Prediction Assignment Project

Terry Jones

November 8, 2018

Project Goal

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 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, etc. The goal of your project is to predict the manner in which they did the exercise. 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.

Acquire Datasets and Prepare for Analysis

```
#load the data libraries that may be needed to support analysis
library(caret, warn.conflicts = FALSE, quietly = TRUE)
library(rpart, warn.conflicts = FALSE, quietly = TRUE)
library(rpart.plot, warn.conflicts = FALSE, quietly = TRUE)
library(RColorBrewer, warn.conflicts = FALSE, quietly = TRUE)
library(rattle, warn.conflicts = FALSE, quietly = TRUE)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest, warn.conflicts = FALSE, quietly = TRUE)
```

```
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(gbm, warn.conflicts = FALSE, quietly = TRUE)
```

```
## Loaded gbm 2.1.4
```

```
library(plyr, warn.conflicts = FALSE, quietly = TRUE)
#download data from the internet
download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
              destfile = "./pml-train.csv", method = "curl")
```

```
# Load the training dataset (df - download file)
dfTrain <- read.csv("./pml-train.csv", na.strings=c("NA","#DIV/0!",""))

#download data from the internet
download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
              destfile = "./pml-test.csv", method = "curl")

# Load the testing dataset (df - download file)
dfTest <- read.csv("./pml-test.csv", na.strings=c("NA","#DIV/0!",""))
```

Clean the Data

Remove columns with missing values, “NA” values, etc. Time dependence values will be removed.

```
features <- names(dfTest[,colSums(is.na(dfTest)) == 0])[8:59]

# Only use features used in testing cases
dfTrain <- dfTrain[,c(features,"classe")]
dfTest <- dfTest[,c(features,"problem_id")]

#View data file dimensions for each file post cleansing
dim(dfTrain); dim(dfTest);
```

```
## [1] 19622    53
## [1] 20 53
```

Partition the Dataset

Use a 60/40 split (train/test) for partitioning the dataset as was alluded to in the training materials to increase performance and accuracy of the model. Therefore, p is set = to 0.6

```
set.seed(10)

inTrain <- createDataPartition(dfTrain$classe, p=0.6, list=FALSE)
train <- dfTrain[inTrain,]
test <- dfTrain[-inTrain,]
```

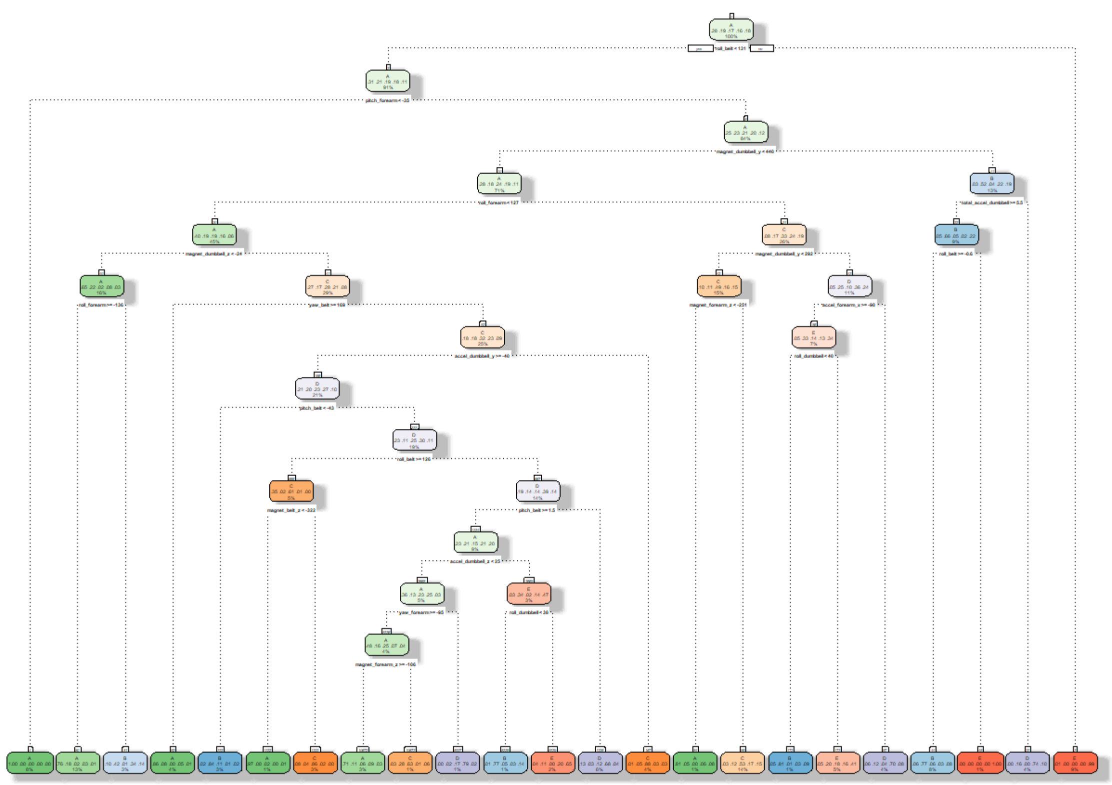
```
dim(train); dim(test);
## [1] 11776    53
## [1] 7846   53
```

Build the Decision Tree

Although easy to interpret, results may be variable which will affect accuracy. Will set cross validation to "cv" and 10 for resampling. Will set method to "class" to get a factor of classifications based on the responses.

```
modFDT <- rpart(classe ~ ., data = train, method="class", control = rpart.control(method = "cv", number = 10))


fancyRpartPlot(modFDT)
```



Rattle 2018-Nov-08 16:05:22 tljon

Predict with the Decision Tree Model

If we can get an accuracy of 70% or above, then we'll consider it to be acceptable.

```
pred <- predict(modFDT, test, type = "class")
confusionMatrix(pred, test$classe)
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1994  264   49   85   15
##          B   65  836   72  108  121
##          C   69  194 1095  174  186
##          D   67  112   78  805   78
##          E   37  112   74  114 1042
##
## Overall Statistics
##
##                Accuracy : 0.7357
##                  95% CI : (0.7258, 0.7454)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6649
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8934   0.5507   0.8004   0.6260   0.7226
## Specificity            0.9264   0.9422   0.9038   0.9489   0.9474
## Pos Pred Value         0.8284   0.6955   0.6374   0.7061   0.7556
## Neg Pred Value         0.9562   0.8974   0.9555   0.9283   0.9381
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2541   0.1066   0.1396   0.1026   0.1328
## Detection Prevalence   0.3068   0.1532   0.2190   0.1453   0.1758
```

```
## Balanced Accuracy      0.9099   0.7464   0.8521   0.7875   0.8350
```

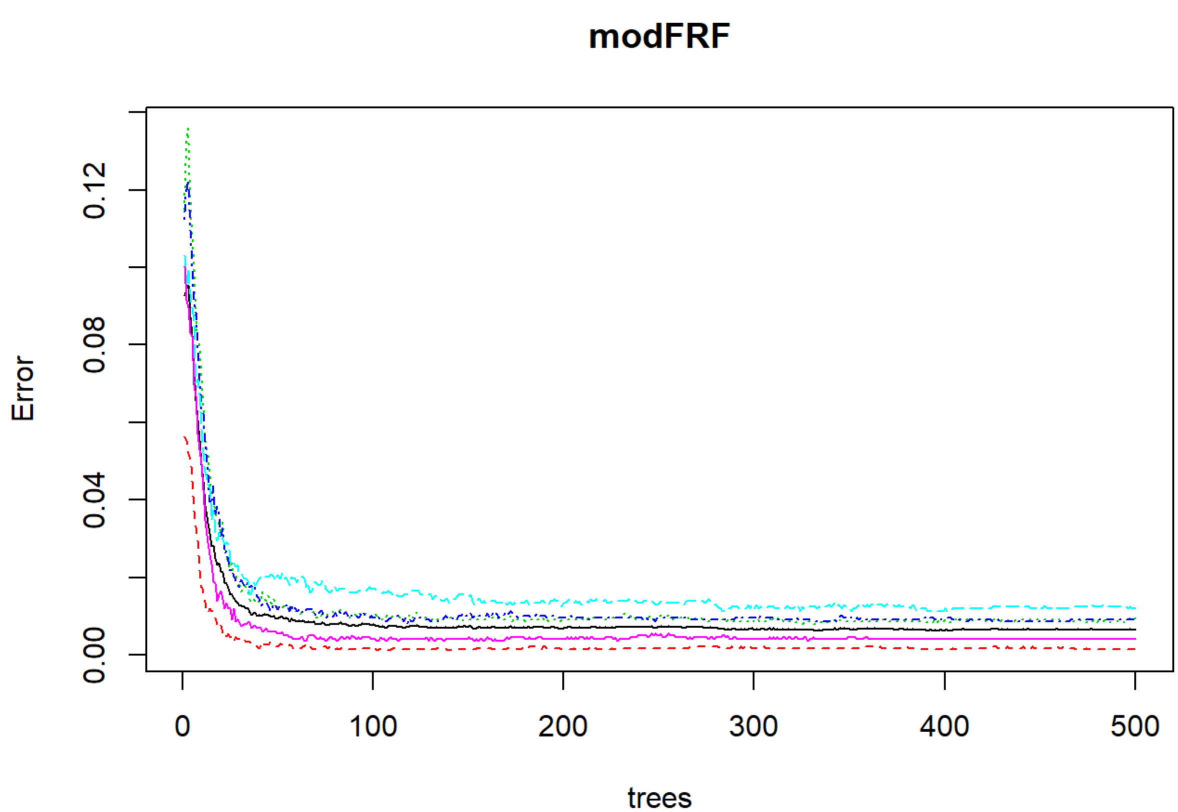
Accuracy is 72.6%, thereby, we consider to be acceptable.

Build the Random Forest (rf) Model

One of the proces of the Random Forest Model is accuracy. But Overfitting can be a problem. As stated previously, we will use a 40% test sample. The error estimate is expected to be less than 5%.

```
modFRF <- randomForest(classe ~ ., data = train, method = "rf", importance =
T, trControl = trainControl(method = "cv", classProbs=TRUE,savePredictions=TR
UE,allowParallel=TRUE, number = 10))


plot(modFRF)
```



Build the Generalized Boosted Regression Model (gbm)

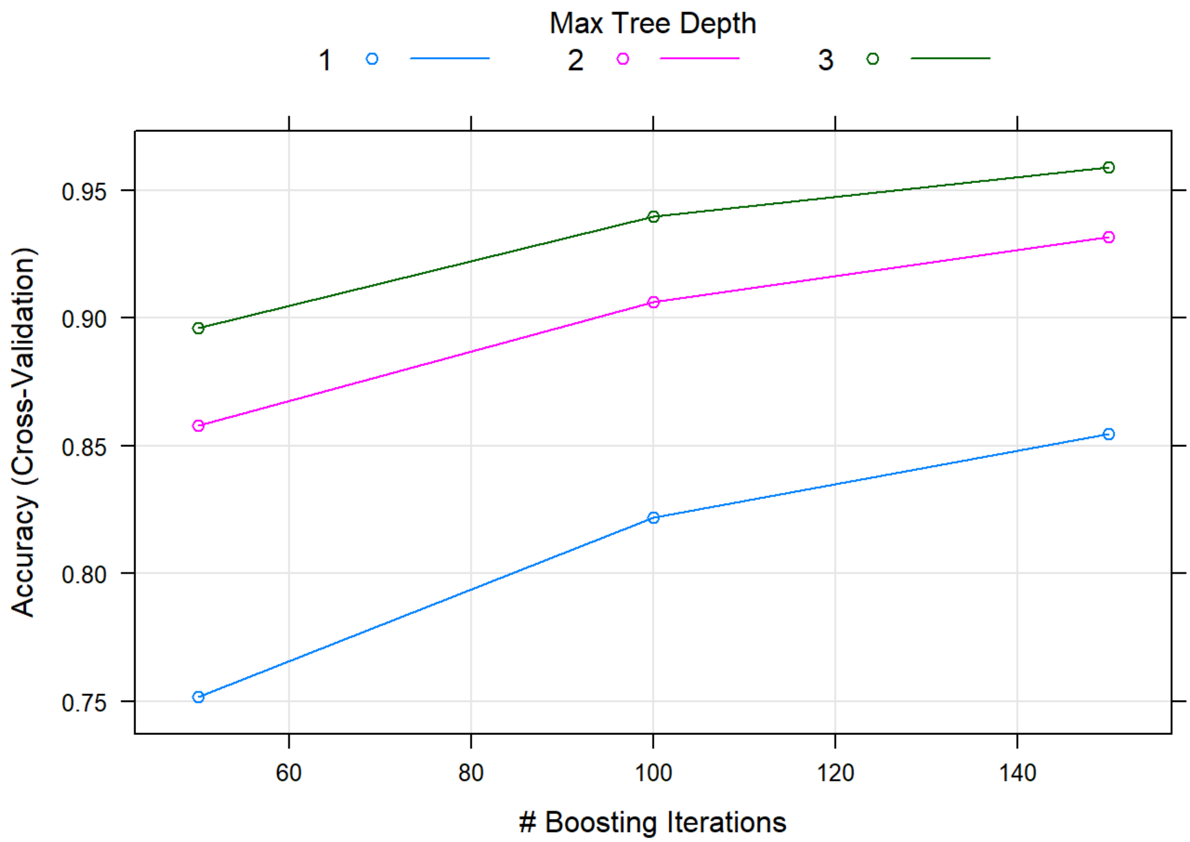
The goal is to minimize error on the training set. We will use gbm (boosting with trees). Will set cross validation to “cv” and 10 for resampling. Will set Verbose to False to avoid the extensive info and error logs being printed.

```
modFB <- train(classe ~ ., method = "gbm", data = train,
                    verbose = F,
                    trControl = trainControl(method = "cv", number = 10))


modFB
```

```
## Stochastic Gradient Boosting
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 10598, 10599, 10600, 10597, 10598, 10599, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7516973  0.6850818
##   1                  100      0.8220076  0.7747438
##   1                  150      0.8546175  0.8159894
##   2                   50      0.8577619  0.8197622
##   2                  100      0.9064200  0.8815455
##   2                  150      0.9318102  0.9136974
##   3                   50      0.8960575  0.8683848
##   3                  100      0.9397924  0.9238092
##   3                  150      0.9588157  0.9478923
##
## 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.
plot(modFB)
```



Predict with the rf Model

```
pred <- predict(modFRF, test, type = "class")
confusionMatrix(pred, test$classe)
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    3    0    0    0
##          B    0 1513    7    1    0
##          C    0    2 1361   13    0
##          D    0    0    0 1271    6
```

```
##          E    0    0    0    1 1436
##
## Overall Statistics
##
##                Accuracy : 0.9958
##                  95% CI : (0.9941, 0.9971)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9947
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9967   0.9949   0.9883   0.9958
## Specificity            0.9995   0.9987   0.9977   0.9991   0.9998
## Pos Pred Value         0.9987   0.9947   0.9891   0.9953   0.9993
## Neg Pred Value         1.0000   0.9992   0.9989   0.9977   0.9991
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1928   0.1735   0.1620   0.1830
## Detection Prevalence   0.2849   0.1939   0.1754   0.1628   0.1832
## Balanced Accuracy      0.9997   0.9977   0.9963   0.9937   0.9978
```

The rf model achieved 99.4% accuracy.

Predict with gbm

```r
pred <- predict(modFB, test)
confusionMatrix(pred, test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2189   48    0    1    3
```

```
##          B   29 1423   42   10   18
##          C    8   44 1308   36   18
##          D    5    1   15 1227   20
##          E    1    2    3   12 1383
##
## Overall Statistics
##
##                Accuracy : 0.9597
##                  95% CI : (0.9551, 0.964)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9491
##  Mcnemar's Test P-Value : 8.925e-09
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9807   0.9374   0.9561   0.9541   0.9591
## Specificity            0.9907   0.9844   0.9836   0.9938   0.9972
## Pos Pred Value         0.9768   0.9350   0.9250   0.9677   0.9872
## Neg Pred Value         0.9923   0.9850   0.9907   0.9910   0.9908
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2790   0.1814   0.1667   0.1564   0.1763
## Detection Prevalence   0.2856   0.1940   0.1802   0.1616   0.1786
## Balanced Accuracy      0.9857   0.9609   0.9699   0.9739   0.9781
```

The gbm achieved 95.9% accuracy.

Predict with the Test Dataset

```
predDT <- predict(modFDT, dfTest)
predDT
```

```
##             A           B          C          D           E
## 1  0.03041363 0.119221411 0.53163017 0.16788321 0.150851582
```

```
## 2  0.75634518 0.182741117 0.01840102 0.03362944 0.008883249
## 3  0.04885993 0.203583062 0.17752443 0.15960912 0.410423453
## 4  0.13056836 0.033794163 0.11981567 0.67895545 0.036866359
## 5  0.70662461 0.110410095 0.06309148 0.09463722 0.025236593
## 6  0.03041363 0.119221411 0.53163017 0.16788321 0.150851582
## 7  0.05823293 0.116465863 0.04016064 0.70080321 0.084337349
## 8  0.70662461 0.110410095 0.06309148 0.09463722 0.025236593
## 9  0.99676724 0.003232759 0.00000000 0.00000000 0.000000000
## 10 0.75634518 0.182741117 0.01840102 0.03362944 0.008883249
## 11 0.03041363 0.119221411 0.53163017 0.16788321 0.150851582
## 12 0.04885993 0.203583062 0.17752443 0.15960912 0.410423453
## 13 0.03041363 0.119221411 0.53163017 0.16788321 0.150851582
## 14 0.99676724 0.003232759 0.00000000 0.00000000 0.000000000
## 15 0.04885993 0.203583062 0.17752443 0.15960912 0.410423453
## 16 0.03816794 0.106870229 0.00000000 0.20229008 0.652671756
## 17 0.96932515 0.000000000 0.01840491 0.00000000 0.012269939
## 18 0.09830508 0.416949153 0.01355932 0.33559322 0.135593220
## 19 0.09830508 0.416949153 0.01355932 0.33559322 0.135593220
## 20 0.04968944 0.813664596 0.01242236 0.03105590 0.093167702
```

Apply the rf Prediction

```
predRF <- predict(modFRF, dfTest)
predRF
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```

Apply the gbm Prediction

```
predgbm <- predict(modFB, dfTest)
predgbm
```

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

File to be Submitted

The rf model appears to have a high level of accuracy at 99.5%. With a level of accuracy this high, we can feel confident that any test cases that are submitted for analysis will be accurate.

```
project_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

project_files(predRF)
```