Assessment of Exercise Quality

Tim Reid

Thursday, January 15, 2015

We are given a set of measurement data from accelerometers attached to 6 subjects during the performance of an exercise routine, and asked to create a model to assess the quality of their performance of that exercise. The measurement data is captured in a set of 19,622 observations of 160 variables, and includes a variable that characterizes the quality of the exercise performance.

An additional testing data set is provided to test the model.

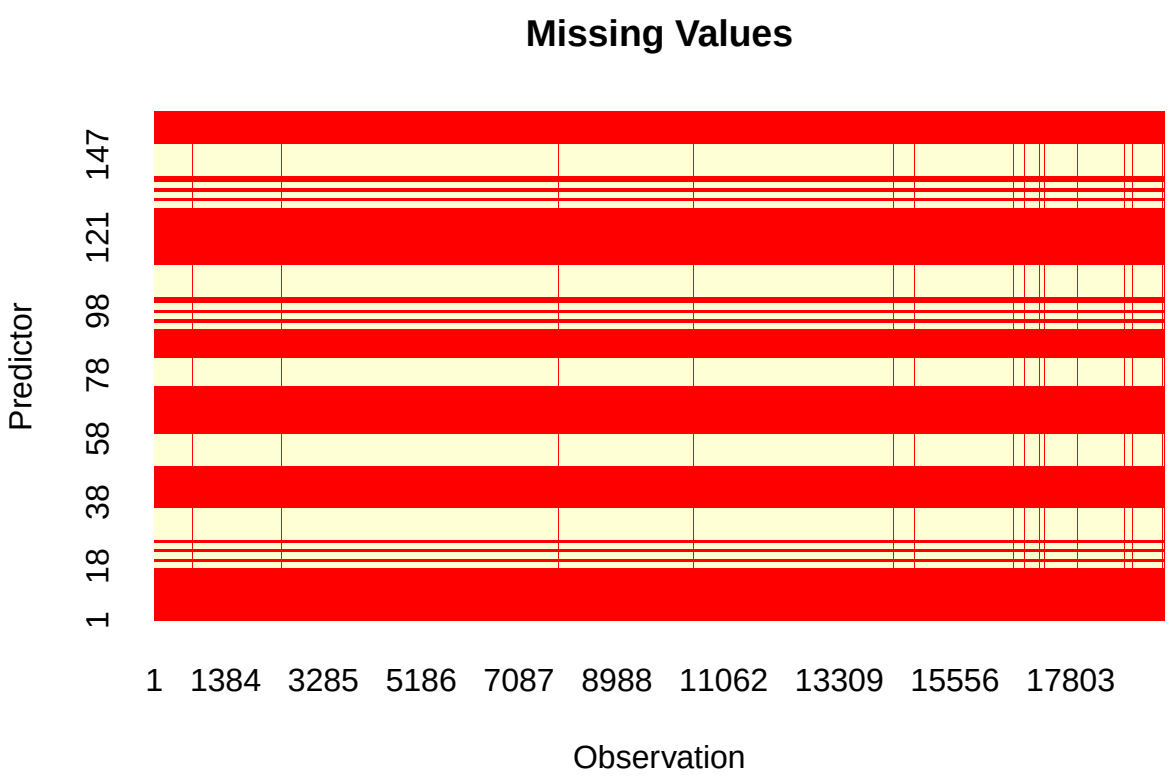
Preliminary Data Exploration

First, access needed libraries and load the training data set.

```
library(lattice)
library(ggplot2)
library(caret)
library(rpart)
library(MASS)
library(corrplot)

pml.training <- read.csv("~/Spectrum-R/Coursera/MachineLearning/pml-training.csv",
              header = TRUE, stringsAsFactors = TRUE)
```

From a "big picture" view of the data we see that there are many missing values. Many of the predictors are nearly devoid of values. In the chart below the light colored areas reflect missing values.



Examining the data set in detail, it appears that this is a set of measurements taken rapidly over a short time interval. The measurements are frequently summarized. Some predictors are only populated during the summaries.

One solution is to select a subset of predictors for which we are provided complete data. With only a little knowledge of the data set we can select a subset containing those predictors and eliminate others (user_name, timestamps, etc.) that may confound the results.

```
pml.training <- pml.training[ , c(8:11, 37:49, 60:68, 84:86, 102, 113:124, 140, 151:160)]
dim(pml.training)
```

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

This leaves 19,622 observations of 53 variables, including the CLASSE response variable.

Preprocessing

Next we divide the pml-training set into a training and a testing set that we can later use to cross validate the models.

```
tr.index <- createDataPartition(y = pml.training$classe, p = 0.80, list = FALSE)
training <- pml.training[ tr.index, ]
testing  <- pml.training[ -tr.index, ]
```

Using the tools in the caret package we next look for near zero-variance predictors which we might wish to eliminate as they may create problems for some model types.

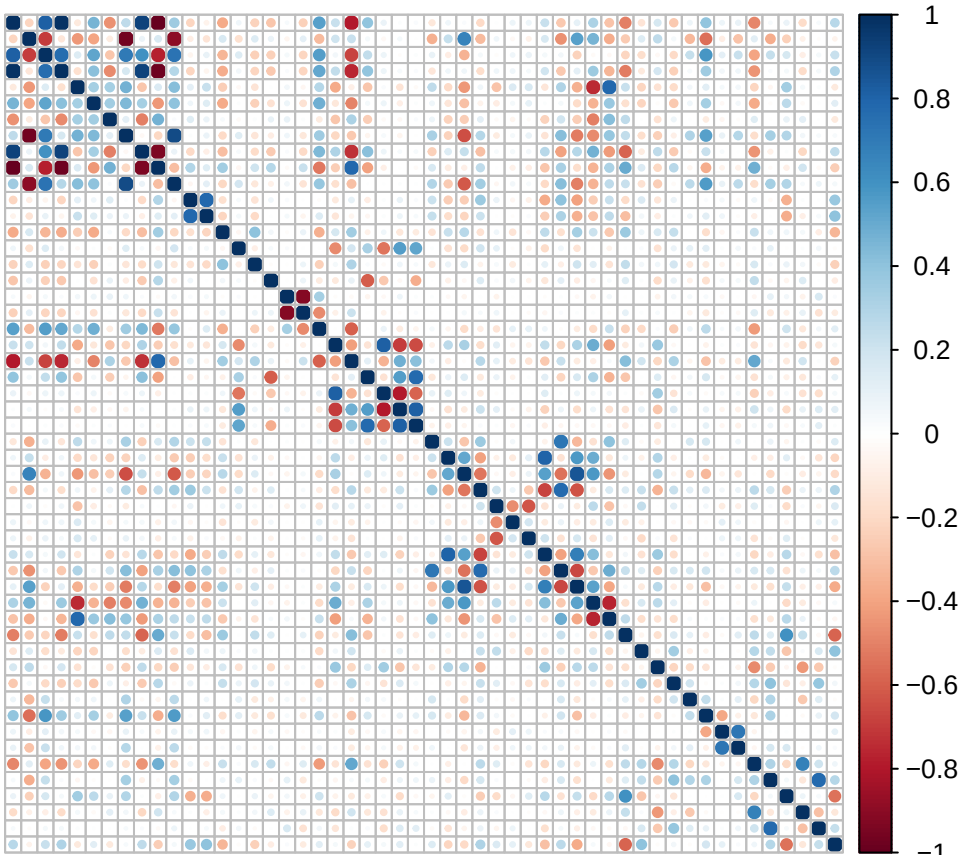
```
nearZeroVar(training, saveMetrics= FALSE)
```

```
## integer(0)
```

The nearZeroVar function returns a null result indicating that there are no near zero-variance variables among the remaining predictors.

We next examine correlations among the predictors, and remove predictors that have a pairwise correlation greater than 0.80.

```
set.seed(1)
M <- cor(training[ , -length(training)])
corrplot(M, order = "original", tl.pos = "n")
```




```
hCor <- findCorrelation(M, 0.80)
training <- training[, -hCor]
testing  <- testing[, -hCor]
```

This leaves

```
dim(training)[2]
```

```
## [1] 43
```

predictors.

Running the Models

We'll first pre-process using principal components analysis.

```
set.seed(8181)
preProc <- preProcess(training[, -length(training)], method = "pca")
trainPC <- predict(preProc, training[, -length(training)])
```

This leaves

```
dim(trainPC)[2]
```

```
## [1] 26
```

predictors.

So, we begin with a recursive partitioning model from the rpart package.

```
set.seed(8181)
rp1 <- train(training$classe ~ ., data = trainPC, method="rpart")
```

The accuracy is disappointing.

```
rp1$results[1,2]
```

```
## [1] 0.3768041
```

Next we try a k-nearest neighbor model.

```
set.seed(8181)
rp2 <- train(training$classe ~ ., data=trainPC, method="knn")
```

The accuracy here is promising. The model accuracy is

```
rp2$results[1,2]
```

```
## [1] 0.9428352
```

Now we need to evaluate the fit using the cross-validation set. An examination of the confusion matrix shows that our accuracy is 96.7%, with a confidence interval of (0.961, 0.973) which is very good.

```
testPC <- predict(preProc, testing[, -length(testing)])
cf <- confusionMatrix(testing$classe,predict(rp2, newdata = testPC))
cf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1103    5    5    2    1
```

```
##          B   15  728   11    2    3
##          C    6    5  657   14    2
##          D    0    0   29  610    4
##          E    0    9    6    4  702
##
## Overall Statistics
##
##                Accuracy : 0.9686
##                  95% CI : (0.9627, 0.9739)
##     No Information Rate : 0.2865
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9603
##  Mcnemar's Test P-Value : 0.01244
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9813   0.9746   0.9280   0.9652   0.9860
## Specificity            0.9954   0.9902   0.9916   0.9900   0.9941
## Pos Pred Value         0.9884   0.9592   0.9605   0.9487   0.9736
## Neg Pred Value         0.9925   0.9940   0.9843   0.9933   0.9969
## Prevalence             0.2865   0.1904   0.1805   0.1611   0.1815
## Detection Rate         0.2812   0.1856   0.1675   0.1555   0.1789
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9883   0.9824   0.9598   0.9776   0.9900
```

An examination of the confusion matrix shows that the accuracy is

```
##  Accuracy
## 0.9686464
```

which is very good.

Summary

Finally, we turn to the pml-testing data provided to "grade" the 20 test samples using our model.

```
pml.testing  <- read.csv("~/Spectrum-R/Coursera/MachineLearning/pml-testing.csv",
                         header = TRUE, stringsAsFactors = TRUE)
pml.testing  <- pml.testing[ ,  c(8:11, 37:49, 60:68, 84:86, 102, 113:124, 140, 151:160)]
pml.testing  <- pml.testing[, -hCor]

tPC <- predict(preProc, pml.testing[, -length(pml.testing)])
result.set <- predict(rp2$finalModel, tPC, type = "class")
result.set
```

```
##  [1] B A B A A E D B A A D C B A E E A B B B
## Levels: A B C D E
```