Machine Learning - Prediction Assignment

Trina Dey

27/12/2020

Overview

There are several types of fitness devices like Jawbone Up, Nike FuelBand, Fitbit etc. which collect large amount of movement data using accelerometers. We are using data collected by http://groupware.les.inf.puc-rio.br/har#ixzz3xsbS5bVX on Weight Lifting Exercise Dataset to predict the manner in which they did the exercise. We are going to build a model and cross validate our model against the test data and will also predict use cases for 20 different test scenarios.

Data Preprocessing

The training data for this project is available here:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data for this project is available here:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Please note that for simplicity, we have hidden the package loading in the report. However all the required packages have been loaded like knitr, caret, rpart, rattle, randomForest and the seed is set for reproducible analysis.

We will download the data and partition it in 70-30% for training set and test set for cross validation purpose.

```
# download the datasets
training <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"))
predictionTesting  <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

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

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

```
dim(testSet)
```

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

Next we will clean up the data and remove NAs and Nearly Zero Variance variables

```r
# remove variables with Nearly Zero Variance
NZV <- nearZeroVar(trainingSet)
trainingSet <- trainingSet[, -NZV]
testSet  <- testSet[, -NZV]

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

# remove identification only variables (columns 1 to 5)
trainingSet <- trainingSet[, -(1:5)]
testSet  <- testSet[, -(1:5)]

dim(trainingSet)
```

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

```r
dim(testSet)
```

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

Prediction Model Analysis

We will use three prediction model techniques and will plot the confusion matrix at the end of each to decide which method fits best. These methods will be - Random Forests, Decision Tress and Generalized Boosted Model.

a. Random Forest

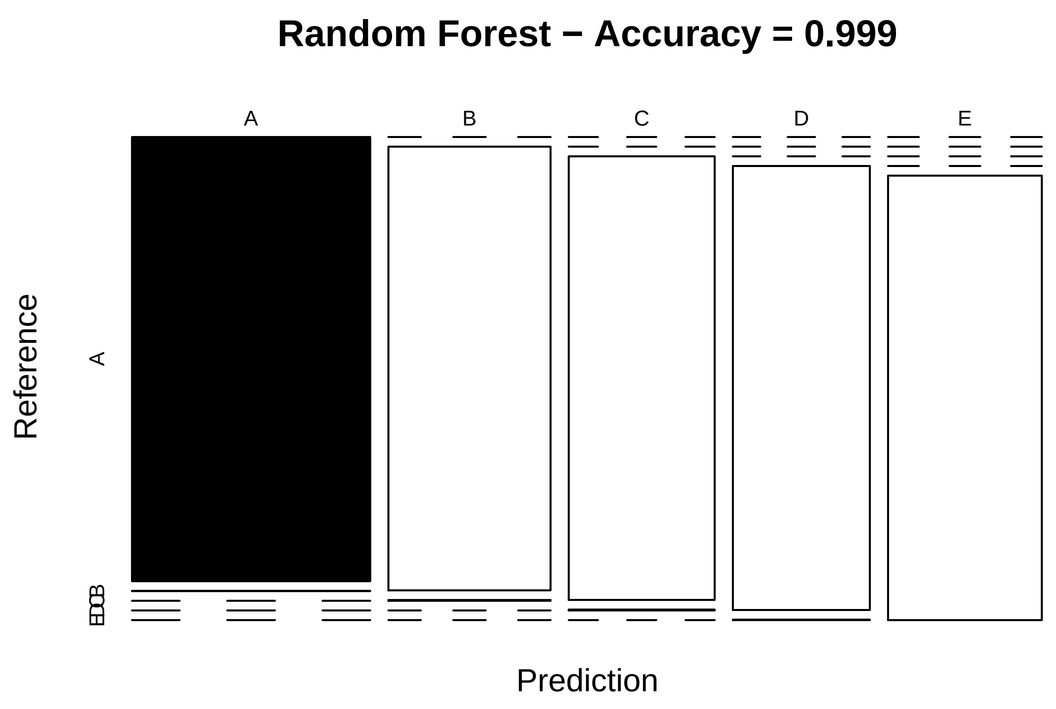
```r
set.seed(12345)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRandForest <- train(classe ~ ., data=trainingSet, method="rf",trControl=controlRF)
modFitRandForest$finalModel
```

```
##
## 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: 0.23%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    2    0    0    0 0.0005120328
## B    6 2647    4    1    0 0.0041384500
## C    0    5 2391    0    0 0.0020868114
## D    0    0    9 2243    0 0.0039964476
## E    0    0    0    5 2520 0.0019801980
```

```r
# prediction on Test dataset
predictRandForest <- predict(modFitRandForest, newdata=testSet)
confMatRandForest <- confusionMatrix(predictRandForest, as.factor(testSet$classe))
confMatRandForest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    1    0    0    0
##          B    0 1138    2    0    0
##          C    0    0 1024    2    0
##          D    0    0    0  962    1
##          E    0    0    0    0 1081
##
## Overall Statistics
##
##                Accuracy : 0.999
##                  95% CI : (0.9978, 0.9996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9987
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9991   0.9981   0.9979   0.9991
## Specificity            0.9998   0.9996   0.9996   0.9998   1.0000
## Pos Pred Value         0.9994   0.9982   0.9981   0.9990   1.0000
## Neg Pred Value         1.0000   0.9998   0.9996   0.9996   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1934   0.1740   0.1635   0.1837
## Detection Prevalence   0.2846   0.1937   0.1743   0.1636   0.1837
## Balanced Accuracy      0.9999   0.9994   0.9988   0.9989   0.9995
```

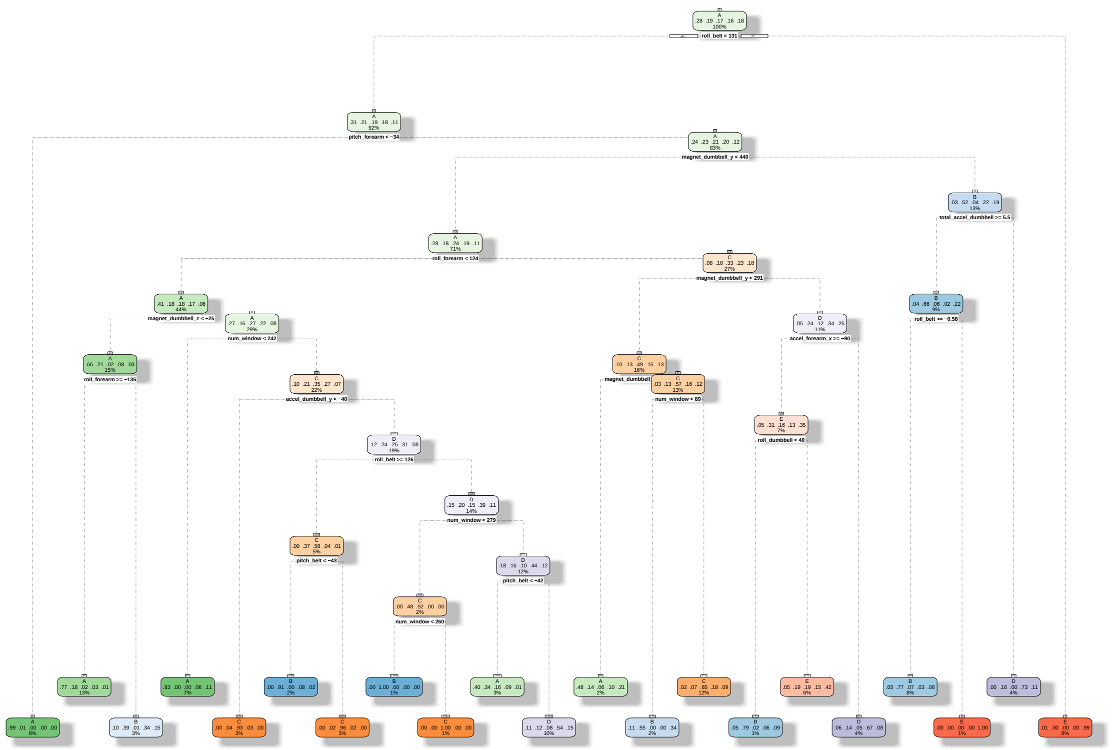
```r
# plot matrix results
plot(confMatRandForest$table, col = confMatRandForest$byClass,
     main = paste("Random Forest - Accuracy =", round(confMatRandForest$overall['Accuracy'], 4)))
```



b. Decision Tree

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

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



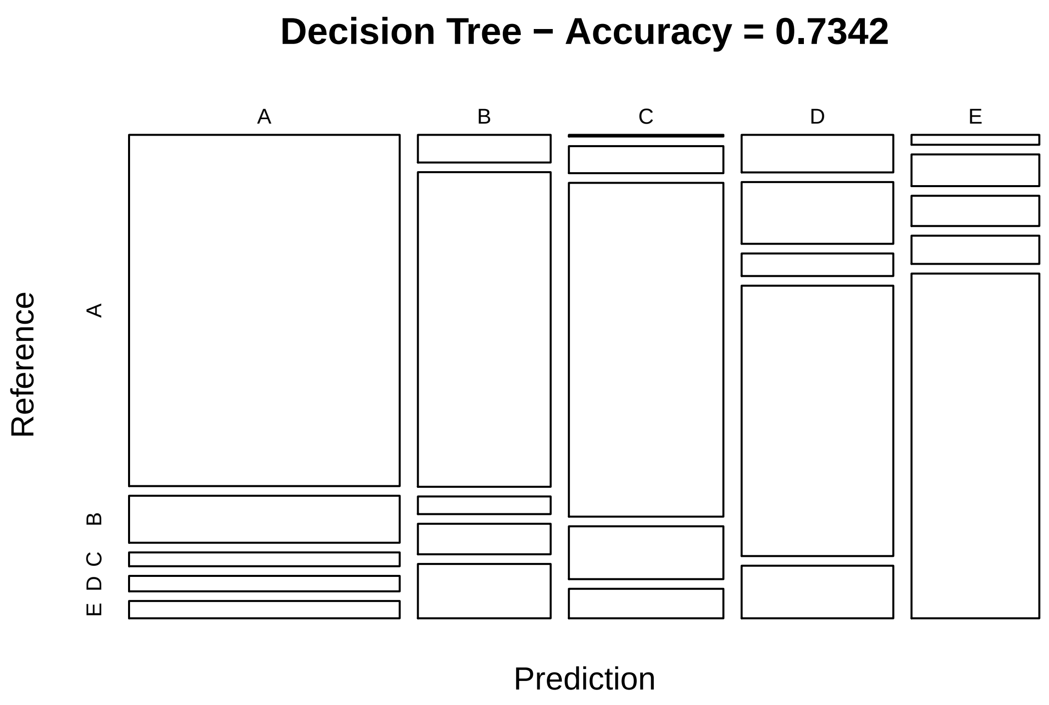
Rattle 2020−Dec−27 23:56:09 trinadey

```
# prediction on Test dataset
predictDecTree <- predict(modFitDecTree, newdata=testSet, type="class")
confMatDecTree <- confusionMatrix(predictDecTree, as.factor(testSet$classe))
confMatDecTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1502  201   59   66   74
##          B   58  660   37   64  114
##          C    4   66  815  129   72
##          D   90  148   54  648  126
##          E   20   64   61   57  696
##
## Overall Statistics
##
##                Accuracy : 0.7342
##                  95% CI : (0.7228, 0.7455)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6625
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8973   0.5795   0.7943   0.6722   0.6433
## Specificity            0.9050   0.9425   0.9442   0.9151   0.9579
## Pos Pred Value         0.7897   0.7074   0.7505   0.6079   0.7751
## Neg Pred Value         0.9568   0.9033   0.9560   0.9344   0.9226
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2552   0.1121   0.1385   0.1101   0.1183
## Detection Prevalence   0.3232   0.1585   0.1845   0.1811   0.1526
## Balanced Accuracy      0.9011   0.7610   0.8693   0.7936   0.8006
```

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



c. Generalized Boosted Model

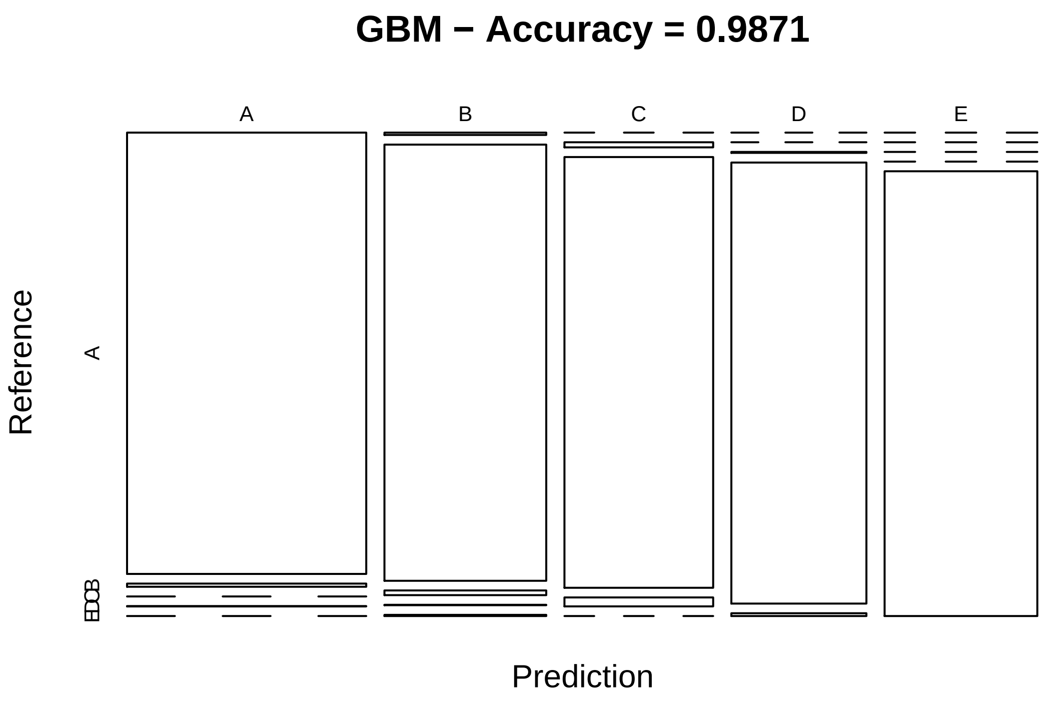
```
# model fit
set.seed(12345)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modFitGBM  <- train(classe ~ ., data=trainingSet, method = "gbm",trControl = controlGBM, verbose = FALSI
modFitGBM$finalModel
```

```
## 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 dataset
predictGBM <- predict(modFitGBM, newdata=testSet)
confMatGBM <- confusionMatrix(predictGBM, as.factor(testSet$classe))
confMatGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1668   12    0    1    0
##          B    6 1115   12    1    3
##          C    0   12 1012   21    0
##          D    0    0    2  941    6
##          E    0    0    0    0 1073
##
## Overall Statistics
##
##                Accuracy : 0.9871
##                  95% CI : (0.9839, 0.9898)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9837
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9789   0.9864   0.9761   0.9917
## Specificity            0.9969   0.9954   0.9932   0.9984   1.0000
## Pos Pred Value         0.9923   0.9807   0.9684   0.9916   1.0000
## Neg Pred Value         0.9986   0.9949   0.9971   0.9953   0.9981
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2834   0.1895   0.1720   0.1599   0.1823
## Detection Prevalence   0.2856   0.1932   0.1776   0.1613   0.1823
## Balanced Accuracy      0.9967   0.9871   0.9898   0.9873   0.9958
```

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



Conclusion

Since for prediction we have separate dataset, having a 70-30 split is sufficient for training and test data set for cross validation purposes. On applying different prediction models, we found that accuracy of the different models are -
a. Random Forest - 99.9%
b. Decision Trees - 73.42%
c. Generalized Boosted Model - 98.71%

So we are going to use Random Forest on our prediction data set as the accuracy is the maximum.

```
predictTEST <- predict(modFitRandForest, newdata=predictionTesting)
predictTEST
```

```
##  [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
```