Machine learning_Prediction using Weigth Lifting Exercise

Goga

2023-06-20

Predictions using the Weight Lifting Exercises Dataset

[GitHub link for Project in HTML:]http://bit.ly/2c8Mbti

1 - Summary

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement ??? a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks.

One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

This project has the purpose to predict the manner in which users perform the exercises. There are 5 possible results, reported in the `classe` variable:

- A: exactly according to the specification
- B: throwing the elbows to the front
- C: lifting the dumbbell only halfway
- D: lowering the dumbbell only halfway
- E: throwing the hips to the front

[Data provided by curtesy of:]http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har

The data for the project is available at:

[Training Dataset:]https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

[Testing Dataset:]https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The objective of this project is to predict the `classe` based on data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants.

2 - Libraries

library(lattice); library(ggplot2); library(plyr); library(caret); install.packages("randomForest") install.packages("tree") install.packages("rattle") install.packages("rpart") install.packages("corrplot") set.seed(6266) # set contact random seed

Read the Data

After downloading the data from the data source, we can read the two csv files into two data frames.

```
training <- read.csv("C:/Users/gogab/Documents/Coursera/Machine
learning/Course project/pml-training.csv")
testing <- read.csv("C:/Users/gogab/Documents/Coursera/Machine
learning/Course project/pml-testing.csv")
dim(training)

## [1] 19622   160

dim(testing)

## [1]  20 160
```

The training data set contains 19622 observations and 160 variables, while the testing data set contains 20 observations and 160 variables. The "classe" variable in the training set is the outcome to predict.

Clean the data

In this step, we will clean the data and get rid of observations with missing values as well as some meaningless variables.

```
sum(complete.cases(training))

## [1] 406
```

First, we remove columns that contain NA missing values.

```
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]
```

Next, we get rid of some columns that do not contribute much to the accelerometer measurements.

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

```
testing <- testing[, !testRemove]
testCleaned <- testing[, sapply(testing, is.numeric)]

dim(trainCleaned)

## [1] 19622    53

dim(testCleaned)

## [1] 20 53
```

The cleaned training data set contains 19622 observations and 53 variables, while the testing data set contains 20 observations and 53 variables. The "classe" variable is still in the cleaned training set.

Slice the data

Then, we can split the cleaned training set into a pure training data set (70%) and a validation data set (30%). We will use the validation data set to conduct cross validation in future steps.

```
set.seed(22519)  ##For reproducibile purpose
inTrain <- caret::createDataPartition(trainCleaned$classe, p=0.70, list=F)
trainData <- trainCleaned[inTrain, ]
testData <- trainCleaned[-inTrain, ]

dim(trainData)

## [1] 13737    53

dim(testData)

## [1] 5885   53
```

We split data into training set and test set. Now we have 13737 observations and 53 variables in training set and 5885 observations and 53 variables in testing set.

Data Modeling

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

## Warning: package 'caret' was built under R version 4.2.3

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.2.3
## Loading required package: lattice

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.

# create a confusion matrix for the model on the sub-testing set to see how
accurate the prediction is
ConMatrix <- confusionMatrix(table(testData$classe, predict(modelRf,
testData)))
print(ConMatrix)

## Confusion Matrix and Statistics
##
##
##        A    B    C    D    E
##   A 1669    2    3    0    0
##   B    5 1130    3    1    0
##   C    0    4 1019    3    0
##   D    0    0   10  954    0
##   E    0    0    4    2 1076
##
## Overall Statistics
##
##                Accuracy : 0.9937
##                  95% CI : (0.9913, 0.9956)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.992
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9947   0.9808   0.9938   1.0000
## Specificity            0.9988   0.9981   0.9986   0.9980   0.9988
## Pos Pred Value         0.9970   0.9921   0.9932   0.9896   0.9945
## Neg Pred Value         0.9988   0.9987   0.9959   0.9988   1.0000
## Prevalence             0.2845   0.1930   0.1766   0.1631   0.1828
## Detection Rate         0.2836   0.1920   0.1732   0.1621   0.1828
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9979   0.9964   0.9897   0.9959   0.9994
```

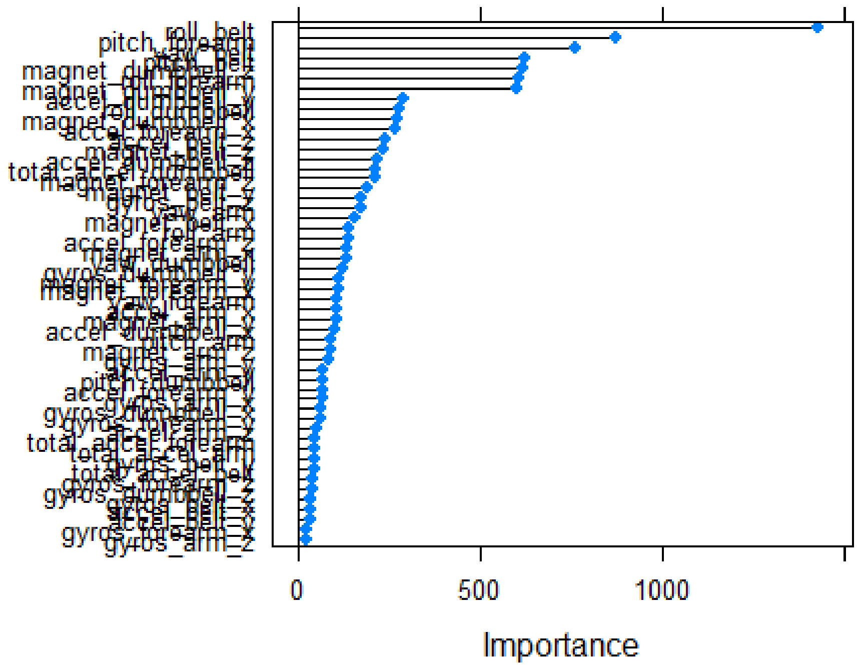
Step 4: Prediction

```
# predict
pred<-predict(modelRf, testing)
pred

##  [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
```

Step 5: Check importance of features

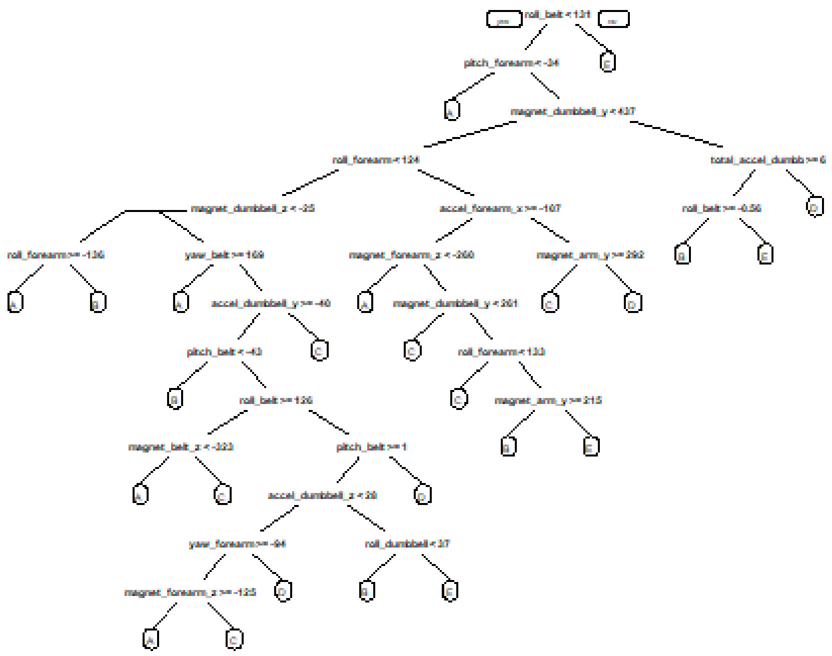
```
importance <- varImp(modelRf, scale=FALSE)
plot(importance)
```



## Appendix: Figures

Decision Tree Visualization

```r
treeModel <-rpart::rpart(classe ~ ., data=trainData, method="class")
rpart.plot::prp(treeModel) # fast plot
```


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 < 123
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