Practical Machine Learning_project

weimf

1/8/2022

Introduction

In this project, I will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to quantify how well they do a particular activity.

Data preparations

I start by installing some useful packages.

```
## 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.
```

Then, loading and browsing the data.

```
#Loading data
train <- read.csv("./pml-training.csv")
test <- read.csv("./pml-testing.csv")

#Browsing the data
dim(train)
```

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

```
dim(test)
```

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

After loading the data, I split the data into training set (‘trainset’) and validation set (‘validset’).

```
#setting seed
set.seed(123)

#split the data
inTrain  <- createDataPartition(y=train$classe, p=0.7, list=F)
trainset <- train[inTrain, ]
validset  <- train[-inTrain, ]
```

Cleaning the data

Steps to clean the data are applied to both 'trainset' and 'validset':
1.removing N/A variables
2.cleaning near zero variance variables
3.removing irrelevant variables

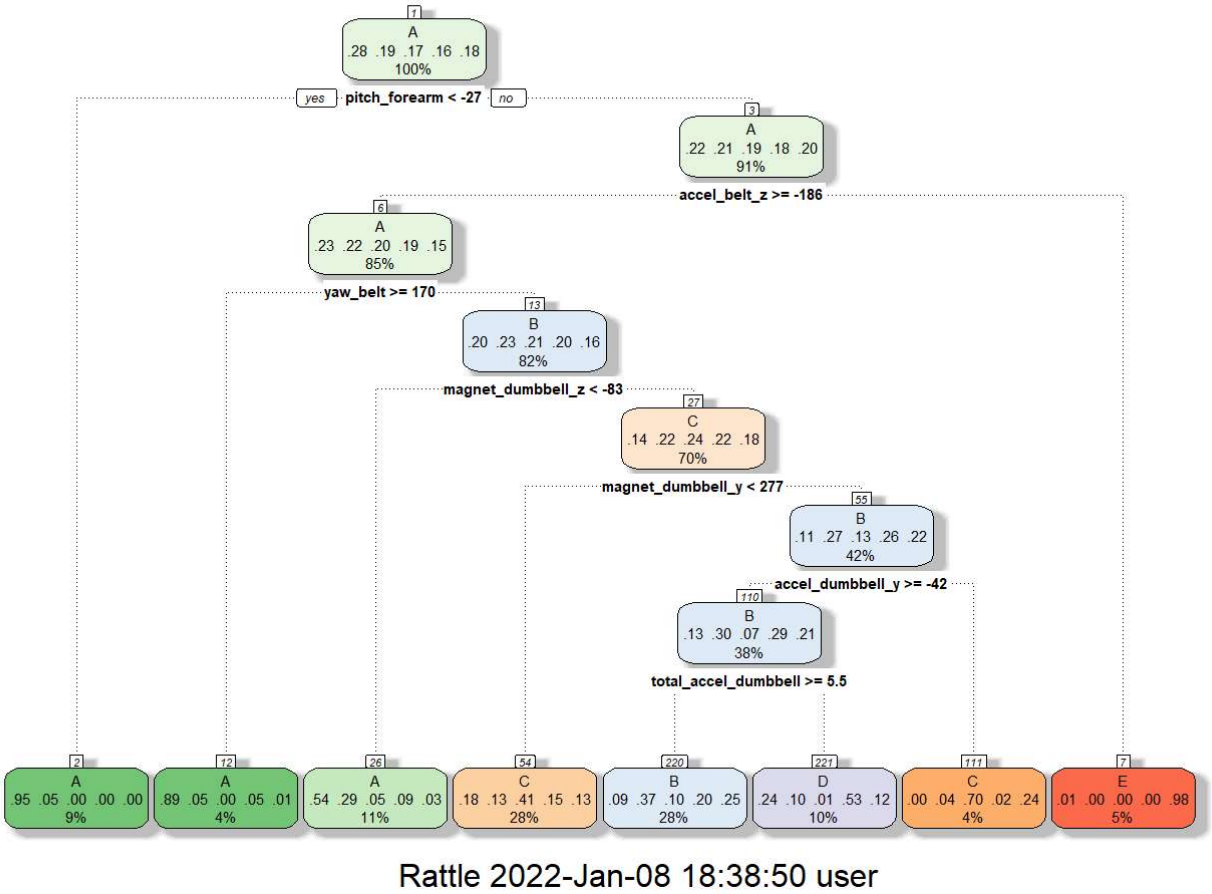
Building model

I will use the Decision Trees and Random Forest to compare the accuracy of each model.

```
# use 3-fold cross validation to select optimal parameters
Control <- trainControl(method="cv", number=3, verboseIter=F)
```

1. Decision Trees

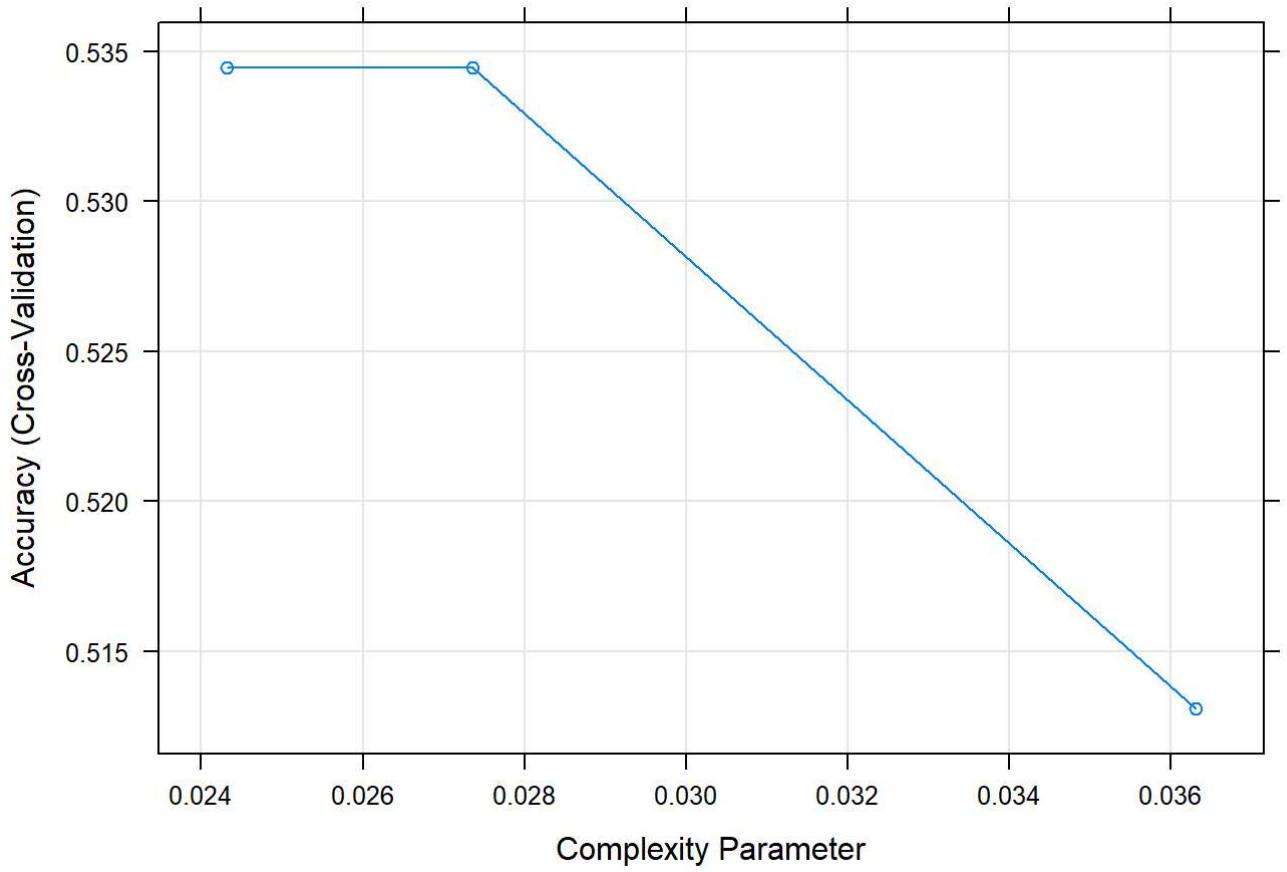
```
model_trees <- train(classe~., data=trainset, method="rpart", trControl = Control)
fancyRpartPlot(model_trees$finalModel)
```




```
prediction_trees <- predict(model_trees, newdata=validset)
cm_trees <- confusionMatrix(prediction_trees, factor(validset$classe))
cm_trees
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1050  235   32   69   18
##          B  167  644  187  314  399
##          C  283  206  801  264  277
##          D  169   54    6  317   73
##          E    5    0    0    0  315
##
## Overall Statistics
##
##                Accuracy : 0.5314
##                  95% CI : (0.5185, 0.5442)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4101
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6272   0.5654   0.7807  0.32884  0.29113
## Specificity            0.9159   0.7752   0.7880  0.93863  0.99896
## Pos Pred Value         0.7479   0.3764   0.4375  0.51212  0.98438
## Neg Pred Value         0.8607   0.8814   0.9445  0.87714  0.86217
## Prevalence             0.2845   0.1935   0.1743  0.16381  0.18386
## Detection Rate         0.1784   0.1094   0.1361  0.05387  0.05353
## Detection Prevalence   0.2386   0.2907   0.3111  0.10518  0.05438
## Balanced Accuracy      0.7716   0.6703   0.7844  0.63373  0.64504
```

```
plot(model_trees)
```



Noted that the accuracy of this model is 0.5366.

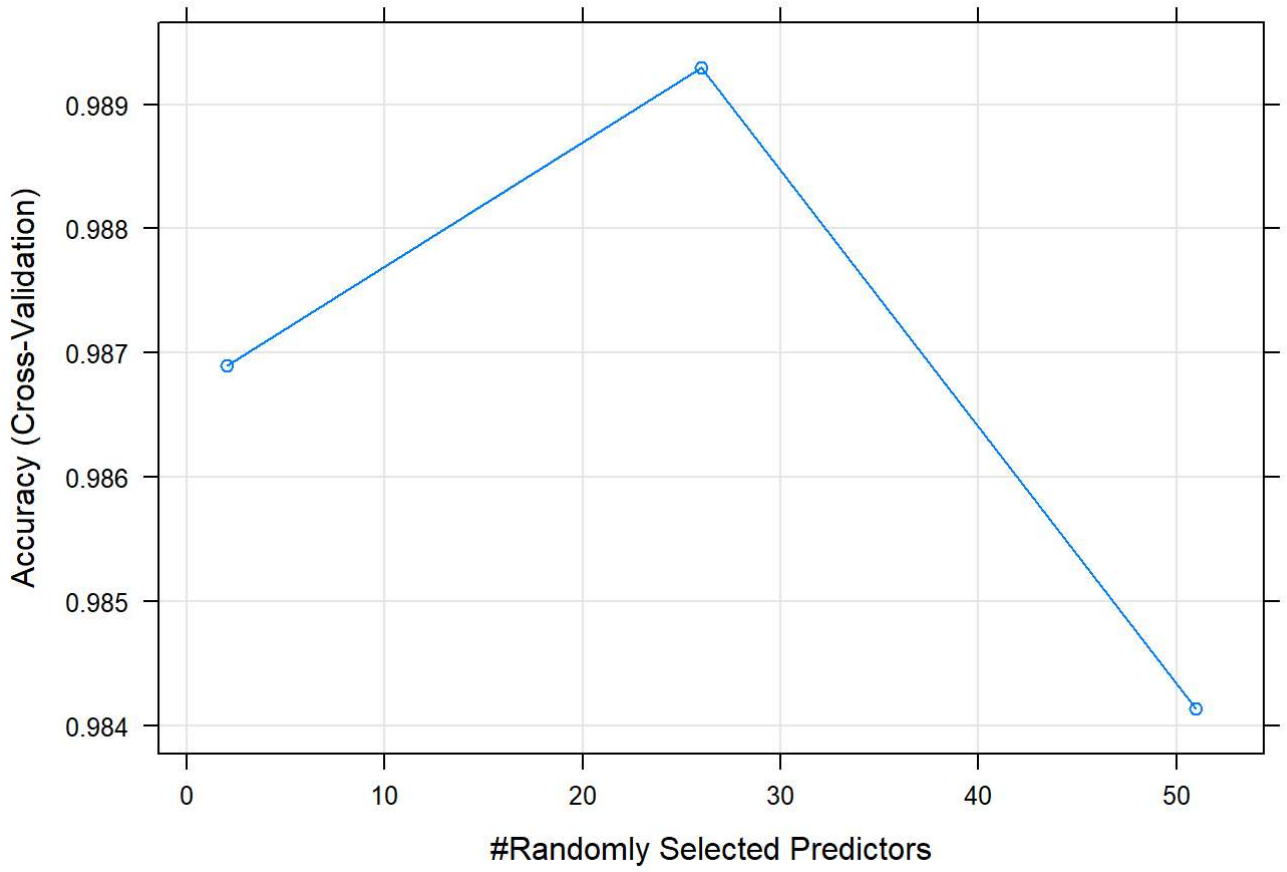
(2)Random Forest

```
model_rf <- train(classe~., data=trainset, method="rf", trControl = Control)

prediction_rf <- predict(model_rf, newdata=validset)
cm_rf <- confusionMatrix(prediction_rf, factor(validset$classe))
cm_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    8    0    0    0
##          B    0 1126    6    0    0
##          C    0    5 1019    6    4
##          D    0    0    1  957    4
##          E    0    0    0    1 1074
##
## Overall Statistics
##
##                Accuracy : 0.9941
##                  95% CI : (0.9917, 0.9959)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9925
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9886   0.9932   0.9927   0.9926
## Specificity            0.9981   0.9987   0.9969   0.9990   0.9998
## Pos Pred Value         0.9952   0.9947   0.9855   0.9948   0.9991
## Neg Pred Value         1.0000   0.9973   0.9986   0.9986   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1913   0.1732   0.1626   0.1825
## Detection Prevalence   0.2858   0.1924   0.1757   0.1635   0.1827
## Balanced Accuracy      0.9991   0.9937   0.9950   0.9959   0.9962
```

```
plot(model_rf)
```



The result indicates that the Random Forest model generates 0.9941 accuracy and thus, will be a better fit to predict the data.

Predictions on Test Set

Finally, I run the `test' data to predict the classe outcome for 20 cases with the Random Forest model.

```
prediction <- predict(model_rf, newdata=test)
print(prediction)
```

```
##  [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
```