Practical Machine Learning Assignment

Zukiswa Mdingi

June 2, 2021

Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. 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. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Introduction

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

```
## Warning: package 'caret' was built under R version 4.0.5
```

```
## Warning: package 'randomForest' was built under R version 4.0.5
```

```
## 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
```

```
## Warning: package 'e1071' was built under R version 4.0.5
## Warning: package 'rpart' was built under R version 4.0.5
## Warning: package 'rpart.plot' was built under R version 4.0.5
## Warning: package 'rattle' was built under R version 4.0.5
## Loading required package: tibble
## Loading required package: bitops
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
##
## Attaching package: 'rattle'
## The following object is masked from 'package:randomForest':
##
##     importance
## Warning: package 'RColorBrewer' was built under R version 4.0.3
```

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

pml_testing <- read.csv("pml-testing.csv")

dim(pml_training)
```

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

```
dim(pml_testing)
```

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

Splitting the data and creating a testing and training dataset

```
in_train <- createDataPartition(pml_training$classe, p=0.75, list = FALSE)

training_data <- pml_training[in_train,]

testing_data <- pml_training[-in_train,]

dim(training_data)
```

```
## [1] 14718   160
```

```
dim(testing_data)
```

```
## [1] 4904  160
```

Removing of NA's and near zero variance observations in the training and testing datasets

```
near.zero.var <- nearZeroVar(training_data)

training_data <- training_data[, -near.zero.var]

testing_data  <- testing_data[, -near.zero.var]


na_data <- sapply(pml_training, function(x) mean(is.na(x))) > 0.95
```

```
training_data <- training_data[, (sapply(training_data, function(x) mean(is.na(x))) > 0.95)==FALSE]

testing_data  <- testing_data[, (sapply(training_data, function(x) mean(is.na(x))) > 0.95)==FALSE]



dim(training_data)
```

```
## [1] 14718    59
```

```
dim(testing_data)
```

```
## [1] 4904  102
```

Removing columns in the datasets that will not be used in the exercise (Removing columns 1-5)

```
training_data <- training_data[ , -(1:5)]

testing_data <- testing_data[ , -(1:5)]

dim(training_data)
```

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

```
dim(testing_data)
```

```
## [1] 4904   97
```

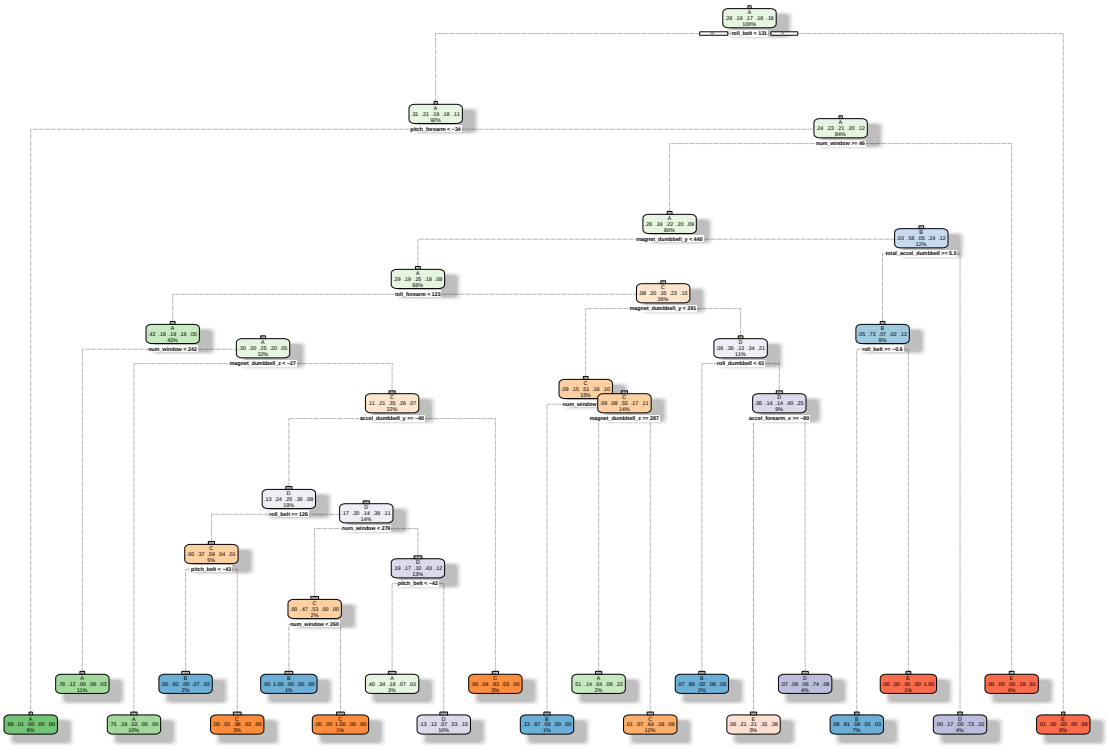
Selecting a model

Decision Tree model

```
rpart.model <- rpart(classe ~ ., data=training_data, method="class")

fancyRpartPlot(rpart.model)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2021−Jun−05 17:01:16 P526851

```
predict_decision_tree <- predict(rpart.model, newdata = testing_data, type="class")

matrix_decision_tree <- confusionMatrix(predict_decision_tree, factor(testing_data$classe))

matrix_decision_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1275  215   45   78   38
##          B   38  540   25   30   23
##          C    6   48  678  115   60
##          D   63  102   54  495  103
##          E   13   44   53   86  677
##
## Overall Statistics
##
##                Accuracy : 0.7473
##                  95% CI : (0.7349, 0.7595)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6786
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9140   0.5690   0.7930   0.6157   0.7514
## Specificity            0.8928   0.9707   0.9434   0.9215   0.9510
## Pos Pred Value         0.7723   0.8232   0.7475   0.6059   0.7755
## Neg Pred Value         0.9631   0.9037   0.9557   0.9244   0.9444
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2600   0.1101   0.1383   0.1009   0.1381
## Detection Prevalence   0.3367   0.1338   0.1850   0.1666   0.1780
## Balanced Accuracy      0.9034   0.7698   0.8682   0.7686   0.8512
```

The model has a low accuracy rate of 0.7473 (74.73%) which is not good enough.

Random Forest Model

```
fitting_rf <- train(classe ~ ., data=training_data, method="rf", trControl=trainControl(method="cv", 3)

fitting_rf$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 100, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.29%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4184    0    0    0    1 0.0002389486
## B    9 2833    5    1    0 0.0052668539
## C    0    7 2558    2    0 0.0035060382
## D    0    0    9 2402    1 0.0041459370
## E    0    1    0    6 2699 0.0025868441
```

```
prediction_tree <- predict(fitting_rf, newdata = testing_data)

confusion_matrix <- confusionMatrix(prediction_tree, factor(testing_data$classe))

confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    3    0    0    0
##          B    0  946    3    0    0
##          C    0    0  852    2    0
##          D    0    0    0  802    1
##          E    0    0    0    0  900
##
## Overall Statistics
##
```

```
##                Accuracy : 0.9982
##                  95% CI : (0.9965, 0.9992)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9977
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9968   0.9965   0.9975   0.9989
## Specificity            0.9991   0.9992   0.9995   0.9998   1.0000
## Pos Pred Value         0.9979   0.9968   0.9977   0.9988   1.0000
## Neg Pred Value         1.0000   0.9992   0.9993   0.9995   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1929   0.1737   0.1635   0.1835
## Detection Prevalence   0.2851   0.1935   0.1741   0.1637   0.1835
## Balanced Accuracy      0.9996   0.9980   0.9980   0.9986   0.9994
```

The model has a high accuracy rate of 0.9982 (99.82%)

Generalized Boosted Model (GBM)

```
ctrl_GBM <- trainControl(method = "repeatedcv", number = 3, repeats = 1)

fit_GBM  <- train(classe ~ ., data = training_data, method = "gbm",
                  trControl = ctrl_GBM, verbose = FALSE)

fit_GBM$finalModel
```

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

```
predict_GBM <- predict(fit_GBM, newdata = testing_data)

conf_matrix_GBM <- confusionMatrix(predict_GBM, factor(testing_data$classe))

conf_matrix_GBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1388    8    0    1    0
##          B    7  935    4    4    1
##          C    0    6  847    9    2
##          D    0    0    4  790   11
##          E    0    0    0    0  887
##
## Overall Statistics
##
##                Accuracy : 0.9884
```

```
##                  95% CI : (0.985, 0.9912)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9853
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9950   0.9852   0.9906   0.9826   0.9845
## Specificity            0.9974   0.9960   0.9958   0.9963   1.0000
## Pos Pred Value         0.9936   0.9832   0.9803   0.9814   1.0000
## Neg Pred Value         0.9980   0.9965   0.9980   0.9966   0.9965
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2830   0.1907   0.1727   0.1611   0.1809
## Detection Prevalence   0.2849   0.1939   0.1762   0.1642   0.1809
## Balanced Accuracy      0.9962   0.9906   0.9932   0.9895   0.9922
```

The accuracy for the GBM can be considered high with an accuracy rate of 0.9847 (98.47%).

Best Predictive Model to the Test Data

For the exercise the random forest model will be used because of its high accuracy.

```
predict_quiz <- as.data.frame(predict(fitting_rf, newdata = pml_testing))
predict_quiz
```

```
##    predict(fitting_rf, newdata = pml_testing)
## 1                                           B
## 2                                           A
## 3                                           B
## 4                                           A
## 5                                           A
## 6                                           E
## 7                                           D
## 8                                           B
## 9                                           A
## 10                                          A
## 11                                          B
## 12                                          C
## 13                                          B
## 14                                          A
## 15                                          E
## 16                                          E
## 17                                          A
## 18                                          B
## 19                                          B
## 20                                          B
```