Practical Machine Learning Project

BHANU

Jan 14,2017

Prediction Assignment

Background

Now a days with devices like JawboneUp, NikeFuelBand, and Fitbitit it is very easy & relatively inexpensive to collect a large amount of data about personal activity. In many cases enthusiasts who use these devices to find patterns in their behaviour only quantify how much activity they are doing and not how effectively they are doing it.
The theme of the project is 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.The five ways are 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). Only Class A corresponds to correct performance. The goal of this project is to predict the manner in which they did the exercise, i.e., Class A to E.

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.

Goal

The goal of your project is to predict the manner in which they did the exercise.

Preparing R packages

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.2
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.3.2
```

```
library(RColorBrewer)
```

```
## Warning: package 'RColorBrewer' was built under R version 3.3.2
```

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.3.2
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.3.2
```

```
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.3.2
```

```
library(AppliedPredictiveModeling)
```

```
## Warning: package 'AppliedPredictiveModeling' was built under R version
## 3.3.2
```

Loading Data

```
set.seed(12345)

train_url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

training_data <- read.csv(url(train_url), na.strings=c("NA","#DIV/0!",""))
testing_data <- read.csv(url(test_url), na.strings=c("NA","#DIV/0!",""))
```

The entire training data consists 19622 rows of observations and 158 features .While the testing data has 20 rows and the same 158 features. There is one column of target outcome named `classe`.

Data Cleaning

Removing 1st column of training data set(Serial numbers)

```
training_data<-training_data[c(-1)]
```

Removing features with near zero variance as they have little impact in the variation of outcome.

```
nzv <- nearZeroVar(training_data, saveMetrics=TRUE)
training_data <- training_data[,nzv$nzv==FALSE]
```

Now, we remove the features that are having NA value more than 40% of the time.

```
training_1 <- training_data
for(i in 1:length(training_data)) {
  if( sum( is.na( training_data[, i] ) ) /nrow(training_data) >= .41) {
        for(j in 1:length(training_1)) {
            if( length( grep(names(training_data[i]), names(training_1)[j]) ) == 1)  {
```

```
                training_1 <- training_1[ , -j]
            }
        }
    }
}

training_data <- training_1
rm(training_1)
```

Removing features(2nd,3rd,4th columns) that have lesser predicting power.

```
training_data <- training_data[, -c(2:3)]
```

Transform the testing data set

```
k <- colnames(training_data[,c(-56)])
testing_data<-testing_data[k]
```

Data Partitioning

We split the cleaned training set training_data into a training set (70%) for prediction and a cross validation set (30%) for cross validation.

```
set.seed(9999)
TrainData <- createDataPartition(training_data$classe, p=0.6, list=FALSE)
Training_train <- training_data[TrainData, ]
Validation_train <- training_data[-TrainData, ]
dim(Training_train); dim(Validation_train)
```

```
## [1] 11776    56
```

```
## [1] 7846   56
```

PREDICTION ALGORITHMS

PREDICTION USING DECISION TREES

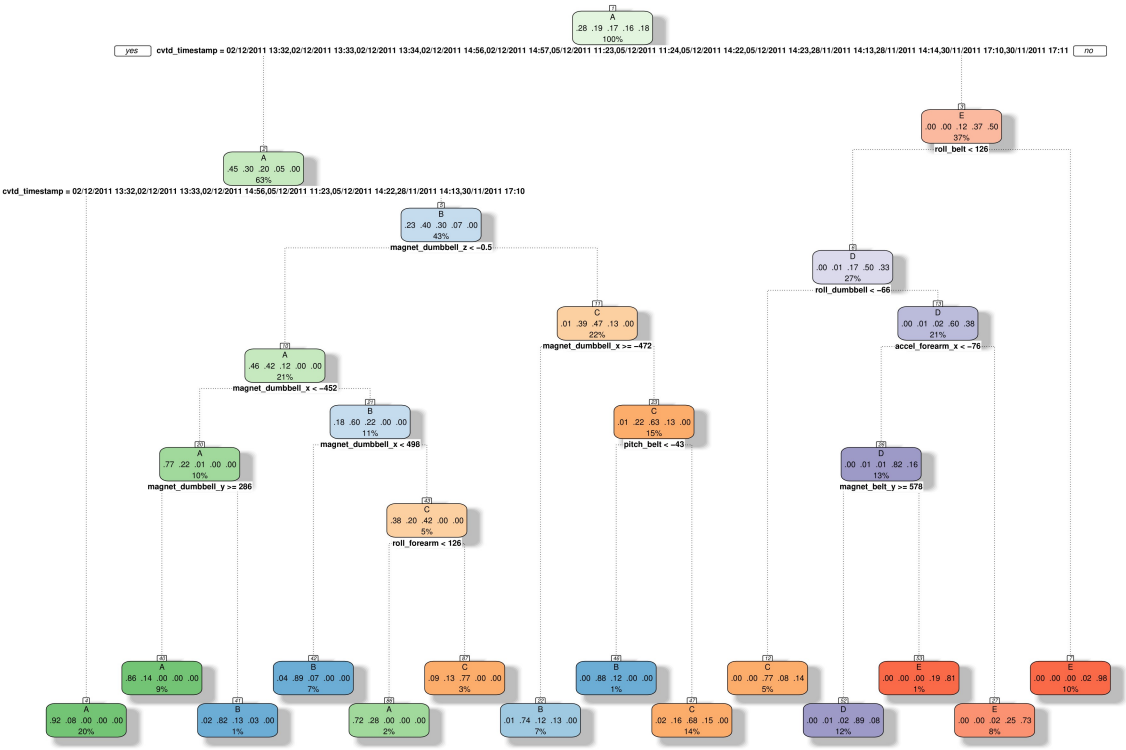
First we use decision tree algorithm to obtain model on Training set and predict for cross validation set.

```
set.seed(9999)
model1 <- rpart(classe ~ ., data=Training_train, method="class")
print(model1,digits=3)
```

```
## n= 11776
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 11776 8430 A (0.28 0.19 0.17 0.16 0.18)
```

```
##    2) cvtd_timestamp=02/12/2011 13:32,02/12/2011 13:33,02/12/2011 13:34,02/12/2011 14:56,02/12/2011
##      4) cvtd_timestamp=02/12/2011 13:32,02/12/2011 13:33,02/12/2011 14:56,05/12/2011 11:23,05/12/201
##      5) cvtd_timestamp=02/12/2011 13:34,02/12/2011 14:57,05/12/2011 11:24,05/12/2011 14:23,28/11/201
##       10) magnet_dumbbell_z< -0.5 2524 1370 A (0.46 0.42 0.12 0.0016 0)
##         20) magnet_dumbbell_x< -452 1201  282 A (0.77 0.22 0.014 0.0033 0)
##           40) magnet_dumbbell_y>=286 1071  154 A (0.86 0.14 0 0 0) *
##           41) magnet_dumbbell_y< 286 130   23 B (0.015 0.82 0.13 0.031 0) *
##         21) magnet_dumbbell_x>=-452 1323  530 B (0.18 0.6 0.22 0 0)
##           42) magnet_dumbbell_x< 498 767   84 B (0.038 0.89 0.072 0 0) *
##           43) magnet_dumbbell_x>=498 556  320 C (0.38 0.2 0.42 0 0)
##             86) roll_forearm< 126 250   69 A (0.72 0.28 0 0 0) *
##             87) roll_forearm>=126 306   70 C (0.095 0.13 0.77 0 0) *
##       11) magnet_dumbbell_z>=-0.5 2594 1380 C (0.013 0.39 0.47 0.13 0)
##         22) magnet_dumbbell_x>=-472 841  219 B (0.0095 0.74 0.12 0.13 0) *
##         23) magnet_dumbbell_x< -472 1753  648 C (0.014 0.22 0.63 0.13 0)
##           46) pitch_belt< -43.2 142   17 B (0 0.88 0.12 0 0) *
##           47) pitch_belt>=-43.2 1611  523 C (0.016 0.16 0.68 0.15 0) *
##    3) cvtd_timestamp=02/12/2011 13:35,02/12/2011 14:58,02/12/2011 14:59,05/12/2011 11:25,05/12/2011
##      6) roll_belt< 126 3156 1590 D (0 0.0051 0.17 0.5 0.33)
##       12) roll_dumbbell< -66 645  149 C (0 0.0047 0.77 0.084 0.14) *
##       13) roll_dumbbell>=-66 2511  999 D (0 0.0052 0.016 0.6 0.38)
##         26) accel_forearm_x< -75.5 1550  281 D (0 0.0084 0.014 0.82 0.16)
##           52) magnet_belt_y>=578 1385  147 D (0 0.0094 0.016 0.89 0.081) *
##           53) magnet_belt_y< 578 165   31 E (0 0 0 0.19 0.81) *
##         27) accel_forearm_x>=-75.5 961  261 E (0 0 0.019 0.25 0.73) *
##      7) roll_belt>=126 1145   18 E (0 0 0 0.016 0.98) *
```

```
fancyRpartPlot(model1)
```



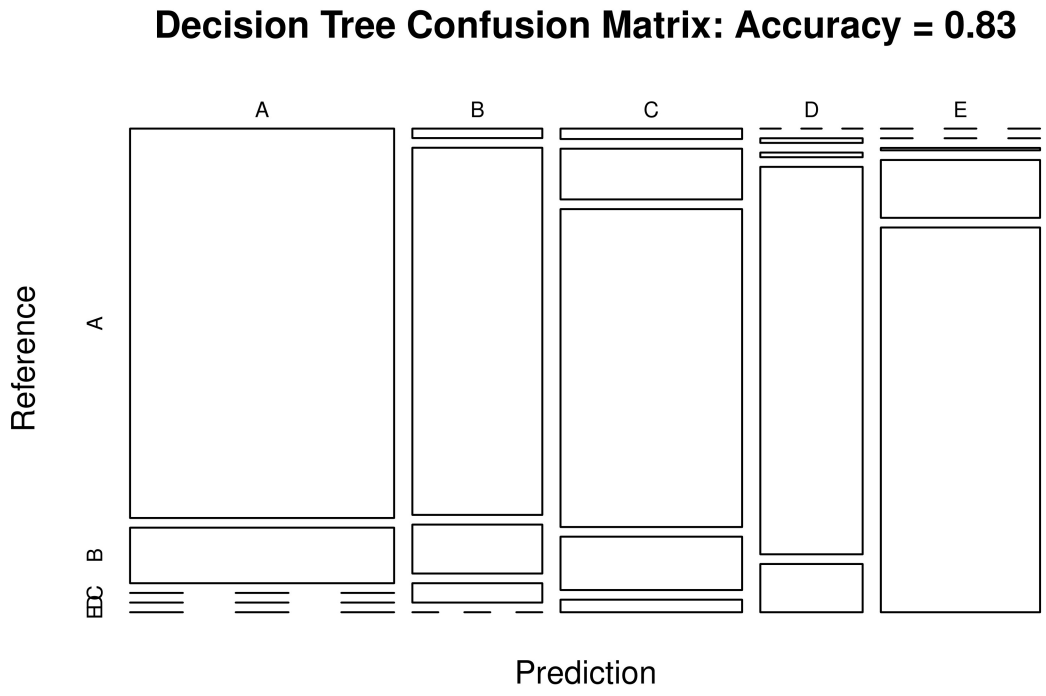
Rattle 2017–Jan–14 19:52:19 Bhanu

```
prediction1 <- predict(model1, Validation_train, type = "class")
conmat <- confusionMatrix(prediction1, Validation_train$classe)
conmat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2166  309    0    0    0
##          B   26 1005  134   53    0
##          C   40  194 1216  204   48
##          D    0   10   10  835  104
##          E    0    0    8  194 1290
##
## Overall Statistics
##
##                Accuracy : 0.83
##                  95% CI : (0.8215, 0.8382)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7841
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9704   0.6621   0.8889   0.6493   0.8946
## Specificity            0.9450   0.9663   0.9250   0.9811   0.9685
## Pos Pred Value         0.8752   0.8251   0.7145   0.8707   0.8646
## Neg Pred Value         0.9877   0.9226   0.9753   0.9345   0.9761
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2761   0.1281   0.1550   0.1064   0.1644
## Detection Prevalence   0.3154   0.1552   0.2169   0.1222   0.1902
## Balanced Accuracy      0.9577   0.8142   0.9069   0.8152   0.9315
```

```
accuracy_conmat <- conmat$overall[1]
plot(conmat$table, col = conmat$byClass, main = paste("Decision Tree Confusion Matrix: Accuracy =", rou
```



From the confusion matrix, the accuracy is 0.83, and so the out-of-sample error rate is 0.17. ### PREDICTION USING RANDOM FORESTS We now use random forest algorithm to obtain model on Training set and predict for cross validation set.

```
set.seed(9999)
model2 <- randomForest(classe ~ ., data=Training_train,method = "class")
print(model2,digits=3)
```

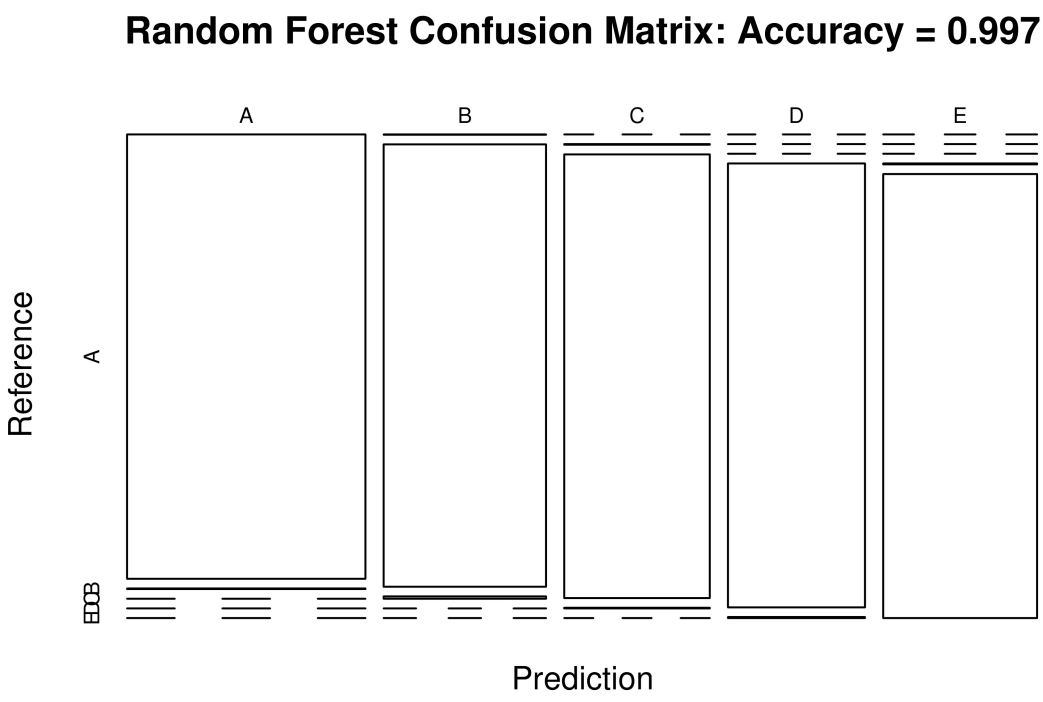
```
##
## Call:
##  randomForest(formula = classe ~ ., data = Training_train, method = "class")
##                Type of random forest: classification
##                      Number of trees: 500
```

```
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.28%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3347    1    0    0    0 0.0002986858
## B    4 2273    2    0    0 0.0026327337
## C    0    8 2043    3    0 0.0053554041
## D    0    0   10 1920    0 0.0051813472
## E    0    0    0    5 2160 0.0023094688
```

```
prediction2 <- predict(model2, Validation_train, type = "class")
conmat2 <- confusionMatrix(prediction2, Validation_train$classe)
conmat2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    3    0    0    0
##          B    1 1513    8    0    0
##          C    0    2 1360    2    0
##          D    0    0    0 1281    3
##          E    0    0    0    3 1439
##
## Overall Statistics
##
##                Accuracy : 0.9972
##                  95% CI : (0.9958, 0.9982)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9965
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9967   0.9942   0.9961   0.9979
## Specificity            0.9995   0.9986   0.9994   0.9995   0.9995
## Pos Pred Value         0.9987   0.9941   0.9971   0.9977   0.9979
## Neg Pred Value         0.9998   0.9992   0.9988   0.9992   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1928   0.1733   0.1633   0.1834
## Detection Prevalence   0.2847   0.1940   0.1738   0.1637   0.1838
## Balanced Accuracy      0.9995   0.9976   0.9968   0.9978   0.9987
```

```
plot(conmat2$table, col = conmat2$byClass, main = paste("Random Forest Confusion Matrix: Accuracy =", r
```



From the confusion matrix, the accuracy is 0.997, and so the out-of-sample error is 0.003.

PREDICTION ON TEST SET

Since accuracy obtained by Random forests algorithm is very high compared to decision trees algorithm & out-of-sample error is low(0.003 against 0.17), we now use ***model2*** to predict the `classe` for our test set.

```
common <- intersect(names(Training_train), names(testing_data))
for (p in common) {
  if (class(Training_train[[p]]) == "factor") {
    levels(testing_data[[p]]) <- levels(Training_train[[p]])
  }
}
predict(model2, testing_data)
```

```
##  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
```