Machine Learning Course project

Ayomi Upekkha

8/7/2020

1.Overview

In this report, we will build a predictions and analyses data based on the dataset given in http://groupware.les.inf.puc-rio.br/har . The main goal will be to use data from accelerometers on the belt, arm and dumbbell of six patients. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. In here, we will do exploratory data analysis, explain the model, do cross validation to check the accuracy of the model and obtain the sample error. Further, we will also use our prediction model to predict 20 different test cases.

2.Analysis

In order to build machine learning model, it is necessary to understand the data. The training data and testing data are given in the project instructions. The source of the dataset are given by the following links.

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

2.1 Data Preprocessing

At first, load the necessary packages that might be want.

```
library(caret)
library(rpart)
library(rpart.plot)
library(rattle)
library(randomForest)
library(corrplot)
library(gbm)
library(tidyverse)
```

```
set.seed(12345)

train_csv <- read.csv("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
test_csv <- read.csv("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
dim(train_csv)
```

```
[1] 19622   160
```

```
dim(test_csv)
```

```
[1]  20 160
```

```r
#split the train test using "classe" variable in the set
split_train  <- createDataPartition(train_csv$classe, p=0.8, list=FALSE)
train <- train_csv[split_train, ]
test  <- train_csv[-split_train, ]
dim(train)
```

```
[1] 15699   160
```

```r
dim(test)
```

```
[1] 3923  160
```

The given train set has 19622 observations with 160 variables and test set has 20 observations. We split the train set in to 80% and remaining 20% for the validation

```r
most_null    <- sapply(train, function(x) mean(is.na(x))) > 0.95
train <- train[, most_null==FALSE]
test  <- test[, most_null==FALSE]
dim(train)
```

```
[1] 15699    93
```

```r
trainData<- train[, colSums(is.na(train)) == 0]
testData <- test[, colSums(is.na(test)) == 0]

near_zero <- nearZeroVar(trainData)
trainData <- trainData[, -near_zero]
testData  <- testData[, -near_zero]
dim(trainData)
```

```
[1] 15699    59
```

We observed some columns have null values. So that we remove the variables that contains missing values and use nearZeroVar function for clean data in caret package. After all, we could be able to reduce variables in to 59 out of 160 variables in the training dataset.

```r
#removed identification variables which is described in the dataset(column 1 to 5)
trainData <- trainData[, -(1:5)]
testData  <- testData[, -(1:5)]
dim(trainData)
```

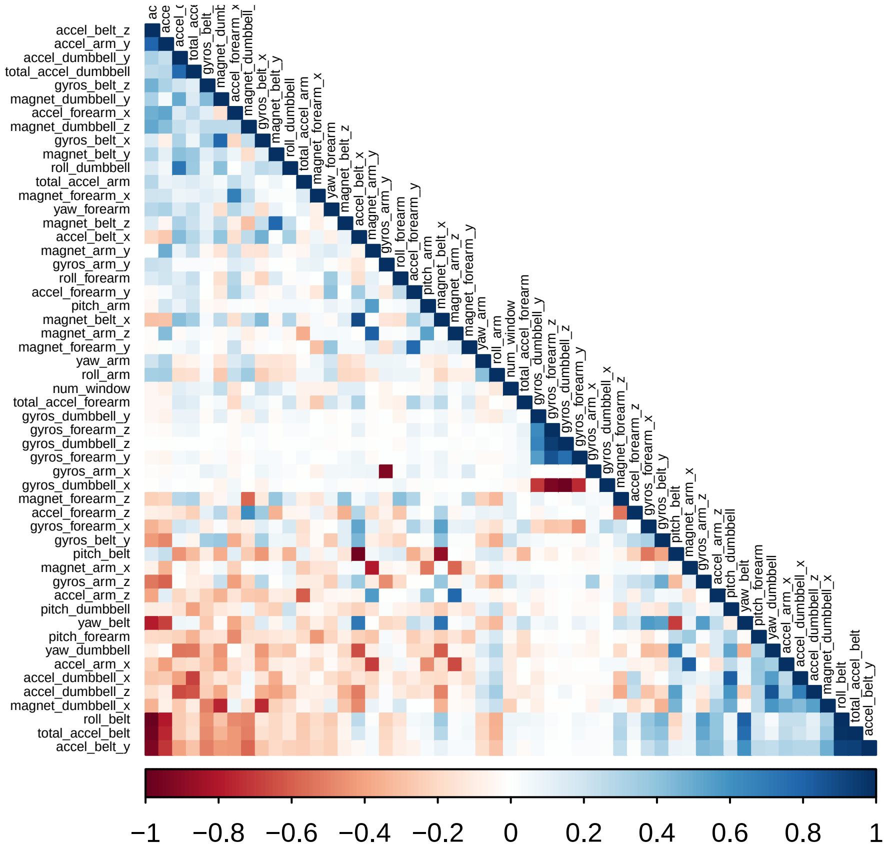
```
[1] 15699    54
```

```r
str(trainData)
```

```
'data.frame':   15699 obs. of  54 variables:
 $ num_window          : int  11 11 11 12 12 12 12 12 12 12 ...
 $ roll_belt           : num  1.41 1.41 1.42 1.48 1.48 1.45 1.43 1.45 1.45 1.42 ...
 $ pitch_belt          : num  8.07 8.07 8.07 8.05 8.07 8.06 8.16 8.17 8.18 8.2 ...
 $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
```

```
 $ total_accel_belt    : int  3 3 3 3 3 3 3 3 3 3 ...
 $ gyros_belt_x        : num  0 0.02 0 0.02 0.02 0.02 0.02 0.03 0.03 0.02 ...
 $ gyros_belt_y        : num  0 0 0 0 0.02 0 0 0 0 0 ...
 $ gyros_belt_z        : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 0 -0.02 0 ...
 $ accel_belt_x        : int  -21 -22 -20 -22 -21 -21 -20 -21 -21 -22 ...
 $ accel_belt_y        : int  4 4 5 3 2 4 2 4 2 4 ...
 $ accel_belt_z        : int  22 22 23 21 24 21 24 22 23 21 ...
 $ magnet_belt_x       : int  -3 -7 -2 -6 -6 0 1 -3 -5 -3 ...
 $ magnet_belt_y       : int  599 608 600 604 600 603 602 609 596 606 ...
 $ magnet_belt_z       : int  -313 -311 -305 -310 -302 -312 -312 -308 -317 -309 ...
 $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
 $ pitch_arm           : num  22.5 22.5 22.5 22.1 22.1 22 21.7 21.6 21.5 21.4 ...
 $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
 $ total_accel_arm     : int  34 34 34 34 34 34 34 34 34 34 ...
 $ gyros_arm_x         : num  0 0.02 0.02 0.02 0 0.02 0.02 0.02 0.02 0.02 ...
 $ gyros_arm_y         : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.03 -0.03 -0.02 ...
 $ gyros_arm_z         : num  -0.02 -0.02 -0.02 0.02 0 0 -0.02 -0.02 0 -0.02 ...
 $ accel_arm_x         : int  -288 -290 -289 -289 -289 -289 -288 -288 -290 -287 ...
 $ accel_arm_y         : int  109 110 110 111 111 111 109 110 110 111 ...
 $ accel_arm_z         : int  -123 -125 -126 -123 -123 -122 -122 -124 -123 -124 ...
 $ magnet_arm_x        : int  -368 -369 -368 -372 -374 -369 -369 -376 -366 -372 ...
 $ magnet_arm_y        : int  337 337 344 344 337 342 341 334 339 338 ...
 $ magnet_arm_z        : int  516 513 513 512 506 513 518 516 509 509 ...
 $ roll_dumbbell       : num  13.1 13.1 12.9 13.4 13.4 ...
 $ pitch_dumbbell      : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
 $ yaw_dumbbell        : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
 $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
 $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
 $ gyros_dumbbell_z    : num  0 0 0 -0.02 0 0 0 0 0 -0.02 ...
 $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -235 -233 -234 ...
 $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 48 47 48 ...
 $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -269 -270 -269 -269 ...
 $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -549 -558 -564 -552 ...
 $ magnet_dumbbell_y   : int  293 296 298 303 292 294 292 291 299 302 ...
 $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -65 -69 -64 -69 ...
 $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.7 27.7 27.6 27.2 ...
 $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 -63.9 ...
 $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -151 ...
 $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
 $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.03 0.02 0.02 0 ...
 $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 0 -0.02 0 ...
 $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 -0.02 -0.02 -0.03 ...
 $ accel_forearm_x     : int  192 192 196 189 189 193 193 190 193 193 ...
 $ accel_forearm_y     : int  203 203 204 206 206 203 204 205 205 205 ...
 $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -214 -215 -214 -215 ...
 $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -16 -22 -17 -15 ...
 $ magnet_forearm_y    : num  654 661 658 658 655 660 653 656 657 655 ...
 $ magnet_forearm_z    : num  476 473 469 469 473 478 476 473 465 472 ...
 $ classe              : chr  "A" "A" "A" "A" ...
```

```r
corMatrix <- cor(trainData[, -54])
M <- (round(corMatrix,2))
corrplot(M, order = "FPC", method = "color",type = "lower",tl.cex = 0.4,tl.col = rgb(0, 0, 0))
```



The variables which has strong positive relationships are shown in dark blue colours and strong negative relationships are shown in dark red colours.

```
highly_Correlated_attr = findCorrelation(M, cutoff=0.75)
names(trainData)[highly_Correlated_attr]
```

```
 [1] "accel_belt_z"      "roll_belt"          "accel_belt_y"
 [4] "accel_arm_y"       "total_accel_belt"   "accel_dumbbell_z"
 [7] "accel_belt_x"      "pitch_belt"         "magnet_dumbbell_x"
[10] "accel_dumbbell_y"  "magnet_dumbbell_y"  "accel_dumbbell_x"
[13] "accel_arm_x"       "accel_arm_z"        "magnet_arm_y"
[16] "magnet_belt_z"     "accel_forearm_y"    "gyros_forearm_y"
[19] "gyros_dumbbell_x"  "gyros_dumbbell_z"   "gyros_arm_x"
```

In this out it can be show that the variables which have high correlated relationship.

3. Model Building

In order to building the model, we used randomForest, Desission Trees and generalized Boosted Model.

3.1 Random Forest

```
set.seed(12345)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRandForest <- train(classe ~ ., data=trainData, method="rf",
                          trControl=controlRF)
modFitRandForest$finalModel
```

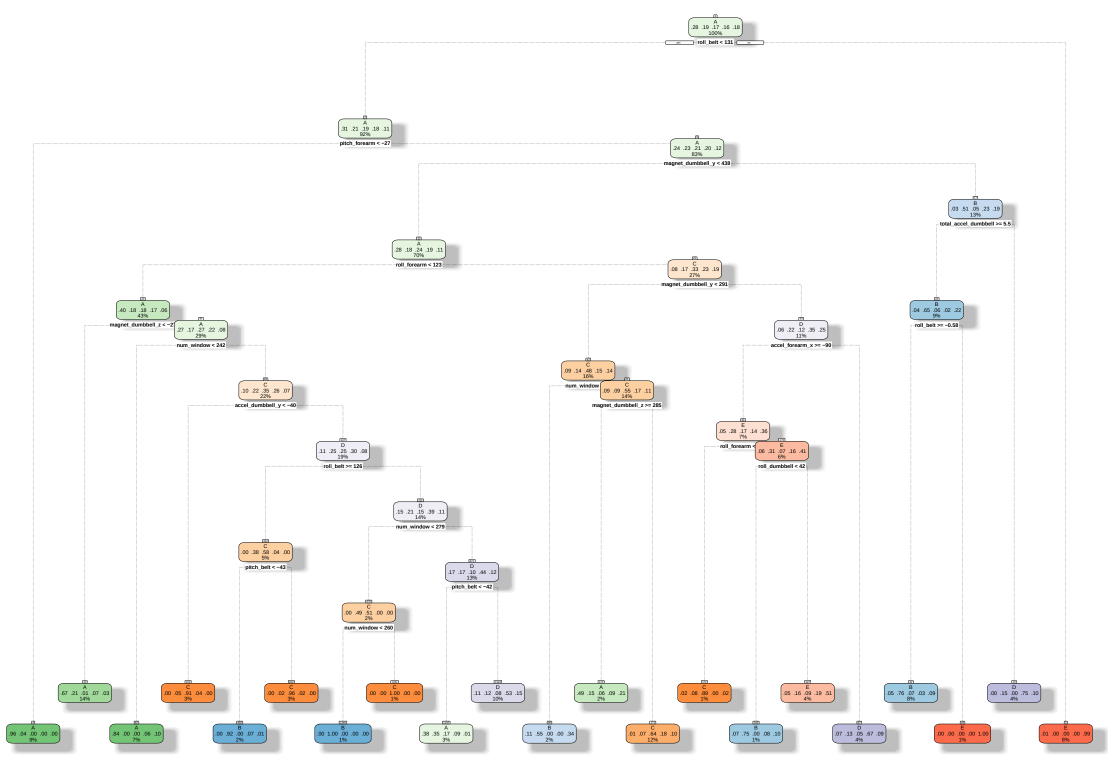
```
Call:
 randomForest(x = x, y = y, mtry = param$mtry)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 27

        OOB estimate of  error rate: 0.18%
Confusion matrix:
     A    B    C    D    E  class.error
A 4463    0    0    0    1 0.0002240143
B    7 3028    2    1    0 0.0032916392
C    0    3 2735    0    0 0.0010956903
D    0    0   13 2560    0 0.0050524679
E    0    0    0    2 2884 0.0006930007
```

```r
#checking the accuracy of the table
predictRF1 <- predict(modFitRandForest,testData)
```

3.2 Decision Trees

```r
set.seed(12345)
tree <- rpart(classe ~ ., data=trainData, method="class")
fancyRpartPlot(tree)
```



Rattle 2020−Aug−08 11:37:02 Dell

```r
predictTree <- predict(tree, newdata=testData, type="class")
```

3.3 Generalized Boosted Model

```r
set.seed(12345)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modFitGBM  <- train(classe ~ ., data=trainData, method = "gbm",
                    trControl = controlGBM, verbose = FALSE)
modFitGBM$finalModel
```

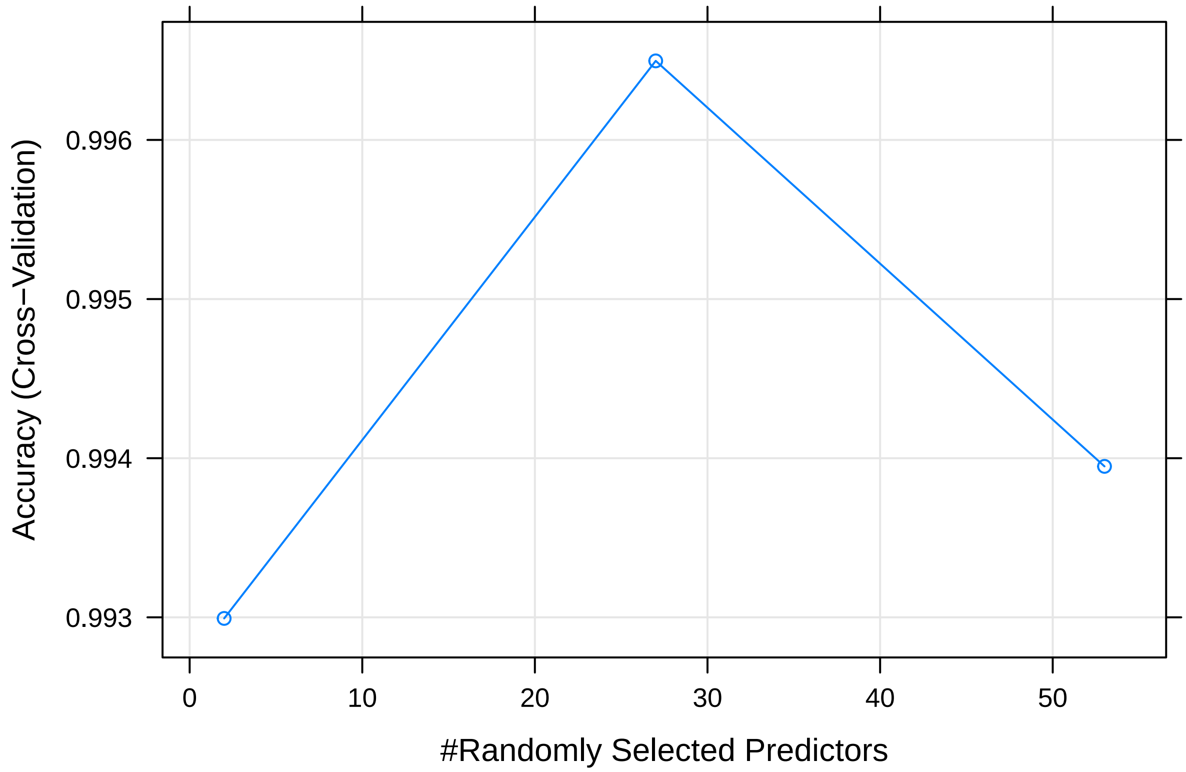
```
A gradient boosted model with multinomial loss function.
150 iterations were performed.
There were 53 predictors of which 53 had non-zero influence.
```

```r
predictGBM <- predict(modFitGBM, newdata=testData)
```

```
# A tibble: 3 x 2
  model accuracy
  <chr>    <dbl>
1 RF       0.996
2 DT       0.737
3 GBM      0.984
```

According to these all outputs it can be say that random forest has the highest accuracy than the generalized boosted model and decision tree. So that we decided to take random forest technique to predict the model. This model has **accuracy 90% and out of sample error rate approximately zero.**

```r
plot(modFitRandForest)
```




```r
Results <- predict(modFitRandForest, newdata=test_csv)
Results
```

```
 [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
```

4. Conclusion

Based on this data, we could be able to find the suitable model to get the prediction. It was random forest method because it had the highest accuracy and lowest sample error. Final prediction is based on the original training set given in the project instruction source link. It has 20 observations and we could be able to build the prediction based on these given datasets.