Practical Machine Learning Project

QW

2023-06-05

Introduction about Prediction

In this prediction project, data is from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. Its goal is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. We will use any of the other variables to predict with. This report includes describing how we built our model, how we used cross validation, what we think the expected out of sample error is, and why we made the final choice. We will also use the final prediction model to predict 20 different test cases.

Read the accelerometer data

```
#Import the training data.
library(gbm)
```

```
## Loaded gbm 2.1.8.1
```

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

```
library(caret)
```

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

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

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

##
## Attaching package: 'randomForest'

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

## The following object is masked from 'package:rattle':
##
##     importance
```

```
train_data <- read.csv("C:/Users/wangq/Downloads/pml-training.csv", na.strings = c("", "NA"))
#str(train_data)
#View(train_data)

#Import the test data with 20 cases.
test_data <- read.csv("C:/Users/wangq/Downloads/pml-testing.csv", na.strings = c("", "NA"))
#str(test_data)

#Partition the training data into two datasets (75% vs 25%)
# Set up the seed for partitioning the train_data dataset.
set.seed(2023)
train <- createDataPartition(y=train_data$classe, p=0.75, list=FALSE)
new_training <- train_data[train, ]
new_testing <- train_data[-train, ]

#Data cleaning by remove both the near-zero-variance (NZV) columns, the NA columns and those 5 identifi
nzv_columns <- nearZeroVar(new_training)
new_training <- new_training[ , -nzv_columns]
new_testing  <- new_testing [ , -nzv_columns]

new_training <- new_training[, colSums(is.na(new_training)) == 0]
new_testing <- new_testing[, colSums(is.na(new_testing)) == 0]

new_training <- new_training[ , -(1:5)]
new_testing  <- new_testing[ , -(1:5)]

#Now the dimensions of both new_training and new_testing datasets were reduced from 160 columns to 54 o
dim(new_training)
```

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

```
dim(new_testing)
```

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

We tried the generalized boosted model, decision tree and random forest models using 5-folds cross validations.

```r
#Generalized Boosted Model (GBM)
gbm_model <- train(classe ~., data = new_training, method = "gbm", verbose = FALSE,
                               trControl = trainControl(method = "cv", number = 5))
gbm_model
```

```
## Stochastic Gradient Boosting
##
## 14718 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11775, 11774, 11776, 11772, 11775
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7565584  0.6911714
##   1                  100      0.8315648  0.7867185
##   1                  150      0.8731487  0.8394008
##   2                   50      0.8887062  0.8590308
##   2                  100      0.9419764  0.9265812
##   2                  150      0.9642623  0.9547845
##   3                   50      0.9317169  0.9135273
##   3                  100      0.9712609  0.9636403
##   3                  150      0.9880428  0.9848751
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```r
gbm_model$finalModel
```

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

```r
#Apply GBM model Prediction on new_testing.
pred_gbm <- predict(gbm_model, new_testing)
table(pred_gbm)
```

```
## pred_gbm
##    A    B    C    D    E
## 1399  947  874  798  886
```

```r
pred_gbm_result <- confusionMatrix(pred_gbm, factor(new_testing$classe))
pred_gbm_result
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    5    0    0    0
##          B    1  930    4    3    9
##          C    0   14  847   12    1
##          D    0    0    3  788    7
##          E    0    0    1    1  884
##
## Overall Statistics
##
##                Accuracy : 0.9876
##                  95% CI : (0.9841, 0.9905)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9843
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9800   0.9906   0.9801   0.9811
## Specificity            0.9986   0.9957   0.9933   0.9976   0.9995
## Pos Pred Value         0.9964   0.9820   0.9691   0.9875   0.9977
## Neg Pred Value         0.9997   0.9952   0.9980   0.9961   0.9958
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1896   0.1727   0.1607   0.1803
## Detection Prevalence   0.2853   0.1931   0.1782   0.1627   0.1807
## Balanced Accuracy      0.9989   0.9878   0.9920   0.9888   0.9903
```

```r
#The accuracy rate of the Generalized Boosted Model (GMB) is 98.61%.

##Decision Tree Model (DTM)
dtm_model <- rpart(classe ~ ., data = new_training, method="class")
printcp(dtm_model)
```

```
##
## Classification tree:
## rpart(formula = classe ~ ., data = new_training, method = "class")
##
## Variables actually used in tree construction:
##  [1] accel_dumbbell_y     accel_dumbbell_z     accel_forearm_x
##  [4] magnet_arm_y         magnet_dumbbell_y    magnet_dumbbell_z
##  [7] magnet_forearm_z     num_window           pitch_belt
## [10] pitch_forearm        roll_belt            roll_dumbbell
## [13] roll_forearm         total_accel_dumbbell
##
## Root node error: 10533/14718 = 0.71565
##
## n= 14718
##
```

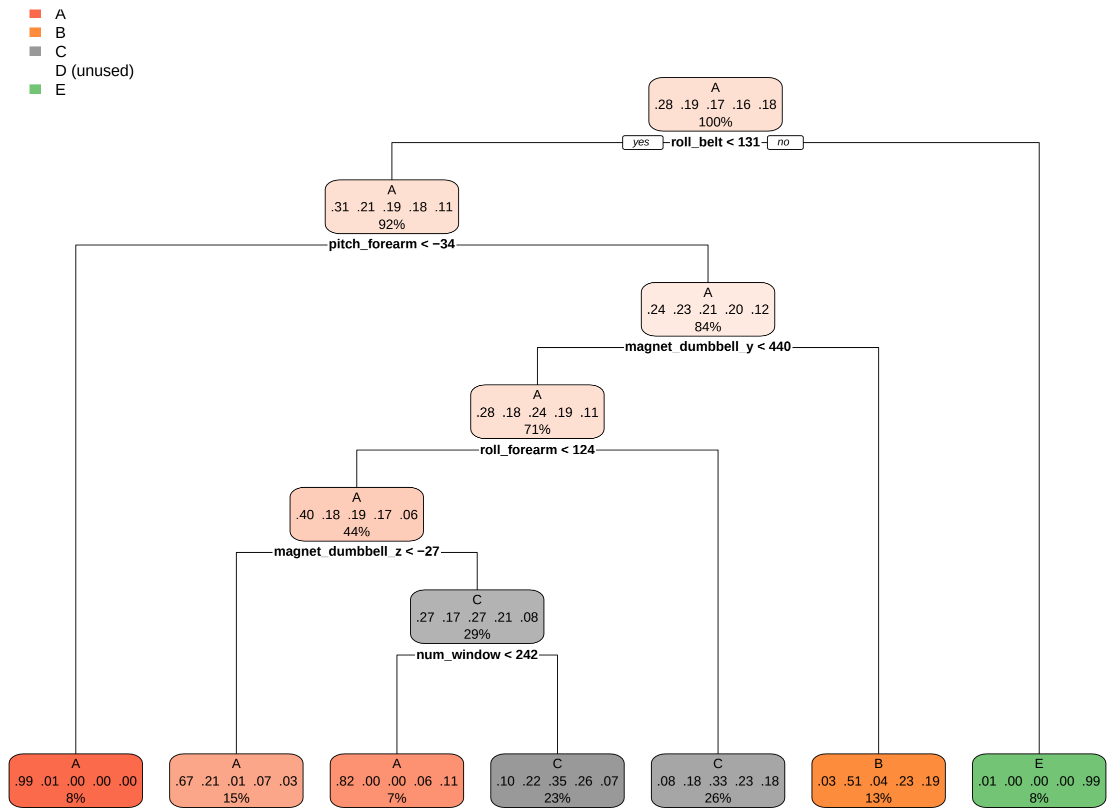
```
##          CP nsplit rel error  xerror      xstd
## 1  0.115826      0   1.00000 1.00000 0.0051957
## 2  0.058926      1   0.88417 0.88417 0.0055522
## 3  0.040017      4   0.70740 0.70891 0.0057583
## 4  0.036077      6   0.62736 0.62955 0.0057307
## 5  0.031235      7   0.59128 0.60106 0.0057025
## 6  0.023450      8   0.56005 0.56556 0.0056535
## 7  0.020032     11   0.48951 0.49350 0.0055050
## 8  0.018703     12   0.46948 0.47422 0.0054537
## 9  0.013956     14   0.43207 0.43682 0.0053392
## 10 0.013007     15   0.41811 0.42400 0.0052953
## 11 0.010443     16   0.40511 0.40492 0.0052252
## 12 0.010285     17   0.39466 0.37492 0.0051033
## 13 0.010000     22   0.33998 0.36248 0.0050484
```

```r
#fancyRpartPlot(dtm_model)

prune.dtm_model <- prune(dtm_model, cp=0.04)   #prune the tree with cp=0.04
printcp(prune.dtm_model)
```

```
##
## Classification tree:
## rpart(formula = classe ~ ., data = new_training, method = "class")
##
## Variables actually used in tree construction:
## [1] magnet_dumbbell_y magnet_dumbbell_z num_window        pitch_forearm
## [5] roll_belt         roll_forearm
##
## Root node error: 10533/14718 = 0.71565
##
## n= 14718
##
##         CP nsplit rel error  xerror      xstd
## 1 0.115826      0   1.00000 1.00000 0.0051957
## 2 0.058926      1   0.88417 0.88417 0.0055522
## 3 0.040017      4   0.70740 0.70891 0.0057583
## 4 0.040000      6   0.62736 0.62955 0.0057307
```

```r
#windows()
rpart.plot(prune.dtm_model)              #pruned tree
```




```r
#dev.off()

##Apply DTM model Prediction on new_testing.
pred_dtm <- predict(dtm_model, new_testing, type="class")
table(pred_dtm)
```

```
## pred_dtm
##    A    B    C    D    E
## 1565  904  985  668  782
```

```r
pred_dtm_result <- confusionMatrix(pred_dtm, factor(new_testing$classe))
pred_dtm_result
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1275  168   30   51   41
##          B   46  574   68   87  129
##          C   22   87  687  115   74
##          D   34   71   44  473   46
##          E   18   49   26   78  611
##
## Overall Statistics
```

```
##
##                Accuracy : 0.7382
##                  95% CI : (0.7256, 0.7504)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6673
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9140   0.6048   0.8035  0.58831   0.6781
## Specificity            0.9174   0.9166   0.9264  0.95244   0.9573
## Pos Pred Value         0.8147   0.6350   0.6975  0.70808   0.7813
## Neg Pred Value         0.9641   0.9062   0.9571  0.92186   0.9296
## Prevalence             0.2845   0.1935   0.1743  0.16395   0.1837
## Detection Rate         0.2600   0.1170   0.1401  0.09645   0.1246
## Detection Prevalence   0.3191   0.1843   0.2009  0.13622   0.1595
## Balanced Accuracy      0.9157   0.7607   0.8650  0.77037   0.8177
```

```
#The accuracy rate of the Decision Tree Model (DTM) is 81.89%, which is lower than GMB model prediction


##Random Forest model (RFM)
rfm_model <- train(classe ~., data = new_training, method = "rf",
                trControl = trainControl("cv", number = 5))
rfm_model
```

```
## Random Forest
##
## 14718 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11775, 11776, 11774, 11774, 11773
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9935457  0.9918346
##   27    0.9965350  0.9956170
##   53    0.9932740  0.9914920
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```
rfm_model$finalModel
```

```
##
## Call:
```

```
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.21%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4184    1    0    0    0 0.0002389486
## B    4 2842    1    1    0 0.0021067416
## C    0    6 2561    0    0 0.0023373588
## D    0    0    9 2402    1 0.0041459370
## E    0    0    0    8 2698 0.0029563932
```

```
##Apply RFM model Prediction on new_testing.
pred_rfm <- predict(rfm_model, new_testing)
table(pred_rfm)
```

```
## pred_rfm
##    A    B    C    D    E
## 1395  948  857  805  899
```

```
pred_rfm_result <- confusionMatrix(pred_rfm, factor(new_testing$classe))
pred_rfm_result
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    1    0    0    0
##          B    0  947    1    0    0
##          C    0    1  854    2    0
##          D    0    0    0  802    3
##          E    1    0    0    0  898
##
## 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            0.9993   0.9979   0.9988   0.9975   0.9967
## Specificity            0.9997   0.9997   0.9993   0.9993   0.9998
## Pos Pred Value         0.9993   0.9989   0.9965   0.9963   0.9989
## Neg Pred Value         0.9997   0.9995   0.9998   0.9995   0.9993
```

```
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1931   0.1741   0.1635   0.1831
## Detection Prevalence   0.2845   0.1933   0.1748   0.1642   0.1833
## Balanced Accuracy      0.9995   0.9988   0.9990   0.9984   0.9982
```

The accuracy rate of the Random Forest model (RFM) is 99.82%, which is close to 100% and higher than other models (GMB and DTM) prediction above. And the out of sample error is almost zero.

Overall, we choose the random forest model as the best predictive model based on its accuracy rate nearly 100% and the expected out of sample error close to zero.

Use Random Forest model (RFM) model to predict 20 different test cases in test_data.

```r
quiz_predict <- as.data.frame(predict(rfm_model, newdata = test_data))

#Obtain the answers for the prediction quiz
quiz_predict
```

```
##    predict(rfm_model, newdata = test_data)
## 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
```