

# Machine Learning Prediction Project

*Yue Liu*

## Summary

Devices such as Jawbone Up, Nike FuelBand, and Fitbit are now possible to collect a large amount of data about personal activity relatively inexpensively. However, People regularly do is quantify how much of their daily exercise rather than how well they do it.

This report is based on training data and we will use testing data to validate our model for prediction performance. Random forest will be undertaken for this project.

## Data Cleaning

```
# Load necessary R packages
library(rpart)
library(caret)

## Loading required package: lattice
## Loading required package: ggplot2
library(rattle)

## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
library(randomForest)

## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:rattle':
##
##     importance
## The following object is masked from 'package:ggplot2':
##
##     margin
library(rpart.plot)

# Load training and testing data
Train_data <- read.csv("pml-training.csv", header = TRUE)
Test_data <- read.csv("pml-testing.csv", header = TRUE)

# Investigate Dependent Variable and Training Sample
str(Train_data)

## 'data.frame':   19622 obs. of  160 variables:
## $ X               : int  1 2 3 4 5 6 7 8 9 10 ...
```

```

## $ user_name : Factor w/ 6 levels "adelmo","carlitos",...: 2 2 2 2 2 2 2 2 2 ...
## $ raw_timestamp_part_1 : int 1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 ...
## $ raw_timestamp_part_2 : int 788290 808298 820366 120339 196328 304277 368296 440390 484323 484...
## $ cvtd_timestamp : Factor w/ 20 levels "02/12/2011 13:32",...: 9 9 9 9 9 9 9 9 9 ...
## $ new_window : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 ...
## $ num_window : int 11 11 11 12 12 12 12 12 12 ...
## $ roll_belt : num 1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
## $ pitch_belt : num 8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
## $ yaw_belt : num -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
## $ total_accel_belt : int 3 3 3 3 3 3 3 3 3 ...
## $ kurtosis_roll_belt : Factor w/ 397 levels "", "-0.016850",...: 1 1 1 1 1 1 1 1 1 ...
## $ kurtosis_pitch_belt : Factor w/ 317 levels "", "-0.021887",...: 1 1 1 1 1 1 1 1 1 ...
## $ kurtosis_yaw_belt : Factor w/ 2 levels "", "#DIV/0!": 1 1 1 1 1 1 1 1 1 ...
## $ skewness_roll_belt : Factor w/ 395 levels "", "-0.003095",...: 1 1 1 1 1 1 1 1 1 ...
## $ skewness_roll_belt.1 : Factor w/ 338 levels "", "-0.005928",...: 1 1 1 1 1 1 1 1 1 ...
## $ skewness_yaw_belt : Factor w/ 2 levels "", "#DIV/0!": 1 1 1 1 1 1 1 1 1 ...
## $ max_roll_belt : num NA NA NA NA NA NA NA NA NA NA ...
## $ max_pitch_belt : int NA NA NA NA NA NA NA NA NA NA ...
## $ max_yaw_belt : Factor w/ 68 levels "", "-0.1", "-0.2",...: 1 1 1 1 1 1 1 1 1 ...
## $ min_roll_belt : num NA NA NA NA NA NA NA NA NA NA ...
## $ min_pitch_belt : int NA NA NA NA NA NA NA NA NA NA ...
## $ min_yaw_belt : Factor w/ 68 levels "", "-0.1", "-0.2",...: 1 1 1 1 1 1 1 1 1 ...
## $ amplitude_roll_belt : num NA NA NA NA NA NA NA NA NA NA ...
## $ amplitude_pitch_belt : int NA NA NA NA NA NA NA NA NA NA ...
## $ amplitude_yaw_belt : Factor w/ 4 levels "", "#DIV/0!", "0.00",...: 1 1 1 1 1 1 1 1 1 ...
## $ var_total_accel_belt : num NA NA NA NA NA NA NA NA NA NA ...
## $ avg_roll_belt : num NA NA NA NA NA NA NA NA NA NA ...
## $ stddev_roll_belt : num NA NA NA NA NA NA NA NA NA NA ...
## $ var_roll_belt : num NA NA NA NA NA NA NA NA NA NA ...
## $ avg_pitch_belt : num NA NA NA NA NA NA NA NA NA NA ...
## $ stddev_pitch_belt : num NA NA NA NA NA NA NA NA NA NA ...
## $ var_pitch_belt : num NA NA NA NA NA NA NA NA NA NA ...
## $ avg_yaw_belt : num NA NA NA NA NA NA NA NA NA NA ...
## $ stddev_yaw_belt : num NA NA NA NA NA NA NA NA NA NA ...
## $ var_yaw_belt : num NA NA NA NA NA NA NA NA NA NA ...
## $ gyros_belt_x : num 0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
## $ gyros_belt_y : num 0 0 0 0 0.02 0 0 0 0 0 ...
## $ gyros_belt_z : num -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
## $ accel_belt_x : int -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
## $ accel_belt_y : int 4 4 5 3 2 4 3 4 2 4 ...
## $ accel_belt_z : int 22 22 23 21 24 21 21 21 24 22 ...
## $ magnet_belt_x : int -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
## $ magnet_belt_y : int 599 608 600 604 600 603 599 603 602 609 ...
## $ magnet_belt_z : int -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
## $ roll_arm : num -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
## $ pitch_arm : num 22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
## $ yaw_arm : num -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
## $ total_accel_arm : int 34 34 34 34 34 34 34 34 34 ...
## $ var_accel_arm : num NA NA NA NA NA NA NA NA NA NA ...
## $ avg_roll_arm : num NA NA NA NA NA NA NA NA NA NA ...
## $ stddev_roll_arm : num NA NA NA NA NA NA NA NA NA NA ...
## $ var_roll_arm : num NA NA NA NA NA NA NA NA NA NA ...
## $ avg_pitch_arm : num NA NA NA NA NA NA NA NA NA NA ...
## $ stddev_pitch_arm : num NA NA NA NA NA NA NA NA NA NA ...

```

```
## $ var_pitch_arm : num NA NA NA NA NA NA NA NA NA NA ...
## $ avg_yaw_arm : num NA NA NA NA NA NA NA NA NA NA ...
## $ stddev_yaw_arm : num NA NA NA NA NA NA NA NA NA NA ...
## $ var_yaw_arm : num NA NA NA NA NA NA NA NA NA NA ...
## $ gyros_arm_x : num 0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
## $ gyros_arm_y : num 0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
## $ gyros_arm_z : num -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
## $ accel_arm_x : int -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
## $ accel_arm_y : int 109 110 110 111 111 111 111 111 109 110 ...
## $ accel_arm_z : int -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
## $ magnet_arm_x : int -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
## $ magnet_arm_y : int 337 337 344 344 337 342 336 338 341 334 ...
## $ magnet_arm_z : int 516 513 513 512 506 513 509 510 518 516 ...
## $ kurtosis_roll_arm : Factor w/ 330 levels "", "-0.02438",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ kurtosis_pitch_arm : Factor w/ 328 levels "", "-0.00484",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ kurtosis_yaw_arm : Factor w/ 395 levels "", "-0.01548",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ skewness_roll_arm : Factor w/ 331 levels "", "-0.00051",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ skewness_pitch_arm : Factor w/ 328 levels "", "-0.00184",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ skewness_yaw_arm : Factor w/ 395 levels "", "-0.00311",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ max_roll_arm : num NA NA NA NA NA NA NA NA NA NA ...
## $ max_pitch_arm : num NA NA NA NA NA NA NA NA NA NA ...
## $ max_yaw_arm : int NA NA NA NA NA NA NA NA NA NA ...
## $ min_roll_arm : num NA NA NA NA NA NA NA NA NA NA ...
## $ min_pitch_arm : num NA NA NA NA NA NA NA NA NA NA ...
## $ min_yaw_arm : int NA NA NA NA NA NA NA NA NA NA ...
## $ amplitude_roll_arm : num NA NA NA NA NA NA NA NA NA NA ...
## $ amplitude_pitch_arm : num NA NA NA NA NA NA NA NA NA NA ...
## $ amplitude_yaw_arm : int NA NA NA NA NA NA NA NA NA NA ...
## $ roll_dumbbell : num 13.1 13.1 12.9 13.4 13.4 ...
## $ pitch_dumbbell : num -70.5 -70.6 -70.3 -70.4 -70.4 ...
## $ yaw_dumbbell : num -84.9 -84.7 -85.1 -84.9 -84.9 ...
## $ kurtosis_roll_dumbbell : Factor w/ 398 levels "", "-0.0035", "-0.0073",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ kurtosis_pitch_dumbbell : Factor w/ 401 levels "", "-0.0163", "-0.0233",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ kurtosis_yaw_dumbbell : Factor w/ 2 levels "", "#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
## $ skewness_roll_dumbbell : Factor w/ 401 levels "", "-0.0082", "-0.0096",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ skewness_pitch_dumbbell : Factor w/ 402 levels "", "-0.0053", "-0.0084",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ skewness_yaw_dumbbell : Factor w/ 2 levels "", "#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
## $ max_roll_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...
## $ max_pitch_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...
## $ max_yaw_dumbbell : Factor w/ 73 levels "", "-0.1", "-0.2",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ min_roll_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...
## $ min_pitch_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...
## $ min_yaw_dumbbell : Factor w/ 73 levels "", "-0.1", "-0.2",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ amplitude_roll_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...
## [list output truncated]
```

```
summary(Train_data$classe)
```

```
##      A      B      C      D      E
## 5580 3797 3422 3216 3607
```

There are 19622 observations and 160 columns in training sample. While dependent variable is a factor variable with 5 levels. It is unlikely to use all 159 columns for prediction, so the following section will perform variable reduction process.

```

# Calcualte NAs for each column
na_count <-sapply(Train_data, function(y) sum(is.na(y))/19216)
head(na_count)

##           X           user_name raw_timestamp_part_1
##           0           0           0
## raw_timestamp_part_2      cvtd_timestamp      new_window
##           0           0           0

table(na_count)

## na_count
##  0  1
## 93 67

# There are 67 columns with all missing values.Remove Columns with all NAs.
# We want to remove the same columns for both Training and Testing Data.

Train_data_mod <- Train_data[,
                             !names(Train_data)%in%names(na_count[na_count==1])]
Test_data_mod <- Test_data[,
                          !names(Test_data)%in%names(na_count[na_count==1])]

# A separate sample is split from training data to validate prediction model.
set.seed(1123)
Random_sample <- createDataPartition(Train_data_mod$classe, p = 0.7, list = FALSE)

Train_Use <- Train_data_mod[Random_sample, ]
Test_Use <- Train_data_mod[-Random_sample, ]

```

## Modeling Procedure

### Random Forest

There is a limitation of levels for categorical variables, therefore, before performing random forest algorithm, another variable reduction is needed.

```

# Separate the dependent variable
classe<-Train_Use$classe

factor_count <-sapply(Train_Use, function(y) sum(is.factor(y)))
factor_name <- names(factor_count[factor_count==1])
Train_Use_mod <- Train_Use[,
                          !names(Train_Use)%in%factor_name]

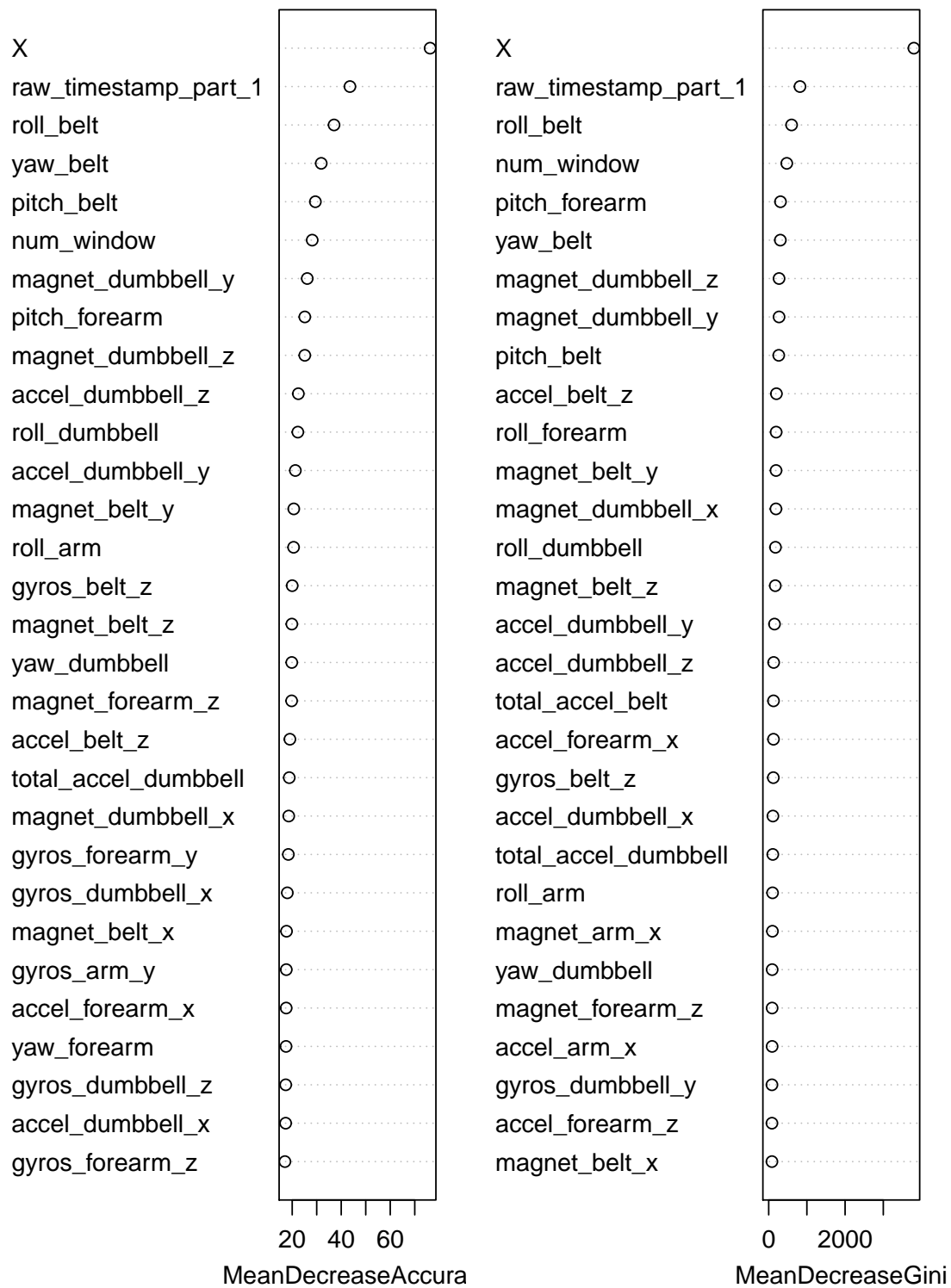
# Create Random Forest

fit_rf <- randomForest(classe~., data=Train_Use_mod
                      ,importance=TRUE
                      , ntree=500 )

# Plot the Random Forest Variable Importance
varImpPlot(fit_rf,main="Variable Importance by Random Forest")

```

## Variable Importance by Random Forest

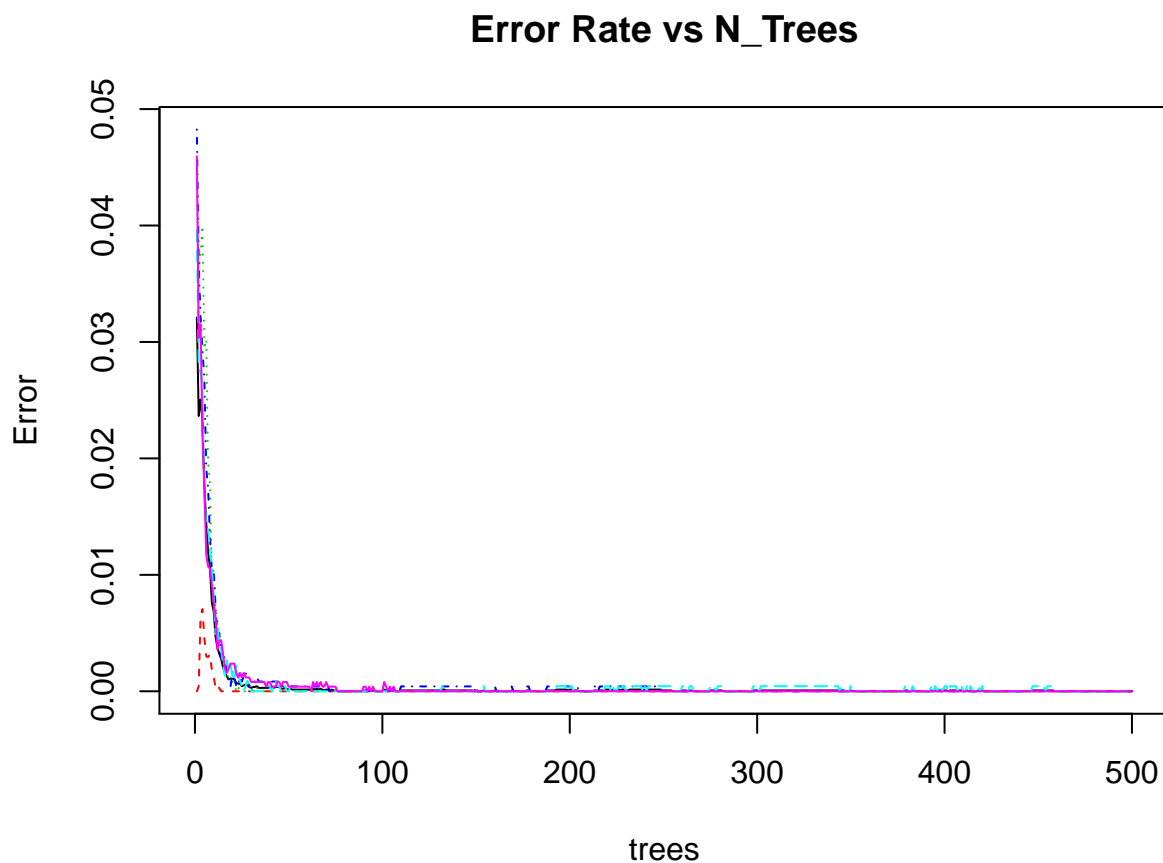


```

print(fit_rf)

##
## Call:
##  randomForest(formula = classe ~ ., data = Train_Use_mod, importance = TRUE,      ntree = 500)
##              Type of random forest: classification
##              Number of trees: 500
## No. of variables tried at each split: 7
##
##              OOB estimate of  error rate: 0%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3906     0     0     0     0         0
## B     0 2658     0     0     0         0
## C     0     0 2396     0     0         0
## D     0     0     0 2252     0         0
## E     0     0     0     0 2525         0
plot(fit_rf,main="Error Rate vs N_Trees")

```



We noticed that the Error rate drops after 100 trees, meanwhile, on testing data, error rate of 0.17% seems a good result. Now we will test this tree on testing sample:

## Validation

```
predict_test <- predict(fit_rf, Test_Use, type = "class")
confusionMatrix(Test_Use$classe, predict_test)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    A    B    C    D    E
##      A 1674    0    0    0    0
##      B    0 1139    0    0    0
##      C    0    1 1025    0    0
##      D    0    0    0  964    0
##      E    0    0    0    0 1082
##
## Overall Statistics
##
##              Accuracy : 0.9998
##              95% CI : (0.9991, 1)
##      No Information Rate : 0.2845
##      P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.9998
##  McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##              Class: A Class: B Class: C Class: D Class: E
## Sensitivity          1.0000  0.9991  1.0000  1.0000  1.0000
## Specificity          1.0000  1.0000  0.9998  1.0000  1.0000
## Pos Pred Value       1.0000  1.0000  0.9990  1.0000  1.0000
## Neg Pred Value       1.0000  0.9998  1.0000  1.0000  1.0000
## Prevalence           0.2845  0.1937  0.1742  0.1638  0.1839
## Detection Rate       0.2845  0.1935  0.1742  0.1638  0.1839
## Detection Prevalence 0.2845  0.1935  0.1743  0.1638  0.1839
## Balanced Accuracy    1.0000  0.9996  0.9999  1.0000  1.0000
```

The Overall Accuracy rate is 0.9987. The in-training and testing validation illustrate that the random forest method is stable across different samples, which leads to high accuracy for this problem. Then we will perform final testing on the dataset.

## Final Prediction

```
orig_predict_test <- predict(fit_rf, Test_data_mod, type = "class")
orig_predict_test
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  A  A  A  A  A  A  A  A  A  A  A  A  A  B  A  A  A  A  A  B
## Levels: A B C D E
```