Practical Machine Learning Course Project

Sooihk Ro

2022-07-10

Objective

In this report, data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict which exercise they performed. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The outcome is the "classe" variable. 3 classification models were trained on this dataset using k-fold cross validation: Decision Tree, Gradient Boosted Trees, and Random Forest. The model with the best accuracy and out of sample error rate will be used to predict the test exercise dataset.

The training dataset is found here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv. The test dataset is found: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv. All of the data is this document comes from this source:

http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har.

Data Preprocessing

Load libraries and datasets

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(rattle)
```

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
set.seed(123)

pml_training <- read.csv("./data/pml-training.csv")
pml_testing <- read.csv("./data/pml-testing.csv")
dim(pml_training)
```

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

```r
dim(pml_testing)
```

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

There are 19,622 observations and 160 variables for the training dataset. There are 20 observations and 160 variables for the testing dataset.

Clean dataset

The dataset needs to be cleaned by removing variables with more than 90% of its data missing, variables irrelevant for the prediction, and variables with near zero variance. The training dataset reduces down to 53 variables.

```r
#remove variables with more than 90% NA
pml_training <- pml_training[, which(colMeans(!is.na(pml_training)) > 0.9)]
#remove irrelavant variables to the outcome
pml_training <- pml_training[,-c(1:7)]
#remove near zero variance predictors
near_ZeroPredictors <- nearZeroVar(pml_training)
pml_training <- pml_training[,-near_ZeroPredictors]
dim(pml_training)
```

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

Data partition

The training dataset is then splitted into a sub training set and a validation set.

```r
inTrain <- createDataPartition(y=pml_training$classe, p=0.7, list=FALSE)
training <- pml_training[inTrain,]
validation <- pml_training[-inTrain,]
```

Model Building

The following classification models were used: Decision Tree, Gradient Boosted Trees, and Random Forest. 5-fold cross validation is applied to all models in order to select optimal tuning parameters.

```r
#Set trControl parameter to do 5 fold cross valdiation
control <- trainControl(method="cv", number=5, verboseIter=FALSE)
```

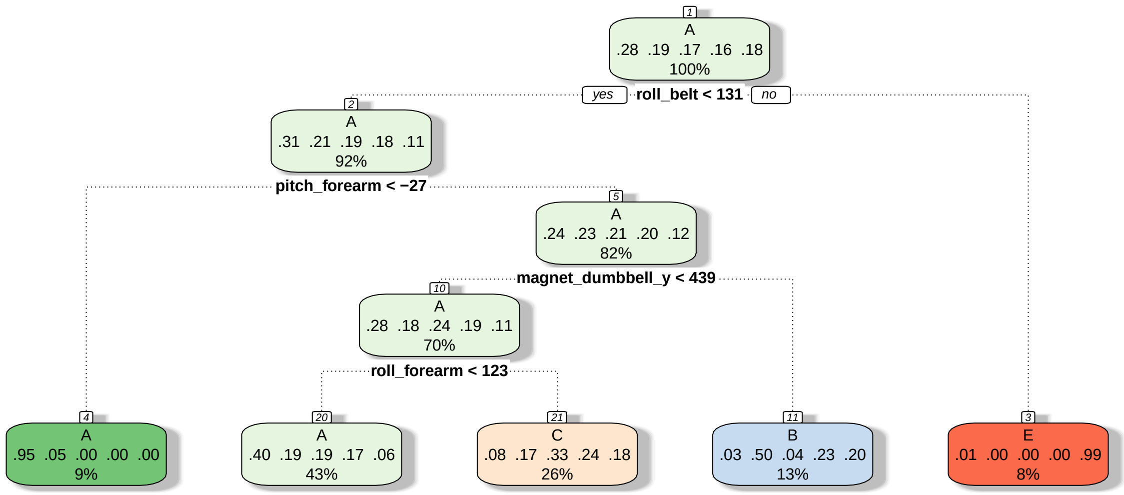
Decision Tree

Model

```r
#Set trControl parameter to do 5 fold cross valdiation
tree_fit <- train(classe~., method="rpart", data=training, trControl=control)
tree_fit$finalModel
```

```
## n= 13737
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 13737 9831 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 12581 8687 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -26.65 1260   60 A (0.95 0.048 0 0 0) *
##      5) pitch_forearm>=-26.65 11321 8627 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 438.5 9562 6924 A (0.28 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 122.5 5955 3591 A (0.4 0.19 0.19 0.17 0.064) *
##         21) roll_forearm>=122.5 3607 2406 C (0.076 0.17 0.33 0.24 0.18) *
##       11) magnet_dumbbell_y>=438.5 1759  883 B (0.032 0.5 0.045 0.23 0.2) *
##    3) roll_belt>=130.5 1156   12 E (0.01 0 0 0 0.99) *
```

```r
fancyRpartPlot(tree_fit$finalModel)
```



Rattle 2022−Jul−10 21:46:04 aduro

Prediction

```
#Set trControl parameter to do 5 fold cross valdiation
#Prediction, estimate performance of model on validation data set
predict_tree <- predict(tree_fit, newdata=validation)
confusion_tree <- confusionMatrix(predict_tree, factor(validation$classe))
confusion_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1530  464  469  440  144
##          B   28  397   30  169  145
##          C  114  278  527  355  306
##          D    0    0    0    0    0
##          E    2    0    0    0  487
##
## Overall Statistics
##
##                Accuracy : 0.4997
##                  95% CI : (0.4869, 0.5126)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3464
##
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9140  0.34855  0.51365   0.0000  0.45009
## Specificity            0.6398  0.92162  0.78329   1.0000  0.99958
## Pos Pred Value         0.5021  0.51625  0.33354      NaN  0.99591
## Neg Pred Value         0.9493  0.85496  0.88409   0.8362  0.88973
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2600  0.06746  0.08955   0.0000  0.08275
## Detection Prevalence   0.5178  0.13067  0.26848   0.0000  0.08309
## Balanced Accuracy      0.7769  0.63508  0.64847   0.5000  0.72484
```

Gradient Boosted Trees

Model

```
gbm_fit <- train(classe~., method="gbm", data=training, trControl=control, verbose=FALSE)
```

Prediction

```
#Prediction, estimate performance of model on validation data set
predict_gbm <- predict(gbm_fit, newdata=validation)
confusion_gbm <- confusionMatrix(predict_gbm, factor(validation$classe))
confusion_gbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1649   28    1    2    3
##          B   17 1079   24    7    8
##          C    4   32  985   28   18
##          D    0    0   15  921   15
##          E    4    0    1    6 1038
##
## Overall Statistics
##
##                Accuracy : 0.9638
##                  95% CI : (0.9587, 0.9684)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9542
##
##  Mcnemar's Test P-Value : 1.577e-06
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9851   0.9473   0.9600   0.9554   0.9593
## Specificity            0.9919   0.9882   0.9831   0.9939   0.9977
## Pos Pred Value         0.9798   0.9507   0.9231   0.9685   0.9895
## Neg Pred Value         0.9941   0.9874   0.9915   0.9913   0.9909
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2802   0.1833   0.1674   0.1565   0.1764
## Detection Prevalence   0.2860   0.1929   0.1813   0.1616   0.1782
## Balanced Accuracy      0.9885   0.9678   0.9716   0.9746   0.9785
```

Random Forest

Model

```
random_fit <- train(classe~., method="rf", data=training, trControl=control)
```

Prediction

```
#Prediction, estimate performance of model on validation data set
predict_rf <- predict(random_fit, newdata=validation)
confusion_rf <- confusionMatrix(predict_rf, factor(validation$classe))
confusion_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    5    0    0    0
##          B    1 1126    5    0    0
##          C    0    8 1018   10    4
##          D    0    0    3  954    4
##          E    1    0    0    0 1074
##
## Overall Statistics
##
##                Accuracy : 0.993
##                  95% CI : (0.9906, 0.995)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9912
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9886   0.9922   0.9896   0.9926
## Specificity            0.9988   0.9987   0.9955   0.9986   0.9998
## Pos Pred Value         0.9970   0.9947   0.9788   0.9927   0.9991
```

```
## Neg Pred Value         0.9995   0.9973   0.9983   0.9980   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1913   0.1730   0.1621   0.1825
## Detection Prevalence   0.2850   0.1924   0.1767   0.1633   0.1827
## Balanced Accuracy      0.9988   0.9937   0.9938   0.9941   0.9962
```

Results, Model Assessment

```
model_names <- c("Decision Tree", "Gradient Boost Trees", "Random Forest")
model_accuracy <- round(c(confusion_tree$overall[1],confusion_gbm$overall[1],confusion_rf$overall[1]),3
model_oos_error <- 1 - model_accuracy
model_info <- data.frame(Models = model_names, Accuracy = model_accuracy, oos_error = model_oos_error)
model_info
```

```
##                 Models Accuracy oos_error
## 1        Decision Tree    0.500     0.500
## 2 Gradient Boost Trees    0.964     0.036
## 3        Random Forest    0.993     0.007
```

Based on the table above, the Random Forest model provides the best accuracy (99.3%) and a small out of sample error rate (0.7%). The Random Forest model will be used for our test dataset to predict the type of exercise.

Predictions on test dataset

```
#Predictions on Test dataset
test_results <- predict(random_fit, pml_testing)
test_results
```

```
##  [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
```