Exercise Classification Study

Eric Johnson

10/6/2020

Exercise Classification Study

This analysis focuses on the quality of different exercises performed. Several different models were tested in order to determine the best fit from the predictors. The best resulting model was a random forest model which yielded a predicted out of sample error of approximately 0.10%

Getting and Cleaning Data

The first step is to get the data loaded in and to appropriately classify it.

```
all.data <- read.csv("./pml-training.csv", na.strings = c("NA", "#DIV/0!"))
## set Class type for all.data
all.data$user_name <- as.factor(all.data$user_name)
all.data$cvtd_timestamp <- dmy_hm(all.data$cvtd_timestamp)
all.data$new_window <- as.factor(all.data$new_window)
all.data$classe <- as.factor(all.data$classe)
```

Unfortunately many of the data variables have very little data in them. In order to prevent this from skewing our model we will exclude the variables that have less than 10% data.

```
sparse <- function(x) { sum(is.na(x)) }
percent.na <- round(apply(all.data, 2, sparse)/dim(all.data)[1],2)
include.col <- percent.na < 0.1
all.data <- all.data[ , include.col]
## Eliminating irrelevant columns such as time and user which should not be predictive
## in real-world settings
all.data <- all.data[ , 7:60]
```

Finally we'll split the data into a few different data sets: * train.data - This will contain 80% of the total data set for the final model training * train.data.small - This will contain 1/8th of the training data (10% of the full data), in order to do some preliminary analysis. * verify.data - This will contain the 20% of the data not in the training dataset in order to calculate the predicted out of sample error rate.

```
set.seed(8178)

split.index <- createDataPartition(y = all.data$classe, p = 0.8, list = FALSE)
train.data <- all.data[split.index, ]
split.index.small <- createDataPartition(y = train.data$classe, p = 1/8, list = FALSE)
train.data.small <- train.data[split.index.small, ]

verify.data <- all.data[-split.index, ]
```

Preliminary Model Analysis

We'll start the analysis by looking at the training a few different models and seeing how well they aling to their own training data.

```
rf.model.small <- train(classe ~ ., method = "rf", data = train.data.small,
                        trControl = trainControl(method="cv"), number = 3)
rpart.model.small <- train(classe ~ ., method = "rpart", data = train.data.small)
lda.model.small <- train(classe ~ ., method = "lda", data = train.data.small)
nb.model.small <- train(classe ~ ., method = "nb", data = train.data.small)
# gbm.model.small <- train(classe ~ ., method = "gbm", data = train.data.small, verbose = FALSE)
```

Next let's look at how well the models based on the small datasets predict the remainder of the training dataset.

```
OOS.Accuracy <- NULL
rf.pred <- predict(rf.model.small, train.data[-split.index.small, ])
OOS.Accuracy$rf <- mean(rf.pred == train.data[-split.index.small,"classe"])

rpart.pred <- predict(rpart.model.small, train.data[-split.index.small, ])
OOS.Accuracy$rpart <- mean(rpart.pred == train.data[-split.index.small,"classe"])

lda.pred <- predict(lda.model.small, train.data[-split.index.small, ])
OOS.Accuracy$lda <- mean(lda.pred == train.data[-split.index.small,"classe"])

nb.pred <- predict(nb.model.small, train.data[-split.index.small, ])
OOS.Accuracy$nb <- mean(nb.pred == train.data[-split.index.small,"classe"])

# gbm.pred <- predict(gbm.model.small, train.data[-split.index.small, ])
# OOS.Accuracy$gbm <- mean(gbm.pred == train.data[-split.index.small,"classe"])

print(OOS.Accuracy)
```

```
## $rf
## [1] 0.9676738
##
## $rpart
## [1] 0.489625
##
## $lda
## [1] 0.7067346
##
## $nb
## [1] 0.7223153
```

Based on these results the random forest model has the most accurate predictions. As a result we will move forward with the random forest model.

Full Model Creation

The full model will be created in the same manner as the preliminary models.

```
OOS.Accuracy.Full <- NULL
rf.model <- train(classe ~ ., method = "rf", data = train.data,
                                trControl = trainControl(method="cv"), number = 3)
rf.pred.full <- predict(rf.model, verify.data)
OOS.Accuracy.Full$rf <- mean(rf.pred.full == verify.data$classe)
print(OOS.Accuracy.Full)
```

```
## $rf
## [1] 0.9987255
```

Based on this the predicted out of sample error rate is: 0.127%. Considering the size of the data set this appears to be a fairly accurate model.