Gurinder

Coursera-Practical Machine Learning Assignment

Problem Description

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. This project uses data (see sources) 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. The goal of the project is to do the prediction about the manner in which participants did the exercise.

Sources

Data

The data both for training and testing in this project come from the source: Human Activity Recognition – HAR's Weight Lifting Exercises Dataset (http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har))*. HAR has many potential applications, like: elderly monitoring, life log systems for monitoring energy expenditure and for supporting weight-loss programs, and digital assistants for weight lifting exercises.

*Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Description of Data

The complete dataset for our experiments consists of 19622 observations for 6 participants on 160 variables. The testing data contains 20 test instances for prediction by the learning model. The 6 participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions.

- Exactly according to the specification (Class A)
- Throwing the elbows to the front (Class B)
- Lifting the dumbbell only halfway (Class C)
- Lowering the dumbbell only halfway (Class D)
- Throwing the hips to the front (Class E)

The classification is done for the above five classes.

Packages

The different packages were used in this project in R environment.

- caret

- randomForest
- kernlab
- corrplot
- knitr

Loading the Packages

The following code will load the packages used in this particular project.

```
## Loading of packages (assumed already installed) Step of Cousera's Practical
## Machine Learning Project
library(caret)
library(kernlab)
library(knitr)
library(corrplot)
library(randomForest)
```

Acquiring the Data

To get the data from HAR project, two files: training and testing files in csv file format were downloaded from the URL "" and stored in the working directory "PMLData".

```
## Data Acquiring Step of Cousera's Practical Machine Learning Project
## Check the existence of PMLData directory for holding training and test data
## and if it does not exist then create new one

if (!file.exists("PMLData")) {
    dir.create("PMLData")
}

## Give Source file address link and the destination file
trainingFileAddress <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-tra
ining.csv"
trainingFile <- "./PMLData/pml-training.csv"
testingFileAddress <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-test
ing.csv"
testingFile <- "./PMLData/pml-testing.csv"

## Download the training and testing data files
download.file(trainingFileAddress, destfile = trainingFile)
download.file(testingFileAddress, destfile = testingFile)
```

Loading the Data

We now need to read the data from these csv files for loading in R environment. The following code enables the training data to be loaded into R.

```
## Loading the training data into R by reading the csv file for training
missing_values = c("NA","#DIV/0!", "","  ");
trainingData <- read.csv("./PMLData/pml-training.csv", na.strings= missing_values)
```

Cleaning the Data

Filtering: Filters transform datasets by removing or adding attributes, re-sampling the dataset, removing examples and so on. R provides useful support for data preprocessing, which is an important step in machine learning. NA values and identifier columns without much significance (first eight) in the training data were removed from the data set to minimize the noise in the model.

```
## Clean the training data
NA_trainingData <- apply(trainingData, 2, function(x) {sum(is.na(x))})
cleanedTrainingData <- trainingData[,which(NA_trainingData == 0)]

## Remove columns which can be removed such as identifier columns
cleanedTrainingData <- cleanedTrainingData[8:length(cleanedTrainingData)]

# Cleaned Data dimensions
dim(cleanedTrainingData)
```

```
[1] 19622    53
```

Building a Learning Model

Model Classifier

Classifiers: A classifier model is an arbitrary complex mapping from all-but-one dataset attributes to the class attribute. The specific form and creation of this mapping, or model, differs from classifier to classifier. Classifiers are at the core of machine learning approaches. The results for analysis in current project setting were obtained by using the “Random Forests” classifier in the experimental settings, which is explained here briefly.

Random Forests

Random forests is a learning method for classification, regression and other tasks which is based on construction of host of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. It assumes that the user knows about the construction of single classification trees and random forests grow many classification trees. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree “votes” for that class. The forest chooses the classification having the most votes (over all the trees in the forest). For our experimentation, random forest model was selected because it works for balancing error in class population unbalanced data sets. A correlation plot was generated to reflect the relationship between variables. . There are seven visualization methods (parameter method). In corrplot package, named “circle”, “square”, “ellipse”, “number”, “shade”, “color”, “pie”. I used “square” method for the correlation plot. In the plot the dark red is used to represent a highly negative relationship between the

variables and blue colour show a positive relationship.

Cross Validation

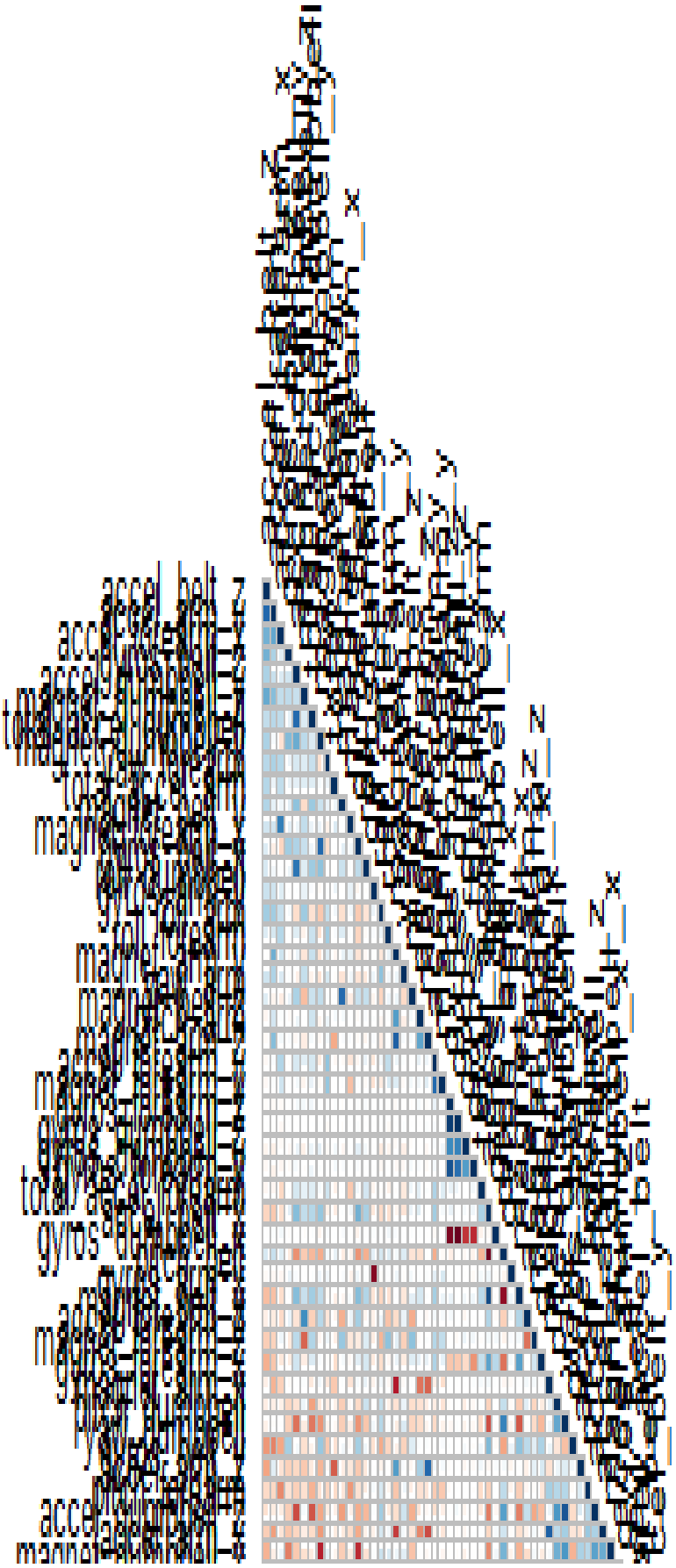
Cross-validation, sometimes called rotation estimation, is a widespread method for assessing the generalization ability of a model in order to tune a regularization parameter or other hyper-parameters of a learning process. The use of cross-validation requires setting yet an additional parameter, the split ratio. The dataset is randomly reordered and then split into n folds of equal size. One fold is used for testing and the other n-1 folds are used for training the classifier in every iteration. The test results are averaged over all folds to get the cross-validation estimate of the accuracy.

```
## Building Model for Predictions
## Split the cleaned training data into training and cross validation data for rati
o 70:30
inTrain <- createDataPartition(y = cleanedTrainingData$classe, p = 0.8, list = FALS
E)
trainingDataSplit <- cleanedTrainingData[inTrain, ]
crossvalidationSplit <- cleanedTrainingData[-inTrain, ]
```

A correlation plot was generated to reflect the relationship between variables. There are seven visualization methods (parameter method). In corrplot package, named “circle”, “square”, “ellipse”, “number”, “shade”, “color”, “pie”. I used “square” method for the correlation plot. In the plot the dark red is used to represent a highly negative relationship between the variables and blue colour show a positive relationship.

```
correlationMat <- cor(trainingDataSplit[, -length(trainingDataSplit)])
corrplot(correlationMat, order = "FPC", method = "square", type = "lower", tl.cex =
 0.8,  tl.col = rgb(0, 0, 0))
```

Correlation Plot



Fitted Model

The fitted model based on random forest is shown here.

```
## Building the model based on random forests classifier
randomForestModel <- randomForest(classe ~ ., data = trainingDataSplit)
randomForestModel
```

```

Call:
 randomForest(formula = classe ~ ., data = trainingDataSplit)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 7

        OOB estimate of  error rate: 0.41%
Confusion matrix:
     A    B    C    D    E  class.error
A 4463    1    0    0    0 0.0002240143
B    9 3024    5    0    0 0.0046082949
C    0   16 2719    3    0 0.0069393718
D    0    0   23 2548    2 0.0097162845
E    0    0    2    4 2880 0.0020790021
```

Performance

The performance of classifiers can be measured by various approaches. In the simplest measures, the performance can be measured by counting the proportion of correctly predicted examples in an unseen test dataset. This value is termed as accuracy, which is also represented sometimes in terms of ErrorRate. Both these measures are in place and found in literature. The training data set was split up in 80:20 ratios into two sets: training and cross validation sets.

From the fitted model, we observe that the OOB estimate of error rate: 0.41%. In present experimentation to train the model, the training data set was split up in 80:20 ratios into two sets: training and cross validation sets.

```
## Fitting the model on the 20% of cross validation data
predictionCV <- predict(randomForestModel, crossvalidationSplit)
confusionMatrix(crossvalidationSplit$classe, predictionCV)
```

```
Overall Statistics

               Accuracy : 0.9957
                 95% CI : (0.9931, 0.9975)
    No Information Rate : 0.2865
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9945
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9929   0.9987   0.9913   0.9969   1.0000
Specificity            1.0000   0.9965   0.9994   0.9991   0.9997
Pos Pred Value         1.0000   0.9855   0.9971   0.9953   0.9986
Neg Pred Value         0.9971   0.9997   0.9981   0.9994   1.0000
Prevalence             0.2865   0.1909   0.1754   0.1637   0.1835
Detection Rate         0.2845   0.1907   0.1738   0.1631   0.1835
Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
Balanced Accuracy      0.9964   0.9976   0.9953   0.9980   0.9998
```

The trained model based on random forests method when run on cross-validation data has given 99.57% accuracy.

Predictions on Test Data

The goal of the project was to predict the manner in which participants did the exercise. This is the “classe” variable in the training set. The data set for testing is read from csv file and cleaned for testing on built model to predict the classification. The dataset for testing contains 20 instances and these are classified in one of the five groups.

A separate data set was then loaded into R and cleaned in the same manner as before. The model was then used to predict the classifications of the 20 results of this new data.

```
## Using the model on testing data of 20 cases
testingData <- read.csv("./PMLData/pml-testing.csv", na.strings= c("NA","","# "))
NA_testingData <- apply(testingData, 2, function(x) {sum(is.na(x))})
cleanedTestingData <- testingData[,which(NA_testingData == 0)]
cleanedTestingData <- cleanedTestingData[8:length(cleanedTestingData)]

## Predicting the classification for the test data
testDataPrediction <- predict(randomForestModel, cleanedTestingData)
testDataPrediction
```

```
1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
 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
```

Conclusion

In this work the investigation was done on qualitative activity recognition on the example of assessing the quality of execution of weight lifting exercises. For this project, the model built by me was based on random forest classifier. The classifier is able to handle large number of variables and balances biases and trade-offs by adopting a balanced model. The model generated built a highly accurate classifier as we could achieve 99.57% accuracy.