Practical Machine Learning Course Project

Paymon Hashemi

Saturday, March 19, 2016

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. In this project, we will use data from accelerometers on the belt, forearm, arm, and dumbell of six (6) participants to predict the manner in which they did the exercise.

The training and test data sets came from the following source: Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6_6.

Read more: http://groupware.les.inf.puc-rio.br/har#sbia_paper_section#ixzz43NCRdV9T (http://groupware.les.inf.puc-rio.br/har#sbia_paper_section#ixzz43NCRdV9T)

Data Preprocessing

```
library(caret)
```

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

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

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(e1071)
library(gbm)
```

```
## 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
```

Download the Data We will download the training and testing files.
Note: We will only alter the training file.

```
TrainUrl <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
TestUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
TrainFile <- "C:/Users/PAYMON/data/pml-training.csv"
TestFile  <- "C:/Users/PAYMON/data/pml-testing.csv"
if (!file.exists("C:/Users/PAYMON/data")) {
  dir.create("C:/Users/PAYMON/data")
}
if (!file.exists(TrainFile)) {
  download.file(TrainUrl, destfile=TrainFile, method="curl")
}
if (!file.exists(TestFile)) {
  download.file(TestUrl, destfile=TestFile, method="curl")
}
```

Read the training and test data sets After downloading the two data sets, we can read the two csv files into two data frames.

```
TrainFull <- read.csv("C:/Users/PAYMON/data/pml-training.csv")
TestFull <- read.csv("C:/Users/PAYMON/data/pml-testing.csv")
```

Splitting the data The testing file contains 20 observations of 160 varibles. The training file contains 19622 observations of 160 variables. In order to clean the data, we will split the training data set (TrainFull) into a pure training data set (60% – Train1) and a validation data set (40% – Train2). We will use the validation data set to conduct a cross validation.

```
set.seed(10928)
inTrain <- createDataPartition(y=TrainFull$classe, p=0.6, list=F)
Train1 <- TrainFull[inTrain, ]
Train2 <- TrainFull[-inTrain, ]
```

We are now going to reduce the number of features by removing variables with nearly zero variance, variables that are almost always NA, and variables that not pertinent to prediction.

Data Cleaning # Remove variables that are NA

```
varNA <- sapply(Train1, function(x) mean(is.na(x))) > 0.95
Train1 <- Train1[, varNA==F]
Train2 <- Train2[, varNA==F]
```

This action removed 67 variables (93 remaining).

Remove variables with near zero variance

```
nzv <- nearZeroVar(Train1)
Train1 <- Train1[, -nzv]
Train2 <- Train2[, -nzv]
```

This action removed 34 variables (59 remaining).

Remove identification only variables (columns 1 to 5)

```
Train1 <- Train1[, -(1:5)]
Train2  <- Train2[, -(1:5)]
```

This action removed 5 variables (54 remaining). The validation data set (Train2) contains 7846 observations from 59 variables. The “classe” variable is in the validation data set (Train2).

Data Modeling using Train2 We will fit a predictive model for activity recognition using the Random Forest and GBM algorithms because they are the most accurate (using the most accurate one). We will use 5-fold cross validation when applying the algorithms.

Model fitting for Random Forest

```
set.seed(10928)
ControlRf <- trainControl(method="cv", number=5, verboseIter=FALSE)
modelRf <- train(classe ~ ., data=Train1, method="rf", trControl=ControlRf)
modelRf$finalModel
```

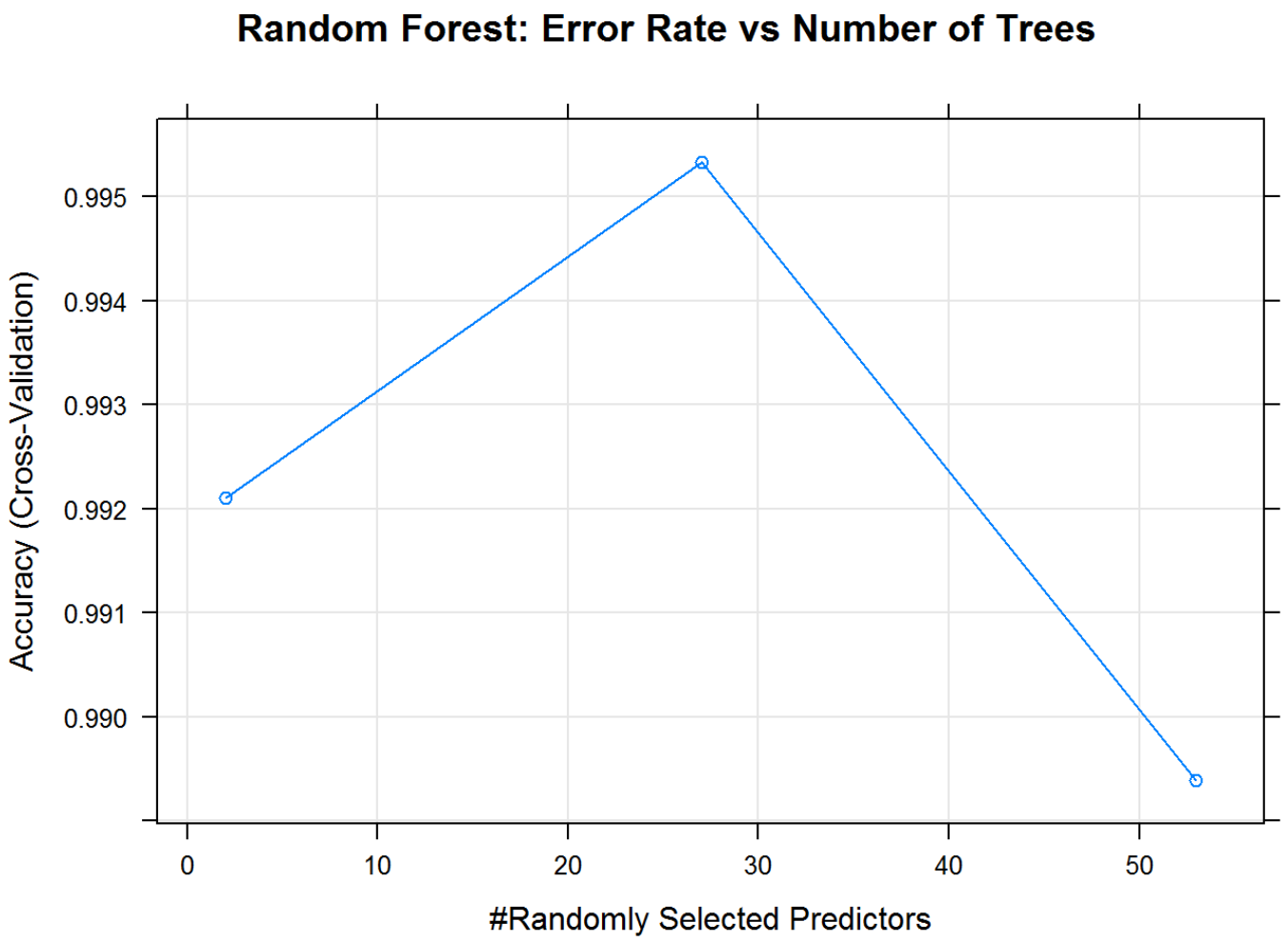
```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.37%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3346    1    0    0    1 0.0005973716
## B    9 2267    3    0    0 0.0052654673
## C    0    8 2043    3    0 0.0053554041
## D    0    0   11 1918    1 0.0062176166
## E    0    0    0    7 2158 0.0032332564
```

Then, we will estimate the performance of the model on the validation data set followed by a plot of the error rate. We will repeat the same process for the GBM method.

```
predictRf <- predict(modelRf, newdata=Train2)
confusionMatrixRf <- confusionMatrix(predictRf, Train2$classe)
confusionMatrixRf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    5    0    0    0
##          B    1 1512    3    0    0
##          C    0    1 1365    9    0
##          D    0    0    0 1277    4
##          E    0    0    0    0 1438
##
## Overall Statistics
##
##                Accuracy : 0.9971
##                  95% CI : (0.9956, 0.9981)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9963
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9960   0.9978   0.9930   0.9972
## Specificity            0.9991   0.9994   0.9985   0.9994   1.0000
## Pos Pred Value         0.9978   0.9974   0.9927   0.9969   1.0000
## Neg Pred Value         0.9998   0.9991   0.9995   0.9986   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1927   0.1740   0.1628   0.1833
## Detection Prevalence   0.2850   0.1932   0.1752   0.1633   0.1833
## Balanced Accuracy      0.9993   0.9977   0.9981   0.9962   0.9986
```

```
plot(modelRf,main="Random Forest: Error Rate vs Number of Trees")
```



Model fitting for GBM

```
set.seed(10928)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modelGBM  <- train(classe ~ ., data=Train1, method = "gbm",
                   trControl = controlGBM, verbose = FALSE)
```

```
## Loading required package: plyr
```

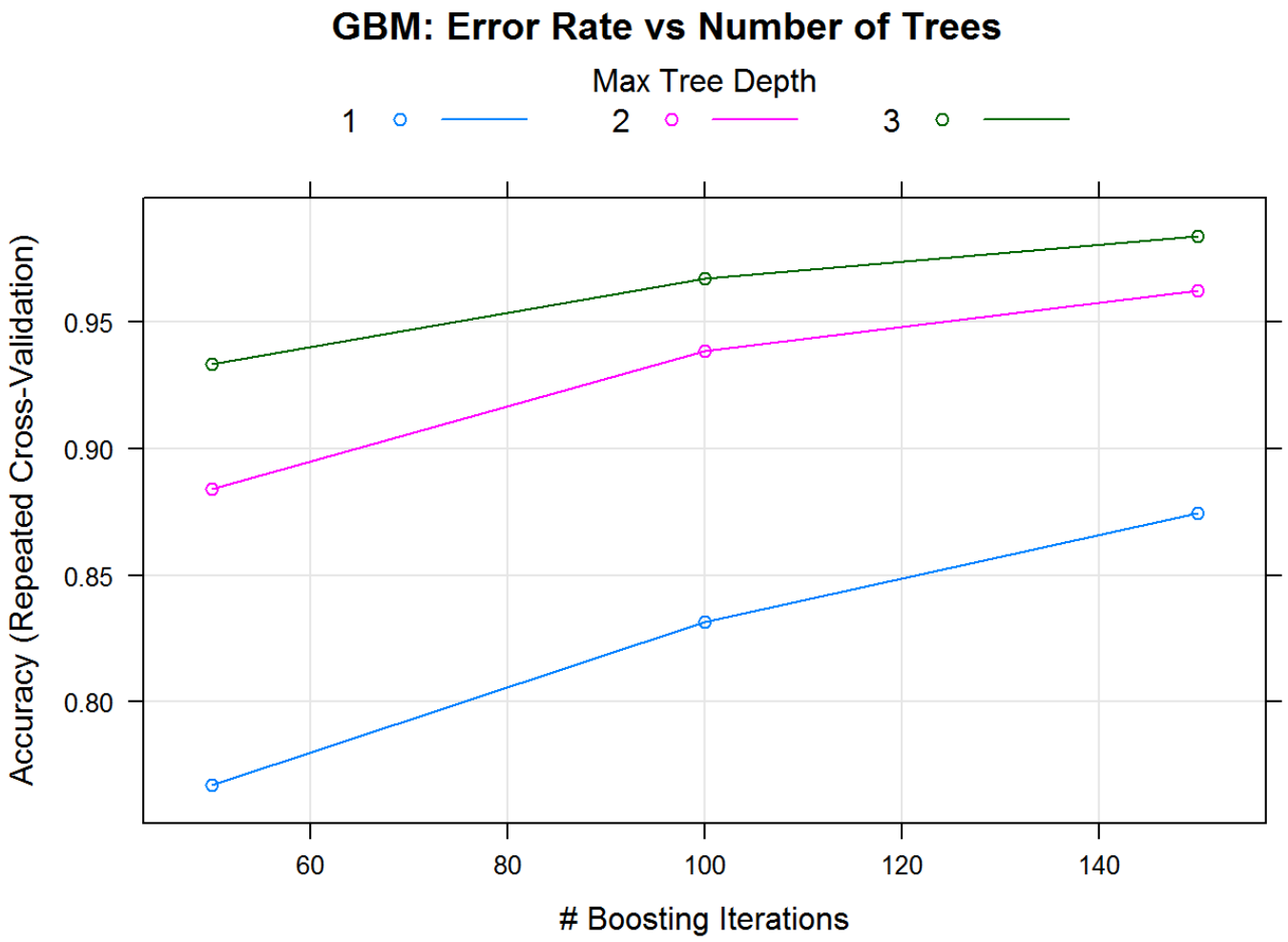
```
modelGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 41 had non-zero influence.
```

```
predictGBM <- predict(modelGBM, newdata=Train2)
confusionMatrixGBM <- confusionMatrix(predictGBM, Train2$classe)
confusionMatrixGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2224   19    0    2    1
##          B    8 1477   20    7    7
##          C    0   20 1339   14    3
##          D    0    1    9 1260    9
##          E    0    1    0    3 1422
##
## Overall Statistics
##
##                Accuracy : 0.9842
##                  95% CI : (0.9812, 0.9868)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.98
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9730   0.9788   0.9798   0.9861
## Specificity            0.9961   0.9934   0.9943   0.9971   0.9994
## Pos Pred Value         0.9902   0.9724   0.9731   0.9851   0.9972
## Neg Pred Value         0.9986   0.9935   0.9955   0.9960   0.9969
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2835   0.1882   0.1707   0.1606   0.1812
## Detection Prevalence   0.2863   0.1936   0.1754   0.1630   0.1817
## Balanced Accuracy      0.9962   0.9832   0.9865   0.9884   0.9928
```

```
plot(modelGBM,main="GBM: Error Rate vs Number of Trees")
```



Upon review of the plots showing the error rates for Random Forest and GBM, the latter clearly demonstrates a stronger rate (i.e. closer to accuracy of 1); therefore, we will apply Random Forest to the Test data set

Application of Random Forest to Test Data Set

Now, we apply the model to the original testing data set.

```
TestFullPredict <- predict(modelRf, newdata=TestFull)
TestFullPredict
```

```
##  [1] 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
```