Activity Prediction

Uma Srinivas Majji

10/25/2020

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 dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

Loading Packages

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

Data Preparation

```
# set the url for the download
urlTrain <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
urlValid <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# download the datasets
training <- read.csv(url(urlTrain))
validation <- read.csv(url(urlValid))
training$classe <- as.factor(training$classe)

# create a partition with the training dataset
inTrain <- createDataPartition(y=training$classe,p=0.7,list=FALSE)
trainSet <- training[inTrain,]
testSet <- training[-inTrain,]
dim(trainSet)
```

```
## [1] 13737   160
```

```
dim(testSet)
```

```
## [1] 5885  160
```

```
# remove variables with near zero variance
NZV <- nearZeroVar(trainSet)
trainSet <- trainSet[,-NZV]
testSet <- testSet[,-NZV]

# remove variables that are mostly NA
allNA <- sapply(trainSet,function(x)mean(is.na(x)))>0.95
trainSet <- trainSet[,allNA==FALSE]
testSet <- testSet[,allNA==FALSE]

# remove identification only variables
trainSet <- trainSet[,-(1:5)]
testSet <- testSet[,-(1:5)]
dim(trainSet)
```

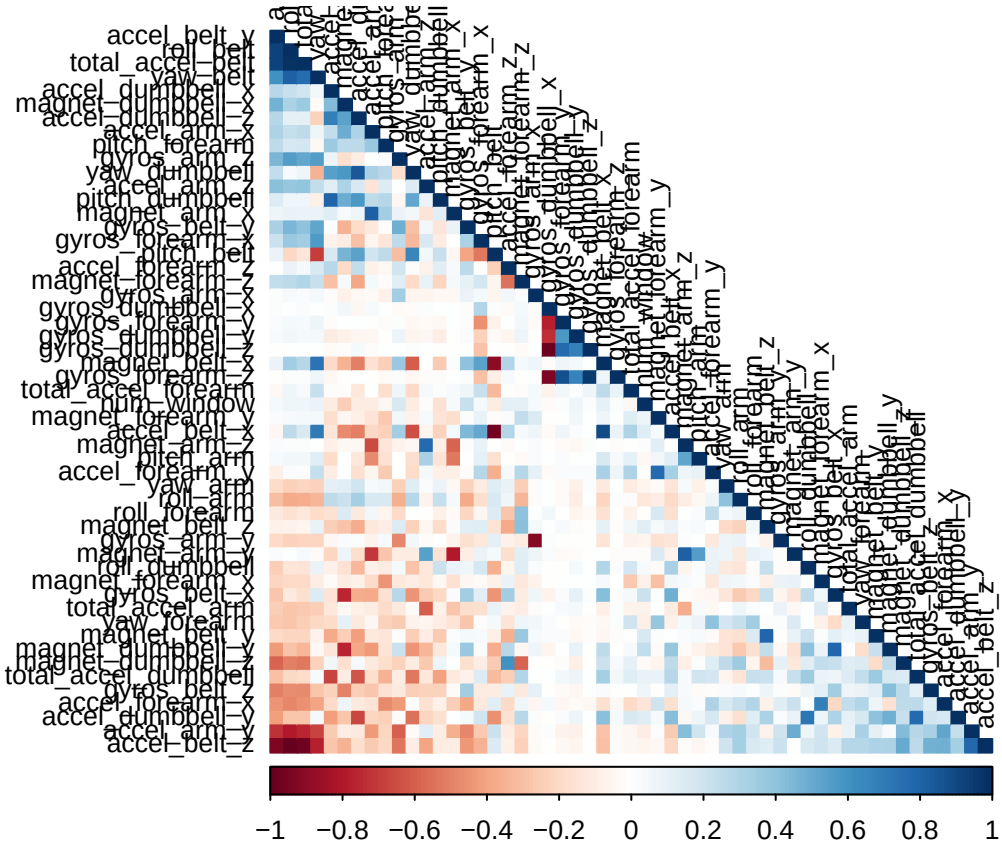
```
## [1] 13737    54
```

```
dim(testSet)
```

```
## [1] 5885   54
```

Correlation Analysis

```
#library(corrplot)
corMatrix <- cor(trainSet[,-54])
corrplot(corMatrix,order="FPC",method="color",type="lower",tl.cex=0.8,tl.col=rgb(0,0,0))
```



Prediction Model building

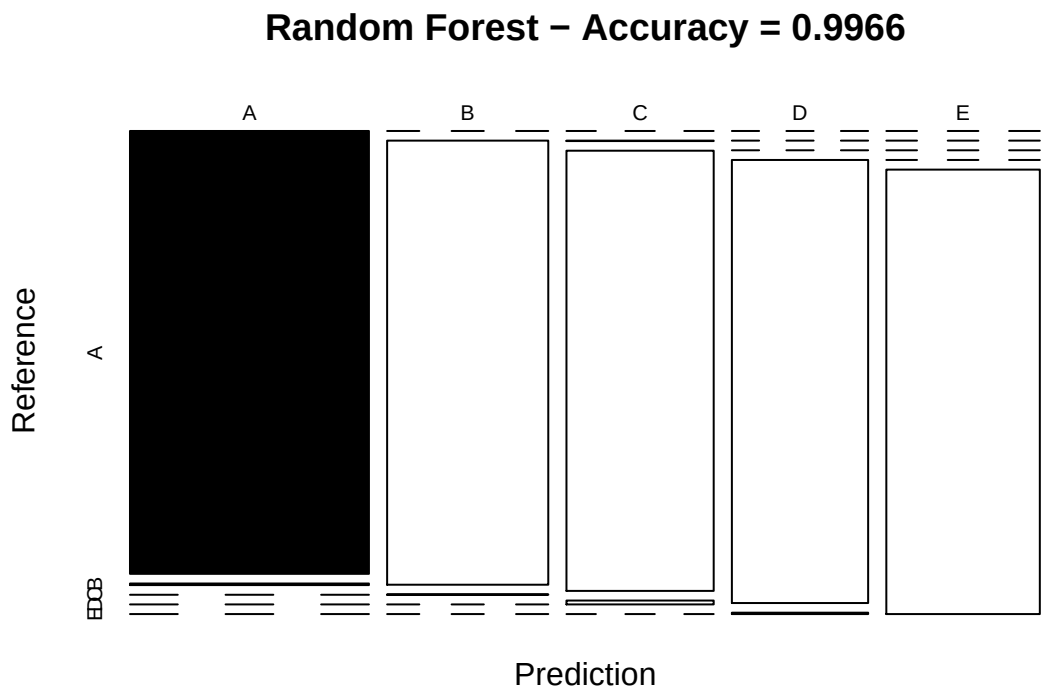
Method - Random Forest

```
# model fit
set.seed(12345)
controlRF <- trainControl(method="cv",number=3,verboseIter=FALSE)
modFitRandForest <- train(classe~.,data=trainSet,method="rf",trControl=controlRF)
modFitRandForest$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.22%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    7 2647    4    0    0 0.0041384500
## C    0    7 2389    0    0 0.0029215359
## D    0    0    4 2247    1 0.0022202487
```

```
## E    0    1    0    4 2520 0.0019801980
```

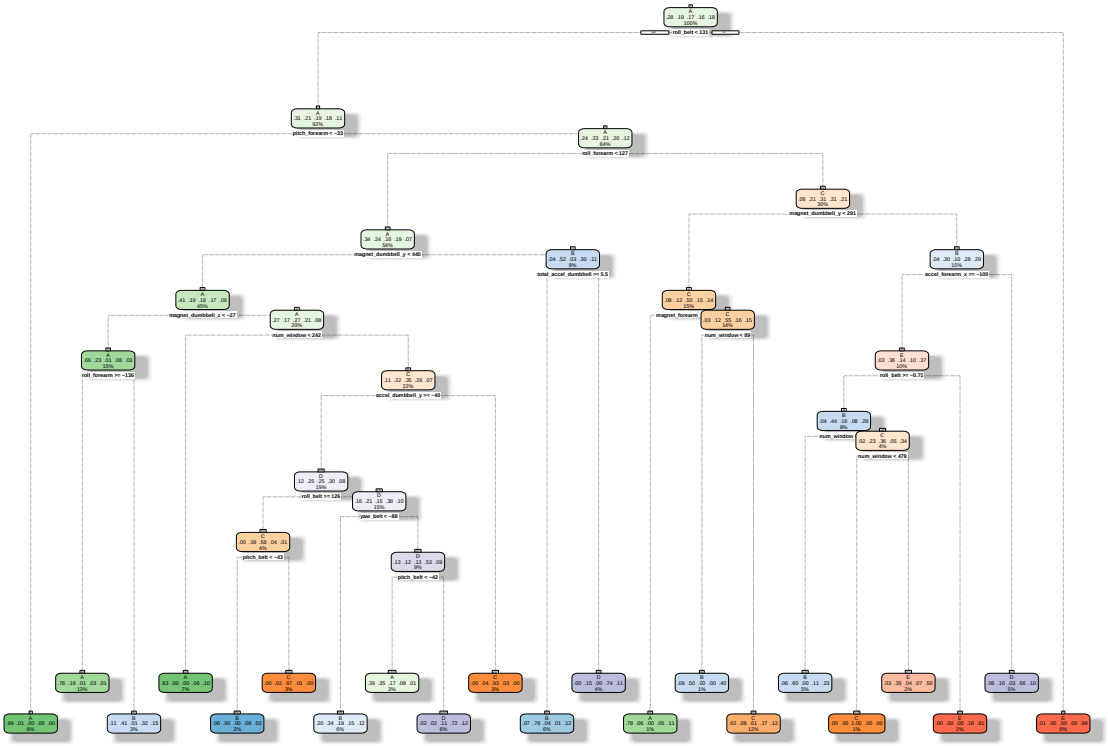
```
# prediction on test set
predRandForest <- predict(modFitRandForest,testSet)
confMatRandForest <- confusionMatrix(predRandForest,testSet$classe)

# plot matrix results
plot(confMatRandForest$table,col=confMatRandForest$byClass,
     main=paste("Random Forest - Accuracy =",round(confMatRandForest$overall['Accuracy'],4)))
```



Method - Decision Trees

```
# model fit
set.seed(12345)
modFitDecTree <- rpart(classe~.,data=trainSet,method="class")
fancyRpartPlot(modFitDecTree)
```

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



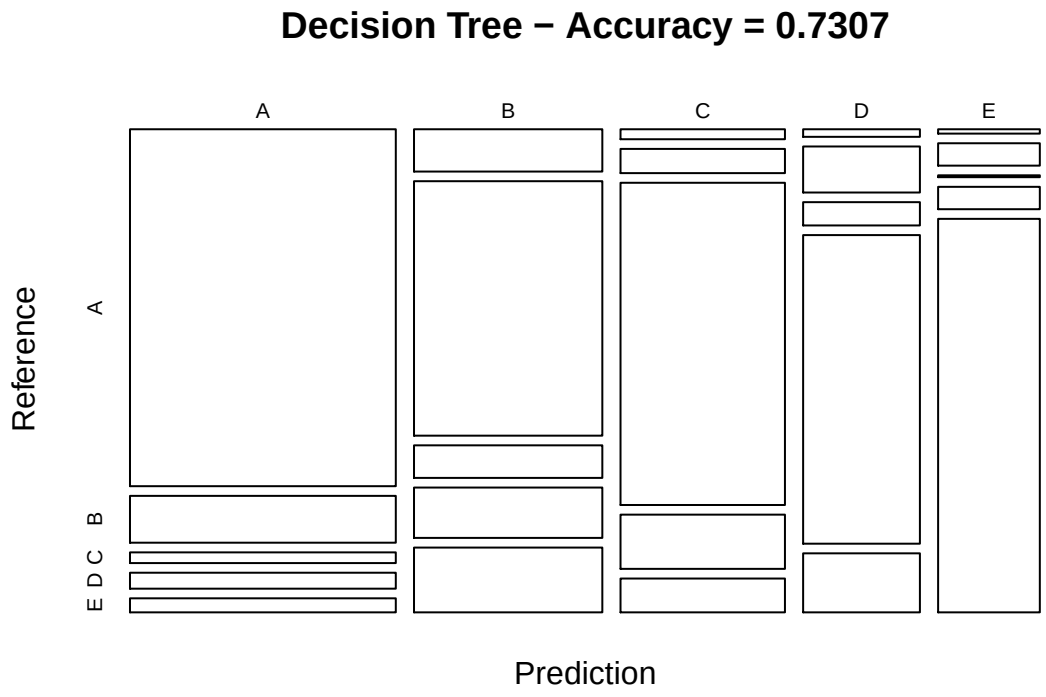
Rattle 2020−Oct−26 23:50:44 umasr

```
# prediction on test data set
predDecTree <- predict(modFitDecTree,newdata=testSet,type="class")
confMatDecTree <- confusionMatrix(predDecTree,testSet$classe)
confMatDecTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1501  197   45   67   59
##          B  126  758   97  150  193
##          C   26   63  838  141   88
##          D   14   85   43  570  109
##          E    7   36    3   36  633
##
## Overall Statistics
##
##                Accuracy : 0.7307
##                  95% CI : (0.7191, 0.742)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6576
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8967   0.6655   0.8168  0.59129   0.5850
## Specificity            0.9126   0.8807   0.9346  0.94899   0.9829
## Pos Pred Value         0.8031   0.5725   0.7249  0.69428   0.8853
## Neg Pred Value         0.9569   0.9165   0.9602  0.92220   0.9132
## Prevalence             0.2845   0.1935   0.1743  0.16381   0.1839
## Detection Rate         0.2551   0.1288   0.1424  0.09686   0.1076
## Detection Prevalence   0.3176   0.2250   0.1964  0.13951   0.1215
## Balanced Accuracy      0.9046   0.7731   0.8757  0.77014   0.7840
```

```r
# plot matrix results
plot(confMatDecTree$table,col=confMatDecTree$byClass,
     main = paste("Decision Tree - Accuracy =",round(confMatDecTree$overall['Accuracy'], 4)))
```



Method - Generalized Boosted Model

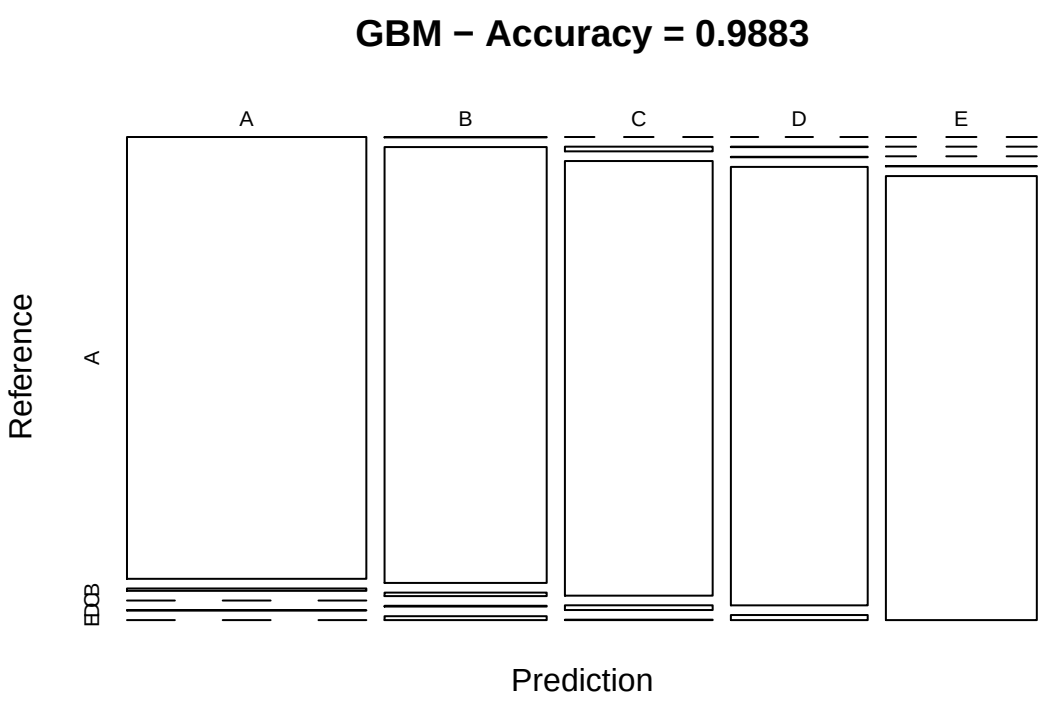
```r
# model fit
set.seed(12345)
controlGBM <- trainControl(method="repeatedcv",number=5,repeats=1)
modFitGBM <- train(classe~.,data=trainSet,method="gbm",trControl=controlGBM,verbose=F)
modFitGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

```
# prediction on test data set
predGBM <- predict(modFitGBM,newdata=testSet)
confMatGBM <- confusionMatrix(predGBM,testSet$classe)
confMatGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    9    0    1    0
##          B    1 1118    9    2   10
##          C    0   11 1016   11    1
##          D    0    1    1  949   11
##          E    0    0    0    1 1060
##
## Overall Statistics
##
##                Accuracy : 0.9883
##                  95% CI : (0.9852, 0.9909)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9852
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9816   0.9903   0.9844   0.9797
## Specificity            0.9976   0.9954   0.9953   0.9974   0.9998
## Pos Pred Value         0.9941   0.9807   0.9779   0.9865   0.9991
## Neg Pred Value         0.9998   0.9956   0.9979   0.9970   0.9954
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1900   0.1726   0.1613   0.1801
## Detection Prevalence   0.2860   0.1937   0.1766   0.1635   0.1803
## Balanced Accuracy      0.9985   0.9885   0.9928   0.9909   0.9897
```

```
# plot matrix results
plot(confMatGBM$table,col=confMatGBM$byClass,
     main=paste("GBM - Accuracy =",round(confMatGBM$overall['Accuracy'],4)))
```



Applying best fit model

```
# Applying the Selected Model to the Validation Data
predValidation <- predict(modFitRandForest,validation)
predValidation
```

```
##  [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
```