Practical Machine Learning

Erik Rehnberg Steeb

8/15/2020

Setup R session

```
## Loading required package: lattice

## Loading required package: ggplot2

## 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

## corrplot 0.84 loaded

## Loading required package: foreach

## Loading required package: iterators

## Loading required package: parallel
```

Load and Clean Data

First step is to load and clean the data

```r
testData <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
trainData <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
# Remove columns with NAs
testData <- testData[, colSums(is.na(testData)) == 0]
trainData <- trainData[, colSums(is.na(trainData)) == 0]
# Remove columns that almost certainly do not have anything to do with prediction
testRemNames <- colnames(testData[,1:7])
trainRemNames <- colnames(trainData[,1:7])
testData <- testData[,!(names(testData) %in% testRemNames)]
```

```r
trainData <- trainData [,!(names(trainData) %in% trainRemNames)]
# Remove empty variables from trainData
trainClasse <- trainData$classe
trainData <- trainData[sapply(trainData, is.numeric)]
trainData$classe <- trainClasse
trainData$classe <- as.factor(trainData$classe)
```

Slice Data

Slice data into training and validation test sets

```r
set.seed(221989) # For reproducible purposes
inTrain <- createDataPartition(trainData$classe, p=0.70, list=F)
trueTrain <- trainData[inTrain,]
validateData <- trainData[-inTrain,]
```

Create the Model

I'll be using the Random Forest method, since it selects important variables itself (and is the default for Caret).

```r
trainctrl <- trainControl(verboseIter = FALSE)
rfFit <- train(x = trueTrain[,1:52], y = trueTrain$classe,
               method = "rf",
               trControl = trainctrl
)

rfFit
```

```
## Random Forest
##
## 13737 samples
##    52 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:
##
##   mtry  Accuracy   Kappa
##    2    0.9881654  0.9850270
##   27    0.9874988  0.9841830
##   52    0.9804433  0.9752563
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Cross Validate

Now that we have a model, I'll estimate the accuracy on our validation dataset.

```r
validateData$classe <- as.factor(validateData$classe)

predict <- predict(rfFit, validateData)
cMatrix <- confusionMatrix(validateData$classe, predict)

cMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    0    0    0    1
##          B   10 1119   10    0    0
##          C    0    6 1019    1    0
##          D    0    0   24  939    1
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9908
##                  95% CI : (0.988, 0.9931)
##     No Information Rate : 0.286
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9884
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9941   0.9947   0.9677   0.9979   0.9982
## Specificity            0.9998   0.9958   0.9986   0.9949   0.9998
## Pos Pred Value         0.9994   0.9824   0.9932   0.9741   0.9991
## Neg Pred Value         0.9976   0.9987   0.9930   0.9996   0.9996
## Prevalence             0.2860   0.1912   0.1789   0.1599   0.1840
## Detection Rate         0.2843   0.1901   0.1732   0.1596   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9969   0.9952   0.9831   0.9964   0.9990
```

This gives us a very low estimated out-of-sample error rate - something like 0.03% of instances are likely to be misclassified.

Test the model

Time to actually test the model using the testData set

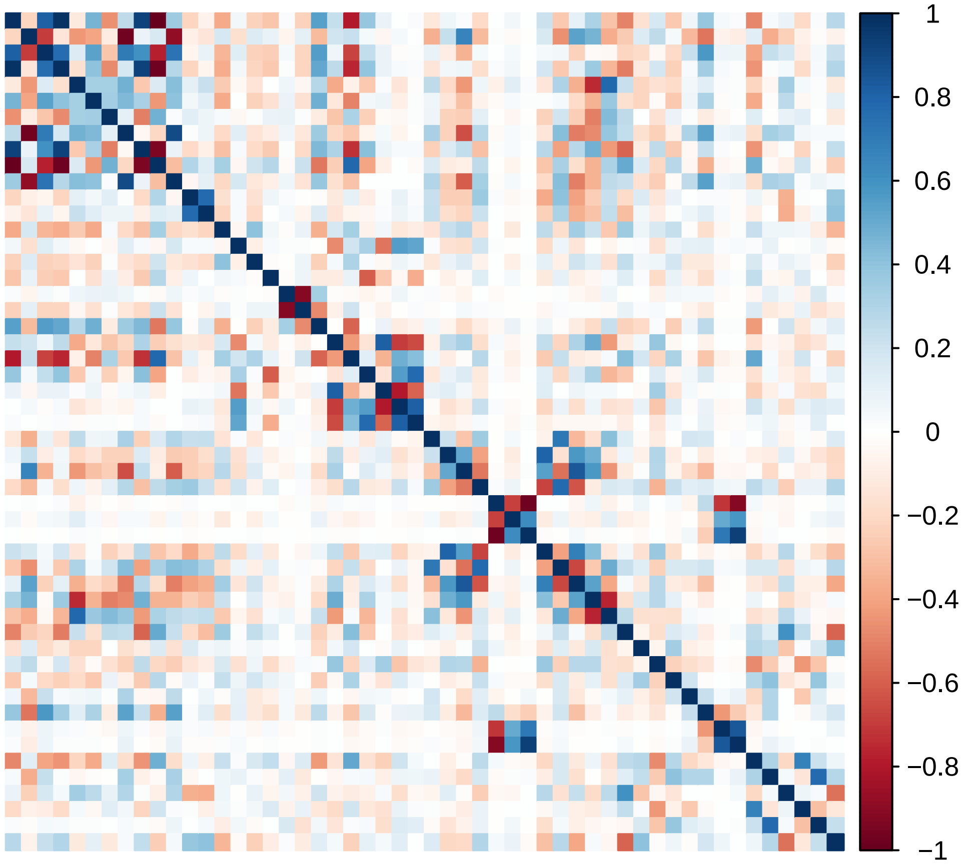
```r
# Remove last column of testData
testData1 <- testData[,1:52]
predictTest <- predict(rfFit, testData)

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
```

Appendix: Figures

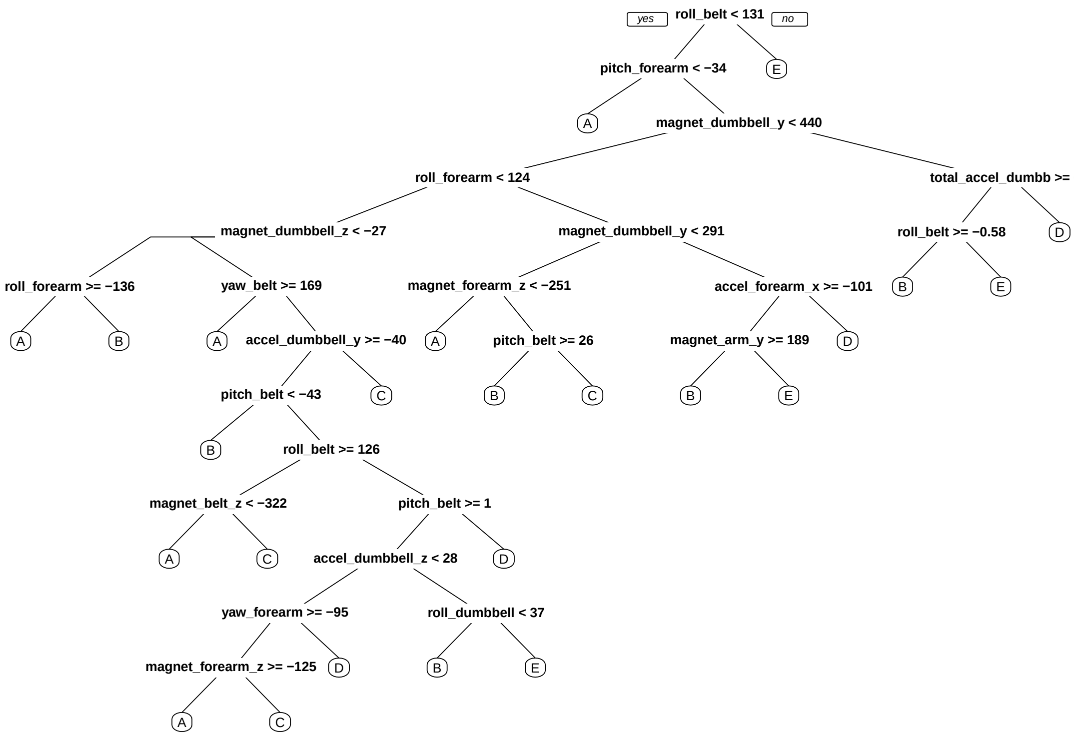
1. Correlation Matrix Visualization

```
corrPlot <- cor(trainData[, -length(names(trainData))])
corrplot(corrPlot, method="color", tl.pos='n')
```



2. Decision Tree Visualization

```
treeModel <- rpart(classe ~ ., data=trainData, method="class")
prp(treeModel) # fast plot
```


yes
roll_belt < 131
no
pitch_forearm < −34
E
A
magnet_dumbbell_y < 440
roll_forearm < 124
total_accel_dumbb >= 6
magnet_dumbbell_z < −27
magnet_dumbbell_y < 291
roll_belt >= −0.58
D
roll_forearm >= −136
yaw_belt >= 169
magnet_forearm_z < −251
accel_forearm_x >= −101
B
E
A
B
A
accel_dumbbell_y >= −40
A
pitch_belt >= 26
magnet_arm_y >= 189
D
pitch_belt < −43
C
B
C
B
E
B
roll_belt >= 126
magnet_belt_z < −322
pitch_belt >= 1
A
C
accel_dumbbell_z < 28
D
yaw_forearm >= −95
roll_dumbbell < 37
magnet_forearm_z >= −125
D
B
E
A
C