Practical Machine Learning Project

Dan Allen

3/24/2021

Executive Summary

The following analysis will look at Decision Tree, Random Forest and Generalized Boosted Model to determine which is the most accurate predictor. The analysis determined the Random Forest at 99.5% was the most accurate. I then used Random Forest on the given test data to produce our predicted results.

Acknowledgments

The data for this project came from http://groupware.les.inf.puc-rio.br/har. 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. For this analysis the data has been split into a training set and testing set. The test data url: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Summary

To read more about the data: http://groupware.les.inf.puc-rio.br/har#wle_paper_section#ixzz6pDRBbWwY. Using data obtained by 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.

Here I have used the 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 (see the section on the Weight Lifting Exercise Dataset).

Data Processing

Load Libaries Download Data

```
set.seed(7)
library(lattice)
library(ggplot2)
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.4
```

```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'

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

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 4.0.4

## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.0.4

## corrplot 0.84 loaded
```

```
training <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", header = TR
testing <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", header = TRUE
```

Clean the Data

These files contain many columns that are unnecessary for our analysis. We therefore are removing the first 7 columns from both the 'training' and 'testing' data frames. Many columns have NA and empty values "", So we kept only data with actual data.

```
train <- training[, -(1:7)]
test <- testing[, -(1:7)]

keep <- colSums(is.na(train)/dim(train)) == 0
train <- train[, keep]
test <- test[, keep]

keep <- colSums(train == "") == 0
train <- train[, keep]
test <- test[, keep]
```

Split the Data

Now that the data is clean we will split the train data into two groups: trainA and trainTest. TrainA will be used to create the models which will then be tested on trainTest. Once we know the most accurate model, will then use it for the real test

```
part <- createDataPartition( y = train$classe,
                             p = 0.7,
                             list = FALSE)
trainA <- train[part,]
trainTest <- train[-part,]
```

Train

We will now run a few training models and pick the most accurate

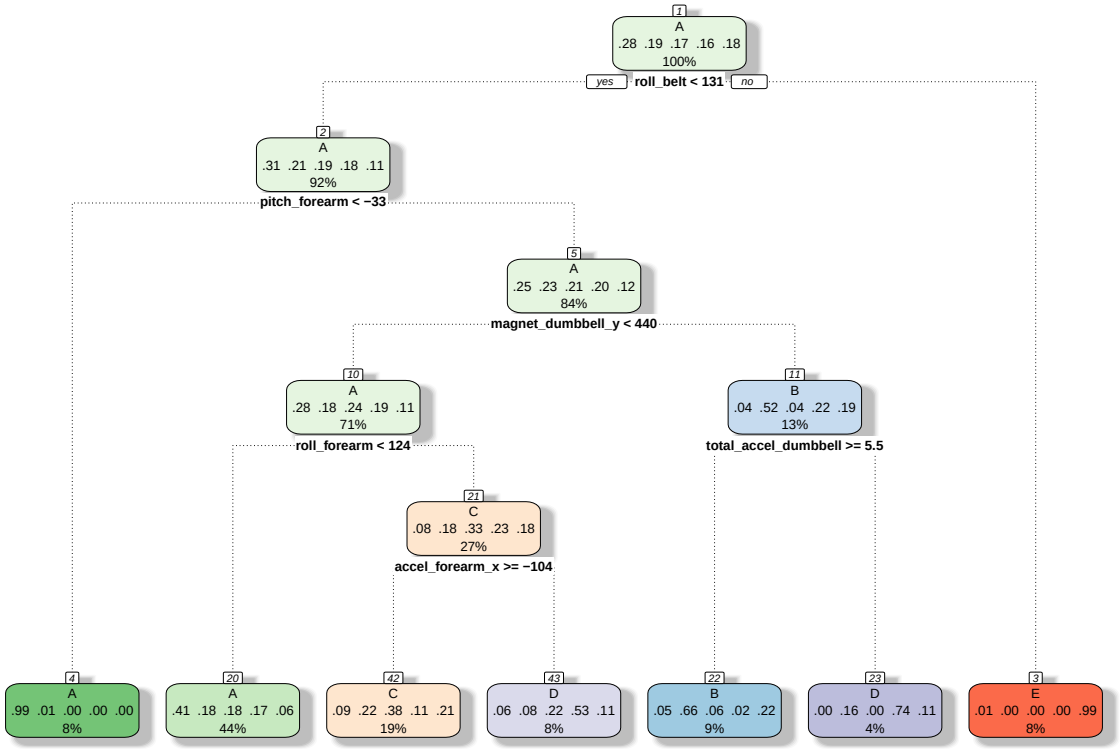
Control

Set up control for training to use 3-fold cross validation.

```
control <- trainControl(method="cv", number=3, verboseIter=F)
```

Decision Tree (DT)

```
model_dt <- train(classe~., data=trainA, method="rpart", trControl = control, tuneLength = 5)
fancyRpartPlot(model_dt$finalModel)
```



Rattle 2021−Mar−24 16:49:02 Home

```
pred_dt <- predict(model_dt, trainTest)
cm_dt <- confusionMatrix(pred_dt, factor(trainTest$classe))
cm_dt
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1543  503  479  442  147
##          B   15  327   30   14  133
##          C   88  229  397  111  257
##          D   22   80  120  397   66
##          E    6    0    0    0  479
##
## Overall Statistics
##
##                Accuracy : 0.5341
##                  95% CI : (0.5212, 0.5469)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3918
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9217  0.28709  0.38694  0.41183  0.44270
## Specificity            0.6269  0.95954  0.85902  0.94148  0.99875
## Pos Pred Value         0.4955  0.63006  0.36691  0.57956  0.98763
## Neg Pred Value         0.9527  0.84868  0.86904  0.89096  0.88833
## Prevalence             0.2845  0.19354  0.17434  0.16381  0.18386
## Detection Rate         0.2622  0.05556  0.06746  0.06746  0.08139
## Detection Prevalence   0.5291  0.08819  0.18386  0.11640  0.08241
## Balanced Accuracy      0.7743  0.62332  0.62298  0.67665  0.72072
```

```
acc_dt <- cm_dt$overall[1]*100
```

Random Forest (RF)

```
model_rf <- train(classe~., data=trainA, method="rf", trControl = control, tuneLength = 5)
pred_rf <- predict(model_rf, trainTest)
cm_rf <- confusionMatrix(pred_rf, factor(trainTest$classe))
cm_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    7    0    0    0
##          B    0 1129    6    0    0
##          C    0    3 1020   11    0
##          D    0    0    0  952    2
##          E    0    0    0    1 1080
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9927, 0.9966)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9912   0.9942   0.9876   0.9982
## Specificity            0.9983   0.9987   0.9971   0.9996   0.9998
## Pos Pred Value         0.9958   0.9947   0.9865   0.9979   0.9991
## Neg Pred Value         1.0000   0.9979   0.9988   0.9976   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1918   0.1733   0.1618   0.1835
```

```
## Detection Prevalence   0.2856   0.1929   0.1757   0.1621   0.1837
## Balanced Accuracy      0.9992   0.9950   0.9956   0.9936   0.9990
```

```
acc_rf <- cm_rf$overall[1]*100
```

Generalized Boosted Model (GBM)

```
model_gbm <- train(classe~., data=trainA, method="gbm", trControl = control, tuneLength = 5, verbose = F
model_gbm
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9159, 9158, 9157
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7524935  0.6861990
##   1                  100      0.8204855  0.7727583
##   1                  150      0.8546998  0.8161363
##   1                  200      0.8746459  0.8412909
##   1                  250      0.8889865  0.8594985
##   2                   50      0.8526623  0.8133137
##   2                  100      0.9045653  0.8792111
##   2                  150      0.9304804  0.9120186
##   2                  200      0.9466409  0.9324809
##   2                  250      0.9556677  0.9439083
##   3                   50      0.8948106  0.8668660
##   3                  100      0.9400164  0.9240920
##   3                  150      0.9600354  0.9494326
##   3                  200      0.9689164  0.9606789
##   3                  250      0.9758318  0.9694286
##   4                   50      0.9227645  0.9022213
##   4                  100      0.9577063  0.9464785
##   4                  150      0.9721192  0.9647294
##   4                  200      0.9788891  0.9732973
##   4                  250      0.9822377  0.9775339
##   5                   50      0.9365227  0.9196499
##   5                  100      0.9672419  0.9585577
##   5                  150      0.9785254  0.9728366
##   5                  200      0.9834025  0.9790061
##   5                  250      0.9862415  0.9825988
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
```

```
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 250, interaction.depth =
##  5, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
pred_gbm <- predict(model_gbm, trainTest)
cm_gbm <- confusionMatrix(pred_gbm, factor(trainTest$classe))
cm_gbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1668   12    0    0    1
##          B    4 1121   14    0    0
##          C    1    6 1006   13    0
##          D    1    0    6  945    3
##          E    0    0    0    6 1078
##
## Overall Statistics
##
##                Accuracy : 0.9886
##                  95% CI : (0.9856, 0.9912)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9856
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9842   0.9805   0.9803   0.9963
## Specificity            0.9969   0.9962   0.9959   0.9980   0.9988
## Pos Pred Value         0.9923   0.9842   0.9805   0.9895   0.9945
## Neg Pred Value         0.9986   0.9962   0.9959   0.9961   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2834   0.1905   0.1709   0.1606   0.1832
## Detection Prevalence   0.2856   0.1935   0.1743   0.1623   0.1842
## Balanced Accuracy      0.9967   0.9902   0.9882   0.9891   0.9975
```

```
acc_gbm <- cm_gbm$overall[1]*100
```

Aanalysis

The Most accurate test was the accuracy rate of the Random Forest Model.

Decision Tree: 53.41 %

Random Forest: 99.49 %

Generalized Boosted Model: 98.86 %

So we will test the Random Forest against the test data.

```r
pred_Test <- predict(model_rf, test)
pred_Test
```

```
##  [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
```