Practical Machine Learning Course Project 1

psegdav

2/24/2021

I. Introduction

This is the Course Project for Practical Machine Learning Course of Johns Hopkins University on Coursera.

Due to devices as Nike FuelBand, Fitbit, and Jawbone Up, personal activity data can be collected in an inexpensive way. These are part of the quantified self movement devices. These devices allow to quantify how much of a particular activity the person does, however it is rarely quantified how well does cavity is done.

In this course project we will use data from accelerometer on the belt, forearm, arm, and dumbell of 6 participants. Each of the participants was asked to perform barbell lift in a proper way and incorrectly, in five different ways. The training and testing sets can be found in https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv and https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv, respectively. For more information regarding the data, please visit: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har.

The goal is to predict the manner in which the participant did the exercise. This is shown in the "classe" variable within the train set. This will be done by creating a prediction model to predict 20 different test cases.

II. Data Analysis

```
library(knitr)
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(rpart)
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(gbm)

## Loaded gbm 2.1.8

train <- read.csv("pml-training.csv")
test <- read.csv("pml-testing.csv")

dim(train); dim(test)

## [1] 19622   160

## [1]  20 160
```

We can see that the training set has 19622 observations and the testing set has 20 observations. Once we examine the each data set to a major extent, we can identify many NA values, these values are not useful for out prediction models and therefore we will now clean the data by eliminating the NA values.The clean training data set will be ctrain

```
ctrain <- train[,colMeans(is.na(train))<0.9]
```

Once the NA values have been removed, we will remove the values with a near zero variance.

```
cctrain <- nearZeroVar(ctrain)
ctrain <- ctrain[,-cctrain]

dim(ctrain)

## [1] 19622    59
```

The ctrain (clean training data set) will now be split, in order to have a validation data set (validation) and a training data set (utrain).

```
div_ctrain <- createDataPartition(y=ctrain$classe, p=0.7, list=FALSE)
validation <- ctrain[-div_ctrain,]
utrain <- ctrain[div_ctrain,]
```

III. Model Creation and Testing

We will set up a control for the training.

```
ctrl <- trainControl(method="cv", number=5)
```

We will start by creating a random forest model, as it is highly accurate.

```
mod_rf <- train(classe ~ ., method="rf", data=utrain, trControl= ctrl,
allowParallel=T)
```

```
mod_rf
```

```
## Random Forest
##
## 13737 samples
##    58 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10988, 10991, 10990, 10989, 10990
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9956327  0.9944757
##   41    0.9999272  0.9999079
##   80    0.9999272  0.9999079
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 41.
```

To also have a more visual model, we will create a decision tree model.

```
mod_dt <- train(classe ~ ., method="rpart", data=utrain)

mod_dt
```

```
## CART
##
## 13737 samples
##    58 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.2437188  0.7296898  0.6565086
##   0.2568406  0.5719617  0.4535485
##   0.2703692  0.3905685  0.1844772
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.2437188.
```

IV. Accuracy on validation and training set

We will now revise the accuracy of Random Forests and Decision Trees on the validation data set.

```
pred_rf <- predict(mod_rf, validation)
cmatrix_rf <- confusionMatrix(pred_rf, factor(validation$classe))
cmatrix_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    0    0    0    0
##          B    1 1139    0    0    0
##          C    0    0 1026    0    0
##          D    0    0    0  964    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9998
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9998
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   0.9998   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   0.9991   1.0000   1.0000   1.0000
## Neg Pred Value         0.9998   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1935   0.1743   0.1638   0.1839
## Detection Prevalence   0.2843   0.1937   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9997   0.9999   1.0000   1.0000   1.0000
```

```
pred_dt <- predict(mod_dt, validation)
cmatrix_dt <- confusionMatrix(pred_dt, factor(validation$classe))
cmatrix_dt
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 1673    0    0    0    0
##          B    1 1139    0    0    0
##          C    0    0    0    0    0
##          D    0    0    0    0    0
##          E    0    0 1026  964 1082
##
## Overall Statistics
##
##                Accuracy : 0.6617
##                  95% CI : (0.6494, 0.6738)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5694
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   1.0000   0.0000   0.0000   1.0000
## Specificity            1.0000   0.9998   1.0000   1.0000   0.5857
## Pos Pred Value         1.0000   0.9991      NaN      NaN   0.3522
## Neg Pred Value         0.9998   1.0000   0.8257   0.8362   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1935   0.0000   0.0000   0.1839
## Detection Prevalence   0.2843   0.1937   0.0000   0.0000   0.5220
## Balanced Accuracy      0.9997   0.9999   0.5000   0.5000   0.7928
```

V. Results & Conclusions

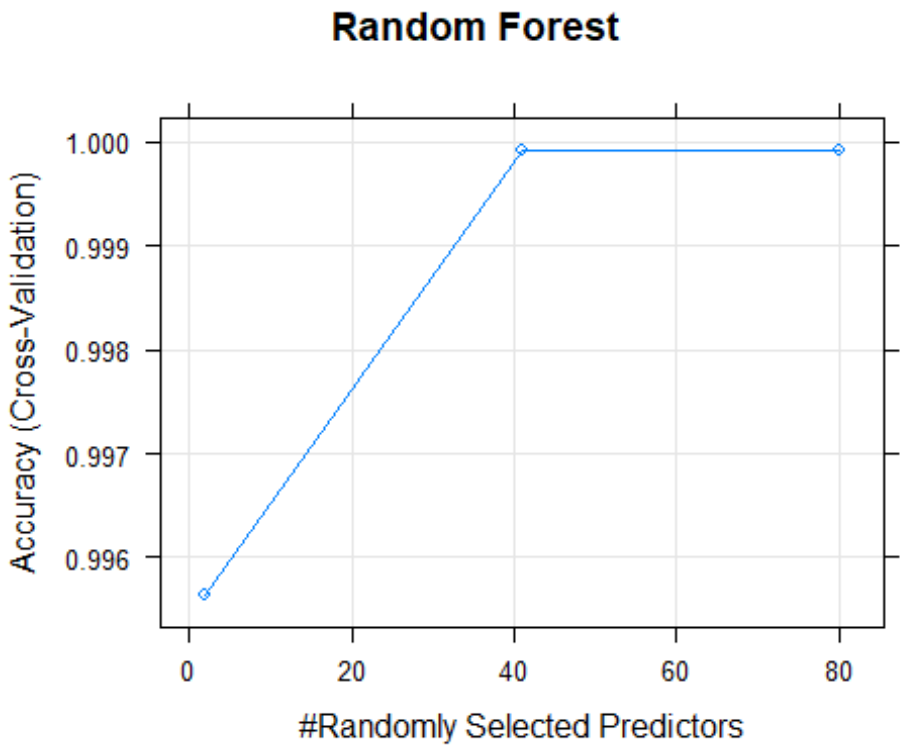
Based on the analyses made, the Random Forest is the best model, as it has an accuracy of 1 and a no information rate of 0.28. The Decision Tree model had an accuracy of 0.66 and a no information rate of 0.28. Due to the accuracy, it can be concluded that the Random Forest model is the most appropriate to make predictions on the test set.

```
pred_test <- predict(mod_rf, test)
pred_test
```

```
##  [1] A A A A A A A A A A A A A A A A A A A A
## Levels: A B C D E
```

V. Appendix

```
plot(mod_rf, main="Random Forest")
```




```
plot(mod_dt, main="Decision Tree")
```


```
plot(pred_test, main="Test Prediction")
```


Test Prediction
0
5
10
15
20
A
B
C
D
E