Practical Machine Learning: Prediction

INTRODUCTION

Goal is to use a Machine Learning Algorithm to predict how well 20 test cases did their exercises correctly from the data obtained from accelerometers on the belt, forearm, arm, and dumbell.

The raw data is already partioned into training and testing set. The training set is further partioned, 70:30, into two sets–train and validate. The train set is used to build and train the models, and the validate set is used to pick the best model based on the lowest out-of-sample error rate. This model is used to predict the 20 test cases.

Environment

Load the libraries needed for analysis and the set the seed for reproducibility

```
library(caret)
```

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

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

```
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(rpart)
set.seed(1029)
```

Get the Data

The raw data is already partioned into the training set and the testing set.

```
setwd("C:/Users/loy_c/Desktop/Rdata/11 Practical Machine Learning")
training <- read.csv("pml-training.csv")
testing <- read.csv("pml-testing.csv")
```

DATA EXPLORATION

Work with the training set to train the models. The testing set is not modified or examined.

```
dim(training)
```

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

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

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

The data has 160 predictor variables, and the output vaiable, classe, is a factor variable with 5 levels. The first level, A, (correctly doing the excercise) has frequency of 5580, and the other 4 levels (incorrectly doing the exercises) are about equal with frequencies in the mid 3000s.

PRE-PROCESS THE DATA

We only pre-process the training data, not the testing data. Drop the first 7 columns of the training set as they are for information purposes only, like user name and time stamps.

```
training  <- subset(training, select=-c(1:7))
dim(training)
```

```
## [1] 19622   153
```

Drop 59 predictors that contain no information, i.e., near zero variance, from the training set

```
noInfoColumns <- nearZeroVar(training)
training <- training[, -noInfoColumns]
dim(training)
```

```
## [1] 19622    94
```

Drop another 41 empty columns from the training set

```
training<-training[,colSums(is.na(training)) == 0]
dim(training)
```

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

PARTITION DATA

Carve out a validation set, validate, that is 30% of the training set–this is the "out-of-sample" data. The remaining 70% of the training set is the train data set.

```
inBuild <- createDataPartition(y=training$classe, p=0.7, list=FALSE)
train <- training[inBuild,]
validate <- training[-inBuild,]
dim(train)
```

```
## [1] 13737    53
```

```
dim(validate)
```

```
## [1] 5885   53
```

MODEL BUILDING

Build Random Forest and Decision Tree models on the train data set

```r
modelRF <- randomForest(classe ~ ., method="class", data=train)
modelDT <- rpart(classe ~ ., method="class", data=train)
```

Validate these models on the validate (out-of-sample) data set

```r
validateRF <- predict(modelRF, validate, type="class")
validateDT <- predict(modelDT, validate, type="class")
```

Build a GAM stacked model of a Random Forest and Decision Tree on the train data set, and validate it against the validate data set

```r
stackDF <- data.frame(validateRF, validateDT, classe=validate$classe)
modelSTACK <- train(classe ~., method="gam", data=stackDF)
```

```
## Loading required package: mgcv

## Loading required package: nlme

## This is mgcv 1.8-17. For overview type 'help("mgcv-package")'.

## Warning in gam.fit3(x = args$X, y = args$y, sp = lsp, Eb = args$Eb, UrS =
## args$UrS, : fitted probabilities numerically 0 or 1 occurred

## Warning in gam.fit3(x = args$X, y = args$y, sp = lsp, Eb = args$Eb, UrS =
## args$UrS, : fitted probabilities numerically 0 or 1 occurred

## Warning in gam.fit3(x = args$X, y = args$y, sp = lsp, Eb = args$Eb, UrS =
## args$UrS, : fitted probabilities numerically 0 or 1 occurred

## Warning in gam.fit3(x = args$X, y = args$y, sp = lsp, Eb = args$Eb, UrS =
## args$UrS, : fitted probabilities numerically 0 or 1 occurred

## Warning in gam.fit3(x = args$X, y = args$y, sp = lsp, Eb = args$Eb, UrS =
## args$UrS, : fitted probabilities numerically 0 or 1 occurred

## Warning in gam.fit3(x = args$X, y = args$y, sp = lsp, Eb = args$Eb, UrS =
## args$UrS, : fitted probabilities numerically 0 or 1 occurred

## Warning in gam.fit3(x = args$X, y = args$y, sp = lsp, Eb = args$Eb, UrS =
## args$UrS, : fitted probabilities numerically 0 or 1 occurred

## Warning in gam.fit3(x = args$X, y = args$y, sp = lsp, Eb = args$Eb, UrS =
## args$UrS, : fitted probabilities numerically 0 or 1 occurred

## Warning in gam.fit3(x = args$X, y = args$y, sp = lsp, Eb = args$Eb, UrS =
## args$UrS, : fitted probabilities numerically 0 or 1 occurred

## Warning in gam.fit3(x = args$X, y = args$y, sp = lsp, Eb = args$Eb, UrS =
## args$UrS, : fitted probabilities numerically 0 or 1 occurred

## Warning in gam.fit3(x = args$X, y = args$y, sp = lsp, Eb = args$Eb, UrS =
## args$UrS, : fitted probabilities numerically 0 or 1 occurred

## Warning in gam.fit3(x = args$X, y = args$y, sp = lsp, Eb = args$Eb, UrS =
## args$UrS, : fitted probabilities numerically 0 or 1 occurred

## Warning in gam.fit3(x = args$X, y = args$y, sp = lsp, Eb = args$Eb, UrS =
```

```
## args$UrS, : fitted probabilities numerically 0 or 1 occurred

## Warning in gam.fit3(x = args$X, y = args$y, sp = lsp, Eb = args$Eb, UrS =
## args$UrS, : fitted probabilities numerically 0 or 1 occurred

## Warning in gam.fit3(x = args$X, y = args$y, sp = lsp, Eb = args$Eb, UrS =
## args$UrS, : fitted probabilities numerically 0 or 1 occurred
```

```
validateSTACK <- predict(modelSTACK, validate)
```

MODEL SELECTION

Pick the most accurate model, i.e., the one with the smallest out-of-sample error. The model's accuracy is obtained from the Confusion Matrix.

```
modelAccuracy <- rbind(confusionMatrix(validate$classe, validateRF)$overall[1],
                  confusionMatrix(validate$classe, validateDT)$overall[1],
                  confusionMatrix(validate$classe, validateSTACK)$overall[1])
row.names(modelAccuracy) <- c("RF", "DT", "GAM STACK")
modelAccuracy
```

```
##            Accuracy
## RF        0.9937128
## DT        0.7369584
## GAM STACK 0.4769754
```

The Random Forest Model, modelRF, is selected because it has the lowest out-of-sample error, less than 1%, i.e., it had the highest accuracy of over 99%, as shown above.

PREDICT RESULTS

Apply the selected model, modelRF, to predict the values of the 20 test cases in the testing data set.

```
predictTEST <- predict(modelRF, testing, type="class")
predictTEST
```

```
##  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
```

CONCLUSION

Three machine learning models, Random Forest, Decision Tree, and General Additive Model (GAM) which is a stacked model of the radom forest and the decision tree, were trained on the training data, and the Random Forest model was selected to predict the 20 test cases. The Random Forest was selected because it had the lowest out of sample error of less than 1%. The stacked GAM model geneated a ton of warnings and returned P value of 1, which means the GAM model failed to find a predictor.