Predict Body Postures and Movements

Practical Machine Learning by Andrea

May 13, 2017

Introduction

Advances in technology have made it possible to collect data about a person's activity using devices such Jawbone Up, Nike FuelBand, and Fitbit. Individuals wearing this type of device record various types of measurements relating to physical activity. This information can then be used to assist in improving one's health. However, while the measurements quantify how much of the physical activity was completed, the device does not quantify how well the individual did. Therefore, the goal of this project is to use the data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they did the exercise.

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)."

```
# Retrieving the Data
target <-"pml-training.csv"
if (!file.exists(target)) {
   url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
   target <- "pml-training.csv"
   download.file(url, destfile = target)
}
# replace missing values and division errors with NA
sample <-read.csv(target, header=T, sep=",", na.strings=c("NA","#DIV/0!"))

target <-"pml-testing.csv"
if (!file.exists(target)) {
   url <-"http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
   download.file(url, destfile = target)
}
quiz <-read.csv(target, header=T, sep=",", na.strings=c("NA","#DIV/0!"))
```

Cross Validation

We learn that it is not advisable to compare the predictive accuracy of the model using the same dataset used for estimating the model. Therefore, in order to assess the model's predictive performance, an independent set of data, the test dataset, will be used.

Since this dataset appears to be medium-sized, the original dataset pml-training.csv dataset will be randomly sliced into two parts: a training set (60%) and a test set (40%). The training set will be used to fit the models. The test set will be used for assessment of the generalization (out-of-sample) error of the final chosen model. The final prediction model will be used to predict 20 different test cases.

```
seed <- 777
set.seed(seed)
#create training set indexes with 60% of data
sample_data <-createDataPartition(y=sample$classe, p=0.6, list=FALSE)
#subset sample data to training
training <- sample[sample_data, ]
```

```r
#subset sample data (the rest) to test
testing <- sample[-sample_data, ]
#dimension of original, training, testing, and validation datasets
rbind("original dataset" = dim(sample),"training set"=dim(training),"testing set"=dim(testing),"validat
```

```
##                   [,1] [,2]
## original dataset 19622  160
## training set     11776  160
## testing set       7846  160
## validation          20  160
```

Data Cleaning

It is assumed that the observations with missing values are missing completely at random; therefore, columns with a large amount of missing values are discarded. In addition, variables that have very little variability in them will be removed since they are not useful predictors.

```r
#count number of missing values in each dataset
sum(is.na(training)==TRUE)
```

```
## [1] 1155161
```

```r
sum(is.na(testing)==TRUE)
```

```
## [1] 769941
```

```r
#calculate the percentage of missing values in each of the variables in the original train and test dat
NApercentTrain <- sapply(training, function(df) {sum(is.na(df)==TRUE)/length(df)})
NApercentTest <- sapply(testing, function(df) {sum(is.na(df)==TRUE)/length(df)})
#remove variables that have more than 95% missing values from testing and training datasets
colnames1 <- names(which(NApercentTrain < 0.95))
trainingData <- training[, colnames1]
colnames2 <- names(which(NApercentTest < 0.95))
testingData <- testing[, colnames2]
#Recheck for Missing Values
#makes sure both datasets are free of missing values
sum(is.na(trainingData)==TRUE); sum(is.na(testingData)==TRUE)
```

```
## [1] 0
```

```
## [1] 0
```

```r
#Remove variables that are not useful for Prediction Models
#Identify variables that have very little variability
nzv_train<-nearZeroVar(trainingData,saveMetrics=TRUE)
nzv_test<-nearZeroVar(testingData,saveMetrics=TRUE)
#remove all variables with nzv = TRUE because predictor is a near-zero-variance predictor
SubCleanTrainData <- trainingData[,which(nzv_train$nzv==FALSE)]
SubCleanTestData <- testingData[,which(nzv_test$nzv==FALSE)]
#remove x and cvtd_timestamp columns
CleanTrainData <- SubCleanTrainData[,c(2:4,6:59)]
CleanTestData <- SubCleanTestData[,c(2:4,6:59)]
```

Prediction Model Consideration

According to Hastie, Tibshirani, and Friedman (2009), "random forests do remarkably well, with very little tuning required" (p. 590). This project will compare the two most widely used and accurate prediction models Random Forest and Boosting.

Prediction using Random Forest

```
set.seed(seed)
#build model on sub-training set
model <-"rfModFit.RData"
if (!file.exists(model)) {
        # Start the clock!
        my.date <- as.character(Sys.time())
        ptm <- proc.time()
        #fit the outcome to be classe and to use any of the other predictive variables as potential pre
        #modFit <- train(classe~ .,data=CleanTrainData, method="rf", prox=TRUE)
        #use randomForest function as it is faster than train()
        rfmodFit <- randomForest(classe~ .,data=CleanTrainData)
        save(rfmodFit, file="rfModFit.RData")
        proc.time() - ptm
        my.enddate <- as.character(Sys.time())
        my.date
        my.enddate
} else {
        load(file="rfModFit.RData", verbose=FALSE)
}
#Evaluate with Random Forest on sub-train set to capture in-sample error
rftrainPC <- predict(rfmodFit, CleanTrainData)
rftrainAcc <- confusionMatrix(CleanTrainData$classe, rftrainPC)$overall

#Evaluate with Random Forest on sub-test set
rftestPC <-predict(rfmodFit,CleanTestData)
```

* *In-Sample Error*

Now let's look at the error that resulted from applying the Random Forest prediction algorithm to the dataset it was built with, the train datset.

```
rftrainAcc[1]
```

```
## Accuracy
##        1
```

With 100% accuracy, there is no resubstitution (in-sample) error

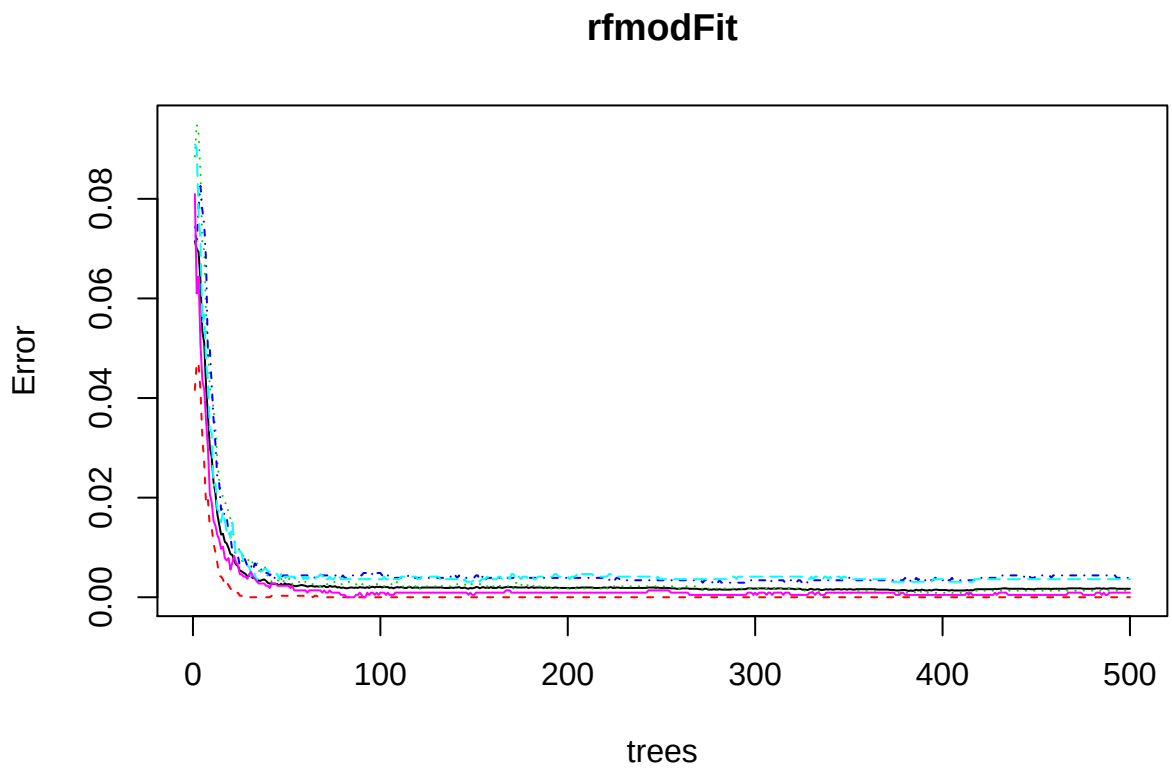
* *Out-of-Sample Error*

Now let's look at the error that resulted from applying the Random Forest prediction algorithm to a new dataset, the test dataset.

The generalization (out-of-sample) error is 0.0025491

Review Accuracy Results on Test Prediction (RF)

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    3    0    0    0
##          B    0 1515    6    0    0
##          C    0    0 1362   11    0
##          D    0    0    0 1275    0
##          E    0    0    0    0 1442
##
## Overall Statistics
##
##                Accuracy : 0.9975
##                  95% CI : (0.9961, 0.9984)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9968
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9980   0.9956   0.9914   1.0000
## Specificity            0.9995   0.9991   0.9983   1.0000   1.0000
## Pos Pred Value         0.9987   0.9961   0.9920   1.0000   1.0000
## Neg Pred Value         1.0000   0.9995   0.9991   0.9983   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1931   0.1736   0.1625   0.1838
## Detection Prevalence   0.2849   0.1939   0.1750   0.1625   0.1838
## Balanced Accuracy      0.9997   0.9985   0.9970   0.9957   1.0000
```



Prediction using Generalized Boosting

```
set.seed(seed)
#build model on sub-training set
model <-"gbmModFit.RData"
if (!file.exists(model)) {
        # Start the clock!
        gbmmy.date <- as.character(Sys.time())
        ptm <- proc.time()
        gbmmodFit <- train(classe~ .,data=CleanTrainData, method="gbm", verbose=FALSE)
        save(gbmmodFit, file="gbmModFit.RData")
        proc.time() - ptm
        gbm.enddate <- as.character(Sys.time())
} else {
        load(file="gbmModFit.RData", verbose=FALSE)
}
```

```
## Stochastic Gradient Boosting
##
## 11776 samples
##    56 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
```

```
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.8105748  0.7594607
##   1                  100      0.8849067  0.8541751
##   1                  150      0.9141674  0.8912809
##   2                   50      0.9328351  0.9149287
##   2                  100      0.9745053  0.9677213
##   2                  150      0.9884133  0.9853329
##   3                   50      0.9644226  0.9549454
##   3                  100      0.9903004  0.9877224
##   3                  150      0.9942198  0.9926837
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.

## Loading required package: plyr
```

*** *In-Sample Error***

Now let's look at the error that resulted from applying the Generalized Boosting prediction algorithm to the dataset it was built with, the train datset.

```
##  Accuracy
## 0.9988961
```

With 99.89% accuracy, resubstitution (in-sample) error is 0.0011039.

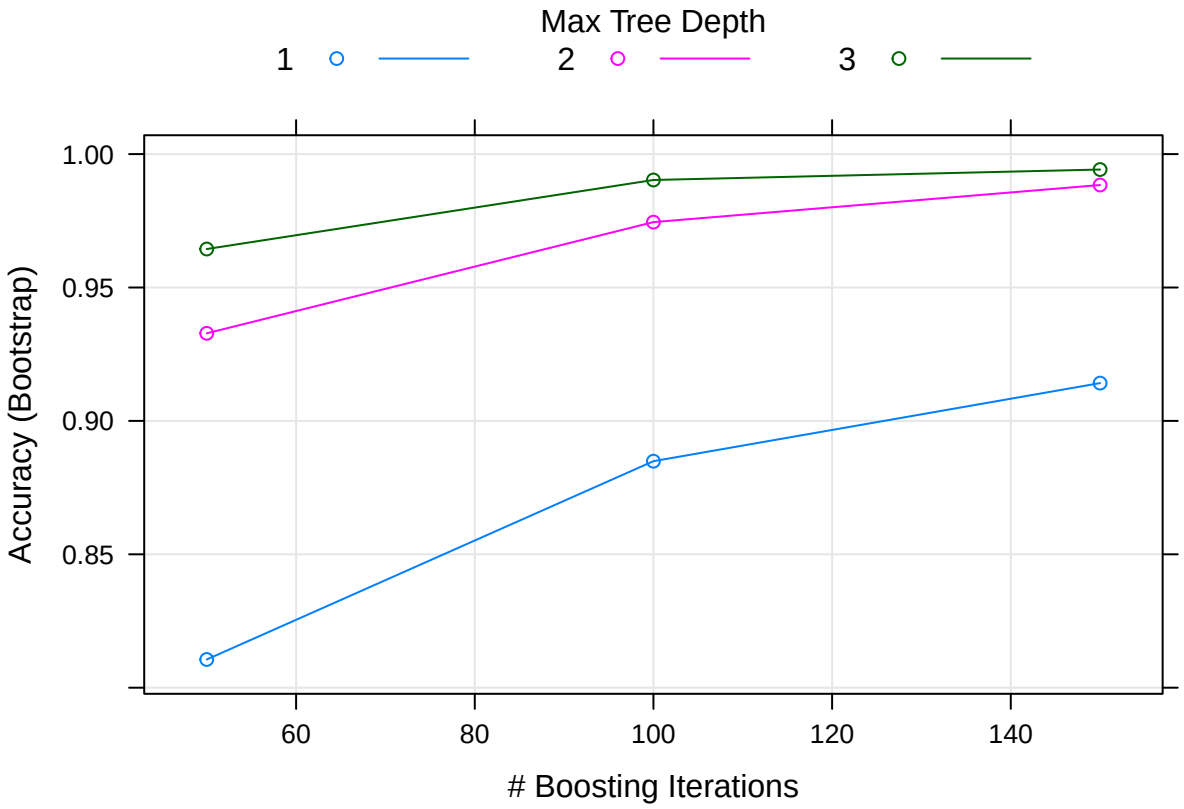
*** *Out-of-Sample Error***

Now let's look at the error that resulted from applying the Generalized Boosting prediction algorithm to a new dataset, the test dataset. The generalization (out-of-sample) error is 0.0043334.

Review Accuracy Results on Test Prediction (GBM)

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    8    0    0    0
##          B    0 1508    3    0    0
##          C    0    2 1357    8    0
##          D    0    0    8 1275    2
##          E    0    0    0    3 1440
##
## Overall Statistics
##
##                Accuracy : 0.9957
##                  95% CI : (0.9939, 0.997)
##     No Information Rate : 0.2845
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9945
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9934   0.9920   0.9914   0.9986
## Specificity            0.9986   0.9995   0.9985   0.9985   0.9995
## Pos Pred Value         0.9964   0.9980   0.9927   0.9922   0.9979
## Neg Pred Value         1.0000   0.9984   0.9983   0.9983   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1922   0.1730   0.1625   0.1835
## Detection Prevalence   0.2855   0.1926   0.1742   0.1638   0.1839
## Balanced Accuracy      0.9993   0.9965   0.9952   0.9950   0.9991
```



Prediction Model Selection

The comparison of the Random Forest and Generalized Boosting models reveals that the Random Forest model at 99.75% is more accurate than the Generalized Boosting model at 99.57%.

Prediction using the Random Forest model will be used for Validation (the project quiz).

Prediction Quiz using Random Forest

```
predquiz <- predict(rfmodFit, quiz)
names(predquiz) <- c(1:20)
predquiz
```

```
##  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
```

```
#assign path for quiz
path <- "~/Coursera/Pratical Machine Learning/Project/Quiz"
pml_write_files = function (x){
        n = length(x)
        for(i in 1:n) {
                filename = paste0("problem_id_",i,".txt")
                write.table(x[i],file=file.path(path, filename),quote=FALSE,row.names=FALSE,col.names=F
        }
}
pml_write_files(predquiz)
```

Prediction on Quiz dataset resulted in 100% Accuracy.

References

Hastie, T., Tibshirani, R., & Friedman, J. H. (2009). The elements of statistical learning data mining, inference, and prediction (2nd ed.). New York, NY: Springer.