Machine Learning Prediction Assignment

Kumar Shaket

04/05/2021

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.In this project , we are going to use the data frrom accelerometer on the belt, forearm,arm and dumbell of 6 participatns and perform prediction to pefcorm barbell lift correctly and incorrectly in 5 different way

Loading Libraries

```
library(lattice)
library(ggplot2)
library(caret)
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:ggplot2':
##
##     margin
```

```
library(rattle)
```

```
## 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.
```

```
##
## Attaching package: 'rattle'

## The following object is masked from 'package:randomForest':
##
##     importance
```

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

```
## corrplot 0.84 loaded
```

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
set.seed(1234)
```

Data Processing

Loading Csv files

```
trainraw <- read.csv(file="~/Desktop/Coursera/practicalmachinelearning/Data/pml-training.csv",header = T
validraw <- read.csv(file="~/Desktop/Coursera/practicalmachinelearning/Data/pml-testing.csv",header = TI
```

Count of records in training and test data

```
dim(trainraw)
```

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

```
dim(validraw)
```

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

Cleanning up data to remove any variables containing missing value

```
trainData<- trainraw[, colSums(is.na(trainraw)) == 0]
dim(trainData)
```

```
## [1] 19622    93
```

```
validData<- validraw[, colSums(is.na(validraw)) == 0]
dim(validData)
```

```
## [1] 20 60
```

Removing first 7 variables as they have little impact in the prediction

```
trainData <- trainData[, -c(1:7)]
dim(trainData)
```

```
## [1] 19622    86
```

```
validData <- validData[,-c(1:7)]
dim(validData)
```

```
## [1] 20 53
```

Futher removing variables that are near zero variance from training data set

```
nzv <- nearZeroVar(trainData)
trainData <-trainData[,-nzv]
dim(trainData)
```

```
## [1] 19622    53
```

Preparing the dataset for prediction

```
inTrain <- createDataPartition(trainData$classe,p=0.7,list=FALSE)
training <- trainData[inTrain,]
testing <- trainData[-inTrain,]
dim(training)
```

```
## [1] 13737    53
```

```
dim(testing)
```

```
## [1] 5885   53
```

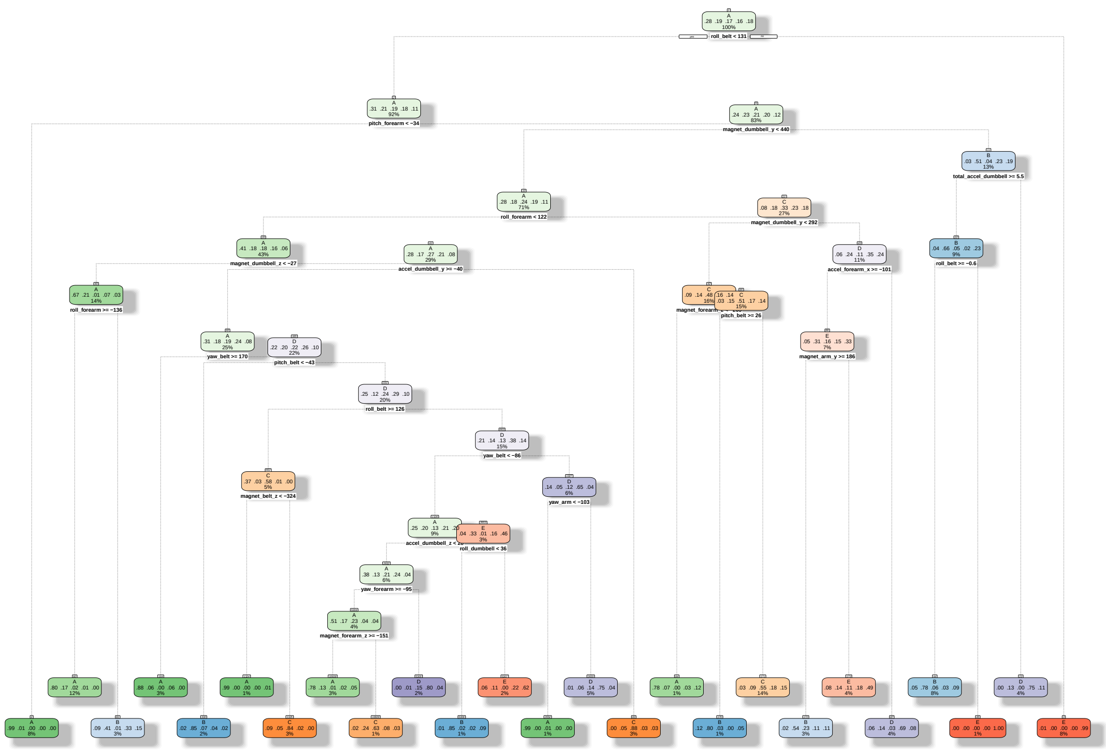
#Model building For this project we will use three different algorithms to predict the outcome. 1. Classification Trees 2. Random Forests 3. Generalized Boosted Model

Prediction with classification trees

We first obtail the model, and then we use the fancyRpartPlot() function to plot the classification tree as a dendogram.

```r
mod_tree <- rpart(classe ~ ., data=training, method="class")
fancyRpartPlot(mod_tree)
```

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



Rattle 2021−May−05 20:33:15 kumarshaket

Running Prediction on Testing dataset and preparing confusion matrix

```r
predict_mod <- predict(mod_tree,testing,type="class")
cmtree <- confusionMatrix(predict_mod,testing$classe)
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1522  167   12   49   13
##          B   58  706  100   79   96
##          C   47  109  819  148  139
##          D   25   94   67  609   52
##          E   22   63   28   79  782
##
## Overall Statistics
##
##                Accuracy : 0.7541
##                  95% CI : (0.7429, 0.7651)
```

```
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6885
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9092   0.6198   0.7982   0.6317   0.7227
## Specificity            0.9428   0.9298   0.9088   0.9516   0.9600
## Pos Pred Value         0.8633   0.6795   0.6490   0.7190   0.8029
## Neg Pred Value         0.9631   0.9106   0.9552   0.9295   0.9389
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2586   0.1200   0.1392   0.1035   0.1329
## Detection Prevalence   0.2996   0.1766   0.2144   0.1439   0.1655
## Balanced Accuracy      0.9260   0.7748   0.8535   0.7917   0.8414
```

We see that the accuracy rate of the model is low: 0.6967

Prediction with Random Forest

```
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modRF1 <- train(classe ~ ., data=training, method="rf", trControl=controlRF)
modRF1$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: 2
##
##         OOB estimate of  error rate: 0.71%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3903    3    0    0    0 0.0007680492
## B   15 2638    5    0    0 0.0075244545
## C    0   18 2374    4    0 0.0091819699
## D    0    0   43 2206    3 0.0204262877
## E    0    0    1    5 2519 0.0023762376
```

Running prediction on Testing Data and preparing confusion matrix

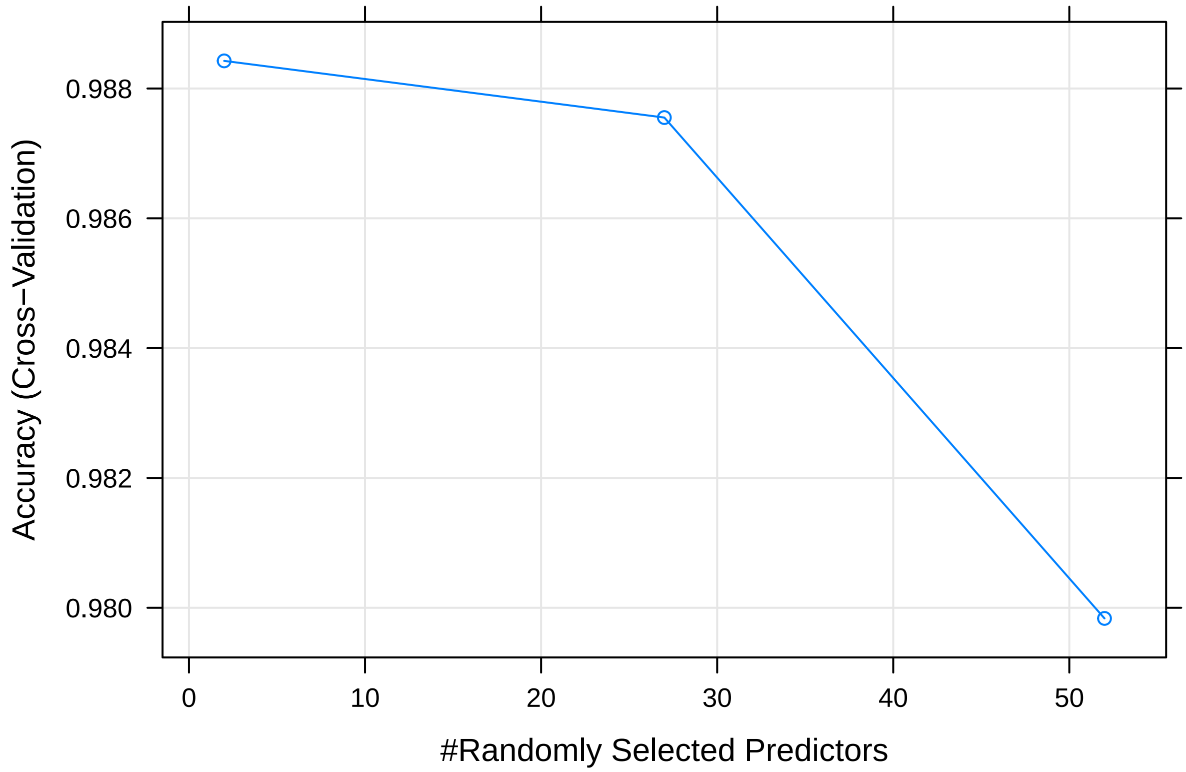
```
predictRF1 <- predict(modRF1, newdata=testing)
cmrf <- confusionMatrix(predictRF1,testing$classe)
cmrf
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    2    0    0    0
##          B    0 1133   15    0    0
##          C    0    4 1010   14    0
##          D    0    0    1  949    1
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9935
##                  95% CI : (0.9911, 0.9954)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9918
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9844   0.9844   0.9991
## Specificity            0.9995   0.9968   0.9963   0.9996   0.9998
## Pos Pred Value         0.9988   0.9869   0.9825   0.9979   0.9991
## Neg Pred Value         1.0000   0.9987   0.9967   0.9970   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1925   0.1716   0.1613   0.1837
## Detection Prevalence   0.2848   0.1951   0.1747   0.1616   0.1839
## Balanced Accuracy      0.9998   0.9958   0.9904   0.9920   0.9994
```

Plot

```r
plot(modRF1)
```



Gradient Boosted Trees

```
set.seed(1234)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modGBM  <- train(classe ~ ., data=trainData, method = "gbm", trControl = controlGBM, verbose = FALSE)
modGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 51 had non-zero influence.
```

```
print(modGBM)
```

```
## Stochastic Gradient Boosting
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 15698, 15697, 15698, 15698, 15697
## Resampling results across tuning parameters:
```

```
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7526748  0.6864515
##   1                  100      0.8205070  0.7727984
##   1                  150      0.8561307  0.8179463
##   2                   50      0.8540413  0.8150474
##   2                  100      0.9062782  0.8813911
##   2                  150      0.9326772  0.9148102
##   3                   50      0.8962894  0.8687150
##   3                  100      0.9425642  0.9273207
##   3                  150      0.9624399  0.9524803
##
## 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 = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

Running prediction on Testing Datasets and preparing confusion matrix

```
predictGBM <- predict(modGBM,newdata=testing)
cmGBM <- confusionMatrix(predictGBM,testing$classe)
cmGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1661   27    0    1    1
##          B    7 1091   30    4    4
##          C    5   21  986   22    4
##          D    1    0   10  933    7
##          E    0    0    0    4 1066
##
## Overall Statistics
##
##                Accuracy : 0.9749
##                  95% CI : (0.9705, 0.9787)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9682
##
##  Mcnemar's Test P-Value : 6.451e-05
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9922   0.9579   0.9610   0.9678   0.9852
## Specificity            0.9931   0.9905   0.9893   0.9963   0.9992
## Pos Pred Value         0.9828   0.9604   0.9499   0.9811   0.9963
## Neg Pred Value         0.9969   0.9899   0.9917   0.9937   0.9967
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
```

```
## Detection Rate         0.2822   0.1854   0.1675   0.1585   0.1811
## Detection Prevalence   0.2872   0.1930   0.1764   0.1616   0.1818
## Balanced Accuracy      0.9927   0.9742   0.9752   0.9821   0.9922
```

Based on comparision , The accuracy rate using the random forest is very high: Accuracy : 0.9897 and therefore the *out-of-sample-error is equal to 0.0103**.

Applying the Best Model to the Validation Data

By comparing the accuracy rate values of three modesl, it is clear the Random Forest model is best model for prediction and hence we are running this model on top of validation data.

```
Results <- predict(modRF1,newdata=validData)
Results
```

```
##  [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
```