Machine Learning Peer Graded Assignment

Stephen Armah

August 9, 2016

Objective

This assignment tasks me with predicting how well participants performed various fitness activities. To do so , I will load the training and testing data, perform exploratory analysis, build models and select the best one, and apply it to the testing data.

Ther first activity is to load appropriate libraries and thedata sets.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.5
```

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.2.5
```

```
library(survival)
library(splines)
library(parallel)
library(plyr)
library(MASS)
set.seed(8675309)

## this is a large data set so clear memory
rm(list=ls())

## load data
training <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", na.
strings ="#DIV/0!")
testing  <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", na.s
trings ="#DIV/0!")
```

Exploratory Analysis

Now we perform exploratory analysis on the training data set for insight into how to proceed.

```
summary(training$skewness_yaw_belt)
```

```
##    Mode    NA's
## logical   19622
```

A summary of the training data set reveals that a great many variables have a prohibitive number of null or NA values. The skewness_yaw_belt variables (above) is an example. To improve model quality and expedite computations, I subset the data set and remove columns that have greater than 90% NA values.

Data Clensing

```
## remove columns that are not measurements
training <- training[,-(1:5)]
testing  <- testing[,-(1:5)]

## determine ratio of NAs to data in each column
x <- 0
for(i in 1:length(training)) {
  x[i] <- print(sum(is.na(training[,i]))/nrow(training))
}

## remove columns with NAs > 90%
training <- subset(training, select = x < 0.90)
testing  <- subset(testing, select = x < 0.90)

## remove columns with very low variances
noVar <- nearZeroVar(training)
training <- training[,-noVar]
testing  <- testing[,-noVar]
```

Cross Validation

Subsample training into subTrain and subTest data sets. Model on subTrain, evaluate on subTest. Find best model and apply to testing

```
## Sub-sample the training data set into a sub training and testing data set
inTrain  <- createDataPartition(y=training$classe, p=0.75, list=FALSE)
subTrain <- training[inTrain,]
subTest  <- training[-inTrain,]
```

Building Models

I fit multiple models to the data to determnine the one with the highest accuracy.

```
## random forest
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modRF  <- train(subTrain$classe~., method="rf", data=subTrain, trControl=controlRF)
```

```
## Loading required package: randomForest
## Warning: package 'randomForest' was built under R version 3.2.5
```

```
## 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
```

```
outRF   <- predict(modRF, subTest)
conRF  <- confusionMatrix(subTest$classe, outRF)

## decision tree
modRP   <- train(subTrain$classe~., method="rpart", data=subTrain)
```

```
## Loading required package: rpart
```

```
outRP   <- predict(modRP, subTest)
conRP  <- confusionMatrix(subTest$classe, outRP)

## LDA
modLDA  <- train(subTrain$classe~., method="lda", data=subTrain)
outLDA  <- predict(modLDA, subTest)
conLDA <- confusionMatrix(subTest$classe, outLDA)

## generalized booster
modGBM  <- train(subTrain$classe~., method="gbm", data=subTrain, verbose=FALSE)
outGBM  <- predict(modGBM, subTest)
conGBM <- confusionMatrix(subTest$classe, outGBM)
```

We now review the accuracy for each model tested.

```
conRF$overall[1]
```

```
##  Accuracy
## 0.9969413
```

```
conRP$overall[1]
```

```
## Accuracy
## 0.487969
```

```
conLDA$overall[1]
```

```
##  Accuracy
## 0.7077896
```

```
conGBM$overall[1]
```

```
##  Accuracy
## 0.9883768
```

Out of Sample Error

We see from the Confusion Matrix that the Random Forest model has a very low out of sample error, the error rate on the new data. Thus, we believe this will allow us to achieve the highest accuracy on our predictions

```
print(conRF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1393    1    0    0    1
##          B    3  945    1    0    0
##          C    0    2  853    0    0
##          D    0    0    3  801    0
##          E    0    0    0    4  897
##
## Overall Statistics
##
##                Accuracy : 0.9969
##                  95% CI : (0.995, 0.9983)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9961
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9979   0.9968   0.9953   0.9950   0.9989
## Specificity            0.9994   0.9990   0.9995   0.9993   0.9990
## Pos Pred Value         0.9986   0.9958   0.9977   0.9963   0.9956
## Neg Pred Value         0.9991   0.9992   0.9990   0.9990   0.9998
## Prevalence             0.2847   0.1933   0.1748   0.1642   0.1831
## Detection Rate         0.2841   0.1927   0.1739   0.1633   0.1829
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9986   0.9979   0.9974   0.9971   0.9989
```

Conclusion

The Random Forest model provided the best accuracy. I will apply RF method to the 20 observations in the testing data set.