Coursera Practical Machine Leaning Project

Robert DeSonier

December 26, 2015

Executive Summary

This report is my submission for the Peer Assessment for the Cousera Class “Practical Machine Learning Project”. The assignment is to analyze the training data set and provide predictions of the manner in which the exercise was completed (the “classe” variable in the training data set).

The predictions are submitted separately on the Coursera site.

Methods and Background

The report was prepared using **R markdown** (R Studio rmd file) and processed by **knitr** and then transformed into a **HTML** file. Besides base R also used the library packages **readr**, **dplyr**, **caret** and **doParallel** for this work.

Data Processing

Raw data was downloaded from the URLs provided on the assignment webpage.

```
# Convert the raw data files into data frames and rename

trainingData1 <- read_csv("pml-training.csv",
         col_names = TRUE, col_types = NULL, na = c("", "NA"), comment = "",
         trim_ws = TRUE, skip = 0, n_max = -1, progress = interactive())

testData1 <- read_csv("pml-testing.csv",
         col_names = TRUE, col_types = NULL, na = c("", "NA"), comment = "",
         trim_ws = TRUE, skip = 0, n_max = -1, progress = interactive())
```

The training data was examined using basic R tools and the following “housekeeping” completed. After this cleaning the training data was then separated into two files for model development.

```
##  Housekeeping
##  Remove "new window" rows
trainingData2 <- filter(trainingData1, trainingData1$new_window != "yes")

##  Remove all NA columns
trainingData3 <- trainingData2[ , !apply( trainingData2, 2, function(x) all(is.na(x)))]

##  Remove timestamp data
trainingData4 <- select(trainingData3, -user_name, -raw_timestamp_part_1, -raw_timestam
p_part_2, -cvtd_timestamp,-new_window, -num_window)

##  convert to data frame
trainingData5 <- as.data.frame(trainingData4)

##  remove index column
trainingData5[,1] <- NULL

##  need to use factors as output for model
trainingData5$classe <- as.factor(trainingData5$classe)

##  divide data into training_data and training_test
set.seed(3456)
testIndex = createDataPartition(trainingData5$classe, p = 0.40,list=FALSE, times = 1)
training_data = trainingData5[-testIndex,]
testing_data = trainingData5[testIndex,]
```

Model Development

Due to the nature of the problem the “random forest” approach using the caret package was selected. To improve the model three fold cross validation was implemented. Aside, needed to use parallel processing to reach three fold.

```
registerDoParallel(cores = 4)
model <- train(training_data$classe ~., data = training_data, method = "rf",
                trControl = trainControl(method = "cv",number = 3),
                prox = TRUE, allowParallel = TRUE)
```

Model Summary

The resulting model is summarized below.

```
model
```

```
## Random Forest
##
## 11527 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 7684, 7684, 7686
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD   Kappa SD
##    2    0.9849046  0.9808981  0.0034058580  0.0043112697
##   27    0.9862932  0.9826565  0.0006512685  0.0008232852
##   52    0.9802201  0.9749719  0.0013832193  0.0017495620
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
model$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, proximity = TRUE,      allowParallel =
TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.88%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3277    3    1    0    1 0.001523461
## B   24 2198    8    0    0 0.014349776
## C    0   13 1988   10    0 0.011437096
## D    0    1   24 1862    1 0.013771186
## E    0    1    6    8 2101 0.007088847
```

Model Performance

Confusion matrix was calculated using model and the testing data

```
modPredTrain <- predict.train(model, testing_data)
confusionMatrix(modPredTrain, testing_data$classe[2:7689])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2184   20    0    0    0
##          B    2 1467   16    0    0
##          C    2    1 1320   13    6
##          D    0    0    5 1244    4
##          E    0    0    0    2 1402
##
## Overall Statistics
##
##                Accuracy : 0.9908
##                  95% CI : (0.9884, 0.9928)
##     No Information Rate : 0.2846
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9883
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9859   0.9843   0.9881   0.9929
## Specificity            0.9964   0.9971   0.9965   0.9986   0.9997
## Pos Pred Value         0.9909   0.9879   0.9836   0.9928   0.9986
## Neg Pred Value         0.9993   0.9966   0.9967   0.9977   0.9984
## Prevalence             0.2846   0.1935   0.1744   0.1638   0.1837
## Detection Rate         0.2841   0.1908   0.1717   0.1618   0.1824
## Detection Prevalence   0.2867   0.1932   0.1746   0.1630   0.1826
## Balanced Accuracy      0.9973   0.9915   0.9904   0.9933   0.9963
```

Submission

The same housekeeping steps applied to the training data were applied to the test data and then predicted “classe” submitted for the class. Resulting total score was 19/20.