Practical Machine Learning Course Project

John Fortin

7/29/2017

Introduction

The purpose of the project was to take a dataset which took measurements of several people doing one type of dumbbell exercise and determine whether the exercise was done correctly. The measurements were taken using various electronic devices and the participants were instructed to perform the exercise in the correct manner as well as several incorrect manners for a total of five outcomes.

Details for this experiment can be seen at http://groupware.les.inf.puc-rio.br/har

Method

As a normal part of data analysis we will first load and cleanup the data. In this case there are many columns which have missing data points. These will be removed.

```
#Load training and testing datasets
URL_TRAIN="https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
URL_TEST="https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

#Download the datasets if we haven't already downloaded them
if (!file.exists("pml-training.csv"))
    download.file(URL_TRAIN,"pml-training.csv")
if (!file.exists("pml-testing.csv"))
    download.file(URL_TEST,"pml-testing.csv")

#Import the datasets
training_ds = read_csv("pml-training.csv")
testing_ds  = read_csv("pml-testing.csv")
```

```
#Remove the first five columns as they only provide identification traits such as timestamps and subjec
training_clean = training_ds[, -(1:5)]

#Remove columns that have any NA entries.  We only want columbs with clean data
empty_columns  = colSums(is.na(training_clean))
training_clean = training_clean[, empty_columns == FALSE]

#Remove any columns that have no significant or zero variance (NZV)
nzv_columns    = nzv(training_clean, saveMetrics = TRUE)[,4]
training_clean = training_clean[, nzv_columns == FALSE]

#make the same changes to testing data
testing_clean = testing_ds[, -(1:5)]
testing_clean = testing_clean[, empty_columns == FALSE]
testing_clean = testing_clean[, nzv_columns == FALSE]
```

Now we will partition the data from the training dataset into a new training dataset and a cross-validation dataset. This will allow us to determine how effective the training was. We will use 70% for training partitions size.

```
partition = createDataPartition(training_clean$classe,
                                p = 0.7,
                                list = FALSE)
training_data = training_clean[partition,]
validation_data = training_clean[-partition,]
```

Create Model

Since we are looking to categorize data we will start with a Random Forest model. In order to save processing time on reruns we will save the model to disk after the first run and load from disk if the file exists. Removing the file will force the model to be regenerated.

Note: as recommended in the forums I will be using parallel processing to speed up the model generation. See https://github.com/lgreski/datasciencectacontent/blob/master/markdown/pml-randomForestPerformance.md for details.

```
saved_model = "model-rf.RData"
if (file.exists(saved_model)) {
    load(file = saved_model, verbose = FALSE)
}else{
    cluster <- makeCluster(detectCores() - 1)
    registerDoParallel(cluster)
    fitControl <- trainControl(method = "cv",
                               number = 5,
                               allowParallel = TRUE)
    model_rf = train(classe ~ .,
                  data=training_data,
                  method="rf",
                  trControl = fitControl)
    stopCluster(cluster)
    registerDoSEQ()
    save(model_rf, file = saved_model)
}
```

Cross-Validation

Now that we have created the model we need to validate the model with the cross-validation dataset we created earlier.

```
pred = predict(model_rf, validation_data)
confusionMatrix(pred, validation_data$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1139    2    0    0
##          C    0    0 1024    0    0
##          D    0    0    0  964    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9997
##                  95% CI : (0.9988, 1)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.9996
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9981   1.0000   1.0000
## Specificity            1.0000   0.9996   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   0.9982   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   0.9996   1.0000   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1935   0.1740   0.1638   0.1839
## Detection Prevalence   0.2845   0.1939   0.1740   0.1638   0.1839
## Balanced Accuracy      1.0000   0.9998   0.9990   1.0000   1.0000
```

As we can see in the confusion matrix, the accuracy rate for the model is 0.9995 with a P-value on the order of 10^-16. The expected out of sample error should be less than 1 out of 1000.

With an accuracy rate this good we can apply this model to the real testing dataset. The result will be fed into the Course Project Prediction Quiz for evaluation.

```r
pred = predict(model_rf, testing_clean)
pred
```

```
##  [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
```

Results

Based on the model generated and the predicted results from from the test dataset, a 100% accuracy rate was achieved.