Predicting Quality Barbell Lifts

Mary

October 14, 2019

Executive Summary

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.

The goal of your project was to predict the manner in which they did the exercise. This is the "classe" variable in the training set. The other variables in the data set were used to predict with.

A Generalized Boosted Model and a Random Forest Model was used to develop a prediction function. Cross validation was used for each method.

Read in and clean data

```
testing <- read.csv("pml-testing.csv")
training <- read.csv("pml-training.csv")
```

```
set.seed(1234)
inTrain <- createDataPartition(y=training$classe,
                               p=0.75, list=FALSE)
trainSet <- training[inTrain,]
testSet <- training[-inTrain,]
```

remove identifying columns, first five columns from both data sets

```
testSet <- testSet[,-(1:5)]
trainSet <- trainSet[,-(1:5)]
```

Remove coluums that are all NA or blank

```
trainSet <- trainSet
trainSet[trainSet ==""]<-NA
trainSet <- trainSet[,colSums(is.na(trainSet))<1]
```

Remove variables with near zero variance

```
NZV <- nearZeroVar(trainSet)
trainSet <- trainSet[, -NZV]
dim(trainSet)
```

```
## [1] 14718    54
```

```
dim(testSet)
```

```
## [1] 4904  155
```

find col_names in trainSet and choose the same cols in trainSet

```
cn <- colnames(trainSet)
testSet <- testSet[,cn[1:length(cn)]]
```

First Method: Generalized Boosted Model

```
set.seed(1234)
library(gbm)
```

```
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
model_Fit_GBM <- train(classe ~., method = "gbm", trControl = controlGBM,data = trainSet, verbose = FALS
model_Fit_GBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 42 had non-zero influence.
```

Calculate the confusion matrix for trainSet

```
predict_GBM_test <- predict(model_Fit_GBM, newdata=trainSet)
confusionMatrix(predict_GBM_test, trainSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4182   10    0    0    0
##          B    3 2817   11    5    5
##          C    0   21 2549   17    2
##          D    0    0    6 2390   13
##          E    0    0    1    0 2686
##
## Overall Statistics
##
##                Accuracy : 0.9936
##                  95% CI : (0.9922, 0.9948)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9919
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9891   0.9930   0.9909   0.9926
## Specificity            0.9991   0.9980   0.9967   0.9985   0.9999
## Pos Pred Value         0.9976   0.9916   0.9846   0.9921   0.9996
## Neg Pred Value         0.9997   0.9974   0.9985   0.9982   0.9983
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2841   0.1914   0.1732   0.1624   0.1825
## Detection Prevalence   0.2848   0.1930   0.1759   0.1637   0.1826
## Balanced Accuracy      0.9992   0.9935   0.9948   0.9947   0.9963
```

The In sample error = 1 - 0.9936 = 0.0064

Calculate the confusion matrix for testSet

```
predict_GBM_test <- predict(model_Fit_GBM, newdata=testSet)
CM_GBM <- confusionMatrix(predict_GBM_test, testSet$classe)
CM_GBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    9    0    0    0
##          B    0  925    8    1    4
##          C    0   13  844   12    5
##          D    0    2    3  791    4
##          E    0    0    0    0  888
##
## Overall Statistics
##
##                Accuracy : 0.9876
##                  95% CI : (0.9841, 0.9905)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9843
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9747   0.9871   0.9838   0.9856
## Specificity            0.9974   0.9967   0.9926   0.9978   1.0000
## Pos Pred Value         0.9936   0.9861   0.9657   0.9888   1.0000
## Neg Pred Value         1.0000   0.9939   0.9973   0.9968   0.9968
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1886   0.1721   0.1613   0.1811
## Detection Prevalence   0.2863   0.1913   0.1782   0.1631   0.1811
## Balanced Accuracy      0.9987   0.9857   0.9899   0.9908   0.9928
```

The out of sample error = 1 - 0.9876 = 0.0124

Second Method: Random Forest Model

```
set.seed(1234)

controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
model_Fit_RandForest <- train(classe ~ ., data=trainSet, method="rf",
                          trControl=controlRF)
model_Fit_RandForest$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.18%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4183    1    0    0    1 0.0004778973
## B    6 2839    2    1    0 0.0031601124
## C    0    5 2562    0    0 0.0019477990
## D    0    0    4 2407    1 0.0020729685
## E    0    1    0    5 2700 0.0022172949
```

Calculate the confusion matrix for trainSet

```
predict_RF_train <- predict(model_Fit_RandForest, newdata=trainSet)
confusionMatrix(predict_RF_train, trainSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4185    0    0    0    0
##          B    0 2848    0    0    0
##          C    0    0 2567    0    0
##          D    0    0    0 2412    0
##          E    0    0    0    0 2706
##
## 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   0.1839
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

The In sample error = 1 - 1 = 0

Calculate the confusion matrix for testSet

```
predict_RF_test <- predict(model_Fit_RandForest, newdata=testSet)
CM_RF <- confusionMatrix(predict_RF_test, testSet$classe)
CM_RF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    3    0    0    0
##          B    0  946    1    0    0
##          C    0    0  854    0    0
##          D    0    0    0  804    0
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.9992
##                  95% CI : (0.9979, 0.9998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.999
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9968   0.9988   1.0000   1.0000
## Specificity            0.9991   0.9997   1.0000   1.0000   1.0000
## Pos Pred Value         0.9979   0.9989   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   0.9992   0.9998   1.0000   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1929   0.1741   0.1639   0.1837
## Detection Prevalence   0.2851   0.1931   0.1741   0.1639   0.1837
## Balanced Accuracy      0.9996   0.9983   0.9994   1.0000   1.0000
```

The out of sample error = 1 - 0.999 = 0.001

Camparison and prediction

The accuracy for the Generalized Boosted Model is 0.988 and the accuracy for the Random Forest Model is 0.999, so I will use the Random Forest Model to predict for the quiz.

```
testing_predictors_for_quiz <- testing[,cn[1:length(cn)-1]]

predict(model_Fit_RandForest, newdata=testing_predictors_for_quiz)
```

```
##  [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
```