PracticalMachineLearning-CourseAssignment

Rakesh Bhatnagar

10 July 2016

Load the Training & Testing Data. The data is pre-downloaded from below websites.
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv) and
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv) websites.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.5
```

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

```
## Warning: package 'ggplot2' was built under R version 3.2.3
```

```
setwd("D:/Rakesh/MYWORK/RSpace/PML")
traindata <- read.csv("pml-training.csv",stringsAsFactors=FALSE,na.strings=c("","NA"))

testingdata <- read.csv("pml-testing.csv",stringsAsFactors=FALSE,na.strings=c("","NA"))
```

Perform data preprocessing. Remove the columns that are not required for building model.

```
traindata <- traindata[,-c(1:7)]
testingdata <- testingdata[,-c(1:7)]
```

Few columns have good amount of missing data. These can be removed for model building Removing the columns that have 95% or more missing data.

```
str(traindata)
```

```
## 'data.frame':    19622 obs. of  153 variables:
##  $ 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 -9
4.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : chr  NA NA NA NA ...
##  $ kurtosis_picth_belt     : chr  NA NA NA NA ...
##  $ kurtosis_yaw_belt       : chr  NA NA NA NA ...
##  $ skewness_roll_belt      : chr  NA NA NA NA ...
##  $ skewness_roll_belt.1    : chr  NA NA NA NA ...
##  $ skewness_yaw_belt       : chr  NA NA NA NA ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : chr  NA NA NA NA ...
##  $ 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            : chr  NA NA NA NA ...
##  $ 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      : chr  NA NA NA NA ...
##  $ 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 -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 -161 ...
##  $ total_accel_arm         : int  34 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       : chr  NA NA NA NA ...
##  $ kurtosis_picth_arm      : chr  NA NA NA NA ...
##  $ kurtosis_yaw_arm        : chr  NA NA NA NA ...
##  $ skewness_roll_arm       : chr  NA NA NA NA ...
##  $ skewness_pitch_arm      : chr  NA NA NA NA ...
##  $ skewness_yaw_arm        : chr  NA NA NA NA ...
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_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  : chr  NA NA NA NA ...
##  $ kurtosis_picth_dumbbell : chr  NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell   : chr  NA NA NA NA ...
##  $ skewness_roll_dumbbell  : chr  NA NA NA NA ...
##  $ skewness_pitch_dumbbell : chr  NA NA NA NA ...
##  $ skewness_yaw_dumbbell   : chr  NA NA NA NA ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : chr  NA NA NA NA ...
##  $ 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        : chr  NA NA NA NA ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_dumbbell: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_dumbbell  : chr  NA NA NA NA ...
##  $ total_accel_dumbbell    : int  37 37 37 37 37 37 37 37 37 37 ...
##  $ var_accel_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_dumbbell    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

```
nr<-dim(traindata)[1]
traindata <- traindata[ , colSums(is.na(traindata)) <= 0.95*nr]
preObj <- preProcess(traindata[,-53],method=c("center","scale"))
trainScaleData<-predict(preObj,traindata[,-53])
testingdata<-testingdata[,colnames(trainScaleData)]
alltrain <- data.frame(trainScaleData,classe=traindata[,53])
```

Build the validation data from the training data. Taking 50% of the training data to be validation.

```
indxTrain<-createDataPartition(y=alltrain$classe,p=0.5,list=FALSE)
training<-traindata[indxTrain,]
validation<-traindata[-indxTrain,]
str(training)
```

```
## 'data.frame':    9812 obs. of  53 variables:
##  $ roll_belt           : num  1.41 1.42 1.48 1.45 1.42 1.43 1.42 1.45 1.48 1.51 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.06 8.13 8.18 8.21 8.2 8.15 8.12 ...
##  $ 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 3 ...
##  $ gyros_belt_x        : num  0.02 0 0.02 0.02 0.02 0.02 0.02 0 0 0 ...
##  $ gyros_belt_y        : num  0 0 0.02 0 0 0 0 0 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 0 0 -0.02 ...
##  $ accel_belt_x        : int  -22 -20 -21 -21 -22 -22 -22 -21 -21 -21 ...
##  $ accel_belt_y        : int  4 5 2 4 4 2 4 2 4 4 ...
##  $ accel_belt_z        : int  22 23 24 21 21 23 21 22 23 22 ...
##  $ magnet_belt_x       : int  -7 -2 -6 0 -2 -2 -8 -1 0 -6 ...
##  $ magnet_belt_y       : int  608 600 600 603 603 602 598 597 592 598 ...
##  $ magnet_belt_z       : int  -311 -305 -302 -312 -313 -319 -310 -310 -305 -317 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -129 -129 -129 ...
##  $ pitch_arm           : num  22.5 22.5 22.1 22 21.8 21.5 21.4 21.4 21.3 21.3 ...
##  $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm     : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x         : num  0.02 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  -0.02 -0.02 -0.03 -0.03 -0.02 -0.03 0 0 0 0 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 0 0 0 0 -0.03 -0.03 -0.03 -0.02 ...
##  $ accel_arm_x         : int  -290 -289 -289 -289 -289 -288 -288 -289 -289 -289 ...
##  $ accel_arm_y         : int  110 110 111 111 111 111 111 111 109 110 ...
##  $ accel_arm_z         : int  -125 -126 -123 -122 -124 -123 -124 -124 -121 -122 ...
##  $ magnet_arm_x        : int  -369 -368 -374 -369 -372 -363 -371 -374 -367 -371 ...
##  $ magnet_arm_y        : int  337 344 337 342 338 343 331 342 340 337 ...
##  $ magnet_arm_z        : int  513 513 506 513 510 520 523 510 509 512 ...
##  $ roll_dumbbell       : num  13.1 12.9 13.4 13.4 12.8 ...
##  $ pitch_dumbbell      : num  -70.6 -70.3 -70.4 -70.8 -70.3 ...
##  $ yaw_dumbbell        : num  -84.7 -85.1 -84.9 -84.5 -85.1 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0.02 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02
 ...
##  $ gyros_dumbbell_z    : num  0 0 0 0 0 0 -0.02 0 0 0 ...
##  $ accel_dumbbell_x    : int  -233 -232 -233 -234 -234 -233 -234 -234 -233 -233 ...
##  $ accel_dumbbell_y    : int  47 46 48 48 46 47 48 47 48 47 ...
##  $ accel_dumbbell_z    : int  -269 -270 -270 -269 -272 -270 -268 -270 -271 -272 ...
##  $ magnet_dumbbell_x   : int  -555 -561 -554 -558 -555 -554 -554 -554 -554 -551 ...
##  $ magnet_dumbbell_y   : int  296 298 292 294 300 291 295 294 297 296 ...
##  $ magnet_dumbbell_z   : num  -64 -63 -68 -66 -74 -65 -68 -63 -73 -56 ...
##  $ roll_forearm        : num  28.3 28.3 28 27.9 27.8 27.5 27.2 27.2 27.1 27.1 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.9 -63.9 -64 -64 ...
##  $ yaw_forearm         : num  -153 -152 -152 -152 -152 -152 -151 -151 -151 -151 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.02 0.03 0.02 0.02 0.02 0.02 0 0 0.02 0.02 ...
##  $ gyros_forearm_y     : num  0 -0.02 0 -0.02 -0.02 0.02 -0.02 -0.02 0 -0.02 ...
##  $ gyros_forearm_z     : num  -0.02 0 -0.02 -0.03 0 -0.03 -0.03 -0.02 0 0 ...
##  $ accel_forearm_x     : int  192 196 189 193 193 191 193 192 194 192 ...
##  $ accel_forearm_y     : int  203 204 206 203 205 203 202 201 204 204 ...
##  $ accel_forearm_z     : int  -216 -213 -214 -215 -213 -215 -214 -214 -215 -213 ...
##  $ magnet_forearm_x    : int  -18 -18 -17 -9 -9 -11 -14 -16 -13 -13 ...
##  $ magnet_forearm_y    : num  661 658 655 660 660 657 659 656 656 653 ...
##  $ magnet_forearm_z    : num  473 469 473 478 474 478 478 472 471 481 ...
##  $ classe              : chr  "A" "A" "A" "A" ...
```

Since the problem is to predict the class. We can start by fitting the basic tree model.

```
mod1 <- train(classe~.,"rpart",data=training)
```

```
## Loading required package: rpart
```

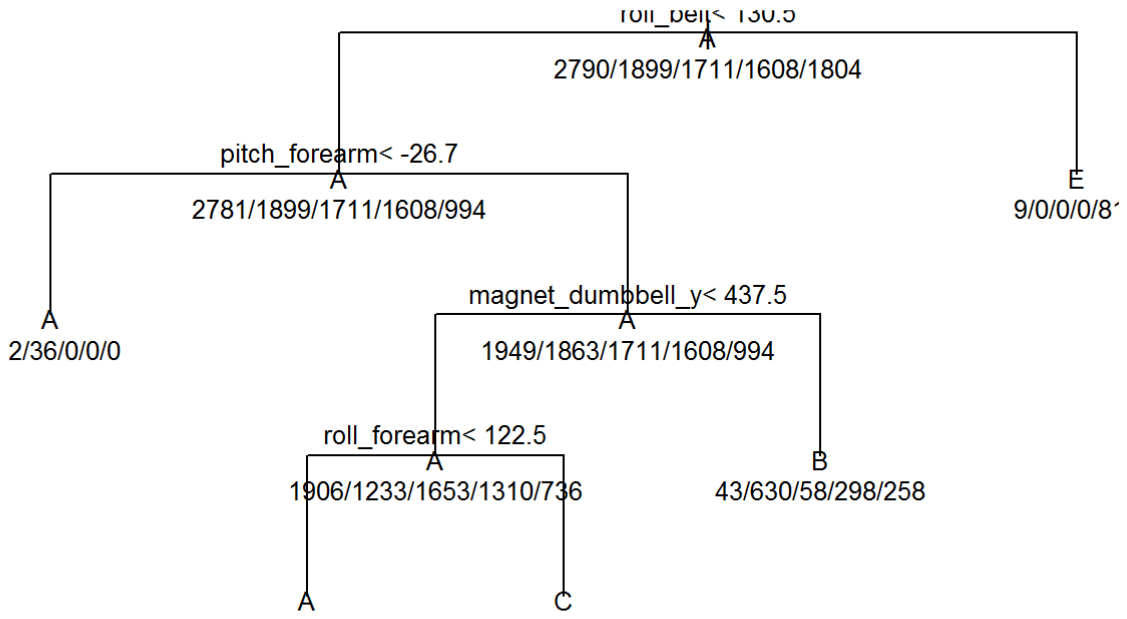
```
print(mod1$finalModel)
```

```
## n= 9812
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 9812 7022 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 8993 6212 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -26.7 868   36 A (0.96 0.041 0 0 0) *
##      5) pitch_forearm>=-26.7 8125 6176 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 437.5 6838 4932 A (0.28 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 122.5 4284 2569 A (0.4 0.18 0.19 0.17 0.063) *
##         21) roll_forearm>=122.5 2554 1709 C (0.075 0.18 0.33 0.23 0.18) *
##       11) magnet_dumbbell_y>=437.5 1287  657 B (0.033 0.49 0.045 0.23 0.2) *
##    3) roll_belt>=130.5 819    9 E (0.011 0 0 0 0.99) *
```

```
prediction <- predict(mod1,validation)
confusionMatrix(prediction,validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2545  815  777  712  251
##          B   43  646   51  280  235
##          C  197  437  883  616  496
##          D    0    0    0    0    0
##          E    5    0    0    0  821
##
## Overall Statistics
##
##                Accuracy : 0.499
##                  95% CI : (0.489, 0.5089)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3452
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9122  0.34036  0.51607   0.0000  0.45535
## Specificity            0.6360  0.92303  0.78442   1.0000  0.99938
## Pos Pred Value         0.4990  0.51474  0.33587      NaN  0.99395
## Neg Pred Value         0.9480  0.85365  0.88470   0.8361  0.89069
## Prevalence             0.2844  0.19348  0.17441   0.1639  0.18379
## Detection Rate         0.2594  0.06585  0.09001   0.0000  0.08369
## Detection Prevalence   0.5199  0.12793  0.26799   0.0000  0.08420
## Balanced Accuracy      0.7741  0.63169  0.65025   0.5000  0.72736
```

```
plot(mod1$finalModel,uniform=TRUE,main="Classification Tree")
text(mod1$finalModel,use.n = TRUE, all=TRUE, cex=.8)
```



As can be seen by the above output the accuracy of the model is low. Let is now try to fit the random forest model.

```
mod2 <- train(classe~.,"rf",data=training)
prediction2 <- predict(mod2,validation)
```

```
## Loading required package: randomForest
```

```
## Warning: package 'randomForest' was built under R version 3.2.5
```

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

```
confusionMatrix(prediction2,validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2788    7    0    0    0
##          B    0 1887    3    0    1
##          C    0    4 1704   12    3
##          D    0    0    4 1596    4
##          E    2    0    0    0 1795
##
## Overall Statistics
##
##                Accuracy : 0.9959
##                  95% CI : (0.9945, 0.9971)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9948
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9942   0.9959   0.9925   0.9956
## Specificity            0.9990   0.9995   0.9977   0.9990   0.9998
## Pos Pred Value         0.9975   0.9979   0.9890   0.9950   0.9989
## Neg Pred Value         0.9997   0.9986   0.9991   0.9985   0.9990
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1924   0.1737   0.1627   0.1830
## Detection Prevalence   0.2849   0.1928   0.1756   0.1635   0.1832
## Balanced Accuracy      0.9991   0.9968   0.9968   0.9958   0.9977
```

The model generated through random forest is very accurate but slow in generation. Now we can apply this model on the test data.

```
prediction3 <- predict(mod2,testingdata)
prediction3
```

```
##  [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
```

The output gives the prediction classes for the 20 observations from testing data set.