Machine Learning Assignment

Tom Chovanec

December 3, 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, your 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).

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

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.

Loading the Data

Load the training and testing data sets.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.5
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.2.5
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.2.5
```

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.2.5
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.2.5
```

```
if(!file.exists("pml-traning.csv")) {

  download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile = "pml
}

if(!file.exists("pml-testing.csv")) {

  download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile = "pml-
}

testing_data <- read.csv("pml-testing.csv",na.strings=c("", "NA", "NULL"))
training_data <- read.csv("pml-training.csv",na.strings=c("", "NA", "NULL"))
```

Data Cleanup

Due to columns having mostly NA, I decided to only include columns where the percent of NA's was less than 75% of the column's data. I also removed variables that are unrelated to the classe variable we are predicting. The columns that were removed are the first 7 columns of the data.

```
model_training <- training_data[ lapply( training_data, function(x) sum(is.na(x)) / length(x) ) < 0.01 ]

testing_data <- testing_data[ lapply( testing_data, function(x) sum(is.na(x)) / length(x) ) < 0.01 ][-(1
```

Partition the Dataset

We are splitting the training data into two sets for cross validation puposes. We randomly subsample 60% of the set for training purposes, while the remaining 40% will be used for testing.

```
set.seed(12345)
inTrain <- createDataPartition(y=model_training$classe, p=0.6, list=FALSE)
fit_training <- model_training[inTrain, ]
fit_testing <- model_training[-inTrain, ]

dim(fit_training)
```

```
## [1] 11776    53
```

```
dim(fit_testing)
```

```
## [1] 7846   53
```

Currently in the fit_training data set (it contains 11776 observations, or about 60% of the entire training data set), and fit_tetsing (it contains 7846 observations, or about 40% of the entire training data set). We now are going to identify and remove zero covariates from the fit_training and fit_testing data sets.

```
nzv_cols <- nearZeroVar(fit_training)
if(length(nzv_cols) > 0) {
  fit_training <- fit_training[, -nzv_cols]
  fit_testing <- fit_testing[, -nzv_cols]
}
dim(fit_training)
```

```
## [1] 11776    53
```

```
dim(fit_testing)
```

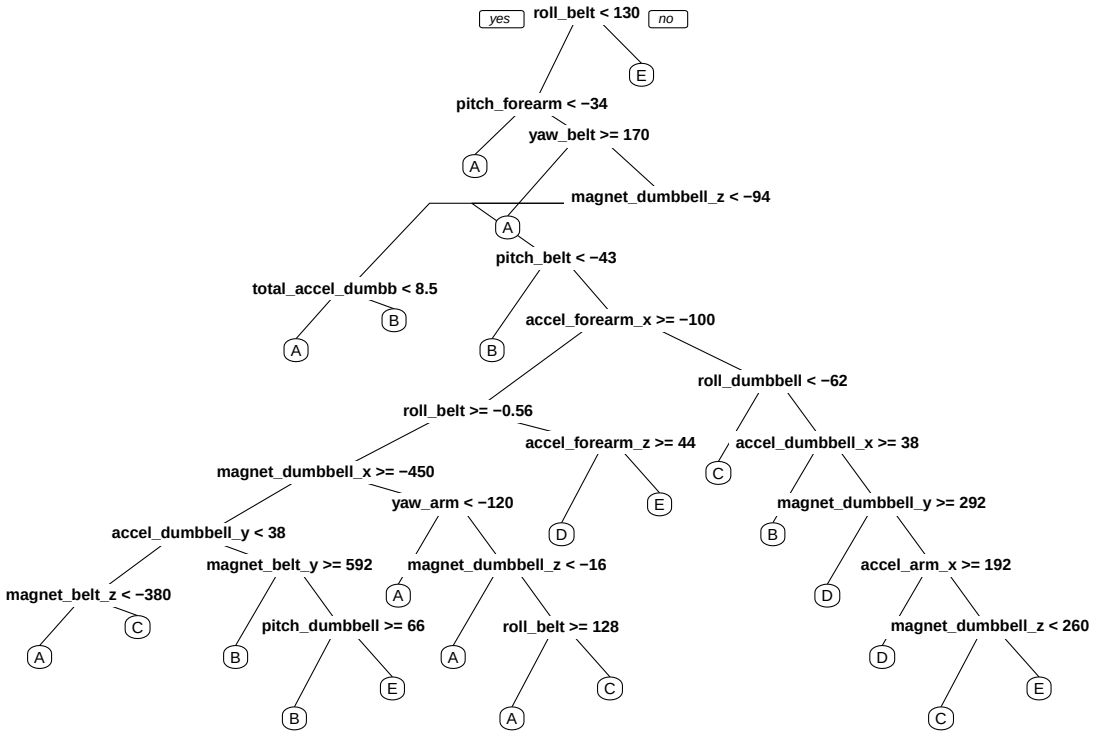
```
## [1] 7846   53
```

This step didn't do anything because the earlier removal of NA was enough to clean the data. We now have 53 clean covariates to build a model for classe.

Decision Tree

I created a quick tree classifier that selects roll_belt as the first discriminant among all 53 covariates.

```
modFitDT <- rpart(classe ~ ., data = fit_training, method="class")
prp(modFitDT)
```



This tree is not expected to have high accuracy.

```
set.seed(12345)
prediction <- predict(modFitDT, fit_testing, type = "class")
confusionMatrix(prediction, fit_testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1879  260   30   69   66
##          B   56  759   88   34   54
##          C  105  340 1226  354  234
##          D  155  132   23  807   57
##          E   37   27    1   22 1031
##
## Overall Statistics
##
##                Accuracy : 0.7267
##                  95% CI : (0.7167, 0.7366)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6546
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8418  0.50000   0.8962   0.6275   0.7150
## Specificity            0.9243  0.96334   0.8405   0.9441   0.9864
## Pos Pred Value         0.8155  0.76589   0.5427   0.6874   0.9222
## Neg Pred Value         0.9363  0.88928   0.9746   0.9282   0.9389
## Prevalence             0.2845  0.19347   0.1744   0.1639   0.1838
## Detection Rate         0.2395  0.09674   0.1563   0.1029   0.1314
## Detection Prevalence   0.2937  0.12631   0.2879   0.1496   0.1425
## Balanced Accuracy      0.8831  0.73167   0.8684   0.7858   0.8507
```

As you can see the decision tree has an accuracy around 73%. We will no longer investigate tree classifiers further because the Random Forest algorithm will prove much more accurate.

Random Forest

We fit a predictive model using Random Forest algorithm using a 2-fold cross-validation control. This is the simplest k-fold cross-validation possible and it will give a reduced computation time.

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.5
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
modelRf <- train(classe ~ ., data=fit_training, method="rf", trControl=trainControl(method="cv",number=
```

```
modelRf
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (2 fold)
## Summary of sample sizes: 5888, 5888
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
```

```
##    2    0.9773268  0.9713128
##   27    0.9791950  0.9736782
##   52    0.9738451  0.9669061
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

We now will validate the performance on the test data set.

```
predictRf <- predict(modelRf, fit_testing)
confusionMatrix(fit_testing$classe, predictRf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2228    4    0    0    0
##          B   10 1503    5    0    0
##          C    0    5 1360    3    0
##          D    0    0   20 1262    4
##          E    0    2    3    5 1432
##
## Overall Statistics
##
##                Accuracy : 0.9922
##                  95% CI : (0.99, 0.994)
##     No Information Rate : 0.2852
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9902
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9955   0.9927   0.9798   0.9937   0.9972
## Specificity            0.9993   0.9976   0.9988   0.9964   0.9984
## Pos Pred Value         0.9982   0.9901   0.9942   0.9813   0.9931
## Neg Pred Value         0.9982   0.9983   0.9957   0.9988   0.9994
## Prevalence             0.2852   0.1930   0.1769   0.1619   0.1830
## Detection Rate         0.2840   0.1916   0.1733   0.1608   0.1825
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9974   0.9952   0.9893   0.9950   0.9978
```

```
se <- 1 - as.numeric(confusionMatrix(fit_testing$classe, predictRf)$overall[1])
se
```

```
## [1] 0.007774662
```

So, the estimated accuracy of the model is 99.22% and the estimated out-of-sample error is 0.77.

Predicting for test data

We will now apply the random forest model to the test data that we downloaded.

```
answers <- predict(modelRf, testing_data)
answers
```

```
##  [1] 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
```