit, newdata=validating)

confusionMatrix(gbmValid.pred,validating$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1646   24    0    0    5
##          B   17 1094   23    0    5
##          C    8   21  991   15    8
##          D    3    0   11  944    8
##          E    0    0    1    5 1056
##
## Overall Statistics
##
##                Accuracy : 0.9738
##                  95% CI : (0.9694, 0.9778)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9669
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9833   0.9605   0.9659   0.9793   0.9760
## Specificity            0.9931   0.9905   0.9893   0.9955   0.9988
## Pos Pred Value         0.9827   0.9605   0.9501   0.9772   0.9944
## Neg Pred Value         0.9933   0.9905   0.9928   0.9959   0.9946
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2797   0.1859   0.1684   0.1604   0.1794
## Detection Prevalence   0.2846   0.1935   0.1772   0.1641   0.1805
## Balanced Accuracy      0.9882   0.9755   0.9776   0.9874   0.9874
```

6.4. Model selection

Both Random Forest (RF) and Generalized Boosted Models (GBM) algorithms provided high accuracy. The result tables above show model generated accuracies of 0.99, 0.96 and 0.67 for RF, GBM and Decision Tree (DT) models.

```
# model testing
rfTest.pred <- predict(rfMod.fit, newdata=testing)

rfTest.pred
```

```
##  [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
```

In addition, RF and GBM models provided the same prediction of the classe features in test data (see the tables below).

```
# comparison of RF and GBM predictions of testing data

gbmTest.pred <- predict(gbmMod.fit, newdata=testing)

table(rfTest.pred, gbmTest.pred)
```

```
##             gbmTest.pred
## rfTest.pred A B C D E
##           A 7 0 0 0 0
##           B 0 8 0 0 0
##           C 0 0 1 0 0
##           D 0 0 0 1 0
##           E 0 0 0 0 3
```

Because the accuracy of RF model was slightly higher, RF was chosen as a predictive model. The top three important variables for the RF based predictive model include 'yaw_belt', pitch_belt', and 'roll_belt'. Figure 2 below summarized the importance of predictor variables graphically.

```
# importance of variables
varImpPlot(rfMod.fit$finalModel, sort = TRUE, type = 1, pch = 19, col = 'blue', cex = 0.8, main = " ")
```

7. Summary

The study involved three machine learning approaches to predict how well people perform weight lifting activities from accelerometer acquired data sets. The three models were subjected to same datasets that were cleaned and preprocessed during this investigation. Random Forest provided the best model with a slight difference from Generalized Boosted Model in terms of accuracy, thus used for prediction.

References:

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

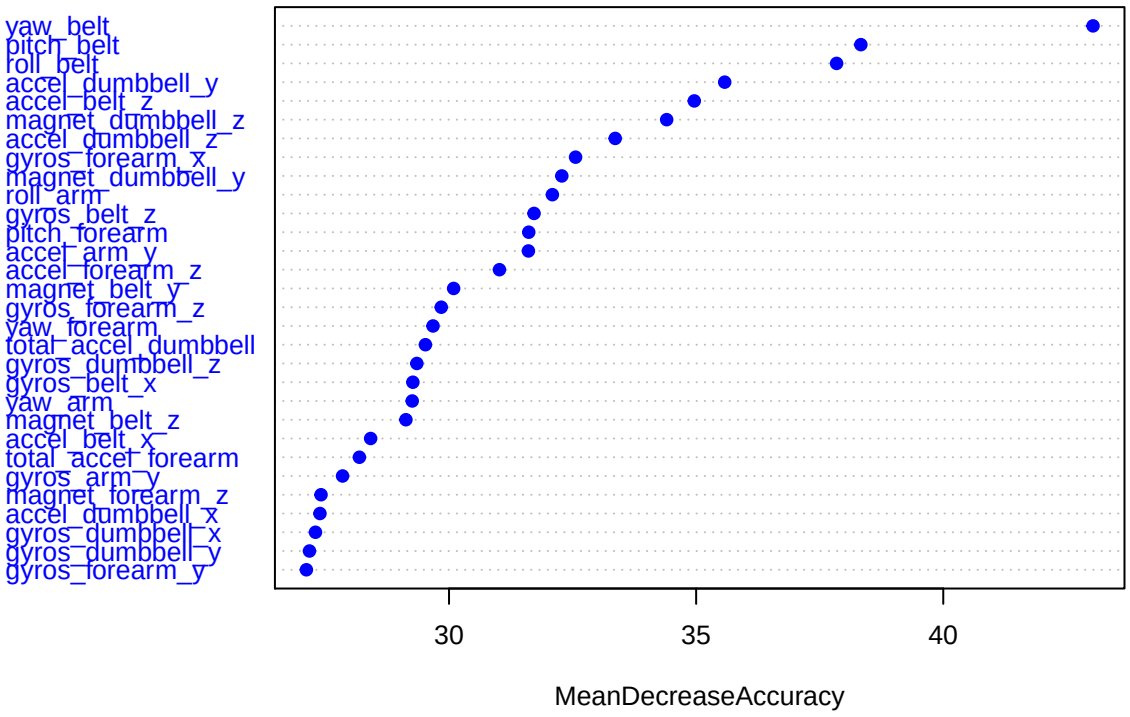

Figure 2: Importance of predictor variables