Prediction Assignment Writeup

K Ajay Rangan

10/13/2020

Aim

To use data from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants and predict the quality of the exercie for each instance.

Dataset

The training data: (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv) The test data : (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

A special thanks to (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) for providing data for this project.

The dataset contains data of 6 participants asked to perfrom various exercises using dumbbels in a correct manner and wrong manner.

Libraries

```
library(knitr)
library(caret)
```

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

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

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

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

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(ggplot2)
set.seed(4592)
```

Loading the data

```
traindata <- read.csv(url("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"))
testdata <- read.csv(url("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"))

inTrain <- createDataPartition(traindata$classe, p=0.7, list=FALSE)
Trainingset <- traindata[inTrain,]
Testingset <- traindata[-inTrain,]
```

Lets Have a look at the data

```
dim(Trainingset)
```

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

```
dim(Testingset)
```

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

Lets remove variables with low variance, variables that are more than 90% NA and Identification Variables which are the first 5 columns.

```
NZV <- nearZeroVar(Trainingset)
Trainingset <- Trainingset[, -NZV]
Testingset  <- Testingset[, -NZV]
AllNA    <- sapply(Trainingset, function(x) mean(is.na(x))) > 0.90
Trainingset <- Trainingset[, AllNA==FALSE]
Testingset  <- Testingset[, AllNA==FALSE]
Trainingset <- Trainingset[, -(1:5)]
Testingset  <- Testingset[, -(1:5)]
dim(Trainingset)
```

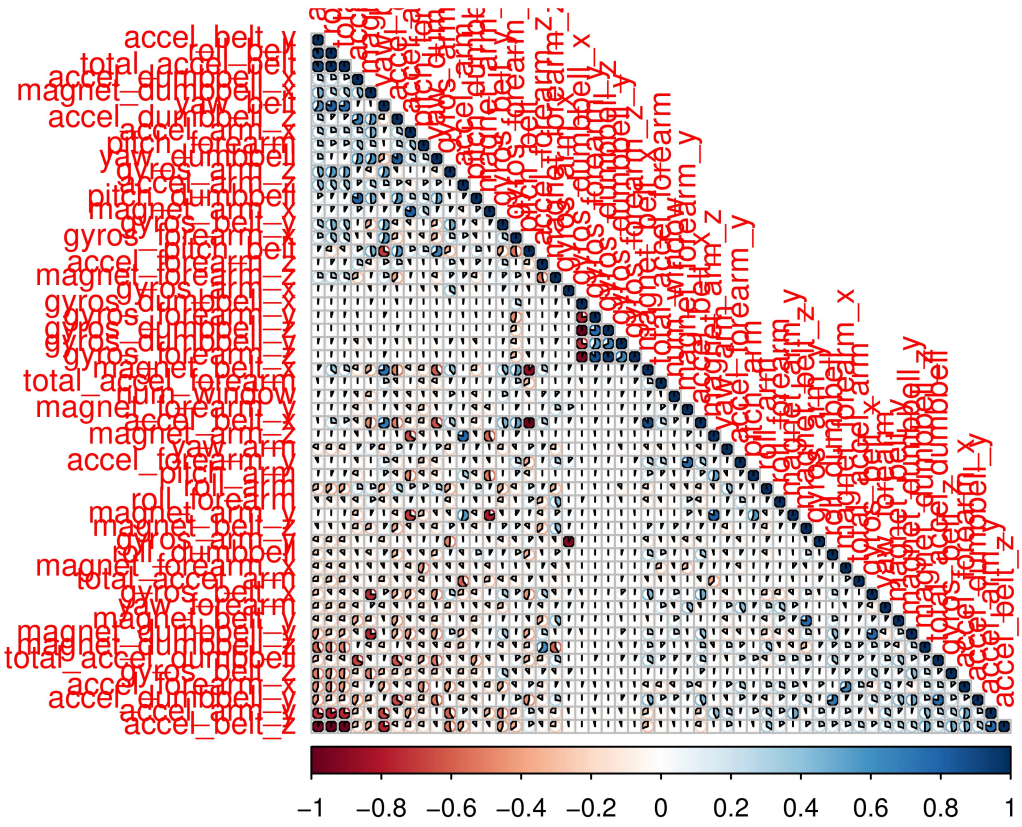
```
## [1] 13737    54
```

```
dim(Testingset)
```

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

Now lets check if we need to remove or modify variables due to high correlation

```
corM <- cor(Trainingset[, -54])
corrplot(corM, order = "FPC", method = "pie", type = "lower")
```



As we can see there are hardly any variables with high correlation but for few stray pair of variables. PCA may not be required. Even if we perform PCA the result will not be too different and it'll only increase runtime of the code.

Lets create the Model

I'm going to use a Generalized Boosting method (GBM)to create a model.

```
control <- trainControl(method="repeatedcv", number=4, repeats = 1)
model <- train(classe ~ ., data=Trainingset, method="gbm",trControl=control, verbose = FALSE)
model$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

Result using the Testing data

```
predicted <- predict(model, newdata=Testingset)
confusion <- confusionMatrix(predicted, Testingset$classe)
confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1668    8    0    0    0
##          B    6 1119    9    5    9
##          C    0   12 1016   13    3
##          D    0    0    1  945   17
##          E    0    0    0    1 1053
##
## Overall Statistics
##
##                Accuracy : 0.9857
##                  95% CI : (0.9824, 0.9886)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9819
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9824   0.9903   0.9803   0.9732
## Specificity            0.9981   0.9939   0.9942   0.9963   0.9998
## Pos Pred Value         0.9952   0.9747   0.9732   0.9813   0.9991
## Neg Pred Value         0.9986   0.9958   0.9979   0.9961   0.9940
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2834   0.1901   0.1726   0.1606   0.1789
## Detection Prevalence   0.2848   0.1951   0.1774   0.1636   0.1791
## Balanced Accuracy      0.9973   0.9882   0.9922   0.9883   0.9865
```

Conclusion

The Model had an accuracy of 98.57% in the testing data.