Project ML

Jimmy

7/3/2020

Synopsis

In this project we will be building a prediction model to know how a certain user is lifting weights based on data obtained from an accelerometer.

The dataset consists on 5 classes:

. The subject is lifting weights exactly according to the specification (Class A).

. Throwing the elbow 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).

For more information and description about the dataset, see the official website: http://groupware.les.inf.puc-rio.br/har

Introduction

This project is being carried out in completion of the "Practical Machine Learning" Coursera course.

A dataset of measurement data has been provided by the course. The dataset is comprised of measurements of acceleration made by individuals who are carrying out one of five classes of physical activity. According to this project's instructions, the measurements are made using devices worn on the belt, forearm, arm, and a dumbbell.

Additional information the dataset is available here: http://groupware.les.inf.puc-rio.br/har

My task is to create a model that can predict the which class of activity is being done.

Download dataset

```
if(!file.exists("training.csv")) {
  download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
                "training.csv")
}
if(!file.exists("test.csv")) {
  download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
                "test.csv")
}
```

Prepare data for model training, cross validation, and testing.

```r
set.seed(2738255)
df <- read.csv("training.csv", stringsAsFactors=TRUE)
# Remove the column named "X", which represents the observation number and is not relevant to preding o
df <- df[,which(names(df) != "X")]
# Let's ignore the data dimensions that have near zero variance.  This eliminates rarely varying
# parameters, which are presumably not very useful for prediction.  In this were a more thorough
# study, the effect of removing these parameters would be determined.
df <- df[,-nearZeroVar(df)]
# Let's partition the training data into training and cross-validation sets.  The
# cross-validation set is a hold-out set that allows us to measure the accuracy of the model.
inTrain = createDataPartition(df$classe, p = 0.8)[[1]]
training = df[ inTrain,]
cving = df[-inTrain,]
testing <- read.csv("test.csv")
testing <- testing[, which(names(df) != "X")]
```

Assess data

Let's inspect the data to see if there's any missing values.

```r
print("Dimensions of training data:")
```

```
## [1] "Dimensions of training data:"
```

```r
paste(dim(df))
```

```
## [1] "19622" "99"
```

```r
print("Dimensions of training data, removing observations with NA values:")
```

```
## [1] "Dimensions of training data, removing observations with NA values:"
```

```r
print(dim(df[rowSums(is.na(df)) == 0, ]))
```

```
## [1] 406  99
```

```r
print("Dimensions of training data, removing data dimensions with NA values:")
```

```
## [1] "Dimensions of training data, removing data dimensions with NA values:"
```

```r
print(dim(df[,colSums(is.na(df)) == 0]))
```

```
## [1] 19622    58
```

Given that only 406 observations contain zero NAs, it seems that NAs were included by design.

Build model

Now that the data is partitioned into distinct sets for training, cross validating, and testing, we can build our model using the training set. The following three models were the ones I wished to try:

1. The package rpart allows a model to be built that contains NA values. Since the dataset has many NAs, this simplifies the problem of how to use the provided data.
2. Random forest with observations removed if NA is present.

3. Random forest with dimensions removed if NA is present. I am assuming that the testing set and training set both have the same NA values. If the testing set were to have NAs, we could try imputing the missing values. (In this assignment, this approach was not necessary.)

I ruled out 2. because so few training examples apply to it. Furthermore, it could not be applied to any test examples that contain NA values.

Build rpart model

```
# Method 1
rpartModel <- rpart(classe ~ ., data=training, method="class")
confusionMatrix(predict(rpartModel, cving, type="class"),
                cving$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1072   27    6    2    0
##          B   34  631   44   41    0
##          C   10   95  616   91    3
##          D    0    6    8  410   36
##          E    0    0   10   99  682
##
## Overall Statistics
##
##                Accuracy : 0.8695
##                  95% CI : (0.8585, 0.8799)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8348
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9606   0.8314   0.9006   0.6376   0.9459
## Specificity            0.9875   0.9624   0.9386   0.9848   0.9660
## Pos Pred Value         0.9684   0.8413   0.7558   0.8913   0.8622
## Neg Pred Value         0.9844   0.9597   0.9781   0.9327   0.9875
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2733   0.1608   0.1570   0.1045   0.1738
## Detection Prevalence   0.2822   0.1912   0.2077   0.1173   0.2016
## Balanced Accuracy      0.9741   0.8969   0.9196   0.8112   0.9559
```

At 95% confidence level, the accuracy is in the range of 85.8-88.0%. Let's compare the rpart model's performance with that of a random forest model.

Build random forest model

```
# Method 3: Remove columns with missing values and do train with random forest method
rfTraining <- training[,colSums(is.na(df)) == 0]
rfCving <- cving[, colSums(is.na(df)) == 0]
rfModel <- train(classe ~ ., data=rfTraining, method="rf")
confusionMatrix(predict(rfModel, rfCving), rfCving$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    0    0    0    0
##          B    0  759    1    0    0
##          C    0    0  678    0    0
##          D    0    0    5  643    1
##          E    0    0    0    0  720
##
## Overall Statistics
##
##                Accuracy : 0.9982
##                  95% CI : (0.9963, 0.9993)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9977
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9912   1.0000   0.9986
## Specificity            1.0000   0.9997   1.0000   0.9982   1.0000
## Pos Pred Value         1.0000   0.9987   1.0000   0.9908   1.0000
## Neg Pred Value         1.0000   1.0000   0.9982   1.0000   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1935   0.1728   0.1639   0.1835
## Detection Prevalence   0.2845   0.1937   0.1728   0.1654   0.1835
## Balanced Accuracy      1.0000   0.9998   0.9956   0.9991   0.9993
```

The proof is in the confusion matrix of the cross validation sample. The overall accuracy is determined to be greater than 99.8% at the 95% confidence level.

Because the random forest model has exceptional accuracy and seems equally suited to identifying all classes, it’s the final model selected for this project.