Weighting Lifting Exercise Analysis

Nazirul Hazim

Saturday, October 24, 2015

Goal

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.

Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). The aim of this report was to use data from accelerometers placed on the belt, forearm, arm, and dumbell of these six participants to predict how well they were doing the exercise in terms of the classification in the data.

Approach

The data for the project was provided as training and testing data. First the variable having high missing values were removed and then the remaining variables, which don't have missing values, were worked upon to find the amount of correlation between them, because these highly correlated variables can lead to multicollinearity which can increase miss classification error rate.

The next big quest is choosing the right algorithm which learns from the data to its best. A random forest model was selected to predict the classification because it has methods for balancing error in class population unbalanced data sets. randomForest will be the best choice for the data, which uses the bragging method that is a high variance, low bias technique.

The learned algorithm from this data is then used to predict the test data.

The libraries used in this report are :

```
library(caret)
library(corrplot)
library(knitr)
library(randomForest)
```

Data Preprocessing

The training data for this project was downloaded from https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data was downloaded from:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The training dataset was loaded into R workspace

```
# read the csv file for training
data_training = read.csv("pml-training.csv", na.strings= c("NA","","  "))
```

Initial viewing of the training data showed a lot of missing or NA values.

```
dim(data_training)
```

```
[1] 19622   160
```

The training data consists of 19622 observations and 160 variables.

There was a lot of NA values in the data which would create a lot of noise for the model. As a result, these columns were removed from the data set. The first eight columns that acted as identifiers for the experiment were also removed.

```
# clean the data by removing columns with NAs etc
data_training_NAs <- apply(data_training, 2, function(x) {sum(is.na(x))})
data_training_clean <- data_training[,which(data_training_NAs == 0)]

# remove identifier columns such as name, timestamps etc
data_training_clean <- data_training_clean[8:length(data_training_clean)]
dim(data_training_clean)
```

```
[1] 19622    53
```

The clean training data now contains 19622 observations and 53 variables.

Multicollinearity

Before building the model, correlations within numeric variables is looked at, to remove multicollinearity.

```
correlations = cor(data_training_clean[, -length(data_training_clean)])
corrplot(correlations, order='hclust', tl.cex=.5)
```

In this type of plot the dark red and blue colours indicate a highly negative and positive relationship respectively between the variables. A cutoff of 0.75 is considered. Removing variables with correlation above the cutoff of 0.75.

```
corrCutoff = findCorrelation(correlations, cutoff=.75)
predictors = data_training_clean[, -corrCutoff]
classe = data_training$classe
train_data_final = cbind(classe, predictors)
```

The train_data_final is the final clean datasets which we will use for model building using Random Forest Algorithm

Model Building

The train data set was split up into training and cross validation sets in a 70:30 ratio in order to train the model and then test it against data it was not specifically fitted to.

```
inTrain = createDataPartition(y=train_data_final$classe, p=0.7, list=FALSE)
training = train_data_final[inTrain,]
crossVal = train_data_final[-inTrain,]
```

A random forest model was selected to predict the classification because it has methods for balancing error in class population unbalanced data sets.

```
model = randomForest(classe~., data=training, importance = TRUE,ntrees = 10)
print(model)
```

```
Call:
 randomForest(formula = classe ~ ., data = training, importance = TRUE,      ntrees = 10)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 5

        OOB estimate of  error rate: 0.72%
Confusion matrix:
     A    B    C    D    E class.error
A 3902    2    1    0    1 0.001024066
B    9 2643    5    0    1 0.005643341
C    0   24 2357   14    1 0.016277129
D    1    0   28 2219    4 0.014653641
E    0    0    2    6 2517 0.003168317
```

```
par(mar=c(3,4,4,4))
plot(model)
```

The model produced a very small OOB error rate of .79%. This was deemed satisfactory enough to progress the testing.

The out of Bag error rate plot . Black line in the middle is the mse which is the average overall trees

Cross-validation

The model was then used to classify the remaining 30% of data. The results were placed in a confusion matrix along with the actual classifications in order to determine the accuracy of the model.

```
predictCrossVal = predict(model, crossVal)
confusionMatrix(crossVal$classe, predictCrossVal)
```

This model yielded a 99.05% prediction accuracy. Again, this model proved very robust and adequete to predict new data.

Predictions

The test data set was then loaded into R and cleaned in the same manner as before.

```
data_testing <- read.csv("pml-testing.csv", na.strings= c("NA","","  "))
data_testing_NAs <- apply(data_testing, 2, function(x) {sum(is.na(x))})
data_testing_clean <- data_testing[,which(data_testing_NAs == 0)]

# remove identifier columns such as name, timestamps etc
data_testing_clean <- data_testing_clean[8:length(data_testing_clean)]
```

The model was then used to predict the classifications of the 20 results of this new data.

```
predictions = predict(model, data_testing_clean)
predictions
```

```
 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
```

Conclusions

With the amount of information provided, which was collected using many measuring gadgets, it can accurately predicted how well a person is performing an exercise, using a simple model such as random forest.