Practical Machine Learning Project

Cory Breaux

3/9/2020

Executive Summary

Loading and Processing the Data

```
# load libraries
library(caret)
library(rpart)
library(rpart.plot)
library(rattle)
library(randomForest)
library(corrplot)
```

```
# download datasets
trainData<- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-tra
ining.csv"),
                     na.strings = c("NA", "#DIV/0",""))
testData<- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-test
ing.csv"),
                    na.strings = c("NA", "#DIV/0",""))

# partition training data
set.seed(123)
inTrain<-createDataPartition(trainData$classe, p=0.65, list=FALSE)
myTrain <- trainData[inTrain,]
myTest <- trainData[-inTrain,]
dim(myTrain)
```

```
## [1] 12757   160
```

```
dim(myTest)
```

```
## [1] 6865  160
```

```
# remove near-zero-variance variables
NZV <- nearZeroVar(myTrain)
myTrain <- myTrain[,-NZV]
myTest <- myTest[,-NZV]
dim(myTrain)
```

```
## [1] 12757   121
```

```
dim(myTest)
```

```
## [1] 6865  121
```

```
# remove mostly (greater than 90%) NA variables
mostNA <- sapply(myTrain, function(x) mean(is.na(x))) > 0.9
myTrain<-myTrain[, mostNA==FALSE]
myTest<-myTest[, mostNA==FALSE]
dim(myTrain)
```

```
## [1] 12757    59
```

```
dim(myTest)
```

```
## [1] 6865   59
```

```
# lastly, remove ID variables
myTrain<-myTrain[, -c(1:5)]
myTest<-myTest[,-c(1:5)]
dim(myTrain)
```

```
## [1] 12757    54
```

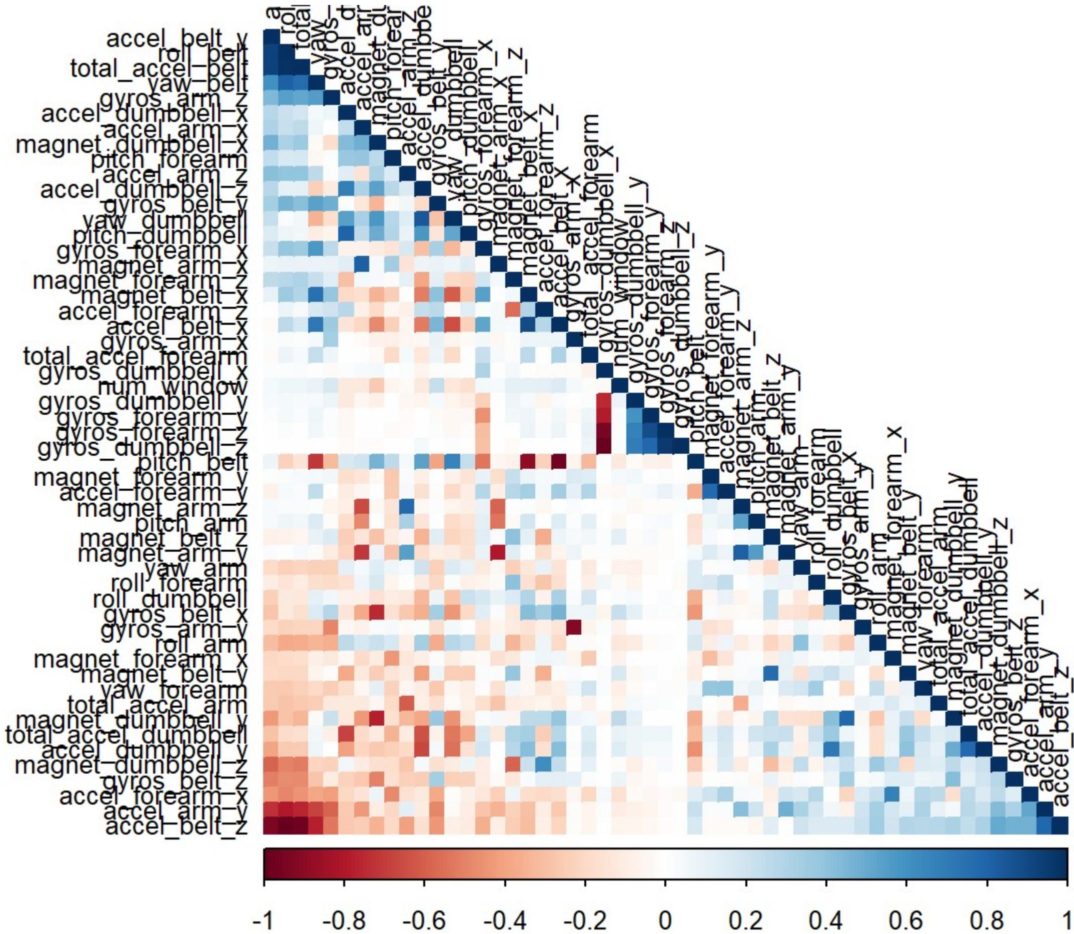
```
dim(myTest)
```

```
## [1] 6865   54
```

Now that the data has been cleaned and processed, we can begin the analysis.

Correlation Analysis

```
correlationMatrix<-cor(myTrain[,-54])
corrplot(correlationMatrix, order="FPC", method="color", type="lower", tl.cex=0.8, t
l.col=rgb(0,0,0))
```



Darker colors indicate higher correlation.

Model Building

To model the data, we will utilize Random Forests, a Decision Tree and a Generalized Boosted Regression Model. We will use the model with the highest accuracy when applied to the test set.

Random Forest

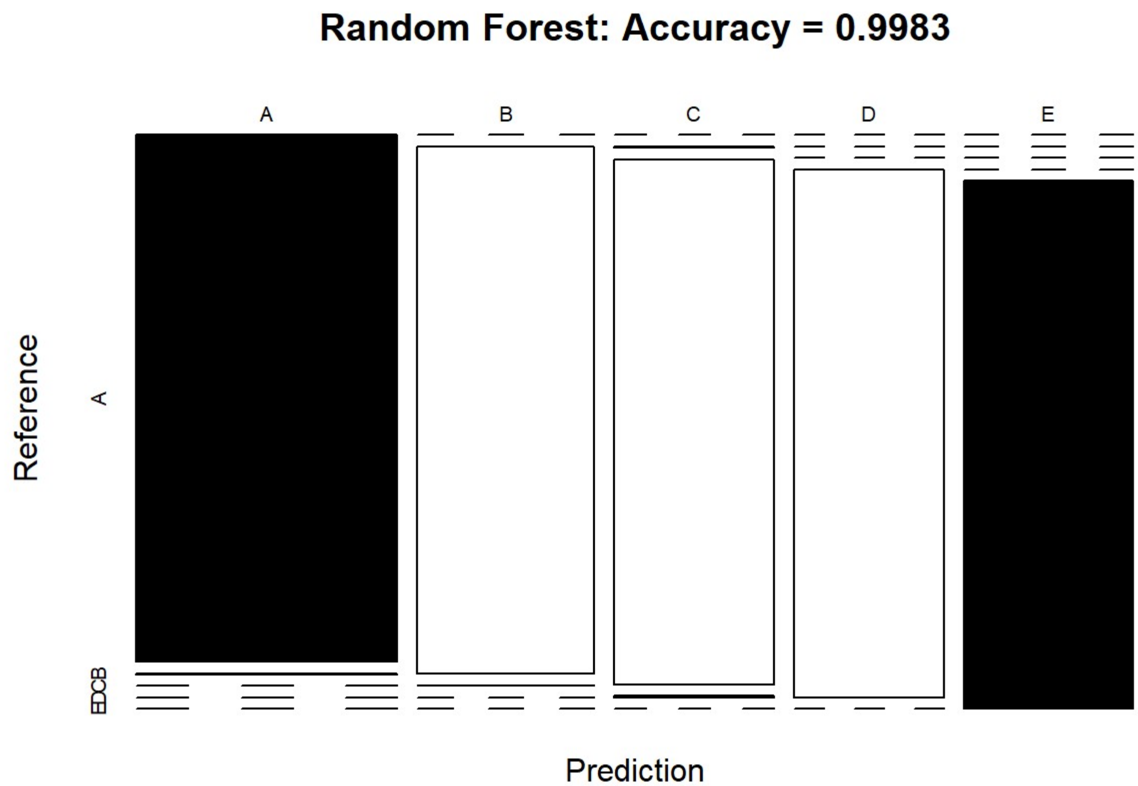
```
#train the model
controlRForest<-trainControl(method="cv", number=3, verboseIter=FALSE)
modelFitRF<-train(classe ~ ., data=myTrain, method="rf", trControl=controlRForest)
modelFitRF$finalModel
```

```
##
## 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: 0.31%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3625    1    0    0    1 0.0005514199
## B    6 2462    1    0    0 0.0028351559
## C    0    5 2219    1    0 0.0026966292
## D    0    0   15 2074    2 0.0081300813
## E    0    1    0    6 2338 0.0029850746
```

```
#validate the model
predictRForest <- predict(modelFitRF, newdata = myTest)
conMatRF<- confusionMatrix(predictRForest, myTest$classe)
conMatRF
```

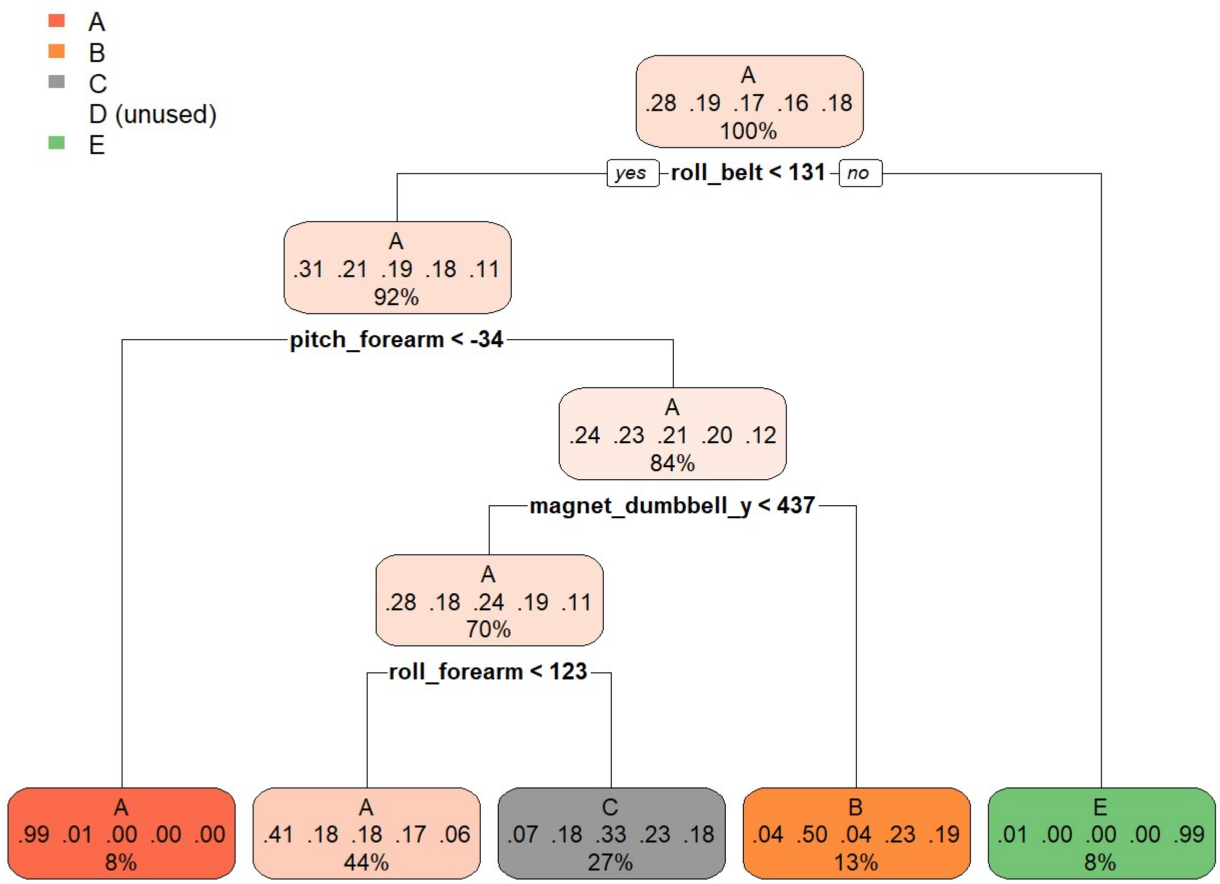
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1953    4    0    0    0
##          B    0 1321    1    0    0
##          C    0    3 1196    4    0
##          D    0    0    0 1121    0
##          E    0    0    0    0 1262
##
## Overall Statistics
##
##                Accuracy : 0.9983
##                  95% CI : (0.9969, 0.9991)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9978
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9992   0.9964   1.0000
## Specificity            0.9992   0.9998   0.9988   1.0000   1.0000
## Pos Pred Value         0.9980   0.9992   0.9942   1.0000   1.0000
## Neg Pred Value         1.0000   0.9987   0.9998   0.9993   1.0000
## Prevalence             0.2845   0.1934   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1924   0.1742   0.1633   0.1838
## Detection Prevalence   0.2851   0.1926   0.1752   0.1633   0.1838
## Balanced Accuracy      0.9996   0.9973   0.9990   0.9982   1.0000
```

```
# plot the results
plot(conMatRF$table, col=conMatRF$byClass, main = paste("Random Forest: Accuracy =",
                                                         round(conMatRF$overall['Accur
acy'],4)))
```



Decision Tree

```
# train the model
controlTree <- trainControl(method="cv", number=5)
modelTree <- train(classe ~ ., data = myTrain, method="rpart", trControl=controlTree)
rpart.plot(modelTree$finalModel)
```




```
# validate the model
predictTree<- predict(modelTree, newdata = myTest)
conMatDT <- confusionMatrix(myTest$classe, predictTree)
conMatDT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1765   31  150    0    7
##          B  531  459  338    0    0
##          C  554   47  596    0    0
##          D  501  196  428    0    0
##          E  178  170  334    0  580
##
## Overall Statistics
##
##                Accuracy : 0.4953
##                  95% CI : (0.4834, 0.5072)
##     No Information Rate : 0.5141
##     P-Value [Acc > NIR] : 0.9991
##
##                   Kappa : 0.3408
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5001  0.50831  0.32286       NA  0.98807
## Specificity            0.9436  0.85424  0.88026   0.8361  0.89137
## Pos Pred Value         0.9037  0.34563  0.49791       NA  0.45959
## Neg Pred Value         0.6409  0.91981  0.77946       NA  0.99875
## Prevalence             0.5141  0.13154  0.26890   0.0000  0.08551
## Detection Rate         0.2571  0.06686  0.08682   0.0000  0.08449
## Detection Prevalence   0.2845  0.19345  0.17436   0.1639  0.18383
## Balanced Accuracy      0.7219  0.68127  0.60156       NA  0.93972
```

Generalized Boosted Regression Model

```
# train the model
controlGBM<-trainControl(method="repeatedcv", number=5, repeats=1)
modelGBM<- train(classe ~ ., data=myTrain, method="gbm", trControl=controlGBM, verbos
e=FALSE)
modelGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

```
# validate the model
predictGBM <- predict(modelGBM, newdata = myTest)
conMatGBM<- confusionMatrix(predictGBM, myTest$classe)
conMatGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1948   18    0    0    0
##          B    4 1293    3    5    4
##          C    1   16 1190   17    3
##          D    0    0    4 1101    5
##          E    0    1    0    2 1250
##
## Overall Statistics
##
##                Accuracy : 0.9879
##                  95% CI : (0.985, 0.9904)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9847
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9974   0.9736   0.9942   0.9787   0.9905
## Specificity            0.9963   0.9971   0.9935   0.9984   0.9995
## Pos Pred Value         0.9908   0.9878   0.9698   0.9919   0.9976
## Neg Pred Value         0.9990   0.9937   0.9988   0.9958   0.9979
## Prevalence             0.2845   0.1934   0.1744   0.1639   0.1838
## Detection Rate         0.2838   0.1883   0.1733   0.1604   0.1821
## Detection Prevalence   0.2864   0.1907   0.1787   0.1617   0.1825
## Balanced Accuracy      0.9969   0.9854   0.9938   0.9885   0.9950
```

Results

After comparing the accuracy of all three models, we can see that although the Generalized Boosted Regression Model provides strong results, the Random Forest is the best option to use on the validation data.

```
conMatDT$overall[1]
```

```
##  Accuracy
## 0.4952658
```

```
conMatGBM$overall[1]
```

```
##  Accuracy
## 0.9879097
```

```
conMatRF$overall[1]
```

```
## Accuracy
## 0.998252
```

```
FinalPredicitons<- predict(modelFitRF, newdata = testData)
FinalPredicitons
```

```
##  [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
```