Practical Machine Learning Course Project

Kevin Bitinsky

03/03/2020

Executive Summary

This project is part of the Practical Machine Learning Class offered by Coursera and Johns Hopkins University. The goal is to predict the *classe* variable, which is the manner (qualitatively, “how well”) a participant performed the exercise. This is to be predicted using sensor data obtained from the chest, wrist, bicep, wrist, and dumbell of a participant during a dumbell bicep curl.

Background

Using devies such as activity trackers or smartphones it is now possible to collect a large amount of data about personal activity relatively inexpensively. Hpwever, one thing that people regularly do is quantify **how much** of a particular activity they do, but they rarely quantify **how well** they do it. In this project, the goal will be to use data from accelerometers and gyroscopes on the belt, forearm, arm, and dumbell of 6 participants. The participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways. * Class A - exactly according to the specification * Class B - throwing the elbows to the front * Class C - lifting the dumbbell only halfway * Class D - lowering the dumbbell only halfway * Class E - throwing the hips to the front

All the data is courtesy of: http://groupware.les.inf.puc-rio.br/har For more information see the section on the Weight Lifting Exercise Dataset.

Data

Data Extration

```
train_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

train <- read_csv(train_url, na = c("","NA","#DIV/0!"))
test <- read_csv(test_url, na = c("","NA","#DIV/0!"))
```

Enable Parallel Processing

Parallel processing is useful in speeding up the models.

```
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)
```

```
# Configure trainControl
fitControl <- trainControl(method = "cv",
                           number = 5,
                           allowParallel = TRUE)
```

Data Cleanup

Set the response variable to be a factor, and remove variables that are not predictors.

```
# Set response var, $classe, to be factor
train$classe <- factor(train$classe)

# Remove summarization rows; new_window == "yes"
tidy_data <- subset(train, new_window == "no")

# Remove columns with no useful data; all NAs
tidy_data <- Filter(function(x)!all(is.na(x)), tidy_data)

# Remove first seven columns of identification information
tidy_data <- tidy_data[, -(1:7)]
```

Data Splitting

Split the original training set into Training and Validation sets.

```
# split the data
set.seed(2020)
index <- createDataPartition(tidy_data$classe, p=0.75, list = FALSE)
training_set <- tidy_data[index, ]
validation_set <- tidy_data[-index, ]
```

Confirmation of split

Confirming that the data split of response cariables is consistent across classe for each data set.

```
a <- train %>% group_by(classe) %>% summarize(n=n()) %>% mutate(per = n/sum(n))
b <- tidy_data %>% group_by(classe) %>% summarize(n=n()) %>% mutate(per = n/sum(n))
c <- training_set %>% group_by(classe) %>% summarize(n=n()) %>% mutate(per = n/sum(n))
d <- validation_set %>% group_by(classe) %>% summarize(n=n()) %>% mutate(per = n/sum(n))
summary<-cbind(a,b$n, b$per ,c$n, c$per,d$n, d$per)
names(summary) <- c("classe","original#","original%","tidy#","tidy%","training#","training%","validatio
summary
```

```
##   classe original# original% tidy#     tidy% training# training% validation#
## 1      A      5580 0.2843747  5471 0.2847107      4104 0.2847232        1367
## 2      B      3797 0.1935073  3718 0.1934846      2789 0.1934924         929
## 3      C      3422 0.1743961  3352 0.1744380      2514 0.1744138         838
## 4      D      3216 0.1638977  3147 0.1637698      2361 0.1637991         786
## 5      E      3607 0.1838243  3528 0.1835970      2646 0.1835715         882
##   validation%
```

```
## 1   0.2846731
## 2   0.1934611
## 3   0.1745106
## 4   0.1636818
## 5   0.1836735
```

Analysis and Model Generation

For this assignment I followed the models as they are presented by: Datacamp https://campus.datacamp.com/courses/machine-learning-with-tree-based-models-in-r So the paradigm may differ from those presented in this course. But the outcomes should be similar.

Classification Tree Model

```
# Classification tree model
tree <- rpart(classe~., data = training_set, method = "class")

# Prediction
pred_tree <- predict(tree, newdata = validation_set, type = "class")

# Confusion Matrix
conf_tree <- confusionMatrix(data = pred_tree, reference = validation_set$classe)
conf_tree$overall[1]
```

```
##  Accuracy
## 0.7469804
```

This has an accuracy of 74.7%

Bagged Model

```
tree2 <- rpart(classe~., data = training_set, method = "class", control = rpart.control(cp = 0.0001))
# printcp(tree2)

# Prune the tree
bestcp <- tree2$cptable[which.min(tree2$cptable[,"xerror"]),"CP"]
tree.pruned <- prune(tree2, cp = bestcp)

# Prediction
pred_bagged <- predict(tree.pruned, newdata = validation_set, type = "class")

# Confusion Matrix
conf_bagged <- confusionMatrix(data = pred_bagged, reference = validation_set$classe)
conf_bagged$overall[1]
```

```
##  Accuracy
## 0.9285714
```

This has an accuracy of 92.86%

Random Forest

There is much discussion on the poor performance of 'Party', the library used in train for method = 'rf'. Instead, the suggestion is to use randomForest.

```
rf <- randomForest(formula = classe ~ ., data = training_set, trControl = fitControl)
pred_rf <- predict(rf, newdata = validation_set)
conf_rf <- confusionMatrix(pred_rf, validation_set$classe)
conf_rf$overall[1]
```

```
##  Accuracy
## 0.9956268
```

This has an accuracy of 99.56%

Generalized Boosted Model (GBM)

```
gbm <- train(classe ~ ., method="gbm",data=training_set, verbose=FALSE,trControl = fitControl)
pred_gbm <- predict(gbm, newdata = validation_set, n.trees = num_trees,type = "raw")
conf_gbm <- confusionMatrix(pred_gbm, validation_set$classe)
conf_gbm$overall[1]
```

```
##  Accuracy
## 0.9625156
```

This has an accuracy of 96.25%

Prediction of Test Data

Based upon the model accuracies, choose the Random Forest model because it yielded the highest score.

```
predict(rf, newdata = 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
```

Clean-up

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