Practical ML Course Project

Akarsh Katiyar

21/10/20

->Introduction

In today's era, using various devices such as Nike FuelBand or Fitbit, now it is possible to collect and process a very large Quanti ty of data related to personal activity by monitoring different conditions using these devices and these datas are Relative ly I nexpensive.Hence, these type of devices now are a part of the quantified self movement which is a small group of enthusiast s who wants to take measureme nts about them selves more often to inhance their health conditions, or to find patterns in their behave or of exe rcises that have a huge impact on their health, or s imply because they are tech geeks. More often, One thing that these people regularly do is that they try to quanti fy how much of a particular activity they do in a day or in a we ek, but they rarely quantify how well they do it. So, in this project, my goal is to use these datas from accelerometers on the belt, forearm, arm, and dumb ell of 6 partici pants. These participants were asked to perform barbell lifts correctly and incorr ectly in five differ rent ways.

The aim of the project is for us to predict the exact manner in which they performed the exercise. This is the `classe` variable in the training
set.

Data description

The outcome variable of the data set is `classe` , a factor variable that have five different levels. For the given data set, all the six participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different methods which defines 5 different classes:

Class A : Exactly same as the specification
Class B : Throwing the elbows to the front
Class C : Lifting the dumbbell only halfway
Class D : Lowering the dumbbell only halfway
Class E : Throwing the hips to the front

Initial configuration

For the initial configuration, I am going to install and load some required packages and initialize some of the variables. Following R code installs and loads the required package for this project.

```
#Data variables
training.file <- './data/pml-training.csv'
test.cases.file <- './data/pml-testing.csv'
training.url <- 'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
test.cases.url <- 'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'
#Directories
if (!file.exists("data")){
dir.create("data")
}
if (!file.exists("data/submission")){
dir.create("data/submission")
}
#R-Packages
IscaretInstalled <- require("caret")
## Loading required package: caret
## Warning: package 'caret' was built under R version 3.6.3
## Loading required package: lattice
## Loading required package: ggplot2
if(!IscaretInstalled){
install.packages("caret")
library("caret")
}
IsrandomForestInstalled <- require("randomForest")
## Loading required package: randomForest
## Warning: package 'randomForest' was built under R version 3.6.3
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
```

```
## margin
if(!IsrandomForestInstalled){
install.packages("randomForest")
library("randomForest")
}
IsRpartInstalled <- require("rpart")
## Loading required package: rpart
if(!IsRpartInstalled){
install.packages("rpart")
library("rpart")
}
IsRpartPlotInstalled <- require("rpart.plot")
## Loading required package: rpart.plot
## Warning: package 'rpart.plot' was built under R version 3.6.3
if(!IsRpartPlotInstalled){
install.packages("rpart.plot")
library("rpart.plot")
}
# Set seed for reproducability
set.seed(9999)
```

Data processing

Moving to data processing section, here I am going to download the data and then process them.Then I will perform some basic transformations and cleanup methods on the downloaded data so that `NA` values are removed from the raw data.Apart from this, I will also remove some of the irrelevant columns such as `user_name` , `raw_timestamp_part_1` , `raw_timestamp_part_2` , `cvtd_timestamp` , `new_window` , and `num_window` in the subset.
The `pml-training.csv` data set is used to conceive training sets and testing sets. The `pml-test.csv` data is used to predict and answer the 20 questions given in the quiz based on the trained model. Following R code will download, clean and process the data set

```
# Download data
download.file(training.url, training.file)
download.file(test.cases.url,test.cases.file )
# Clean data
training <-read.csv(training.file, na.strings=c("NA","#DIV/0!", ""))
testing <-read.csv(test.cases.file , na.strings=c("NA", "#DIV/0!", ""))
training<-training[,colSums(is.na(training)) == 0]
testing <-testing[,colSums(is.na(testing)) == 0]
# Subset data
training <-training[,-c(1:7)]
testing <-testing[,-c(1:7)]
```

Cross-validation

Now, in this section, I am going to perform cross-validation by splitting the training data in training and testing data. The training data set consists of 75% of the total data set and the testing data set consists of remaining 25% of the data set. Following R code is going to perform cross-validation process.

```
subSamples <- createDataPartition(y=training$classe, p=0.75, list=FALSE)
subTraining <- training[subSamples, ]
subTesting <- training[-subSamples, ]
```

Expected out-of-sample error

Now, as we know that the expected out-of-sample error corresponds to the quantity: 1-accuracy in the cross-validation data. Similarly, the Accuracy is the proportion of the total correct classified observation over the total sample in the subtesting
data set. Also, the Expected accuracy is the calculated expected accuracy of the out-of-sample data set (also known as the original testing data set). Hence, the expected value of the out-of-sample error will corresponds to the expected number of missclassified observations or the total observations in the Test data set, which is the quantity: 1-accuracy found from the cross-validation data set.

Exploratory analysis

Now, moving to the exploratory analysis of the data set, we know that the variable `classe` contains 5 levels. The plot of the outcome variable of these 5 levels shows the frequency of each levels in the subTraining data. So, we will plot the frequency of each level using the following R code.

```
plot(subTraining$classe, col="orange", main="Levels of the variable classe", xlab="classe levels", ylab
```

```
="Frequency")
```
From the plot shown above, we can say that the Level A is the most frequent ‘classe’ and the level D is the least frequent one.

Prediction models

Moving to the Prediction model, here I am going to apply a decision tree and random forest to the data.

Decision tree

Following R code is being used to draw the decision tree.

```
# Fit model
modFitDT <- rpart(classe ~ ., data=subTraining, method="class")
# Perform prediction
predictDT <- predict(modFitDT, subTesting, type = "class")
# Plot result
rpart.plot(modFitDT, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```

The confusion matrix below shows all the errors that comes from the prediction algorithm.

```
confusionMatrix(predictDT, subTesting$classe)
## Confusion Matrix and Statistics
##
## Reference
## Prediction A B C D E
## A 1247 212 23 83 30
## B 32 530 73 23 73
## C 35 96 695 112 121
## D 60 66 46 532 46
## E 21 45 18 54 631
##
## Overall Statistics
##
## Accuracy : 0.7412
## 95% CI : (0.7287, 0.7534)
## No Information Rate : 0.2845
## P-Value [Acc > NIR] : < 2.2e-16
##
## Kappa : 0.6712
##
## Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
## Class: A Class: B Class: C Class: D Class: E
## Sensitivity 0.8939 0.5585 0.8129 0.6617 0.7003
## Specificity 0.9008 0.9492 0.9101 0.9468 0.9655
## Pos Pred Value 0.7818 0.7250 0.6563 0.7093 0.8205
## Neg Pred Value 0.9553 0.8996 0.9584 0.9345 0.9347
## Prevalence 0.2845 0.1935 0.1743 0.1639 0.1837
## Detection Rate 0.2543 0.1081 0.1417 0.1085 0.1287
## Detection Prevalence 0.3252 0.1491 0.2159 0.1529 0.1568
## Balanced Accuracy 0.8974 0.7538 0.8615 0.8043 0.8329
```

Random forest

Following R code is used to predict using the random forest.

```
# Fit model
modFitRF <- randomForest(classe ~ ., data=subTraining, method="class")
# Perform prediction
predictRF <- predict(modFitRF, subTesting, type = "class")
```

The confusion matrix below shows all the errors that comes from the prediction algorithm.

```
confusionMatrix(predictRF, subTesting$classe)
## Confusion Matrix and Statistics
##
## Reference
## Prediction A B C D E
## A 1393 5 0 0 0
## B 2 943 2 0 0
## C 0 1 853 8 0
```

```
## D 0 0 0 795 2
## E 0 0 0 1 899
##
## Overall Statistics
##
## Accuracy : 0.9957
## 95% CI : (0.9935, 0.9973)
## No Information Rate : 0.2845
## P-Value [Acc > NIR] : < 2.2e-16
##
## Kappa : 0.9946
##
## Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
## Class: A Class: B Class: C Class: D Class: E
## Sensitivity 0.9986 0.9937 0.9977 0.9888 0.9978
## Specificity 0.9986 0.9990 0.9978 0.9995 0.9998
## Pos Pred Value 0.9964 0.9958 0.9896 0.9975 0.9989
## Neg Pred Value 0.9994 0.9985 0.9995 0.9978 0.9995
## Prevalence 0.2845 0.1935 0.1743 0.1639 0.1837
## Detection Rate 0.2841 0.1923 0.1739 0.1621 0.1833
## Detection Prevalence 0.2851 0.1931 0.1758 0.1625 0.1835
## Balanced Accuracy 0.9986 0.9963 0.9977 0.9942 0.9988
```

Conclusion

Result

From comparing the confusion matrices of the random forest algorithm and decision tree algorithm, we can conclude that the Random Forest algorithm performs better than the decision tree algorithm. The accuracy of the Random Forest model is 0.995 (95% CI: (0.993, 0.997)) whereas the accuracy of the Decision Tree model is 0.739 (95% CI: (0.727, 0.752)). Hence, the random Forest model is choosen.

Expected out-of-sample error

While calculating the expected out-of-sample error, we came to know that 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 crossvalidation

set. Our Test data set has 20 cases. So, with an accuracy of 99.5% on our cross-validation data, we can expect that only a very few, or none, of the test samples will be missclassified.