Practical Machine Learning Week 4 Peer Review Project

JP Dunlap

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, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

The goal of this analysis is to predict the manner in which the participants did the exercise. The “classe” variable in the data training data set is the outcome variable in the analysis.

Data Preprocessing

The training data for this project are taken from:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
(https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are taken from:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv
(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 (http://groupware.les.inf.puc-rio.br/har). They have been very generous in allowing their data to be used for this assignment. For more information please see:

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human ’13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Downloading the Data from the Original Data Source

Download the training and test data sets. Create two data frames, one for the training data (trainingDF) and one for the testing data (testingDF).

```
trainingUrl <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testingUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
trainingFile <- "./data/pml-training.csv"
testingFile  <- "./data/pml-testing.csv"
if (!file.exists("./data")) {
  dir.create("./data")
}
if (!file.exists(trainingFile)) {
  download.file(trainingUrl, destfile=trainingFile)
}
if (!file.exists(testingFile)) {
  download.file(testingUrl, destfile=testingFile)
}
```

```
trainingDF <- read.csv("./data/pml-training.csv")
testingDF <- read.csv("./data/pml-testing.csv")
```

The unedited training data set contains 19622 observations, while the unedited testing data set contains 20 observations. The unedited data frames both have 160 variables, including the “classe” variable which is the outcome to predict.

Cleaning the data

Exploratory analysis has indicated that very few of the cases in the training data frame, and none of the cases in the testing data frame have all of the predictive data elements recorded. In addition, there are some variables recorded in the data set that are not actual predictors. These missing elements, and non predictors must be removed before model building.

The first step is to pull the factor variable “classe”" out of the training data frames and place it in a separate variable so it does not get lost later as we clean the data. It is returned to the cleaned data frame before model building. Then the variables that do not actually serve as predictors can be eliminated. These include things such as user identifiers and time stamps.Then variables that are predominately missing can be eliminated.

```
#Preserve the classe variable
classe <- trainingDF$classe

#Start by converting blanks to NAs
trainingDF[trainingDF == ""] <- NA
testingDF[testingDF == ""] <- NA

#Do Column-wise elimination of all variables with NA values
trainingDF <- trainingDF[, colSums(is.na(trainingDF)) == 0]
testingDF <- testingDF[, colSums(is.na(testingDF)) == 0]

#Remove variables that are named "timestamp", and those that are non numeric (which is why it wa
s necessary to preserve classe)
trainingTmp <- grepl("^X|timestamp|window", names(trainingDF))
trainingDF <- trainingDF[, !trainingTmp]
trainingData <- trainingDF[, sapply(trainingDF, is.numeric)]

#Remerge classe
trainingData$classe <- classe

#Do the same for the testing data frame
testingTmp <- grepl("^X|timestamp|window", names(testingDF))
testingDF <- testingDF[, !testingTmp]
testingData <- testingDF[, sapply(testingDF, is.numeric)]
```

Now, the cleaned training data set (“trainingData”) now has only 53 variables, one of which is “classe” and the rest of which are predictors.

Creating a Validation Data Set

This assignment actually requires three data sets. The fist is the training, the second is a true testing set, and the third is a data set of 20 observations on which the learner is actually “tested”. Even though it is not completely appropriate to call the second data set a validation data set, that name will be used here for the name of the data set against which the training model will be tested. A new validation data set will be created from the original training set before the model is run.

```
set.seed(32257)
inValidate <- createDataPartition(trainingData$classe, p=0.70, list=F)
trainData <- trainingData[inValidate, ]
validateData <- trainingData[-inValidate, ]
```

Random Forest Modeling

Random Forest Modeling was selected because it is a classification method that combines different approaches such as decision trees and bagging into one approach. It is computationally intensive, but for this data set that is not a consideration. In addition, the algorithm can perform cross-validation. This model includes k-fold validation with k = 5.

```
#setup crossvalidation
controlValidation <- trainControl(method="cv", 5)

#run model
modelRf <- train(classe ~ ., data=trainData, method="rf", trControl=controlValidation, ntree=150
)
modelRf
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10991, 10989, 10990, 10989
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9892259  0.9863691
##   27    0.9892991  0.9864631
##   52    0.9839851  0.9797416
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Model Performance

The performance of the model is ascertained by looking at a few factors. The first indicator is the confusion matrix which shows the actual performance of the model against its own test group (the validateData data frame in this case) to see how well it predicts).The accuracy and sampling error are also reviewed. These are provided in the following table.

```
predictRf <- predict(modelRf, validateData)

confusionMatrix(validateData$classe, predictRf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    7 1129    2    1    0
##          C    0    3 1014    9    0
##          D    0    0    2  961    1
##          E    0    0    4    3 1075
##
## Overall Statistics
##
##                Accuracy : 0.9946
##                  95% CI : (0.9923, 0.9963)
##     No Information Rate : 0.2856
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9931
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9958   0.9973   0.9922   0.9867   0.9991
## Specificity            1.0000   0.9979   0.9975   0.9994   0.9985
## Pos Pred Value         1.0000   0.9912   0.9883   0.9969   0.9935
## Neg Pred Value         0.9983   0.9994   0.9984   0.9974   0.9998
## Prevalence             0.2856   0.1924   0.1737   0.1655   0.1828
## Detection Rate         0.2845   0.1918   0.1723   0.1633   0.1827
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9979   0.9976   0.9949   0.9930   0.9988
```

```
precisionRF <- confusionMatrix(validateData$classe, predictRf)
```

The confusion matrix shows a high level of accuracy between the predicted categories and the actual categories in the validation data set.

The **estimated accuracy of the model is 0.9945624** and the **out of sample error is 0.0054376**.

Predicting for Test Data Set

Now, we apply the model to the original testing data set downloaded from the data source. We remove the `problem_id` column first.

```
result <- predict(modelRf, testingData[, -length(names(testingData))])
result
```

```
##  [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
```

Appendix: Comparison to Other ML Approaches

The Random Forest Algorithm is certainly more accurate than other approaches, but in some cases one might argue for simplicity, especially if interpretability is of prime concern. So as a contrast a simple classification tree approach was attempted for comparison. The results are as follows:

```
modelTree <- train(classe ~ ., data=trainData, method="rpart")
modelTree
```

```
## CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.02634523  0.5603120  0.43484311
##   0.04380022  0.4608664  0.28463034
##   0.11392534  0.3291565  0.06740691
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.02634523.
```

```
predictTree <- predict(modelTree, validateData)
confusionMatrix(validateData$classe, predictTree)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1012  182  324  126   30
##          B  196  660  228   55    0
##          C   29   36  818  143    0
##          D   45  124  510  285    0
##          E   13  250  248   36  535
##
## Overall Statistics
##
##                Accuracy : 0.5624
##                  95% CI : (0.5497, 0.5752)
##     No Information Rate : 0.3616
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4514
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7815   0.5272   0.3844  0.44186  0.94690
## Specificity            0.8558   0.8966   0.9446  0.87042  0.89718
## Pos Pred Value         0.6045   0.5795   0.7973  0.29564  0.49445
## Neg Pred Value         0.9328   0.8753   0.7304  0.92684  0.99375
## Prevalence             0.2201   0.2127   0.3616  0.10960  0.09601
## Detection Rate         0.1720   0.1121   0.1390  0.04843  0.09091
## Detection Prevalence   0.2845   0.1935   0.1743  0.16381  0.18386
## Balanced Accuracy      0.8186   0.7119   0.6645  0.65614  0.92204
```

```
precisionTree <- confusionMatrix(validateData$classe, predictTree)
```

The confusion matrix shows numerous misclassifications with a fairlow low accuracy.

The estimated accuracy of the Classification Tree model is 0.5624469 and the estimated sample error is 0.4375531. As a reminder, the estimated accuracy of the Random Forest model is 0.9945624 and the estimated sample error is 0.0054376 - *substantially better*.

```
resultTree <- predict(modelTree, testingData[, -length(names(testingData))])

ctable <- confusionMatrix(result,resultTree)
```

When the predictions are compared using the Classification Tree to the Random Forest predictions for the test data set, the results are significantly different.

The predicted results from the Random Forest Model for the test set are:

B, A, B, A, A, E, D, B, A, A, B, C, B, A, E, E, A, B, B, B

The predicted results from the Classification Tree Model for the test set are:

D, C, A, C, B, C, C, C, A, A, C, C, B, A, C, B, D, C, D, B

Finally, the confusion matrix of the Random Forest prediction versus the Classification Tree prediction is:

```
ctable$table
```

```
##           Reference
## Prediction A B C D E
##          A 3 1 2 1 0
##          B 1 2 3 2 0
##          C 0 0 1 0 0
##          D 0 0 1 0 0
##          E 0 1 2 0 0
```