Prediction Assignment

Emma Engler

4/10/2020

Executive Summary

With the usage of devices such as Jawbone Up, Nike, FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensivly. 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. The goal of this report is to use data from accelerometers on the belt, forearm, arm and dumbbell of 6 participants to predict the manner in which they did their exercise. This is the "classe" variable in the training set. They were asked to perform barbell lifts correclty and incorrectly in 5 different ways. The following report evaluates 3 possible models in which to use prediction.

```
## Loading Necessary Packages
library(knitr)
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

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

```
## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:rattle':
##
##     importance

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

```
library(RColorBrewer)
library(RGtk2)
```

```
## Error in dyn.load(file, DLLpath = DLLpath, ...) :
##   unable to load shared object '/Library/Frameworks/R.framework/Versions/3.6/Resources/library/RGtk2/
##   dlopen(/Library/Frameworks/R.framework/Versions/3.6/Resources/library/RGtk2/libs/RGtk2.so, 6): Lib
##   Referenced from: /Library/Frameworks/R.framework/Versions/3.6/Resources/library/RGtk2/libs/RGtk2.s
##   Reason: image not found
```

```
## Warning: Failed to load RGtk2 dynamic library, attempting to install it.
```

```
## Please install GTK+ from http://r.research.att.com/libs/GTK_2.24.17-X11.pkg
```

```
## If the package still does not load, please ensure that GTK+ is installed and that it is on your PATH
```

```
## IN ANY CASE, RESTART R BEFORE TRYING TO LOAD THE PACKAGE AGAIN
```

```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

Loading Data

```
### Setting the URL for download
url_train <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url_test <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
### Downloading the datasets
TrainData <-read.csv(url(url_train))
TestData <-read.csv(url(url_test))
### Checking dimensions
dim(TrainData)
```

```
## [1] 19622   160
```

```
dim(TestData)
```

```
## [1]  20 160
```

Cleaning Data

Both the training and test set contain 160 variables. Next, through a cleaning process, NAs will be repoved, as well as Near Zero variance (NZV) variables and nonnumerical variables.

```
### Removing Near Zerio Variance
nzv <-nearZeroVar(TrainData)
data_train <-TrainData[,-nzv]
data_test <-TestData[,-nzv]
dim(data_train)
```

```
## [1] 19622   100
```

```
dim(data_test)
```

```
## [1]  20 100
```

```
### Removing NAs
na_val <-sapply(data_train, function(x) mean(is.na(x))) >0.95
data_train <-data_train[,na_val==FALSE]
data_test <-data_test[,na_val==FALSE]
dim(data_train)
```

```
## [1] 19622    59
```

```
dim(data_test)
```

```
## [1] 20 59
```

```
### Removing ID variables
data_train <-data_train[, 8:59]
data_test <-data_test[, 8:59]
dim(data_train)
```

```
## [1] 19622    52
```

```
dim(data_test)
```

```
## [1] 20 52
```

After the cleaning process, the number of variables that will be used in analysis is reduced to 52.

Data Partitioning

The data_train set will now be partitioned into "training" (60%) and "testing"(40%). The "testing" set will also serve as validation.

```
inTrain <-createDataPartition(data_train$classe, p=0.6, list=FALSE)
training <-data_train[inTrain,]
testing <-data_train[-inTrain,]
dim(training)
```

```
## [1] 11776    52
```

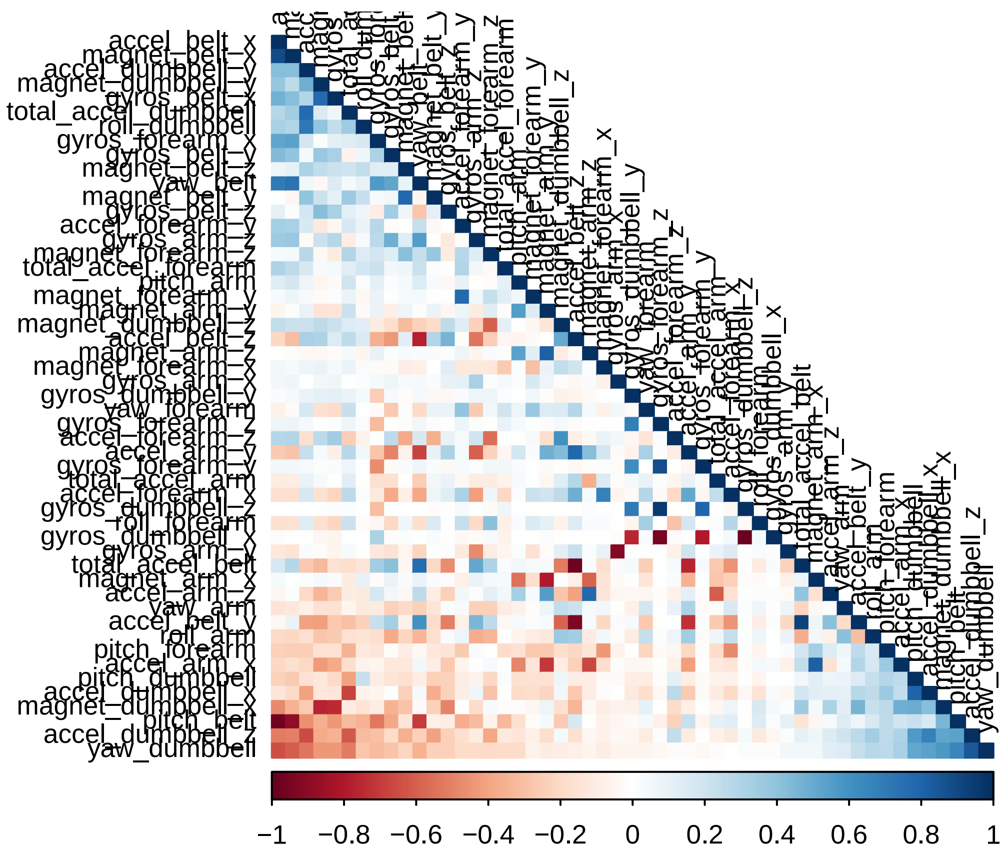
```r
dim(testing)
```

```
## [1] 7846   52
```

Correlation Analysis

Before moving to the modeling process, correlation among variables will be analyzed.

```r
corMatrix <-cor(training[, -52])
corrplot(corMatrix,order="FPC", method="color",type="lower",
         tl.cex=0.8, tl.col=rgb(0,0,0))
```



The dark colors in the graph designate highly correlated variables. As demonstrated, there are few and no further steps will be taken in light of this.

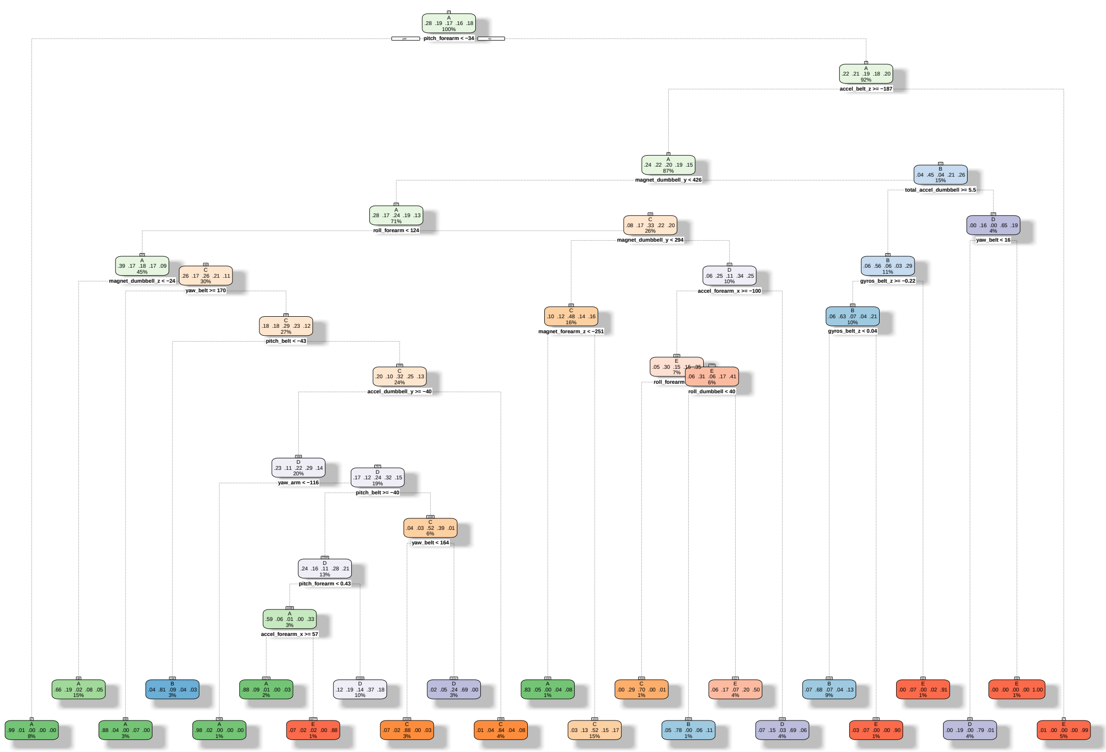
Prediction Model Building

Here, Random Forests, Decision Tree and Generalized Boosted Model methods will be used. This will allow for modeling the regression (in the train dataset) and the best model will be used later for prediction. A Confusion Matrix is used to visualize accuracy of the models.

Decision Tree

```
### fitting and plotting the model
set.seed(12345)
dt_model <-rpart(classe~., data=training, method="class")
fancyRpartPlot(dt_model)
```

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



Rattle 2020−Apr−13 14:45:09 emmaengler

```
### predicton
dt_predict <-predict(dt_model, testing, type="class")
confusionMatrix(dt_predict, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##          A 1980  295   24  110   75
##          B   74  720   79   33  100
##          C   45  210 1031  204  229
##          D  111  214  193  865  178
##          E   22   79   41   74  860
##
```

```
## Overall Statistics
##
##                Accuracy : 0.6954
##                  95% CI : (0.6851, 0.7056)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.614
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8871  0.47431   0.7537   0.6726   0.5964
## Specificity            0.9102  0.95480   0.8938   0.8939   0.9663
## Pos Pred Value         0.7971  0.71571   0.5998   0.5541   0.7993
## Neg Pred Value         0.9530  0.88333   0.9450   0.9330   0.9140
## Prevalence             0.2845  0.19347   0.1744   0.1639   0.1838
## Detection Rate         0.2524  0.09177   0.1314   0.1102   0.1096
## Detection Prevalence   0.3166  0.12822   0.2191   0.1990   0.1371
## Balanced Accuracy      0.8987  0.71456   0.8237   0.7833   0.7813
```

The Decision Tree Model produced an accuracy level of ~70%. This is below a satisfactory level and therefore will not be utilized in prediction.

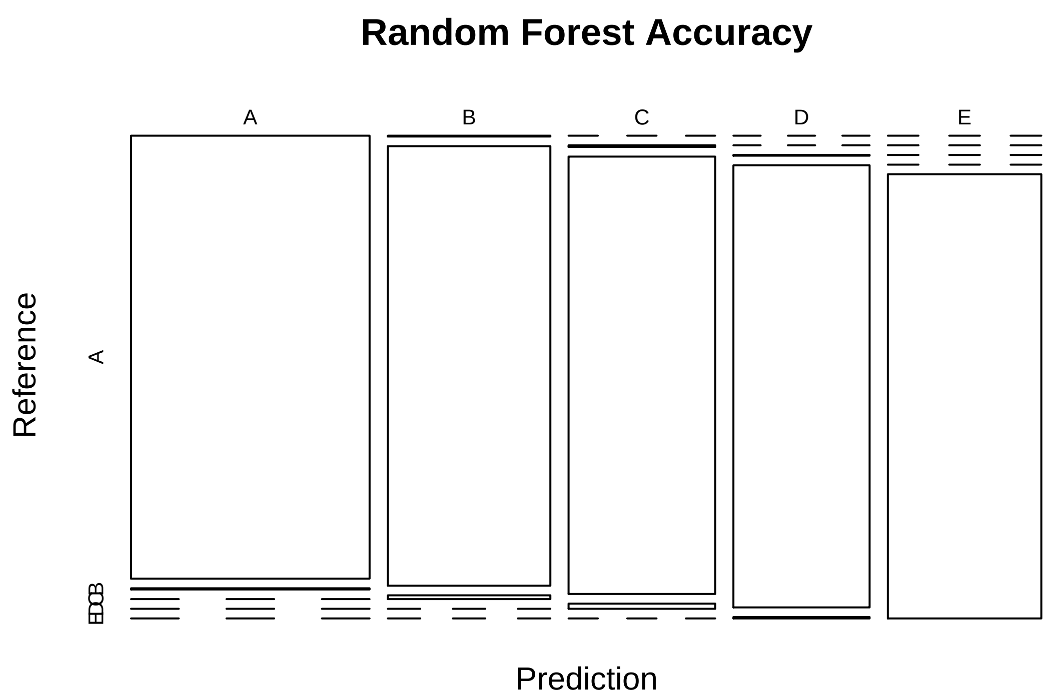
Random Forest Model

```r
### Fitting the Model
set.seed(12345)
rf_model <-randomForest(classe~., data=training, ntree=1000)
### Prediction
rf_predict <-predict(rf_model, testing, type="class")
rf_cm <-confusionMatrix(rf_predict, testing$classe)
rf_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2229    6    0    0    0
##          B    3 1507   13    0    0
##          C    0    5 1353   16    0
##          D    0    0    2 1270    4
##          E    0    0    0    0 1438
##
## Overall Statistics
##
##                Accuracy : 0.9938
##                  95% CI : (0.9918, 0.9954)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                    Kappa : 0.9921
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9928   0.9890   0.9876   0.9972
## Specificity            0.9989   0.9975   0.9968   0.9991   1.0000
## Pos Pred Value         0.9973   0.9895   0.9847   0.9953   1.0000
## Neg Pred Value         0.9995   0.9983   0.9977   0.9976   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2841   0.1921   0.1724   0.1619   0.1833
## Detection Prevalence   0.2849   0.1941   0.1751   0.1626   0.1833
## Balanced Accuracy      0.9988   0.9951   0.9929   0.9933   0.9986
```

```r
###plot
plot(rf_cm$table, col=rf_cm$byClass, main="Random Forest Accuracy")
```



It can be seen from the model and plot that the Random Forest Model has quite a satisfactory accurcacy level at about 99%.

Gradient Boosting model

```
### Fitting the Model
set.seed(12345)
library(gbm)
gbm_control <-trainControl(method="repeatedcv",number=5,repeats=1)
gbm_model <-train(classe~., data=training, method="gbm",
                  trControl=gbm_control, verbose=FALSE)
gbm_model$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 51 predictors of which 51 had non-zero influence.
```

```
### prediction
gbm_predict <-predict(gbm_model, testing)
gbm_cm <-confusionMatrix(gbm_predict, testing$classe)
gbm_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2199   46    0    0    1
##          B   24 1423   44    5   11
##          C    8   41 1306   44   22
##          D    1    3   16 1227   15
##          E    0    5    2   10 1393
##
## Overall Statistics
##
##                Accuracy : 0.962
##                  95% CI : (0.9576, 0.9661)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9519
##
##  Mcnemar's Test P-Value : 2.156e-07
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9852   0.9374   0.9547   0.9541   0.9660
## Specificity            0.9916   0.9867   0.9822   0.9947   0.9973
## Pos Pred Value         0.9791   0.9443   0.9191   0.9723   0.9879
## Neg Pred Value         0.9941   0.9850   0.9904   0.9910   0.9924
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2803   0.1814   0.1665   0.1564   0.1775
## Detection Prevalence   0.2863   0.1921   0.1811   0.1608   0.1797
## Balanced Accuracy      0.9884   0.9621   0.9685   0.9744   0.9817
```

Both the Random Forest and Gradient Boosing Models produced a satisfactory level of accuracy. Therefore, they will be compared to see which is more accurate

```
### Random forest accuracy
rf_cm$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9937548      0.9921000      0.9917518      0.9953763      0.2844762
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
### Gradient Boosting accuracy
gbm_cm$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.620189e-01   9.519491e-01   9.575504e-01   9.661398e-01   2.844762e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00   2.156200e-07
```

Conclusion

From the report, conclusions suggest that of the 3 models evaluated Random Forest is the most accurate (about 99%). Therefore predictions on the dataset model will be done using the Random Forest Model on the testing data.

Predictions

```
prediction_test <-predict(rf_model, TestData)
prediction_test
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```