Practical ML Course Project

Daniel Acker

10/21/2017

Load libraries and set the seed for the analysis.

```
library(dplyr);library(reshape2);library(caret);library(ggplot2);library(randomForest())
set.seed(1692)
```

Loading data

Load the data and partition into training and testing sets for cross validation.

```
# load training data
all_data = read.csv("pml-training.csv")

# split into training and testing set
train_idx = createDataPartition(all_data$classe, p=0.6, list=F)
training = all_data[train_idx,]
testing = all_data[-train_idx,]
```

Preprocessing

Now I'll do some preprocessing to remove irrelevant and low variance variables. I'll also impute missing values in the remaining variables.

```
# separate the outcome
training_y = training$classe

# preprocessing
## remove factor variables and variables with little meaning
factor_idx = sapply(training, is.factor)
training_x = training[,!factor_idx] %>%
  dplyr::select(-X, -raw_timestamp_part_1, -raw_timestamp_part_2, -num_window)

## remove variables with low variance and impute missing data
processing = preProcess(training_x, method=c("nzv", "knnImpute"))
training_processed = predict(processing, training_x)
```

Model training

Even after preprocessing to remove low variance variables, there are many variables, and their relative importance is difficult to intuit. This data seems like a good candidate for a random forest model, which can be highly accurate but difficult to interpret.

```
# fit a random forest model
model = randomForest(x=training_processed, y=training_y)
```

```r
# show statistics
confusionMatrix(predict(model, training_processed), training_y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    0    0    0
##          C    0    0 2054    0    0
##          D    0    0    0 1930    0
##          E    0    0    0    0 2165
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Cross-validation

The random forest model is perfectly accurate, indicating that this model fits the training data very well and may be overfitted. I expect that the model will not perform as well on the test data.

Now I'll test the model's ability on out-of-sample prediction using the testing set.

```r
# separate outcome from testing set
testing_y = testing$classe

# process testing set
factor_idx = sapply(training, is.factor)
testing_x = testing[,!factor_idx] %>%
  dplyr::select(-X, -raw_timestamp_part_1, -raw_timestamp_part_2, -num_window)
testing_processed = predict(processing, testing_x)

# predict outcome on the testing set
Y = predict(model, testing_processed)
```

```r
# show statistics
confusionMatrix(Y, testing_y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2226    8    0    0    2
##          B    4 1499   12    0    0
##          C    0    6 1351   28    1
##          D    1    2    5 1256    7
##          E    1    3    0    2 1432
##
## 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            0.9973   0.9875   0.9876   0.9767   0.9931
## Specificity            0.9982   0.9975   0.9946   0.9977   0.9991
## Pos Pred Value         0.9955   0.9894   0.9747   0.9882   0.9958
## Neg Pred Value         0.9989   0.9970   0.9974   0.9954   0.9984
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2837   0.1911   0.1722   0.1601   0.1825
## Detection Prevalence   0.2850   0.1931   0.1767   0.1620   0.1833
## Balanced Accuracy      0.9978   0.9925   0.9911   0.9872   0.9961
```

The model is 99% accurate on the test data, indicating that it generalizes well to out-of-sample data.

Final predictions

Finally, I'll load the true test data and make predictions to submit.

```r
test_data = read.csv("pml-testing.csv")

# process test data
test_data_x = test_data[,!factor_idx] %>%
  dplyr::select(-X, -raw_timestamp_part_1, -raw_timestamp_part_2, -num_window)
test_data_x_processed = predict(processing, test_data_x)

# make predictions
Y_test_data = predict(model, test_data_x_processed)
print(Y_test_data)
```

```
##  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
```