Random Forest

Krissie

2024-11-27

R Markdown

```
#1. Loading data
  setwd("/Users/kitsanasudsaneh/Desktop/fall2024/MAX503/final Project")
  wine.raw <- read.csv("vivino.csv")

#2.Inspect the dataset

#---Dimensions of the dataset
  n_rows <- nrow (wine.raw) #Number of observations (rows)
  n_columns <- ncol (wine.raw) #Number of variables (columns)

  cat("The dataset contains: \n")
```

```
## The dataset contains:
```

```
  cat(n_rows,"Oberservation (rows)\n")
```

```
## 1599 Oberservation (rows)
```

```
  cat(n_columns,"Variables (column)\n")
```

```
## 12 Variables (column)
```

```
#---Sumarize the dataset
  summary.wine <- summary(wine.raw)
  cat("Summary of the dataset: \n")
```

```
## Summary of the dataset:
```

```
  summary.wine
```

```
##  fixed.acidity   volatile.acidity  citric.acid    residual.sugar
##  Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900
##  1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900
##  Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200
##  Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539
```

```
##  3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600
##  Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.01200   Min.   : 1.00       Min.   :  6.00       Min.   :0.9901
##  1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956
##  Median :0.07900   Median :14.00       Median : 38.00       Median :0.9968
##  Mean   :0.08747   Mean   :15.87       Mean   : 46.47       Mean   :0.9967
##  3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978
##  Max.   :0.61100   Max.   :72.00       Max.   :289.00       Max.   :1.0037
##        pH          sulphates         alcohol         quality
##  Min.   :2.740   Min.   :0.3300   Min.   : 8.40   Min.   :3.000
##  1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000
##  Median :3.310   Median :0.6200   Median :10.20   Median :6.000
##  Mean   :3.311   Mean   :0.6581   Mean   :10.42   Mean   :5.636
##  3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10   3rd Qu.:6.000
##  Max.   :4.010   Max.   :2.0000   Max.   :14.90   Max.   :8.000
```

```
#---Structure of the dataset
  cat("Structure of the dataset: \n")
```

```
## Structure of the dataset:
```

```
  str(wine.raw)
```

```
## 'data.frame':    1599 obs. of  12 variables:
##  $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
##  $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
##  $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
##  $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
##  $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
##  $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
##  $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
##  $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
##  $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
##  $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
##  $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
##  $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
```

```
#---Correlation Plot
  wine.df <- wine.raw[,-12]
  #Exclude 'quality': group the data under the quality column

#compute correlation matrix:
  library(corrplot)
```

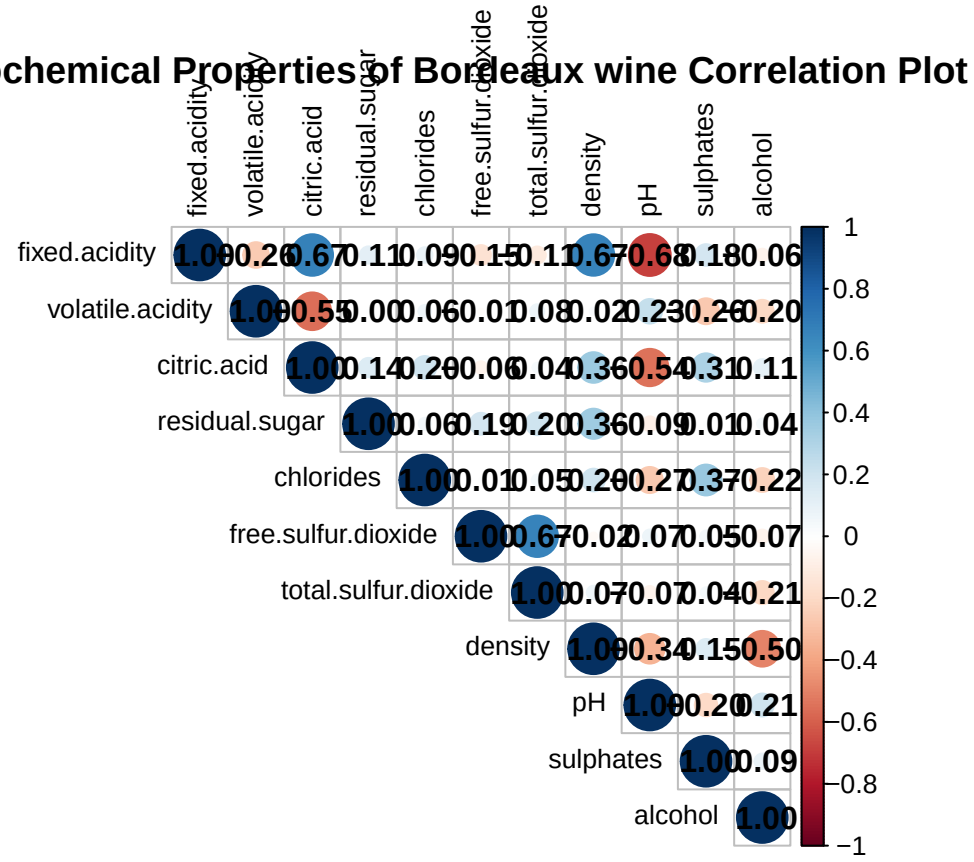
```
## Warning: package 'corrplot' was built under R version 4.3.3
```

```
## corrplot 0.95 loaded
```

```
  corr_matrix <-cor(wine.df, use = "complete.obs")
  corr_matrix
```

```
##                       fixed.acidity volatile.acidity citric.acid residual.sugar
## fixed.acidity           1.00000000     -0.256130895  0.67170343    0.114776724
## volatile.acidity       -0.25613089      1.000000000 -0.55249568    0.001917882
## citric.acid             0.67170343     -0.552495685  1.00000000    0.143577162
## residual.sugar          0.11477672      0.001917882  0.14357716    1.000000000
## chlorides               0.09370519      0.061297772  0.20382291    0.055609535
## free.sulfur.dioxide    -0.15379419     -0.010503827 -0.06097813    0.187048995
## total.sulfur.dioxide   -0.11318144      0.076470005  0.03553302    0.203027882
## density                 0.66804729      0.022026232  0.36494718    0.355283371
## pH                     -0.68297819      0.234937294 -0.54190414   -0.085652422
## sulphates               0.18300566     -0.260986685  0.31277004    0.005527121
## alcohol                -0.06166827     -0.202288027  0.10990325    0.042075437
##                         chlorides free.sulfur.dioxide total.sulfur.dioxide
## fixed.acidity         0.093705186        -0.153794193          -0.11318144
## volatile.acidity      0.061297772        -0.010503827           0.07647000
## citric.acid           0.203822914        -0.060978129           0.03553302
## residual.sugar        0.055609535         0.187048995           0.20302788
## chlorides             1.000000000         0.005562147           0.04740047
## free.sulfur.dioxide   0.005562147         1.000000000           0.66766645
## total.sulfur.dioxide  0.047400468         0.667666450           1.00000000
## density               0.200632327        -0.021945831           0.07126948
## pH                   -0.265026131         0.070377499          -0.06649456
## sulphates             0.371260481         0.051657572           0.04294684
## alcohol              -0.221140545        -0.069408354          -0.20565394
##                          density          pH    sulphates     alcohol
## fixed.acidity         0.66804729 -0.68297819  0.183005664 -0.06166827
## volatile.acidity      0.02202623  0.23493729 -0.260986685 -0.20228803
## citric.acid           0.36494718 -0.54190414  0.312770044  0.10990325
## residual.sugar        0.35528337 -0.08565242  0.005527121  0.04207544
## chlorides             0.20063233 -0.26502613  0.371260481 -0.22114054
## free.sulfur.dioxide  -0.02194583  0.07037750  0.051657572 -0.06940835
## total.sulfur.dioxide  0.07126948 -0.06649456  0.042946836 -0.20565394
## density               1.00000000 -0.34169933  0.148506412 -0.49617977
## pH                   -0.34169933  1.00000000 -0.196647602  0.20563251
## sulphates             0.14850641 -0.19664760  1.000000000  0.09359475
## alcohol              -0.49617977  0.20563251  0.093594750  1.00000000
```

```r
#plot the correltion:
  corrplot(corr_matrix, method = "circle", type = "upper",
      tl.cex = 0.8, tl.col = "black", addCoef.col = "black",
      main = "\n\n\nPhysicochemical Properties of Bordeaux wine Correlation Plot")
```




```
#3.Cleaning and Preparing the data

#---Create a Naw Column with two groups(bad and good)
  wine.raw$quality.label <-ifelse(wine.raw$quality > 6, "Good","Bad")
  print(head(wine.raw$quality.label))
```

```
## [1] "Bad" "Bad" "Bad" "Bad" "Bad" "Bad"
```

```
  table(wine.raw$quality.label)
```

```
##
##  Bad Good
## 1382  217
```

```
  head(wine.raw)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4             0.70        0.00            1.9     0.076
## 2           7.8             0.88        0.00            2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4             0.70        0.00            1.9     0.076
## 6           7.4             0.66        0.00            1.8     0.075
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
```

```
## 1                  11                   34  0.9978 3.51      0.56     9.4
## 2                  25                   67  0.9968 3.20      0.68     9.8
## 3                  15                   54  0.9970 3.26      0.65     9.8
## 4                  17                   60  0.9980 3.16      0.58     9.8
## 5                  11                   34  0.9978 3.51      0.56     9.4
## 6                  13                   40  0.9978 3.51      0.56     9.4
##   quality quality.label
## 1       5           Bad
## 2       5           Bad
## 3       5           Bad
## 4       6           Bad
## 5       5           Bad
## 6       5           Bad
```

```r
#---Splitting Data into Training and Testing Sets
  set.seed(04625)
  train.prop <- 0.70 #70% training, 30% testing
  train.cases <- sample(nrow(wine.raw), nrow(wine.raw) * train.prop)
  train.cases
```

```
##    [1]  360  429  591  831 1108  350 1138 1127 1477   31  869  529  842  327
##   [15]  239  678 1497 1057  230   87  608  399  977  232  286 1136  471  486
##   [29]  705 1455  473  154 1088   57 1140  619  806 1052  582 1114  930 1401
##   [43] 1560  878  982 1448  249 1244  504  521 1080  148 1333 1503  382  994
##   [57]  534 1226 1087  646  255 1469   70  309  159 1273  737  256 1089  143
##   [71]  246  225  207 1517  361 1433 1579   43  613 1451  400  433  901 1397
##   [85]  459  729  155  233   85  115 1308  579  338  880  170 1188 1212  689
##   [99]  622 1202 1269   45  570  750  587  557  601  250 1093 1494 1305 1432
##  [113]  947 1518  541   27 1042  709  223   60  710 1192  369 1468  291 1535
##  [127] 1291    1  510 1032 1169 1153  183  563 1224 1383  941 1250 1354  417
##  [141]  348  943  359 1373 1381   54  766 1464  224 1148   38 1293  659  317
##  [155] 1595  687 1123 1067  824 1098 1151  735  387   90  547 1006 1105  986
##  [169]  100  518 1191  390 1589 1003  826 1558  685  667  192  108  193 1454
##  [183]  961 1453 1068 1133 1499  912  762 1144  823 1425 1315 1320  817  320
##  [197]  712  370  453  310 1211 1566  857  336  380 1241 1440  141 1311   11
##  [211] 1427 1498  716 1084 1475   35 1073 1272 1116 1015  421  684  408  307
##  [225] 1520  791  852   74   26  820  850 1312  173  515  500 1394  630  662
##  [239]  535  959  927  796  337  383  103 1081 1339 1100   32  460  810 1221
##  [253]  565  606  341  968   28  413 1360  558  507 1165  631 1036 1322  180
##  [267]  278   23   29  536  424 1059  954  248  396  264  161 1287 1082 1387
##  [281]  254  756 1172  358 1078  458  844  477  780  372  877  995  538  924
##  [295]  328  435 1086  438 1150 1450 1280 1235  283  938  765  945 1259  321
##  [309]   71   99  568   15  907  479  663 1024  543 1157 1160 1069  922  411
##  [323] 1090 1205  502 1332   67  990 1438  829 1301  102  185 1121  128  105
##  [337] 1565 1023  751  733 1183   69 1353  949   24  861 1298  726  292  539
##  [351]  293  404   53  395  940 1079  195  420   40 1213  322  957 1483    2
##  [365] 1178  879 1285  862 1361  993  371 1118  655 1393  594  306 1594 1257
##  [379]   80  795 1139 1545  569 1508  497 1430  524  713  926  599 1255  178
##  [393]  523 1302  586 1135  566  326 1045  279 1346  653 1487  316 1245  186
##  [407]   86  813 1562  456 1268 1474  632  189  648 1179  323  257 1486  245
##  [421] 1016  618 1395  138  969  723 1247 1573 1058  491  485  635  120  537
##  [435]  349 1142  346  998 1553 1209  285 1203  643  715 1352  260  335 1584
##  [449]  866   49  126 1227 1435   97  449 1480   34  894 1400   68    3 1130
##  [463]  325 1296  992  190  512 1230 1403  140  163  149  770  690  445 1107
```

```
##   [477] 1509  373 1532  564 1412  258 1356 1066  987  785 1033  794  778    6
##   [491] 1225   20  873  683  481 1345  624  123   58  574 1194  302  137  527
##   [505] 1330  719 1514 1367 1472  480 1347  450  609 1369  581  944 1364   98
##   [519]  454 1375  983 1064  533 1115  848  288  275  895  634  833 1005 1587
##   [533]  540  984   84  644  525  989 1094  871  244  808  585  747 1208 1569
##   [547] 1031  962  846 1270  812  867 1261 1220 1349  150 1419  131  775   61
##   [561]  802  617  657  860  929 1362 1274  682  393    4 1129  718  410  980
##   [575]  967 1263  104 1557 1335  658 1001 1174  991 1318  893  273  281  626
##   [589] 1544  654  876  332  800 1132 1465  432   59  830 1456  437 1223  607
##   [603]  707 1546   18  628  211 1404  935 1590  671  739  681 1189  478  375
##   [617]  295  647  164 1041 1398  517  670 1051  722  845 1228 1054  641  872
##   [631]  891 1488 1161  516  550  748  113  129  472 1409  827  639  532  144
##   [645]  567 1442  253  779 1334 1104  700 1452  905   76   56  401  948  381
##   [659]  378  908  978  767   55 1176  151  666  546  714  466  577  496 1193
##   [673]  467  237 1017  633 1458  196  819  837 1038  571  252 1251  259  236
##   [687]  915  973  240  475 1444 1149 1522 1555 1239 1384  637  677 1526 1567
##   [701] 1175 1543  355 1530 1292 1538 1237 1185 1365 1267  296  112 1170 1554
##   [715] 1145  175 1598 1206   19  971   93  202  669  706  476  673  815  702
##   [729] 1571  580  427    9 1559 1583 1278  711  548  482  717  625  906 1408
##   [743]  886  367  452  443   36  134  612  593  168  651 1187  828 1277  483
##   [757]  953   50 1158 1025  299  832  352  782 1040  909 1309  106  772  976
##   [771]   14  996 1493  821  228  605  201  809  441  855 1495 1216 1290 1420
##   [785] 1249  167  499 1266  562  470  522 1037 1577 1252 1204  305 1529 1026
##   [799]  610 1011  206  402  124 1162  874 1556  166   16 1462  199  970  732
##   [813]  741 1576 1460 1531  981 1184 1218  636  725  597 1549  892 1338  263
##   [827] 1541 1447   13  788  447  203  703  474   52 1101  268  660 1463  847
##   [841]  234 1313  446  132  545  514 1177  692 1217  900  362   51  364  590
##   [855]  210  738 1146  549  925 1441  531   78 1491  629 1207  389  484  966
##   [869] 1106  985  488 1119  542 1126  884 1195  856  864 1421 1446 1378  870
##   [883]  578  315  934  675  160 1271  955  181  490  665   10 1168 1186  530
##   [897]  781   91 1593  572  958  312 1070 1128  300  790 1248  724  721   48
##   [911] 1143  965  875  801  135  863  147  271 1022   95  919  461  544 1382
##   [925]  218  374   79 1306  917  448  600  825 1467 1336  388 1152   47 1007
##   [939]  415  814 1197 1550  776  503 1540  270  229  933  792  950  627  640
##   [953] 1578  695  386 1295  351 1426  133  761  734  130   66 1570  889  885
##   [967]  588 1310  266  414  645 1200  854  290  153  743  674  385 1370  469
##   [981]  804 1234  951  122 1008 1340  556  730  708 1428 1337  805 1390  759
##   [995] 1406  220  304 1363 1479  736  693 1085  114  379  462 1159 1163  267
##  [1009] 1283 1300   22 1344  511 1021 1449 1095 1473 1122  284 1399 1439 1489
##  [1023]  274  227 1281  287   25  952  928  745  405  960 1396  493 1415  997
##  [1037]  354 1359 1096  261 1484 1459  946  799 1470  194 1062 1043  276  554
##  [1051]  789  528  686  834  903   92  701 1581 1591  598  904  787  208  226
##  [1065] 1262 1260 1358 1155 1582  188  918  868 1265  742 1515  931   94 1355
##  [1079] 1551  818 1060 1125  881 1501  691  584 1233  265  440 1478  798 1342
##  [1093]  744 1506 1156  757  182  899 1552  282  157 1083  505  162  592 1507
##  [1107] 1376  174  125 1034  975 1020  939  212  890 1536  465 1099 1167
```

```
  wine.df.train <- wine.raw[train.cases,]
  wine.df.test <- wine.raw[-train.cases,]

#---Checking the detail of sampling
  cat("Training data dimensions:", dim(wine.df.train), "\n")
```

```
## Training data dimensions: 1119 13
```

```r
  print(table(wine.df.train$quality.label))
```

```
##
##  Bad Good
##  956  163
```

```r
  #>> The training data set contains 956 wines labeled as "Bad" and
  #>> 163 wines labeled as "Good": model will learn patterns from these sampling
  cat("Testing data dimensions :", dim(wine.df.test), "\n")
```

```
## Testing data dimensions : 480 13
```

```r
  print(table(wine.df.test$quality.label))
```

```
##
##  Bad Good
##  426   54
```

```r
  #>> The testing data set contains 426 wines labeled as "Bad" and
  #>> 54 wines labeled as "Good": not seen these sampling


###-------------------------------------------------------------------------###
##----------------------------Random Forest--------------------------##

#---Ensure the target variable is a factor
  wine.df.train$quality.label <-as.factor(wine.df.train$quality.label)
  wine.df.test$quality.label <- as.factor(wine.df.test$quality.label)

#4.Building a Random Forest Model:
  library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.3.3
```

```
## randomForest 4.7-1.2
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```r
  set.seed(123)
  wine.rf <- randomForest(quality.label ~.-quality, #all features except quality
                          data = wine.df.train, ntree = 3000)
  wine.rf
```

```
##
## Call:
##  randomForest(formula = quality.label ~ . - quality, data = wine.df.train,      ntree = 3000)
##                Type of random forest: classification
##                      Number of trees: 3000
## No. of variables tried at each split: 3
##
```

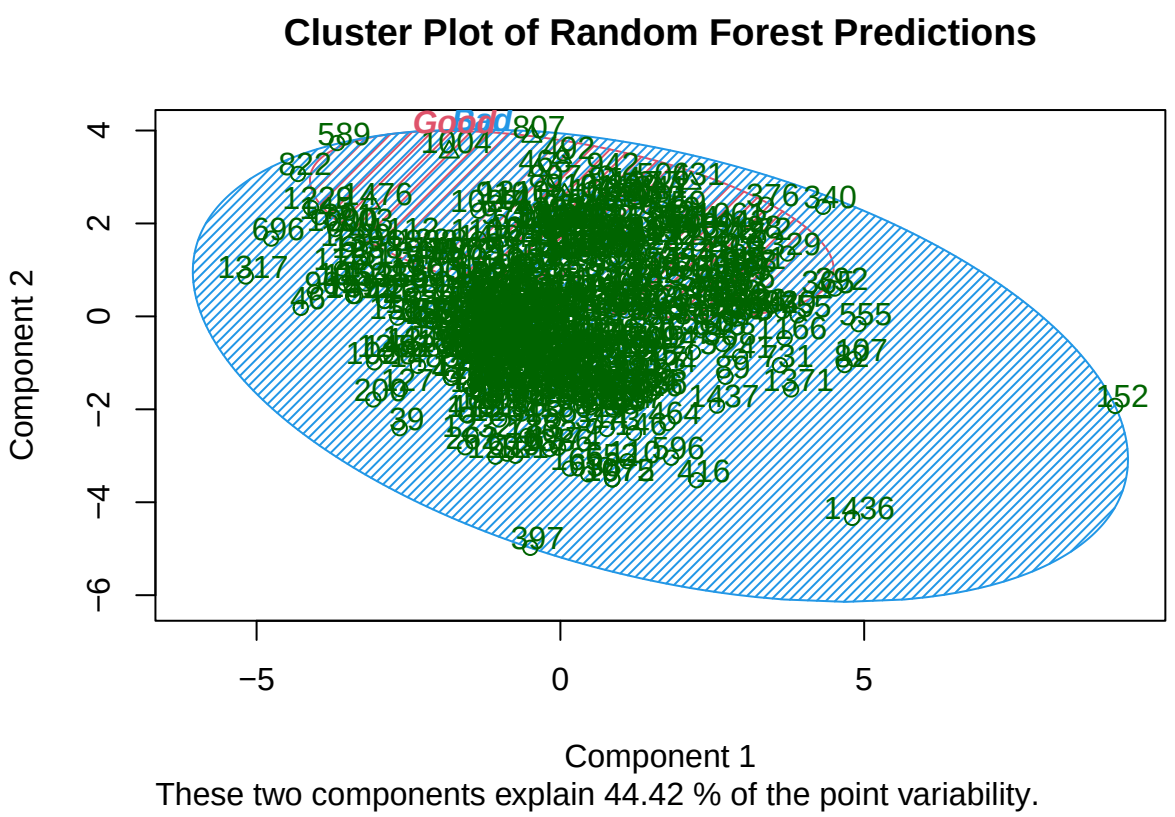
```
##         OOB estimate of  error rate: 8.94%
## Confusion matrix:
##      Bad Good class.error
## Bad  931   25  0.02615063
## Good  75   88  0.46012270
```

```
#5.Making Predictions for the test data:
  wine.rf.predict <- predict(wine.rf,wine.df.test)
  wine.rf.predict
```

```
##    5    7    8   12   17   21   30   33   37   39   41   42   44   46   62   63
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
##   64   65   72   73   75   77   81   82   83   88   89   96  101  107  109  110
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
##  111  116  117  118  119  121  127  136  139  142  145  146  152  156  158  165
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
##  169  171  172  176  177  179  184  187  191  197  198  200  204  205  209  213
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
##  214  215  216  217  219  221  222  231  235  238  241  242  243  247  251  262
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
##  269  272  277  280  289  294  297  298  301  303  308  311  313  314  318  319
##  Bad  Bad  Bad Good Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good
##  324  329  330  331  333  334  339  340  342  343  344  345  347  353  356  357
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
##  363  365  366  368  376  377  384  391  392  394  397  398  403  406  407  409
##  Bad Good Good  Bad  Bad  Bad  Bad Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good
##  412  416  418  419  422  423  425  426  428  430  431  434  436  439  442  444
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good  Bad  Bad  Bad  Bad  Bad
##  451  455  457  463  464  468  487  489  492  494  495  498  501  506  508  509
##  Bad  Bad  Bad  Bad  Bad Good  Bad  Bad Good  Bad  Bad  Bad  Bad Good  Bad  Bad
##  513  519  520  526  551  552  553  555  559  560  561  573  575  576  583  589
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
##  595  596  602  603  604  611  614  615  616  620  621  623  638  642  649  650
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
##  652  656  661  664  668  672  676  679  680  688  694  696  697  698  699  704
##  Bad  Bad  Bad Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
##  720  727  728  731  740  746  749  752  753  754  755  758  760  763  764  768
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
##  769  771  773  774  777  783  784  786  793  797  803  807  811  816  822  835
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good Good  Bad  Bad  Bad  Bad
##  836  838  839  840  841  843  849  851  853  858  859  865  882  883  887  888
##  Bad Good Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good  Bad  Bad
##  896  897  898  902  910  911  913  914  916  920  921  923  932  936  937  942
##  Bad Good  Bad Good  Bad  Bad  Bad Good Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad
##  956  963  964  972  974  979  988  999 1000 1002 1004 1009 1010 1012 1013 1014
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good Good  Bad  Bad  Bad  Bad
## 1018 1019 1027 1028 1029 1030 1035 1039 1044 1046 1047 1048 1049 1050 1053 1055
##  Bad  Bad  Bad  Bad  Bad Good  Bad Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
## 1056 1061 1063 1065 1071 1072 1074 1075 1076 1077 1091 1092 1097 1102 1103 1109
##  Bad  Bad  Bad  Bad Good  Bad  Bad  Bad  Bad Good  Bad  Bad  Bad  Bad  Bad  Bad
## 1110 1111 1112 1113 1117 1120 1124 1131 1134 1137 1141 1147 1154 1164 1166 1171
##  Bad  Bad  Bad Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good
## 1173 1180 1181 1182 1190 1196 1198 1199 1201 1210 1214 1215 1219 1222 1229 1231
## Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good
```

```
## 1232 1236 1238 1240 1242 1243 1246 1253 1254 1256 1258 1264 1275 1276 1279 1282
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
## 1284 1286 1288 1289 1294 1297 1299 1303 1304 1307 1314 1316 1317 1319 1321 1323
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good
## 1324 1325 1326 1327 1328 1329 1331 1341 1343 1348 1350 1351 1357 1366 1368 1371
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
## 1372 1374 1377 1379 1380 1385 1386 1388 1389 1391 1392 1402 1405 1407 1410 1411
## Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good  Bad  Bad
## 1413 1414 1416 1417 1418 1422 1423 1424 1429 1431 1434 1436 1437 1443 1445 1457
## Good  Bad  Bad  Bad Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
## 1461 1466 1471 1476 1481 1482 1485 1490 1492 1496 1500 1502 1504 1505 1510 1511
##  Bad  Bad  Bad Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good  Bad
## 1512 1513 1516 1519 1521 1523 1524 1525 1527 1528 1533 1534 1537 1539 1542 1547
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
## 1548 1561 1563 1564 1568 1572 1574 1575 1580 1585 1586 1588 1592 1596 1597 1599
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
## Levels: Bad Good
```

```
#6.Visualizing Predictions:
  library(cluster)
  clusplot(wine.df.test[-13], wine.rf.predict, color = TRUE, shade = TRUE,
           labels = 2, lines = 0,
           main ="Cluster Plot of Random Forest Predictions")
```



```r
#7.Handing Imbalanced Classes

#---Building a Random Forest Model: with Balanced Sampling
  #the model is now forced to focus equally on both classes.

  library(randomForest)
  set.seed(123)
  wine.rf <- randomForest(quality.label ~.-quality, #all features except quality
                          data = wine.df.train, ntree = 3000,
                          sampsize = c(163,163)) #Match the smaller class size
  wine.rf
```

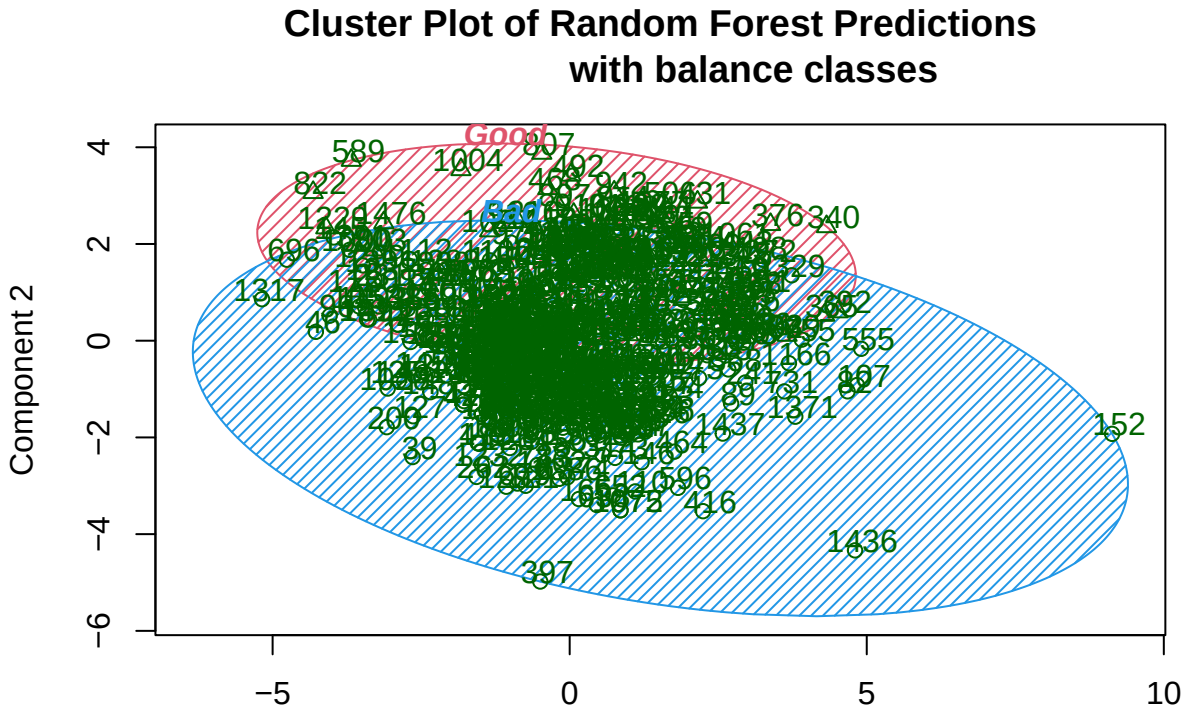
```
##
## Call:
##  randomForest(formula = quality.label ~ . - quality, data = wine.df.train,      ntree = 3000, sampsi
##                Type of random forest: classification
##                      Number of trees: 3000
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 14.21%
## Confusion matrix:
##      Bad Good class.error
## Bad  830  126   0.1317992
## Good  33  130   0.2024540
```

```r
#---Making Predictions for the test data:
  wine.rf.predict <- predict(wine.rf,wine.df.test)
  wine.rf.predict
```

```
##    5    7    8   12   17   21   30   33   37   39   41   42   44   46   62   63
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
##   64   65   72   73   75   77   81   82   83   88   89   96  101  107  109  110
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
##  111  116  117  118  119  121  127  136  139  142  145  146  152  156  158  165
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
##  169  171  172  176  177  179  184  187  191  197  198  200  204  205  209  213
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good  Bad  Bad  Bad  Bad Good
##  214  215  216  217  219  221  222  231  235  238  241  242  243  247  251  262
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good  Bad  Bad Good  Bad
##  269  272  277  280  289  294  297  298  301  303  308  311  313  314  318  319
##  Bad  Bad  Bad Good Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good
##  324  329  330  331  333  334  339  340  342  343  344  345  347  353  356  357
##  Bad  Bad  Bad Good  Bad  Bad Good Good Good  Bad  Bad  Bad Good  Bad  Bad Good
##  363  365  366  368  376  377  384  391  392  394  397  398  403  406  407  409
##  Bad Good Good  Bad Good  Bad  Bad Good Good  Bad  Bad Good  Bad Good Good Good
##  412  416  418  419  422  423  425  426  428  430  431  434  436  439  442  444
##  Bad  Bad  Bad Good  Bad  Bad  Bad  Bad  Bad  Bad Good  Bad  Bad  Bad Good Good
##  451  455  457  463  464  468  487  489  492  494  495  498  501  506  508  509
##  Bad  Bad  Bad Good  Bad Good  Bad Good Good  Bad  Bad  Bad  Bad Good  Bad  Bad
##  513  519  520  526  551  552  553  555  559  560  561  573  575  576  583  589
##  Bad Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good  Bad  Bad  Bad Good  Bad Good
##  595  596  602  603  604  611  614  615  616  620  621  623  638  642  649  650
##  Bad  Bad  Bad  Bad  Bad  Bad Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
```

```
##  652  656  661  664  668  672  676  679  680  688  694  696  697  698  699  704
##  Bad  Bad  Bad Good  Bad  Bad Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
##  720  727  728  731  740  746  749  752  753  754  755  758  760  763  764  768
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
##  769  771  773  774  777  783  784  786  793  797  803  807  811  816  822  835
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good Good  Bad  Bad Good  Bad
##  836  838  839  840  841  843  849  851  853  858  859  865  882  883  887  888
##  Bad Good Good  Bad Good  Bad  Bad  Bad  Bad Good Good  Bad  Bad Good  Bad Good
##  896  897  898  902  910  911  913  914  916  920  921  923  932  936  937  942
##  Bad Good  Bad Good Good Good Good Good Good Good  Bad Good  Bad Good Good Good
##  956  963  964  972  974  979  988  999 1000 1002 1004 1009 1010 1012 1013 1014
## Good  Bad Good Good  Bad  Bad  Bad  Bad  Bad Good Good Good  Bad  Bad  Bad  Bad
## 1018 1019 1027 1028 1029 1030 1035 1039 1044 1046 1047 1048 1049 1050 1053 1055
## Good Good Good  Bad  Bad Good  Bad Good Good  Bad  Bad  Bad Good Good  Bad  Bad
## 1056 1061 1063 1065 1071 1072 1074 1075 1076 1077 1091 1092 1097 1102 1103 1109
##  Bad Good Good  Bad Good  Bad  Bad  Bad  Bad Good Good Good  Bad  Bad  Bad  Bad
## 1110 1111 1112 1113 1117 1120 1124 1131 1134 1137 1141 1147 1154 1164 1166 1171
##  Bad  Bad  Bad Good  Bad Good Good  Bad Good Good  Bad  Bad Good  Bad  Bad Good
## 1173 1180 1181 1182 1190 1196 1198 1199 1201 1210 1214 1215 1219 1222 1229 1231
## Good Good Good Good  Bad  Bad  Bad  Bad  Bad Good Good  Bad Good Good Good Good
## 1232 1236 1238 1240 1242 1243 1246 1253 1254 1256 1258 1264 1275 1276 1279 1282
##  Bad  Bad  Bad  Bad  Bad Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
## 1284 1286 1288 1289 1294 1297 1299 1303 1304 1307 1314 1316 1317 1319 1321 1323
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good
## 1324 1325 1326 1327 1328 1329 1331 1341 1343 1348 1350 1351 1357 1366 1368 1371
## Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
## 1372 1374 1377 1379 1380 1385 1386 1388 1389 1391 1392 1402 1405 1407 1410 1411
## Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good  Bad  Bad
## 1413 1414 1416 1417 1418 1422 1423 1424 1429 1431 1434 1436 1437 1443 1445 1457
## Good  Bad  Bad  Bad Good  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad
## 1461 1466 1471 1476 1481 1482 1485 1490 1492 1496 1500 1502 1504 1505 1510 1511
##  Bad  Bad  Bad Good  Bad Good  Bad Good  Bad  Bad  Bad  Bad  Bad  Bad Good  Bad
## 1512 1513 1516 1519 1521 1523 1524 1525 1527 1528 1533 1534 1537 1539 1542 1547
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good  Bad
## 1548 1561 1563 1564 1568 1572 1574 1575 1580 1585 1586 1588 1592 1596 1597 1599
##  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad  Bad Good Good  Bad  Bad Good  Bad Good
## Levels: Bad Good
```

```
#---Visualizing Predictions:
  library(cluster)
  clusplot(wine.df.test[-13], wine.rf.predict, color = TRUE, shade = TRUE,
           labels = 2, lines = 0,
           main ="Cluster Plot of Random Forest Predictions
                  with balance classes")
```




```
#8.Evaluating the model:How well did we predict the test data?

#---Accuracy Method: which is the proportion pf correct predictions.
  mean(wine.df.test$quality.label == wine.rf.predict)
```

```
## [1] 0.8291667
```

```
#---Adjust Rand Index (ARI)
  library(mclust)
```

```
## Warning: package 'mclust' was built under R version 4.3.2
```

```
## Package 'mclust' version 6.1.1
## Type 'citation("mclust")' for citing this R package in publications.
```

```
  adjustedRandIndex(wine.df.test$quality.label, wine.rf.predict)
```

```
## [1] 0.3205036
```

```
#---Confusion Matrix
  table(wine.rf.predict, wine.df.test$quality.label)
```

```
##
## wine.rf.predict Bad Good
##            Bad  355   11
##            Good  71   43
```

```
#---Summary Functions:
  wine.summ <- function(data, groups)
  {
    aggregate(data, list(groups), function(x) mean(as.numeric(x)))
  }
  wine.summ(wine.df.test, wine.rf.predict) #proposed quality
```

```
##   Group.1 fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
## 1     Bad      8.009836        0.5739754   0.2312568       2.518852 0.09119126
## 2    Good      9.085965        0.3892105   0.3965789       2.544737 0.07586842
##   free.sulfur.dioxide total.sulfur.dioxide   density       pH sulphates
## 1            16.92213             52.24454 0.9968152 3.328743 0.6346175
## 2            13.64035             35.09649 0.9961684 3.275263 0.7420175
##    alcohol  quality quality.label
## 1 10.15574 5.434426      1.030055
## 2 11.47515 6.315789      1.377193
```

```
  wine.summ(wine.df.test, wine.df.test$quality.label) #actual quality
```

```
##   Group.1 fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
## 1     Bad      8.261737        0.5419953   0.2649765       2.498944 0.08976761
## 2    Good      8.294444        0.4362037   0.3142593       2.730556 0.07007407
##   free.sulfur.dioxide total.sulfur.dioxide  density       pH sulphates  alcohol
## 1            15.97770             48.73826 0.996787 3.312230 0.6499296 10.32113
## 2            17.44444             43.70370 0.995672 3.346111 0.7405556 11.63642
##    quality quality.label
## 1 5.464789             1
## 2 7.055556             2
```

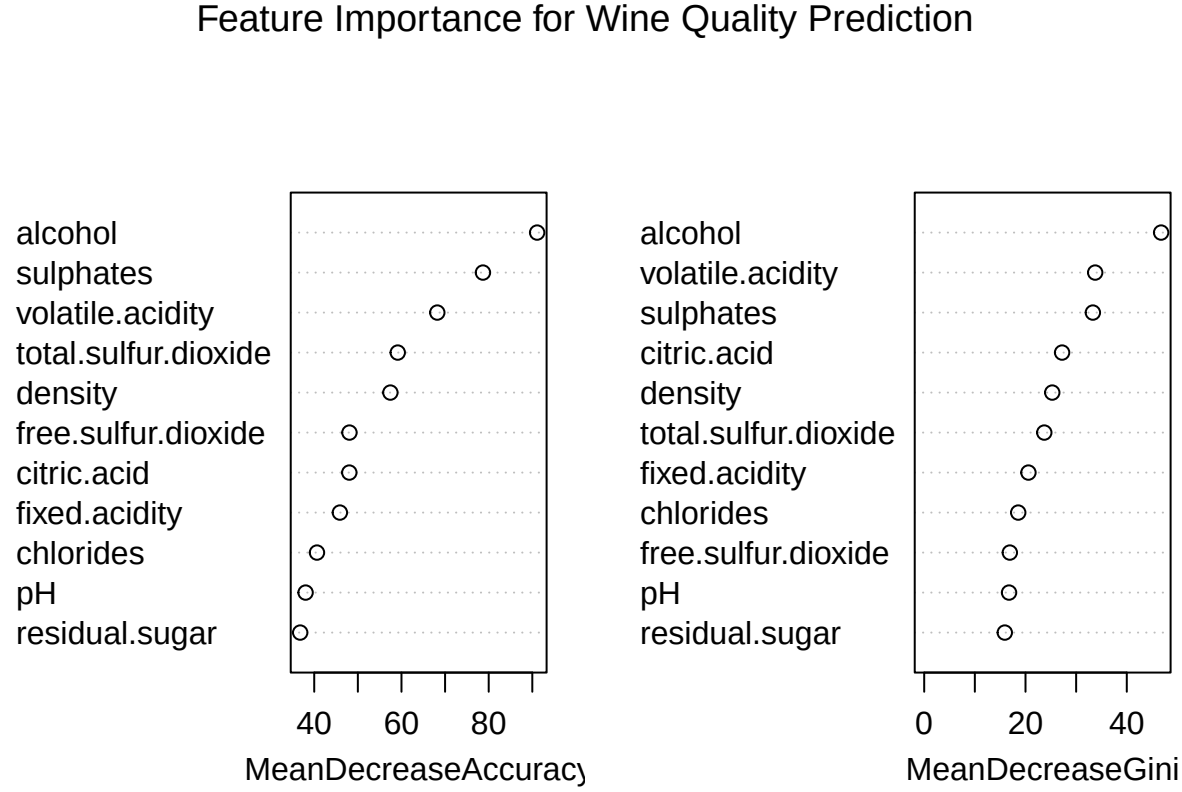
```
#7.Variable importance: Random Forest can tell us which variable (features) are
  #the most important for making predictions

wine.rf <- randomForest(quality.label ~.-quality, #all features except quality
                                  data = wine.df.train, ntree = 3000,
                        importance = TRUE)
wine.rf
```

```
##
## Call:
##  randomForest(formula = quality.label ~ . - quality, data = wine.df.train,      ntree = 3000, importa
##                Type of random forest: classification
##                      Number of trees: 3000
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 9.12%
## Confusion matrix:
##      Bad Good class.error
## Bad  927   29  0.03033473
## Good  73   90  0.44785276
```

```r
importance(wine.rf)
```

```
##                         Bad     Good MeanDecreaseAccuracy MeanDecreaseGini
## fixed.acidity        23.07886 43.93875              45.88104         20.57973
## volatile.acidity     21.12378 83.23243              68.21415         33.76012
## citric.acid          13.31114 49.27367              48.04063         27.22622
## residual.sugar       20.44845 34.10328              36.78252         15.90856
## chlorides            27.10404 33.37016              40.62654         18.56683
## free.sulfur.dioxide  33.30032 38.04191              48.07403         16.90554
## total.sulfur.dioxide 32.11146 65.17008              59.14898         23.70860
## density              38.37314 46.22053              57.45747         25.28253
## pH                   16.59061 42.56024              38.01753         16.75284
## sulphates            24.41116 93.79164              78.67998         33.30075
## alcohol              42.02788 99.17418              91.10301         46.77968
```

```r
#---Variable importance plot
varImpPlot(wine.rf, main = "Feature Importance for Wine Quality Prediction")
```




```r
#---Heatmap for variable important
library(gplots)
```

```
## Warning: package 'gplots' was built under R version 4.3.3
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
library(RColorBrewer)
heatmap.2(t(importance(wine.rf)[,1:2]), key = FALSE, col = brewer.pal(9, "Blues"),
          dend = "none", trace = "none", margins = c(10,10),
          main = "\n\n\nHeatmap of Feature Importance \n for Wine Quality Prediction",
          cexRow = 1.5)
```