Module 8 Final Project

2023-01-20

Project Overview

The goal of the project is to predict the manner in which people in the dataset did the exercise and chose the best model to use on the test data set.

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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har).

Set working directory, load packages, and load data

```
setwd("~/R/Coursera/module8-final")

library(ggplot2)
library(lattice)
library(caret)
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.
```

```r
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
```

```r
train_set <- read.csv("pml-training.csv")
test_set <- read.csv("pml-testing.csv")
```

Data cleaning

Since we are training the models, only the training set needs to be cleaned for now

```r
# Remove first few columns of data, since they aren't needed
train_set <- train_set[, -c(1:7)]

# Remove high correlation columns
train_set <- train_set[,colMeans(is.na(train_set)) < 0.95]

# Remove near zero variances
nzv <- nearZeroVar(train_set)
train_set <- train_set[,-nzv]
```

Split the training data into the validation and training data sets

```
new_train <- createDataPartition(train_set$classe, p = 0.8, list = FALSE)
training <- train_set[new_train,]
validation <- train_set[-new_train,]
```

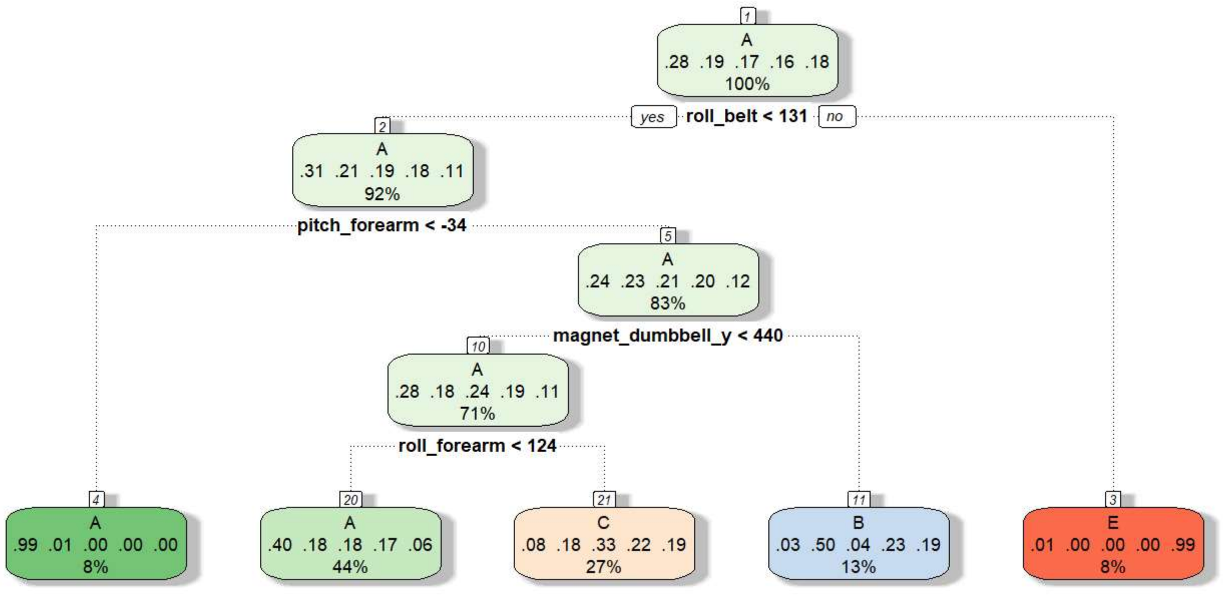
Create and Test the Models

We'll use a couple common models to test our predictions

Decision Tree

Model

```
tree <- train(classe~., data = training, method = "rpart")
fancyRpartPlot(tree$finalModel, sub = "Decision Tree")
```



Decision Tree

Prediction

```
tree_prediction <- predict(tree, validation)
confusionMatrix(tree_prediction, factor(validation$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1022  300  320  270  120
##          B   17  280   20   97   94
##          C   74  179  344  276  182
##          D    0    0    0    0    0
##          E    3    0    0    0  325
##
## Overall Statistics
##
##                Accuracy : 0.5024
##                  95% CI : (0.4867, 0.5182)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3499
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9158  0.36891  0.50292   0.0000  0.45076
## Specificity            0.6402  0.92794  0.78049   1.0000  0.99906
## Pos Pred Value         0.5030  0.55118  0.32607      NaN  0.99085
## Neg Pred Value         0.9503  0.85974  0.88145   0.8361  0.88985
## Prevalence             0.2845  0.19347  0.17436   0.1639  0.18379
## Detection Rate         0.2605  0.07137  0.08769   0.0000  0.08284
## Detection Prevalence   0.5180  0.12949  0.26893   0.0000  0.08361
## Balanced Accuracy      0.7780  0.64842  0.64171   0.5000  0.72491
```

Random Forest

```
control <- trainControl(method="cv", number=3, verboseIter=F)

forest <- train(classe~., data = training, method = "rf", trControl = control, tuneLength = 5)

forest_prediction <- predict(forest, validation)
confusionMatrix(forest_prediction, factor(validation$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    1    0    0    0
##          B    0  755    2    0    0
##          C    0    3  680   11    0
##          D    0    0    2  632    1
##          E    0    0    0    0  720
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9921, 0.9969)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9942   0.9829   0.9986
## Specificity            0.9996   0.9994   0.9957   0.9991   1.0000
## Pos Pred Value         0.9991   0.9974   0.9798   0.9953   1.0000
## Neg Pred Value         1.0000   0.9987   0.9988   0.9967   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1925   0.1733   0.1611   0.1835
## Detection Prevalence   0.2847   0.1930   0.1769   0.1619   0.1835
## Balanced Accuracy      0.9998   0.9970   0.9949   0.9910   0.9993
```

Between the two, the random forest is the better model, and is what we'll use in the test sets.

Test data predictions

```
test_pred <- predict(forest, test_set)
print(test_pred)
```

```
##  [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
```