Activity Recognition

Chelsi Snellen

December 15, 2019

Overview

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.

For the purpose of this exercise, the following steps will be followed:

- Data Processing
- Exploratory Data Analysis
- Model Selection
- Predicting on Test Set

Data Processing

The first thing that needs to be done is to download the data and them spilt the training data into both training and validation sets

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

# Read the data into data frames
training <- read.csv(url(trainingURL))
test <- read.csv(url(testURL))

# Now we need to split the training data into a training and validation set at a 70/30 split
parition <- createDataPartition(training$classe, p = 0.7, list = FALSE)
trainData <- training[parition, ]
validatData <- training[-parition, ]

# Lets look at how much of the data is NA
sum(is.na(trainData))
```

```
## [1] 902222
```

It looks like there are approximately 900,000 NA entries. Lets do something to reduce those numbers.

```
# Find the columns where the are more than 95 percent NA
naEnteries <- apply(trainData, 2, function(x) mean(is.na(x))) > 0.95

# Remove those columns
trainData <- trainData[, -which(naEnteries, naEnteries == FALSE)]
```

```r
validatData <- validatData[, -which(naEnteries, naEnteries == FALSE)]

# Calculate the leftover amount of NAs
sum(is.na(trainData))
```

```
## [1] 0
```

Perfect! Now the NAs have been reduced down to 0. This also reduced the number of variables to use from 160 to 93.

Now some of the variables also have near zero variance within them. This means that they would be less useful in classifying the different types of activities.

```r
# Find which variable have a near zero variance
NZV <- nearZeroVar(trainData)

# Remove those variables from the data sets
trainData <- trainData[ ,-NZV]
validatData <- validatData[ ,-NZV]
```

Now we know that each of the data sets contain variables that should have a chance of being a predictor for our outcome variable. Removing the varaibles with near zero variance has also reduced the number of predictors from 93 to 59. The last thing to do is remove the columns that contain identification information (The first five columns).

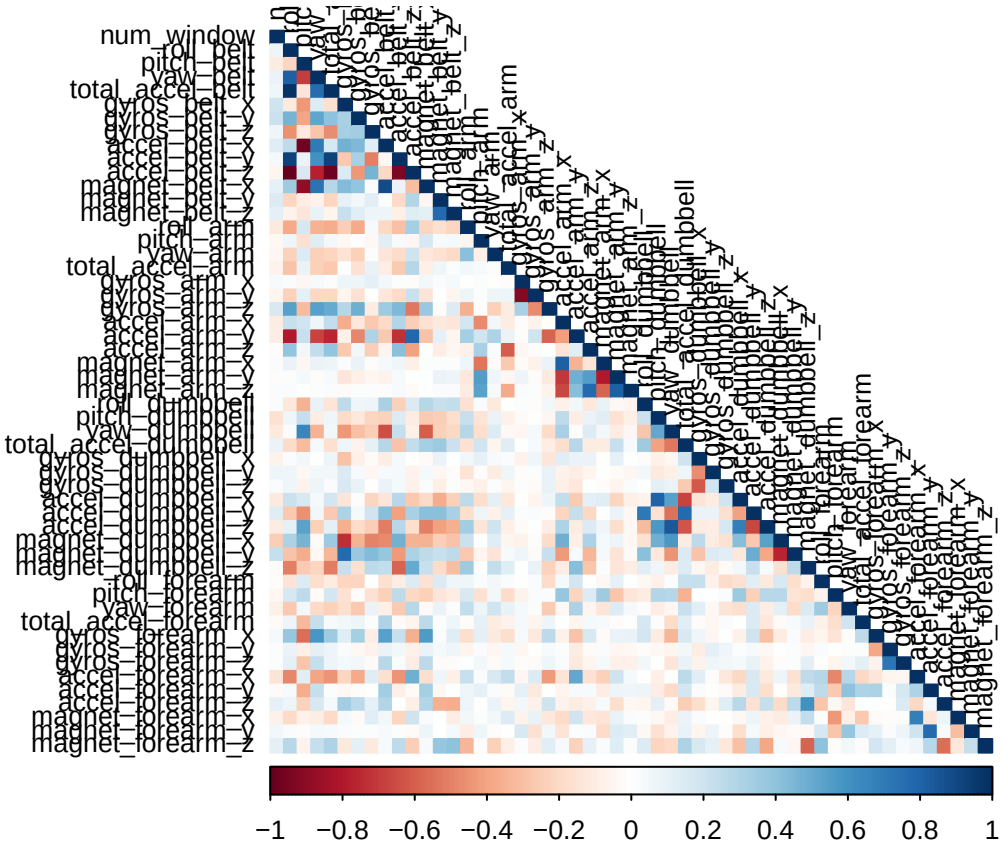
```r
# Remove the first five columns that only contain identification information
trainData <- trainData[ , -(1:5)]
validatData <- validatData[ , -(1:5)]
```

Through all the data processing, we have been able to drastically reduce the number of variable from 160 to 54. This will prove useful down the line when the model needs to predict the activity.

Exploratory Data Analysis

Now that we have cleaned the dataof variables that didn't contain useful information, lets see if we can reduce our variables more by seeing if there are any variables that are heavily correlated with each other.

```r
# Create the correlation matrix and then plot it
corrMat <- cor(trainData[,-54])
corrplot(corrMat, method = "color", type = "lower", tl.cex = 0.8, tl.col = rgb(0,0,0))
```



For this style of plot, the darker the color (either red or blue for negative and postive correlation respectively) the more highly correlated the variables are. It seems that alot of the data cleaning we did reduced alot of the highly correlated variables. So this set of variables should be go to use for prediction.

Model Selection

For this problem set up, we are going to try three different models: Random Forests, Decision Trees and Generalized Boosted Model. We will choose the final model that performs the best on the validation set.

Random Forest

```
library(caret)
set.seed(13908)
control <- trainControl(method = "cv", number = 3, verboseIter=FALSE)
RFModel <- train(classe ~ ., data = trainData, method = "rf", trControl = control)
RFModel$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: 0.21%
## Confusion matrix:
##      A    B    C    D    E class.error
```

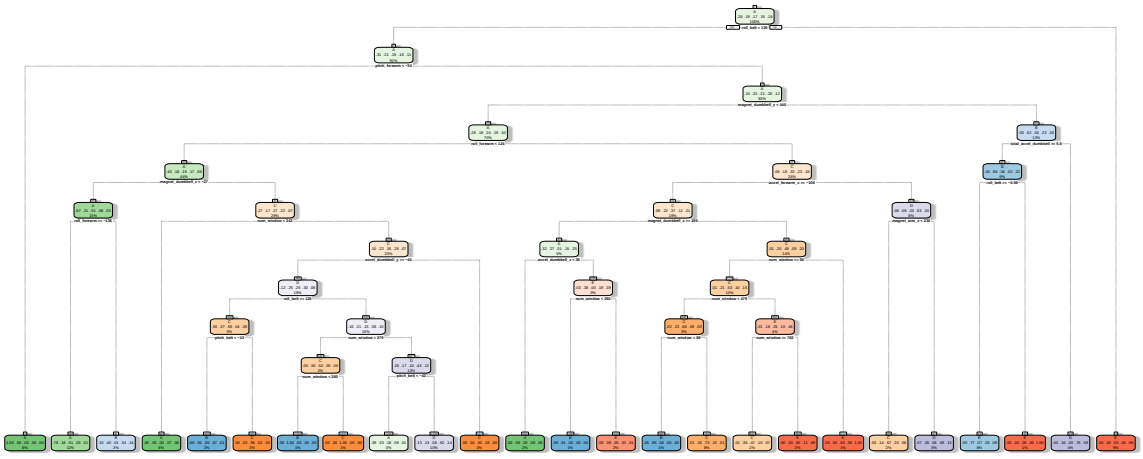
```
## A 3906    0    0    0    0 0.000000000
## B    4 2652    1    1    0 0.002257336
## C    0    6 2388    2    0 0.003338898
## D    0    0    7 2244    1 0.003552398
## E    0    0    0    7 2518 0.002772277
```

```
predictRF <- predict(RFModel, validatData)
confMatRF <- confusionMatrix(predictRF, validatData$classe)
confMatRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    1    0    0    0
##          B    1 1137    2    0    3
##          C    0    1 1024    2    0
##          D    0    0    0  962    1
##          E    1    0    0    0 1078
##
## Overall Statistics
##
##                Accuracy : 0.998
##                  95% CI : (0.9964, 0.9989)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9974
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9982   0.9981   0.9979   0.9963
## Specificity            0.9998   0.9987   0.9994   0.9998   0.9998
## Pos Pred Value         0.9994   0.9948   0.9971   0.9990   0.9991
## Neg Pred Value         0.9995   0.9996   0.9996   0.9996   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1932   0.1740   0.1635   0.1832
## Detection Prevalence   0.2843   0.1942   0.1745   0.1636   0.1833
## Balanced Accuracy      0.9993   0.9985   0.9987   0.9989   0.9980
```

Decision Tree

```
set.seed(13908)
modelDT <- rpart(classe ~ ., data = trainData, method = "class")
fancyRpartPlot(modelDT)
```



Rattle 2019-Dec-15 21:17:45 cache

```
predictDT <- predict(modelDT, validatData, type = "class")
confMatDT <- confusionMatrix(predictDT, validatData$classe)
confMatDT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1486  200   46   56   40
##          B   49  638   34   64   62
##          C   14  160  878  106   26
##          D   95  128   67  681  142
##          E   30   13    1   57  812
##
## Overall Statistics
##
##                Accuracy : 0.7638
##                  95% CI : (0.7527, 0.7746)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7007
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8877   0.5601   0.8558   0.7064   0.7505
## Specificity            0.9188   0.9560   0.9370   0.9122   0.9790
## Pos Pred Value         0.8129   0.7532   0.7416   0.6119   0.8894
## Neg Pred Value         0.9537   0.9006   0.9685   0.9407   0.9457
```

```
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2525   0.1084   0.1492   0.1157   0.1380
## Detection Prevalence   0.3106   0.1439   0.2012   0.1891   0.1551
## Balanced Accuracy      0.9032   0.7581   0.8964   0.8093   0.8647
```

Generalized Boosted Model

```
set.seed(13908)
control <- trainControl(method = "repeatedcv", number = 5, repeats = 1, verboseIter = FALSE)
modelGBM <- train(classe ~ ., data = trainData, trControl = control, method = "gbm", verbose = 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.
```

```
predictGBM <- predict(modelGBM, validatData)
confMatGBM <- confusionMatrix(predictGBM, validatData$classe)
confMatGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    7    0    1    0
##          B    3 1123   10    3    5
##          C    0    9 1013   11    3
##          D    0    0    2  949   11
##          E    0    0    1    0 1063
##
## Overall Statistics
##
##                Accuracy : 0.9888
##                  95% CI : (0.9858, 0.9913)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9858
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9860   0.9873   0.9844   0.9824
## Specificity            0.9981   0.9956   0.9953   0.9974   0.9998
## Pos Pred Value         0.9952   0.9816   0.9778   0.9865   0.9991
## Neg Pred Value         0.9993   0.9966   0.9973   0.9970   0.9961
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1908   0.1721   0.1613   0.1806
## Detection Prevalence   0.2853   0.1944   0.1760   0.1635   0.1808
## Balanced Accuracy      0.9982   0.9908   0.9913   0.9909   0.9911
```

The Random Forest model ended up with the highest accuracy so we will go with that one for the final test set prediction

Test Set Prediction

```
predictRF <- predict(RFModel, test)
predictRF
```

```
##  [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
```