m8w4.rmd

raymond

June 1, 2017

Prediction Assignment Writeup

This project asks to analyze data by personal activity devices. We first load the data

```
training <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.cs
v"))
testing <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"))
```

Now, we load the necessary packages, set the seed (for it to be reproducible) and a look at the 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
 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1
323084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390
 484323 484434 ...
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 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 1 ...
##  $ num_window              : int  11 11 11 12 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 -9
4.4 ...
##  $ total_accel_belt        : int  3 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 1
 ...
##  $ kurtosis_picth_belt     : Factor w/ 317 levels "","-0.021887",..: 1 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 1 ...
##  $ skewness_roll_belt      : Factor w/ 395 levels "","-0.003095",..: 1 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 1
 ...
##  $ skewness_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 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 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
 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 -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       : Factor w/ 330 levels "","-0.02438",..: 1 1 1 1 1 1 1 1 1 1
 ...
##  $ kurtosis_picth_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_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  : Factor w/ 398 levels "","-0.0035","-0.0073",..: 1 1 1 1 1 1
 1 1 1 ...
##  $ kurtosis_picth_dumbbell : Factor w/ 401 levels "","-0.0163","-0.0233",..: 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 ...
##  $ skewness_pitch_dumbbell : Factor w/ 402 levels "","-0.0053","-0.0084",..: 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_picth_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]
```

Cleaning the data

Now, we commence the cleaning of data, which includes deleting off unrelated and useless columns (containing a lot of NAs). First off is the training dataset.

```
cols_na <- nearZeroVar(training) #cols with little/no variance
training <- training[, -cols_na]

keep_index <- !sapply(training, function(x) any(is.na(x))) #del cols containing NAs
training <- training[, keep_index]
keep_index <- sapply(colnames(training), function(x) !grepl("X|time|window",x))
# ^ remove cols with labeling functions
training <- training[, keep_index]
dim(training)
```

```
## [1] 19622    54
```

Now, we do the same filtering to the testing dataset.

```
keep_index <- !sapply(testing, function(x) any(is.na(x)))
testing <- testing[, keep_index]
keep_index <- sapply(colnames(testing), function(x) !grepl("X|time|window",x))
testing <- testing[, keep_index];
#remove problem_id col
idx1 <- which(colnames(testing)=="problem_id")
testing <- testing[,-idx1]
dim(testing)
```

```
## [1] 20 53
```

Machine Learning

Now, we splice the training dataset so we have a ‘train’ and ‘test’ data from the training dataset

```
index_train <- createDataPartition(training$classe, p = 0.7, list=FALSE)
training1 <- training[index_train, ]
testing1 <- training[-index_train, ]
```

Side note:-

```
control <- trainControl(method = "cv", number = 5)
```

We set the number of cross validation to 5 instead of the default 10 to save computation time. Also my computer ran out of memory with the default 10.

LDA

First, we try Linear Discriminant Analysis (LDA)

```
modFit_lda <- train(classe ~., data=training1, method="lda")
print(modFit_lda, digits = 4)
```

```
## Linear Discriminant Analysis
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.7355    0.6648
```

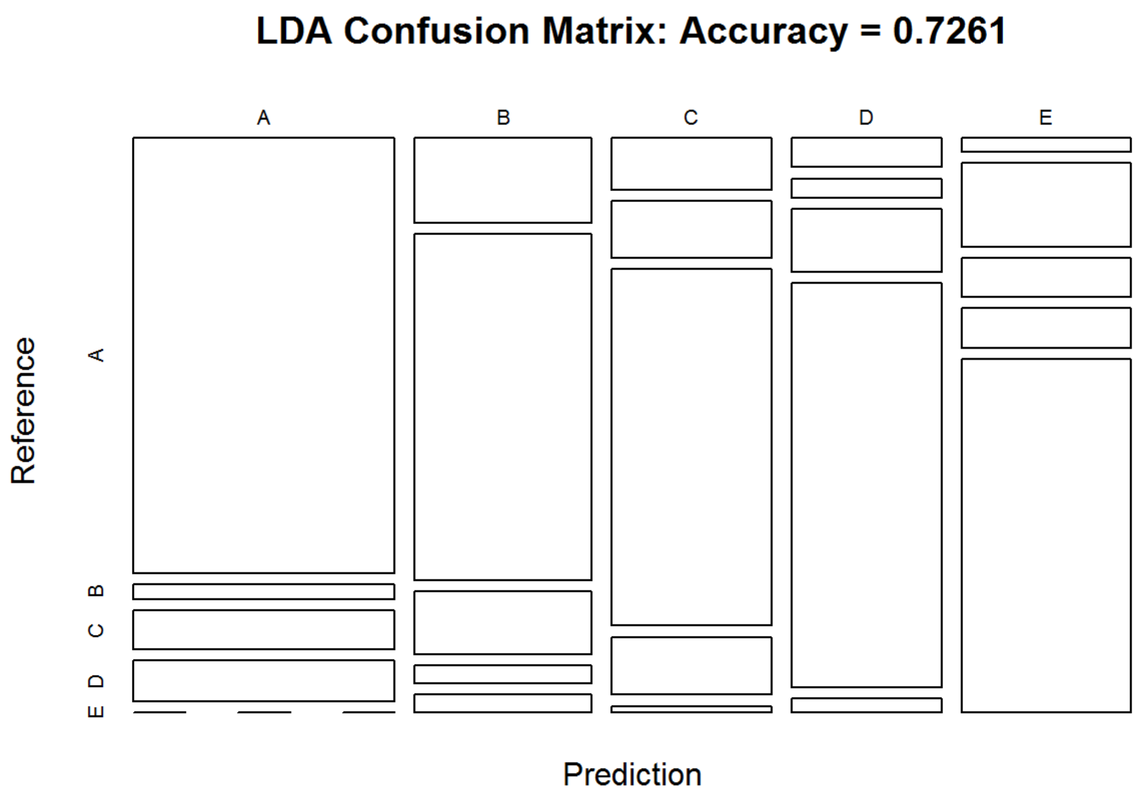
```
predict_lda <- predict(modFit_lda, testing1)
(conf_lda <- confusionMatrix(testing1$classe, predict_lda))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1377   46  123  128    0
##          B  182  745  135   38   39
##          C  100  110  692  111   13
##          D   53   35  114  736   26
##          E   28  171   79   81  723
##
## Overall Statistics
##
##                Accuracy : 0.7261
##                  95% CI : (0.7145, 0.7374)
##     No Information Rate : 0.2957
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6533
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7914   0.6730   0.6054   0.6728   0.9026
## Specificity            0.9283   0.9175   0.9296   0.9524   0.9294
## Pos Pred Value         0.8226   0.6541   0.6745   0.7635   0.6682
## Neg Pred Value         0.9138   0.9237   0.9072   0.9273   0.9838
## Prevalence             0.2957   0.1881   0.1942   0.1859   0.1361
## Detection Rate         0.2340   0.1266   0.1176   0.1251   0.1229
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.8599   0.7953   0.7675   0.8126   0.9160
```

```
(accuracy_lda <- conf_lda$overall[1])
```

```
##  Accuracy
## 0.7260833
```

```
plot(conf_lda$table, col = conf_lda$byClass, main = paste("LDA Confusion Matrix: Accuracy =",
 round(conf_lda$overall['Accuracy'], 4)))
```



Classification Tree

Next, we try the Classification Tree method (rpart)

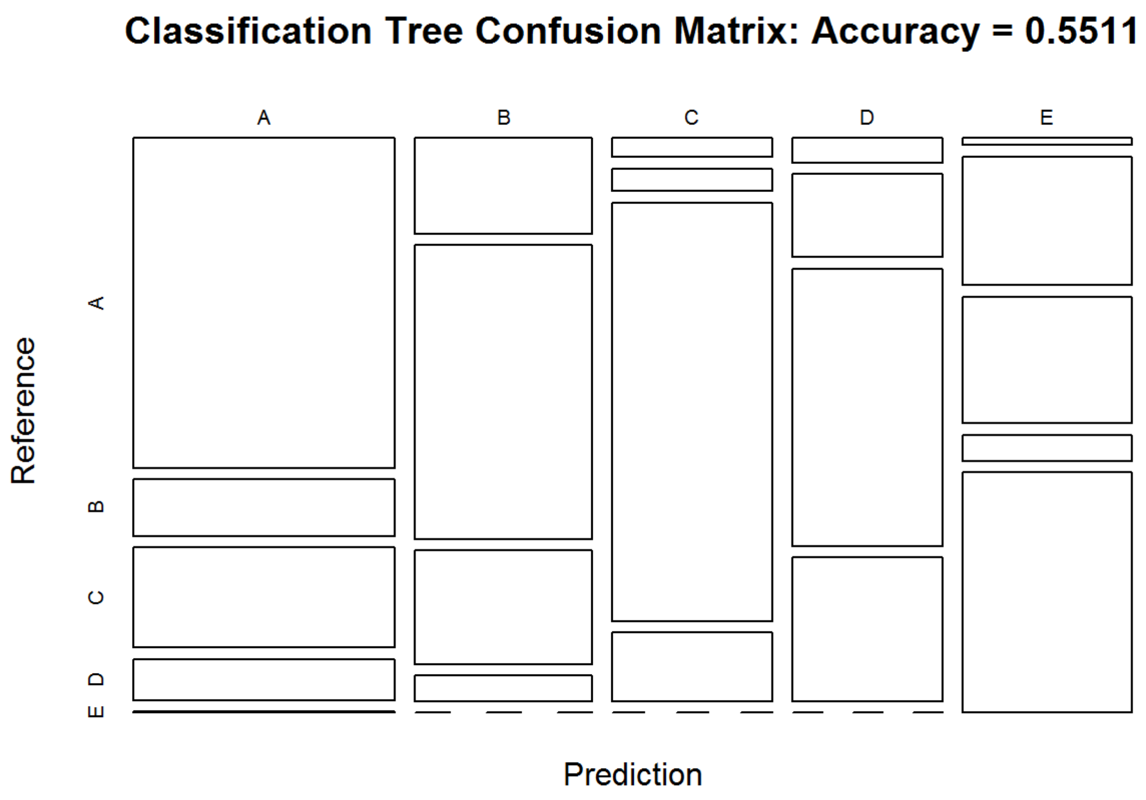
```
modFit_rpart <- train(classe ~ ., data = training1, method = "rpart",
                   trControl = control)
print(modFit_rpart, digits = 4)
```

```
## CART
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10991, 10991, 10989, 10988, 10989
## Resampling results across tuning parameters:
##
##   cp       Accuracy  Kappa
##   0.02777  0.5754    0.4576
##   0.04315  0.4959    0.3406
##   0.11474  0.3316    0.0721
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.02777.
```

```
predict_rpart <- predict(modFit_rpart, testing1)
(conf_rpart <- confusionMatrix(testing1$classe, predict_rpart))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1044  179  318  130    3
##          B  206  633  245   55    0
##          C   37   44  812  133    0
##          D   45  151  506  262    0
##          E   14  264  259   53  492
## 
## Overall Statistics
## 
##                Accuracy : 0.5511
##                  95% CI : (0.5382, 0.5638)
##     No Information Rate : 0.3636
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.4365
##  Mcnemar's Test P-Value : < 2.2e-16
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7756   0.4980   0.3794  0.41390  0.99394
## Specificity            0.8612   0.8903   0.9429  0.86634  0.89054
## Pos Pred Value         0.6237   0.5558   0.7914  0.27178  0.45471
## Neg Pred Value         0.9283   0.8656   0.7267  0.92461  0.99938
## Prevalence             0.2287   0.2160   0.3636  0.10756  0.08411
## Detection Rate         0.1774   0.1076   0.1380  0.04452  0.08360
## Detection Prevalence   0.2845   0.1935   0.1743  0.16381  0.18386
## Balanced Accuracy      0.8184   0.6942   0.6611  0.64012  0.94224
```

```
plot(conf_rpart$table, col = conf_rpart$byClass, main = paste("Classification Tree Confusion
 Matrix: Accuracy =", round(conf_rpart$overall['Accuracy'], 4)))
```



Random Forest

Lastly, we try random forest

```
modFit_rf <- train(classe ~., data = training1, method = "rf", trControl=control )
print(modFit_rf, digits = 4)
```

```
## Random Forest
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10990, 10990, 10988, 10990
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    0.9895    0.9867
##   29    0.9908    0.9884
##   57    0.9877    0.9844
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 29.
```

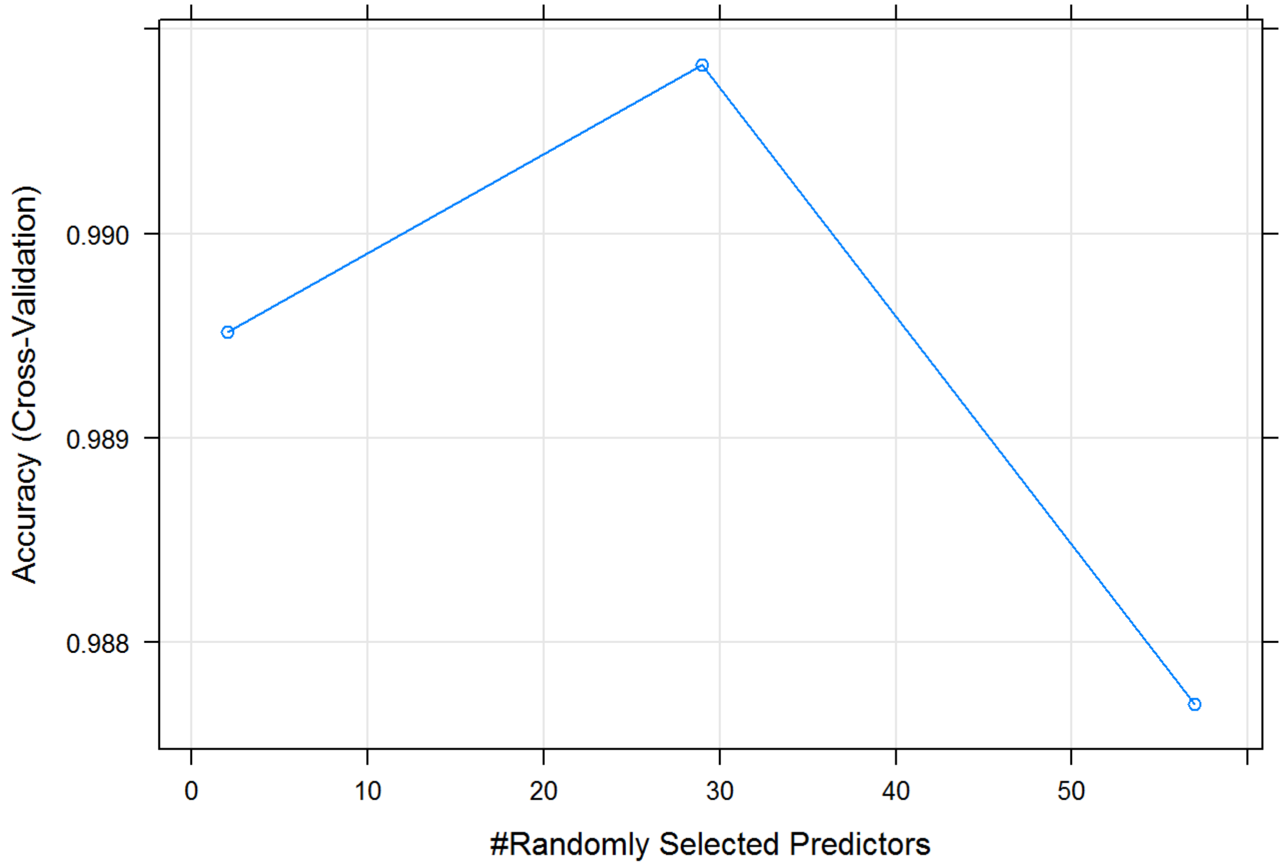
```
# predict outcomes using validation set
predict_rf <- predict(modFit_rf, testing1)
# Show prediction result
(conf_rf <- confusionMatrix(testing1$classe, predict_rf))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B   10 1127    2    0    0
##          C    0    6 1016    4    0
##          D    0    0   11  953    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9944
##                  95% CI : (0.9921, 0.9961)
##     No Information Rate : 0.2862
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9929
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9941   0.9947   0.9874   0.9958   1.0000
## Specificity            1.0000   0.9975   0.9979   0.9978   1.0000
## Pos Pred Value         1.0000   0.9895   0.9903   0.9886   1.0000
## Neg Pred Value         0.9976   0.9987   0.9973   0.9992   1.0000
## Prevalence             0.2862   0.1925   0.1749   0.1626   0.1839
## Detection Rate         0.2845   0.1915   0.1726   0.1619   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9970   0.9961   0.9927   0.9968   1.0000
```

```
(accuracy_rf <- conf_rf$overall[1])
```

```
##  Accuracy
## 0.9943925
```

```
plot(modFit_rf)
```



So, from the three models (LDA, Classification Tree, Random Forest) The accuracies are as follow

LDA: 72.6%

Classification Tree: 55%

Random Forest: 99%

As you can see, random forest so far has the best accuracy. The prediction of classe on testing dataset as follow

```
(predict(modFit_rf, testing))
```

```
##  [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
```