Peer-graded Assignment: Prediction Assignment Writeup

Andualem Bekele

July 28, 2018

1. Summary

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 dumbell 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 (see the section on the Weight Lifting Exercise Dataset). ##2. Data The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

```
library(caret)

## Warning: package 'caret' was built under R version 3.4.4

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.4.2

library(rpart);
library(ggplot2);
library(randomForest)

## Warning: package 'randomForest' was built under R version 3.4.4

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

Reading and Cleaning Data

```
train<- read.csv("C:/Users/abekele/Documents/Corsera/Machine Learning/pml-
training.csv")
test<- read.csv("C:/Users/abekele/Documents/Corsera/Machine Learning/pml-
testing.csv")

dim(train)

## [1] 19622   160

dim(test)

## [1]  20 160
```

Removing missing values that contain N/A.

```
train <- train[, colSums(is.na(train)) == 0]
test <- test[, colSums(is.na(test)) == 0]
```

Listing column names of traning dataset

```
head(colnames(train))

## [1] "X"                    "user_name"            "raw_timestamp_part_1"
## [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"

classe <- train$classe
trainR <- grepl("^X|timestamp|window", names(train))
train <- train[, !trainR]
trainM <- train[, sapply(train, is.numeric)]
trainM$classe <- classe
testR <- grepl("^X|timestamp|window", names(test))
test<- test[, !testR]
testM <- test[, sapply(test, is.numeric)]


dim(train)

## [1] 19622    87

dim(test)

## [1] 20 54

set.seed(12345)
inTrain <- createDataPartition(trainM$classe, p=0.70, list=F)
train_data <- trainM[inTrain, ]
test_data <- trainM[-inTrain, ]
```

Data Prediction and Modelling

Randon Forest

Random Forest as the predictive Model.

```
setting <- trainControl(method="cv", 5)
RandomForest <- train(classe ~ ., data=train_data, method="rf",
trControl=setting, ntree=250)
RandomForest

## 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: 10990, 10989, 10990, 10989, 10990
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9904636  0.9879365
##   27    0.9900997  0.9874754
##   52    0.9841301  0.9799219
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

estimating the performance of the model, getting the accuracy and estimated out-of-sample error.

```
predict_RandomForest <- predict(RandomForest, test_data)
confusionMatrix(test_data$classe, predict_RandomForest)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    2    0    0    0
##          B   11 1122    6    0    0
##          C    0   13 1009    4    0
##          D    0    0   25  939    0
##          E    0    0    0    4 1078
##
## Overall Statistics
##
##                Accuracy : 0.989
##                  95% CI : (0.9859, 0.9915)
##     No Information Rate : 0.286
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.986
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9935   0.9868   0.9702   0.9916   1.0000
## Specificity            0.9995   0.9964   0.9965   0.9949   0.9992
## Pos Pred Value         0.9988   0.9851   0.9834   0.9741   0.9963
## Neg Pred Value         0.9974   0.9968   0.9936   0.9984   1.0000
## Prevalence             0.2860   0.1932   0.1767   0.1609   0.1832
## Detection Rate         0.2841   0.1907   0.1715   0.1596   0.1832
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9965   0.9916   0.9833   0.9932   0.9996
```

Decision Tree

Uing Decision Tree as the predictive Model.

```
model_tree <- rpart(classe ~ ., data=train_data, method="class")
prediction_tree <- predict(model_tree, test_data, type="class")
class_tree <- confusionMatrix(prediction_tree, test_data$classe)
class_tree
```

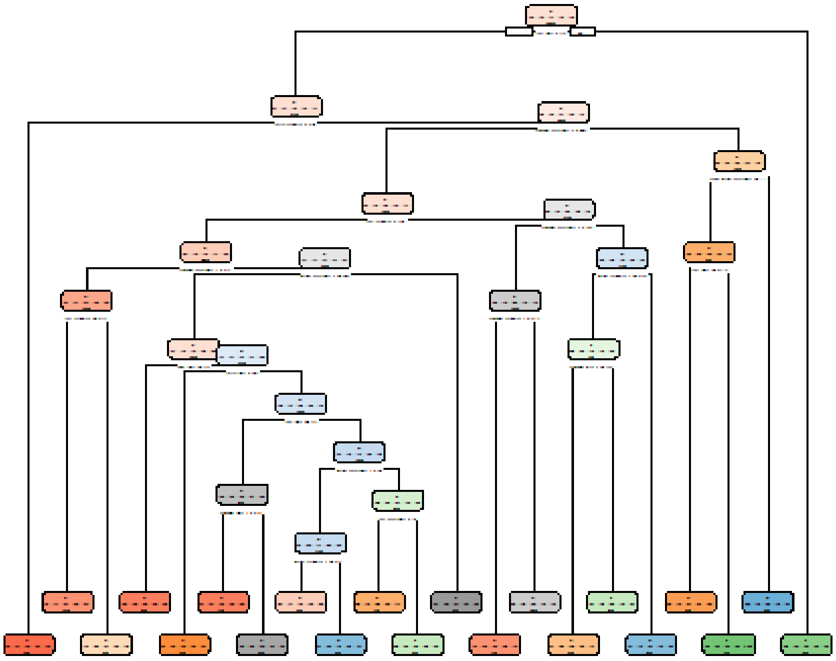
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1498  196   69  106   25
##          B   42  669   85   86   92
##          C   43  136  739  129  131
##          D   33   85   98  553   44
##          E   58   53   35   90  790
##
## Overall Statistics
##
##                Accuracy : 0.722
##                  95% CI : (0.7104, 0.7334)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6467
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity          0.8949   0.5874   0.7203  0.57365   0.7301
## Specificity          0.9060   0.9357   0.9097  0.94717   0.9509
## Pos Pred Value       0.7909   0.6869   0.6273  0.68020   0.7700
## Neg Pred Value       0.9559   0.9043   0.9390  0.91897   0.9399
## Prevalence           0.2845   0.1935   0.1743  0.16381   0.1839
## Detection Rate       0.2545   0.1137   0.1256  0.09397   0.1342
## Detection Prevalence 0.3218   0.1655   0.2002  0.13815   0.1743
## Balanced Accuracy    0.9004   0.7615   0.8150  0.76041   0.8405
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.4.4
```

```
rpart.plot(model_tree)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



4.Conclusions

The Random Forest is a much accurate predictive model than the Decision TreeIt has an accuracy of 99.9%.

In this study, the characteristics of NA values , low variance, correlation and skewness of both traning and testing datasets (train and test) are reduced. Therefore, the variables of the data sets are scaled. The training dataset is divided into training and validation parts to construct a predictive model and evaluate its accuracy. Decision Tree and Random Forest were used.The Random Forest produced a much better predictive model (more than 99%)

than the Decision Tree. This analysis is reproducible and the environment and information is as follows:

```r
sessionInfo()
```

```
## R version 3.4.1 (2017-06-30)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 7 x64 (build 7601) Service Pack 1
## 
## Matrix products: default
## 
## locale:
## [1] LC_COLLATE=English_United States.1252 
## [2] LC_CTYPE=English_United States.1252   
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C                          
## [5] LC_TIME=English_United States.1252    
## 
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base     
## 
## other attached packages:
## [1] rpart.plot_2.2.0    randomForest_4.6-14 rpart_4.1-11       
## [4] caret_6.0-80        ggplot2_2.2.1       lattice_0.20-35    
## 
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.13       lubridate_1.7.4    tidyr_0.8.1       
##  [4] class_7.3-14       assertthat_0.2.0   rprojroot_1.2     
##  [7] digest_0.6.12      ipred_0.9-6        psych_1.8.4       
## [10] foreach_1.4.4      R6_2.2.2           plyr_1.8.4        
## [13] backports_1.1.1    magic_1.5-8        stats4_3.4.1      
## [16] e1071_1.6-8        evaluate_0.10.1    rlang_0.2.1       
## [19] lazyeval_0.2.0     kernlab_0.9-26     Matrix_1.2-10     
## [22] rmarkdown_1.6      splines_3.4.1      CVST_0.2-2        
## [25] ddalpha_1.3.4      gower_0.1.2        stringr_1.2.0     
## [28] foreign_0.8-69     munsell_0.4.3      broom_0.4.4       
## [31] compiler_3.4.1     pkgconfig_2.0.1    mnormt_1.5-5      
## [34] dimRed_0.1.0       htmltools_0.3.6    tidyselect_0.2.4  
## [37] nnet_7.3-12        tibble_1.3.4       prodlim_2018.04.18
## [40] DRR_0.0.3          codetools_0.2-15   RcppRoll_0.3.0    
## [43] withr_2.1.2        dplyr_0.7.4        MASS_7.3-47       
## [46] recipes_0.1.3      ModelMetrics_1.1.0 grid_3.4.1        
## [49] nlme_3.1-131       gtable_0.2.0       magrittr_1.5      
## [52] scales_0.5.0       stringi_1.1.5      reshape2_1.4.2    
## [55] bindrcpp_0.2       timeDate_3043.102  robustbase_0.93-1 
## [58] geometry_0.3-6     pls_2.6-0          lava_1.6.1        
## [61] iterators_1.0.9    tools_3.4.1        glue_1.2.0        
## [64] DEoptimR_1.0-8     purrr_0.2.5        sfsmisc_1.1-2     
## [67] abind_1.4-5        parallel_3.4.1     survival_2.41-3   
## [70] yaml_2.1.14        colorspace_1.3-2   knitr_1.17
## [73] bindr_0.1
```