Predicting The Manner Of Exercise

Jerry Kiely

22 January 2015

The Introduction

This project looks at a HAR dataset:

> Human Activity Recognition - HAR - has emerged as a key research area in the last years and is
> gaining increasing attention by the pervasive computing research community, especially for the
> development of context-aware systems.

From the project brief:

> Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large
> amount of data about personal activity relatively inexpensively. These type of devices are part of
> the quantified self movement – a group of enthusiasts who take measurements about themselves
> regularly to improve their health, to find patterns in their behavior, or because they are tech
> geeks.

Relating to the current task:

> One thing that people regularly do is quantify how much of a particular activity they do, but
> they rarely quantify how well they do it. In this project, your goal will be to use 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.

We will attempt to predict the manner in which participants performed barbell lifts. We will do this by
training a classifier with the provided dataset.

The Data

First we load the data:

```
train <- read.csv('pml-training.csv', na.strings = c('NA', ''));
test  <- read.csv('pml-testing.csv',  na.strings = c('NA', ''));
```

Next we clean the data by removing any columns that contain null data:

```
nas   <- colSums(is.na(train)) > 0;

names <- colnames(train);
cols  <- names[!nas];
train <- train[, cols];

names <- colnames(test);
cols  <- names[!nas];
test  <- test[, cols];
```

Next we remove columns for predictors with nearly zero variance - i.e. columns that are almost constant and which are unlikely to affect the outcome:

```
zvs   <- nearZeroVar(train);

train <- train[, -zvs];
test  <- test[, -zvs];
```

Then we ommit the first five columns (which contain index, user, and timestamp information):

```
train <- train[, 6:ncol(train)];
test  <- test[, 6:ncol(test)];
```

Now we can split the training data into two sets - for training and cross validation:

```
partition <- createDataPartition(y = train$classe, p = 0.8, list = FALSE);
part_tr   <- train[partition, ];
part_cv   <- train[-partition, ];
```

The Training

Now we go straight into building our model - I choose to use a random forest due to the number of predictors, with classe as the outcome, and everything else as predictors.

Note: in truth we don't need to perform cross validation as the random forest does this internally - we do so only see how well the model performs in terms of estimating out of sample error - anything less than 1% is acceptable:

```
model <- train(
    classe ~ .,
    method = 'rf',
    data = part_tr,
    trControl = trainControl(method = 'oob', number = 4)
);
```

Now we look at how the model fares against the training data, and the cross validation data:

```
predict_tr <- predict(model, part_tr);
cm_tr      <- confusionMatrix(predict_tr, part_tr$classe);

predict_cv <- predict(model, part_cv);
cm_cv      <- confusionMatrix(predict_cv, part_cv$classe);
```

Looking at the confusion matrix for the predictions of the training set:

```
cm_tr;
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##              A 4464    0    0    0    0
##              B    0 3038    0    0    0
##              C    0    0 2738    0    0
##              D    0    0    0 2573    0
##              E    0    0    0    0 2886
##
## Overall Statistics
##
##                   Accuracy : 1
##                     95% CI : (0.9998, 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
```

we see an accuracy of 100%, as expected. Looking at the confusion matrix for the predictions of the cross validation set:

```
    cm_cv;
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    1    0    0    0
##          B    0  758    4    0    0
##          C    0    0  680    1    0
##          D    0    0    0  642    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    0.9987    0.9942    0.9984    0.9986
## Specificity          0.9996    0.9987    0.9997    0.9997    1.0000
## Pos Pred Value        0.9991    0.9948    0.9985    0.9984    1.0000
## Neg Pred Value        1.0000    0.9997    0.9988    0.9997    0.9997
## Prevalence           0.2845    0.1935    0.1744    0.1639    0.1838
## Detection Rate       0.2845    0.1932    0.1733    0.1637    0.1835
## Detection Prevalence  0.2847    0.1942    0.1736    0.1639    0.1835
## Balanced Accuracy    0.9998    0.9987    0.9969    0.9991    0.9993
```

we see an accuracy of 99.82%. From this we can estimate the out of sample error to be 0.1784% which is well within the 1% we set for ourselves.

The Testing

Now lets run our model against the test set and see how it fares:

```
predict_te <- predict(model, test);
cm_te      <- confusionMatrix(predict_te, answers);
```

Looking at the confusion matrix for the predictions of the test set against the correct outcomes:

```
cm_te;
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction A B C D E
##          A 7 0 0 0 0
##          B 0 8 0 0 0
##          C 0 0 1 0 0
##          D 0 0 0 1 0
##          E 0 0 0 0 3
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.8316, 1)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : 1.1e-08
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                    Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.00      1.0     1.00     1.00     1.00
## Specificity            1.00      1.0     1.00     1.00     1.00
## Pos Pred Value         1.00      1.0     1.00     1.00     1.00
## Neg Pred Value         1.00      1.0     1.00     1.00     1.00
```

```
## Prevalence               0.35    0.4    0.05    0.05    0.15
## Detection Rate           0.35    0.4    0.05    0.05    0.15
## Detection Prevalence     0.35    0.4    0.05    0.05    0.15
## Balanced Accuracy        1.00    1.0    1.00    1.00    1.00
```

we see our model has performed with 100% accuracy.