Machine Learning Project

Brandon Robinson

2024-06-10

Synopsis

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

Data

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

The test data is 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

Project Task

The goal of your project is to predict the manner in which the user did the exercise. This is the "classe" variable in the training set. We will train 3 models for this exercise: **Decision Tree** , **Random Forest**, and **Gradient Boosted Trees** using k-cross validation on the training set. A validation set will be used to predict the accuracy and out of sample error. The selected model will be used to predict 20 different test cases using a test csv.

Loading and Pre-Processing the Data

First the following packages will be loaded and the seed will be set:

```
library(ggplot2)
library(rattle)
```

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(caret)
```

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

```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(data.table)
library(randomForest)
```

```
## randomForest 4.7-1.1

## 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(rpart)
set.seed(1234)
```

The next step is to load the csv files:

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

The files were previously downloaded locally to the project folder.

Next, the data will be cleaned by removing NAs, irrelevant columns, and near non zero variables

```
inTrain <- createDataPartition(y=training$classe, p=0.7,list=F)
train <- training[inTrain, ]
validate <- training[-inTrain, ]
```

Removing unnecessary columns from the data table

```
trainsub <- subset(train, select = -c(X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_ti
validatesub <- subset(validate, select = -c(X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, c
```

Removing NAs

```
train_NA <- colSums(is.na(trainsub))/nrow(trainsub) < 0.9
train2 <- trainsub[,train_NA]
validate2 <- validatesub[,train_NA]
```

Removing non-zero variables

```
nvz <- nearZeroVar(train2)
dim(train2)
```

```
## [1] 13737    88
```

```
train_final <- train2[,-nvz]
validate_final <- validate2[,-nvz]
```

The final step of pre-processing is to change classe to a factor variable in both sets

```
train_final$classe <- as.factor(train_final$classe)
validate_final$classe <- as.factor(validate_final$classe)
```

Now that the data is cleaned / pre-processed we can begin the next of building the 3 models

Building the Models

Random forest and boosting are known to produce the most accurate models. Control will be a 3-fold cross validation

First model will be the random forest:

```
modelRF <- train(classe ~ ., data=train_final, method="rf", prox=TRUE, trControl = trainControl(method=
predictRF <- predict(modelRF, validate_final)
cmRF <- confusionMatrix(predictRF, factor(validate_final$classe))
print(cmRF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    3    0    0    0
##          B    0 1135    5    0    0
```

```
##          C    0    1 1021    4    0
##          D    0    0    0  959    0
##          E    0    0    0    1 1082
##
## Overall Statistics
##
##                Accuracy : 0.9976
##                  95% CI : (0.996, 0.9987)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.997
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9965   0.9951   0.9948   1.0000
## Specificity            0.9993   0.9989   0.9990   1.0000   0.9998
## Pos Pred Value         0.9982   0.9956   0.9951   1.0000   0.9991
## Neg Pred Value         1.0000   0.9992   0.9990   0.9990   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1929   0.1735   0.1630   0.1839
## Detection Prevalence   0.2850   0.1937   0.1743   0.1630   0.1840
## Balanced Accuracy      0.9996   0.9977   0.9970   0.9974   0.9999
```

The next model will be the boosted model:

```
modelGBM <- train(classe~., data=train_final, method="gbm", verbose = FALSE, trControl = trainControl(m
predictGBM <- predict(modelGBM, validate_final)
cmGBM <- confusionMatrix(predictGBM, factor(validate_final$classe))
print(cmGBM)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    6    0    3    0
##          B    1 1119    5    3    3
##          C    0   14 1017    7    0
##          D    0    0    4  950    7
##          E    0    0    0    1 1072
##
## Overall Statistics
##
##                Accuracy : 0.9908
##                  95% CI : (0.988, 0.9931)
##     No Information Rate : 0.2845
##     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            0.9994   0.9824   0.9912   0.9855   0.9908
## Specificity            0.9979   0.9975   0.9957   0.9978   0.9998
## Pos Pred Value         0.9946   0.9894   0.9798   0.9886   0.9991
## Neg Pred Value         0.9998   0.9958   0.9981   0.9972   0.9979
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1901   0.1728   0.1614   0.1822
## Detection Prevalence   0.2858   0.1922   0.1764   0.1633   0.1823
## Balanced Accuracy      0.9986   0.9900   0.9935   0.9916   0.9953
```

The final model will be the decision tree:

```
modelRPART <- train(classe~., data=train_final, method="rpart", trControl = trainControl(method="cv",nu
predictRPART <- predict(modelRPART, validate_final)
cmRPART <- confusionMatrix(predictRPART, factor(validate_final$classe))
print(cmRPART)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1519  473  484  451  156
##          B   28  401   45  167  148
##          C  123  265  497  346  289
##          D    0    0    0    0    0
##          E    4    0    0    0  489
##
## Overall Statistics
##
##                Accuracy : 0.4938
##                  95% CI : (0.4809, 0.5067)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.338
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9074  0.35206  0.48441   0.0000  0.45194
## Specificity            0.6286  0.91825  0.78946   1.0000  0.99917
## Pos Pred Value         0.4927  0.50824  0.32697      NaN  0.99189
## Neg Pred Value         0.9447  0.85518  0.87881   0.8362  0.89002
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2581  0.06814  0.08445   0.0000  0.08309
## Detection Prevalence   0.5239  0.13407  0.25828   0.0000  0.08377
## Balanced Accuracy      0.7680  0.63516  0.63693   0.5000  0.72555
```

The last step is to compare the accuracy and out of sample errors for the 3 models:

```
Accuracy <- rbind(cmRF$overall[1], cmGBM$overall[1], cmRPART$overall[1])
results <- data.frame(
        Model = c('RF', 'GBM', 'RPART'),
        Accuracy,
        oos_error = 1 - Accuracy
)
print(results)
```

```
##   Model  Accuracy  Accuracy.1
## 1    RF 0.9976211 0.002378929
## 2   GBM 0.9908241 0.009175871
## 3 RPART 0.4937978 0.506202209
```

Based on the above results, it can be concluded the random forest model produces the highest accuracy and lowest error rate. This model will be used for the test sets.

Testing of the model

The last step is to test the selected model against the test data set:

```
testResult <- predict(modelRF, testing)
print(testResult)
```

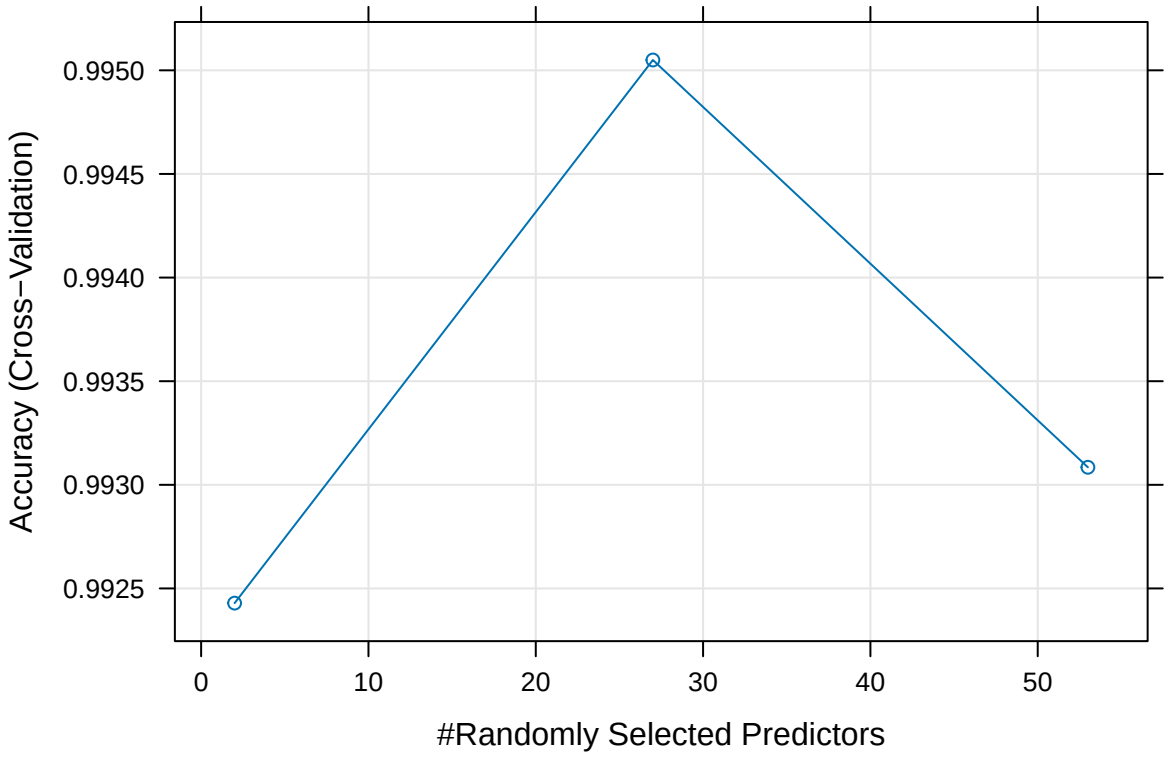
```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

Appendix

Below are the model plots:

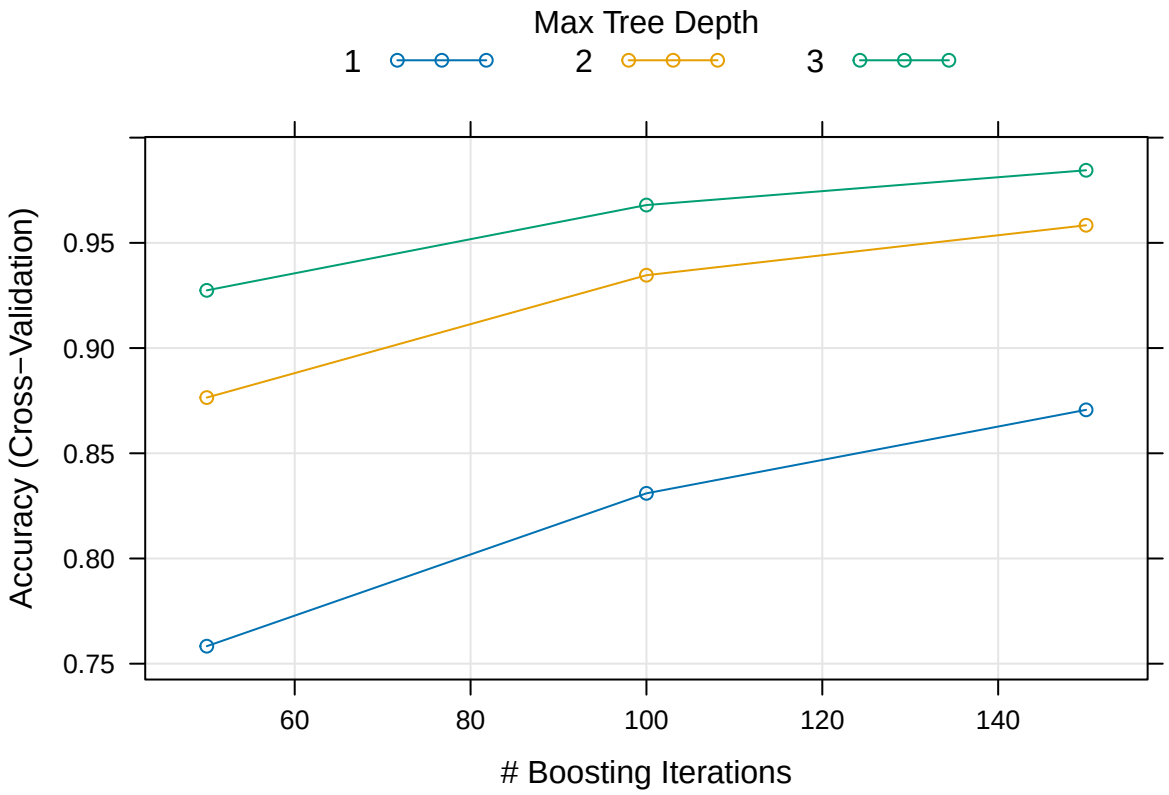
Random Forest plot

```
plot(modelRF)
```



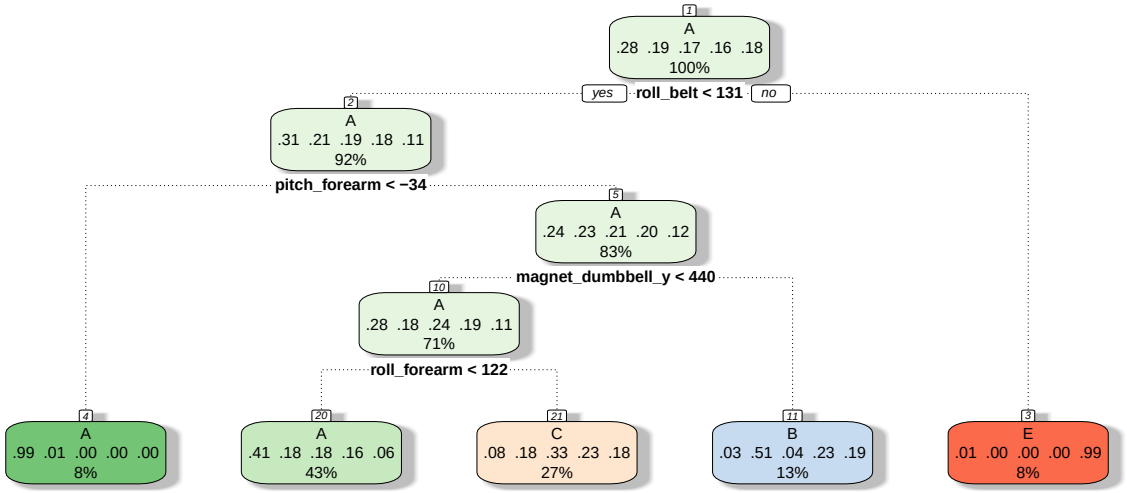
Boosted plot

```
plot(modelGBM)
```



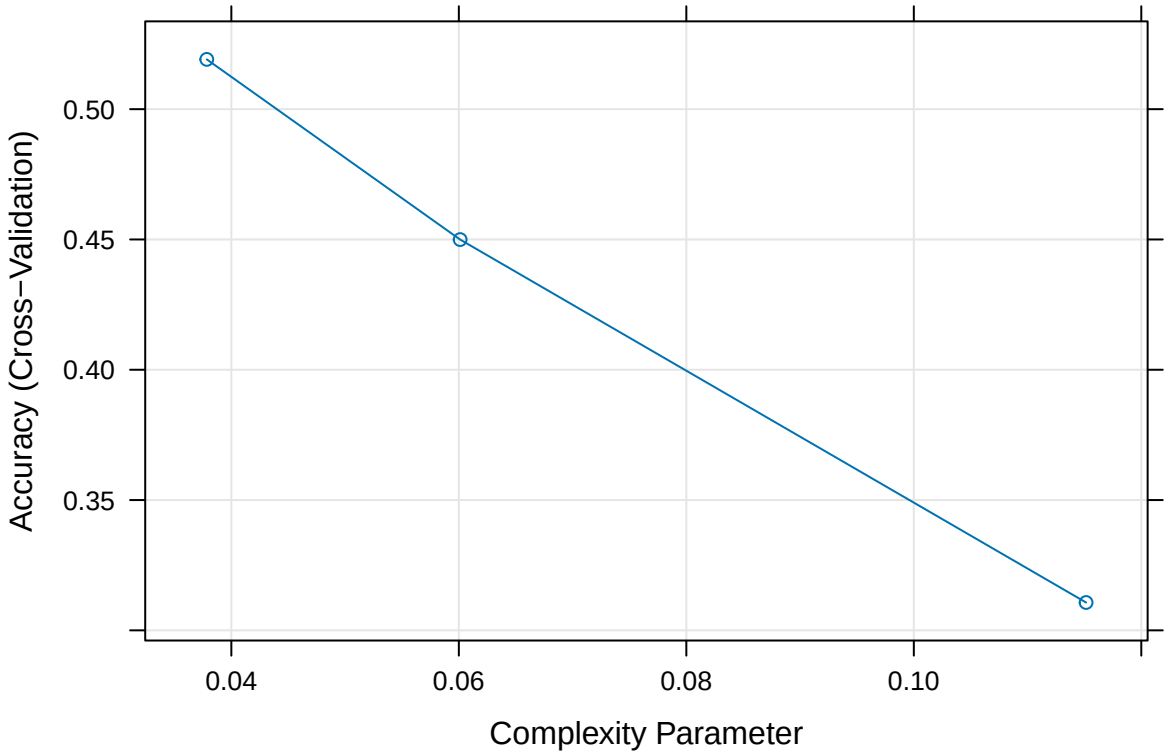
Decision Tree plot

```
fancyRpartPlot(modelRPART$finalModel)
```



Rattle 2024−Jun−10 09:15:05 brandon

```
plot(modelRPART)
```


Accuracy (Cross−Validation)
0.50
0.45
0.40
0.35
0.04
0.06
0.08
0.10
Complexity Parameter