Predicting Analysis Performance using Accelerometer Data

2023-05-18

1. Introduction

The quantified self movement has enabled the collection of vast amounts of personal activity data through wearable devices such as Jawbone Up, Nike FuelBand, and Fitbit. While people often quantify the amount of activity they engage in, they rarely measure the quality or effectiveness of their performance. In this project, our goal is to utilize accelerometer data from the belt, forearm, arm, and dumbbell of 6 participants to predict the manner in which they perform barbell lifts.

2. Project Goal

The goal of this project 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.

3. Data Source

The data is sourced from the Weight Lifting Exercise Dataset provided by the Human Activity Recognition (HAR) group at PUC-Rio. We express our gratitude to the HAR group for generously allowing the use of their data for this project. Please cite them if you use this document for any purpose.

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

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

4. Data Loading, Exploration and Preprocessing

4.1 Library loading for this project

```
library(caret)
```

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

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

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

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

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

```
library(rpart)
library(rpart.plot)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

4.2 Data Loading

```
TrainData <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
TestData <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
```

4.3 Data Exploration and Preprocessing

show data set dimension

```
dim(TrainData)
```

```
## [1] 19622   160
```

```
dim(TestData)
```

```
## [1]  20 160
```

The training data contains 19622 observations and 160 variables, test data contains 20 observations and 160 variables.

remove the column with more than 70% NAs or empty

```
TrainClean <- TrainData[, colMeans(is.na(TrainData)|TrainData=="")<0.7]
TestClean <- TestData[, colMeans(is.na(TestData)|TestData=="")<0.7]
```

After data cleaning, the training data contains 19622 observations and 93 variables, test data contains 20 observations and 60 variables.

remove the redundant variables

```
TrainClean<-TrainClean[,-c(1:7)]
TestClean<-TestClean[,-c(1:7)]
TrainClean$classe<-factor(TrainClean$classe)
```

Remove the first 7 variables from both data sets, which don't contribute much to the modeling.The "classe" in the training set is the outcome to predict.Covert "classe" into factor.

5. Partition the training set into training and validation sets.

```
preproc <- preProcess(TrainClean, method = c("center", "scale"))
TrainClean <- predict(preproc, TrainClean)
TestClean <- predict(preproc, TestClean)
```

split training and validation sets

```
set.seed(888)
foo <- createDataPartition(TrainClean$classe, p=0.70, list=FALSE)
forTrain <- TrainClean[foo, ]
forVali <- TrainClean[-foo, ]
```

We randomly split the training data into 70% for training and 30% for validation sets to evaluate the model's performance and tune hyperparameters, ensuring the model does not overfit the training data. we set.seed here to ensure the analysis can be reproduced.

6. Model Building

6.1 Train Random Forest (RF) Model and validation

We train the RF model with the preprocessed forTrain data set.

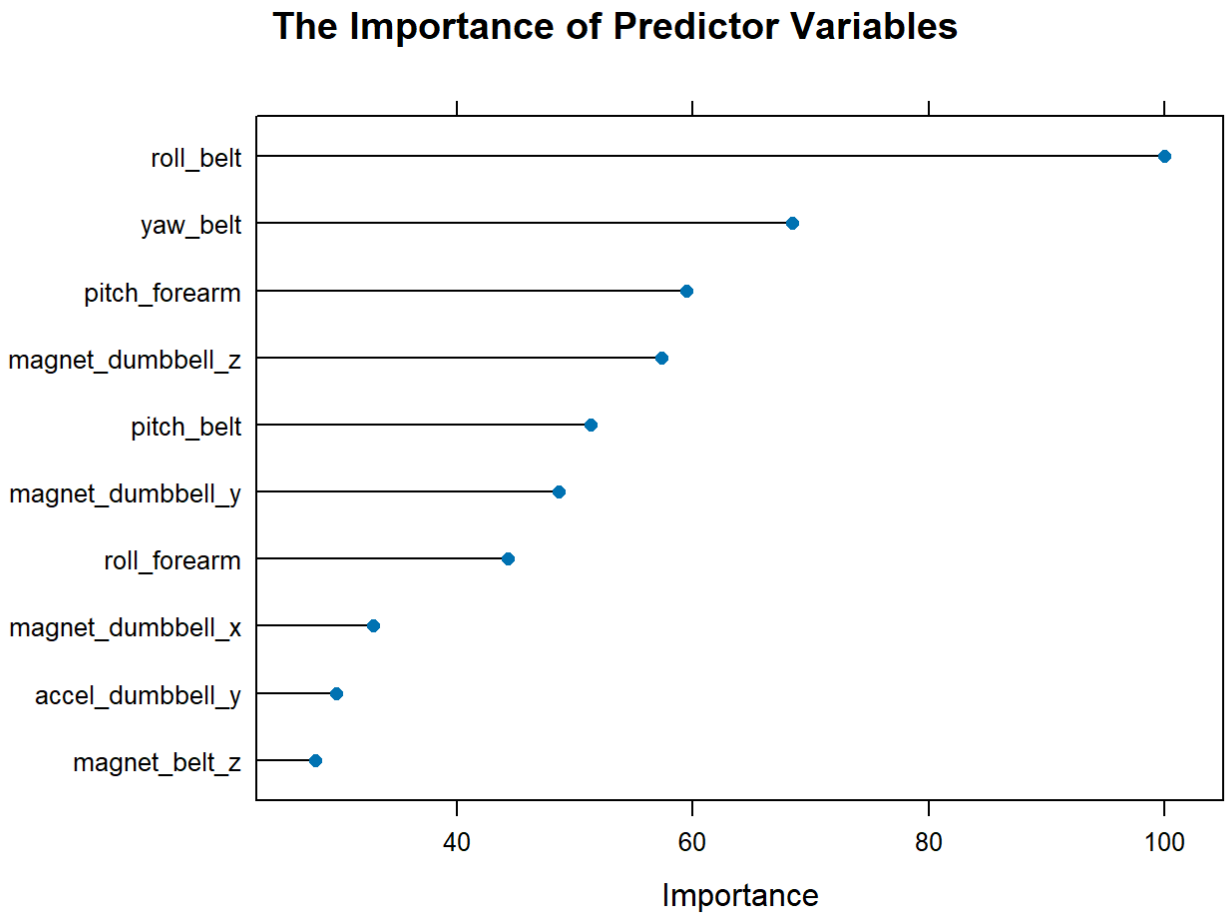
```
controlRF <- trainControl(method="cv", number=5)
tuneG<-expand.grid(mtry = c(4, 8))
modelRF <- train(classe ~ ., data=forTrain, method="rf", trControl=controlRF, tuneGrid=tun
eG)
modelRF # result hidden
```

Then, we estimate the performance of the model on the validation data set.

```
predictRF <- predict(modelRF, forVali)
confusionMatrix(forVali$classe, predictRF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    0    0    1    1
##          B    4 1134    1    0    0
##          C    0   10 1014    2    0
##          D    0    0    5  958    1
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9956
##                  95% CI : (0.9935, 0.9971)
##     No Information Rate : 0.2848
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9944
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9913   0.9941   0.9958   0.9982
## Specificity            0.9995   0.9989   0.9975   0.9988   0.9998
## Pos Pred Value         0.9988   0.9956   0.9883   0.9938   0.9991
## Neg Pred Value         0.9991   0.9979   0.9988   0.9992   0.9996
## Prevalence             0.2848   0.1944   0.1733   0.1635   0.1840
## Detection Rate         0.2841   0.1927   0.1723   0.1628   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9986   0.9951   0.9958   0.9973   0.9990
```

```
ImpoVar<-varImp(modelRF)
plot(ImpoVar, top=10, main="The Importance of Predictor Variables")
```



So, after training and validation, we obtained an optimized model with satisfactory performance metrics. The overall accuracy of the model is 99.6%, the No Information Rate is 28.5%. The model demonstrated good accuracy.

6.2 Train Decision Tree (DT) Model and Validation

We train the DT model with the preprocessed forTrain data set.

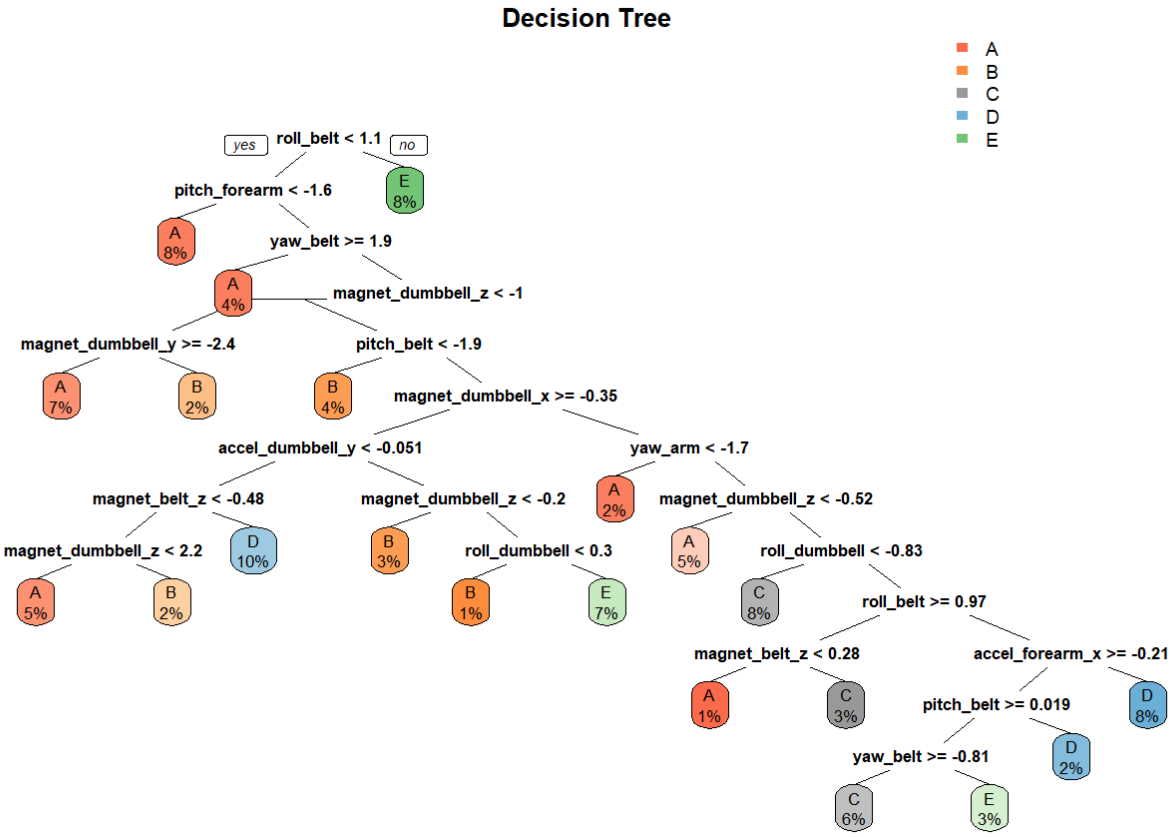
```
modelDT <- rpart(classe ~ ., data=forTrain, method="class")
modelDT # result hidden
```

Then, we estimate the performance of the model on the validation data set.

```
predictDT <- predict(modelDT, forVali, type = "class")
confusionMatrix(predictDT, forVali$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1497  176   28   49   70
##          B   29  600   54   73   30
##          C   42  114  686   82   59
##          D   71  129  216  652  194
##          E   35  120   42  108  729
##
## Overall Statistics
##
##                Accuracy : 0.7076
##                  95% CI : (0.6958, 0.7192)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6297
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8943   0.5268   0.6686   0.6763   0.6738
## Specificity            0.9233   0.9608   0.9389   0.8760   0.9365
## Pos Pred Value         0.8225   0.7634   0.6979   0.5166   0.7050
## Neg Pred Value         0.9565   0.8943   0.9306   0.9325   0.9272
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2544   0.1020   0.1166   0.1108   0.1239
## Detection Prevalence   0.3093   0.1336   0.1670   0.2144   0.1757
## Balanced Accuracy      0.9088   0.7438   0.8037   0.7762   0.8051
```

```
prp(modelDT,extra=100, box.palette="auto", main="Decision Tree")
```



We obtained an optimized model with satisfactory performance metrics. The overall accuracy of the model is 70.8%, the No Information Rate 28.5%. The model demonstrated acceptable accuracy.

The RF model has better performance compared with DT model, and has the ability to predict exercise performance based on accelerometer data.

7. Predicting for Test Data Set with random forest model

We apply our trained model to predict the manner in which 20 test data set.

```
predictResult <- predict(modelRF, TestClean[, -length(names(TestClean))])
predictResult
```

```
##  [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
```

8. Conclusion

In conclusion, this project aimed to predict exercise performance based on accelerometer data. It turned out the random forest model is the good model for prediciting the manner in which they did the exercise.

-END-