Machine learning course project

Project 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. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. 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).

Data

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

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Assignment

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Getting and Cleaning Data

Load library

```
library(caret)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
```

```r
library(randomForest)
library(knitr)
```

Getting Data

```r
training_data <- read.csv("pml-training.csv")
testing_data <- read.csv("pml-testing.csv")
inTrain <- createDataPartition(training_data$classe, p=0.6, list=FALSE)
myTraining <- training_data[inTrain, ]
myTesting <- training_data[-inTrain, ]
```

Cleaning Data

```r
# remove variables with nearly zero variance
nzv <- nearZeroVar(myTraining)
myTraining <- myTraining[, -nzv]
myTesting <- myTesting[, -nzv]

# remove variables that are almostly NA
mostlyNA <- sapply(myTraining, function(x) mean(is.na(x))) > 0.95
myTrainig <- myTraining[, mostlyNA==F]
myTesting <- myTesting[, mostlyNA==F]

# remove identification only variables (columns 1 to 5)
myTraining <- myTrainig[, -(1:5)]
myTesting  <- myTesting[, -(1:5)]
```

Predict Data by various models

1. Random forest

```r
modFit <- randomForest(classe ~ ., data=myTraining)
modFit
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = myTraining)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.29%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3347    1    0    0    0 0.0002986858
## B    4 2274    1    0    0 0.0021939447
## C    0    9 2043    2    0 0.0053554041
## D    0    0   13 1917    0 0.0067357513
## E    0    0    0    4 2161 0.0018475751
```

```
# Prediction using Random forest
predict <- predict(modFit, myTesting, type="class")
confusionMatrix(myTesting$classe, predict)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    0    0    0    1
##          B    0 1515    3    0    0
##          C    0    6 1362    0    0
##          D    0    0   11 1274    1
##          E    0    0    0    4 1438
##
## Overall Statistics
##
##                Accuracy : 0.9967
##                  95% CI : (0.9951, 0.9978)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9958
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9961   0.9898   0.9969   0.9986
## Specificity            0.9998   0.9995   0.9991   0.9982   0.9994
## Pos Pred Value         0.9996   0.9980   0.9956   0.9907   0.9972
## Neg Pred Value         1.0000   0.9991   0.9978   0.9994   0.9997
## Prevalence             0.2843   0.1939   0.1754   0.1629   0.1835
## Detection Rate         0.2843   0.1931   0.1736   0.1624   0.1833
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9999   0.9978   0.9944   0.9975   0.9990
```

2. Decision tree

```
modFit_T <- rpart(classe~., myTraining)

# Prediction using Decision tree
predict_T <- predict(modFit_T, myTesting, type="class")
confusionMatrix(myTesting$classe, predict_T)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1920   69   15  169   59
##          B  113 1096  111  125   73
##          C    2   69 1209   46   42
##          D   56   79  251  744  156
##          E   12   50   97   32 1251
```

```
##
## Overall Statistics
##
##                Accuracy : 0.7928
##                  95% CI : (0.7836, 0.8017)
##     No Information Rate : 0.268
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7385
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9130   0.8041   0.7184  0.66667   0.7913
## Specificity            0.9457   0.9349   0.9742  0.91947   0.9695
## Pos Pred Value         0.8602   0.7220   0.8838  0.57854   0.8675
## Neg Pred Value         0.9674   0.9578   0.9268  0.94329   0.9485
## Prevalence             0.2680   0.1737   0.2145  0.14224   0.2015
## Detection Rate         0.2447   0.1397   0.1541  0.09483   0.1594
## Detection Prevalence   0.2845   0.1935   0.1744  0.16391   0.1838
## Balanced Accuracy      0.9293   0.8695   0.8463  0.79307   0.8804
```

3. Generalized Boosted Model (GBM)

```
control_GBM <- trainControl(method = "repeatedcv", number=5, repeats=1)
modFit_GBM <- train(classe~., myTraining, method="gbm", trControl=control_GBM, verbose=FALSE)
```

```
# Prediction using GBM
predict_GBM <- predict(modFit_GBM, myTesting)
confusionMatrix(predict_GBM, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##          A 2228   11    0    1    0
##          B    3 1490   24    9    7
##          C    0   14 1337   15    4
##          D    1    3    7 1257   18
##          E    0    0    0    4 1413
##
## Overall Statistics
##
##                Accuracy : 0.9846
##                  95% CI : (0.9816, 0.9872)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9805
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9816   0.9773   0.9774   0.9799
## Specificity            0.9979   0.9932   0.9949   0.9956   0.9994
## Pos Pred Value         0.9946   0.9720   0.9759   0.9774   0.9972
## Neg Pred Value         0.9993   0.9956   0.9952   0.9956   0.9955
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1899   0.1704   0.1602   0.1801
## Detection Prevalence   0.2855   0.1954   0.1746   0.1639   0.1806
## Balanced Accuracy      0.9980   0.9874   0.9861   0.9865   0.9896
```

Error

Random forest, Dicision tree, and GBM models give us 99.6 %, 75.4 %, and 98.8 % as accuracy, respectively.

The expected sample errors for Random forest, Dicision tree, and GBM are 0.4 %, 24.6 %, and 1.2 %, respectively.

Final test

Run the algorithm to the 20 test cases in the test data using most accurate model Random forest.

```
predict_test <- predict(modFit, testing_data, type = "class")
predict_test
```

```
##  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
```