Practical Machine Learning Prediction Final Assignment

Ken Dye

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 6 participants. The participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The goal is to predict the manner in which they did the exercise.

Data Processing

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

Import Data

```
trainUrl <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"
trainFile <- "./data/pml-training.csv"
testFile  <- "./data/pml-testing.csv"
if (!file.exists("./data")) {
  dir.create("./data")
}
if (!file.exists(trainFile)) {
  download.file(trainUrl, destfile=trainFile)
}
if (!file.exists(testFile)) {
  download.file(testUrl, destfile=testFile)
}
```

Reading in the two .csv files in order to create two data frames.

The training .csv has 16922 observations, 160 variables.

```
trainRaw <- read.csv("./data/pml-training.csv")
dim(trainRaw)
## [1] 19622   160
```

The testing .csv has 20 observations, 160 variables.

```
testRaw <- read.csv("./data/pml-testing.csv")
dim(testRaw)
## [1]  20 160
```

Note that the “classe” variable in the training set is the outcome we wish to to predict.

Data Cleaning

Incomplete observations and data with missing values and values/variables that have no relevance to the analysis will be removed.

```
sum(complete.cases(trainRaw))
## [1] 406
```

Remove col. with NA values.

```
trainRaw <- trainRaw[, colSums(is.na(trainRaw)) == 0]
testRaw <- testRaw[, colSums(is.na(testRaw)) == 0]
```

Remove col, that are not relevant to the analysis.

```
classe <- trainRaw$classe
trainRemove <- grepl("^X|timestamp|window", names(trainRaw))
trainRaw <- trainRaw[, !trainRemove]
trainCleaned <- trainRaw[, sapply(trainRaw, is.numeric)]
trainCleaned$classe <- classe
testRemove <- grepl("^X|timestamp|window", names(testRaw))
testRaw <- testRaw[, !testRemove]
testCleaned <- testRaw[, sapply(testRaw, is.numeric)]
```

The cleaned training data now contains 19622 observations, 53 variables. The testing data set now contains 20 observations, 53 variables. The “classe” variable was not lost during cleaning.

Data Slicing

The cleaned training set is split into a training data set (70%) and a validation data set (30%).

Reproducible Data

```
set.seed(88123)
inTrain <- createDataPartition(trainCleaned$classe, p=0.70, list=F)
trainData <- trainCleaned[inTrain, ]
testData <- trainCleaned[-inTrain, ]
```

Create Model

Using **Random Forest** algorithm to fit the activity recognition predictive model. The **5-fold cross validation** will be used for validation.

```
controlRf <- trainControl(method="cv", 5)
modelRf <- train(classe ~ ., data=trainData, method="rf",
trControl=controlRf, ntree=250)
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, 10990, 10991, 10989, 10989
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9893714  0.9865531
##   27    0.9903180  0.9877527
##   52    0.9817282  0.9768862
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Estimating the performance of the validation data set models yeilds:

```
predictRf <- predict(modelRf, testData)
confusionMatrix(testData$classe, predictRf)
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B   15 1122    2    0    0
##          C    0    4 1018    4    0
##          D    0    0   11  953    0
##          E    0    0    1    3 1078
##
## Overall Statistics
##
##                Accuracy : 0.9932
```

```
##                  95% CI : (0.9908, 0.9951)
##     No Information Rate : 0.287
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9914
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9911   0.9964   0.9864   0.9927   1.0000
## Specificity            1.0000   0.9964   0.9984   0.9978   0.9992
## Pos Pred Value         1.0000   0.9851   0.9922   0.9886   0.9963
## Neg Pred Value         0.9964   0.9992   0.9971   0.9986   1.0000
## Prevalence             0.2870   0.1913   0.1754   0.1631   0.1832
## Detection Rate         0.2845   0.1907   0.1730   0.1619   0.1832
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9956   0.9964   0.9924   0.9952   0.9996
accuracy <- postResample(predictRf, testData$classe)
accuracy
##  Accuracy     Kappa
## 0.9932031 0.9914002
oose <- 1 - as.numeric(confusionMatrix(testData$classe,
predictRf)$overall[1])
oose
## [1] 0.006796941
```

99.42% with an out-of-sample error of 0.58%.

Test Data Set Prediction

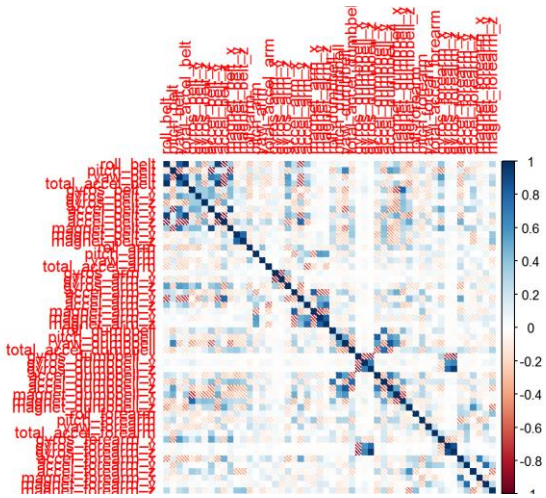
Applying the model to the test data set.

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

Figures

Correlation Matrix

```
corrPlot <- cor(trainData[, -length(names(trainData))])
corrplot(corrPlot, method="shade")
```



Decision Tree

```
treeModel <- rpart(classe ~ ., data=trainData, method="class")
prp(treeModel, nn=TRUE,shadow.col="gray",branch.lty=3, branch=.5,faclen=0,
trace=1,split.cex=1.2,split.prefix="is ", split.suffix="?")
## cex 0.4   xlim c(0, 1)   ylim c(0, 1)
```