MachineLearningAssignment

Kunal Sharma

17/07/2020

Overview

This is a document intending to show the model build process for predicting the manner in which people do exercise. This report describes how I built my model, how I used cross-validation and why I made the choices I did.

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, my goal was 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 and this forms the data set on which the predictive model is buily.

Finally I use my chosen model to predict 20 different test cases for how individuals carried out their activity.

Loading data set and packages etc.

```
getwd()
dir()
setwd("~/R")

library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.6.3
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.6.3
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.6.3
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.6.3
```

```
url_training_data <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url_testing_data <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training_data<- read.csv(url(url_training_data))
testing_data<- read.csv(url(url_testing_data))

head(training_data)
names(training_data)
```

```
dim(training_data)
```

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

```
dim(testing_data)
```

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

Condensation of variables

In order to help build the models with relevant variables only, I condensed the variables in the training data set by removing those columns with low variance of values, removing columns with high “NA” proportions, and also removing the first few columns which held non-numric values.

```
training_data_condensed <- training_data[,8:dim(training_data)[2]]

columns_high_na <- sapply(training_data_condensed, function(x) mean(is.na(x))) > 0.90
training_data_condensed <- training_data_condensed[,columns_high_na == FALSE]

dim(training_data_condensed)
```

```
## [1] 19622    86
```

```
variables_low_variance <- nearZeroVar(training_data_condensed, saveMetrics=TRUE)
summary(variables_low_variance$nzv)
```

```
##    Mode   FALSE    TRUE
## logical      53      33
```

```
training_data_condensed <- training_data_condensed[,c(variables_low_variance$nzv==FALSE)]

dim(training_data_condensed)
```

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

Split the training data set

In order to be able to test models as I built, I needed to construct my own training/testing data split and I did so using the following code to split on the outcome variable.

```
set.seed(123)
trainIndex = createDataPartition(y=training_data_condensed$classe, p = 0.70, list=FALSE)
training_train <- training_data_condensed[trainIndex,]
training_test <- training_data_condensed[-trainIndex,]

dim(training_train)
```

```
## [1] 13737    53
```

```
dim(training_test)
```

```
## [1] 5885   53
```

Model type 1 - tree

```
tree_model <- train(classe ~ ., data = training_data_condensed, method="rpart")
tree_predict <- predict(tree_model, training_test)
confusionMatrix(tree_predict, training_test$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1529  446  469  444  144
##          B   27  403   30  167  145
##          C  116  290  527  353  306
##          D    0    0    0    0    0
##          E    2    0    0    0  487
## 
## Overall Statistics
## 
##                Accuracy : 0.5006
##                  95% CI : (0.4877, 0.5135)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.3477
## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9134  0.35382  0.51365   0.0000  0.45009
## Specificity            0.6431  0.92225  0.78082   1.0000  0.99958
## Pos Pred Value         0.5043  0.52202  0.33103      NaN  0.99591
## Neg Pred Value         0.9492  0.85605  0.88376   0.8362  0.88973
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2598  0.06848  0.08955   0.0000  0.08275
## Detection Prevalence   0.5152  0.13118  0.27052   0.0000  0.08309
## Balanced Accuracy      0.7782  0.63803  0.64723   0.5000  0.72484
```

Model type 2 - tree extension to a random forest

```
forest_model_10 <- train(classe ~ ., data = training_data_condensed, method = "rf", ntree = 10)
forest_predict_10 <- predict(forest_model_10, training_test)
confusionMatrix(forest_predict_10, training_test$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1139    0    0    0
##          C    0    0 1026    0    0
##          D    0    0    0  964    0
##          E    0    0    0    0 1082
## 
## Overall Statistics
## 
##                Accuracy : 1
##                  95% CI : (0.9994, 1)
##     No Information Rate : 0.2845
##     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             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Final model - random forest

```
testing_data <- testing_data[,8:dim(training_data)[2]]
testing_data <- testing_data[,columns_high_na == FALSE]
testing_data <- testing_data[,c(variables_low_variance$nzv==FALSE)]

dim(testing_data)
```

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

```
forest_predict_10_assignment <- predict(forest_model_10, testing_data[,-53])
forest_predict_10_assignment
```

```
##  [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
```