Practical Machine Learning - Course Project

Sleiman Ghusayni

6/27/2020

Prediction Assignment Writeup

The goal of your project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Loading Libraries

```
rm(list = ls(all = TRUE))
library(knitr)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

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

```
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(corrplot)
```

```
## corrplot 0.84 loaded
```

```
set.seed(12345)
```

Getting Data

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

# partitioning the training dataset
inTrain  <- createDataPartition(training$classe, p=0.7, list=FALSE)
TrainSet <- training[inTrain, ]
TestSet  <- training[-inTrain, ]
dim(TrainSet)
```

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

```
dim(TestSet)
```

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

Cleaning Data

We got 160 variables with many NA values.

```
# clean variables with Nearly Zero Variance
remove_NZvar <- nearZeroVar(TrainSet)
TrainSet <- TrainSet[, -remove_NZvar]
TestSet  <- TestSet[, -remove_NZvar]
dim(TrainSet)
```

```
## [1] 13737   104
```

```
dim(TestSet)
```

```
## [1] 5885  104
```

```
# clean varaibles mostly NA
AllNA    <- sapply(TrainSet, function(x) mean(is.na(x))) > 0.95
TrainSet <- TrainSet[, AllNA==FALSE]
TestSet  <- TestSet[, AllNA==FALSE]
dim(TrainSet)
```

```
## [1] 13737    59
```

```
dim(TestSet)
```

```
## [1] 5885   59
```

```
# clean identification variables
TrainSet <- TrainSet[, -(1:5)]
TestSet  <- TestSet[, -(1:5)]
dim(TrainSet)
```

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

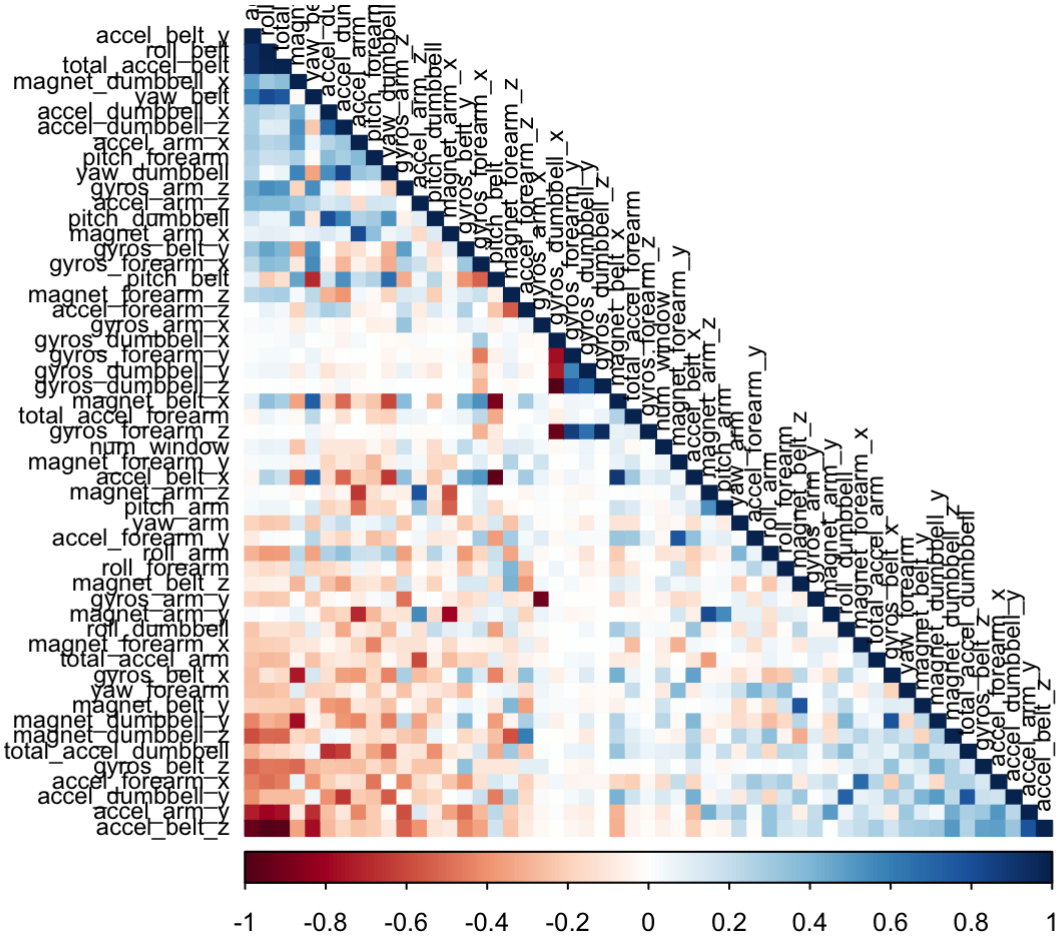
```
dim(TestSet)
```

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

Correlation Analysis

Analysing the correlation before we build model.

```
corMatrix <- cor(TrainSet[, -54])
corrplot(corMatrix, order = "FPC", method = "color", type = "lower", tl.cex = 0.75, t
l.col = rgb(0, 0, 0))
```



Selecting Prediction Models

The methods are: Random Forests and Decision Tree.

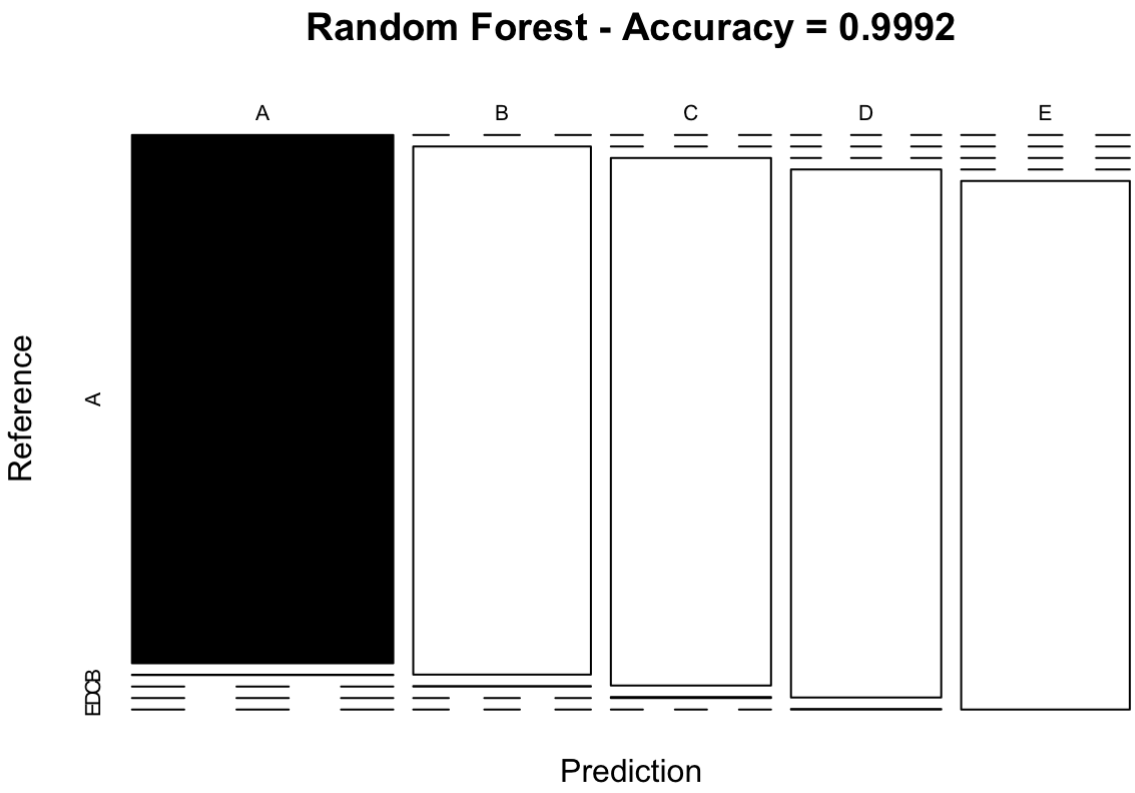
```
# 1 - Random Forest
# model fit
set.seed(12345)
controlRF <- trainControl(method="cv", number=5, allowParallel=TRUE)
modFitRandForest <- train(classe ~ ., data=TrainSet, method="rf", trControl=controlR
F)

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    1    0    0    1 0.0005120328
## B    6 2648    3    1    0 0.0037622272
## C    0    6 2388    2    0 0.0033388982
## D    0    0    8 2244    0 0.0035523979
## E    0    0    0    3 2522 0.0011881188
```

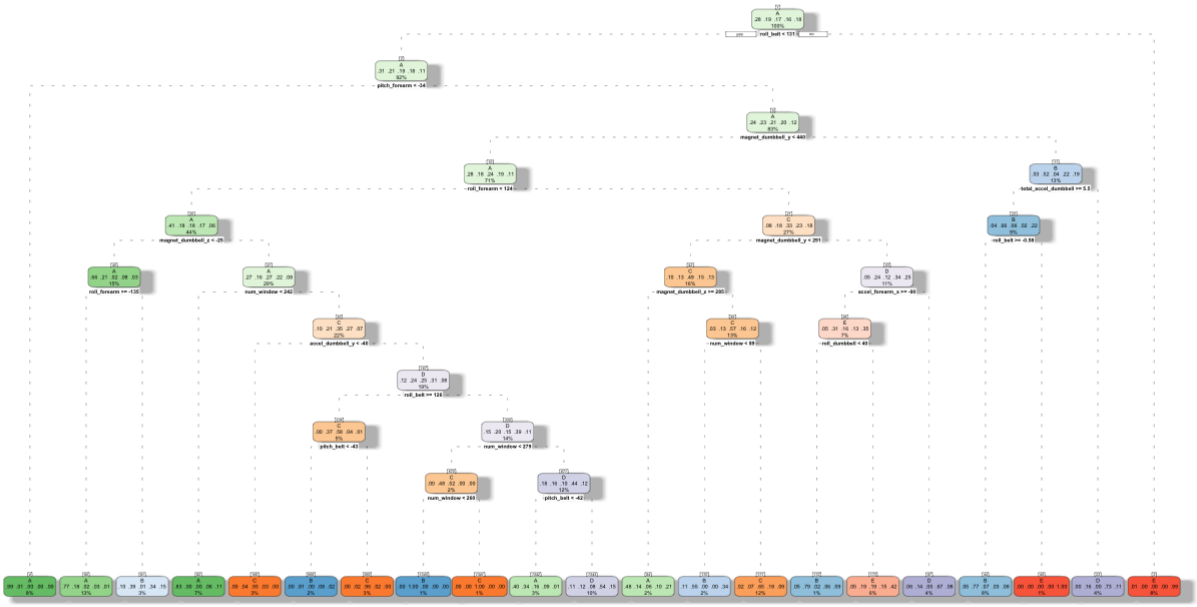
```
# prediction on Test dataset
predictRandForest <- predict(modFitRandForest, newdata=TestSet)
confMatRandForest <- confusionMatrix(predictRandForest, as.factor(TestSet$classe))
confMatRandForest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    1    0    0    0
##          B    0 1138    1    0    0
##          C    0    0 1025    2    0
##          D    0    0    0  962    1
##          E    0    0    0    0 1081
##
## Overall Statistics
##
##                Accuracy : 0.9992
##                  95% CI : (0.998, 0.9997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9989
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9991   0.9990   0.9979   0.9991
## Specificity            0.9998   0.9998   0.9996   0.9998   1.0000
## Pos Pred Value         0.9994   0.9991   0.9981   0.9990   1.0000
## Neg Pred Value         1.0000   0.9998   0.9998   0.9996   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1934   0.1742   0.1635   0.1837
## Detection Prevalence   0.2846   0.1935   0.1745   0.1636   0.1837
## Balanced Accuracy      0.9999   0.9995   0.9993   0.9989   0.9995
```

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




```
# 2 - Decision Trees

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

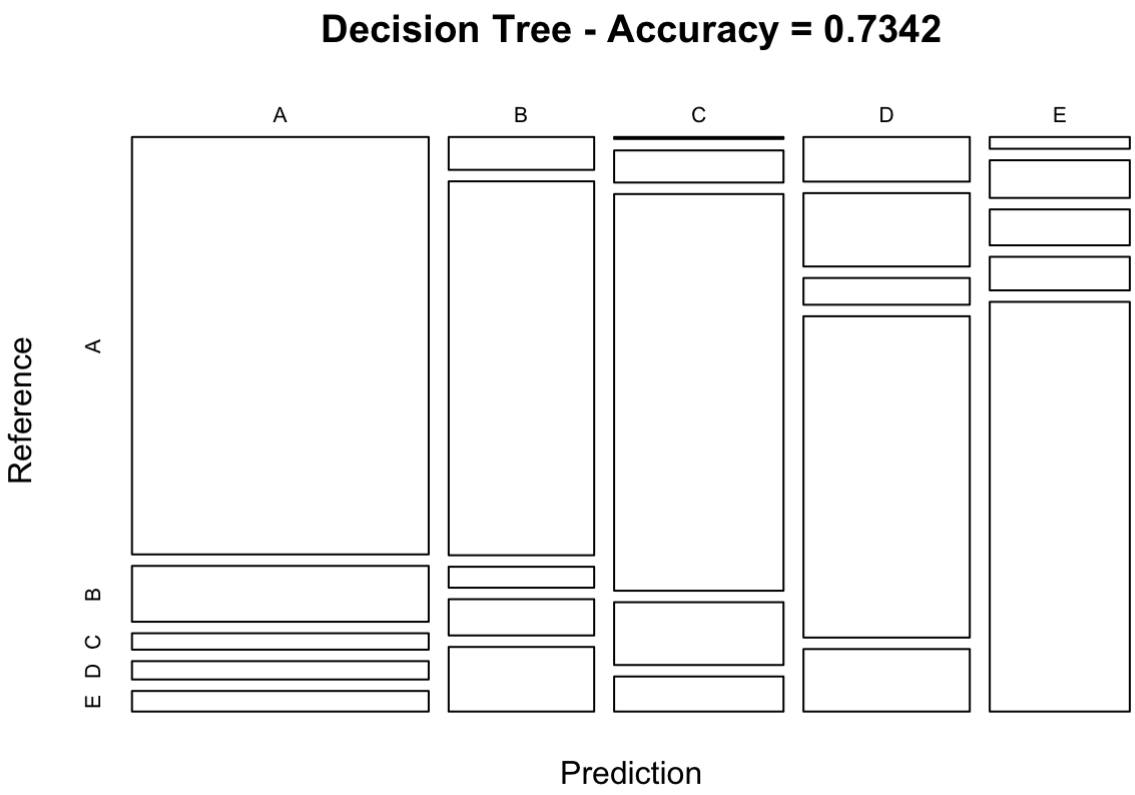


Rattle 2020-Jun-27 21:43:24 i331473

```
# 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)))
```



The accuracy of the 2 models done above are:

Random Forest : 0.9992 Decision Tree : 0.7342

Applying the best model to the validation data

By comparing the accuracy rate values of the two models, it is clear the the 'Random Forest' model is the best So will use it on the validation data

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