Predictions using the Weight Lifting Dataset

Jatin chawda

2/24/2020

Summary

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har.

training data > https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

testing data > https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv Based on a dataset provide by HAR http://groupware.les.inf.puc-rio.br/har we will try to train a predictive model to predict what exercise was performed using a dataset with 159 features

We'll take the following steps:

1. Process the data
2. Explore the data
3. Model selection
4. Test results

Process the data

```r
trainRaw <- read.csv("data.csv")

testRaw <- read.csv("validation.csv")
```

Explore the data

```r
dim(trainRaw)

dim(testRaw)

sum(complete.cases(trainRaw))
```

We Have Seen Lots of NA Values Lets Remove those

```r
trainRaw <- trainRaw[, colSums(is.na(trainRaw)) == 0]
testRaw <- testRaw[, colSums(is.na(testRaw)) == 0]
```

Also remove all time related data, since we won't use those Then convert all factors to integers

```r
classe <- trainRaw$classe
trainRemove <- grepl("^X|timestamp|window", names(trainRaw))
trainRaw <- trainRaw[, !trainRemove]
trainCleaned <- trainRaw[, sapply(trainRaw, is.numeric)]
trainCleaned$classe <- classe
testRemove <- grepl("^X|timestamp|window", names(testRaw))
testRaw <- testRaw[, !testRemove]
testCleaned <- testRaw[, sapply(testRaw, is.numeric)]
```

Now We Will Do Some Expolatory Data Analysis

Since the test set provided is the the ultimate validation set, we will split the current training in a test and train set to work with.

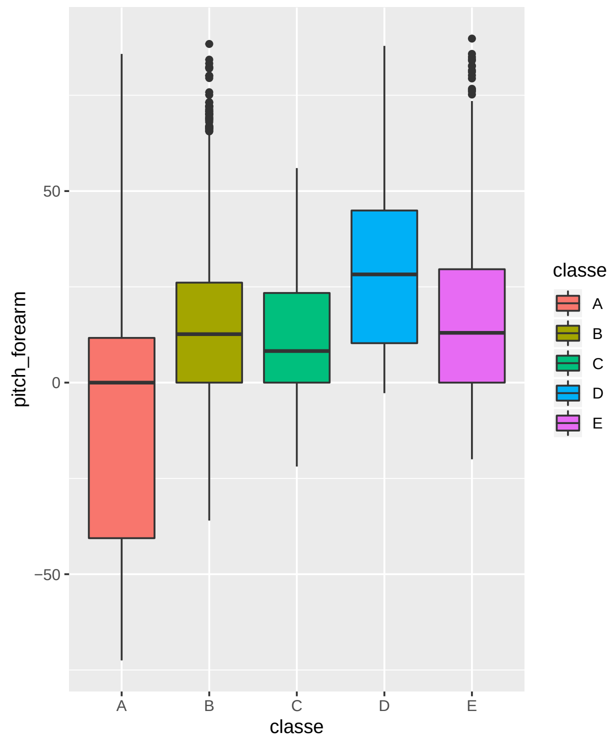
```r
set.seed(22519)
inTrain <- createDataPartition(trainCleaned$classe, p=0.70, list=F)
trainData <- trainCleaned[inTrain, ]
testData <- trainCleaned[-inTrain, ]
```

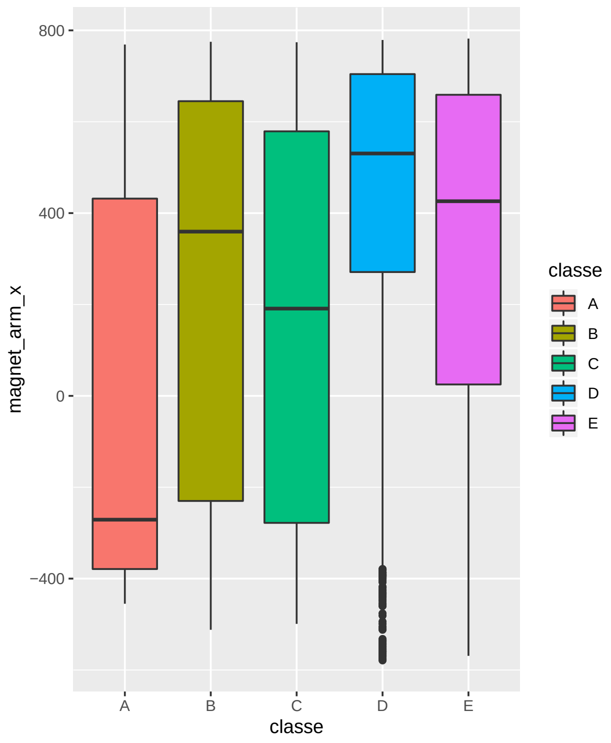
Let's check visually if there is indeed hard to use these 2 as possible simple linear predictors.

```r
library(Rmisc)
library(ggplot2)
p1 <- ggplot(trainData, aes(classe,pitch_forearm)) +
  geom_boxplot(aes(fill=classe))

p2 <- ggplot(trainData, aes(classe, magnet_arm_x)) +
  geom_boxplot(aes(fill=classe))

multiplot(p1,p2,cols=2)
```





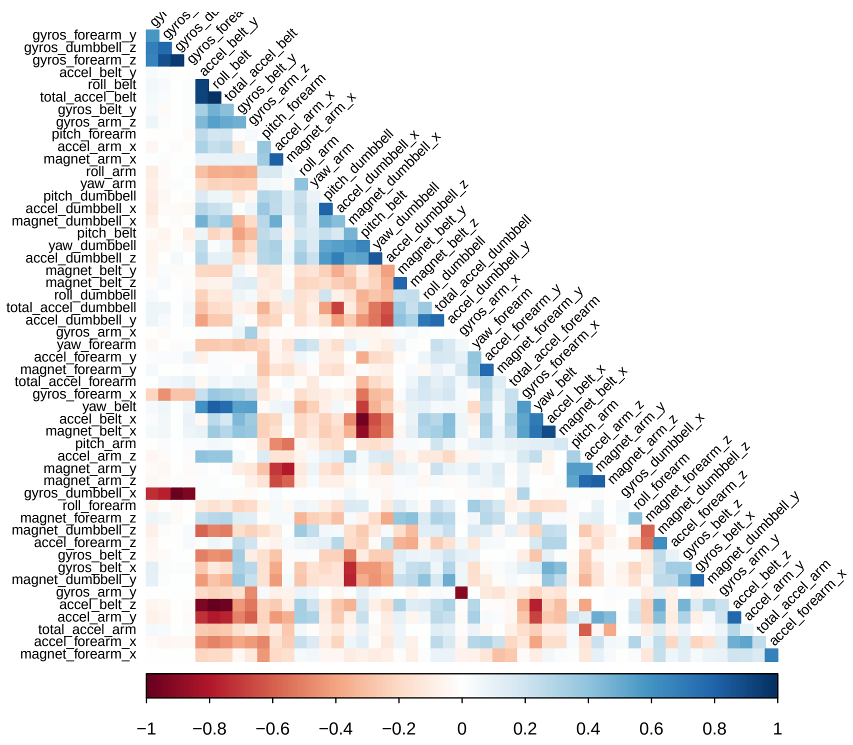
Clearly there is no hard seperation of classes possible using only these 'highly' correlated features. Let's train some models to get closer to a way of predicting these classe's

Model selection

Let's identify variables with high correlations amongst each other in our set, so we can possibly exclude them from the pca or training.

We will check afterwards if these modifications to the dataset make the model more accurate (and perhaps even faster)

```
corrPlot <- cor(trainData[, -length(names(trainData))])
corrplot(corrPlot, method="color", type="lower", order="hclust", tl.cex=0.70, tl.col="black", tl.srt = 4
```



We see that there are some features that aree quite correlated with each other.

Now we'll do some actual Random Forest training. We fit a predictive model for activity recognition using Random Forest algorithm because it automatically selects important variables and is robust to correlated covariates & outliers in general. We will use 5-fold cross validation when applying the algorithm.

```
controlRf <- trainControl(method="cv", 5)
modelRf <- train(classe ~ ., data=trainData, method="rf", trControl=controlRf, ntree=250)
modelRf
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10988, 10989, 10989, 10991, 10991
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9912654  0.9889499
##   27    0.9916291  0.9894104
##   52    0.9842766  0.9801110
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

We will estimate the performance of the model on the validation data set

The estimated accuracy of the model is 99.30% and the estimated out-of-sample error is 0.70%.

#Test results

we apply the model to the original testing data set downloaded from the data source. We remove the problem_id column first.

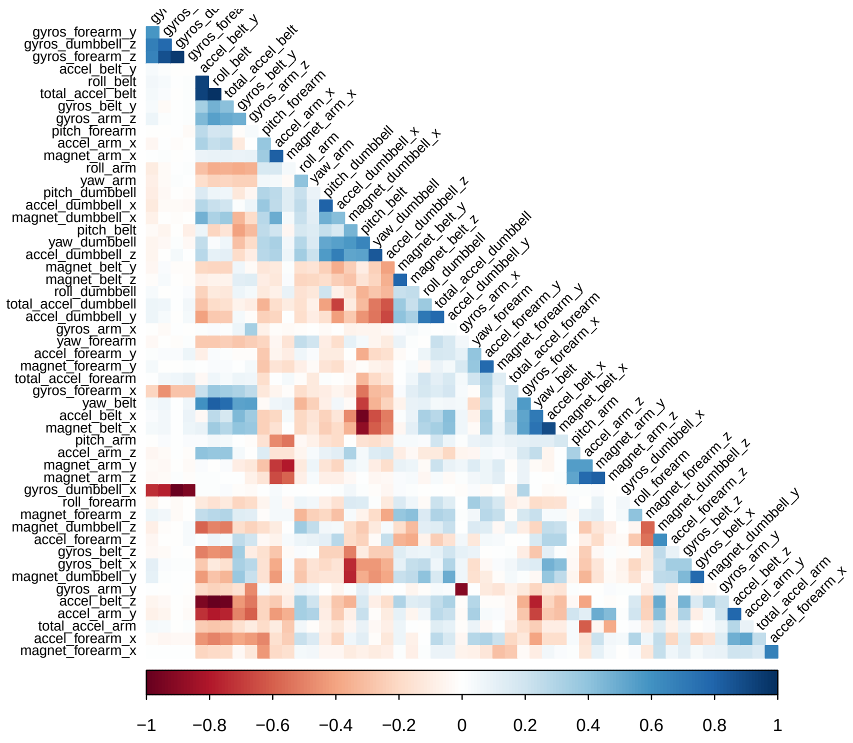
```
result <- predict(modelRf, testCleaned[, -length(names(testCleaned))])
result
```

```
##  [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
```

Appendix: Figures

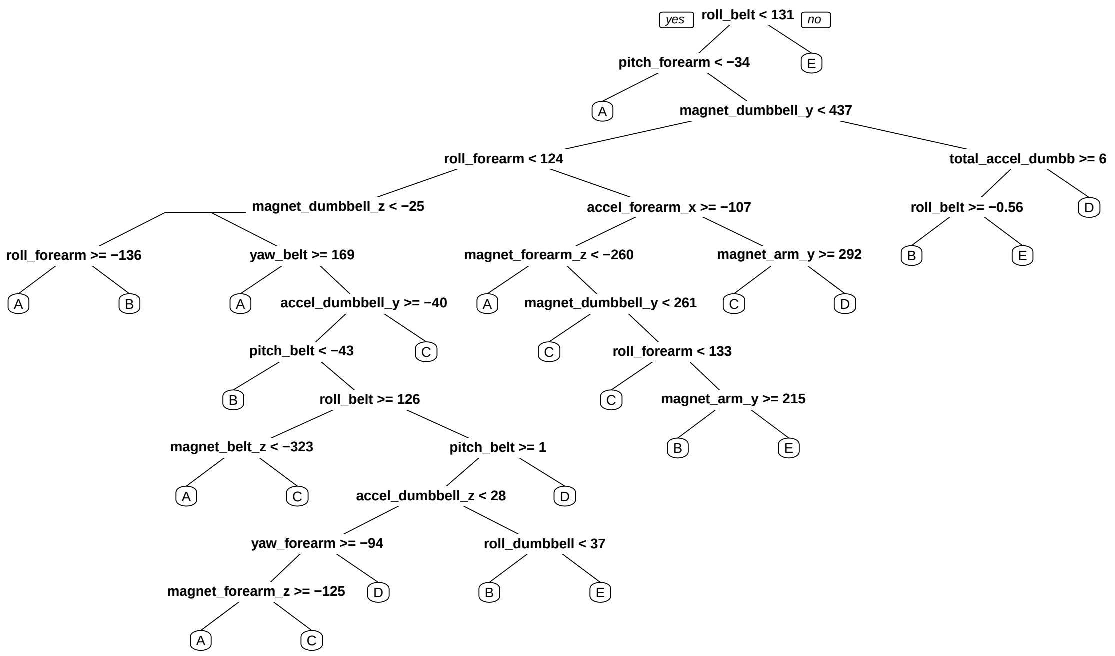
1. Correlation Matrix Visualization

```
corrPlot <- cor(trainData[, -length(names(trainData))])
corrplot(corrPlot, method="color", type="lower", order="hclust", tl.cex=0.70, tl.col="black", tl.srt = 4
```



2. Decision Tree Visualization

```
treeModel <- rpart(classe ~ ., data=trainData, method="class")
prp(treeModel)
```


yes
roll_belt < 131
no
pitch_forearm < −34
E
A
magnet_dumbbell_y < 437
roll_forearm < 124
total_accel_dumbb >= 6
magnet_dumbbell_z < −25
accel_forearm_x >= −107
roll_belt >= −0.56
D
roll_forearm >= −136
yaw_belt >= 169
magnet_forearm_z < −260
magnet_arm_y >= 292
B
E
A
B
A
accel_dumbbell_y >= −40
A
magnet_dumbbell_y < 261
C
D
pitch_belt < −43
C
C
roll_forearm < 133
B
roll_belt >= 126
C
magnet_arm_y >= 215
magnet_belt_z < −323
pitch_belt >= 1
B
E
A
C
accel_dumbbell_z < 28
D
yaw_forearm >= −94
roll_dumbbell < 37
magnet_forearm_z >= −125
D
B
E
A
C