ML Project - Week 4: WLE

Aiman

2022-10-04

Weight Lifting Exercise Predictions

background

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.

This report:

In this report, we will describe how we were able to predict in an accurate way the manner the athletes performed the exercise (column "classe"). We will be splitting the training dataset into a training and test dataset, while we will be using the provided test set as a validation one. We will use cross-validation to keep the best model and comment the expected out of sample error.

Step by step

Data loading and pre-processing

Firstly, we will load the data and cast the NAs, take a look to the datasets summaries and get rid of the first 7 column as we are not considering the time dimension in our processing.

```
library(caret)
```

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

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

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

```
## Loading required package: rpart
```

```
set.seed(412)

training_data <- read.csv(
  "~/Projects/ml_practice_week_4/pml-training.csv",
  na.strings=c("NA","#DIV/0!","")
  )
validation_data <- read.csv(
  "~/Projects/ml_practice_week_4/pml-testing.csv",
  na.strings=c("NA","#DIV/0!","")
  )

training_data <- training_data[,-seq(1:7)]
validation_data <- validation_data[,-seq(1:7)]

summary(training_data)
summary(validation_data)
```

While looking at the different features, we can see that some of the columns have a ratio > 90% of NAs, we will get rid of these columns. This will leave us with 52 predictors

```
columnIndex <- colSums(is.na(training_data))/nrow(training_data) < 0.9
training_data <- training_data[,columnIndex]
validation_data <- validation_data[,columnIndex]
```

Now, we will run the data normalization (scaling and centering). For that we will use the preProcess object.

```
classes <- training_data$classe
preObj <-preProcess(training_data[,1:52],method=c('center', 'scale'))
training_data <- predict(preObj, training_data[,1:52])
training_data$classe <- as.factor(classes)

val_id <- validation_data$problem_id
validation_data <- predict(preObj,validation_data[,1:52])
validation_data$problem_id <- val_id
```

Finally, we will split our training data into training and testing data

```
inTrain <- createDataPartition(training_data$classe, p=0.75)[[1]]
training_data <- training_data[inTrain,]
testing_data <- training_data[-inTrain,]
```

Models training

Here we will train 3 different models and compare their results.

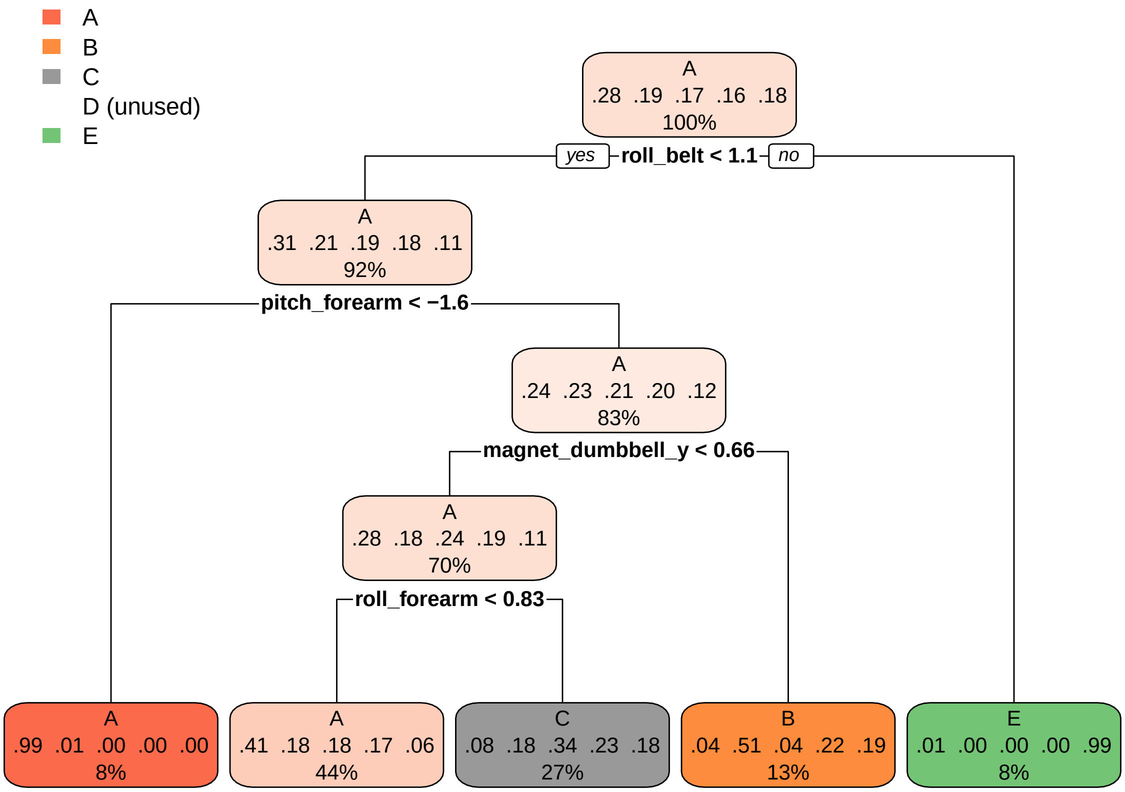
decision tree

```
decisionTreeMod <- train(classe~., data=training_data, method='rpart')
```

```
decisionTreePrediction <- predict(decisionTreeMod, testing_data)
cmTree <- confusionMatrix(testing_data$classe, decisionTreePrediction)
cmTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 919  17  91   0   1
##          B 317 243 193   0   0
##          C 282  15 332   0   0
##          D 271 109 214   0   0
##          E 101  95 172   0 296
##
## Overall Statistics
##
##                Accuracy : 0.488
##                  95% CI : (0.4717, 0.5043)
##     No Information Rate : 0.5153
##     P-Value [Acc > NIR] : 0.9995
##
##                   Kappa : 0.3327
##
##  Mcnemar’s Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4862  0.50731  0.33134       NA  0.99663
## Specificity            0.9387  0.84008  0.88860   0.8381  0.89083
## Pos Pred Value         0.8940  0.32271  0.52782       NA  0.44578
## Neg Pred Value         0.6322  0.91904  0.77953       NA  0.99967
## Prevalence             0.5153  0.13059  0.27317   0.0000  0.08097
## Detection Rate         0.2505  0.06625  0.09051   0.0000  0.08070
## Detection Prevalence   0.2803  0.20529  0.17148   0.1619  0.18103
## Balanced Accuracy      0.7125  0.67369  0.60997       NA  0.94373
```

```
rpart.plot(decisionTreeMod$finalModel)
```



Random Forest

```
RFMod <- train(classe~., data=training_data, method='rf')

RFPrediction <- predict(RFMod, testing_data)
cmRF <- confusionMatrix(testing_data$classe, RFPrediction)
cmRF
```

GBM

```
GBMMod <- train(classe~., data=training_data, method='gbm')



GBMPrediction <- predict(GBMMod, testing_data)
cmGBM <- confusionMatrix(testing_data$classe, GBMPrediction)
cmGBM
```

Comparison

```
results <- data.frame(
  Model = c('decision tree', 'RF', 'GBM'),
  Accuracy = rbind(cmTree$overall[1], cmRF$overall[1], cmGBM$overall[1])
)
print(results)
```

We can see that GBM and Random forest perform better on the testing dataset. We will keep the random forest as the best model for the prediction on the validation set.

Validation set

```
RFValPrediction <- predict(RFMod, validation_data)
RFValPrediction
```

Expected out-of-sample error

The expected out-of-sample error is estimated at 0.005, or 0.5%. The expected out-of-sample error is calculated as 1 - accuracy for predictions made against the cross-validation set. The test data we have contains 20 cases. With an accuracy above 99% on cross-validation set, we can expect that almost none of the test samples will be misclassified.