Machine Learning assignment

JayR

20/06/2020

Synopsis

The aim of this project is to create a model that uses data to predict outcomes. The data being analysed is from accelerometers on the belt, forearm, arm and dumbell of 6 paritipants. The participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

Load required libraries

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------------------------------- t
idyverse 1.2.1 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts -------------------------------------------------------------------------------- tidyver
se_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(skimr)
```

```
## Warning: package 'skimr' was built under R version 3.6.3
```

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.6.3
```

```
## 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.
```

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.6.3
```

```
## Loaded gbm 2.1.5
```

Data analysis

Import data

```
training <- read_csv("./data/pml-training.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   user_name = col_character(),
##   cvtd_timestamp = col_character(),
##   new_window = col_character(),
##   kurtosis_roll_belt = col_character(),
##   kurtosis_picth_belt = col_character(),
##   kurtosis_yaw_belt = col_character(),
##   skewness_roll_belt = col_character(),
##   skewness_roll_belt.1 = col_character(),
##   skewness_yaw_belt = col_character(),
##   max_yaw_belt = col_character(),
##   min_yaw_belt = col_character(),
##   amplitude_yaw_belt = col_character(),
##   kurtosis_picth_arm = col_character(),
##   kurtosis_yaw_arm = col_character(),
##   skewness_pitch_arm = col_character(),
##   skewness_yaw_arm = col_character(),
##   kurtosis_yaw_dumbbell = col_character(),
##   skewness_yaw_dumbbell = col_character(),
##   kurtosis_roll_forearm = col_character(),
##   kurtosis_picth_forearm = col_character()
##   # ... with 8 more columns
## )
```

```
## See spec(...) for full column specifications.
```

```
## Warning: 182 parsing failures.
##  row               col expected  actual                       file
## 2231 kurtosis_roll_arm a double #DIV/0! './data/pml-training.csv'
## 2231 skewness_roll_arm a double #DIV/0! './data/pml-training.csv'
## 2255 kurtosis_roll_arm a double #DIV/0! './data/pml-training.csv'
## 2255 skewness_roll_arm a double #DIV/0! './data/pml-training.csv'
## 2282 kurtosis_roll_arm a double #DIV/0! './data/pml-training.csv'
## .... ................. ........ ....... ........................
## See problems(...) for more details.
```

```
testing <- read_csv("./data/pml-testing.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   .default = col_logical(),
##   X1 = col_double(),
##   user_name = col_character(),
##   raw_timestamp_part_1 = col_double(),
##   raw_timestamp_part_2 = col_double(),
##   cvtd_timestamp = col_character(),
##   new_window = col_character(),
##   num_window = col_double(),
##   roll_belt = col_double(),
##   pitch_belt = col_double(),
##   yaw_belt = col_double(),
##   total_accel_belt = col_double(),
##   gyros_belt_x = col_double(),
##   gyros_belt_y = col_double(),
##   gyros_belt_z = col_double(),
##   accel_belt_x = col_double(),
##   accel_belt_y = col_double(),
##   accel_belt_z = col_double(),
##   magnet_belt_x = col_double(),
##   magnet_belt_y = col_double(),
##   magnet_belt_z = col_double()
##   # ... with 40 more columns
## )
## See spec(...) for full column specifications.
```

```
dim(training)
```

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

```
dim(testing)
```

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

View data

```
head(training)
```

```
## # A tibble: 6 x 160
##      X1 user_name raw_timestamp_p~ raw_timestamp_p~ cvtd_timestamp
##   <dbl> <chr>                <dbl>            <dbl> <chr>
## 1     1 carlitos        1323084231           788290 05/12/2011 11~
## 2     2 carlitos        1323084231           808298 05/12/2011 11~
## 3     3 carlitos        1323084231           820366 05/12/2011 11~
## 4     4 carlitos        1323084232           120339 05/12/2011 11~
## 5     5 carlitos        1323084232           196328 05/12/2011 11~
## 6     6 carlitos        1323084232           304277 05/12/2011 11~
## # ... with 155 more variables: new_window <chr>, num_window <dbl>,
## #   roll_belt <dbl>, pitch_belt <dbl>, yaw_belt <dbl>,
## #   total_accel_belt <dbl>, kurtosis_roll_belt <chr>,
## #   kurtosis_picth_belt <chr>, kurtosis_yaw_belt <chr>,
## #   skewness_roll_belt <chr>, skewness_roll_belt.1 <chr>,
## #   skewness_yaw_belt <chr>, max_roll_belt <dbl>, max_picth_belt <dbl>,
## #   max_yaw_belt <chr>, min_roll_belt <dbl>, min_pitch_belt <dbl>,
## #   min_yaw_belt <chr>, amplitude_roll_belt <dbl>,
## #   amplitude_pitch_belt <dbl>, amplitude_yaw_belt <chr>,
## #   var_total_accel_belt <dbl>, avg_roll_belt <dbl>,
## #   stddev_roll_belt <dbl>, var_roll_belt <dbl>, avg_pitch_belt <dbl>,
## #   stddev_pitch_belt <dbl>, var_pitch_belt <dbl>, avg_yaw_belt <dbl>,
## #   stddev_yaw_belt <dbl>, var_yaw_belt <dbl>, gyros_belt_x <dbl>,
## #   gyros_belt_y <dbl>, gyros_belt_z <dbl>, accel_belt_x <dbl>,
## #   accel_belt_y <dbl>, accel_belt_z <dbl>, magnet_belt_x <dbl>,
## #   magnet_belt_y <dbl>, magnet_belt_z <dbl>, roll_arm <dbl>,
## #   pitch_arm <dbl>, yaw_arm <dbl>, total_accel_arm <dbl>,
## #   var_accel_arm <dbl>, avg_roll_arm <dbl>, stddev_roll_arm <dbl>,
## #   var_roll_arm <dbl>, avg_pitch_arm <dbl>, stddev_pitch_arm <dbl>,
## #   var_pitch_arm <dbl>, avg_yaw_arm <dbl>, stddev_yaw_arm <dbl>,
## #   var_yaw_arm <dbl>, gyros_arm_x <dbl>, gyros_arm_y <dbl>,
## #   gyros_arm_z <dbl>, accel_arm_x <dbl>, accel_arm_y <dbl>,
## #   accel_arm_z <dbl>, magnet_arm_x <dbl>, magnet_arm_y <dbl>,
## #   magnet_arm_z <dbl>, kurtosis_roll_arm <dbl>, kurtosis_picth_arm <chr>,
## #   kurtosis_yaw_arm <chr>, skewness_roll_arm <dbl>,
## #   skewness_pitch_arm <chr>, skewness_yaw_arm <chr>, max_roll_arm <dbl>,
## #   max_picth_arm <dbl>, max_yaw_arm <dbl>, min_roll_arm <dbl>,
## #   min_pitch_arm <dbl>, min_yaw_arm <dbl>, amplitude_roll_arm <dbl>,
## #   amplitude_pitch_arm <dbl>, amplitude_yaw_arm <dbl>,
## #   roll_dumbbell <dbl>, pitch_dumbbell <dbl>, yaw_dumbbell <dbl>,
## #   kurtosis_roll_dumbbell <dbl>, kurtosis_picth_dumbbell <dbl>,
## #   kurtosis_yaw_dumbbell <chr>, skewness_roll_dumbbell <dbl>,
## #   skewness_pitch_dumbbell <dbl>, skewness_yaw_dumbbell <chr>,
## #   max_roll_dumbbell <dbl>, max_picth_dumbbell <dbl>,
## #   max_yaw_dumbbell <dbl>, min_roll_dumbbell <dbl>,
## #   min_pitch_dumbbell <dbl>, min_yaw_dumbbell <dbl>,
## #   amplitude_roll_dumbbell <dbl>, amplitude_pitch_dumbbell <dbl>,
## #   amplitude_yaw_dumbbell <dbl>, total_accel_dumbbell <dbl>,
## #   var_accel_dumbbell <dbl>, avg_roll_dumbbell <dbl>,
## #   stddev_roll_dumbbell <dbl>, ...
```

Look at how complete the columns are

```
skim(training)
```

There are several columns that are less than 3% complete - remove these columns from the training and testing datasets. Also remove any non-numeric variables (except `classe`)

```
training_ed <- training %>%
  select_if(~ !any(is.na(.))) %>%
  select(-c(X1:num_window))

testing_ed <- testing %>%
  select_if(~ !any(is.na(.))) %>%
  select(-c(X1:num_window))
```

Set `classe` as a factor

```
training_ed$classe <- as.factor(training_ed$classe)
```

Data partitioning

Split the training data into training and testing datasets

```
set.seed(100)
inTrain <- createDataPartition(training_ed$classe, p = 0.6, list = FALSE)

traindata <- training_ed[inTrain,]
testdata <- training_ed[-inTrain,]
dim(traindata)
```

```
## [1] 11776    53
```

```
dim(testdata)
```

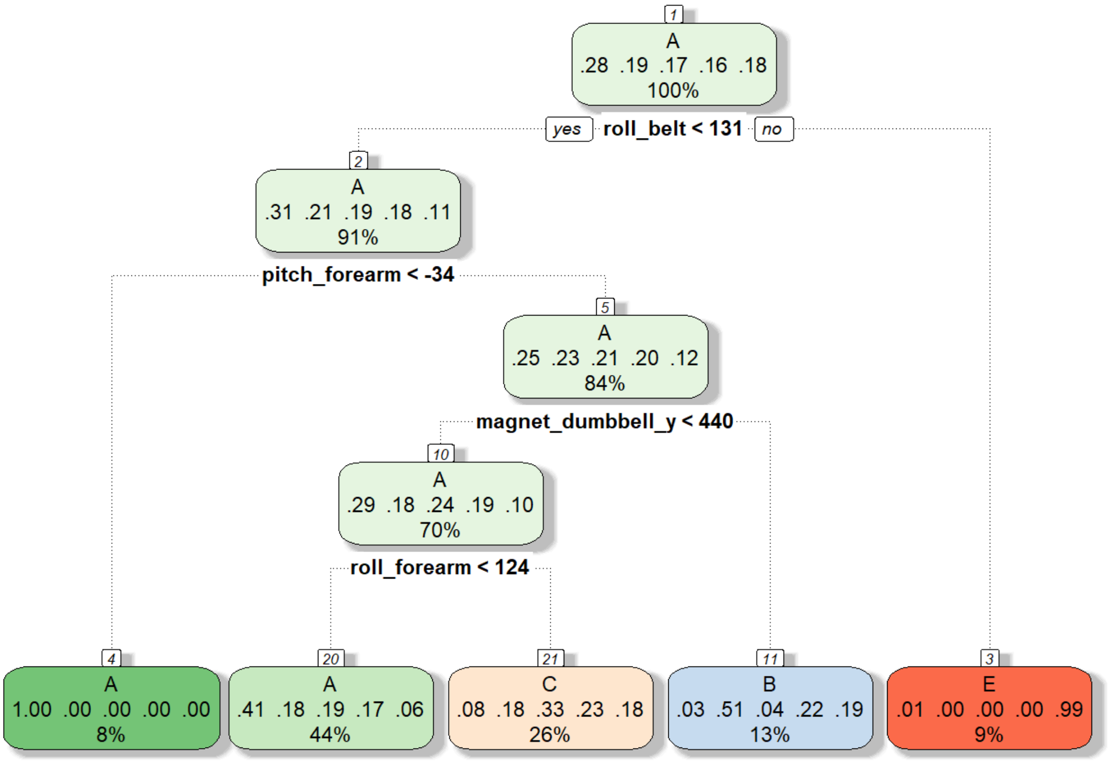
```
## [1] 7846   53
```

Use cross validation to construct models

Decision Tree Model

```
modFitDecT <- train(classe ~., data = traindata, method = "rpart")
```

```
fancyRpartPlot(modFitDecT$finalModel)
```



Rattle 2020-Jun-21 01:17:14 jwatk

```
predDecT <- predict(modFitDecT, testdata)
confDecT <- confusionMatrix(predDecT, testdata$classe)
confDecT
```

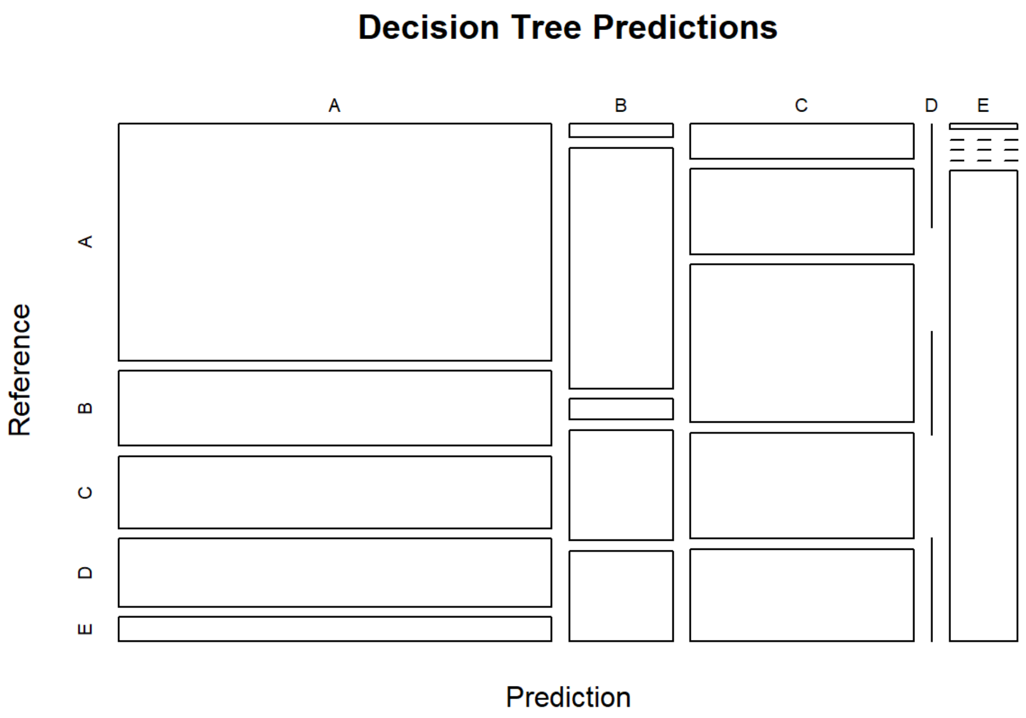
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2042  646  624  587  212
##          B   28  494   42  227  186
##          C  155  378  702  472  410
##          D    0    0    0    0    0
##          E    7    0    0    0  634
##
## Overall Statistics
##
##                Accuracy : 0.4935
##                  95% CI : (0.4824, 0.5046)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3377
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9149  0.32543  0.51316   0.0000  0.43967
## Specificity            0.6315  0.92367  0.78157   1.0000  0.99891
## Pos Pred Value         0.4967  0.50563  0.33160      NaN  0.98908
## Neg Pred Value         0.9491  0.85092  0.88375   0.8361  0.88786
## Prevalence             0.2845  0.19347  0.17436   0.1639  0.18379
## Detection Rate         0.2603  0.06296  0.08947   0.0000  0.08081
## Detection Prevalence   0.5240  0.12452  0.26982   0.0000  0.08170
## Balanced Accuracy      0.7732  0.62455  0.64736   0.5000  0.71929
```

The Decision Tree model predicts with accuracy of 0.4934999.

```
table(predDecT, testdata$classe)
```

```
##
## predDecT    A    B    C    D    E
##        A 2042  646  624  587  212
##        B   28  494   42  227  186
##        C  155  378  702  472  410
##        D    0    0    0    0    0
##        E    7    0    0    0  634
```

```
plot(confDecT$table, col = confDecT$byClass, main = "Decision Tree Predictions")
```



Gradient Boosting Model

```
set.seed(25621)
modFitGBM <- train(classe ~., data = traindata, method = "gbm", verbose = FALSE)
```

```
predGBM <- predict(modFitGBM, testdata)
confGBM <- confusionMatrix(predGBM, testdata$classe)
confGBM
```

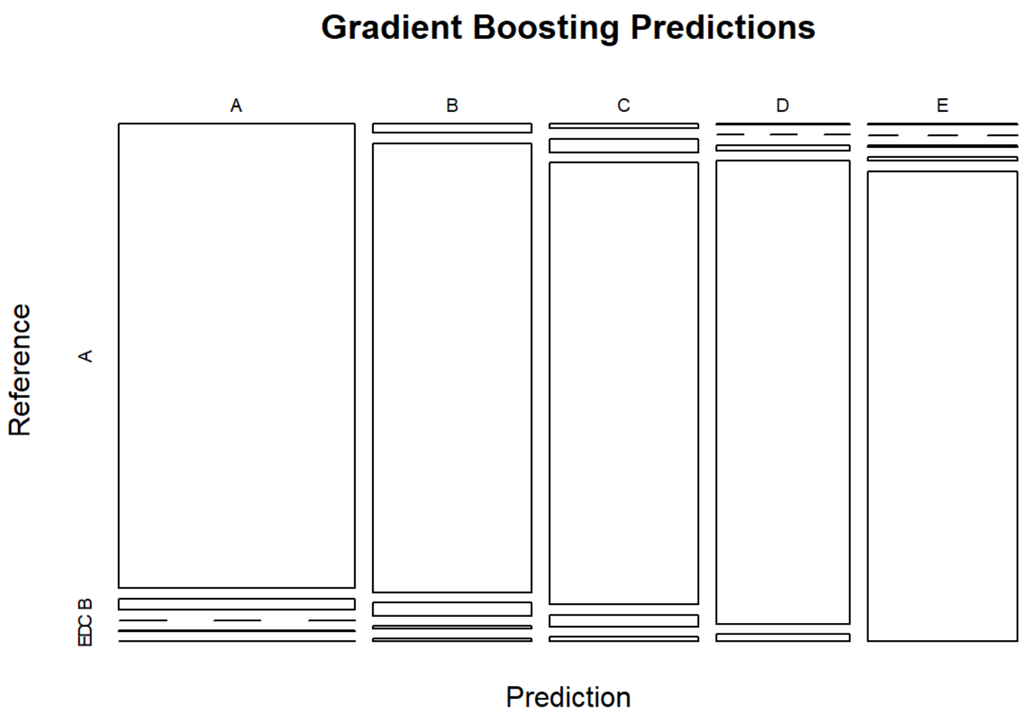
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2188   54    0    1    1
##          B   28 1423   41    7    9
##          C   13   41 1308   34   14
##          D    1    0   15 1233   19
##          E    2    0    4   11 1399
##
## Overall Statistics
##
##                Accuracy : 0.9624
##                  95% CI : (0.958, 0.9665)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9524
##
##  Mcnemar's Test P-Value : 8.709e-08
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9803   0.9374   0.9561   0.9588   0.9702
## Specificity            0.9900   0.9866   0.9843   0.9947   0.9973
## Pos Pred Value         0.9750   0.9436   0.9277   0.9724   0.9880
## Neg Pred Value         0.9921   0.9850   0.9907   0.9919   0.9933
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2789   0.1814   0.1667   0.1572   0.1783
## Detection Prevalence   0.2860   0.1922   0.1797   0.1616   0.1805
## Balanced Accuracy      0.9852   0.9620   0.9702   0.9767   0.9838
```

The Gradient Boosting model predicts with accuracy of 0.9624012.

```
table(predGBM, testdata$classe)
```

```
##
## predGBM    A    B    C    D    E
##       A 2188   54    0    1    1
##       B   28 1423   41    7    9
##       C   13   41 1308   34   14
##       D    1    0   15 1233   19
##       E    2    0    4   11 1399
```

```
plot(confGBM$table, col = confGBM$byClass, main = "Gradient Boosting Predictions")
```



Linear Discriminant Analysis model

```
modFitLDA <- train(classe ~., data = traindata, method = "lda")
```

```
predLDA <- predict(modFitLDA, testdata)
confLDA <- confusionMatrix(predLDA, testdata$classe)
confLDA
```

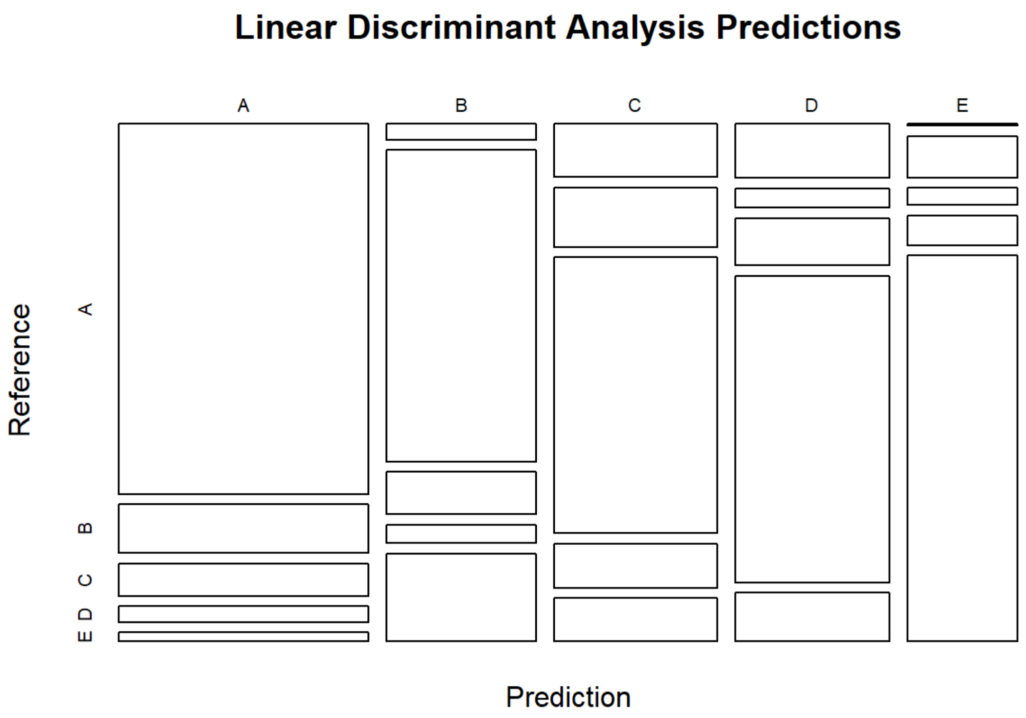
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1842  243  161   79   44
##          B   47  930  127   55  263
##          C  173  194  897  144  140
##          D  166   60  146  942  150
##          E    4   91   37   66  845
##
## Overall Statistics
##
##                Accuracy : 0.6954
##                  95% CI : (0.6851, 0.7056)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6142
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8253   0.6126   0.6557   0.7325   0.5860
## Specificity            0.9061   0.9223   0.8995   0.9204   0.9691
## Pos Pred Value         0.7775   0.6540   0.5795   0.6434   0.8102
## Neg Pred Value         0.9288   0.9085   0.9252   0.9461   0.9122
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2348   0.1185   0.1143   0.1201   0.1077
## Detection Prevalence   0.3019   0.1812   0.1973   0.1866   0.1329
## Balanced Accuracy      0.8657   0.7674   0.7776   0.8265   0.7775
```

The Linear Discriminant Analysis model predicts with accuracy of 0.6953862.

```
table(predLDA, testdata$classe)
```

```
##
## predLDA    A    B    C    D    E
##       A 1842  243  161   79   44
##       B   47  930  127   55  263
##       C  173  194  897  144  140
##       D  166   60  146  942  150
##       E    4   91   37   66  845
```

```
plot(confLDA$table, col = confLDA$byClass, main = "Linear Discriminant Analysis Predictions")
```



Random Forest Model

```
modFitRF <- train(classe ~., data = traindata, method = "rf", ntree = 100)
```

```
predRF <- predict(modFitRF, testdata)
confRF <- confusionMatrix(predRF, testdata$classe)
confRF
```

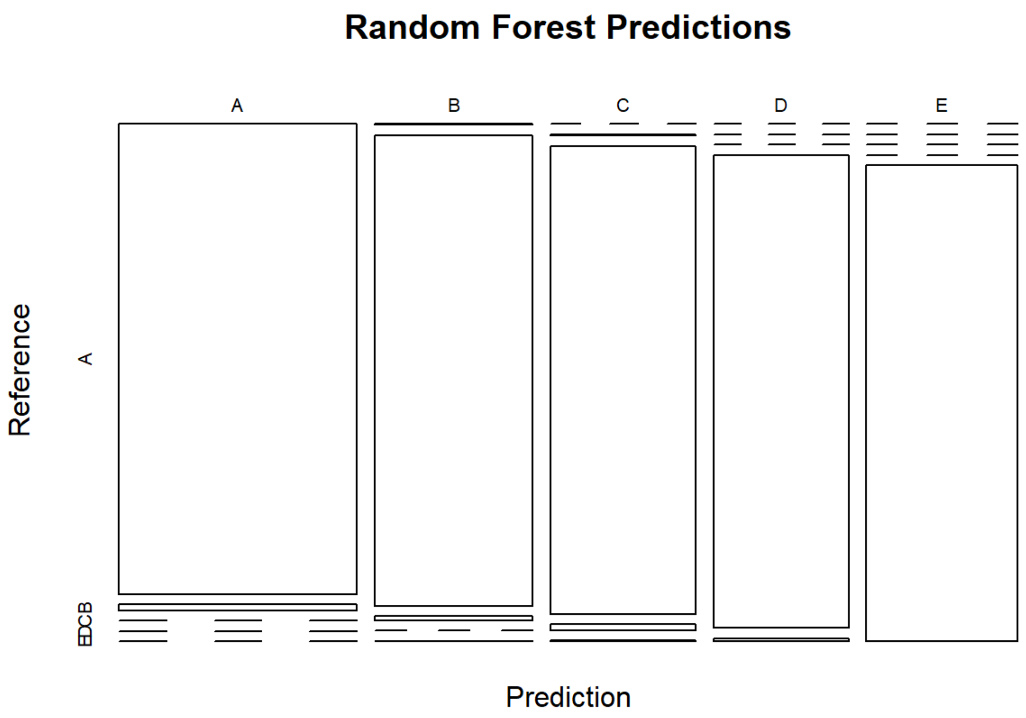
```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2229   28    0    0    0
##          B    3 1486   14    0    1
##          C    0    4 1354   17    3
##          D    0    0    0 1269    8
##          E    0    0    0    0 1430
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9901          
##                  95% CI : (0.9876, 0.9921)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9874          
##                                           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9789   0.9898   0.9868   0.9917
## Specificity            0.9950   0.9972   0.9963   0.9988   1.0000
## Pos Pred Value         0.9876   0.9880   0.9826   0.9937   1.0000
## Neg Pred Value         0.9995   0.9950   0.9978   0.9974   0.9981
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2841   0.1894   0.1726   0.1617   0.1823
## Detection Prevalence   0.2877   0.1917   0.1756   0.1628   0.1823
## Balanced Accuracy      0.9968   0.9880   0.9930   0.9928   0.9958
```

The Random Forest model predicts with accuracy of 0.9900586.

```
table(predRF, testdata$classe)
```

```
## 
## predRF    A    B    C    D    E
##      A 2229   28    0    0    0
##      B    3 1486   14    0    1
##      C    0    4 1354   17    3
##      D    0    0    0 1269    8
##      E    0    0    0    0 1430
```

```
plot(confRF$table, col = confRF$byClass, main = "Random Forest Predictions")
```



Final model chosen

The Random Forest model has produced the highest accuracy so is likely to be the best model to chose.

Final prediction

```
Final_pred <- predict(modFitRF, testing_ed)
Final_pred
```

```
##  [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
```