Weightlifting Prediction

Vera Sysoeva

9/21/2018

Introduction

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, my goal is 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. I will train multiple models on this data in order to determine the best model and then predict 20 test cases.

Data Processing

Importing the Data

We begin by loading the necessary libraries and downloading/reading the data into R.

```r
library(caret)
library(e1071)
library(rattle)
library(randomForest)

trainURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
if(!exists("Training Data.csv")){
    download.file(trainURL, "Training Data.csv")
}
if(!exists("Testing Data.csv")){
    download.file(testURL, "Testing Data.csv")
}
training <- read.csv("Training Data.csv", na.strings = c("NA",""))
testing <- read.csv("Testing Data.csv", na.strings = c("NA",""))
```

Data Cleaning

Viewing a summary of the data shows us that many columns are filled with mostly 0s or NAs. We can delete columns that contain missing values since these will not be useful for out analysis.

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

We can also remove the label and timestamp columns that just serve to label the data and will not be part of the analysis.

```
trainingClean <- training[,-c(1:7)]
testingClean <- testing[,-c(1:7)]
```

Cross-validation/Data Splitting

In order to cross-validate we create a training and testing set that are subsets of our original training set.

```
set.seed(8)
inTrain <- createDataPartition(y=trainingClean$classe, p=0.7, list=FALSE)
train <- trainingClean[inTrain,]
test <- trainingClean[-inTrain,]
```

Model Building

The data is now ready to train a model. I tried out three different types of models: classification tree, random forest, and gradient boosting.

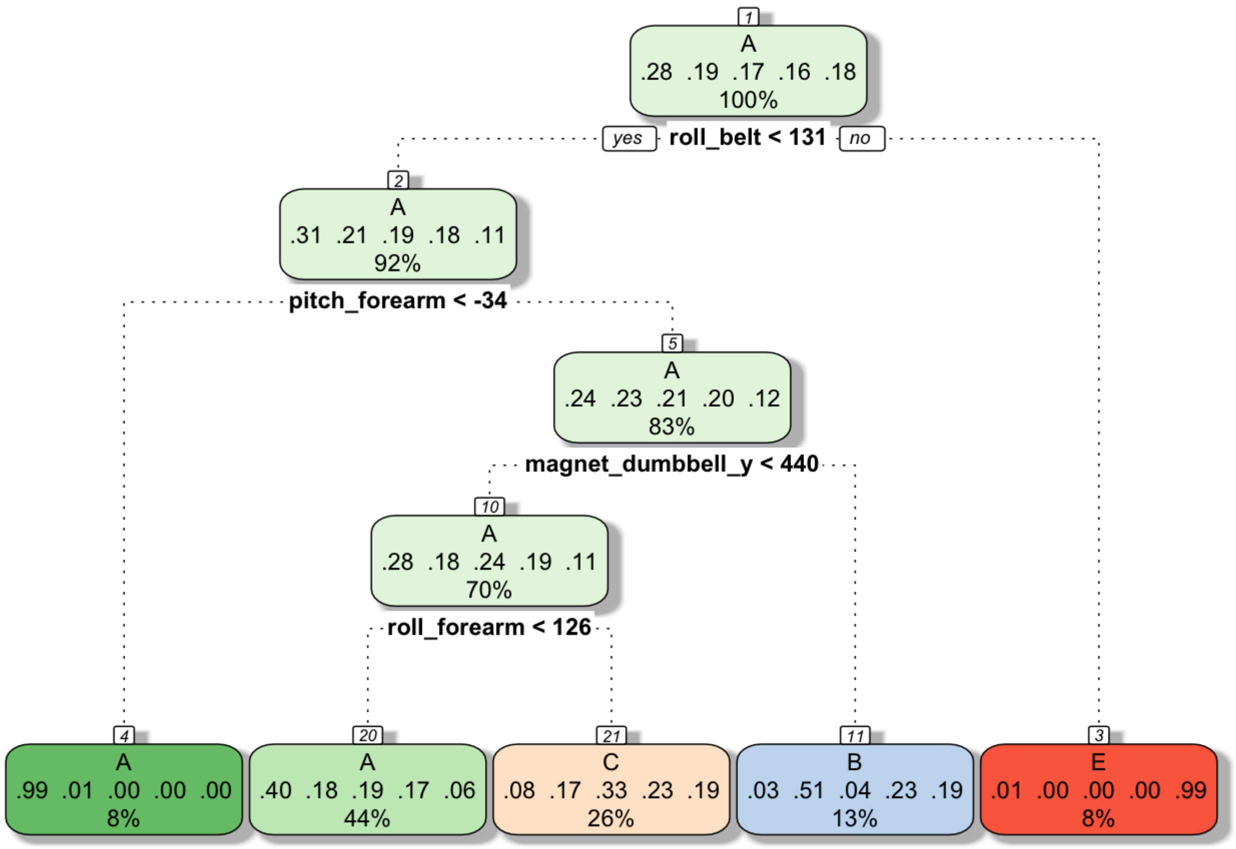
Classification Tree

We set the seed before we begin in order to ensure reproducibility. We then can train a classification tree model on our subset of the training data. This model uses k=5, or 5 folds.

```
set.seed(8)
modFit1 <- train(classe ~. , method="rpart", data=train, trControl = trainControl(method="cv"
, number=5))
```

This plot shows us the classification structure this model has determined.

```
fancyRpartPlot(modFit1$finalModel)
```



Rattle 2018-Sep-23 00:02:06 rebeccakotula

We can then use this model to predict the class of the "test" subset of our original training data. The confusion matrix for these predictions gives us the overall accuracy of our model.

```
pred1 <- predict(modFit1, newdata=test)
confRpart <- confusionMatrix(test$classe, pred1)
confRpart$overall[1]
```

```
##  Accuracy
## 0.4963466
```

The accuracy shown by the confusion matrix is 0.4963466, which indicates that the classification tree method does not predict our outcome very well. This means that we would predict that the out-of-sample error rate to be 0.5036534.

Random Forests

We set the seed again before we begin in order to ensure reproducibility. Now we will train a random forest model.

```
set.seed(8)
modFit2 <- train(classe ~. , method="rf", data=train)
```

We can then use this model to predict the class of the "test" subset of our original training data. The confusion matrix for these predictions gives us the overall accuracy of our model.

```
pred2 <- predict(modFit2, newdata=test)
confRF <- confusionMatrix(test$classe, pred2)
confRF$overall[1]
```

```
##  Accuracy
## 0.9942226
```

The accuracy shown by the confusion matrix is 0.9942226, meaning that our random forest is a very good predictor of the class. This means that we would predict that the out-of-sample error rate to be 0.0057774. So far this is the more accurate of the two models.

Gradient Boosting

Even though our accuracy was quite high with the random forest model, we will try one more method- gradient boosting. Again, we set the seed before we begin in order to ensure reproducibility.

```
set.seed(8)
modFit3 <- train(classe ~. , method="gbm", data=train, trControl=trainControl(method="cv", nu
mber=5), verbose=FALSE)
```

Again, we use the model to predict the class of the “test” subset of our original training data. The confusion matrix for these predictions gives us the overall accuracy of our model.

```
pred3 <- predict(modFit3, newdata=test)
confGB <- confusionMatrix(test$classe, pred3)
confGB$overall[1]
```

```
## Accuracy
## 0.959898
```

The accuracy shown by the confusion matrix is 0.959898, meaning that our gradient boosting model is also a good predictor of the class, but does not achieve quite as high of an accuracy as the random forest model. This means that we would predict that the out-of-sample error rate as 0.040102.

Further Exploration

We can view summaries of each of our models to see the amount of cases they assigned to each class. It is obvious in these tables that the first model, the classification tree, had a much different outcome than the other two models.

```
summary(pred1)
```

```
##    A    B    C    D    E
## 3079  756 1559    0  491
```

```
summary(pred2)
```

```
##    A    B    C    D    E
## 1681 1133 1031  957 1083
```

```
summary(pred3)
```

```
##    A    B    C    D    E 
## 1694 1124 1071  957 1039
```

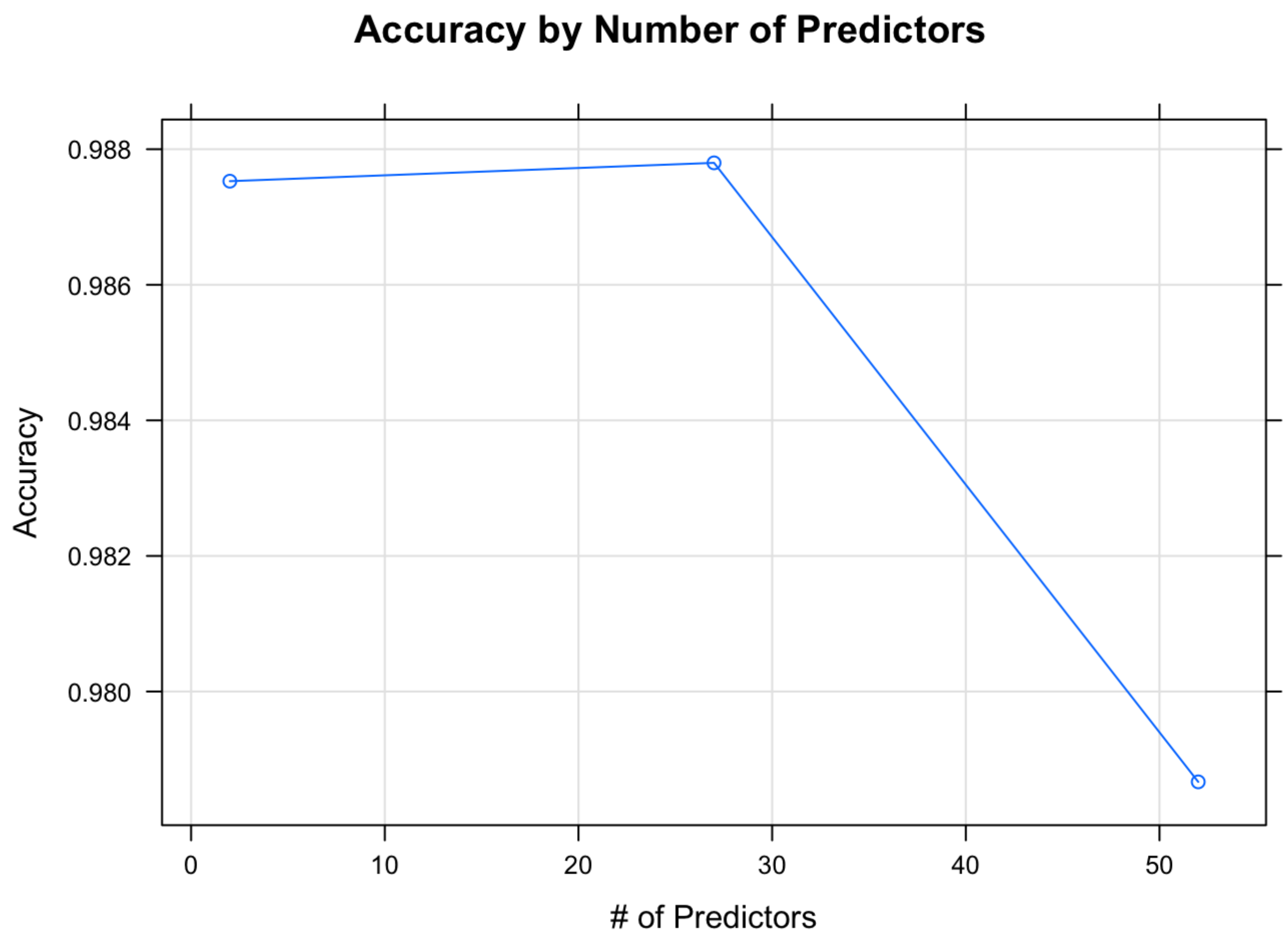
Now that we have determined that the random forest method is the best method, it can be interesting to look at the variables' importance in the model. This shows us the top 20 variables that have the highest impact on predicting the class.

```
varImp(modFit2)
```

```
## rf variable importance
## 
##   only 20 most important variables shown (out of 52)
## 
##                      Overall
## roll_belt            100.000
## pitch_forearm         58.511
## yaw_belt              54.753
## pitch_belt            43.926
## magnet_dumbbell_z     43.469
## magnet_dumbbell_y     41.963
## roll_forearm          41.959
## accel_dumbbell_y      20.495
## accel_forearm_x       18.605
## roll_dumbbell         18.001
## magnet_dumbbell_x     17.888
## magnet_belt_z         15.063
## total_accel_dumbbell  13.897
## accel_belt_z          13.627
## accel_dumbbell_z      13.611
## magnet_forearm_z      13.090
## magnet_belt_y         12.512
## yaw_arm               11.498
## gyros_belt_z           9.781
## magnet_belt_x          9.622
```

If we look at a plot of the model's accuracy determined by the number of predictors used, we see that there is a peak in the middle, but using too many predictors begins to lower the accuracy.

```
plot(modFit2, main="Accuracy by Number of Predictors", xlab="# of Predictors", ylab="Accurac
y")
```



Prediction

Since it achieved the highest accuracy, we will be using the random forest model to predict our actual test cases.

```
preds <- predict(modFit2, newdata=testingClean)
preds
```

```
##  [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
```

These are the predicted classes of the 20 test cases.