Accelerometers Data Analysis

Archit Dasgupta

10/20/2020

Brief Summary of the project

Using various devices nowadays that contain accelerometers we can get accurate telemetry and data on how people workout.With this project we want to use data of 6 participants from accelerometers attached to various parts of their body and equipments while they workout.

##Preprocessing of Data

Loading training and testing sets via the datasets given to us and then we divide the training set furthur into training and testing sets.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

```
setwd("C:/Users/MAHE/Documents/Archit/R progs/Practical-Machine-Learning")
trainURL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testURL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
trainin <- read.csv(url(trainURL))
testin <- read.csv(url(testURL))
label<- createDataPartition(trainin$classe, p = 0.7, list = FALSE)
trainb <- trainin[label, ]
testb <- trainin[-label, ]
```

We now remove NA variable and also the variables that dont have any variance (0 variance) from the data set

```
NonZero <- nearZeroVar(trainb)
trainb <- trainb[ ,-NonZero]
testb <- testb[ ,-NonZero]
label<- apply(trainb, 2, function(x) mean(is.na(x))) > 0.95
trainb <- trainb[, -which(label, label== FALSE)]
testb <- testb[, -which(label, label == FALSE)]
trainb <- trainb[ , -(1:5)]
testb <- testb[ , -(1:5)]
```

Now we can operate with 54 meaningful variables.

Coorelation Plot via Exploratory analysis

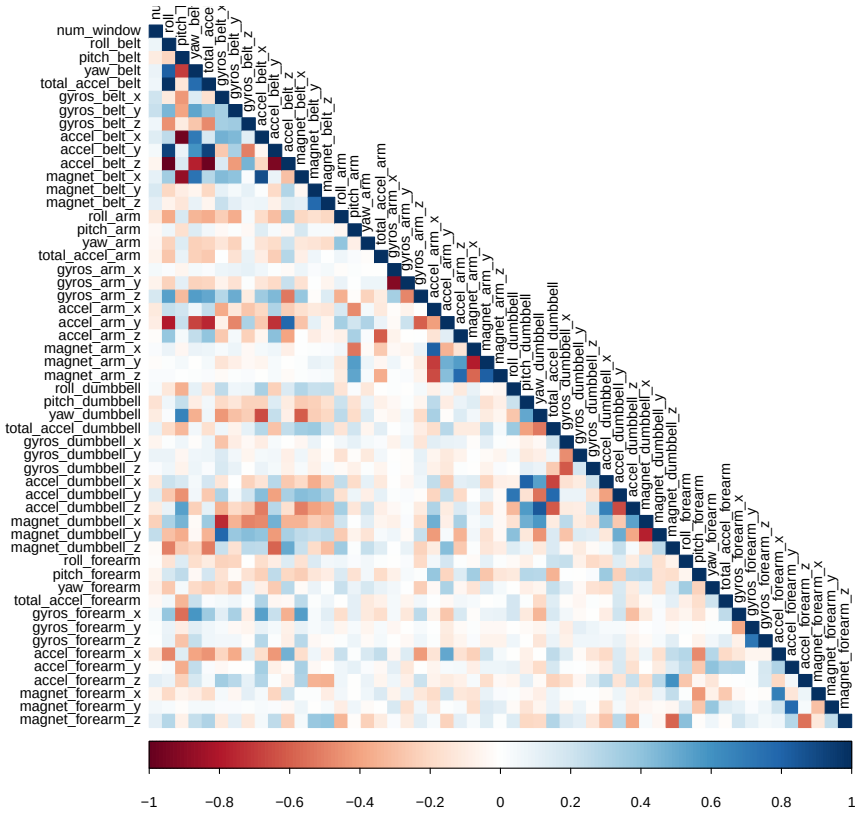
We now look at the dependence of these variables on each other using a correlation plot.

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.6.3
```

```
## corrplot 0.84 loaded
```

```
corrMat <- cor(trainb[,-54])
corrplot(corrMat, method = "color", type = "lower", tl.cex = 0.8, tl.col = rgb(0,0,0))
```



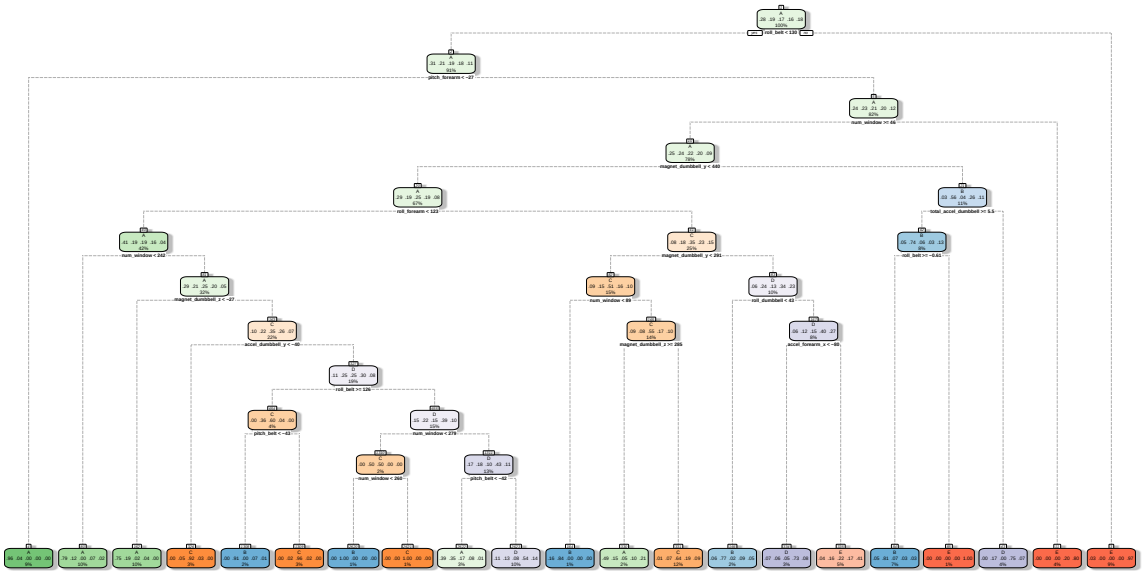
From the plot above we can now see that the darker area has more ccorelation than others.

Selecting Prediction Models

We use Decision Tree, Random Forest and Generalized Boosted model, methods to model our training set and hence select the one with the most accuracy to predict the output variables in the testing set. For better visualzation a confusion matrix is also plotted below.

Method 1: Decision Tree

```
library(rpart)
library(rpart.plot)
library(rattle)
set.seed(13908)
modelD <- rpart(classe ~ ., data = trainb, method = "class")
fancyRpartPlot(modelD)
```



Rattle 2020−Oct−21 00:50:37 MAHE

```
predictD <- predict(modelD, testb, type = "class")
confMatrD <- confusionMatrix(predictD, testb$classe)
confMatrD
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1490  275   44   95   46
##          B   44  665   39   24   25
##          C    5   54  819  135   61
##          D   90  102   60  625  139
##          E   45   43   64   85  811
##
## Overall Statistics
##
##                Accuracy : 0.7494
##                  95% CI : (0.7381, 0.7604)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6814
##
```

```
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8901   0.5838   0.7982   0.6483   0.7495
## Specificity            0.8908   0.9722   0.9475   0.9205   0.9507
## Pos Pred Value         0.7641   0.8344   0.7626   0.6152   0.7739
## Neg Pred Value         0.9532   0.9068   0.9570   0.9304   0.9440
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2532   0.1130   0.1392   0.1062   0.1378
## Detection Prevalence   0.3314   0.1354   0.1825   0.1726   0.1781
## Balanced Accuracy      0.8904   0.7780   0.8729   0.7844   0.8501
```

Method 2:Random Forest

```
library(caret)
set.seed(13908)
control <- trainControl(method = "cv", number = 3, verboseIter=FALSE)
modelR <- train(classe ~ ., data = trainb, method = "rf", trControl = control)
modelR$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: 0.26%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3906    0    0    0    0 0.000000000
## B    8 2647    3    0    0 0.004138450
## C    0    5 2391    0    0 0.002086811
## D    0    0   13 2239    0 0.005772647
## E    0    1    0    6 2518 0.002772277
```

```
predictR <- predict(modelR, testb)
confMatrR <- confusionMatrix(predictR, testb$classe)
confMatrR
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    2    0    0    0
##          B    0 1131    5    0    0
##          C    0    6 1021    4    0
##          D    0    0    0  959    6
##          E    1    0    0    1 1076
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9958
##                  95% CI : (0.9937, 0.9972)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9946
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9930   0.9951   0.9948   0.9945
## Specificity            0.9995   0.9989   0.9979   0.9988   0.9996
## Pos Pred Value         0.9988   0.9956   0.9903   0.9938   0.9981
## Neg Pred Value         0.9998   0.9983   0.9990   0.9990   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1922   0.1735   0.1630   0.1828
## Detection Prevalence   0.2846   0.1930   0.1752   0.1640   0.1832
## Balanced Accuracy      0.9995   0.9960   0.9965   0.9968   0.9970
```

Method 3: Generalized Boosted Model

```
library(caret)
set.seed(13908)
control <- trainControl(method = "repeatedcv", number = 5, repeats = 1, verboseIter = FALSE)
modelGB <- train(classe ~ ., data = trainb, trControl = control, method = "gbm", verbose = FALSE)
modelGB$finalModel
```

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

```
predictGB <- predict(modelGB, testb)
confMatrGB <- confusionMatrix(predictGB, testb$classe)
confMatrGB
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1667    8    0    0    0
##          B    7 1110   11    3    8
##          C    0   20 1013   12    3
##          D    0    1    2  949   16
##          E    0    0    0    0 1055
##
## Overall Statistics
```

```
##
##                Accuracy : 0.9845
##                  95% CI : (0.981, 0.9875)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9804
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9958   0.9745   0.9873   0.9844   0.9750
## Specificity            0.9981   0.9939   0.9928   0.9961   1.0000
## Pos Pred Value         0.9952   0.9745   0.9666   0.9804   1.0000
## Neg Pred Value         0.9983   0.9939   0.9973   0.9969   0.9944
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2833   0.1886   0.1721   0.1613   0.1793
## Detection Prevalence   0.2846   0.1935   0.1781   0.1645   0.1793
## Balanced Accuracy      0.9970   0.9842   0.9901   0.9903   0.9875
```

We see the method 2 has the highest accuracy hence Random forest is chosen.

Prediction of Test set outputs

```
predictR <- predict(modelR, testin)
predictR
```

```
##  [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
```