Quality of Exercise Prediction

B Porter

May 13, 2016

Loading Data and Exploratory Analysis

Start by downloading the dataset. This data for this project are available here: Training: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv) Validation Dataset: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv) The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har). In order to load the data set we use read.csv and set the na.strings parameter so that the empty and '#DIV/0!' entries will be set as NA values.

```
library(caret)
require(randomForest)
require(gbm)
require(plyr)
```

```
train <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-trai
ning.csv",na.strings=c("NA","#DIV/0!",""), header = T)
valid <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-test
ing.csv",na.strings=c("NA","#DIV/0!",""), header = T)
dim(train)
```

```
## [1] 19622   160
```

```
dim(valid)
```

```
## [1]  20 160
```

I used the str() and summary() functions to do some exploratory data analysis. I performed the exploratory data analysis on the training set so that I do not incorporate any of the validation set into the design. I am not going to print all this data into this portion because there are quite a few variables and can be difficult to read.

```
str(train)
summary(train)
```

The next step that we need to perform is some cleaning of the data. In order to predict we are supposed to use data from the accelerometers on the belt, forearm, arm and dumbell. To start we remove the first seven columns as they are meta-data about the exercise such as the timestamp and user who performed the activity. We do not want this data to be included. We want to use accelerometer data only. It is important that we perform the same data transformations on both the training and the test data.

```
train <- train[,-(1:7)]
valid <- valid[,-(1:7)]

nearZVar <- nearZeroVar(train)
train <- train[,-nearZVar]
valid <- valid[,-nearZVar]

na_cols <- (colSums(is.na(train)) > 0)
train <- train[,!na_cols]
valid <- valid[,!na_cols]

names(train)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

```
dim(train)
```

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

```
dim(valid)
```

```
## [1] 20 53
```

The last thing we should do is create a data set partition. We want to use about 70% for training and 30% for testing. In order to make this reproducible we set the seed at this point.

```
set.seed(336338)
inTrain <- createDataPartition(y=train$classe, p=0.7, list=FALSE)
train <- train[inTrain, ]
test <- train[-inTrain, ]
```

Now the train dataset and test dataset are ready to build a predictive model.

Prediction Model

In this report we are going to experiment with boosting and random forests and see which of those two would give us the best accuracy.

Cross Validation

We want to use cross validation with both of our models. This will help prevent overfitting. We do have a large data set but approaches such as random forests are prone to overfitting. For this report we will use 10 fold cross validation. The following code section shows how to set up cross validation for use in caret train.

```
tr_control <- trainControl(method='cv', number = 5)
```

Generalized Boosting Regression Model

We will start with boosting. This is going to try and fit a lot of weak predictors and then weight them and combine them. By doing this we should end up with a much stronger predictor. Notice that the trControl parameter is set using the variable we created earlier telling train() to do cross validation while training.

```
model_gbm <- train(classe ~ .,  data=train, trControl=tr_control,  method='gb
m', verbose=F)
```

Now let's use the boosting model we trained to predict on the test set we split off our original training set. This will give us an idea about how well this model is going to perform on new data.

```
predict_gbm <- predict(model_gbm, newdata=test)
cm_gbm <- confusionMatrix(predict_gbm, test$classe)
print(cm_gbm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1172   28    0    0    0
##          B    7  740   16    1    3
##          C    3   11  710   25    4
##          D    3    0    7  654    9
##          E    0    0    1    3  694
##
## Overall Statistics
##
##                Accuracy : 0.9704
##                  95% CI : (0.9648, 0.9754)
##     No Information Rate : 0.2897
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9625
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9890   0.9499   0.9673   0.9575   0.9775
## Specificity            0.9904   0.9918   0.9872   0.9944   0.9988
## Pos Pred Value         0.9767   0.9648   0.9429   0.9718   0.9943
## Neg Pred Value         0.9955   0.9883   0.9928   0.9915   0.9953
## Prevalence             0.2897   0.1904   0.1794   0.1670   0.1736
## Detection Rate         0.2865   0.1809   0.1736   0.1599   0.1696
## Detection Prevalence   0.2933   0.1875   0.1841   0.1645   0.1706
## Balanced Accuracy      0.9897   0.9709   0.9772   0.9760   0.9881
```

Random Forests

Next let's compare the random forest method. Again we are going to use the same train control to do cross validation.

```
model_rf <- train(classe ~ .,  data=train, trControl=tr_control,  method='rf',
verbose=F)
```

Now we use the random forests model to predict on the same testing set we used to evaluate the boosting method.

```
predict_rf <- predict(model_rf, newdata=test)
cm_rf <- confusionMatrix(predict_rf, test$classe)
print(cm_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1185    0    0    0    0
##          B    0  779    0    0    0
##          C    0    0  734    0    0
##          D    0    0    0  683    0
##          E    0    0    0    0  710
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2897
##     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    1.000   1.0000
## Specificity            1.0000   1.0000   1.0000    1.000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000    1.000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000    1.000   1.0000
## Prevalence             0.2897   0.1904   0.1794    0.167   0.1736
## Detection Rate         0.2897   0.1904   0.1794    0.167   0.1736
## Detection Prevalence   0.2897   0.1904   0.1794    0.167   0.1736
## Balanced Accuracy      1.0000   1.0000   1.0000    1.000   1.0000
```

Prediction

The final step in this report is to chose the best model and then run prediction on the validation set. I chose the random forests prediction model. It performed better on the test set compared to boosting. The following code runs prediction on the validation set loaded from the pml_testing.csv.

```
predict_valid <- predict(model_rf, newdata=valid)
predict_valid_df <- data.frame(problem_id=valid$problem_id, predicted=predict_v
alid)
print(predict_valid_df)
```

```
##    problem_id predicted
## 1           1         B
## 2           2         A
## 3           3         B
## 4           4         A
## 5           5         A
## 6           6         E
## 7           7         D
## 8           8         B
## 9           9         A
## 10         10         A
## 11         11         B
## 12         12         C
## 13         13         B
## 14         14         A
## 15         15         E
## 16         16         E
## 17         17         A
## 18         18         B
## 19         19         B
## 20         20         B
```

Summary

The first part of this report we loaded the data and cleaned it by removing meta-data. I also used nearZeroVar() to remove features that have almost no variance. These features do not contribute much to a prediction model. I set up the caret train function to use 5 fold cross validation to help avoid overfitting in the models. I also removed features that contain NA values. Next I compared two models on this data set. The first model was a boosting model(gbm) and it preformed very well with accuracy of 0.9704229. The next model was random forests. This model also was able to learn very well with an accuracy of 1. I then took the model with the best accuracy which was random forests and applied that to the validation set.