Human Activity Recognition

Alex Merg

2/28/2021

```
library(knitr)
library(caret)
library(rattle)
library(ggcorrplot)
library(tidyverse)
```

Introduction 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://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

The test data are 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.

Purpose

The goal of the project is to predict the manner in which participants performed the excercise, which is the "classe" variable in the dataset. Any variable can be used to predict.

```
trainURL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testURL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

training_data <- read.csv(url(trainURL))
testing_data <- read.csv(url(testURL))

inTrain = createDataPartition(training_data$classe, p = 0.7, list = FALSE)
training = training_data[ inTrain,]
testing = training_data[-inTrain,]

dim(training)
```

Load and Setup Data

```
## [1] 13737   160
```

```
dim(testing)
```

```
## [1] 5885  160
```

Clean Data Remove variables with variances at or near zero

```
NZV <- nearZeroVar(training, saveMetrics = TRUE)
training <- training[, NZV$nzv == FALSE]
```

Remove first 5 columns of dataset, as they are merely ID related variables

```
training <- training[,-(1:5)]
```

Remove variables where greater than 70% of their observations are NAs

```
training <- training[, colSums(is.na(training)) < (length(training)* 0.7)]

dim(training)
```

```
## [1] 13737    54
```

Apply the same transformations to the testing sets

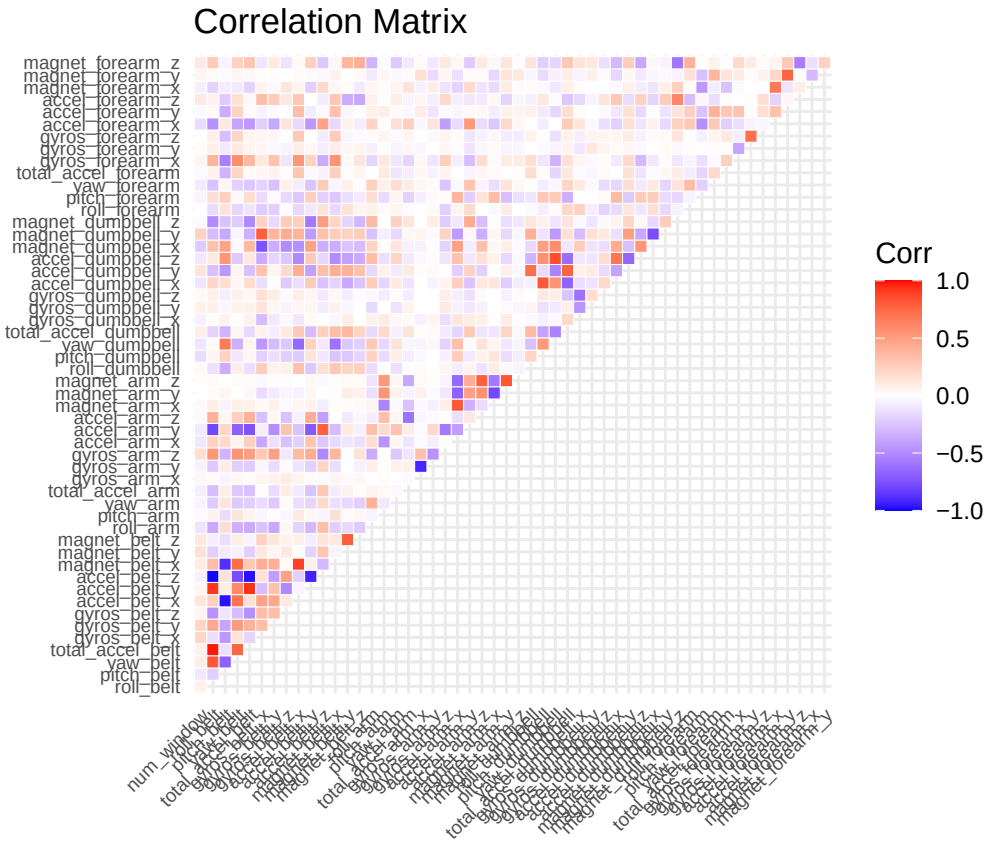
```
columns1 <- colnames(training)
columns2 <- colnames(training[, -54])

testing <- testing[columns1]
testing_data <- testing_data[columns2]

dim(testing)
```

```
## [1] 5885   54
```

```
dim(testing_data)
```

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

```
training %>%
  select(-54) %>%
  cor() %>%
  ggcorrplot(
    outline.color = "white",
    tl.cex = 6.5,
    type = "upper",
    insig = c("blank"),
    title = "Correlation Matrix"
    )
```



Correlation Analysis

As evidenced by the plot, there aren't a significant amount of correlated predictors, so PCA not necessarily needed.

Model Selection Will use two different modeling techniques - decision trees with CART and random forest - and determine which one has the best out-of-sample accuracy.

First, let's set up our cross validation parameters for each model where we'll use k = 3.

```
fitCV <- trainControl(method = "cv", number = 3)
```

Model Fitting

```
rpartModel <- train(classe ~ ., data = training, method = "rpart", trControl = fitCV)
rfModel <- train(classe ~ ., data = training, method = "rf", trControl = fitCV)
```

Model Assessment

```
rpartPred <- predict(rpartModel, newdata = testing)
rpartCM <- confusionMatrix(rpartPred, as.factor(testing$classe))
rfPred <- predict(rfModel, newdata = testing)
rfCM <- confusionMatrix(rfPred, as.factor(testing$classe))

accuracy <- data.frame(
  Model = c('CART', 'RF'),
  Accuracy = rbind(rpartCM$overall[1], rfCM$overall[1])
)

print(accuracy)
```

```
##   Model  Accuracy
```

```
## 1  CART 0.3631266
## 2    RF 0.9986406
```

As seen above, the random forest algorithm fits a more accurate model than classification (99.8%). Here's the full summary of results for the random forest model, which we'll use for the final prediction:

```
print(rfCM)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    0 1138    2    0    0
##          C    0    0 1024    4    0
##          D    0    0    0  960    0
##          E    1    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9986
##                  95% CI : (0.9973, 0.9994)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9983
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9991   0.9981   0.9959   1.0000
## Specificity            0.9998   0.9996   0.9992   1.0000   0.9998
## Pos Pred Value         0.9994   0.9982   0.9961   1.0000   0.9991
## Neg Pred Value         0.9998   0.9998   0.9996   0.9992   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1934   0.1740   0.1631   0.1839
## Detection Prevalence   0.2845   0.1937   0.1747   0.1631   0.1840
## Balanced Accuracy      0.9996   0.9994   0.9986   0.9979   0.9999
```

Prediction In order to predict the 20 quiz results, we'll use the random forest model applied to the original test data

```
final_prediction <- predict(rfModel, newdata = testing_data)

final_prediction
```

```
##  [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
```