Coursera Practical Machine Learning Project

Zach Eisenstein

December 17, 2016

Background

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 study, participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The goal of this project will be to use data from accelerometers on the belt, forearm, arm, and dumbell of participants to correctly predict the "classe" or manner in which the exercise was performed.

More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Library Setup & Data Input

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

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

```
#load necessary packages & dataset
library(RCurl); library(caret); library(rpart); library(rpart.plot);
library(RColorBrewer); library(rattle); library(randomForest);

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

#download the data files to the working directory
download.file(train_url, destfile = "./train.csv", method="libcurl")
download.file(test_url, destfile = "./test.csv", method="libcurl")

#read in data files
org_train <- read.csv("train.csv", na.strings=c("NA","#DIV/0!",""))
org_test <- read.csv("test.csv", na.strings=c("NA","#DIV/0!",""))
```

Data Preprocessing

The goal of this section is to prepare the data for analysis. This involves reducing the number of variables and partitioning the training set to facilitate cross validation. We also set the random seed for reproducibility.

```
set.seed(456146)

#Keep only the numeric variables (except for classe)
training <- org_train[,which(lapply(org_train, class) %in% "numeric")]
training$classe <- org_train$classe

#Clean up number of variables by removing columns with a large percentage (>10%) of blanks or er
rors
training <- training[ lapply(training, function(x) sum(is.na(x)) / length(x) ) < 0.1 ]

#Splitting the original training set into a training and a cross validation set
inTrain <- createDataPartition(y=training$classe, p=0.75, list=FALSE)
training <- training[inTrain, ]; testing <- training[-inTrain, ]
dim(training); dim(testing)
```

```
## [1] 14718    28
```

```
## [1] 3682   28
```

The data has now been reduced from 160 to just 28 variables. We first start by fitting a decision tree model utilizing the rpart package. The model will fit the classe variable from all 27 other variables.

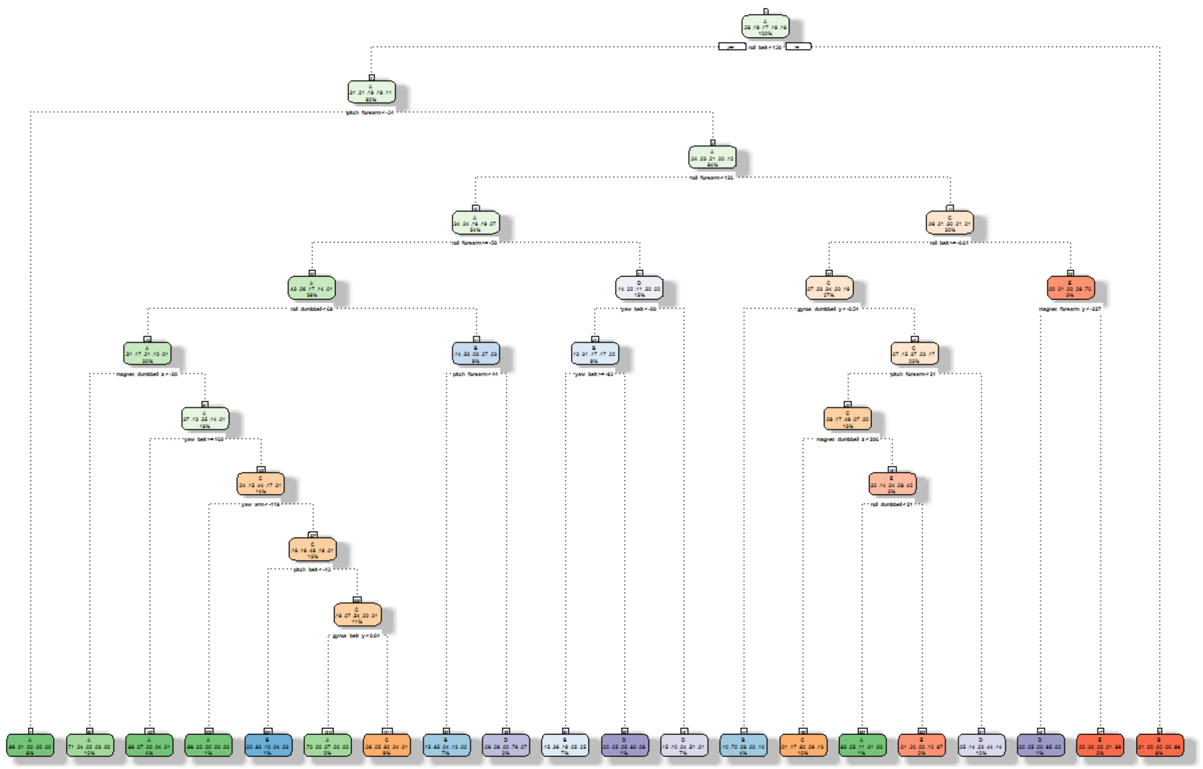
Train Model 1 - Decision Tree

```
model_tree <- rpart(classe ~., data=training, method="class")

# Top five most important variables
head(model_tree$variable.importance)
```

```
##         roll_belt     pitch_forearm          yaw_belt      roll_forearm
##         1775.7138         1218.4206         1164.8296          941.5239
## magnet_dumbbell_z        pitch_belt
##          916.4698          597.8146
```

```
# Visualizing the tree
fancyRpartPlot(model_tree)
```



Rattle 2016-Dec-17 14:23:13 ZachE

Accuracy on training set and cross validation set

```
# Training set accuracy
tree_train_pred <- predict(model_tree, training, type = "class")
confusionMatrix(tree_train_pred, training$classe)$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull 
##   6.888164e-01   6.072580e-01   6.812660e-01   6.962922e-01   2.843457e-01 
## AccuracyPValue  McnemarPValue 
##   0.000000e+00  4.749160e-202
```

The prediction accuracy in the testing set was 69%, probably a bit lower than we'd like. Now we will use the decision tree to predict the classe variable in the hold out set.

```
# Holdout set accuracy
tree_test_pred <- predict(model_tree, testing, type = "class")
confusionMatrix(tree_test_pred, testing$classe)$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull 
##   6.890277e-01   6.068041e-01   6.737947e-01   7.039601e-01   2.903313e-01 
## AccuracyPValue  McnemarPValue 
##   0.000000e+00   1.107965e-47
```

We see again in the holdout set a prediction accuracy of around 69%. While the decision tree did not exhibit overfitting, we should explore if another method would prove to have greater prediction accuracy.

Train Model 2 - Random Forest

Next we fit the model using the random forest method, this algorithm works as a collection of uncorrelated decision trees.

```
model_rf <- train(classe ~., method="rf", data=training, trControl=trainControl(method='cv'), nu
mber=5, allowParallel=TRUE )
model_rf
```

```
## Random Forest
##
## 14718 samples
##    27 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 13246, 13245, 13246, 13246, 13247, 13249, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9927973  0.9908885
##   14    0.9923899  0.9903738
##   27    0.9891957  0.9863340
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
# Top five most important variables
rownames(varImp(model_rf)$importance)[1:5]
```

```
## [1] "roll_belt"    "pitch_belt"   "yaw_belt"     "gyros_belt_x"
## [5] "gyros_belt_y"
```

Accuracy on training set and cross validation set

The random forest model has an an expected accuracy of over 99%. This is a good result, however, it could also mean the model was overfit. As such, we also cross validate with the holdout data in the original training set.

```
trainingPred <- predict(model_rf, testing)
confusionMatrix(trainingPred, testing$classe)$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      1.0000000      1.0000000      0.9989986      1.0000000      0.2903313
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

The 100% accuracy in the holdout set leaves us with much confidence of the predictive power of the random forest model. We will now employ the model to predict on the original test set.

Predictions on the original test set

```
testingPred <- predict(model_rf, org_test)
testingPred
```

```
##  [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
```