Practical Machine Learning - Project

Suresh

November 27, 2017

Background

The data that is provided for the assignment is the 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. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset). The training data has the the variable 'classe' which defines how well the exercise was performed.

Our outcome variable is classe, a factor variable with 5 levels. For this data set, "participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 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)?

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.

The goal of your project is to predict the manner in which they did the exercise. The prediction needs to be done on the 20 cases in the testing data set. 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)]

The test data are available here: [https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)]

The data for this project come from this source: [http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har)].

Summary of analysis

The steps to analyse the data and do the prediction are as follows

- Download the provided datasets and load into R
- Do some exploratory data analysis on the data to under the structure and issues
- Clean up the data to remove columns having NAs, variables which add no value to the analysis and variables which are little or no variance.
- Split the training dataset into training and testing datasets to do cross validation.
- Rin the Classification Tree and Random Forest methods on the training dataset and create the models.
- Do predictions for each model and geneate the accuracy metrics.
- Using the model with the better accuracy, run prediction on the testing dataset provided and do the prediction for the test cases.

Download the data and import necessary libraries

```
#Get libraries needed
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.2
```

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

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

```
#Download the data and read into R
modelDf <- read.csv("pml-training.csv")
predictDf <- read.csv("pml-testing.csv")
```

Exploratory Data Analysis

```
summary(modelDf$classe)
```

```
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
#Check how many columns have all NA values
sum(colSums(is.na(modelDf)) == 0)
```

```
## [1] 93
```

```
#Check how many columns have no variance or near zero variance
nearzerovar <- nearZeroVar(modelDf, saveMetrics=TRUE)
sum(nearzerovar$zeroVar)
```

```
## [1] 0
```

```
sum(nearzerovar$nzv)
```

```
## [1] 60
```

```
#check for columns that may not add value to the model
names(modelDf[,c(1:7)])
```

```
## [1] "X"                    "user_name"            "raw_timestamp_part_1"
## [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
## [7] "num_window"
```

The analysis shows that

- the variable 'classe' has 5 levels.
- there are a large number of columns having all NA values and these need to be deleted because they may interfere with the model creation.
- there are no columns with 0 variance
- there are 60 columns with near zero variance. These can also be deleted before creating the model.
- The first 7 coulumns are names and ids and can be removed.

Data Cleanup

The data will be cleaned up to

- delete columns which have a large percentage of NAs (over 60%)
- columns which have near zero variance and
- the first 7 columns

```
#remove columns with more than 60% NAs,
#create logical vector for columns where the NA are less than 50%
naCount <- colSums(is.na(modelDf))/nrow(modelDf) < 0.4
#apply the vector on the dataframe to only select the columns where the % is less than 50%
modelDf <- modelDf[,naCount]
dim(modelDf)
```

```
## [1] 19622    93
```

```
#The first 7 columns may not add any value to the prediction
modelDf <- modelDf[,-c(1:7)]
dim(modelDf)
```

```
## [1] 19622    86
```

```
#remove columns whose variace is near zero
nearzerovar <- nearZeroVar(modelDf, saveMetrics=TRUE)
modelDf <- modelDf[,!nearzerovar$nzv]
dim(modelDf)
```

```
## [1] 19622    53
```

The number of variables has been reduced from 160 to 53.

Splitting data for Cross validation

The dataset will be split into a training and testing data sets to do cross validation.

```
#Split the training data into training and testing sets at ratio of 60:40 to do cross validatio
n.
inTrain <- createDataPartition(modelDf$classe, p=0.6, list=FALSE)
training <- modelDf[inTrain,]
dim(training)
```

```
## [1] 11776    53
```

```
testing <- modelDf[-inTrain,]
dim(testing)
```

```
## [1] 7846   53
```

Cross Validation

The trainControl() function will be used to do the cross validation and PCA and the result will be used in the model building.

```
traincontrol <- trainControl(method="cv", numbe=5, verboseIter = FALSE, preProcOptions = "pca",
allowParallel = TRUE)
```

Create a prediction model using the Classification Tree

```
#Run the Classification Tree
fitTree <- train(classe ~ ., data=training, method='rpart', trControl=traincontrol)
#Predict the classifications for testing dataset
predictTree <- predict(fitTree, testing)
#Create confusion matrix to see the accuracy
confusionMatrix(predictTree, testing$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2003  604  635  580  186
##          B   39  529   47  216  200
##          C  156  385  686  490  383
##          D    0    0    0    0    0
##          E   34    0    0    0  673
## 
## Overall Statistics
## 
##                Accuracy : 0.4959
##                  95% CI : (0.4848, 0.507)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.342
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8974  0.34848  0.50146   0.0000  0.46671
## Specificity            0.6429  0.92067  0.78172   1.0000  0.99469
## Pos Pred Value         0.4998  0.51309  0.32667      NaN  0.95191
## Neg Pred Value         0.9403  0.85488  0.88131   0.8361  0.89228
## Prevalence             0.2845  0.19347  0.17436   0.1639  0.18379
## Detection Rate         0.2553  0.06742  0.08743   0.0000  0.08578
## Detection Prevalence   0.5108  0.13140  0.26765   0.0000  0.09011
## Balanced Accuracy      0.7701  0.63458  0.64159   0.5000  0.73070
```

Create a prediction model using Random Forest

```
#Do the same for random forest
fitRf <- train(classe ~ ., data=training, method='rf', trControl=traincontrol)
```

```
## Warning: package 'randomForest' was built under R version 3.4.2
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
## 
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
predictRf <- predict(fitRf, testing)
confusionMatrix(predictRf, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2228   22    0    0    0
##          B    3 1490   14    0    0
##          C    0    6 1350   29    0
##          D    0    0    4 1256    2
##          E    1    0    0    1 1440
##
## Overall Statistics
##
##                Accuracy : 0.9895
##                  95% CI : (0.987, 0.9917)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9868
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9816   0.9868   0.9767   0.9986
## Specificity            0.9961   0.9973   0.9946   0.9991   0.9997
## Pos Pred Value         0.9902   0.9887   0.9747   0.9952   0.9986
## Neg Pred Value         0.9993   0.9956   0.9972   0.9954   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1899   0.1721   0.1601   0.1835
## Detection Prevalence   0.2868   0.1921   0.1765   0.1608   0.1838
## Balanced Accuracy      0.9971   0.9894   0.9907   0.9879   0.9992
```

The above shows that there is an accuracy for around 50% for Classification Tree and around 99% for Random Forest. The out of sample errors are 1 - Acccuracy and are 0.5 and 0.01 respectively. This shows that Random Forest prediction is better by far and can be used to predict the 20 test cases.

Final Predictions

The final prediction is performed by applying the above created Random Forest model to the ‘pml-testing.csv’ which contains the 20 test cases.

```
# Choose the best model out of the list and predict using the model and the testing file.
finalPredictions <- predict(fitRf, predictDf)
finalPredictions
```

```
##  [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
```

```
#Add the predictions to the testing data set and display it
predictDf <- cbind(predictDf, classe.predicted =finalPredictions)
predictDf[,c("user_name", "problem_id", "classe.predicted")]
```

```
##    user_name problem_id classe.predicted
## 1      pedro          1                B
## 2     jeremy          2                A
## 3     jeremy          3                B
## 4     adelmo          4                A
## 5     eurico          5                A
## 6     jeremy          6                E
## 7     jeremy          7                D
## 8     jeremy          8                B
## 9   carlitos          9                A
## 10   charles         10                A
## 11  carlitos         11                B
## 12    jeremy         12                C
## 13    eurico         13                B
## 14    jeremy         14                A
## 15    jeremy         15                E
## 16    eurico         16                E
## 17     pedro         17                A
## 18  carlitos         18                B
## 19     pedro         19                B
## 20    eurico         20                B
```

The final predictions are displayed above.