Predicting Weight Lifting Exercises from Wearable Technology

Erdinc Albayrak

4/25/2021

Overview

In this report, I have used the Weight Lifting Exercises Dataset which is generated from wearable technology to predict how the activities performed. The activity was one set of Unilateral Dumbbell Biceps Curl but in 5 different ways.

As a result, we have a classification problem with 5 classes.

Pre-Processing

I have cleaned up the dataset with pre-processing because it included mostly empty columns and as well as some metadata. This lowered the number of columns from 160 to 53.

```
dataset <- dataset[!sapply(dataset, function(x) all(x == "" || is.na(x)))]
truetest <- truetest[!sapply(truetest, function(x) all(x == "" || is.na(x)))]
dataset <- dataset[, -c(1:7)]
truetest <- truetest[, -c(1:7)]
```

Method

First of all, as I will not contain myself to one model, I need a validation set in addition to a test set. This allows me to use the validation set accuracy for model selection. Once my model is selected, this model is used predict the test set for the final evaluation and consequent metrics are declared as model's success.

Since we have a large enough dataset (19622 data points), I have decided to use a 80-10-10 split for training, validation and test sets. This maximizes my training set size, while still giving me large enough validation and test sets that will grant accurate estimations for evaluation metrics.

```
trainingIndices <- createDataPartition(y=dataset$classe,p=0.80,list=FALSE)
training <- dataset[trainingIndices,]
dataset <- dataset[-trainingIndices,]
validationIndices <- createDataPartition(y=dataset$classe,p=0.50,list=FALSE)
validation <- dataset[validationIndices,]
testing <- dataset[-validationIndices,]
```

Model Selection

Since caret provides a generic interface, I tried several learning methods without much hassle. In total, I generated a boosted logistic regression, a decision tree, a bagged tree, a random forest, and a naive bayes fits.

Naive Bayes

Naive bayes model is trained and tested against the validation set.

```
naivefit <- train(classe~.,data=training,method="nb",trControl=trainControl(method="cv",number=3,verbos
naivepredictions <- predict(naivefit,validation)
confusionMatrix(data = naivepredictions,reference = validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 493  78  77  68  23
##          B  10 260  16   2  30
##          C  27  19 236  36  10
##          D  28  19  13 204  15
##          E   0   4   0  12 283
##
## Overall Statistics
##
##                Accuracy : 0.7519
##                  95% CI : (0.7322, 0.7709)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6824
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8835   0.6842   0.6901   0.6335   0.7839
## Specificity            0.8249   0.9634   0.9432   0.9543   0.9900
## Pos Pred Value         0.6671   0.8176   0.7195   0.7312   0.9465
## Neg Pred Value         0.9469   0.9271   0.9352   0.9299   0.9531
## Prevalence             0.2843   0.1936   0.1742   0.1640   0.1839
## Detection Rate         0.2511   0.1325   0.1202   0.1039   0.1442
## Detection Prevalence   0.3765   0.1620   0.1671   0.1421   0.1523
## Balanced Accuracy      0.8542   0.8238   0.8167   0.7939   0.8870
```

Logistic Regression

Boosted logistic regression model is trained and tested against the validation set.

```
logisticfit <- train(classe~.,data=training,method="LogitBoost",trControl=trainControl(method="cv",numb
logisticpredictions <- predict(logisticfit,validation)
confusionMatrix(data = logisticpredictions,reference = validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 517  20   3  11   1
##          B   5 294   9   0   6
##          C   4   6 273  24   5
##          D   2   6   7 240  11
##          E   1   2   1   4 313
##
```

```
## Overall Statistics
##
##                Accuracy : 0.9275
##                  95% CI : (0.9144, 0.9391)
##     No Information Rate : 0.2997
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9077
##
##  Mcnemar's Test P-Value : 2.314e-05
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9773   0.8963   0.9317   0.8602   0.9315
## Specificity            0.9717   0.9861   0.9735   0.9825   0.9944
## Pos Pred Value         0.9366   0.9363   0.8750   0.9023   0.9751
## Neg Pred Value         0.9901   0.9766   0.9862   0.9740   0.9841
## Prevalence             0.2997   0.1858   0.1660   0.1581   0.1904
## Detection Rate         0.2929   0.1666   0.1547   0.1360   0.1773
## Detection Prevalence   0.3127   0.1779   0.1768   0.1507   0.1819
## Balanced Accuracy      0.9745   0.9412   0.9526   0.9214   0.9630
```

Decision Tree

Decision tree model is trained and tested against the validation set.

```
treefit <- train(classe~.,data=training,method="rpart",trControl=trainControl(method="cv",number=3,verbo
treepredictions <- predict(treefit,validation)
confusionMatrix(data = treepredictions,reference = validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 506 145 157 129  52
##          B   9 130  10   2  38
##          C  28  42 150  48  50
##          D  14  63  25 143  61
##          E   1   0   0   0 160
##
## Overall Statistics
##
##                Accuracy : 0.5548
##                  95% CI : (0.5325, 0.5769)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4212
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            0.9068  0.34211  0.43860  0.44410  0.44321
## Specificity            0.6562  0.96273  0.89636  0.90067  0.99938
## Pos Pred Value         0.5116  0.68783  0.47170  0.46732  0.99379
## Neg Pred Value         0.9466  0.85908  0.88328  0.89197  0.88846
## Prevalence             0.2843  0.19358  0.17422  0.16403  0.18390
## Detection Rate         0.2578  0.06623  0.07641  0.07285  0.08151
## Detection Prevalence   0.5038  0.09628  0.16200  0.15588  0.08202
## Balanced Accuracy      0.7815  0.65242  0.66748  0.67238  0.72129
```

Bagged Decision Tree

Bagged decision tree model is trained and tested against the validation set.

```
bagtreefit <- train(classe~.,data=training,method="treebag",trControl=trainControl(method="cv",number=3
bagtreepredictions <- predict(bagtreefit,validation)
confusionMatrix(data = bagtreepredictions,reference = validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 556   6   0   1   0
##          B   1 372   4   0   0
##          C   0   2 335  10   0
##          D   1   0   3 311   1
##          E   0   0   0   0 360
##
## Overall Statistics
##
##                Accuracy : 0.9852
##                  95% CI : (0.9789, 0.9901)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9813
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9789   0.9795   0.9658   0.9972
## Specificity            0.9950   0.9968   0.9926   0.9970   1.0000
## Pos Pred Value         0.9876   0.9867   0.9654   0.9842   1.0000
## Neg Pred Value         0.9986   0.9950   0.9957   0.9933   0.9994
## Prevalence             0.2843   0.1936   0.1742   0.1640   0.1839
## Detection Rate         0.2832   0.1895   0.1707   0.1584   0.1834
## Detection Prevalence   0.2868   0.1921   0.1768   0.1610   0.1834
## Balanced Accuracy      0.9957   0.9879   0.9861   0.9814   0.9986
```

Random Forest

Random forest model is trained and tested against the validation set.

```
forestfit <- train(classe~.,data=training,method="rf",trControl=trainControl(method="cv",number=3,verbos
forestpredictions <- predict(forestfit,validation)
```

```
confusionMatrix(data = forestpredictions,reference = validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 558   0   0   0   0
##          B   0 379   1   0   0
##          C   0   1 341   9   0
##          D   0   0   0 313   1
##          E   0   0   0   0 360
##
## Overall Statistics
##
##                Accuracy : 0.9939
##                  95% CI : (0.9893, 0.9968)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9923
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9974   0.9971   0.9720   0.9972
## Specificity            1.0000   0.9994   0.9938   0.9994   1.0000
## Pos Pred Value         1.0000   0.9974   0.9715   0.9968   1.0000
## Neg Pred Value         1.0000   0.9994   0.9994   0.9945   0.9994
## Prevalence             0.2843   0.1936   0.1742   0.1640   0.1839
## Detection Rate         0.2843   0.1931   0.1737   0.1594   0.1834
## Detection Prevalence   0.2843   0.1936   0.1788   0.1600   0.1834
## Balanced Accuracy      1.0000   0.9984   0.9955   0.9857   0.9986
```

Final Model

In my trials random forest model was the most successful so I finalized it as my model and performed predictions on the test set for the final evaluation metrics.

```
testingpredictions <- predict(forestfit,testing)
confusionMatrix(data = testingpredictions,reference = testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 558   1   0   0   0
##          B   0 376   1   0   0
##          C   0   2 341   0   0
##          D   0   0   0 321   2
##          E   0   0   0   0 358
##
## Overall Statistics
##
```

```
##                Accuracy : 0.9969
##                  95% CI : (0.9933, 0.9989)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9961
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9921   0.9971   1.0000   0.9944
## Specificity            0.9993   0.9994   0.9988   0.9988   1.0000
## Pos Pred Value         0.9982   0.9973   0.9942   0.9938   1.0000
## Neg Pred Value         1.0000   0.9981   0.9994   1.0000   0.9988
## Prevalence             0.2847   0.1934   0.1745   0.1638   0.1837
## Detection Rate         0.2847   0.1918   0.1740   0.1638   0.1827
## Detection Prevalence   0.2852   0.1923   0.1750   0.1648   0.1827
## Balanced Accuracy      0.9996   0.9957   0.9979   0.9994   0.9972
```

After this I also performed predictions on the quiz questions.

```
testpredictions <- predict(forestfit,truetest)
testpredictions
```

```
##  [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
```