Predicting How Well Weight Lifting Exercises Are Performed

Brian Buckley Sunday, January 25 2015

Objective

The purpose of this exercise is to use the weight lifting data set of the Human Activity Recognition program from Groupware. In these data, 6 participants performed a weight lifting exercise in 5 different ways whereby one of the five was the correct technique. Data from a set of accelerometers was collected and a machine learning algorithm used to predict if the user was performing the weight lifting correctly.

Our objective is to use the "classe" variable in the training set to predict the manner in which Groupware did the exercise.

Data

We loaded the data from the two links provided.

```r
library(RCurl)
```

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

```r
data1 <- getURL("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
    ssl.verifypeer = 0L, followlocation = 1L)
writeLines(data1, "./data/pml-training.csv")
data2 <- getURL("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
    ssl.verifypeer = 0L, followlocation = 1L)
writeLines(data2, "./data/pml-testing.csv")
dateDownloaded <- date()
```

When we look at the data using str() we see that it has 19622 observations on 160 variables. We also note a lot of NA and empty columns plus some irrelevant variables like user name, timestamp, etc. So first we cleaned it up by removing all NAs

```r
training1 <- read.csv("./data/pml-training.csv", na.strings = c("NA", "", " "))
training2 <- apply(training1, 2, function(x) {
    sum(is.na(x))
})
training3 <- training1[, which(training2 == 0)]
```

Next we removed identifier columns such as name, timestamps etc

```r
training3 <- training3[8:length(training3)]
```

After the above data cleansing we were left with 53 variables out of the original 160.

Cross Validation Set

We allocated 30% of the traing data for cross validation to test the accuracy of our model.

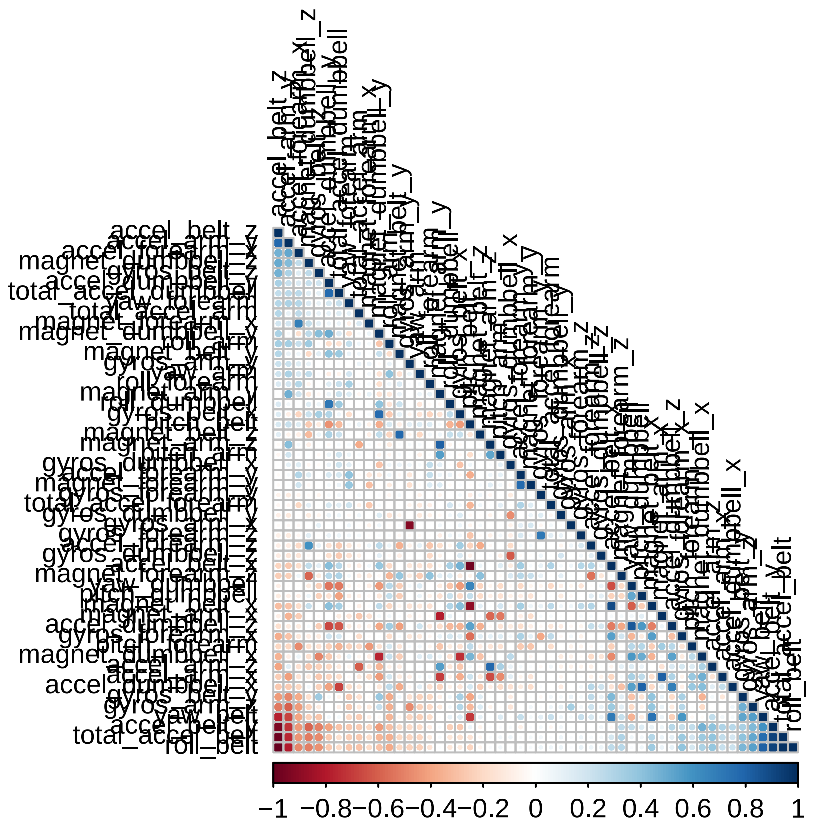
```r
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```r
inTrain <- createDataPartition(y = training3$classe, p = 0.7, list = FALSE)
training <- training3[inTrain, ]
crossval <- training3[-inTrain, ]
```

As we still have a lot of variables (53) we used a correlation matrix to identify if there are any highly correlated variables in the data. Highly correlated variables can be removed thereby improving the efficiency of our process and reducing overfitting.

```r
library(corrplot)
corM <- cor(training[, -length(training)])
corrplot(corM, order = "FPC", method = "circle", type = "lower", tl.cex = 0.8,
    tl.col = rgb(0, 0, 0))
```



The correlation plot identifies correlation by the strength of the color. As can be seen in the plot it looks like we do have quite a few correlated variables.

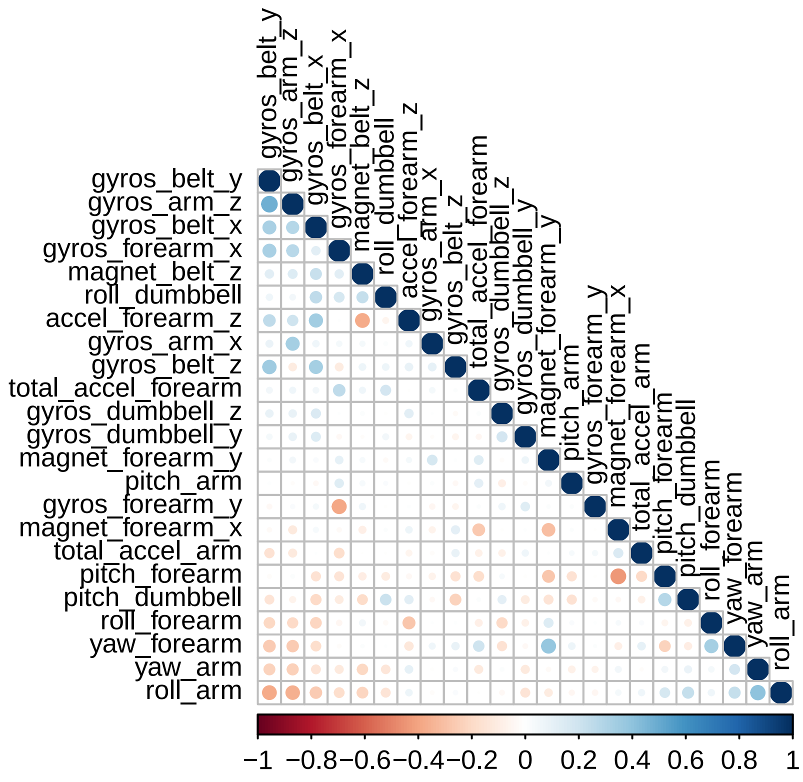
We used a correlation coefficient r > 0.5 to remove highly correlated variables.

```
highCorr <- findCorrelation(corM, cutoff = 0.5)
trainingCor <- training[, -highCorr]
ncol(trainingCor)
```

```
## [1] 24
```

Our training data now has 24 non-correlated variables from the original 53. If we plot a correlation matrix we confirm the data is not highly correlated.

```
corM2 <- cor(trainingCor[, -length(trainingCor)])
corrplot(corM2, order = "FPC", method = "circle", type = "lower", tl.cex = 0.8,
    tl.col = rgb(0, 0, 0))
```



Model Fitting

We fit a random forest model to predict the 'classe' variable using everything else as a predictor. We used random forest as that is stated to be the most accurate in our lecture notes.

```
library(caret)
library(kernlab)
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
model <- randomForest(classe ~ ., data = trainingCor)
```

Model Accuracy using the Cross Validation Data

Now we want to test the accuracy of our model so we crossvalidate the model using the remaining 30% of data.

```
predictCrossVal <- predict(model, crossval)
confusionMatrix(crossval$classe, predictCrossVal)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1662    3    1    3    5
##          B    8 1124    4    1    2
##          C    2   11 1005    8    0
##          D    0    1   29  932    2
##          E    0    0    1    2 1079
##
## Overall Statistics
##
##                Accuracy : 0.9859
##                  95% CI : (0.9825, 0.9888)
##     No Information Rate : 0.2841
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9822
##  Mcnemar's Test P-Value : 0.001347
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9940   0.9868   0.9663   0.9852   0.9917
## Specificity            0.9972   0.9968   0.9957   0.9935   0.9994
## Pos Pred Value         0.9928   0.9868   0.9795   0.9668   0.9972
## Neg Pred Value         0.9976   0.9968   0.9928   0.9972   0.9981
## Prevalence             0.2841   0.1935   0.1767   0.1607   0.1849
## Detection Rate         0.2824   0.1910   0.1708   0.1584   0.1833
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9956   0.9918   0.9810   0.9894   0.9956
```

The accuracy of our model is tested by comparing the predicted values from our model to the actual values in the CV data.

```
accuracy <- sum((predictCrossVal == crossval$classe))/dim(crossval)[1]
```

Our model accuracy is 98.59%.

We also want to know the out-of-sample error. This is the complement of accuracy.

```
1 - accuracy
```

```
## [1] 0.01410365
```

In our case the out-of-sample error is 1.4%.

Use the Testing Data to Predict Weight Lifting Performance

We then apply the same data cleaning treatment to the final testing data.

```
test1 <- read.csv("./data/pml-testing.csv", na.strings = c("NA", "", " "))
test2 <- apply(test1, 2, function(x) {
    sum(is.na(x))
})
test3 <- test1[, which(test2 == 0)]
test3 <- test3[8:length(test3)]
```

Now we predict the classes of the test set on the cleaned testing set.

```
predictTest <- predict(model, test3)
print(predictTest)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

Save the predicted results in the 20 files for Coursera grading submission.

```
pml_write_files = function(x) {
    n = length(x)
    for (i in 1:n) {
        filename = paste0("problem_id_", i, ".txt")
        write.table(x[i], file = filename, quote = FALSE, row.names = FALSE,
            col.names = FALSE)
    }
}
pml_write_files(predictTest)
```

Conclusion

We used a random forest prediction model on cleaned and uncorrelated data from Groupware. Our model had an accuracy of 98.59% with an out-of-sample error of 1.4%. When applied to the testing data set our submitted prediction results to Coursera were all correctly identified.