Machine Learning

karla

12/9/2020

Background

This is the course project for the Coursera Practical Machine Learning class. Methods detailed in this report will be similar to those discussed during the course and using the training data provided will include decision trees, random forests, gradient boosted trees, and support vector machines. Predictions will be developed using the provided test set to obtain the accuracy and the out of sample error rates for each model. Using this information, the best model will be determined.

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, the goal is to use data from accelerometers on the belt, forearm, arm, and dumbell. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions:

1. exactly according to the specification (Class A),
2. throwing the elbows to the front (Class B),
3. lifting the dumbbell only halfway (Class C),
4. lowering the dumbbell only halfway (Class D) and
5. throwing the hips to the front (Class E).

Libraries and Data

First, all required libraries for the analysis will be loaded, followed by the data sets. More information on the data set provided is available at http://groupware.les.inf.puc-rio.br/har. 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

```
library(caret)
library(kernlab)
library(ggplot2)
library(rattle)
library(randomForest)

#First, determine if the files exist in the working directory. If not, download the files.

#if(!file.exists("./data/pml-training.csv")){
#    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv","pml-training
#if(!file.exists("./data/pml-testing.csv")){
#    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv","pml-testing.c
```

```
training<-read.csv("pml-training.csv")
testing<-read.csv("pml-testing.csv")
dim(training); dim(testing)
```

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

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

```
table(training$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

From the above information, we can see that the training set consists of 19622 records, the testing set consists of 20 records, and both contain 160 variables. We can also see that the distribution for variable of interest, the "classe" variable.

Data review and preprocessing

First, with 160 variables we should clean up the data and remove columns with mostly NAs, remove the metadata information and remove near zero variance variables.

```
# Remove columns with mostly NAs
training<-training[,colMeans(is.na(training)) < .9]

# Remove metadata columns
training<-training[,-c(1:7)]

# Removing near zero variance variables
nzv<-nearZeroVar(training)
training<-training[,-nzv]
dim(training)
```

```
## [1] 19622    53
```

```
# Create training and validation subsets
inTrain<-createDataPartition(y=training$classe, p=0.7, list=FALSE)
training.subset<-training[inTrain,]
valid.subset<-training[-inTrain,]
```

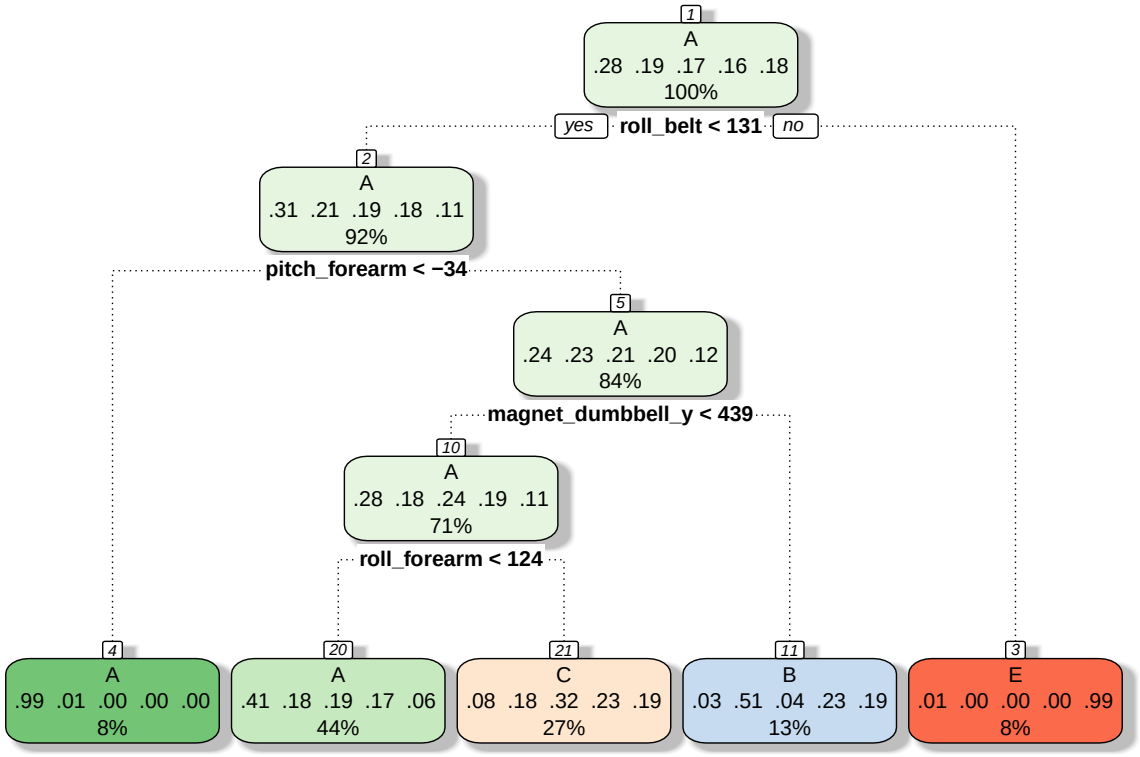
Now, with our training data tidied up and subsets created for model training and validation, we can move on to model formulation and testing.

Model development and review

Decision Tree

The first model we evaluate is the decision tree, with the model developed, predictions forecasted and the accuracy detailed below.

```
# Create model
mod.trees<-train(classe~., data=training.subset, method="rpart")
fancyRpartPlot(mod.trees$finalModel)
```




```
# Develop predictions and calculate confusion matrix and accuracy
pred.trees<-predict(mod.trees, valid.subset)
cm.trees<-confusionMatrix(pred.trees, factor(valid.subset$classe))
cm.trees
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1536  494  458  442  168
##          B   25  376   39  170  159
##          C  111  269  529  352  260
##          D    0    0    0    0    0
##          E    2    0    0    0  495
##
## Overall Statistics
##
##                Accuracy : 0.4989
##                  95% CI : (0.486, 0.5118)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.3444
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9176  0.33011  0.51559   0.0000  0.45749
## Specificity            0.6291  0.91719  0.79584   1.0000  0.99958
## Pos Pred Value         0.4958  0.48895  0.34780      NaN  0.99598
## Neg Pred Value         0.9505  0.85086  0.88611   0.8362  0.89105
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2610  0.06389  0.08989   0.0000  0.08411
## Detection Prevalence   0.5264  0.13067  0.25845   0.0000  0.08445
## Balanced Accuracy      0.7733  0.62365  0.65572   0.5000  0.72853
```

Random Forest

Next, a random forest model will be evaluated, with the model developed, predictions forecasted and the accuracy detailed below.

```
# Create model
control <- trainControl(method="cv", number=3, verboseIter=F)
mod.rf<-randomForest(classe~., data=training.subset, method="rf",  trControl=control, tuneLength = 5, pr
# Develop predictions and calculate confusion matrix and accuracy
pred.rf<-predict(mod.rf, valid.subset)
cm.rf<-confusionMatrix(pred.rf, factor(valid.subset$classe))
cm.rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    4    0    0    0
##          B    0 1134    6    0    0
##          C    0    1 1016    8    0
##          D    0    0    4  954    2
##          E    0    0    0    2 1080
##
## Overall Statistics
##
##                Accuracy : 0.9954
##                  95% CI : (0.9933, 0.997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9942
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9956   0.9903   0.9896   0.9982
## Specificity            0.9991   0.9987   0.9981   0.9988   0.9996
## Pos Pred Value         0.9976   0.9947   0.9912   0.9937   0.9982
## Neg Pred Value         1.0000   0.9989   0.9979   0.9980   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1927   0.1726   0.1621   0.1835
## Detection Prevalence   0.2851   0.1937   0.1742   0.1631   0.1839
## Balanced Accuracy      0.9995   0.9972   0.9942   0.9942   0.9989
```

Gradient Boosted Trees

Next, a gradient boosted model will be evaluated, with the model developed, predictions forecasted and the accuracy detailed below.

```
# Create model
mod.gbm<-train(classe~., data=training.subset, method="gbm",  verbose=FALSE)
# Develop predictions and calculate confusion matrix and accuracy
pred.gbm<-predict(mod.gbm, valid.subset)
cm.gbm<-confusionMatrix(pred.gbm, factor(valid.subset$classe))
cm.gbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1657   38    0    0    2
##          B    8 1070   36    1   10
##          C    4   28  975   30    8
##          D    4    3   12  928   12
##          E    1    0    3    5 1050
##
## Overall Statistics
##
##                Accuracy : 0.9652
##                  95% CI : (0.9602, 0.9697)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9559
##
##  Mcnemar's Test P-Value : 8.222e-08
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9898   0.9394   0.9503   0.9627   0.9704
## Specificity            0.9905   0.9884   0.9856   0.9937   0.9981
## Pos Pred Value         0.9764   0.9511   0.9330   0.9677   0.9915
## Neg Pred Value         0.9959   0.9855   0.9895   0.9927   0.9934
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2816   0.1818   0.1657   0.1577   0.1784
## Detection Prevalence   0.2884   0.1912   0.1776   0.1630   0.1799
## Balanced Accuracy      0.9902   0.9639   0.9679   0.9782   0.9843
```

Support Vector Machines

Finally, a support vector machine model will be evaluated, with the model developed, predictions forecasted and the accuracy detailed below.

```
# Create model
mod.svm<-train(classe~., data=training.subset, method="svmLinear", verbose=FALSE)
# Develop predictions and calculate confusion matrix and accuracy
pred.svm<-predict(mod.svm, valid.subset)
cm.svm<-confusionMatrix(pred.svm, factor(valid.subset$classe))
cm.svm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1574  158   84   68   63
##          B   30  814   85   40  147
##          C   25   72  797   97   60
##          D   41   26   29  722   59
##          E    4   69   31   37  753
##
## Overall Statistics
##
##                Accuracy : 0.7918
##                  95% CI : (0.7812, 0.8022)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7351
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9403   0.7147   0.7768   0.7490   0.6959
## Specificity            0.9114   0.9364   0.9477   0.9685   0.9706
## Pos Pred Value         0.8084   0.7294   0.7583   0.8233   0.8423
## Neg Pred Value         0.9746   0.9319   0.9526   0.9517   0.9341
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2675   0.1383   0.1354   0.1227   0.1280
## Detection Prevalence   0.3308   0.1896   0.1786   0.1490   0.1519
## Balanced Accuracy      0.9258   0.8255   0.8623   0.8587   0.8333
```

Model Comparisons and conclusions

When looking at the accuracy across all 4 model types, we can see that the accuracy is highest for the Random Forest model.

```
cm.trees$overall[1]
```

```
##  Accuracy
```

```
## 0.4988955
```

```
cm.rf$overall[1]
```

```
##  Accuracy
## 0.9954121
```

```
cm.gbm$overall[1]
```

```
##  Accuracy
## 0.9651657
```

```
cm.svm$overall[1]
```

```
##  Accuracy
## 0.7918437
```

Applying the best model to the test data

The Random Forest model is applied to predict the 20 test data set results as shown below.

```
predict.testing<-predict(mod.rf, newdata=testing)
predict.testing
```

```
##  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
```