prediction_cours8_project

Diyana Nanova

20/02/2022

Executive Summary

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. The goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and to apply a machine learning algorithm to the 20 test cases available in the test data and to submit the predictions.

Data Source

The training and test data for this project are collected using the link below:

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

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

Exercise

1.Loading of Data

```
url_train <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url_test  <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

data_train <- read.csv(url(url_train), strip.white = TRUE, na.strings = c("NA",""))
data_test  <- read.csv(url(url_test),  strip.white = TRUE, na.strings = c("NA",""))

dim(data_train)
```

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

```
dim(data_test)
```

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

2.Cleaning of Data

```
#Remove firsdt 6 columns
data_train1 <- data_train[ , -(1:6)]
data_test1  <- data_test[ , -(1:6)]

#Remove variables that are mostly NA
data_train2 <- colnames(data_train1)[!colSums(is.na(data_train1)) > 0]
train_set <- data_train[data_train2]
```

```
data_test2 <- colnames(data_test1)[!colSums(is.na(data_test1)) > 0]
test_set <- data_test[data_test2]
```

3.Split Data into random train and test

```
train_part = createDataPartition(train_set$classe, p = 0.75)[[1]]
training = train_set[ train_part,]
testing = train_set[-train_part,]

dim(training)
```

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

```
dim(testing)
```

```
## [1] 4904   54
```

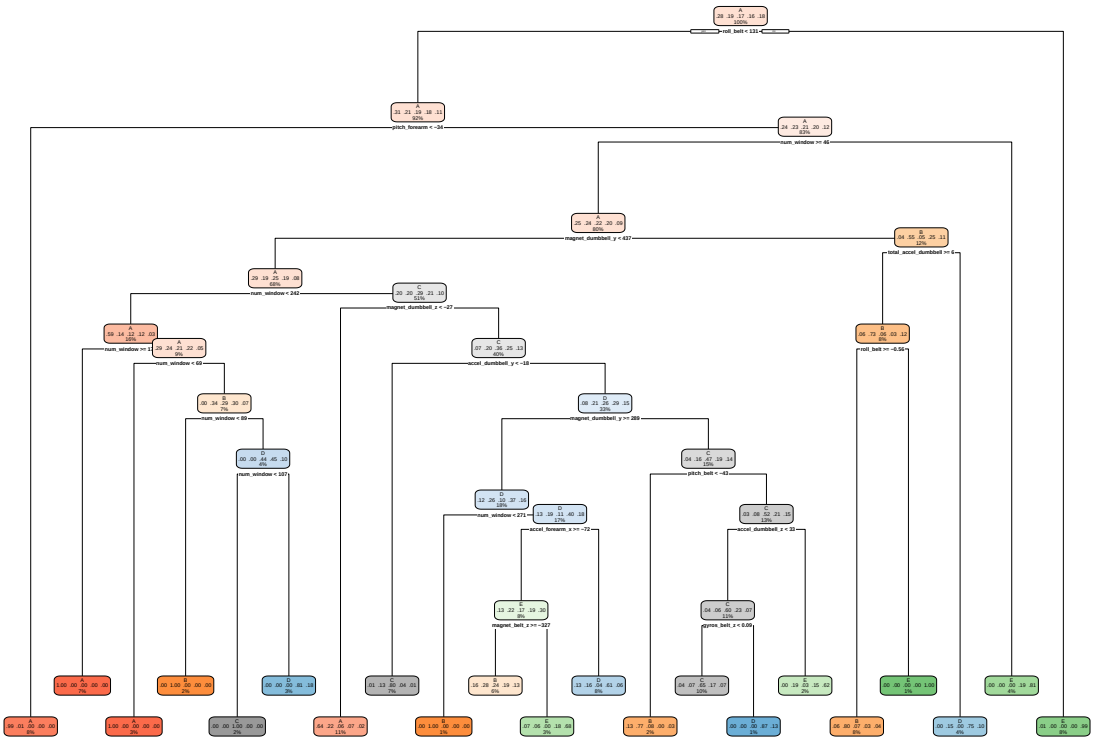
4The different modells

4.1.Decision Tree Model

```
set.seed(12345)
decision_tree_training <- rpart(classe ~ ., data = training, method="class")
rpart.plot(decision_tree_training)
```

4.1.1. Set the model

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



```
decision_tree_predicting <- predict(decision_tree_training, newdata = testing, type="class")
decision_tree_matrix <- confusionMatrix(decision_tree_predicting, factor(testing$classe))
decision_tree_matrix
```

4.1.2.Predictions of the decision tree model on testing

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1238  115   41   36   13
##          B   76  617  102   70   50
##          C   27   89  686  108   29
##          D   43   92   23  502   78
##          E   11   36    3   88  731
##
## Overall Statistics
##
##                Accuracy : 0.7696
##                  95% CI : (0.7575, 0.7813)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7082
##
##  Mcnemar's Test P-Value : 6.336e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8875   0.6502   0.8023   0.6244   0.8113
## Specificity            0.9416   0.9247   0.9375   0.9424   0.9655
## Pos Pred Value         0.8579   0.6743   0.7306   0.6802   0.8412
## Neg Pred Value         0.9546   0.9168   0.9574   0.9275   0.9579
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2524   0.1258   0.1399   0.1024   0.1491
## Detection Prevalence   0.2942   0.1866   0.1915   0.1505   0.1772
## Balanced Accuracy      0.9145   0.7874   0.8699   0.7834   0.8884
```

4.2. Random Forest Model

```
set.seed(12345)
rfm_control<- trainControl(method = "repeatedcv", number = 5, repeats = 2)
rfm_training <- train(classe ~ ., data = training, method = "rf",
                      trControl = rfm_control, verbose = FALSE)
```

4.2.1. Set the model

```
rfm_predicting <- predict(rfm_training, testing)
matrix_rfm <- confusionMatrix(rfm_predicting, factor(testing$classe))
matrix_rfm
```

4.2.2.Predictions of the random forest model

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    2    0    0    0
##          B    0  946    0    0    0
##          C    0    1  855    2    0
##          D    0    0    0  801    1
##          E    0    0    0    1  900
##
## Overall Statistics
##
##                Accuracy : 0.9986
##                  95% CI : (0.9971, 0.9994)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9982
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9968   1.0000   0.9963   0.9989
## Specificity            0.9994   1.0000   0.9993   0.9998   0.9998
## Pos Pred Value         0.9986   1.0000   0.9965   0.9988   0.9989
## Neg Pred Value         1.0000   0.9992   1.0000   0.9993   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1929   0.1743   0.1633   0.1835
## Detection Prevalence   0.2849   0.1929   0.1750   0.1635   0.1837
## Balanced Accuracy      0.9997   0.9984   0.9996   0.9980   0.9993
```

4.3.Generalized Boosted Model (GBM)

```
set.seed(12345)
gbm_control<- trainControl(method = "repeatedcv", number = 5, repeats = 2)
gbm_training <- train(classe ~ ., data = training, method = "gbm",
                      trControl = gbm_control, verbose = FALSE)
```

4.1.1. Set the model

```
gbm_predicting <- predict(gbm_training, newdata = testing)
matrix_gbm <- confusionMatrix(gbm_predicting, factor(testing$classe))
matrix_gbm
```

4.1.2.Predictions of the generalized boosted model

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 1393    9    0    0    0
##          B    2  927    7    5    1
##          C    0   13  845    6    0
##          D    0    0    2  791    4
##          E    0    0    1    2  896
##
## Overall Statistics
##
##                Accuracy : 0.9894
##                  95% CI : (0.9861, 0.9921)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9866
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9986   0.9768   0.9883   0.9838   0.9945
## Specificity            0.9974   0.9962   0.9953   0.9985   0.9993
## Pos Pred Value         0.9936   0.9841   0.9780   0.9925   0.9967
## Neg Pred Value         0.9994   0.9944   0.9975   0.9968   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2841   0.1890   0.1723   0.1613   0.1827
## Detection Prevalence   0.2859   0.1921   0.1762   0.1625   0.1833
## Balanced Accuracy      0.9980   0.9865   0.9918   0.9912   0.9969
```

5.Applying the Best Predictive Model to the Test Data

The following are the predictive accuracy of the three models:

Decision Tree Model: 74,14% Generalized Boosted Model: 98,67% Random Forest Model: 99,63%

The Random Forest Model is with a better accuracy and will be used to make predictions on the 20 data points from the original testing dataset(data_test).

```
predict_modell <- as.data.frame(predict(rfm_training, newdata = data_test))
predict_modell
```

```
##    predict(rfm_training, newdata = data_test)
## 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
```