Machine Learning Project

Duncan Sallstrom

September 18, 2016

This is my submission for the Machine Learning final project.

Synopsis:

Using accelerometer and other movement-related data attached to individuals performing a bicep curl exercise, we were asked to predict the outcome of the exercise using standard machine learning processes.

R was used for this project. As its focus was primarily machine learning, the project leans heavily on the CARET package. The report also makes use of the dplyr, sqldf, and ggplot2 packages in smaller degrees.

Data

I was provided with movement data. Each observation has several numbers related to study participants' movements, timestamps, and the user_names, as well as the outcome variable. The outcome variable – names "classe"" – describes whether or not the participant performed the exercise correctly, and if the participant does not perform the exercise correctly, how he or she failed. It has five outcomes: A, B, C, D, and E.

The data used was the "Weight Lifting Exercises" dataset, documented at URL http://groupware.les.inf.puc-rio.br/har. See the bottom of the report for a full citation.

```
# Load training and testing data
training <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
testing <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
```

It's worth noting that the data contains variables that are statistics. These are frequently missing – plus it does not make intuitive sense to include them in a predicitve model, since they just reflect other data already in the raw data. Machine learning models also function best when complete observations are fed into the models and no assumptions are made to fill-in empty missing data.

I remove these "stats vars" here. Variable "X", which is just the observation number, is also removed. The code chunk here also outputs the variable names that will be used for model training.

```
statsVars <- grep("kurtosis|max|min|skewness|avg|stddev|var|amplitude", value=TRUE, names(training))
trainingEdit <- training[, !names(training) %in% c(statsVars, "X")]

# Preview variable names actually used
names(trainingEdit)
```

```
##  [1] "user_name"           "raw_timestamp_part_1" "raw_timestamp_part_2"
##  [4] "cvtd_timestamp"      "new_window"           "num_window"
##  [7] "roll_belt"           "pitch_belt"           "yaw_belt"
## [10] "total_accel_belt"    "gyros_belt_x"         "gyros_belt_y"
## [13] "gyros_belt_z"        "accel_belt_x"         "accel_belt_y"
## [16] "accel_belt_z"        "magnet_belt_x"        "magnet_belt_y"
## [19] "magnet_belt_z"       "roll_arm"             "pitch_arm"
## [22] "yaw_arm"             "total_accel_arm"      "gyros_arm_x"
```

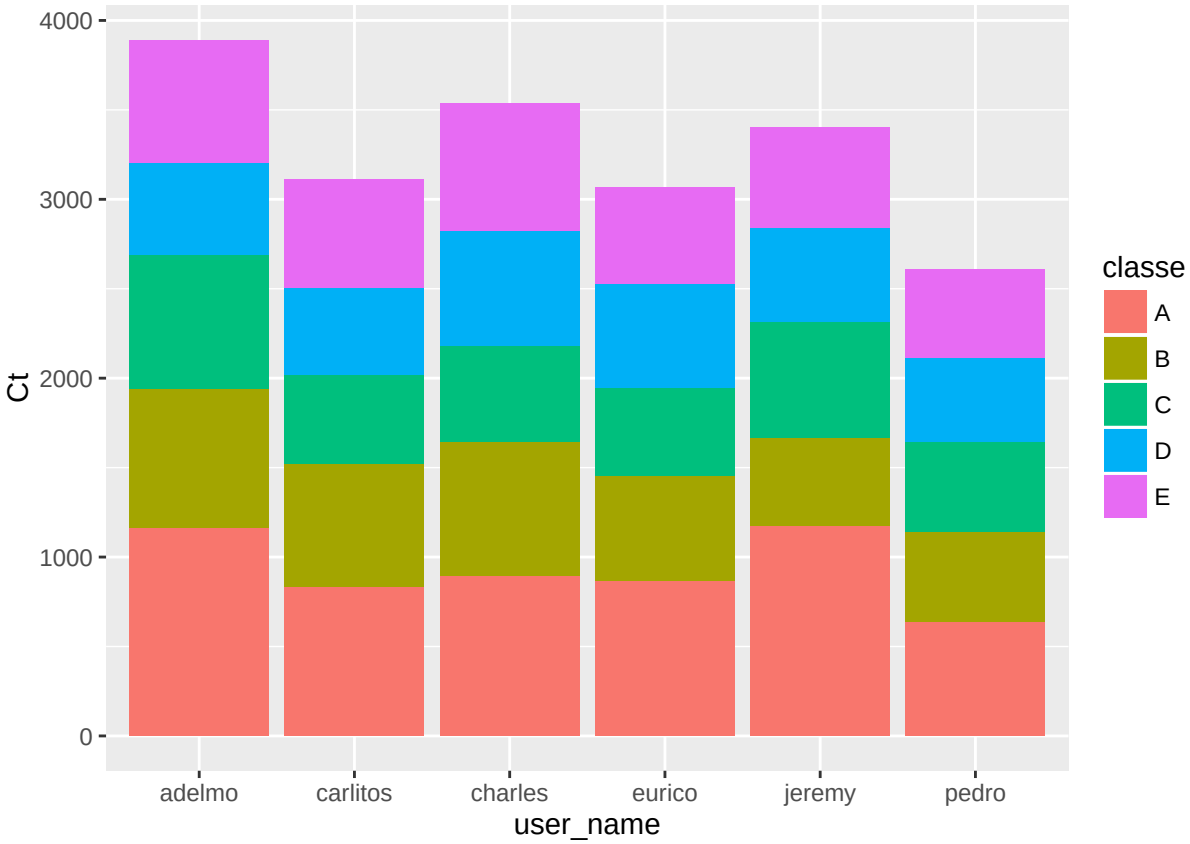
```
## [25] "gyros_arm_y"           "gyros_arm_z"           "accel_arm_x"
## [28] "accel_arm_y"           "accel_arm_z"           "magnet_arm_x"
## [31] "magnet_arm_y"          "magnet_arm_z"          "roll_dumbbell"
## [34] "pitch_dumbbell"        "yaw_dumbbell"          "total_accel_dumbbell"
## [37] "gyros_dumbbell_x"      "gyros_dumbbell_y"      "gyros_dumbbell_z"
## [40] "accel_dumbbell_x"      "accel_dumbbell_y"      "accel_dumbbell_z"
## [43] "magnet_dumbbell_x"     "magnet_dumbbell_y"     "magnet_dumbbell_z"
## [46] "roll_forearm"          "pitch_forearm"         "yaw_forearm"
## [49] "total_accel_forearm"   "gyros_forearm_x"       "gyros_forearm_y"
## [52] "gyros_forearm_z"       "accel_forearm_x"       "accel_forearm_y"
## [55] "accel_forearm_z"       "magnet_forearm_x"      "magnet_forearm_y"
## [58] "magnet_forearm_z"      "classe"
```

Data Exploration

These are two visuals of the training data. The first groups outcomes types by participant's name. The second cherry-picks a random variable and compares it against outcome.

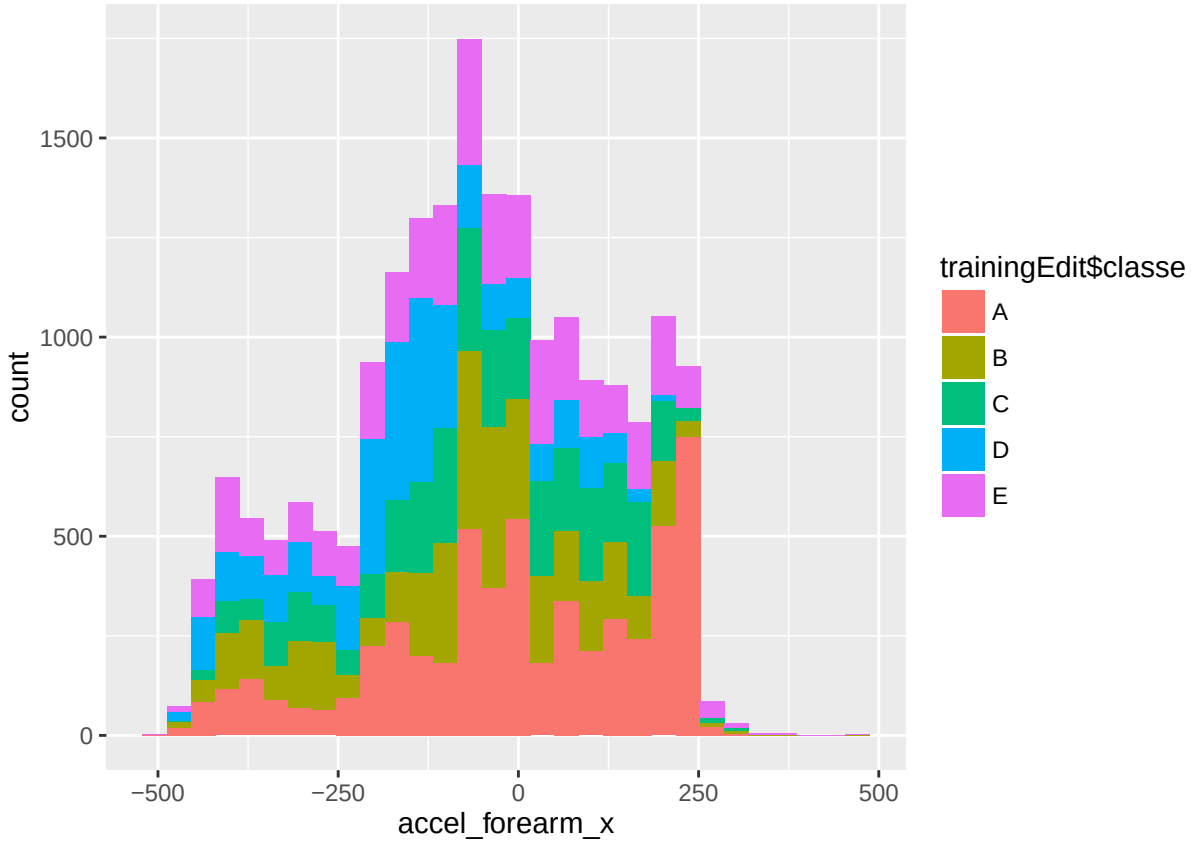
```
Count_names <- sqldf("select user_name, classe, count(*) as Ct
                      from trainingEdit
                      group by user_name, classe
                      order by user_name, classe")

# This counts the number outcomes by individual
ggplot(data = Count_names, mapping = aes(user_name, Ct)) +
        geom_bar(aes(fill=classe), stat="identity")
```



The outcome does not appear to be predictable solely based on the participant, as expected.

```
# Check relationship of one variable to outcome
ggplot(data = trainingEdit, mapping = aes(accel_forearm_x)) +
       geom_histogram(aes(fill=trainingEdit$classe))
```



Very high forearm acceleration seems to be related to a successful outcome. This will not be relevant to my model creation, but it is interesting to note.

It's also important to check for missing values:

```
nrow(trainingEdit[complete.cases(trainingEdit) == FALSE,])
```

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

Good – no observations contain NA values.

Model training

The machine learning models are trained below. To prevent overfitting, the models are trained against a subset of the training data ("training_training"") and then tested against another subset of the training data ("training_testing"). The model with the most accurate outcome when compared to the training_testing data will be selected.

```
training_rows <- createDataPartition(trainingEdit$classe, p=0.80, list=FALSE)
training_training <- trainingEdit[training_rows,]
training_testing <- trainingEdit[-training_rows,]
```

I also create a training control parameter for one model "rpart" that is used. The other two models are already slow to process, so this parameter is not passed to them.

```
train_control <- trainControl(method="repeatedcv", number=10, repeats=3) # Use object in train function
```

Training models using methods rpart, gbm, and rf. "rpart" is a simple classification tree model. "gbm" is a boosted tree model. "rf" stands for random forests and, like rpart, is a classification tree model, but it instead bootstraps at each node.

The rf and gbm models take significantly longer to process than the rpart model. They are also expected to be more accurate. I do not train the models in this RMD file because of processing time constraints.

```
# Using a boosted random forests model
trainGBM <- train(training_training$classe ~ .
                              , method = "gbm"
                              , data = training_training
                              , na.action = na.pass)

# Simpler tree but with additional training controls
trainRPART <- train(training_training$classe ~ .
                    , method = "rpart"
                    , data = training_training
                    , na.action = na.pass
                    , trControl = train_control)

# Using trees, different model, with training controls added
trainRF <- train(training_training$classe ~ .
                             , method = "rf"
                             , data = training_training
                             , na.action = na.pass)
```

Plot visuals are saved separately in the github repository.

Model selection

These models are compared against the training_testing data to determine which will be selected. They will be select based on accuracy.

```
# Predict "classe" on training_testing data
RPART_eval = predict(trainRPART, newdata=training_testing)
GBM_eval = predict(trainGBM, newdata=training_testing)
RF_eval = predict(traiRF, newdata=training_testing)

# Get accuracies
confusionMatrix(RPART_eval, training_testing$classe)
confusionMatrix(GBM_eval, training_testing$classe)
confusionMatrix(RF_eval, training_testing$classe)
```

The accuracies for the gbm and the rf methods were much higher than the accuracy for the rpart method. The rpart method only had an accuracy of about 50% whereas the gbm and rf methods had accuracies greater than 99%. I used the rf method to predict against the test set for the quiz. But the gbm model processing was slightly faster – which might make it more appealing in a real-world setting.

These accuracies also estimate the "out-of-sample error" because they should mirror the true error rate of the models. With out-of-sample accuracies greater than 99% for both the gbm and the rf models, the out-of-sample errors are less than 1%.

```
# Predict "classe" on the testing data
testingEdit <- testing[, !names(training) %in% statsVars]
predict(trainRF, newdata = testingEdit)
```

Results

The gbm and rf models both output the following: B A B A A E D B A A B C B A E E A B B B

Data citation

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.

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz4JtZ5RzZ6