Prediction Assignment Writeup

Nikolas

29/10/2020

Introduction

In data obtained from http://groupware.les.inf.puc-rio.br/har accelerometers were placed on the belt, forearm, arm, and dumbell of 6 participants. The goal of this project is to identify the type of movement made (“classe”), based on the data collected from these accelometers. The movement can either be “A”, “B”, “C”, “D”, “E”, or “F”. To do this we will first create tidy data which will be used to feed the model. We will also create two subsets of the training data, a training and a testing set, as the imported testing data will be used to answer the 20 quiz questions and does not contain the column “classe”. Finally we will use different models to predict the test set, the best one will be used fo the quiz.

Loading required packages and data

```r
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```r
training <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
testing <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
```

Creating Tidy data

We first want to split up our data into a training and test set. A common split is to use 60% for the training data, and 40% for the testing data. This is exactly what we will do.

```r
set.seed(1234)
intrain <- createDataPartition(training$classe, p = 0.6, list = FALSE)

train1 <- training[intrain, ]
test1 <- training[-intrain, ]
```

We want to create a data set that is easier for the machine learning model to interpret. To do this, we first need to get rid of all predictors that mostly consist of Na’s, in this case I am going to set the threshold to 90%, any variable with a higher percentage of NA’s will be excluded. We also want to get rid of any predictors that have only one or very few unique variables, as this is not going to help the model predict the test data set values. Finally we want to get rid of predictors that provide no additional value (as they are not related to the variable we are trying to predict), such as row number, user name and timestamps.

```r
# Removing predictors containing more than 90% Na's
NoNAs <- lapply(train1, function(x) mean(is.na(x)) ) < 0.9
train1 <- train1[, NoNAs == TRUE]
test1 <- test1[, NoNAs == TRUE]

# Removing predictors with (near) zero variance
zero <- nearZeroVar(train1)
train1 <- train1[, -(zero)]
test1 <- test1[, -(zero)]

# Removing predictors that provide no extra value
novalue <- c(1:6)
train1 <- train1[, -(novalue)]
test1 <- test1[, -(novalue)]
```

Model Creation

Since I do not have the most powerful computer and in order to prevent over fitting I want to limit the number of resampling iterations. For this I will use the trainControl function, which I will include in all of the

```r
tc=trainControl(method="cv", number=5)
```

1. Decision Tree

```r
set.seed(1234)
mod_trees <- train(classe~., method = "rpart", data = train1, trControl = tc,
                   metric = "Accuracy")
pred_trees <- predict(mod_trees, newdata = test1)
table_trees <- table(pred_trees, test1$classe)
confusionMatrix(table_trees)
```

```
## Confusion Matrix and Statistics
##
##
## pred_trees    A    B    C    D    E
##          A 1997  635  642  585  192
##          B   38  503   44  244  180
##          C  163  380  682  457  396
##          D    0    0    0    0    0
##          E   34    0    0    0  674
##
## Overall Statistics
##
##                Accuracy : 0.4915
##                  95% CI : (0.4803, 0.5026)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3357
```

```
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8947  0.33136  0.49854   0.0000  0.46741
## Specificity            0.6341  0.92004  0.78450   1.0000  0.99469
## Pos Pred Value         0.4930  0.49851  0.32820      NaN  0.95198
## Neg Pred Value         0.9381  0.85154  0.88107   0.8361  0.89241
## Prevalence             0.2845  0.19347  0.17436   0.1639  0.18379
## Detection Rate         0.2545  0.06411  0.08692   0.0000  0.08590
## Detection Prevalence   0.5163  0.12860  0.26485   0.0000  0.09024
## Balanced Accuracy      0.7644  0.62570  0.64152   0.5000  0.73105
```

2. Boosting

```
set.seed(1234)
mod_boost <- train(classe ~ ., method="gbm", data=train1, trControl = tc,
                   metric = "Accuracy", verbose = FALSE)
pred_boost <- predict(mod_boost, test1)
table_boost <- table(pred_boost, test1$classe)
confusionMatrix(table_boost)
```

```
## Confusion Matrix and Statistics
##
##
## pred_boost    A    B    C    D    E
##          A 2203   43    0    0    3
##          B   18 1433   54    4   21
##          C    7   35 1293   33   12
##          D    3    4   21 1239   24
##          E    1    3    0   10 1382
##
## Overall Statistics
##
##                Accuracy : 0.9623
##                  95% CI : (0.9578, 0.9664)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9523
##
##  Mcnemar's Test P-Value : 5.06e-09
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9870   0.9440   0.9452   0.9635   0.9584
## Specificity            0.9918   0.9847   0.9866   0.9921   0.9978
## Pos Pred Value         0.9795   0.9366   0.9370   0.9597   0.9900
## Neg Pred Value         0.9948   0.9865   0.9884   0.9928   0.9907
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2808   0.1826   0.1648   0.1579   0.1761
## Detection Prevalence   0.2866   0.1950   0.1759   0.1645   0.1779
## Balanced Accuracy      0.9894   0.9643   0.9659   0.9778   0.9781
```

3. Random Forests

```
set.seed(1234)
mod_rf <- train(classe ~., method = "rf", data = train1, trControl = tc,
                   metric = "Accuracy")
pred_rf <- predict(mod_rf, test1)
table_rf <- table(pred_rf, test1$classe)
confusionMatrix(table_rf)
```

```
## Confusion Matrix and Statistics
##
##
## pred_rf    A    B    C    D    E
##       A 2231   11    0    0    0
##       B    0 1506    8    0    0
##       C    0    1 1359   20    4
##       D    1    0    1 1264    5
##       E    0    0    0    2 1433
##
## Overall Statistics
##
##                Accuracy : 0.9932
##                  95% CI : (0.9912, 0.9949)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9915
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9921   0.9934   0.9829   0.9938
## Specificity            0.9980   0.9987   0.9961   0.9989   0.9997
## Pos Pred Value         0.9951   0.9947   0.9819   0.9945   0.9986
## Neg Pred Value         0.9998   0.9981   0.9986   0.9967   0.9986
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1919   0.1732   0.1611   0.1826
## Detection Prevalence   0.2858   0.1930   0.1764   0.1620   0.1829
## Balanced Accuracy      0.9988   0.9954   0.9948   0.9909   0.9967
```

Model Selection

Based on the three models used to predict the test set, we can conclude that the most accurate model was the random forests model, with an accuracy of 99.32%. This was quite a bit better than Boosting with an accuracy of 96.19% and abnormally better when compared to decision trees which had a mere accuracy of 49.15%.