Machine Learning Project

Thursday, June 18, 2015

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. The 5 possible methods include - * A: exactly according to the specification * B: throwing the elbows to the front * C: lifting the dumbbell only halfway * D: lowering the dumbbell only halfway * E: throwing the hips to the front

The goal of project is to predict the manner in which device users did the exercise. This is represented by "classe" variable in the training set.

Reading, Cleaning and Preprocessing Data

```
# Load required libraries
library(caret)
```

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

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.1
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.2.1
```

```
set.seed(9999)

# Training Data - Identifying "NA", "" and "#DIV/0!" as NA strings while read
ing the data
training <- read.csv("pml-training.csv",na.strings=c("NA","","#DIV/0!"))

# Testing Data - Identifying "NA", "" and "#DIV/0!" as NA strings while readi
ng the data
testing <- read.csv("pml-testing.csv",na.strings=c("NA","","#DIV/0!"))
```

```
# Delete columns with all missing values
training <-training[,colSums(is.na(training))== 0]
testing  <-testing[,colSums(is.na(testing)) == 0]

# Some variables are not required for the current project - user_name, raw_ti
mestamp_part_1, raw_timestamp_part_,2 cvtd_timestamp, new_window, and  num_wi
ndow (columns 1 to 7) so removing these variables.

training <- training[,-c(1:7)]
testing <- testing[,-c(1:7)]

# Partitioning the original training set into a training set and a validation
set to validate the model. Splitting on the 'classe' variable (variable of in
terest) with a 70-30 split

trainSet <- createDataPartition(y = training$classe, p = 0.7, list = FALSE)
trainData <- training[trainSet,]
validData <- training[-trainSet,]
```

Training and Building the Model

We will apply 3 different learning methods Classification tree, Gradient boosting (gbm) and Random forest models.

```
# Classification Tree Model

CTmodel <- train(classe ~ ., method="rpart", data = trainData)

## Loading required package: rpart

CTpredict <- predict(CTmodel, validData)

confusionMatrix(CTpredict, validData$classe)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1008  176   27   55   15
##          B    4  198   20    3    7
##          C  509  498  609  301  404
##          D  148  267  370  605  150
##          E    5    0    0    0  506
##
## Overall Statistics
##
##                Accuracy : 0.4972
##                  95% CI : (0.4843, 0.5101)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.3737
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6022  0.17384   0.5936   0.6276  0.46765
## Specificity            0.9352  0.99284   0.6477   0.8100  0.99896
## Pos Pred Value         0.7869  0.85345   0.2624   0.3929  0.99022
## Neg Pred Value         0.8553  0.83354   0.8830   0.9174  0.89282
## Prevalence             0.2845  0.19354   0.1743   0.1638  0.18386
## Detection Rate         0.1713  0.03364   0.1035   0.1028  0.08598
## Detection Prevalence   0.2177  0.03942   0.3944   0.2617  0.08683
## Balanced Accuracy      0.7687  0.58334   0.6206   0.7188  0.73331
```

```
# Gradient Boosting (GBM)

GBmodel <- train(classe ~ ., method="gbm", data = trainData, verbose=F)
```

```
## Loading required package: gbm
```

```
## Warning: package 'gbm' was built under R version 3.2.1
```

```
## Loading required package: survival
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.1
## Loading required package: plyr
```

```
GBpredict <- predict(GBmodel, validData)

confusionMatrix(GBpredict, validData$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1647   45    0    1    3
##          B   18 1063   31    1   13
##          C    3   27  981   37   10
##          D    4    3   11  918   18
##          E    2    1    3    7 1038
##
```

```
## Overall Statistics
##
##                Accuracy : 0.9596
##                  95% CI : (0.9542, 0.9644)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9488
##  Mcnemar's Test P-Value : 1.881e-07
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9839   0.9333   0.9561   0.9523   0.9593
## Specificity            0.9884   0.9867   0.9842   0.9927   0.9973
## Pos Pred Value         0.9711   0.9440   0.9272   0.9623   0.9876
## Neg Pred Value         0.9936   0.9840   0.9907   0.9907   0.9909
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2799   0.1806   0.1667   0.1560   0.1764
## Detection Prevalence   0.2882   0.1913   0.1798   0.1621   0.1786
## Balanced Accuracy      0.9861   0.9600   0.9701   0.9725   0.9783
```

```
# Random Forest Model
RFmodel <- randomForest(classe ~. , method="class", data=trainData, importanc
e=TRUE)

RFpredict <- predict(RFmodel, validData, type = "class")

confusionMatrix(RFpredict, validData$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    6    0    0    0
##          B    0 1131   10    0    0
##          C    0    2 1014   11    3
##          D    0    0    2  951    0
##          E    0    0    0    2 1079
##
## Overall Statistics
##
##                Accuracy : 0.9939
##                  95% CI : (0.9915, 0.9957)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9923
##  Mcnemar's Test P-Value : NA
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9930   0.9883   0.9865   0.9972
## Specificity            0.9986   0.9979   0.9967   0.9996   0.9996
## Pos Pred Value         0.9964   0.9912   0.9845   0.9979   0.9981
## Neg Pred Value         1.0000   0.9983   0.9975   0.9974   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1922   0.1723   0.1616   0.1833
## Detection Prevalence   0.2855   0.1939   0.1750   0.1619   0.1837
## Balanced Accuracy      0.9993   0.9954   0.9925   0.9931   0.9984
```

Interpreting Model Results

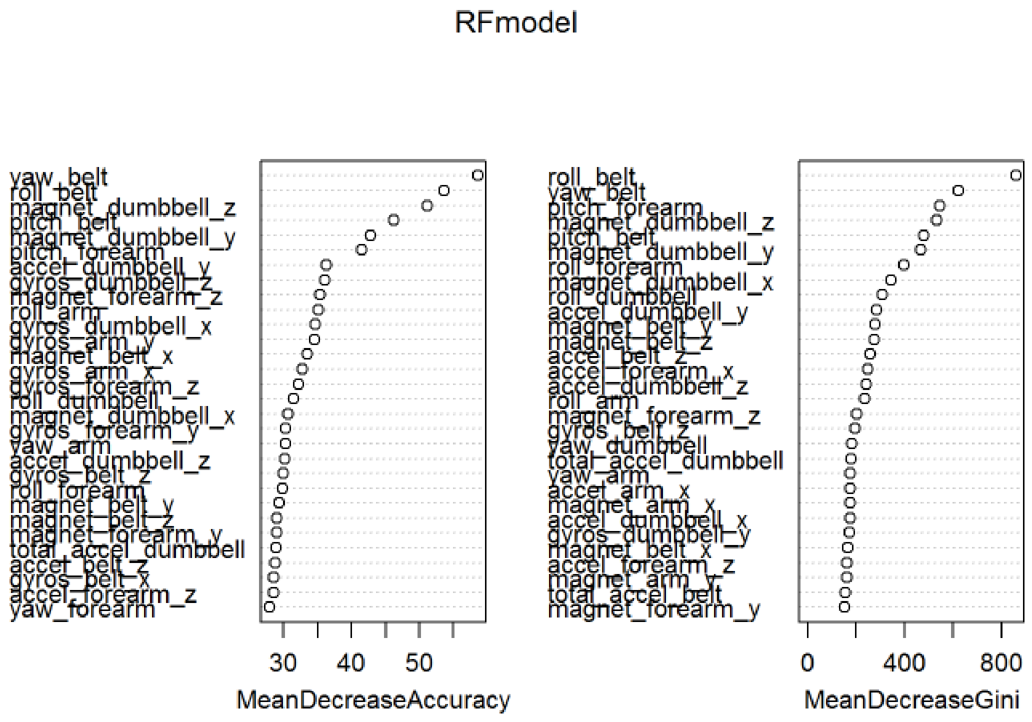
1. Random Forest model performed better than classification Trees and Gradient Boosting (GBM).
2. Random Forest model has accuracy of 99.3% compared to 95.9% for GBM and classification Tree for 49.7%.
3. Expected out-of-sample error (1 - accuracy) is estimated at 0.6% for Random Forest.

Finding the importance of variables in Random Forest Model

```
print("Importance of variable in the Random Forest model")

## [1] "Importance of variable in the Random Forest model"

varImpPlot(RFmodel)
```



As per Variable Importance Charts, yaw_belt and roll_belt are 2 important predictors for the model obtained with random forest and tuned by cross validation.

Predicting 20 test cases

Finally, the Random forest model (RFmodel) tuned with cross-validation set will be used to predict 20 test cases available in the test data loaded earlier.

```
# Predicting on the testing data set using Random Forest Model

prediction <- predict(RFmodel, testing)
print(prediction)
```

```
##  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
```

```
answers <- as.vector(prediction)
pml_write_files(answers)
```