Prediction Assignment Writeup - Module 8: Practical Machine Learning

Roddy Mendoza Marriott

2026-02-09

Background

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. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

The data

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

-The test data are available here: [https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv]

-The data for this project come from this source: [http://groupware.les.inf.puc-rio.br/har]. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Some insights about the project

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.

Data processing

First, loading the packages and libraries.

```
library(plyr)
library(dplyr)
library(lattice)
library(ggplot2)
library(caret)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(kernlab)
library(randomForest)
library(knitr)
library(e1071)
```

Getting and cleaning data

```
trainingst <- read.csv("pml-training.csv")
testingst <- read.csv("pml-testing.csv")
```

```
dim(trainingst)
```

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

```
dim(testingst)
```

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

Preprocessing and cleaning data

we should Exclude the obvious columns i.e `X`, `user_name`,`raw_timestamp_part_1`,`raw_timestamp_part_2`, `cvtd_timestamp`,`roll_belt` which are the first 7 columns. We should also delete missing values and variables with near zero variance.

```
#Deleting missing values
trainingst <- read.csv("pml-training.csv", na.strings=c("NA","#DIV/0!",""))
testingst <- read.csv("pml-testing.csv", na.strings=c("NA","#DIV/0!",""))
```

```
#Deleting missing values
trainingst<-trainingst[,colSums(is.na(trainingst)) == 0]
testingst <-testingst[,colSums(is.na(testingst)) == 0]
```

```
#Removing columns that are not predictors, which are the the seven first columns
trainingst   <-trainingst[,-c(1:7)]
testingst <-testingst[,-c(1:7)]
```

```
dim(trainingst)
```

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

```
dim(testingst)
```

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

From the above code block sum(completeCase) == nrows confirm that the number of complete case is equal to number of rows in `trainingdf` same for `testingdf`

Now we have only 53 columns(features) are left. we can preproccess the training and testing i.e converting into scales of `0` to `1` and replacing any `NA` values to average of that columns.

Partition the data set into training and testing data from `trainingst`

```
inTrain <- createDataPartition(y = trainingst$classe, p=0.75, list = FALSE)
training <- trainingst[inTrain, ]
testing <- trainingst[-inTrain, ]
```

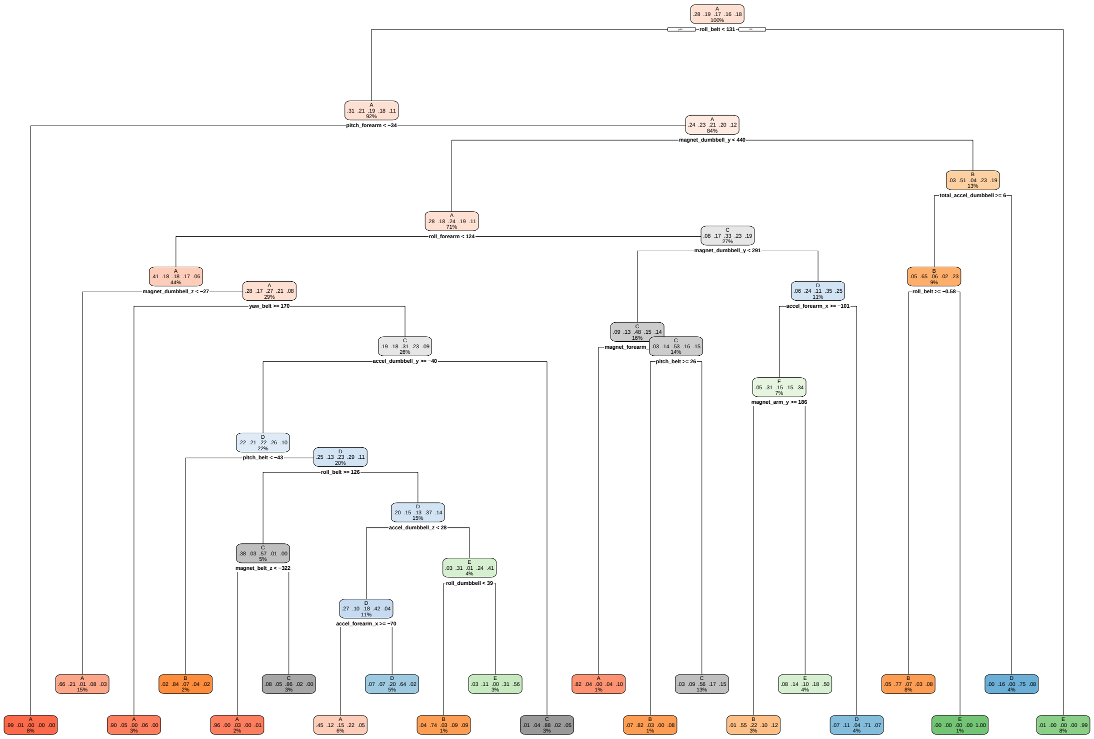
Training the model

Two methods will be applied to model, and the best one will be used for the(testingst) predictions.

The methods are: `Decision Tree` and `Random Forests`.

Model 1: Training the model with Decision Trees

```
set.seed(40000)
fitDT <- rpart(classe ~ .,training, method="class")
# Normal plot
rpart.plot(fitDT)
```



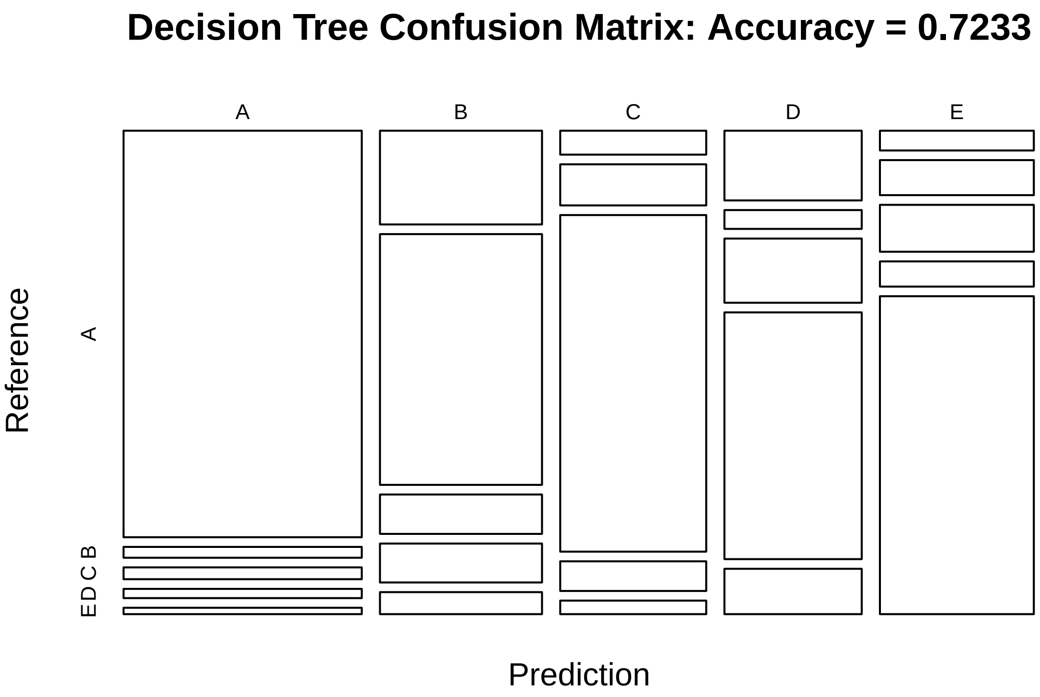
```
#Use model to predict classe in validation testing set
predictionDT <- predict(fitDT, testing, type = "class")
```

```
#Estimate the errors of the prediction algorithm in the Decision Tree model
cmdt <-confusionMatrix(as.factor(testing$classe), predictionDT)
cmdt
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1275   34   37   29   20
##          B  200  535   84   83   47
##          C   46   79  647   57   26
##          D  126   34  116  446   82
##          E   40   71   95   51  644
##
## Overall Statistics
##
##                Accuracy : 0.7233
##                  95% CI : (0.7105, 0.7358)
##     No Information Rate : 0.344
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6474
```

```
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7558   0.7105   0.6609  0.66967   0.7863
## Specificity            0.9627   0.9003   0.9470  0.91553   0.9371
## Pos Pred Value         0.9140   0.5638   0.7567  0.55473   0.7148
## Neg Pred Value         0.8826   0.9449   0.9180  0.94634   0.9563
## Prevalence             0.3440   0.1535   0.1996  0.13581   0.1670
## Detection Rate         0.2600   0.1091   0.1319  0.09095   0.1313
## Detection Prevalence   0.2845   0.1935   0.1743  0.16395   0.1837
## Balanced Accuracy      0.8592   0.8054   0.8039  0.79260   0.8617
```

```
# Accuracy plot
plot(cmdt$table, col = cmdt$byClass,
main = paste("Decision Tree Confusion Matrix: Accuracy =", round(cmdt$overall['Accuracy'], 4)))
```



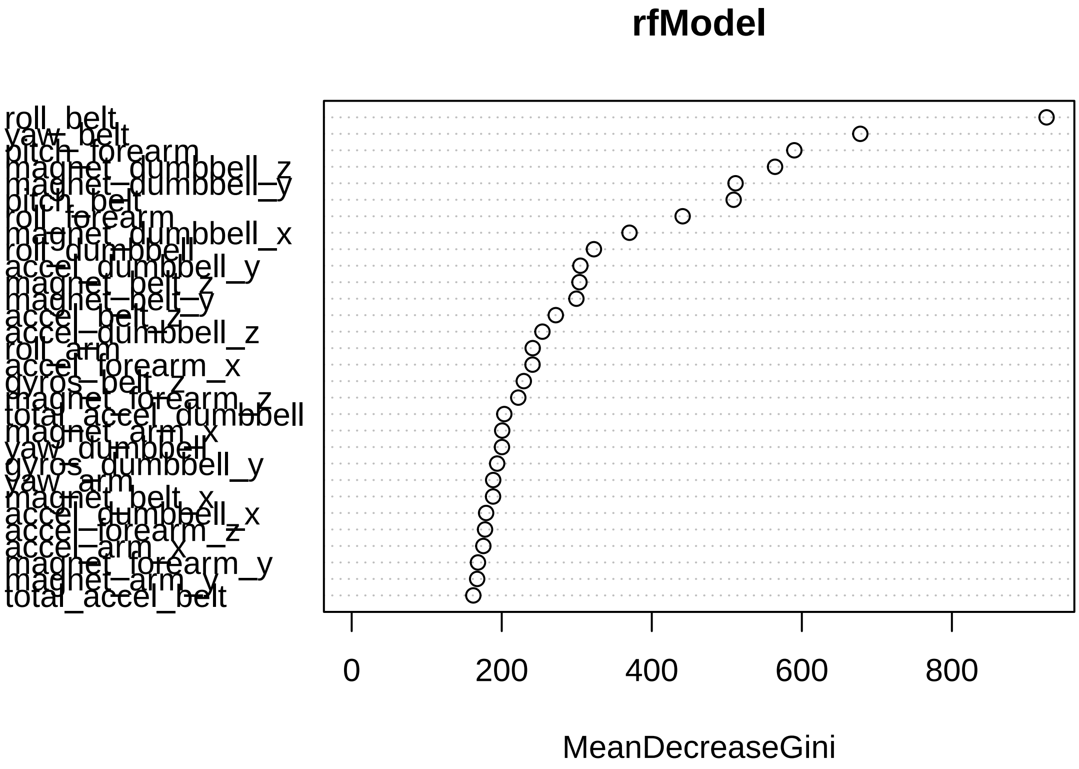
Model 2: Training the model using Random Forest

```
rfModel <- randomForest(as.factor(classe)~., data=training)
# Summary of the model
rfModel
```

```
##
```

```
## Call:
##  randomForest(formula = as.factor(classe) ~ ., data = training)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.53%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4180    3    0    1    1 0.001194743
## B   12 2830    6    0    0 0.006320225
## C    0   17 2549    1    0 0.007012076
## D    0    0   28 2384    0 0.011608624
## E    0    0    1    8 2697 0.003325942
```

```
# Plot the variable importance
varImpPlot(rfModel)
```

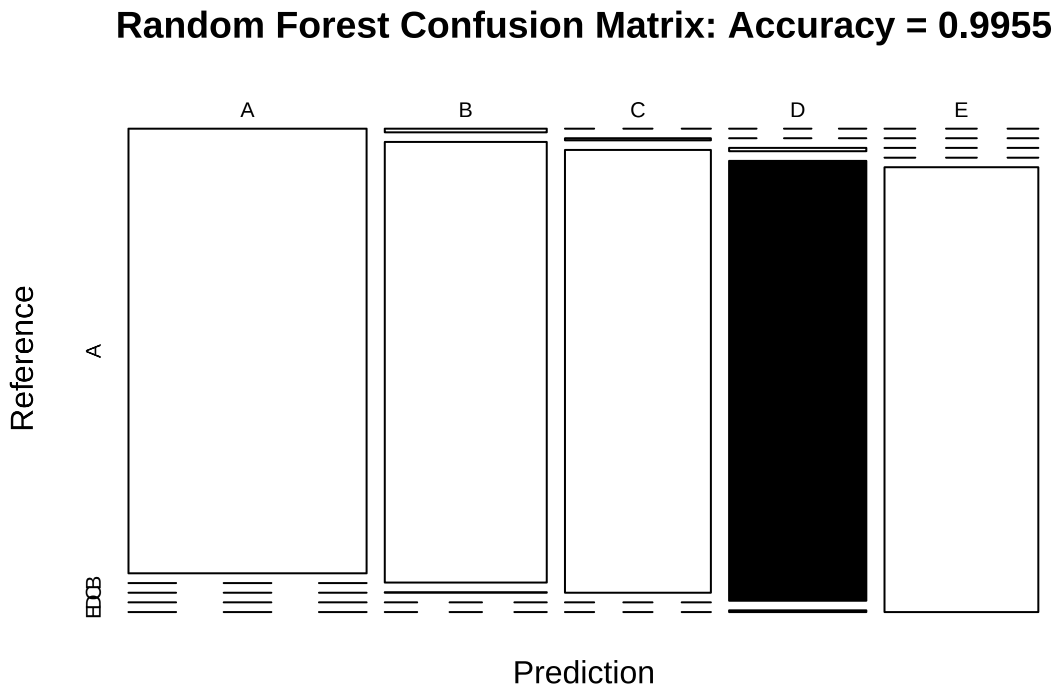


```
# Confusion matrix with testing
predTesting <- predict(rfModel, testing)
rfcfm  <- confusionMatrix(as.factor(testing$classe), predTesting)
rfcfm
```

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

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    8  940    1    0    0
##          C    0    4  851    0    0
##          D    0    0    6  795    3
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.9955
##                  95% CI : (0.9932, 0.9972)
##     No Information Rate : 0.2861
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9943
##
##  Mcnemar’s Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9943   0.9958   0.9918   1.0000   0.9967
## Specificity            1.0000   0.9977   0.9990   0.9978   1.0000
## Pos Pred Value         1.0000   0.9905   0.9953   0.9888   1.0000
## Neg Pred Value         0.9977   0.9990   0.9983   1.0000   0.9993
## Prevalence             0.2861   0.1925   0.1750   0.1621   0.1843
## Detection Rate         0.2845   0.1917   0.1735   0.1621   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9971   0.9967   0.9954   0.9989   0.9983
```

```
plot(rfcfm$table, col = rfcfm$byClass, main = paste("Random Forest Confusion Matrix: Accuracy =", round
```



Remarks

-`Decision Tree Model` is the worst model running, it has the low mean and the highest standard deviation.

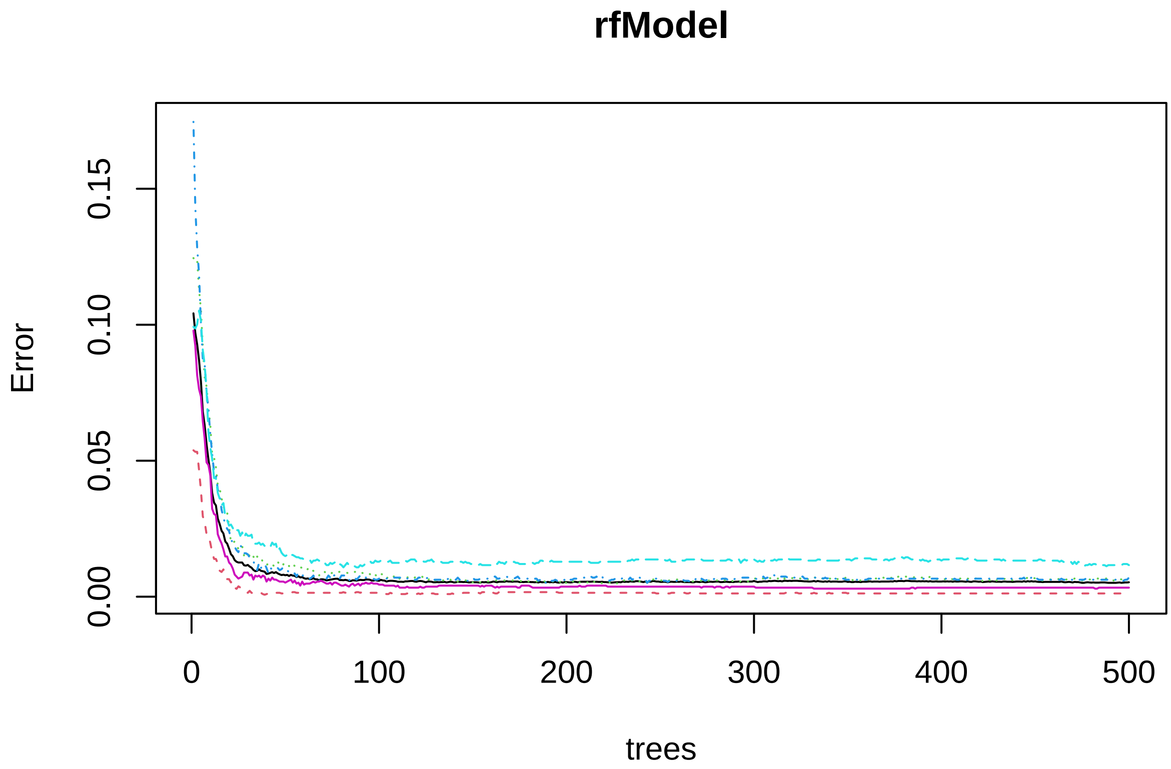
-`Random Forest Model` it has the highest mean accuracy and lowest standard deviation.

Depending on how your model is to be used, the interpretation of the kappa statistic might vary One common interpretation is shown as follows:

- Poor agreement = Less than 0.20
- Fair agreement = 0.20 to 0.40
- Moderate agreement = 0.40 to 0.60
- Good agreement = 0.60 to 0.80
- Very good agreement = 0.80 to 1.00

This two models preforms as expected, the deviation from the cross validation accuracy is low.

```r
#plot the model
plot(rfModel)
```



The predictive accuracy of the `Random Forest Model` is excellent at 99.8 %. Accuracy has plateaued, and further tuning would only yield decimal gain.

Making prediction on the 20 data pointsusing random forest

`Decision Tree Model`: 73.43%, Random Forest Model: 99.53% The Random Forest model is selected and applied to make predictions on the 20 data points from the original testing dataset (testingst)

```
rfPredictions <- predict(rfModel, testingst,type= "class")
rfPredictions
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```