Machine Learning Course Project

Amanda Salvesen

December 4, 2016

Introduction

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. 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, my goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Loading the Data

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

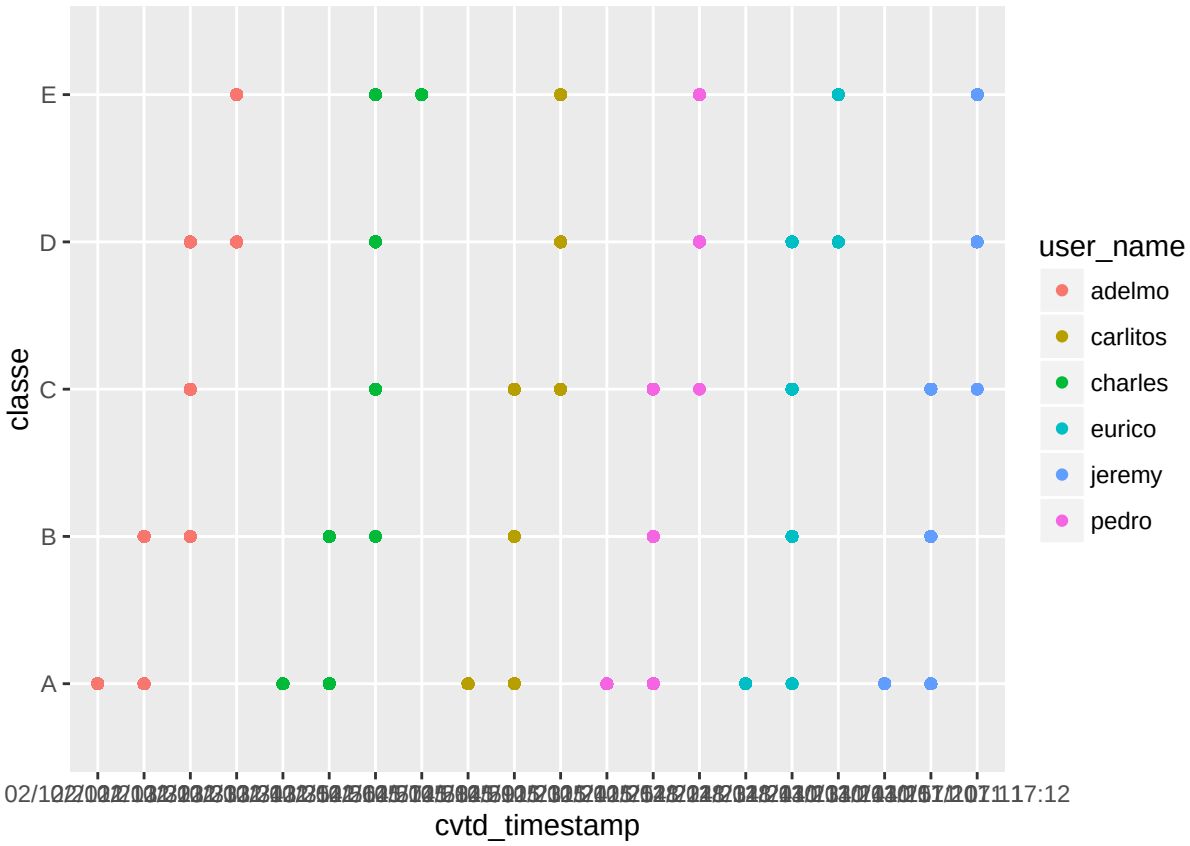
The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

```
library(caret); library(rattle); library(rpart.plot); library(randomForest)
train_raw = read.csv("pml-training.csv", na.strings = c("", "#DIV/0!", "NA"))
test_raw = read.csv("pml-testing.csv", na.strings = c("", "#DIV/0!", "NA"))
```

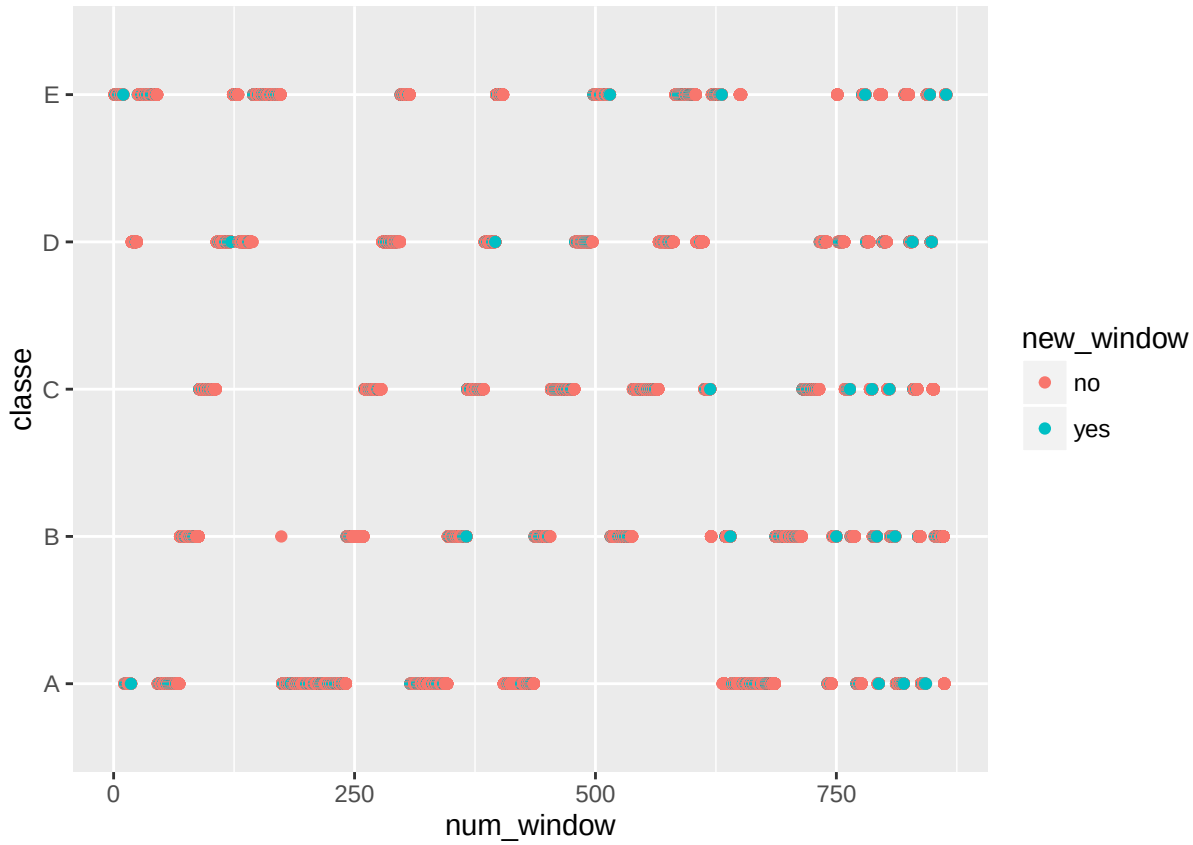
Data Cleaning

The first few columns do not contain diagnostic information. There are also some columns that are mostly NA's. Let's get rid of those for both the training and testing sets before we begin.

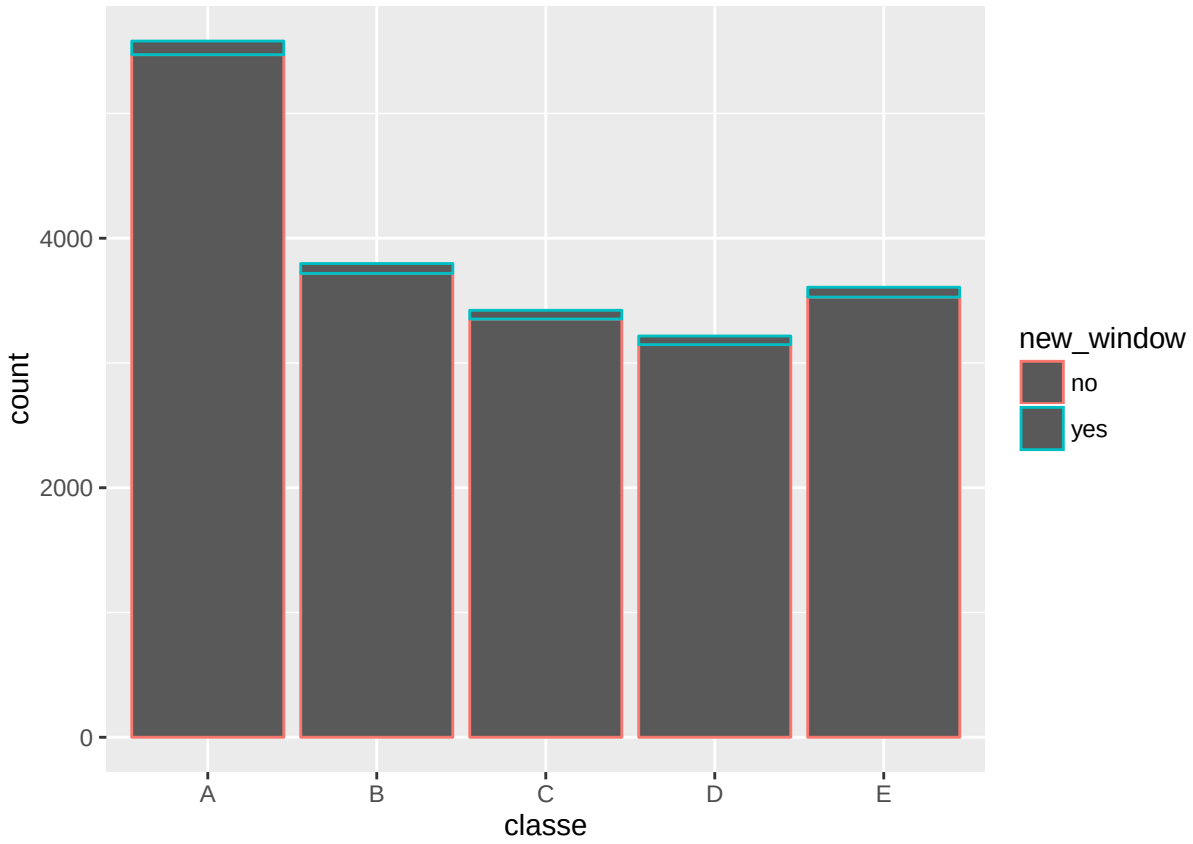
```
ggplot(data = train_raw) +
        geom_point(aes(x = cvtd_timestamp, y = classe,
                       color = user_name))
```




```
ggplot(data = train_raw) +
       geom_point(aes(x = num_window, y = classe,
                      color = new_window))
```




```
ggplot(data = train_raw) +
       geom_bar(aes(x = classe, color = new_window))
```




```
train_noNA <- train_raw[ , colSums(is.na(train_raw)) == 0]
test_noNA <- test_raw[ , colSums(is.na(test_raw)) == 0]

train_all <- train_noNA[, -c(1:7)]
test <- test_noNA[, -c(1:7)]
```

Cross Validation

```
set.seed(1234)
trainIndex = createDataPartition(train_all$classe, p = 0.60,list=FALSE)
train = train_all[trainIndex,]
cv = train_all[-trainIndex,]
```

Models

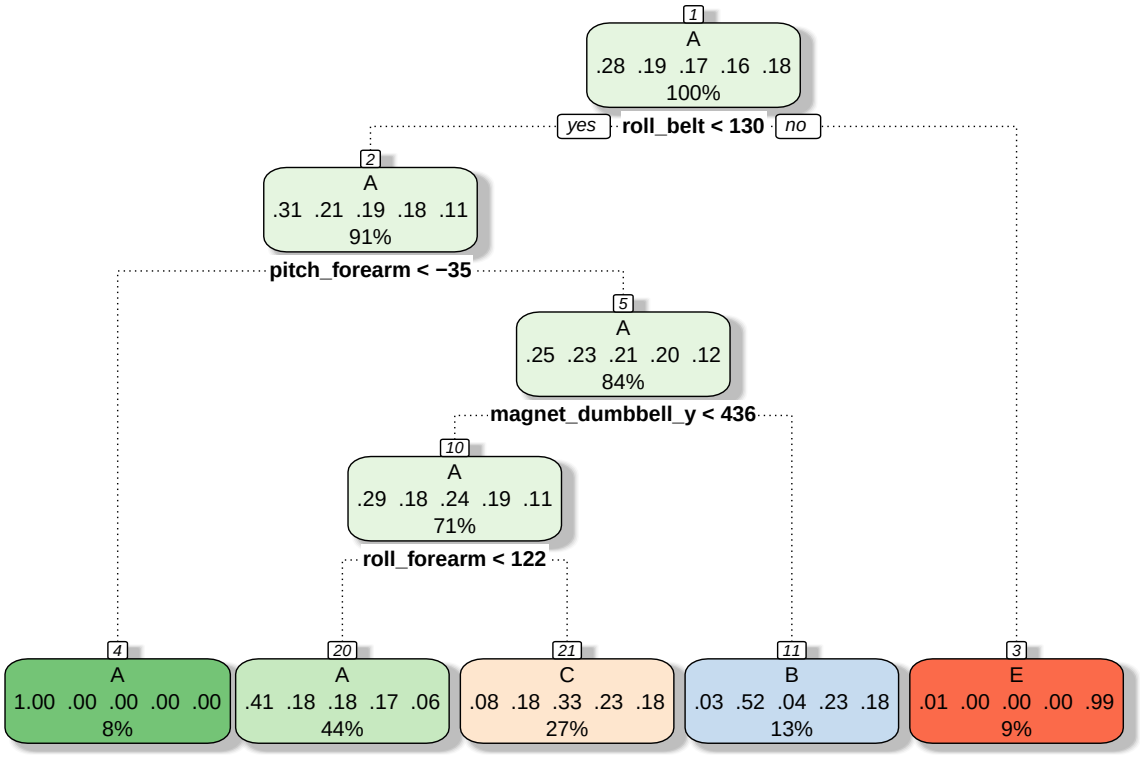
Random Forest

```
fitRF <- randomForest(classe ~ ., data = train, method = "class")
predRF <- predict(fitRF, cv)
confusionMatrix(cv$classe, predRF)$overall['Accuracy']
```

```
## Accuracy
## 0.993245
```

Rpart

```
fitRPART <- train(classe ~ ., method = "rpart", data = train)
fancyRpartPlot(fitRPART$finalModel)
```




```
predRPART <- predict(fitRPART, cv)
confusionMatrix(cv$classe, predRPART)$overall['Accuracy']
```

```
## Accuracy
## 0.490441
```

Random Forests with Preprocessing

```
preProc <- preProcess(train[, -53], method = "pca", thresh = 0.80)
trainPC <- predict(preProc, train[, -53])
cvPC <- predict(preProc, cv[, -53])
testPC <- predict(preProc, test[, -53])

fitRF_PC <- randomForest(train$classe ~ ., data = trainPC, method = "class")
confusionMatrix(cv$classe, predict(fitRF_PC, cvPC))$overall['Accuracy']
```

```
##  Accuracy
## 0.9551364
```

Model Selection

The random forest model without preprocessing is the most accurate, at .99 accuracy, so I will use it to predict the test data.

```
predict(fitRF, 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
```

20 out of 20!! Hooray!!