Practical Machine Learning Project

Scott Milner

Sunday, November 15, 2015

Introduction

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. 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).

Intended Goals

For this project we use data from accelerometers on the belt, forearm, arm, and dumbell of 6 study participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The goal of this project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set.

Initial Data Processing

```
library( caret )
```

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

```
library( randomForest )
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

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

```
trainingDataURL <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testDataURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
trainingDataFile <- "./data/pml-training.csv"
testDataFile  <- "./data/pml-testing.csv"
if( !file.exists("./data") )
{
  dir.create( "./data" )
}
```

```
if( !file.exists( trainingDataFile ) )
{
  download.file( trainingDataURL, destfile=trainingDataFile, method="curl" )
}
if( !file.exists( testDataFile ) )
{
  download.file( testDataURL, destfile=testDataFile, method="curl")
}
```

Download the data

Reading the data Read the two downloaded csv files into training and test set data frames.

```
trainingSetData <- read.csv( "./data/pml-training.csv" )
testSetData <- read.csv( "./data/pml-testing.csv" )
dim( trainingSetData )
```

```
## [1] 19622   160
```

```
dim( testSetData )
```

```
## [1]  20 160
```

There are 19622 observations and 160 variables in the training data set. The test data set contains only 20 observations and 160 variables. The "classe" variable in the training set represents how each excersise was performed and is the outcome we will attempt to predict.

Cleaning the data We clean the raw data to filter out some unnecessary variables and remove observations with missing values that may impact our model.

```
sum( complete.cases( trainingSetData ) )
```

```
## [1] 406
```

Remove columns that have NA missing values.

```
trainingSetData <- trainingSetData[, colSums( is.na( trainingSetData ) ) == 0 ]
testSetData <- testSetData[, colSums( is.na( testSetData ) ) == 0 ]
```

Filter out data types that do not significantly contribute to the accelerometer measurements.

```
classe <- trainingSetData$classe
trainingDataToRemove <- grepl( "^X|timestamp|window", names( trainingSetData ) )
trainingSetData <- trainingSetData[, !trainingDataToRemove]
trainingDataCleaned <- trainingSetData[, sapply( trainingSetData, is.numeric )]
trainingDataCleaned$classe <- classe
testDataToRemove <- grepl( "^X|timestamp|window", names( testSetData ) )
testSetData <- testSetData[, !testDataToRemove]
testDataCleaned <- testSetData[, sapply( testSetData, is.numeric )]
```

Our cleaned training data set now contains 19622 observations and 53 variables. Our testing data set contains 20 observations and 53 variables.

Splitting the data We now split the cleaned training set into a pure training data set and a training validation data set. The intention here is to allow for cross validation within the training set before moving to the test set data. The training validation data set is split at 30% of the full training data set.

```
set.seed( 2972 )
inTrain <- createDataPartition( trainingDataCleaned$classe, p=0.70, list=F )
trainingData <- trainingDataCleaned[inTrain, ]
testTrainingData <- trainingDataCleaned[-inTrain, ]
```

Data Modeling

Our predictive model is derived using the Random Forest algorithm. The benefits of using a Random Forest include its robustness against over-fitting and its handling of outliers. Our validation method when applying the algorithm is 5-fold cross validation.

```
randForestCtl <- trainControl( method="cv", 5 )
randForestModel <- train( classe ~ ., data=trainingData, method="rf", trControl=randForestCtl, ntree=25
randForestModel
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10990, 10989, 10988, 10991
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9897353  0.9870148  0.001812718  0.002294043
##   27    0.9900999  0.9874757  0.002080659  0.002632925
##   52    0.9842039  0.9800208  0.005709352  0.007214733
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Use the training test validation data set to estimate our random forest model's performance.

```
randForestPredict <- predict( randForestModel, testTrainingData )
confusionMatrix( testTrainingData$classe, randForestPredict )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    7 1131    1    0    0
##          C    0    7 1018    1    0
##          D    0    0   10  951    3
##          E    1    1    2    5 1073
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9934
##                  95% CI : (0.991, 0.9953)
##     No Information Rate : 0.2856
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9916
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9952   0.9921   0.9874   0.9937   0.9972
## Specificity            0.9998   0.9983   0.9984   0.9974   0.9981
## Pos Pred Value         0.9994   0.9930   0.9922   0.9865   0.9917
## Neg Pred Value         0.9981   0.9981   0.9973   0.9988   0.9994
## Prevalence             0.2856   0.1937   0.1752   0.1626   0.1828
## Detection Rate         0.2843   0.1922   0.1730   0.1616   0.1823
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9975   0.9952   0.9929   0.9955   0.9977
```

```
accuracy <- postResample( randForestPredict, testTrainingData$classe )
accuracy
```

```
##  Accuracy     Kappa
## 0.9933730 0.9916163
```

```
accPct <- 1 - as.numeric( confusionMatrix( testTrainingData$classe, randForestPredict )$overall[1] )
accPct
```

```
## [1] 0.006627018
```

Our model's estimated accuracy is 99.42% with an estimated out-of-sample error of 0.58%.

Prediction of the Test Data Set

Apply our model to the original raw testing data set (without the `problem_id` column data) to derive a predition result.

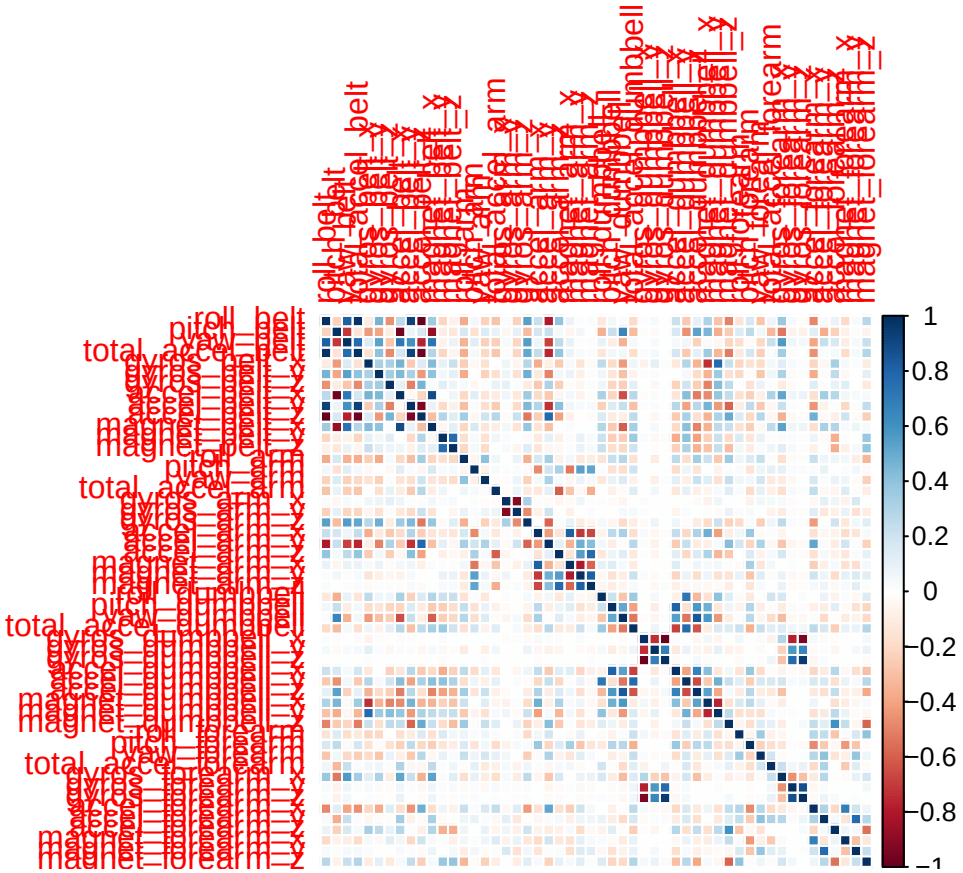
```
predictionResult <- predict( randForestModel, testDataCleaned[, -length( names( testDataCleaned ) )] )
predictionResult
```

```
##  [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
```

Appendix: Visualizations

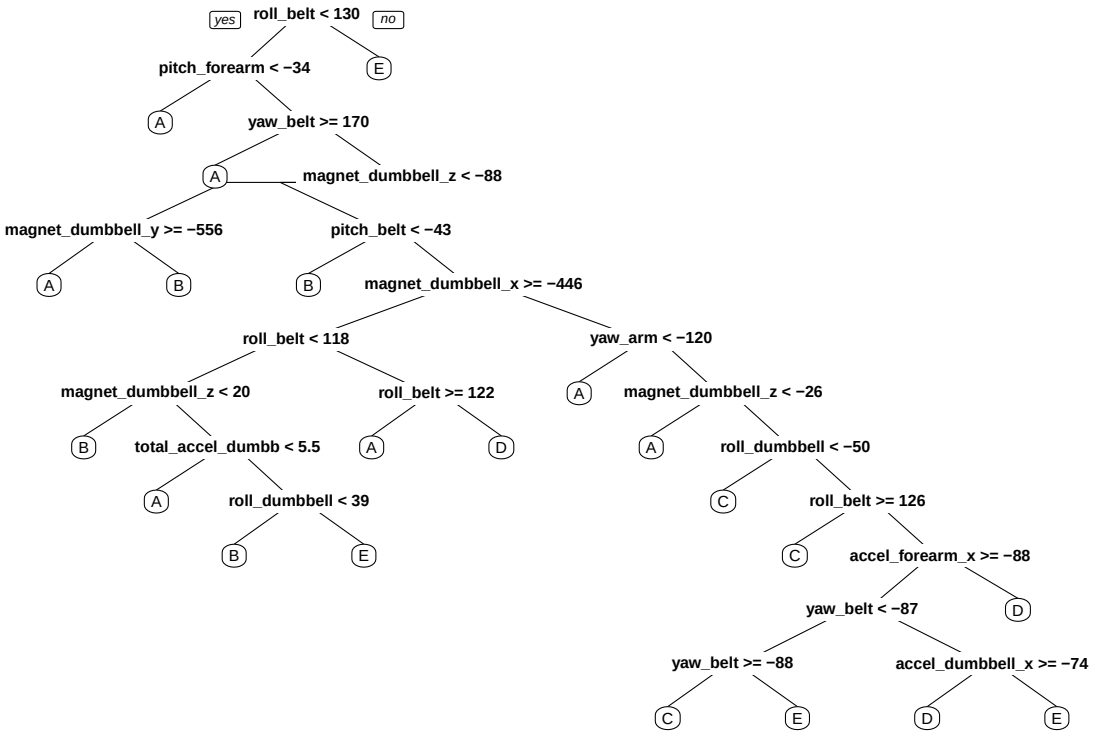
1. Correlation Matrix

```
corrPlot <- cor( trainingData[, -length( names( trainingData ) )] )
corrplot( corrPlot, method="color" )
```



2. Decision Tree

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


yes
roll_belt < 130
no
pitch_forearm < −34
E
A
yaw_belt >= 170
A
magnet_dumbbell_z < −88
magnet_dumbbell_y >= −556
pitch_belt < −43
A
B
B
magnet_dumbbell_x >= −446
roll_belt < 118
yaw_arm < −120
magnet_dumbbell_z < 20
roll_belt >= 122
A
magnet_dumbbell_z < −26
B
total_accel_dumbb < 5.5
A
D
A
roll_dumbbell < −50
A
roll_dumbbell < 39
C
roll_belt >= 126
B
E
C
accel_forearm_x >= −88
yaw_belt < −87
D
yaw_belt >= −88
accel_dumbbell_x >= −74
C
E
D
E