Machine Learning Assignment

Markus Gälli

2/18/2021

Overview

This document covers the analysis of excerise data collected from accelerometers from http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har)

Different models were evaluated to maximize the prediction accuracy of the how the person exercised (correct, or one of 5 incorrect ways).

Data

Data was downloaded from the assigned repository: The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

```
setwd("~/R/Projects/Coursera Class/MachineLearningAssignment")
training <- read.csv("./data/pml-training.csv")
testing <- read.csv("./data/pml-testing.csv")
```

Data Preparation

To facilitate the subsequent analysis, the training data has to be transformed somewhat:

- Turn the “classe” variable (indicating the exercise type) into a factor variable
- Put all the data into a data frame
- Make sure the data is imported as numeric (and not as character)
- Remove any variables with “NA” in them, as they can’t be used for prediction
- Remove the first 7 columns as they do not contain data useful for this analysis

```
training$classe <- factor(training$classe)
oldw <- getOption("warn")
options(warn = -1)
trainData <- as.data.frame(apply(training[,1:159],2,as.numeric))
options(warn = oldw)
trainData <- trainData %>% mutate(classe=training$classe)
cntNA <- apply(is.na(trainData), 2, sum)
ignore=which(cntNA != 0)
trainData <- trainData %>% select(-all_of(ignore)) %>% select(-(1:7))
```

The remaining data has the following dimension: 19622 rows and 50 columns.

Next, we want to check for highly correlated variables. The diagonal has to be removed, and then a list is created with pairs of measurement variables that have a correlation of > 0.8. As the whole correlation matrix is included, each pair shows up twice.

```
res <- cor(trainData[,-50])
high_corr <- which(abs(res)>0.8)
nc <- ncol(trainData)
dn <- seq(1,nc-1,by=1)
diagonal <- function(x){
    (x-1)*(length(x)+1)+1
}
diag <- diagonal(dn)
diag_ord <- high_corr %in% diag
remove <- high_corr[!diag_ord]
remove_row <- as.integer((remove-1)/49)+1
remove_col <- (remove-1) %% 49 +1

removeNames <- data.frame(rowN = names(trainData)[remove_row])
removeNames <- cbind(removeNames,colN = names(trainData)[remove_col])
removeNames
```

```
##                rowN             colN
## 1  total_accel_belt     accel_belt_y
## 2  total_accel_belt     accel_belt_z
## 3      accel_belt_x    magnet_belt_x
## 4      accel_belt_y total_accel_belt
## 5      accel_belt_y     accel_belt_z
## 6      accel_belt_z total_accel_belt
## 7      accel_belt_z     accel_belt_y
## 8     magnet_belt_x     accel_belt_x
## 9       gyros_arm_x      gyros_arm_y
## 10      gyros_arm_y      gyros_arm_x
## 11      accel_arm_x     magnet_arm_x
## 12     magnet_arm_x      accel_arm_x
## 13     magnet_arm_y     magnet_arm_z
## 14     magnet_arm_z     magnet_arm_y
## 15   pitch_dumbbell accel_dumbbell_x
## 16     yaw_dumbbell accel_dumbbell_z
## 17 gyros_dumbbell_x gyros_dumbbell_z
## 18 gyros_dumbbell_x  gyros_forearm_z
## 19 gyros_dumbbell_z gyros_dumbbell_x
## 20 gyros_dumbbell_z  gyros_forearm_z
## 21 accel_dumbbell_x   pitch_dumbbell
## 22 accel_dumbbell_z     yaw_dumbbell
## 23  gyros_forearm_y  gyros_forearm_z
## 24  gyros_forearm_z gyros_dumbbell_x
## 25  gyros_forearm_z gyros_dumbbell_z
## 26  gyros_forearm_z  gyros_forearm_y
```

The highly correlated variables are removed from the data frame (not all of the above variables, but one from each pair).

```
excludeNames <- c("gyros_forearm_z","accel_belt_y","accel_belt_z","gyros_dumbbell_x","gyros_arm_
y","magnet_arm_x","magnet_arm_z",
                  "magnet_belt_x","pitch_dumbbell","yaw_dumbbell")

smallData <- trainData %>% select(-all_of(excludeNames))
```

This data set is now the one to work on.

Modeling

First we break the data set into a training set and a testing set.

```
set.seed(666)
inTrain <- createDataPartition(y=smallData$classe,p=0.7,list=FALSE)
trainingData <- smallData[inTrain,]
testingData <- smallData[-inTrain,]
```

The training set contains 13737 rows, and the testing set contains 5885 rows. Both contain 40 columns.

The following methods will be evaluated:

- Regression Trees
- Random Forests
- Boosting With Trees
- Support Vector Machines
- Linear Discriminant Analysis

For each method, a model is created using the training set, which is then used to predict on the training set. The results are evaluated using a confusion matrix.

```
modFitTree <- train(classe~.,data=trainingData,method="rpart")
predTree <- predict(modFitTree,trainingData)
accTree<- confusionMatrix(predTree,trainingData$classe)
```

```
modFitRF <- train(classe ~ ., data=trainingData, method="rf",trControl = trainControl(method="c
v"),number=3)
predRF <- predict(modFitRF,trainingData)
accRF <- confusionMatrix(predRF,trainingData$classe)
```

```
modFitGBM <- train(classe ~ ., data=trainingData,method="gbm",verbose=FALSE)
predGBM <- predict(modFitGBM,trainingData)
accGBM <- confusionMatrix(predGBM, trainingData$classe)
```

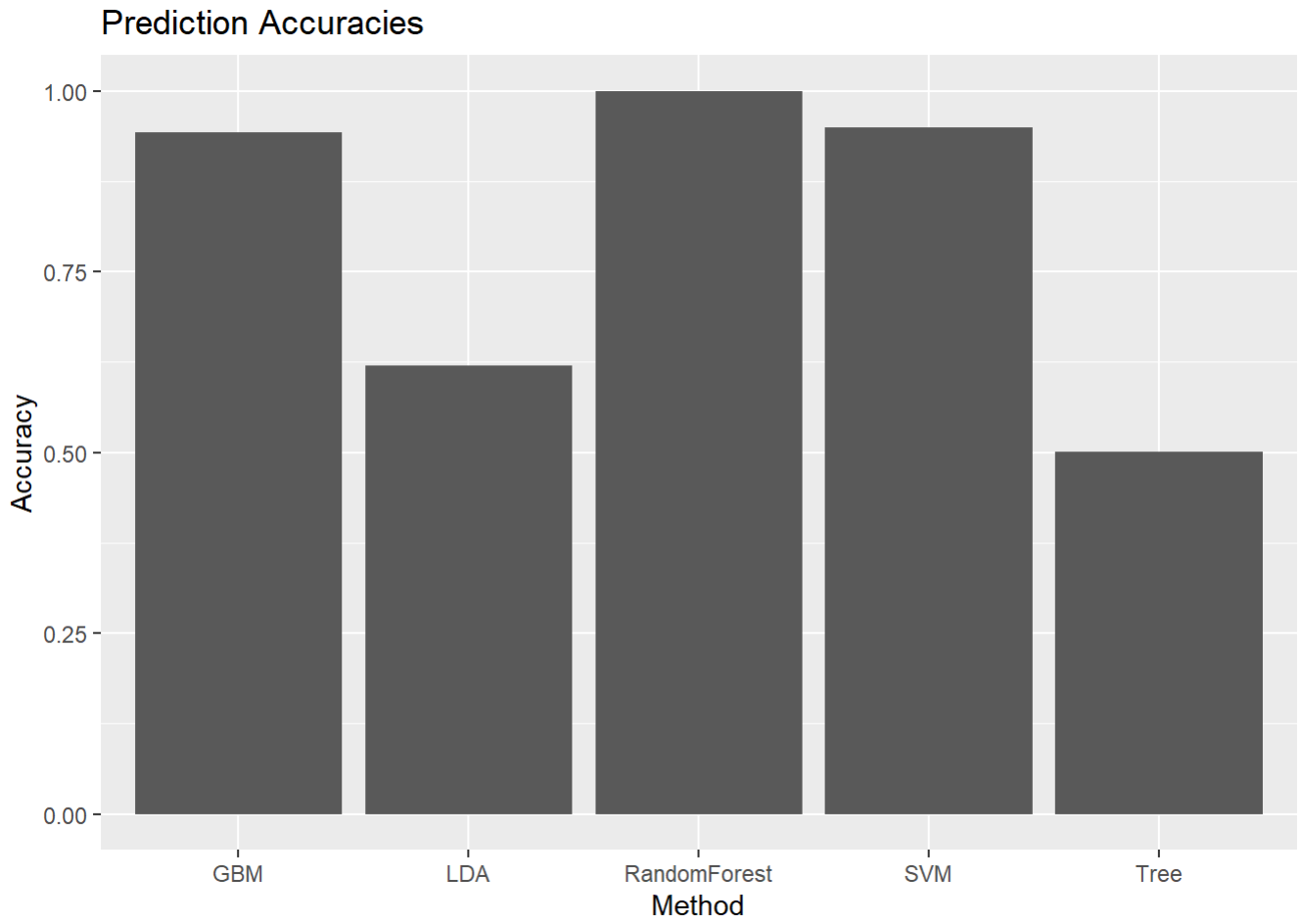
```
modFitSVM <- svm(classe ~ ., data=trainingData)
predSVM <- predict(modFitSVM,trainingData)
accSVM <- confusionMatrix(predSVM,trainingData$classe)
```

```
modFitLDA <- train(classe ~ ., data=trainingData, method= "lda")
predLDA <- predict(modFitLDA,trainingData)
accLDA <- confusionMatrix(predLDA,trainingData$classe)
```

The following graph shows the model accuracy for all four tested method.

```
resModel <- c("Tree","RandomForest","GBM","LDA","SVM")
resAcc <- c(accTree$overall[1],accRF$overall[1],accGBM$overall[1],accLDA$overall[1],accSVM$overa
ll[1])
res <- data.frame(model=resModel,train_accuracy = resAcc)

g <- res %>% ggplot(aes(model,train_accuracy)) + geom_col() +
    labs(title="Prediction Accuracies",x="Method",y="Accuracy")
g
```

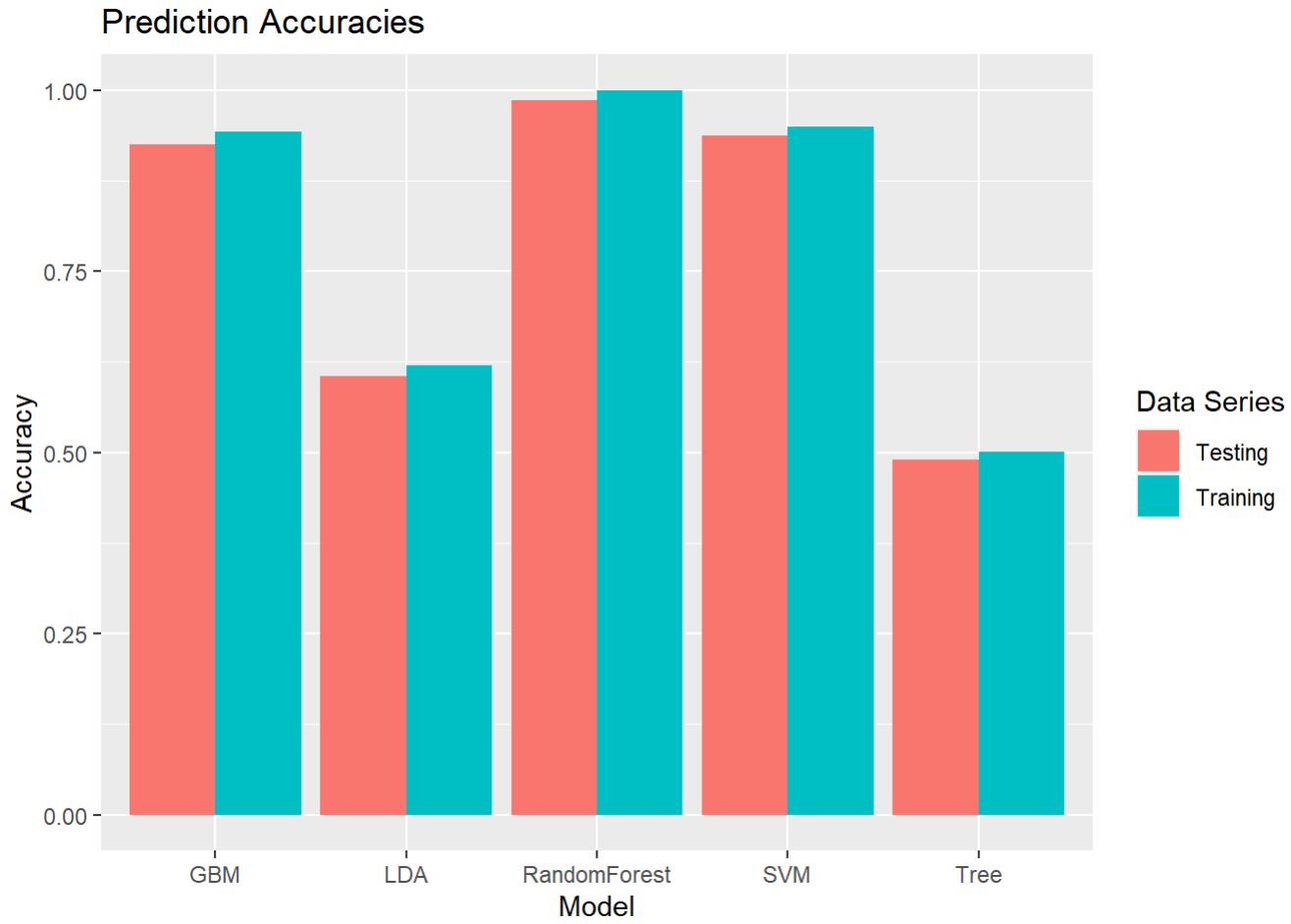


Random Forest, Support Vector Machines and Boosting WIth Trees clearly give the best answers, with high accuracy. Now we validate the models by applying them to the testing data, which also provides the out of sample accuracy.

```
resTree <- confusionMatrix(predict(modFitTree,testingData),testingData$classe)
resRF <- confusionMatrix(predict(modFitRF,testingData),testingData$classe)
resGBM <- confusionMatrix(predict(modFitGBM,testingData),testingData$classe)
resLDA <- confusionMatrix(predict(modFitLDA,testingData),testingData$classe)
resSVM <- confusionMatrix(predict(modFitSVM,testingData),testingData$classe)
resTest <- c(resTree$overall[1],resRF$overall[1],resGBM$overall[1],resLDA$overall[1],resSVM$over
all[1])

res <- cbind(res,test_accuracy=resTest)
reslong <- res %>% gather("series","acc",-model)

g <- reslong %>% ggplot(aes(model,acc,fill=series)) + geom_col(position="dodge") +
    labs(title="Prediction Accuracies",x="Model",y="Accuracy") +
    scale_fill_discrete(name="Data Series",labels = c("Testing","Training"))
g
```



The graph shows both the in-sample and out-of-sample accuracies.

For the top 3 methods, the table below shows the accuracies:

Method	In Model Accuracy	Out Of Model Accuracy
Random Forest	1	0.99
SVM	0.95	0.94
Boosting with Trees	0.94	0.92

Based on these results, Random Forest seems to be the best method for the analysis of the provided data.